Stellaris® LM3S9U81 Microcontroller

DATA SHEET

DS-LM3S9U81-11425

Copyright

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

Texas Instruments Incorporated
108 Wild Basin, Suite 350
Austin, TX 78746
http://www.ti.com/stellaris
http://www-k.ext.ti.com/sc/technical-support/product-information-centers.htm

Table of Contents

List of Figures

List of Tables

List of Registers

Revision History

The revision history table notes changes made between the indicated revisions of the LM3S9U81 data sheet.

Table 1. Revision History

Date	Revision	Description
January 2012	11425	■ In System Control chapter: – Clarified that an external LDO cannot be used. – Clarified system clock requirements when the ADC module is in operation. – Added important note to write the **RCC** register before the **RCC2** register. ■ In Internal Memory chapter, clarified programming and use of the non-volatile registers. ■ In GPIO chapter, corrected "GPIO Pins With Non-Zero Reset Values" table and added note that if the same signal is assigned to two different GPIO port pins, the signal is assigned to the port with the lowest letter. ■ In EPI chapter: – Clarified table "Capabilities of Host Bus 8 and Host Bus 16 Modes". – Corrected bit and register resets for FREQ (Frequency Range) in **EPI SDRAM Configuration (EPISDRAMCFG)** register. – Corrected bit and register resets for MAXWAIT (Maximum Wait) in **EPI Host-Bus 8 Configuration (EPIHB8CFG)** and **EPI Host-Bus 16 Configuration (EPIHB16CFG)** registers. Also clarified bit descriptions in these registers. – Corrected bit definitions for the EPSZ and ERSZ bits in the **EPI Address Map (EPIADDRMAP)** register. – Corrected size of **COUNT** bit field in **EPI Read FIFO Count (EPIRFIFOCNT)** register. ■ In Timer chapter, clarified timer modes and interrupts. ■ In ADC chapter, added "ADC Input Equivalency Diagram". ■ In UART chapter, clarified interrupt behavior. ■ In SSI chapter, corrected SSIClk in the figure "Synchronous Serial Frame Format (Single Transfer)" and clarified behavior of transmit bits in interrupt registers. ■ In I^2C chapter, corrected bit and register reset values for IDLE bit in **I^2C Master Control/Status (I2CMCS)** register. ■ In USB chapter: – Clarified that when the USB module is in operation, MOSC must be provided with a clock source, and the system clock must be at least 30 MHz. – Removed MULTTRAN bit from **USB Transmit Hub Address Endpoint n (USBTXHUBADDRn)** and **USB Receive Hub Address Endpoint n (USBRXHUBADDRn)** registers. – Corrected description for the **USB Device RESUME Interrupt Mask (USBDRIM)** register. ■ In Analog Comparators chapter, clarified internal reference programming. ■ In Signal Tables chapter, clarified VDDC and LDO pin descriptions. ■ In Electrical Characteristics chapter:

Table 1. Revision History *(continued)*

Date	Revision	Description
		– In Maximum Ratings table, deleted parameter "Input voltage for a GPIO configured as an analog input". – In Recommended DC Operating Conditions table, corrected values for I_{OH} parameter. – In JTAG Characteristics, table, corrected values for parameters "TCK clock Low time" and "TCK clock High time". – In LDO Regulator Characteristics table, added clarifying footnote to C_{LDO} parameter. – In System Clock Characteristics with ADC Operation table, added clarifying footnote to F_{sysadc} parameter. – Added "System Clock Characteristics with USB Operation" table. – In Sleep Modes AC Characteristics table, split parameter "Time to wake from interrupt" into sleep mode and deep-sleep mode parameters. – In SSI Characteristics table, corrected value for parameter "SSIClk cycle time". – In Analog Comparator Characteristics table, added parameter "Input voltage range" and corrected values for parameter "Input common mode voltage range". – In Analog Comparator Voltage Reference Characteristics table, corrected values for absolute accuracy parameters. – Deleted table "USB Controller DC Characteristics". – In Nominal Power Consumption table, added parameter for sleep mode. – In Maximum Current Consumption section, changed reference value for MOSC and temperature in tables that follow. – Deleted table "External VDDC Source Current Specifications". ■ Additional minor data sheet clarifications and corrections.
July 2011	9970	■ Corrected "Reset Sources" table. ■ Added Important Note that **RCC** register must be written before **RCC2** register. ■ Added a note that all GPIO signals are 5-V tolerant when configured as inputs except for PB0 and PB1, which are limited to 3.6 V. ■ Corrected LIN Mode bit names in **UART Interrupt Clear (UARTICR)** register. ■ Corrected pin number for $\overline{RST}$ and added missing pin number for ERBIAS in table "Connections for Unused Signals" (other pin tables were correct). ■ In the "Operating Characteristics" chapter: – In the "Thermal Characteristics" table, the Thermal resistance value was changed. – In the "ESD Absolute Maximum Ratings" table, the V_{ESDCDM} parameter was changed and the V_{ESDMM} parameter was deleted. ■ The "Electrical Characteristics" chapter was reorganized by module. In addition, some of the Recommended DC Operating Conditions, LDO Regulator, Clock, GPIO, EPI, ADC, and SSI characteristics were finalized. ■ Additional minor data sheet clarifications and corrections.
March 2011	9538	Started tracking revision history.

About This Document

This data sheet provides reference information for the LM3S9U81 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 related documents are available on the Stellaris® web site at www.ti.com/stellaris:

- *Stellaris® Errata*
- *ARM® Cortex™-M3 Errata*
- *Cortex™-M3/M4 Instruction Set Technical User's Manual*
- *Stellaris® Boot Loader User's Guide*
- *Stellaris® Graphics Library User's Guide*
- *Stellaris® Peripheral Driver Library User's Guide*
- *Stellaris® ROM User's Guide*
- *Stellaris® USB Library User's Guide*

The following related documents are also referenced:

- *ARM® Debug Interface V5 Architecture Specification*
- *ARM® Embedded Trace Macrocell Architecture Specification*
- *IEEE Standard 1149.1-Test Access Port and Boundary-Scan Architecture*

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

Documentation Conventions

This document uses the conventions shown in Table 2 on page 42.

Table 2. Documentation Conventions

Notation	Meaning
General Register Notation	
REGISTER	APB registers are indicated in uppercase bold. For example, **PBORCTL** is the Power-On and Brown-Out Reset Control register. If a register name contains a lowercase n, it represents more than one register. For example, **SRCRn** represents any (or all) of the three Software Reset Control registers: **SRCR0, SRCR1** , and **SRCR2**.
bit	A single bit in a register.
bit field	Two or more consecutive and related bits.
offset 0x*nnn*	A hexadecimal increment to a register's address, relative to that module's base address as specified in Table 2-4 on page 84.
Register *N*	Registers are numbered consecutively throughout the document to aid in referencing them. The register number has no meaning to software.
reserved	Register bits marked *reserved* are reserved for future use. In most cases, reserved bits are set to 0; however, user software should not rely on the value of a reserved bit. To provide software compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
yy:xx	The range of register bits inclusive from xx to yy. For example, 31:15 means bits 15 through 31 in that register.
Register Bit/Field Types	This value in the register bit diagram indicates whether software running on the controller can change the value of the bit field.
RC	Software can read this field. The bit or field is cleared by hardware after reading the bit/field.
RO	Software can read this field. Always write the chip reset value.
R/W	Software can read or write this field.
R/WC	Software can read or write this field. Writing to it with any value clears the register.
R/W1C	Software can read or write this field. A write of a 0 to a W1C bit does not affect the bit value in the register. A write of a 1 clears the value of the bit in the register; the remaining bits remain unchanged. This register type is primarily used for clearing interrupt status bits where the read operation provides the interrupt status and the write of the read value clears only the interrupts being reported at the time the register was read.
R/W1S	Software can read or write a 1 to this field. A write of a 0 to a R/W1S bit does not affect the bit value in the register.
W1C	Software can write this field. A write of a 0 to a W1C bit does not affect the bit value in the register. A write of a 1 clears the value of the bit in the register; the remaining bits remain unchanged. A read of the register returns no meaningful data. This register is typically used to clear the corresponding bit in an interrupt register.
WO	Only a write by software is valid; a read of the register returns no meaningful data.
Register Bit/Field Reset Value	This value in the register bit diagram shows the bit/field value after any reset, unless noted.
0	Bit cleared to 0 on chip reset.
1	Bit set to 1 on chip reset.
-	Nondeterministic.
Pin/Signal Notation	
[ ]	Pin alternate function; a pin defaults to the signal without the brackets.
pin	Refers to the physical connection on the package.
signal	Refers to the electrical signal encoding of a pin.

Table 2. Documentation Conventions *(continued)*

Notation	Meaning
assert a signal	Change the value of the signal from the logically False state to the logically True state. For active High signals, the asserted signal value is 1 (High); for active Low signals, the asserted signal value is 0 (Low). The active polarity (High or Low) is defined by the signal name (see SIGNAL and $\overline{\text{SIGNAL}}$ below).
deassert a signal	Change the value of the signal from the logically True state to the logically False state.
$\overline{\text{SIGNAL}}$	Signal names are in uppercase and in the Courier font. An overbar on a signal name indicates that it is active Low. To assert $\overline{\text{SIGNAL}}$ is to drive it Low; to deassert $\overline{\text{SIGNAL}}$ is to drive it High.
SIGNAL	Signal names are in uppercase and in the Courier font. An active High signal has no overbar. To assert SIGNAL is to drive it High; to deassert SIGNAL is to drive it Low.
Numbers	
X	An uppercase X indicates any of several values is allowed, where X can be any legal pattern. For example, a binary value of 0X00 can be either 0100 or 0000, a hex value of 0xX is 0x0 or 0x1, and so on.
0x	Hexadecimal numbers have a prefix of 0x. For example, 0x00FF is the hexadecimal number FF. All other numbers within register tables are assumed to be binary. Within conceptual information, binary numbers are indicated with a b suffix, for example, 1011b, and decimal numbers are written without a prefix or suffix.

1 Architectural Overview

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

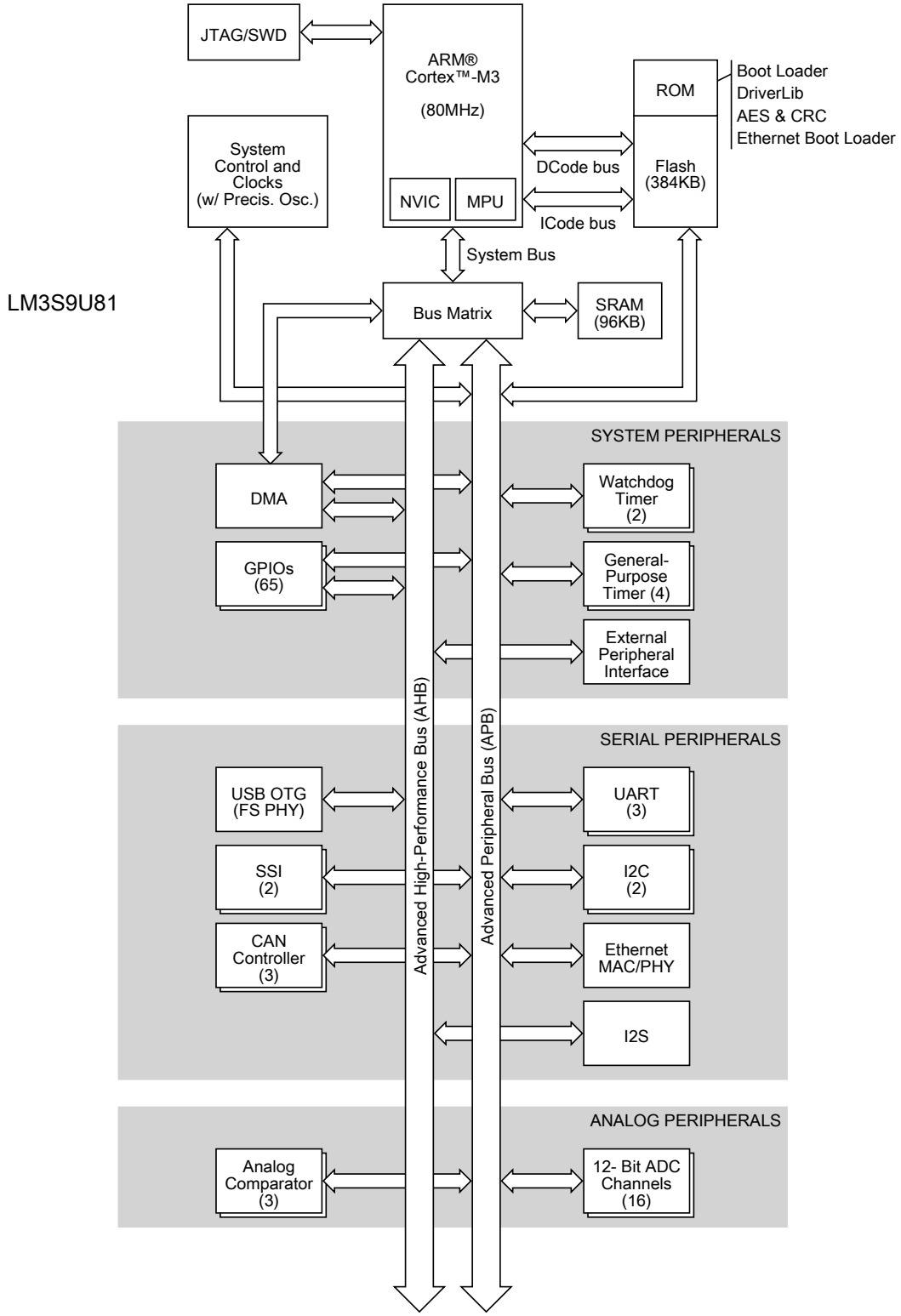
1.1 Overview

The Stellaris LM3S9U81 microcontroller combines complex integration and high performance with the following feature highlights:

- ARM Cortex-M3 Processor Core
- High Performance: 80-MHz operation; 100 DMIPS performance
- 384 KB single-cycle Flash memory
- 96 KB single-cycle SRAM
- Internal ROM loaded with StellarisWare® software
- External Peripheral Interface (EPI)
- Advanced Communication Interfaces: UART, SSI, I2C, I2S, CAN, Ethernet MAC and PHY, USB
- System Integration: general-purpose timers, watchdog timers, DMA, general-purpose I/Os
- Analog support: analog and digital comparators, Analog-to-Digital Converters (ADC), on-chip voltage regulator
- JTAG and ARM Serial Wire Debug (SWD)
- 100-pin LQFP package
- 108-ball BGA package
- Industrial (-40°C to 85°C) temperature range

Figure 1-1 on page 45 depicts the features on the Stellaris LM3S9U81 microcontroller. Note that there are two on-chip buses that connect the core to the peripherals. The Advanced Peripheral Bus (APB) bus is the legacy bus. The Advanced High-Performance Bus (AHB) bus provides better back-to-back access performance than the APB bus.

Figure 1-1. Stellaris LM3S9U81 Microcontroller High-Level Block Diagram



In addition, the LM3S9U81 microcontroller offers the advantages of ARM's widely available development tools, System-on-Chip (SoC) infrastructure IP applications, and a large user community. Additionally, the microcontroller uses ARM's Thumb®-compatible Thumb-2 instruction set to reduce memory requirements and, thereby, cost. Finally, the LM3S9U81 microcontroller is code-compatible to all members of the extensive Stellaris family; providing flexibility to fit precise needs.

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

1.2 Target Applications

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

- Gaming equipment
- Network appliances and switches
- Home and commercial site monitoring and control
- Electronic point-of-sale (POS) machines
- Motion control
- Medical instrumentation
- Remote connectivity and monitoring
- Test and measurement equipment
- Factory automation
- Fire and security
- Lighting control
- Transportation

1.3 Features

The LM3S9U81 microcontroller component features and general function are discussed in more detail in the following section.

1.3.1 ARM Cortex-M3 Processor Core

All members of the Stellaris product family, including the LM3S9U81 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.

1.3.1.1 Processor Core (see page 65)

- 32-bit ARM Cortex-M3 architecture optimized for small-footprint embedded applications
- 80-MHz operation; 100 DMIPS performance
- Outstanding processing performance combined with fast interrupt handling
- Thumb-2 mixed 16-/32-bit instruction set delivers the high performance expected of a 32-bit ARM core in a compact memory size usually associated with 8- and 16-bit devices, typically in the range of a few kilobytes of memory for microcontroller-class applications
  - Single-cycle multiply instruction and hardware divide

  - Atomic bit manipulation (bit-banding), delivering maximum memory utilization and streamlined peripheral control
  - Unaligned data access, enabling data to be efficiently packed into memory
- Fast code execution permits slower processor clock or increases sleep mode time
- Harvard architecture characterized by separate buses for instruction and data
- Efficient processor core, system and memories
- Hardware division and fast digital-signal-processing orientated multiply accumulate
- Saturating arithmetic for signal processing
- Deterministic, high-performance interrupt handling for time-critical applications
- Memory protection unit (MPU) to provide a privileged mode for protected operating system functionality
- Enhanced system debug with extensive breakpoint and trace capabilities
- Serial Wire Debug and Serial Wire Trace reduce the number of pins required for debugging and tracing
- Migration from the ARM7 processor family for better performance and power efficiency
- Optimized for single-cycle Flash memory usage
- Ultra-low power consumption with integrated sleep modes

1.3.1.2 System Timer (SysTick) (see page 108)

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

- An RTOS tick timer that fires at a programmable rate (for example, 100 Hz) and invokes a SysTick routine
- A high-speed alarm timer using the system clock
- A variable rate alarm or signal timer—the duration is range-dependent on the reference clock used and the dynamic range of the counter
- A simple counter used to measure time to completion and time used
- An internal clock-source control based on missing/meeting durations.

1.3.1.3 Nested Vectored Interrupt Controller (NVIC) (see page 109)

The LM3S9U81 controller includes the ARM Nested Vectored Interrupt Controller (NVIC). The NVIC and Cortex-M3 prioritize and handle all exceptions in Handler Mode. The processor state is automatically stored to the stack on an exception and automatically restored from the stack at the end of the Interrupt Service Routine (ISR). The interrupt vector is fetched in parallel to the state saving, enabling efficient interrupt entry. The processor supports tail-chaining, meaning that

back-to-back interrupts can be performed without the overhead of state saving and restoration. Software can set eight priority levels on 7 exceptions (system handlers) and 47 interrupts.

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

1.3.1.4 System Control Block (SCB) (see page 111)

The SCB provides system implementation information and system control, including configuration, control, and reporting of system exceptions.

1.3.1.5 Memory Protection Unit (MPU) (see page 111)

The MPU supports the standard ARM7 Protected Memory System Architecture (PMSA) model. The MPU provides full support for protection regions, overlapping protection regions, access permissions, and exporting memory attributes to the system.

1.3.2 On-Chip Memory

The LM3S9U81 microcontroller is integrated with the following set of on-chip memory and features:

- 96 KB single-cycle SRAM
- 384 KB single-cycle Flash memory up to 50 MHz; a prefetch buffer improves performance above 50 MHz
- Internal ROM loaded with StellarisWare software:
  - Stellaris Peripheral Driver Library
  - Stellaris Boot Loader
  - Advanced Encryption Standard (AES) cryptography tables
  - Cyclic Redundancy Check (CRC) error detection functionality

1.3.2.1 SRAM (see page 289)

The LM3S9U81 microcontroller provides 96 KB of single-cycle on-chip SRAM. The internal SRAM of the Stellaris devices is located at offset 0x2000.0000 of the device memory map.

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

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

1.3.2.2 Flash Memory (see page 291)

The LM3S9U81 microcontroller provides 384 KB of single-cycle on-chip Flash memory (above 50 MHz, the Flash memory can be accessed in a single cycle as long as the code is linear; branches incur a one-cycle stall). The Flash memory 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.

1.3.2.3 ROM (see page 289)

The LM3S9U81 ROM is preprogrammed with the following software and programs:

- Stellaris Peripheral Driver Library
- Stellaris Boot Loader
- Advanced Encryption Standard (AES) cryptography tables
- Cyclic Redundancy Check (CRC) error-detection functionality

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

The Advanced Encryption Standard (AES) is a publicly defined encryption standard used by the U.S. Government. AES is a strong encryption method with reasonable performance and size. In addition, it is fast in both hardware and software, is fairly easy to implement, and requires little memory. The Texas Instruments encryption package is available with full source code, and is based on lesser general public license (LGPL) source. An LGPL means that the code can be used within an application without any copyleft implications for the application (the code does not automatically become open source). Modifications to the package source, however, must be open source.

CRC (Cyclic Redundancy Check) is a technique to validate a span of data has the same contents as when previously checked. This technique can be used to validate correct receipt of messages (nothing lost or modified in transit), to validate data after decompression, to validate that Flash memory contents have not been changed, and for other cases where the data needs to be validated. A CRC is preferred over a simple checksum (e.g. XOR all bits) because it catches changes more readily.

1.3.3 External Peripheral Interface (see page 451)

The External Peripheral Interface (EPI) provides access to external devices using a parallel path. Unlike communications peripherals such as SSI, UART, and I^2C, the EPI is designed to act like a bus to external peripherals and memory.

The EPI has the following features:

- 8/16/32-bit dedicated parallel bus for external peripherals and memory
- Memory interface supports contiguous memory access independent of data bus width, thus enabling code execution directly from SDRAM, SRAM and Flash memory
- Blocking and non-blocking reads

- Separates processor from timing details through use of an internal write FIFO
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Separate channels for read and write
  - Read channel request asserted by programmable levels on the internal non-blocking read FIFO (NBRFIFO)
  - Write channel request asserted by empty on the internal write FIFO (WFIFO)

The EPI supports three primary functional modes: Synchronous Dynamic Random Access Memory (SDRAM) mode, Traditional Host-Bus mode, and General-Purpose mode. The EPI module also provides custom GPIOs; however, unlike regular GPIOs, the EPI module uses a FIFO in the same way as a communication mechanism and is speed-controlled using clocking.

- Synchronous Dynamic Random Access Memory (SDRAM) mode
  - Supports x16 (single data rate) SDRAM at up to 50 MHz
  - Supports low-cost SDRAMs up to 64 MB (512 megabits)
  - Includes automatic refresh and access to all banks/rows
  - Includes a Sleep/Standby mode to keep contents active with minimal power draw
  - Multiplexed address/data interface for reduced pin count
- Host-Bus mode
  - Traditional x8 and x16 MCU bus interface capabilities
  - Similar device compatibility options as PIC, ATmega, 8051, and others
  - Access to SRAM, NOR Flash memory, and other devices, with up to 1 MB of addressing in unmultiplexed mode and 256 MB in multiplexed mode (512 MB in Host-Bus 16 mode with no byte selects)
  - Support of both muxed and de-muxed address and data
  - Access to a range of devices supporting the non-address FIFO x8 and x16 interface variant, with support for external FIFO (XFIFO) EMPTY and FULL signals
  - Speed controlled, with read and write data wait-state counters
  - Chip select modes include ALE, CSn, Dual CSn and ALE with dual CSn
  - Manual chip-enable (or use extra address pins)
- General-Purpose mode
  - Wide parallel interfaces for fast communications with CPLDs and FPGAs
  - Data widths up to 32 bits
  - Data rates up to 150 MB/second

  - Optional "address" sizes from 4 bits to 20 bits
  - Optional clock output, read/write strobes, framing (with counter-based size), and clock-enable input
- General parallel GPIO
  - 1 to 32 bits, FIFOed with speed control
  - Useful for custom peripherals or for digital data acquisition and actuator controls

1.3.4 Serial Communications Peripherals

The LM3S9U81 controller supports both asynchronous and synchronous serial communications with:

- 10/100 Ethernet MAC and PHY
- Three CAN 2.0 A/B controllers
- USB 2.0 OTG/Host/Device
- Three UARTs with IrDA and ISO 7816 support (one UART with modem flow control and status)
- Two I^2C modules
- Two Synchronous Serial Interface modules (SSI)
- Integrated Interchip Sound (I^2S) module

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

1.3.4.1 Ethernet Controller (see page 910)

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

The Stellaris Ethernet Controller consists of a fully integrated media access controller (MAC) and network physical (PHY) interface and has the following features:

- Conforms to the IEEE 802.3-2002 specification
  - 10BASE-T/100BASE-TX IEEE-802.3 compliant. Requires only a dual 1:1 isolation transformer interface to the line
  - 10BASE-T/100BASE-TX ENDEC, 100BASE-TX scrambler/descrambler
  - Full-featured auto-negotiation
- Multiple operational modes
  - Full- and half-duplex 100 Mbps
  - Full- and half-duplex 10 Mbps

  - Power-saving and power-down modes
- Highly configurable
  - Programmable MAC address
  - LED activity selection
  - Promiscuous mode support
  - CRC error-rejection control
  - User-configurable interrupts
- Physical media manipulation
  - MDI/MDI-X cross-over support through software assist
  - Register-programmable transmit amplitude
  - Automatic polarity correction and 10BASE-T signal reception
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Separate channels for transmit and receive
  - Receive channel request asserted on packet receipt
  - Transmit channel request asserted on empty transmit FIFO

1.3.4.2 Controller Area Network (see page 859)

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

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

The LM3S9U81 microcontroller includes three CAN units with the following features:

- CAN protocol version 2.0 part A/B
- Bit rates up to 1 Mbps
- 32 message objects with individual identifier masks
- Maskable interrupt
- Disable Automatic Retransmission mode for Time-Triggered CAN (TTCAN) applications
- Programmable Loopback mode for self-test operation

- Programmable FIFO mode enables storage of multiple message objects
- Gluelessly attaches to an external CAN transceiver through the CANnTX and CANnRX signals

1.3.4.3 USB (see page 969)

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

The LM3S9U81 microcontroller supports three configurations in USB 2.0 full and low speed: USB Device, USB Host, and USB On-The-Go (negotiated on-the-go as host or device when connected to other USB-enabled systems).

The USB module has the following features:

- Complies with USB-IF certification standards
- USB 2.0 full-speed (12 Mbps) and low-speed (1.5 Mbps) operation with integrated PHY
- 4 transfer types: Control, Interrupt, Bulk, and Isochronous
- 32 endpoints
  - 1 dedicated control IN endpoint and 1 dedicated control OUT endpoint
  - 15 configurable IN endpoints and 15 configurable OUT endpoints
- 4 KB dedicated endpoint memory: one endpoint may be defined for double-buffered 1023-byte isochronous packet size
- VBUS droop and valid ID detection and interrupt
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Separate channels for transmit and receive for up to three IN endpoints and three OUT endpoints
  - Channel requests asserted when FIFO contains required amount of data

1.3.4.4 UART (see page 678)

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 LM3S9U81 microcontroller includes three fully programmable 16C550-type UARTs. Although the functionality is similar to a 16C550 UART, this UART design is not register compatible. The UART can generate individually masked interrupts from the Rx, Tx, modem flow control, modem status, and error conditions. The module generates a single combined interrupt when any of the interrupts are asserted and are unmasked.

The three UARTs have the following features:

- Programmable baud-rate generator allowing speeds up to 5 Mbps for regular speed (divide by 16) and 10 Mbps for high speed (divide by 8)
- Separate 16x8 transmit (TX) and receive (RX) FIFOs to reduce CPU interrupt service loading

- Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
- FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
- Standard asynchronous communication bits for start, stop, and parity
- Line-break generation and detection
- Fully programmable serial interface characteristics
  - 5, 6, 7, or 8 data bits
  - Even, odd, stick, or no-parity bit generation/detection
  - 1 or 2 stop bit generation
- IrDA serial-IR (SIR) encoder/decoder providing
  - Programmable use of IrDA Serial Infrared (SIR) or UART input/output
  - Support of IrDA SIR encoder/decoder functions for data rates up to 115.2 Kbps half-duplex
  - Support of normal 3/16 and low-power (1.41-2.23 µs) bit durations
  - Programmable internal clock generator enabling division of reference clock by 1 to 256 for low-power mode bit duration
- Support for communication with ISO 7816 smart cards
- Full modem handshake support (on UART1)
- LIN protocol support
- Standard FIFO-level and End-of-Transmission interrupts
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Separate channels for transmit and receive
  - Receive single request asserted when data is in the FIFO; burst request asserted at programmed FIFO level
  - Transmit single request asserted when there is space in the FIFO; burst request asserted at programmed FIFO level

1.3.4.5 I^2C (see page 784)

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

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

The LM3S9U81 microcontroller includes two I^2C modules with the following features:

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

1.3.4.6 SSI (see page 742)

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

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

The LM3S9U81 microcontroller includes two SSI modules with the following features:

- Programmable interface operation for Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces
- Master or slave operation
- Programmable clock bit rate and prescaler

- Separate transmit and receive FIFOs, each 16 bits wide and 8 locations deep
- Programmable data frame size from 4 to 16 bits
- Internal loopback test mode for diagnostic/debug testing
- Standard FIFO-based interrupts and End-of-Transmission interrupt
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Separate channels for transmit and receive
  - Receive single request asserted when data is in the FIFO; burst request asserted when FIFO contains 4 entries
  - Transmit single request asserted when there is space in the FIFO; burst request asserted when FIFO contains 4 entries

1.3.4.7 Inter-Integrated Circuit Sound (I^2S) Interface (see page 822)

The I^2S interface is a configurable serial audio core that contains a transmit module and a receive module. The module is configurable for the I^2S as well as Left-Justified and Right-Justified serial audio formats. Data can be in one of four modes: Stereo, Mono, Compact 16-bit Stereo and Compact 8-Bit Stereo.

The transmit and receive modules each have an 8-entry audio-sample FIFO. An audio sample can consist of a Left and Right Stereo sample, a Mono sample, or a Left and Right Compact Stereo sample. In Compact 16-Bit Stereo, each FIFO entry contains both the 16-bit left and 16-bit right samples, allowing efficient data transfers and requiring less memory space. In Compact 8-bit Stereo, each FIFO entry contains an 8-bit left and an 8-bit right sample, reducing memory requirements further.

Both the transmitter and receiver are capable of being a master or a slave.

The Stellaris I^2S interface has the following features:

- Configurable audio format supporting I^2S, Left-justification, and Right-justification
- Configurable sample size from 8 to 32 bits
- Mono and Stereo support
- 8-, 16-, and 32-bit FIFO interface for packing memory
- Independent transmit and receive 8-entry FIFOs
- Configurable FIFO-level interrupt and µDMA requests
- Independent transmit and receive MCLK direction control
- Transmit and receive internal MCLK sources
- Independent transmit and receive control for serial clock and word select
- MCLK and SCLK can be independently set to master or slave
- Configurable transmit zero or last sample when FIFO empty

- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Separate channels for transmit and receive
  - Burst requests
  - Channel requests asserted when FIFO contains required amount of data

1.3.5 System Integration

The LM3S9U81 microcontroller provides a variety of standard system functions integrated into the device, including:

- Direct Memory Access Controller (DMA)
- System control and clocks including on-chip precision 16-MHz oscillator
- Four 32-bit timers (up to eight 16-bit), with real-time clock capability
- Eight Capture Compare PWM (CCP) pins
- Two Watchdog Timers
  - One timer runs off the main oscillator
  - One timer runs off the precision internal oscillator
- Up to 65 GPIOs, depending on configuration
  - Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
  - Independently configurable to 2, 4 or 8 mA drive capability
  - Up to 4 GPIOs can have 18 mA drive capability

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

1.3.5.1 Direct Memory Access (see page 334)

The LM3S9U81 microcontroller includes a Direct Memory Access (DMA) controller, known as micro-DMA (µDMA). The µDMA controller provides a way to offload data transfer tasks from the Cortex-M3 processor, allowing for more efficient use of the processor and the available bus bandwidth. The µDMA controller can perform transfers between memory and peripherals. It has dedicated channels for each supported on-chip module and can be programmed to automatically perform transfers between peripherals and memory as the peripheral is ready to transfer more data. The µDMA controller provides the following features:

- ARM PrimeCell® 32-channel configurable µDMA controller
- Support for memory-to-memory, memory-to-peripheral, and peripheral-to-memory in multiple transfer modes
  - Basic for simple transfer scenarios
  - Ping-pong for continuous data flow
  - Scatter-gather for a programmable list of arbitrary transfers initiated from a single request
- Highly flexible and configurable channel operation
  - Independently configured and operated channels

  - Dedicated channels for supported on-chip modules
  - Primary and secondary channel assignments
  - One channel each for receive and transmit path for bidirectional modules
  - Dedicated channel for software-initiated transfers
  - Per-channel configurable priority scheme
  - Optional software-initiated requests for any channel
- Two levels of priority
- Design optimizations for improved bus access performance between µDMA controller and the processor core
  - µDMA controller access is subordinate to core access
  - RAM striping
  - Peripheral bus segmentation
- Data sizes of 8, 16, and 32 bits
- Transfer size is programmable in binary steps from 1 to 1024
- Source and destination address increment size of byte, half-word, word, or no increment
- Maskable peripheral requests
- Interrupt on transfer completion, with a separate interrupt per channel

1.3.5.2 System Control and Clocks (see page 186)

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

- Device identification information: version, part number, SRAM size, Flash memory size, and so on
- Power control
  - On-chip fixed Low Drop-Out (LDO) voltage regulator
  - Low-power options for microcontroller: Sleep and Deep-sleep modes with clock gating
  - Low-power options for on-chip modules: software controls shutdown of individual peripherals and memory
  - 3.3-V supply brown-out detection and reporting via interrupt or reset
- Multiple clock sources for microcontroller system clock
  - Precision Oscillator (PIOSC): On-chip resource providing a 16 MHz ±1% frequency at room temperature

- 16 MHz ±3% across temperature
- Can be recalibrated with 7-bit trim resolution
- Software power down control for low power modes

– Main Oscillator (MOSC): A frequency-accurate clock source by one of two means: an external single-ended clock source is connected to the OSC0 input pin, or an external crystal is connected across the OSC0 input and OSC1 output pins.
  - External crystal used with or without on-chip PLL: select supported frequencies from 1 MHz to 16.384 MHz.
  - External oscillator: from DC to maximum device speed

– Internal 30-kHz Oscillator: on chip resource providing a 30 kHz ± 50% frequency, used during power-saving modes

■ Flexible reset sources

– Power-on reset (POR)

– Reset pin assertion

– Brown-out reset (BOR) detector alerts to system power drops

– Software reset

– Watchdog timer reset

– MOSC failure

1.3.5.3 Programmable Timers (see page 526)

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

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

■ Operating modes:

– 16- or 32-bit programmable one-shot timer

– 16- or 32-bit programmable periodic timer

– 16-bit general-purpose timer with an 8-bit prescaler

– 32-bit Real-Time Clock (RTC) when using an external 32.768-KHz clock as the input

– 16-bit input-edge count- or time-capture modes

– 16-bit PWM mode with software-programmable output inversion of the PWM signal

■ Count up or down

■ Daisy chaining of timer modules to allow a single timer to initiate multiple timing events

■ ADC event trigger

- User-enabled stalling when the microcontroller asserts CPU Halt flag during debug (excluding RTC mode)
- Ability to determine the elapsed time between the assertion of the timer interrupt and entry into the interrupt service routine.
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Dedicated channel for each timer
  - Burst request generated on timer interrupt

1.3.5.4 CCP Pins (see page 534)

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

The LM3S9U81 microcontroller includes eight Capture Compare PWM pins (CCP) that can be programmed to operate in the following modes:

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

1.3.5.5 Watchdog Timers (see page 573)

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

The LM3S9U81 microcontroller has two Watchdog Timer modules: Watchdog Timer 0 uses the system clock for its timer clock; Watchdog Timer 1 uses the PIOSC as its timer clock. The Stellaris Watchdog Timer module has the following features:

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

1.3.5.6 Programmable GPIOs (see page 395)

General-purpose input/output (GPIO) pins offer flexibility for a variety of connections. The Stellaris GPIO module is comprised of nine physical GPIO blocks, each corresponding to an individual GPIO port. The GPIO module is FiRM-compliant (compliant to the ARM Foundation IP for Real-Time Microcontrollers specification) and supports 0-65 programmable input/output pins. The number of GPIOs available depends on the peripherals being used (see “Signal Tables” on page 1124 for the signals available to each GPIO pin).

- Up to 65 GPIOs, depending on configuration
- Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
- 5-V-tolerant in input configuration
- Two means of port access: either Advanced High-Performance Bus (AHB) with better back-to-back access performance, or the legacy Advanced Peripheral Bus (APB) for backwards-compatibility with existing code
- Fast toggle capable of a change every clock cycle for ports on AHB, every two clock cycles for ports on APB
- Programmable control for GPIO interrupts
  - Interrupt generation masking
  - Edge-triggered on rising, falling, or both
  - Level-sensitive on High or Low values
- Bit masking in both read and write operations through address lines
- Can be used to initiate an ADC sample sequence
- Pins configured as digital inputs are Schmitt-triggered
- Programmable control for GPIO pad configuration
  - Weak pull-up or pull-down resistors
  - 2-mA, 4-mA, and 8-mA pad drive for digital communication; up to four pads can sink 18-mA for high-current applications
  - Slew rate control for the 8-mA drive
  - Open drain enables
  - Digital input enables

1.3.6 Analog

The LM3S9U81 microcontroller provides analog functions integrated into the device, including:

- Two 12-bit Analog-to-Digital Converters (ADC) with 16 analog input channels and a sample rate of one million samples/second
- Three analog comparators

- 16 digital comparators
- On-chip voltage regulator

The following provides more detail on these analog functions.

1.3.6.1 ADC (see page 598)

An analog-to-digital converter (ADC) is a peripheral that converts a continuous analog voltage to a discrete digital number. The Stellaris ADC module features 12-bit conversion resolution and supports 16 input channels plus an internal temperature sensor. Four buffered sample sequencers allow rapid sampling of up to 16 analog input sources without controller intervention. Each sample sequencer provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequencer priority. Each ADC module has a digital comparator function that allows the conversion value to be diverted to a comparison unit that provides eight digital comparators.

The LM3S9U81 microcontroller provides two ADC modules with the following features:

- 16 shared analog input channels
- 12-bit precision ADC with an accurate 10-bit data compatibility mode
- Single-ended and differential-input configurations
- On-chip internal temperature sensor
- Maximum sample rate of one million samples/second
- Optional phase shift in sample time programmable from 22.5º to 337.5º
- Four programmable sample conversion sequencers from one to eight entries long, with corresponding conversion result FIFOs
- Flexible trigger control
  - Controller (software)
  - Timers
  - Analog Comparators
  - GPIO
- Hardware averaging of up to 64 samples
- Digital comparison unit providing eight digital comparators
- Converter uses an internal 3-V reference or an external reference
- Power and ground for the analog circuitry is separate from the digital power and ground
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Dedicated channel for each sample sequencer
  - ADC module uses burst requests for DMA

1.3.6.2 Analog Comparators (see page 1108)

An analog comparator is a peripheral that compares two analog voltages and provides a logical output that signals the comparison result. The LM3S9U81 microcontroller provides three independent integrated analog comparators that can be configured to drive an output or generate an interrupt or ADC event.

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

The LM3S9U81 microcontroller provides three independent integrated analog comparators with the following functions:

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

1.3.7 JTAG and ARM Serial Wire Debug (see page 174)

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

- IEEE 1149.1-1990 compatible Test Access Port (TAP) controller
- Four-bit Instruction Register (IR) chain for storing JTAG instructions
- IEEE standard instructions: BYPASS, IDCODE, SAMPLE/PRELOAD, EXTEST and INTEST
- ARM additional instructions: APACC, DPACC and ABORT
- Integrated ARM Serial Wire Debug (SWD)
  - Serial Wire JTAG Debug Port (SWJ-DP)
  - Flash Patch and Breakpoint (FPB) unit for implementing breakpoints
  - Data Watchpoint and Trace (DWT) unit for implementing watchpoints, trigger resources, and system profiling

– Instrumentation Trace Macrocell (ITM) for support of printf style debugging

– Trace Port Interface Unit (TPIU) for bridging to a Trace Port Analyzer

1.3.8 Packaging and Temperature

- Industrial-range (-40°C to 85°C) 100-pin RoHS-compliant LQFP package
- Industrial-range (-40°C to 85°C) 108-ball RoHS-compliant BGA package

1.4 Hardware Details

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

- “Pin Diagram” on page 1122
- “Signal Tables” on page 1124
- “Operating Characteristics” on page 1191
- “Electrical Characteristics” on page 1192
- “Package Information” on page 1268

2 The Cortex-M3 Processor

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

- 32-bit ARM® Cortex™-M3 architecture optimized for small-footprint embedded applications
- 80-MHz operation; 100 DMIPS performance
- Outstanding processing performance combined with fast interrupt handling
- Thumb-2 mixed 16-/32-bit instruction set delivers the high performance expected of a 32-bit ARM core in a compact memory size usually associated with 8- and 16-bit devices, typically in the range of a few kilobytes of memory for microcontroller-class applications
  - Single-cycle multiply instruction and hardware divide
  - Atomic bit manipulation (bit-banding), delivering maximum memory utilization and streamlined peripheral control
  - Unaligned data access, enabling data to be efficiently packed into memory
- Fast code execution permits slower processor clock or increases sleep mode time
- Harvard architecture characterized by separate buses for instruction and data
- Efficient processor core, system and memories
- Hardware division and fast digital-signal-processing orientated multiply accumulate
- Saturating arithmetic for signal processing
- Deterministic, high-performance interrupt handling for time-critical applications
- Memory protection unit (MPU) to provide a privileged mode for protected operating system functionality
- Enhanced system debug with extensive breakpoint and trace capabilities
- Serial Wire Debug and Serial Wire Trace reduce the number of pins required for debugging and tracing
- Migration from the ARM7 processor family for better performance and power efficiency
- Optimized for single-cycle Flash memory usage
- Ultra-low power consumption with integrated sleep modes

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

This chapter provides information on the Stellaris implementation of the Cortex-M3 processor, including the programming model, the memory model, the exception model, fault handling, and power management.

For technical details on the instruction set, see the *Cortex™-M3/M4 Instruction Set Technical User's Manual*.

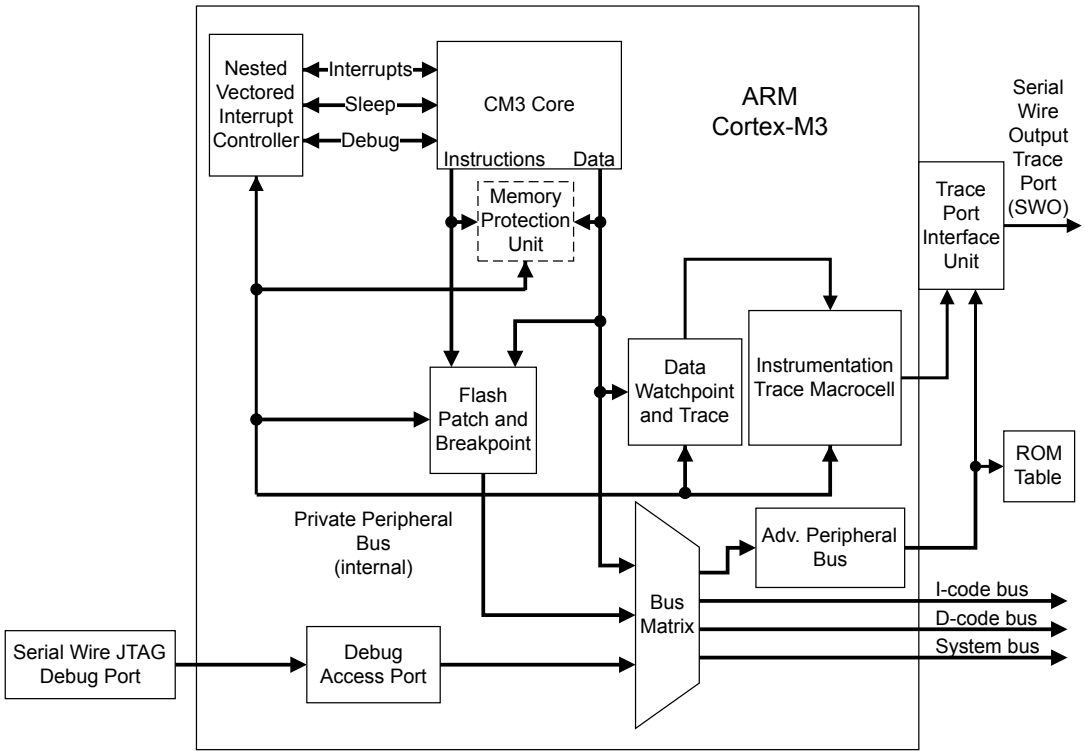
2.1 Block Diagram

The Cortex-M3 processor is built on a high-performance processor core, with a 3-stage pipeline Harvard architecture, making it ideal for demanding embedded applications. The processor delivers exceptional power efficiency through an efficient instruction set and extensively optimized design, providing high-end processing hardware including a range of single-cycle and SIMD multiplication and multiply-with-accumulate capabilities, saturating arithmetic and dedicated hardware division.

To facilitate the design of cost-sensitive devices, the Cortex-M3 processor implements tightly coupled system components that reduce processor area while significantly improving interrupt handling and system debug capabilities. The Cortex-M3 processor implements a version of the Thumb® instruction set based on Thumb-2 technology, ensuring high code density and reduced program memory requirements. The Cortex-M3 instruction set provides the exceptional performance expected of a modern 32-bit architecture, with the high code density of 8-bit and 16-bit microcontrollers.

The Cortex-M3 processor closely integrates a nested interrupt controller (NVIC), to deliver industry-leading interrupt performance. The Stellaris NVIC includes a non-maskable interrupt (NMI) and provides eight interrupt priority levels. The tight integration of the processor core and NVIC provides fast execution of interrupt service routines (ISRs), dramatically reducing interrupt latency. The hardware stacking of registers and the ability to suspend load-multiple and store-multiple operations further reduce interrupt latency. Interrupt handlers do not require any assembler stubs which removes code overhead from the ISRs. Tail-chaining optimization also significantly reduces the overhead when switching from one ISR to another. To optimize low-power designs, the NVIC integrates with the sleep modes, including Deep-sleep mode, which enables the entire device to be rapidly powered down.

Figure 2-1. CPU Block Diagram



2.2 Overview

2.2.1 System-Level Interface

The Cortex-M3 processor provides multiple interfaces using AMBA® technology to provide high-speed, low-latency memory accesses. The core supports unaligned data accesses and implements atomic bit manipulation that enables faster peripheral controls, system spinlocks, and thread-safe Boolean data handling.

The Cortex-M3 processor has a memory protection unit (MPU) that provides fine-grain memory control, enabling applications to implement security privilege levels and separate code, data and stack on a task-by-task basis.

2.2.2 Integrated Configurable Debug

The Cortex-M3 processor implements a complete hardware debug solution, providing high system visibility of the processor and memory through either a traditional JTAG port or a 2-pin Serial Wire Debug (SWD) port that is ideal for microcontrollers and other small package devices. The Stellaris implementation replaces the ARM SW-DP and JTAG-DP with the ARM CoreSight™-compliant Serial Wire JTAG Debug Port (SWJ-DP) interface. The SWJ-DP interface combines the SWD and JTAG debug ports into one module. See the *ARM® Debug Interface V5 Architecture Specification* for details on SWJ-DP.

For system trace, the processor integrates an Instrumentation Trace Macrocell (ITM) alongside data watchpoints and a profiling unit. To enable simple and cost-effective profiling of the system trace events, a Serial Wire Viewer (SWV) can export a stream of software-generated messages, data trace, and profiling information through a single pin.

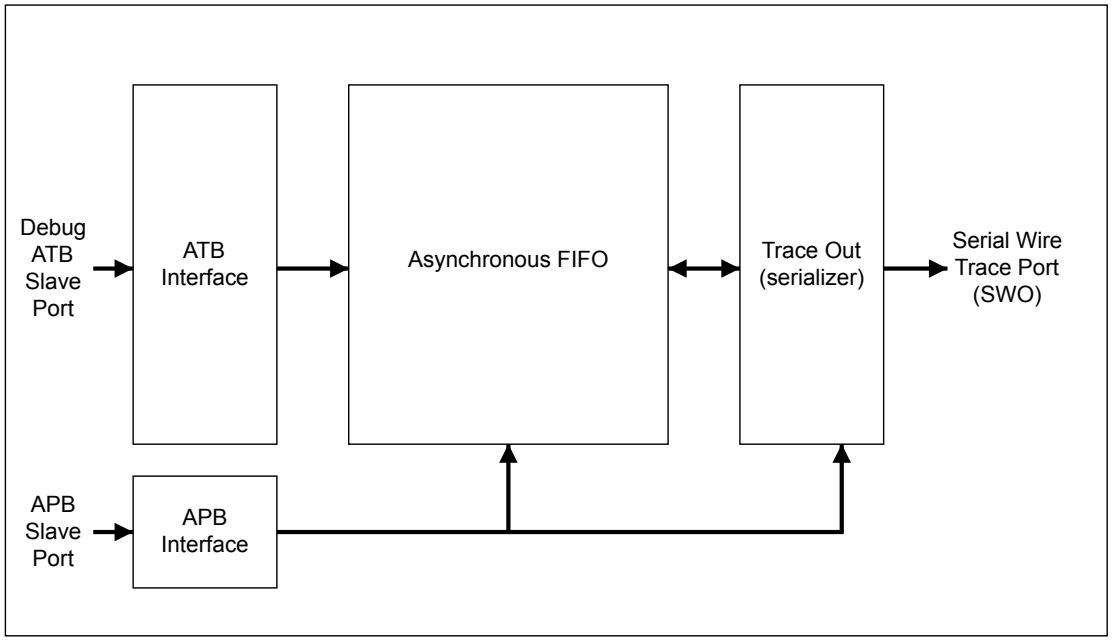
The Flash Patch and Breakpoint Unit (FPB) provides up to eight hardware breakpoint comparators that debuggers can use. The comparators in the FPB also provide remap functions of up to eight words in the program code in the CODE memory region. This enables applications stored in a read-only area of Flash memory to be patched in another area of on-chip SRAM or Flash memory. If a patch is required, the application programs the FPB to remap a number of addresses. When those addresses are accessed, the accesses are redirected to a remap table specified in the FPB configuration.

For more information on the Cortex-M3 debug capabilities, see the*ARM® Debug Interface V5 Architecture Specification*.

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, as shown in Figure 2-2 on page 68.

Figure 2-2. TPIU Block Diagram



2.2.4 Cortex-M3 System Component Details

The Cortex-M3 includes the following system components:

- SysTick

  A 24-bit count-down timer that can be used as a Real-Time Operating System (RTOS) tick timer or as a simple counter (see “System Timer (SysTick)” on page 108).

- Nested Vectored Interrupt Controller (NVIC)

  An embedded interrupt controller that supports low latency interrupt processing (see “Nested Vectored Interrupt Controller (NVIC)” on page 109).

- System Control Block (SCB)

The programming model interface to the processor. The SCB provides system implementation information and system control, including configuration, control, and reporting of system exceptions (see “System Control Block (SCB)” on page 111).

- Memory Protection Unit (MPU)

  Improves system reliability by defining the memory attributes for different memory regions. The MPU provides up to eight different regions and an optional predefined background region (see “Memory Protection Unit (MPU)” on page 111).

2.3 Programming Model

This section describes the Cortex-M3 programming model. In addition to the individual core register descriptions, information about the processor modes and privilege levels for software execution and stacks is included.

2.3.1 Processor Mode and Privilege Levels for Software Execution

The Cortex-M3 has two modes of operation:

- Thread mode

  Used to execute application software. The processor enters Thread mode when it comes out of reset.

- Handler mode

  Used to handle exceptions. When the processor has finished exception processing, it returns to Thread mode.

In addition, the Cortex-M3 has two privilege levels:

- Unprivileged

  In this mode, software has the following restrictions:

  – Limited access to the MSR and MRS instructions and no use of the CPS instruction

  – No access to the system timer, NVIC, or system control block

  – Possibly restricted access to memory or peripherals

- Privileged

  In this mode, software can use all the instructions and has access to all resources.

In Thread mode, the **CONTROL** register (see page 83) controls whether software execution is privileged or unprivileged. In Handler mode, software execution is always privileged.

Only privileged software can write to the **CONTROL** register to change the privilege level for software execution in Thread mode. Unprivileged software can use the SVC instruction to make a supervisor call to transfer control to privileged software.

2.3.2 Stacks

The processor uses a full descending stack, meaning that the stack pointer indicates the last stacked item on the memory. When the processor pushes a new item onto the stack, it decrements the stack pointer and then writes the item to the new memory location. The processor implements two stacks:

the main stack and the process stack, with a pointer for each held in independent registers (see the **SP** register on page 73).

In Thread mode, the **CONTROL** register (see page 83) controls whether the processor uses the main stack or the process stack. In Handler mode, the processor always uses the main stack. The options for processor operations are shown in Table 2-1 on page 70.

Table 2-1. Summary of Processor Mode, Privilege Level, and Stack Use

Processor Mode	Use	Privilege Level	Stack Used
Thread	Applications	Privileged or unprivileged [a]	Main stack or process stack [a]
Handler	Exception handlers	Always privileged	Main stack

a. See **CONTROL** (page 83).

2.3.3 Register Map

Figure 2-3 on page 70 shows the Cortex-M3 register set. Table 2-2 on page 71 lists the Core registers. The core registers are not memory mapped and are accessed by register name, so the base address is n/a (not applicable) and there is no offset.

Figure 2-3. Cortex-M3 Register Set

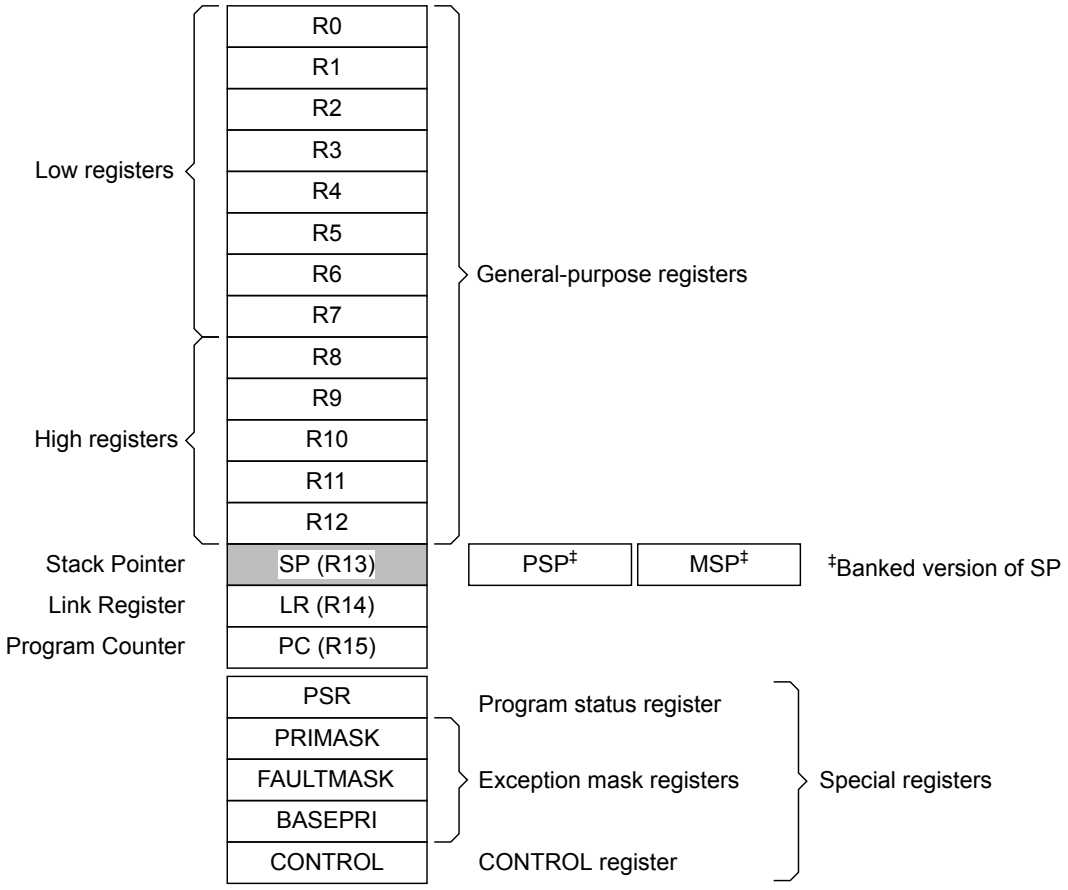


Table 2-2. Processor Register Map

Offset	Name	Type	Reset	Description	See page
-	R0	R/W	-	Cortex General-Purpose Register 0	72
-	R1	R/W	-	Cortex General-Purpose Register 1	72
-	R2	R/W	-	Cortex General-Purpose Register 2	72
-	R3	R/W	-	Cortex General-Purpose Register 3	72
-	R4	R/W	-	Cortex General-Purpose Register 4	72
-	R5	R/W	-	Cortex General-Purpose Register 5	72
-	R6	R/W	-	Cortex General-Purpose Register 6	72
-	R7	R/W	-	Cortex General-Purpose Register 7	72
-	R8	R/W	-	Cortex General-Purpose Register 8	72
-	R9	R/W	-	Cortex General-Purpose Register 9	72
-	R10	R/W	-	Cortex General-Purpose Register 10	72
-	R11	R/W	-	Cortex General-Purpose Register 11	72
-	R12	R/W	-	Cortex General-Purpose Register 12	72
-	SP	R/W	-	Stack Pointer	73
-	LR	R/W	0xFFFF.FFFF	Link Register	74
-	PC	R/W	-	Program Counter	75
-	PSR	R/W	0x0100.0000	Program Status Register	76
-	PRIMASK	R/W	0x0000.0000	Priority Mask Register	80
-	FAULTMASK	R/W	0x0000.0000	Fault Mask Register	81
-	BASEPRI	R/W	0x0000.0000	Base Priority Mask Register	82
-	CONTROL	R/W	0x0000.0000	Control Register	83

2.3.4 Register Descriptions

This section lists and describes the Cortex-M3 registers, in the order shown in Figure 2-3 on page 70. The core registers are not memory mapped and are accessed by register name rather than offset.

Note: The register type shown in the register descriptions refers to type during program execution in Thread mode and Handler mode. Debug access can differ.

Register 1: Cortex General-Purpose Register 0 (R0)

Register 2: Cortex General-Purpose Register 1 (R1)

Register 3: Cortex General-Purpose Register 2 (R2)

Register 4: Cortex General-Purpose Register 3 (R3)

Register 5: Cortex General-Purpose Register 4 (R4)

Register 6: Cortex General-Purpose Register 5 (R5)

Register 7: Cortex General-Purpose Register 6 (R6)

Register 8: Cortex General-Purpose Register 7 (R7)

Register 9: Cortex General-Purpose Register 8 (R8)

Register 10: Cortex General-Purpose Register 9 (R9)

Register 11: Cortex General-Purpose Register 10 (R10)

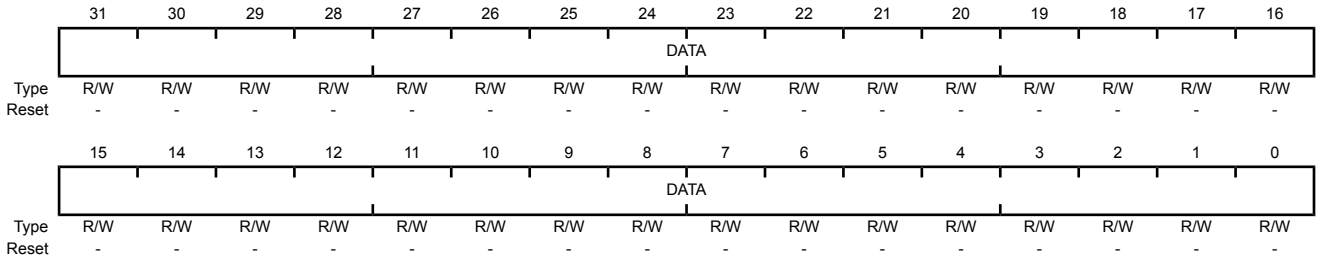
Register 12: Cortex General-Purpose Register 11 (R11)

Register 13: Cortex General-Purpose Register 12 (R12)

The **Rn** registers are 32-bit general-purpose registers for data operations and can be accessed from either privileged or unprivileged mode.

Cortex General-Purpose Register 0 (R0)

Type R/W, reset -



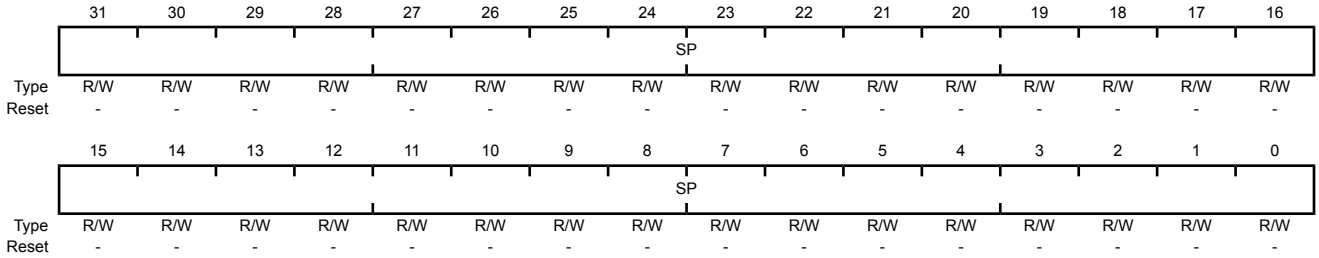
Bit/Field	Name	Type	Reset	Description
31:0	DATA	R/W	-	Register data.

Register 14: Stack Pointer (SP)

The **Stack Pointer (SP)** is register R13. In Thread mode, the function of this register changes depending on the ASP bit in the **Control Register (CONTROL)** register. When the ASP bit is clear, this register is the **Main Stack Pointer (MSP)**. When the ASP bit is set, this register is the **Process Stack Pointer (PSP)**. On reset, the ASP bit is clear, and the processor loads the **MSP** with the value from address 0x0000.0000. The **MSP** can only be accessed in privileged mode; the **PSP** can be accessed in either privileged or unprivileged mode.

Stack Pointer (SP)

Type R/W, reset -



Bit/Field	Name	Type	Reset	Description
31:0	SP	R/W	-	This field is the address of the stack pointer.

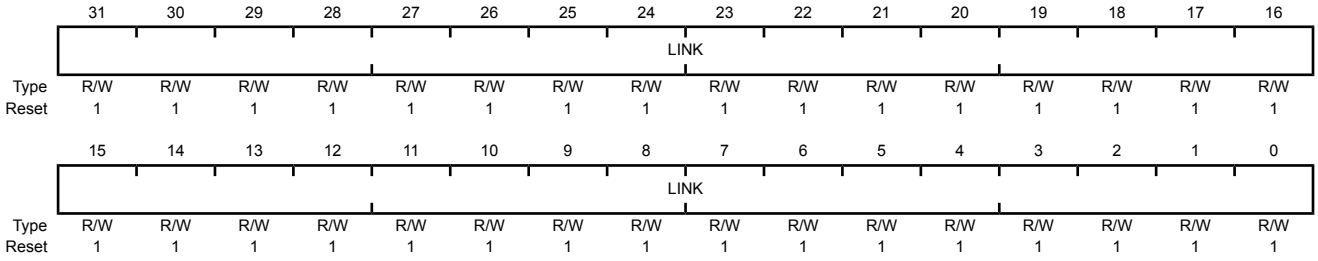
Register 15: Link Register (LR)

The **Link Register (LR)** is register R14, and it stores the return information for subroutines, function calls, and exceptions. **LR** can be accessed from either privileged or unprivileged mode.

EXC_RETURN is loaded into **LR** on exception entry. See Table 2-10 on page 101 for the values and description.

Link Register (LR)

Type R/W, reset 0xFFFF.FFFF



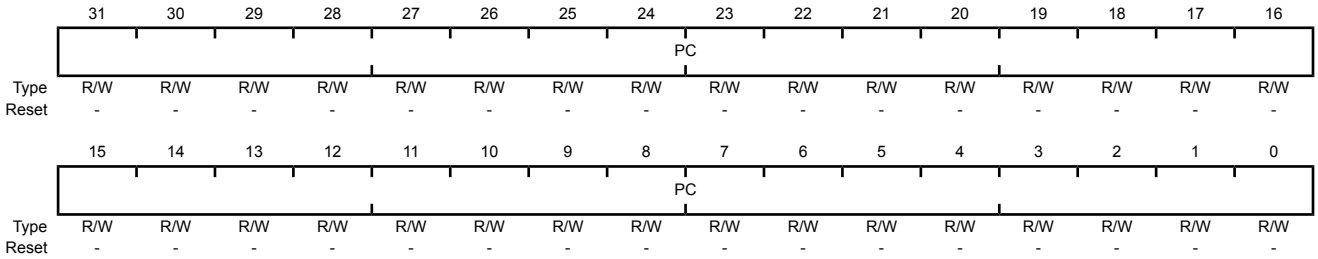
Bit/Field	Name	Type	Reset	Description
31:0	LINK	R/W	0xFFFF.FFFF	This field is the return address.

Register 16: Program Counter (PC)

The **Program Counter (PC)** is register R15, and it contains the current program address. On reset, the processor loads the **PC** with the value of the reset vector, which is at address 0x0000.0004. Bit 0 of the reset vector is loaded into the THUMB bit of the **EPSR** at reset and must be 1. The **PC** register can be accessed in either privileged or unprivileged mode.

Program Counter (PC)

Type R/W, reset -



Bit/Field	Name	Type	Reset	Description
31:0	PC	R/W	-	This field is the current program address.

Register 17: Program Status Register (PSR)

Note: This register is also referred to as **xPSR**.

The **Program Status Register (PSR)** has three functions, and the register bits are assigned to the different functions:

- **Application Program Status Register (APSR)**, bits 31:27,
- **Execution Program Status Register (EPSR)**, bits 26:24, 15:10
- **Interrupt Program Status Register (IPSR)**, bits 6:0

The **PSR**, **IPSR**, and **EPSR** registers can only be accessed in privileged mode; the **APSR** register can be accessed in either privileged or unprivileged mode.

APSR contains the current state of the condition flags from previous instruction executions.

EPSR contains the Thumb state bit and the execution state bits for the If-Then (`IT`) instruction or the Interruptible-Continuable Instruction (`ICI`) field for an interrupted load multiple or store multiple instruction. Attempts to read the **EPSR** directly through application software using the `MSR` instruction always return zero. Attempts to write the **EPSR** using the `MSR` instruction in application software are always ignored. Fault handlers can examine the **EPSR** value in the stacked **PSR** to determine the operation that faulted (see “Exception Entry and Return” on page 99).

IPSR contains the exception type number of the current Interrupt Service Routine (ISR).

These registers can be accessed individually or as a combination of any two or all three registers, using the register name as an argument to the `MSR` or `MRS` instructions. For example, all of the registers can be read using **PSR** with the `MRS` instruction, or **APSR** only can be written to using **APSR** with the `MSR` instruction. page 76 shows the possible register combinations for the **PSR**. See the `MRS` and `MSR` instruction descriptions in the *Cortex™-M3/M4 Instruction Set Technical User's Manual* for more information about how to access the program status registers.

Table 2-3. PSR Register Combinations

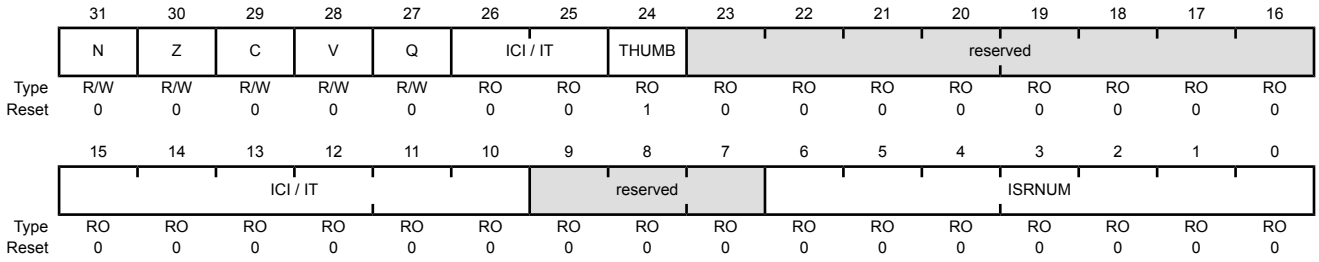
Register	Type	Combination
PSR	R/W[a, b]	**APSR**, **EPSR**, and **IPSR**
IEPSR	RO	**EPSR** and **IPSR**
IAPSR	R/W[a]	**APSR** and **IPSR**
EAPSR	R/W[b]	**APSR** and **EPSR**

a. The processor ignores writes to the **IPSR** bits.

b. Reads of the **EPSR** bits return zero, and the processor ignores writes to these bits.

Program Status Register (PSR)

Type R/W, reset 0x0100.0000



Bit/Field	Name	Type	Reset	Description
31	N	R/W	0	**APSR** Negative or Less Flag Value Description 1 The previous operation result was negative or less than. 0 The previous operation result was positive, zero, greater than, or equal. The value of this bit is only meaningful when accessing **PSR** or **APSR**.
30	Z	R/W	0	**APSR** Zero Flag Value Description 1 The previous operation result was zero. 0 The previous operation result was non-zero. The value of this bit is only meaningful when accessing **PSR** or **APSR**.
29	C	R/W	0	**APSR** Carry or Borrow Flag Value Description 1 The previous add operation resulted in a carry bit or the previous subtract operation did not result in a borrow bit. 0 The previous add operation did not result in a carry bit or the previous subtract operation resulted in a borrow bit. The value of this bit is only meaningful when accessing **PSR** or **APSR**.
28	V	R/W	0	**APSR** Overflow Flag Value Description 1 The previous operation resulted in an overflow. 0 The previous operation did not result in an overflow. The value of this bit is only meaningful when accessing **PSR** or **APSR**.
27	Q	R/W	0	**APSR** DSP Overflow and Saturation Flag Value Description 1 DSP Overflow or saturation has occurred. 0 DSP overflow or saturation has not occurred since reset or since the bit was last cleared. The value of this bit is only meaningful when accessing **PSR** or **APSR**. This bit is cleared by software using an MRS instruction.

Bit/Field	Name	Type	Reset	Description
26:25	ICI / IT	RO	0x0	**EPSR** ICI / IT status These bits, along with bits 15:10, contain the Interruptible-Continuable Instruction (ICI) field for an interrupted load multiple or store multiple instruction or the execution state bits of the IT instruction. When **EPSR** holds the ICI execution state, bits 26:25 are zero. The If-Then block contains up to four instructions following an IT instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. See the *Cortex™-M3/M4 Instruction Set Technical User's Manual* for more information. The value of this field is only meaningful when accessing **PSR** or **EPSR**.
24	THUMB	RO	1	**EPSR** Thumb State This bit indicates the Thumb state and should always be set. The following can clear the THUMB bit: ■ The BLX, BX and POP{PC} instructions ■ Restoration from the stacked **xPSR** value on an exception return ■ Bit 0 of the vector value on an exception entry or reset Attempting to execute instructions when this bit is clear results in a fault or lockup. See “Lockup” on page 103 for more information. The value of this bit is only meaningful when accessing **PSR** or **EPSR**.
23:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:10	ICI / IT	RO	0x0	**EPSR** ICI / IT status These bits, along with bits 26:25, contain the Interruptible-Continuable Instruction (ICI) field for an interrupted load multiple or store multiple instruction or the execution state bits of the IT instruction. When an interrupt occurs during the execution of an LDM, STM, PUSH or POP instruction, the processor stops the load multiple or store multiple instruction operation temporarily and stores the next register operand in the multiple operation to bits 15:12. After servicing the interrupt, the processor returns to the register pointed to by bits 15:12 and resumes execution of the multiple load or store instruction. When **EPSR** holds the ICI execution state, bits 11:10 are zero. The If-Then block contains up to four instructions following a 16-bit IT instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. See the *Cortex™-M3/M4 Instruction Set Technical User's Manual* for more information. The value of this field is only meaningful when accessing **PSR** or **EPSR**.
9:7	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

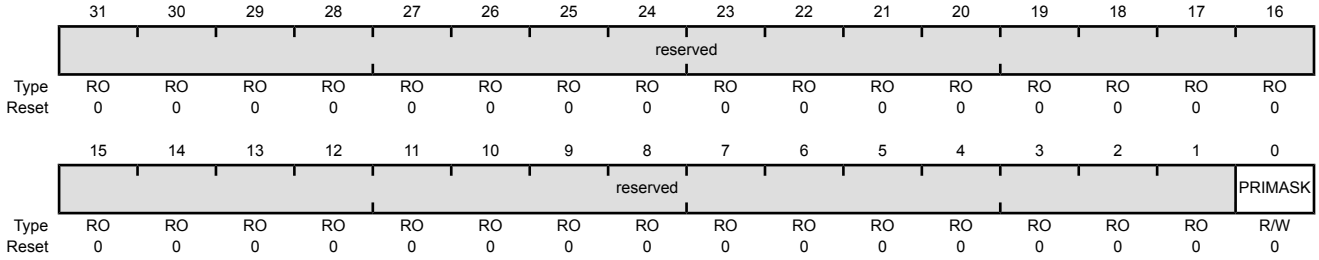
Bit/Field	Name	Type	Reset	Description
6:0	ISRNUM	RO	0x00	**IPSR** ISR Number This field contains the exception type number of the current Interrupt Service Routine (ISR). Value — Description 0x00 — Thread mode 0x01 — Reserved 0x02 — NMI 0x03 — Hard fault 0x04 — Memory management fault 0x05 — Bus fault 0x06 — Usage fault 0x07-0x0A — Reserved 0x0B — SVCall 0x0C — Reserved for Debug 0x0D — Reserved 0x0E — PendSV 0x0F — SysTick 0x10 — Interrupt Vector 0 0x11 — Interrupt Vector 1 ... — ... 0x46 — Interrupt Vector 54 0x47-0x7F — Reserved See “Exception Types” on page 94 for more information. The value of this field is only meaningful when accessing **PSR** or **IPSR**.

Register 18: Priority Mask Register (PRIMASK)

The **PRIMASK** register prevents activation of all exceptions with programmable priority. Reset, non-maskable interrupt (NMI), and hard fault are the only exceptions with fixed priority. Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. The MSR and MRS instructions are used to access the **PRIMASK** register, and the CPS instruction may be used to change the value of the **PRIMASK** register. See the *Cortex™-M3/M4 Instruction Set Technical User's Manual* for more information on these instructions. For more information on exception priority levels, see “Exception Types” on page 94.

Priority Mask Register (PRIMASK)

Type R/W, reset 0x0000.0000



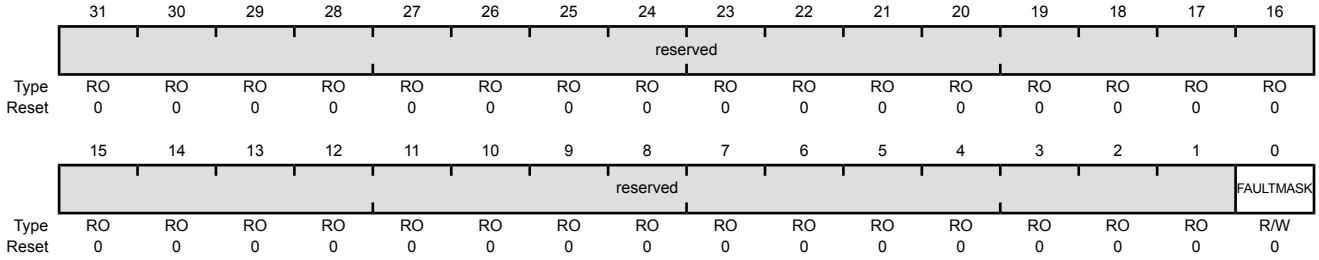
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	PRIMASK	R/W	0	Priority Mask Value Description 1 Prevents the activation of all exceptions with configurable priority. 0 No effect.

Register 19: Fault Mask Register (FAULTMASK)

The **FAULTMASK** register prevents activation of all exceptions except for the Non-Maskable Interrupt (NMI). Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. The MSR and MRS instructions are used to access the **FAULTMASK** register, and the CPS instruction may be used to change the value of the **FAULTMASK** register. See the *Cortex™-M3/M4 Instruction Set Technical User's Manual* for more information on these instructions. For more information on exception priority levels, see “Exception Types” on page 94.

Fault Mask Register (FAULTMASK)

Type R/W, reset 0x0000.0000



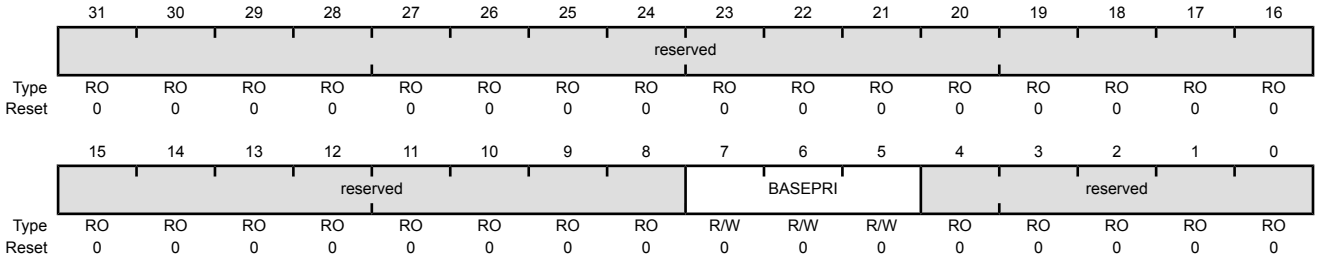
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	FAULTMASK	R/W	0	Fault Mask Value Description 1 Prevents the activation of all exceptions except for NMI. 0 No effect. The processor clears the FAULTMASK bit on exit from any exception handler except the NMI handler.

Register 20: Base Priority Mask Register (BASEPRI)

The **BASEPRI** register defines the minimum priority for exception processing. When **BASEPRI** is set to a nonzero value, it prevents the activation of all exceptions with the same or lower priority level as the **BASEPRI** value. Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. For more information on exception priority levels, see “Exception Types” on page 94.

Base Priority Mask Register (BASEPRI)

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:5	BASEPRI	R/W	0x0	Base Priority Any exception that has a programmable priority level with the same or lower priority as the value of this field is masked. The **PRIMASK** register can be used to mask all exceptions with programmable priority levels. Higher priority exceptions have lower priority levels. Value Description 0x0 All exceptions are unmasked. 0x1 All exceptions with priority level 1-7 are masked. 0x2 All exceptions with priority level 2-7 are masked. 0x3 All exceptions with priority level 3-7 are masked. 0x4 All exceptions with priority level 4-7 are masked. 0x5 All exceptions with priority level 5-7 are masked. 0x6 All exceptions with priority level 6-7 are masked. 0x7 All exceptions with priority level 7 are masked.
4:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 21: Control Register (CONTROL)

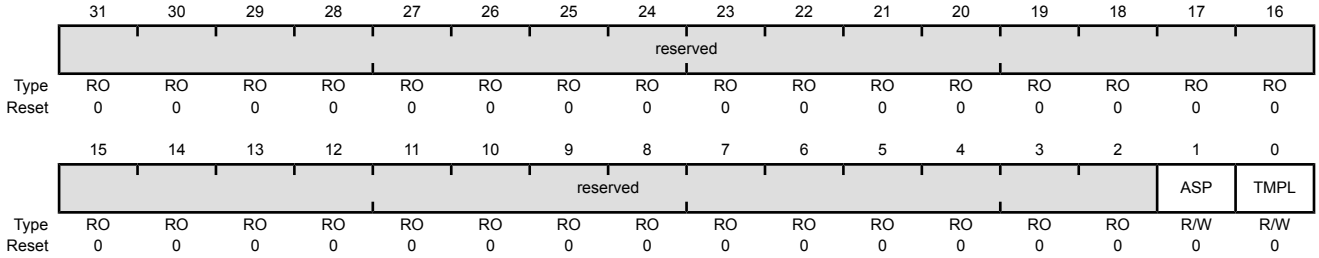
The **CONTROL** register controls the stack used and the privilege level for software execution when the processor is in Thread mode. This register is only accessible in privileged mode.

Handler mode always uses **MSP**, so the processor ignores explicit writes to the ASP bit of the **CONTROL** register when in Handler mode. The exception entry and return mechanisms automatically update the **CONTROL** register based on the EXC_RETURN value (see Table 2-10 on page 101). In an OS environment, threads running in Thread mode should use the process stack and the kernel and exception handlers should use the main stack. By default, Thread mode uses **MSP**. To switch the stack pointer used in Thread mode to **PSP**, either use the MSR instruction to set the ASP bit, as detailed in the *Cortex™-M3/M4 Instruction Set Technical User's Manual*, or perform an exception return to Thread mode with the appropriate EXC_RETURN value, as shown in Table 2-10 on page 101.

Note: When changing the stack pointer, software must use an ISB instruction immediately after the MSR instruction, ensuring that instructions after the ISB execute use the new stack pointer. See the *Cortex™-M3/M4 Instruction Set Technical User's Manual*.

Control Register (CONTROL)

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	ASP	R/W	0	Active Stack Pointer Value Description 1 **PSP** is the current stack pointer. 0 **MSP** is the current stack pointer In Handler mode, this bit reads as zero and ignores writes. The Cortex-M3 updates this bit automatically on exception return.
0	TMPL	R/W	0	Thread Mode Privilege Level Value Description 1 Unprivileged software can be executed in Thread mode. 0 Only privileged software can be executed in Thread mode.

2.3.5 Exceptions and Interrupts

The Cortex-M3 processor supports interrupts and system exceptions. The processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions. An exception changes the normal flow of software control. The processor uses Handler mode to handle all exceptions except for reset. See “Exception Entry and Return” on page 99 for more information.

The NVIC registers control interrupt handling. See “Nested Vectored Interrupt Controller (NVIC)” on page 109 for more information.

2.3.6 Data Types

The Cortex-M3 supports 32-bit words, 16-bit halfwords, and 8-bit bytes. The processor also supports 64-bit data transfer instructions. All instruction and data memory accesses are little endian. See “Memory Regions, Types and Attributes” on page 86 for more information.

2.4 Memory Model

This section describes the processor memory map, the behavior of memory accesses, and the bit-banding features. The processor has a fixed memory map that provides up to 4 GB of addressable memory.

The memory map for the LM3S9U81 controller is provided in Table 2-4 on page 84. In this manual, register addresses are given as a hexadecimal increment, relative to the module’s base address as shown in the memory map.

The regions for SRAM and peripherals include bit-band regions. Bit-banding provides atomic operations to bit data (see “Bit-Banding” on page 89).

The processor reserves regions of the Private peripheral bus (PPB) address range for core peripheral registers (see “Cortex-M3 Peripherals” on page 108).

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

Table 2-4. Memory Map

Start	End	Description	For details, see page ...
Memory			
0x0000.0000	0x0005.FFFF	On-chip Flash	291
0x0006.0000	0x00FF.FFFF	Reserved	-
0x0100.0000	0x1FFF.FFFF	Reserved for ROM	289
0x2000.0000	0x2001.FFFF	Bit-banded on-chip SRAM	289
0x2002.0000	0x21FF.FFFF	Reserved	-
0x2200.0000	0x222F.FFFF	Bit-band alias of bit-banded on-chip SRAM starting at 0x2000.0000	289
0x2230.0000	0x3FFF.FFFF	Reserved	-
FiRM Peripherals			
0x4000.0000	0x4000.0FFF	Watchdog timer 0	576
0x4000.1000	0x4000.1FFF	Watchdog timer 1	576
0x4000.2000	0x4000.3FFF	Reserved	-
0x4000.4000	0x4000.4FFF	GPIO Port A	408
0x4000.5000	0x4000.5FFF	GPIO Port B	408
0x4000.6000	0x4000.6FFF	GPIO Port C	408

Table 2-4. Memory Map *(continued)*

Start	End	Description	For details, see page ...
0x4000.7000	0x4000.7FFF	GPIO Port D	408
0x4000.8000	0x4000.8FFF	SSI0	756
0x4000.9000	0x4000.9FFF	SSI1	756
0x4000.A000	0x4000.BFFF	Reserved	-
0x4000.C000	0x4000.CFFF	UART0	692
0x4000.D000	0x4000.DFFF	UART1	692
0x4000.E000	0x4000.EFFF	UART2	692
0x4000.F000	0x4001.FFFF	Reserved	-
Peripherals			
0x4002.0000	0x4002.0FFF	I^2C 0	800
0x4002.1000	0x4002.1FFF	I^2C 1	800
0x4002.2000	0x4002.3FFF	Reserved	-
0x4002.4000	0x4002.4FFF	GPIO Port E	408
0x4002.5000	0x4002.5FFF	GPIO Port F	408
0x4002.6000	0x4002.6FFF	GPIO Port G	408
0x4002.7000	0x4002.7FFF	GPIO Port H	408
0x4002.8000	0x4002.FFFF	Reserved	-
0x4003.0000	0x4003.0FFF	Timer 0	542
0x4003.1000	0x4003.1FFF	Timer 1	542
0x4003.2000	0x4003.2FFF	Timer 2	542
0x4003.3000	0x4003.3FFF	Timer 3	542
0x4003.4000	0x4003.7FFF	Reserved	-
0x4003.8000	0x4003.8FFF	ADC0	620
0x4003.9000	0x4003.9FFF	ADC1	620
0x4003.A000	0x4003.BFFF	Reserved	-
0x4003.C000	0x4003.CFFF	Analog Comparators	1108
0x4003.D000	0x4003.DFFF	GPIO Port J	408
0x4003.E000	0x4003.FFFF	Reserved	-
0x4004.0000	0x4004.0FFF	CAN0 Controller	880
0x4004.1000	0x4004.1FFF	CAN1 Controller	880
0x4004.2000	0x4004.2FFF	CAN2 Controller	880
0x4004.3000	0x4004.7FFF	Reserved	-
0x4004.8000	0x4004.8FFF	Ethernet Controller	923
0x4004.9000	0x4004.FFFF	Reserved	-
0x4005.0000	0x4005.0FFF	USB	996
0x4005.1000	0x4005.3FFF	Reserved	-
0x4005.4000	0x4005.4FFF	I^2S0	834
0x4005.5000	0x4005.7FFF	Reserved	-
0x4005.8000	0x4005.8FFF	GPIO Port A (AHB aperture)	408
0x4005.9000	0x4005.9FFF	GPIO Port B (AHB aperture)	408
0x4005.A000	0x4005.AFFF	GPIO Port C (AHB aperture)	408

Table 2-4. Memory Map *(continued)*

Start	End	Description	For details, see page ...
0x4005.B000	0x4005.BFFF	GPIO Port D (AHB aperture)	408
0x4005.C000	0x4005.CFFF	GPIO Port E (AHB aperture)	408
0x4005.D000	0x4005.DFFF	GPIO Port F (AHB aperture)	408
0x4005.E000	0x4005.EFFF	GPIO Port G (AHB aperture)	408
0x4005.F000	0x4005.FFFF	GPIO Port H (AHB aperture)	408
0x4006.0000	0x4006.0FFF	GPIO Port J (AHB aperture)	408
0x4006.1000	0x400C.FFFF	Reserved	-
0x400D.0000	0x400D.0FFF	EPI 0	482
0x400D.1000	0x400F.CFFF	Reserved	-
0x400F.D000	0x400F.DFFF	Flash memory control	298
0x400F.E000	0x400F.EFFF	System control	203
0x400F.F000	0x400F.FFFF	µDMA	356
0x4010.0000	0x41FF.FFFF	Reserved	-
0x4200.0000	0x43FF.FFFF	Bit-banded alias of 0x4000.0000 through 0x400F.FFFF	-
0x4400.0000	0x5FFF.FFFF	Reserved	-
0x6000.0000	0xDFFF.FFFF	EPI0 mapped peripheral and RAM	-
Private Peripheral Bus			
0xE000.0000	0xE000.0FFF	Instrumentation Trace Macrocell (ITM)	67
0xE000.1000	0xE000.1FFF	Data Watchpoint and Trace (DWT)	67
0xE000.2000	0xE000.2FFF	Flash Patch and Breakpoint (FPB)	67
0xE000.3000	0xE000.DFFF	Reserved	-
0xE000.E000	0xE000.EFFF	Cortex-M3 Peripherals (SysTick, NVIC, MPU and SCB)	116
0xE000.F000	0xE003.FFFF	Reserved	-
0xE004.0000	0xE004.0FFF	Trace Port Interface Unit (TPIU)	68
0xE004.1000	0xFFFF.FFFF	Reserved	-

2.4.1 Memory Regions, Types and Attributes

The memory map and the programming of the MPU split the memory map into regions. Each region has a defined memory type, and some regions have additional memory attributes. The memory type and attributes determine the behavior of accesses to the region.

The memory types are:

- Normal: The processor can re-order transactions for efficiency and perform speculative reads.
- Device: The processor preserves transaction order relative to other transactions to Device or Strongly Ordered memory.
- Strongly Ordered: The processor preserves transaction order relative to all other transactions.

The different ordering requirements for Device and Strongly Ordered memory mean that the memory system can buffer a write to Device memory but must not buffer a write to Strongly Ordered memory.

An additional memory attribute is Execute Never (XN), which means the processor prevents instruction accesses. A fault exception is generated only on execution of an instruction executed from an XN region.

2.4.2 Memory System Ordering of Memory Accesses

For most memory accesses caused by explicit memory access instructions, the memory system does not guarantee that the order in which the accesses complete matches the program order of the instructions, providing the order does not affect the behavior of the instruction sequence. Normally, if correct program execution depends on two memory accesses completing in program order, software must insert a memory barrier instruction between the memory access instructions (see “Software Ordering of Memory Accesses” on page 88).

However, the memory system does guarantee ordering of accesses to Device and Strongly Ordered memory. For two memory access instructions A1 and A2, if both A1 and A2 are accesses to either Device or Strongly Ordered memory, and if A1 occurs before A2 in program order, A1 is always observed before A2.

2.4.3 Behavior of Memory Accesses

Table 2-5 on page 87 shows the behavior of accesses to each region in the memory map. See “Memory Regions, Types and Attributes” on page 86 for more information on memory types and the XN attribute. Stellaris devices may have reserved memory areas within the address ranges shown below (refer to Table 2-4 on page 84 for more information).

Table 2-5. Memory Access Behavior

Address Range	Memory Region	Memory Type	Execute Never (XN)	Description
0x0000.0000 - 0x1FFF.FFFF	Code	Normal	-	This executable region is for program code. Data can also be stored here.
0x2000.0000 - 0x3FFF.FFFF	SRAM	Normal	-	This executable region is for data. Code can also be stored here. This region includes bit band and bit band alias areas (see Table 2-6 on page 89).
0x4000.0000 - 0x5FFF.FFFF	Peripheral	Device	XN	This region includes bit band and bit band alias areas (see Table 2-7 on page 89).
0x6000.0000 - 0x9FFF.FFFF	External RAM	Normal	-	This executable region is for data.
0xA000.0000 - 0xDFFF.FFFF	External device	Device	XN	This region is for external device memory.
0xE000.0000- 0xE00F.FFFF	Private peripheral bus	Strongly Ordered	XN	This region includes the NVIC, system timer, and system control block.
0xE010.0000- 0xFFFF.FFFF	Reserved	-	-	-

The Code, SRAM, and external RAM regions can hold programs. However, it is recommended that programs always use the Code region because the Cortex-M3 has separate buses that can perform instruction fetches and data accesses simultaneously.

The MPU can override the default memory access behavior described in this section. For more information, see “Memory Protection Unit (MPU)” on page 111.

The Cortex-M3 prefetches instructions ahead of execution and speculatively prefetches from branch target addresses.

2.4.4 Software Ordering of Memory Accesses

The order of instructions in the program flow does not always guarantee the order of the corresponding memory transactions for the following reasons:

- The processor can reorder some memory accesses to improve efficiency, providing this does not affect the behavior of the instruction sequence.
- The processor has multiple bus interfaces.
- Memory or devices in the memory map have different wait states.
- Some memory accesses are buffered or speculative.

"Memory System Ordering of Memory Accesses" on page 87 describes the cases where the memory system guarantees the order of memory accesses. Otherwise, if the order of memory accesses is critical, software must include memory barrier instructions to force that ordering. The Cortex-M3 has the following memory barrier instructions:

- The Data Memory Barrier (`DMB`) instruction ensures that outstanding memory transactions complete before subsequent memory transactions.
- The Data Synchronization Barrier (`DSB`) instruction ensures that outstanding memory transactions complete before subsequent instructions execute.
- The Instruction Synchronization Barrier (`ISB`) instruction ensures that the effect of all completed memory transactions is recognizable by subsequent instructions.

Memory barrier instructions can be used in the following situations:

- MPU programming
  - If the MPU settings are changed and the change must be effective on the very next instruction, use a `DSB` instruction to ensure the effect of the MPU takes place immediately at the end of context switching.
  - Use an `ISB` instruction to ensure the new MPU setting takes effect immediately after programming the MPU region or regions, if the MPU configuration code was accessed using a branch or call. If the MPU configuration code is entered using exception mechanisms, then an `ISB` instruction is not required.
- Vector table

  If the program changes an entry in the vector table and then enables the corresponding exception, use a `DMB` instruction between the operations. The `DMB` instruction ensures that if the exception is taken immediately after being enabled, the processor uses the new exception vector.
- Self-modifying code

  If a program contains self-modifying code, use an `ISB` instruction immediately after the code modification in the program. The `ISB` instruction ensures subsequent instruction execution uses the updated program.
- Memory map switching

If the system contains a memory map switching mechanism, use a DSB instruction after switching the memory map in the program. The DSB instruction ensures subsequent instruction execution uses the updated memory map.

- Dynamic exception priority change

  When an exception priority has to change when the exception is pending or active, use DSB instructions after the change. The change then takes effect on completion of the DSB instruction.

Memory accesses to Strongly Ordered memory, such as the System Control Block, do not require the use of DMB instructions.

For more information on the memory barrier instructions, see the *Cortex™-M3/M4 Instruction Set Technical User's Manual*.

2.4.5 Bit-Banding

A bit-band region maps each word in a bit-band alias region to a single bit in the bit-band region. The bit-band regions occupy the lowest 1 MB of the SRAM and peripheral memory regions. Accesses to the 32-MB SRAM alias region map to the 1-MB SRAM bit-band region, as shown in Table 2-6 on page 89. Accesses to the 32-MB peripheral alias region map to the 1-MB peripheral bit-band region, as shown in Table 2-7 on page 89. For the specific address range of the bit-band regions, see Table 2-4 on page 84.

Note: A word access to the SRAM or the peripheral bit-band alias region maps to a single bit in the SRAM or peripheral bit-band region.

A word access to a bit band address results in a word access to the underlying memory, and similarly for halfword and byte accesses. This allows bit band accesses to match the access requirements of the underlying peripheral.

Table 2-6. SRAM Memory Bit-Banding Regions

Address Range	Memory Region	Instruction and Data Accesses
0x2000.0000 - 0x200F.FFFF	SRAM bit-band region	Direct accesses to this memory range behave as SRAM memory accesses, but this region is also bit addressable through bit-band alias.
0x2200.0000 - 0x23FF.FFFF	SRAM bit-band alias	Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not remapped.

Table 2-7. Peripheral Memory Bit-Banding Regions

Address Range	Memory Region	Instruction and Data Accesses
0x4000.0000 - 0x400F.FFFF	Peripheral bit-band region	Direct accesses to this memory range behave as peripheral memory accesses, but this region is also bit addressable through bit-band alias.
0x4200.0000 - 0x43FF.FFFF	Peripheral bit-band alias	Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not permitted.

The following formula shows how the alias region maps onto the bit-band region:

```
bit_word_offset = (byte_offset x 32) + (bit_number x 4)

bit_word_addr = bit_band_base + bit_word_offset
```

where:

bit_word_offset
The position of the target bit in the bit-band memory region.

bit_word_addr
The address of the word in the alias memory region that maps to the targeted bit.

bit_band_base
The starting address of the alias region.

byte_offset
The number of the byte in the bit-band region that contains the targeted bit.

bit_number
The bit position, 0-7, of the targeted bit.

Figure 2-4 on page 91 shows examples of bit-band mapping between the SRAM bit-band alias region and the SRAM bit-band region:

- The alias word at 0x23FF.FFE0 maps to bit 0 of the bit-band byte at 0x200F.FFFF:

  ```
  0x23FF.FFE0 = 0x2200.0000 + (0x000F.FFFF*32) + (0*4)
  ```

- The alias word at 0x23FF.FFFC maps to bit 7 of the bit-band byte at 0x200F.FFFF:

  ```
  0x23FF.FFFC = 0x2200.0000 + (0x000F.FFFF*32) + (7*4)
  ```

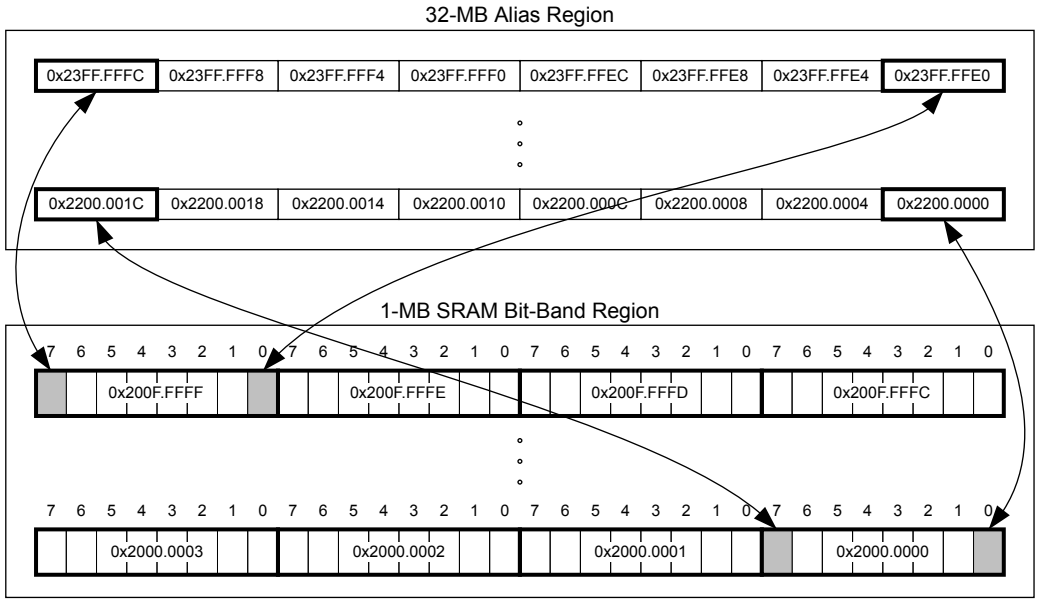
- The alias word at 0x2200.0000 maps to bit 0 of the bit-band byte at 0x2000.0000:

  ```
  0x2200.0000 = 0x2200.0000 + (0*32) + (0*4)
  ```

- The alias word at 0x2200.001C maps to bit 7 of the bit-band byte at 0x2000.0000:

  ```
  0x2200.001C = 0x2200.0000+ (0*32) + (7*4)
  ```

Figure 2-4. Bit-Band Mapping



2.4.5.1 Directly Accessing an Alias Region

Writing to a word in the alias region updates a single bit in the bit-band region.

Bit 0 of the value written to a word in the alias region determines the value written to the targeted bit in the bit-band region. Writing a value with bit 0 set writes a 1 to the bit-band bit, and writing a value with bit 0 clear writes a 0 to the bit-band bit.

Bits 31:1 of the alias word have no effect on the bit-band bit. Writing 0x01 has the same effect as writing 0xFF. Writing 0x00 has the same effect as writing 0x0E.

When reading a word in the alias region, 0x0000.0000 indicates that the targeted bit in the bit-band region is clear and 0x0000.0001 indicates that the targeted bit in the bit-band region is set.

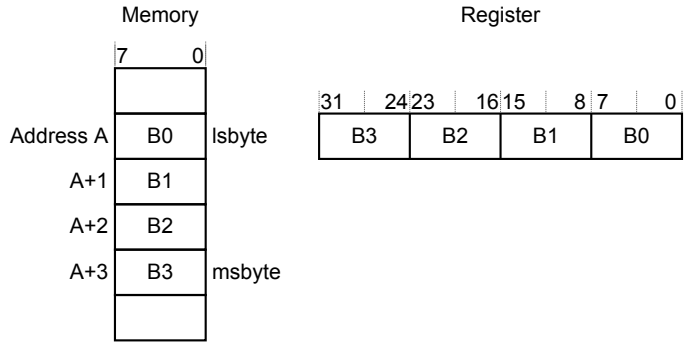
2.4.5.2 Directly Accessing a Bit-Band Region

"Behavior of Memory Accesses" on page 87 describes the behavior of direct byte, halfword, or word accesses to the bit-band regions.

2.4.6 Data Storage

The processor views memory as a linear collection of bytes numbered in ascending order from zero. For example, bytes 0-3 hold the first stored word, and bytes 4-7 hold the second stored word. Data is stored in little-endian format, with the least-significant byte (lsbyte) of a word stored at the lowest-numbered byte, and the most-significant byte (msbyte) stored at the highest-numbered byte. Figure 2-5 on page 92 illustrates how data is stored.

Figure 2-5. Data Storage



2.4.7 Synchronization Primitives

The Cortex-M3 instruction set includes pairs of synchronization primitives which provide a non-blocking mechanism that a thread or process can use to obtain exclusive access to a memory location. Software can use these primitives to perform a guaranteed read-modify-write memory update sequence or for a semaphore mechanism.

A pair of synchronization primitives consists of:

- A Load-Exclusive instruction, which is used to read the value of a memory location and requests exclusive access to that location.
- A Store-Exclusive instruction, which is used to attempt to write to the same memory location and returns a status bit to a register. If this status bit is clear, it indicates that the thread or process gained exclusive access to the memory and the write succeeds; if this status bit is set, it indicates that the thread or process did not gain exclusive access to the memory and no write was performed.

The pairs of Load-Exclusive and Store-Exclusive instructions are:

- The word instructions `LDREX` and `STREX`
- The halfword instructions `LDREXH` and `STREXH`
- The byte instructions `LDREXB` and `STREXB`

Software must use a Load-Exclusive instruction with the corresponding Store-Exclusive instruction.

To perform an exclusive read-modify-write of a memory location, software must:

1. Use a Load-Exclusive instruction to read the value of the location.
2. Modify the value, as required.
3. Use a Store-Exclusive instruction to attempt to write the new value back to the memory location.
4. Test the returned status bit.

   If the status bit is clear, the read-modify-write completed successfully. If the status bit is set, no write was performed, which indicates that the value returned at step 1 might be out of date. The software must retry the entire read-modify-write sequence.

Software can use the synchronization primitives to implement a semaphore as follows:

1. Use a Load-Exclusive instruction to read from the semaphore address to check whether the semaphore is free.
2. If the semaphore is free, use a Store-Exclusive to write the claim value to the semaphore address.
3. If the returned status bit from step 2 indicates that the Store-Exclusive succeeded, then the software has claimed the semaphore. However, if the Store-Exclusive failed, another process might have claimed the semaphore after the software performed step 1.

The Cortex-M3 includes an exclusive access monitor that tags the fact that the processor has executed a Load-Exclusive instruction. The processor removes its exclusive access tag if:

- It executes a CLREX instruction.
- It executes a Store-Exclusive instruction, regardless of whether the write succeeds.
- An exception occurs, which means the processor can resolve semaphore conflicts between different threads.

For more information about the synchronization primitive instructions, see the *Cortex™-M3/M4 Instruction Set Technical User's Manual*.

2.5 Exception Model

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

Table 2-8 on page 95 lists all exception types. Software can set eight priority levels on seven of these exceptions (system handlers) as well as on 47 interrupts (listed in Table 2-9 on page 96).

Priorities on the system handlers are set with the NVIC **System Handler Priority n (SYSPRIn)** registers. Interrupts are enabled through the NVIC **Interrupt Set Enable n (ENn)** register and prioritized with the NVIC **Interrupt Priority n (PRIn)** registers. Priorities can be grouped by splitting priority levels into preemption priorities and subpriorities. All the interrupt registers are described in "Nested Vectored Interrupt Controller (NVIC)" on page 109.

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

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

See "Nested Vectored Interrupt Controller (NVIC)" on page 109 for more information on exceptions and interrupts.

2.5.1 Exception States

Each exception is in one of the following states:

- **Inactive.** The exception is not active and not pending.
- **Pending.** The exception is waiting to be serviced by the processor. An interrupt request from a peripheral or from software can change the state of the corresponding interrupt to pending.
- **Active.** An exception that is being serviced by the processor but has not completed.

  Note: An exception handler can interrupt the execution of another exception handler. In this case, both exceptions are in the active state.

- **Active and Pending.** The exception is being serviced by the processor, and there is a pending exception from the same source.

2.5.2 Exception Types

The exception types are:

- **Reset.** Reset is invoked on power up or a warm reset. The exception model treats reset as a special form of exception. When reset is asserted, the operation of the processor stops, potentially at any point in an instruction. When reset is deasserted, execution restarts from the address provided by the reset entry in the vector table. Execution restarts as privileged execution in Thread mode.
- **NMI.** A non-maskable Interrupt (NMI) can be signaled using the NMI signal or triggered by software using the **Interrupt Control and State (INTCTRL)** register. This exception has the highest priority other than reset. NMI is permanently enabled and has a fixed priority of -2. NMIs cannot be masked or prevented from activation by any other exception or preempted by any exception other than reset.
- **Hard Fault.** A hard fault is an exception that occurs because of an error during exception processing, or because an exception cannot be managed by any other exception mechanism. Hard faults have a fixed priority of -1, meaning they have higher priority than any exception with configurable priority.
- **Memory Management Fault.** A memory management fault is an exception that occurs because of a memory protection related fault, including access violation and no match. The MPU or the fixed memory protection constraints determine this fault, for both instruction and data memory transactions. This fault is used to abort instruction accesses to Execute Never (XN) memory regions, even if the MPU is disabled.
- **Bus Fault.** A bus fault is an exception that occurs because of a memory-related fault for an instruction or data memory transaction such as a prefetch fault or a memory access fault. This fault can be enabled or disabled.
- **Usage Fault.** A usage fault is an exception that occurs because of a fault related to instruction execution, such as:
  - An undefined instruction
  - An illegal unaligned access
  - Invalid state on instruction execution

– An error on exception return

An unaligned address on a word or halfword memory access or division by zero can cause a usage fault when the core is properly configured.

- **SVCall.** A supervisor call (SVC) is an exception that is triggered by the SVC instruction. In an OS environment, applications can use SVC instructions to access OS kernel functions and device drivers.
- **Debug Monitor.** This exception is caused by the debug monitor (when not halting). This exception is only active when enabled. This exception does not activate if it is a lower priority than the current activation.
- **PendSV.** PendSV is a pendable, interrupt-driven request for system-level service. In an OS environment, use PendSV for context switching when no other exception is active. PendSV is triggered using the **Interrupt Control and State (INTCTRL)** register.
- **SysTick.** A SysTick exception is an exception that the system timer generates when it reaches zero when it is enabled to generate an interrupt. Software can also generate a SysTick exception using the **Interrupt Control and State (INTCTRL)** register. In an OS environment, the processor can use this exception as system tick.
- **Interrupt (IRQ).** An interrupt, or IRQ, is an exception signaled by a peripheral or generated by a software request and fed through the NVIC (prioritized). All interrupts are asynchronous to instruction execution. In the system, peripherals use interrupts to communicate with the processor. Table 2-9 on page 96 lists the interrupts on the LM3S9U81 controller.

For an asynchronous exception, other than reset, the processor can execute another instruction between when the exception is triggered and when the processor enters the exception handler.

Privileged software can disable the exceptions that Table 2-8 on page 95 shows as having configurable priority (see the **SYSHNDCTRL** register on page 152 and the **DIS0** register on page 125).

For more information about hard faults, memory management faults, bus faults, and usage faults, see “Fault Handling” on page 101.

Table 2-8. Exception Types

Exception Type	Vector Number	Priority[a]	Vector Address or Offset[b]	Activation
-	0	-	0x0000.0000	Stack top is loaded from the first entry of the vector table on reset.
Reset	1	-3 (highest)	0x0000.0004	Asynchronous
Non-Maskable Interrupt (NMI)	2	-2	0x0000.0008	Asynchronous
Hard Fault	3	-1	0x0000.000C	-
Memory Management	4	programmable[c]	0x0000.0010	Synchronous
Bus Fault	5	programmable[c]	0x0000.0014	Synchronous when precise and asynchronous when imprecise
Usage Fault	6	programmable[c]	0x0000.0018	Synchronous
-	7-10	-	-	Reserved
SVCall	11	programmable[c]	0x0000.002C	Synchronous
Debug Monitor	12	programmable[c]	0x0000.0030	Synchronous
-	13	-	-	Reserved

Table 2-8. Exception Types *(continued)*

Exception Type	Vector Number	Priority[a]	Vector Address or Offset[b]	Activation
PendSV	14	programmable[c]	0x0000.0038	Asynchronous
SysTick	15	programmable[c]	0x0000.003C	Asynchronous
Interrupts	16 and above	programmable[d]	0x0000.0040 and above	Asynchronous

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

b. See “Vector Table” on page 97.

c. See **SYSPRI1** on page 149.

d. See **PRIn** registers on page 133.

Table 2-9. Interrupts

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Vector Address or Offset	Description
0-15	-	0x0000.0000 - 0x0000.003C	Processor exceptions
16	0	0x0000.0040	GPIO Port A
17	1	0x0000.0044	GPIO Port B
18	2	0x0000.0048	GPIO Port C
19	3	0x0000.004C	GPIO Port D
20	4	0x0000.0050	GPIO Port E
21	5	0x0000.0054	UART0
22	6	0x0000.0058	UART1
23	7	0x0000.005C	SSI0
24	8	0x0000.0060	I^2C0
25-29	9-13	-	Reserved
30	14	0x0000.0078	ADC0 Sequence 0
31	15	0x0000.007C	ADC0 Sequence 1
32	16	0x0000.0080	ADC0 Sequence 2
33	17	0x0000.0084	ADC0 Sequence 3
34	18	0x0000.0088	Watchdog Timers 0 and 1
35	19	0x0000.008C	Timer 0A
36	20	0x0000.0090	Timer 0B
37	21	0x0000.0094	Timer 1A
38	22	0x0000.0098	Timer 1B
39	23	0x0000.009C	Timer 2A
40	24	0x0000.00A0	Timer 2B
41	25	0x0000.00A4	Analog Comparator 0
42	26	0x0000.00A8	Analog Comparator 1
43	27	0x0000.00AC	Analog Comparator 2
44	28	0x0000.00B0	System Control
45	29	0x0000.00B4	Flash Memory Control
46	30	0x0000.00B8	GPIO Port F
47	31	0x0000.00BC	GPIO Port G
48	32	0x0000.00C0	GPIO Port H

Table 2-9. Interrupts *(continued)*

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Vector Address or Offset	Description
49	33	0x0000.00C4	UART2
50	34	0x0000.00C8	SSI1
51	35	0x0000.00CC	Timer 3A
52	36	0x0000.00D0	Timer 3B
53	37	0x0000.00D4	I^2C1
54	38	-	Reserved
55	39	0x0000.00DC	CAN0
56	40	0x0000.00E0	CAN1
57	41	0x0000.00E4	CAN2
58	42	0x0000.00E8	Ethernet Controller
59	43	-	Reserved
60	44	0x0000.00F0	USB
61	45	-	Reserved
62	46	0x0000.00F8	µDMA Software
63	47	0x0000.00FC	µDMA Error
64	48	0x0000.0100	ADC1 Sequence 0
65	49	0x0000.0104	ADC1 Sequence 1
66	50	0x0000.0108	ADC1 Sequence 2
67	51	0x0000.010C	ADC1 Sequence 3
68	52	0x0000.0110	I^2S0
69	53	0x0000.0114	EPI
70	54	0x0000.0118	GPIO Port J

2.5.3 Exception Handlers

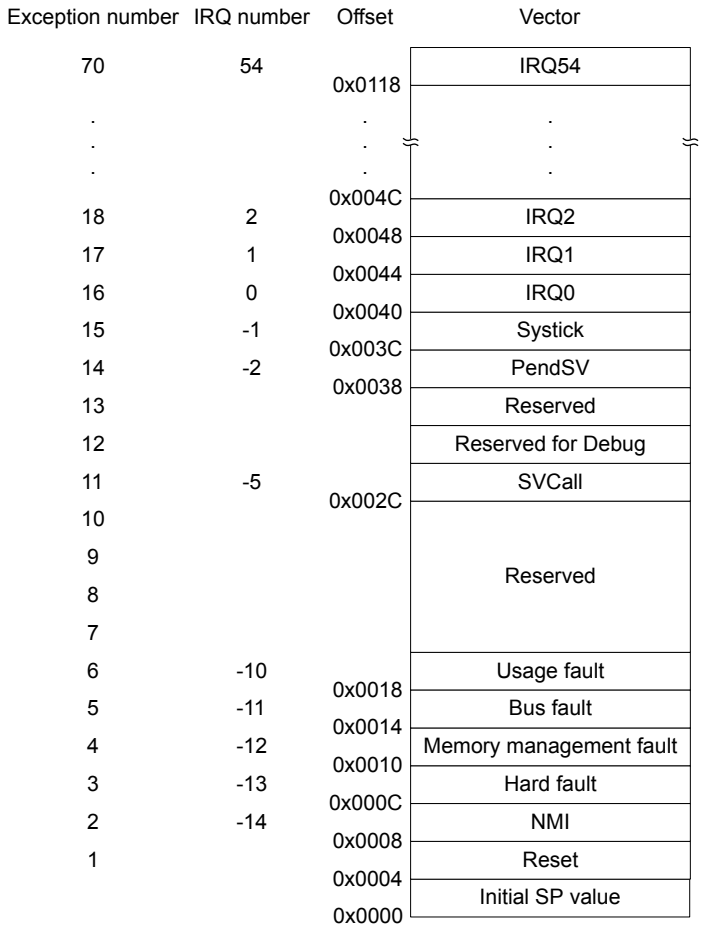
The processor handles exceptions using:

- **Interrupt Service Routines (ISRs).** Interrupts (IRQx) are the exceptions handled by ISRs.
- **Fault Handlers.** Hard fault, memory management fault, usage fault, and bus fault are fault exceptions handled by the fault handlers.
- **System Handlers.** NMI, PendSV, SVCall, SysTick, and the fault exceptions are all system exceptions that are handled by system handlers.

2.5.4 Vector Table

The vector table contains the reset value of the stack pointer and the start addresses, also called exception vectors, for all exception handlers. The vector table is constructed using the vector address or offset shown in Table 2-8 on page 95. Figure 2-6 on page 98 shows the order of the exception vectors in the vector table. The least-significant bit of each vector must be 1, indicating that the exception handler is Thumb code

Figure 2-6. Vector Table



On system reset, the vector table is fixed at address 0x0000.0000. Privileged software can write to the **Vector Table Offset (VTABLE)** register to relocate the vector table start address to a different memory location, in the range 0x0000.0200 to 0x3FFF.FE00 (see "Vector Table" on page 97). Note that when configuring the **VTABLE** register, the offset must be aligned on a 512-byte boundary.

2.5.5 Exception Priorities

As Table 2-8 on page 95 shows, all exceptions have an associated priority, with a lower priority value indicating a higher priority and configurable priorities for all exceptions except Reset, Hard fault, and NMI. If software does not configure any priorities, then all exceptions with a configurable priority have a priority of 0. For information about configuring exception priorities, see page 149 and page 133.

Note: Configurable priority values for the Stellaris implementation are in the range 0-7. This means that the Reset, Hard fault, and NMI exceptions, with fixed negative priority values, always have higher priority than any other exception.

For example, assigning a higher priority value to IRQ[0] and a lower priority value to IRQ[1] means that IRQ[1] has higher priority than IRQ[0]. If both IRQ[1] and IRQ[0] are asserted, IRQ[1] is processed before IRQ[0].

If multiple pending exceptions have the same priority, the pending exception with the lowest exception number takes precedence. For example, if both IRQ[0] and IRQ[1] are pending and have the same priority, then IRQ[0] is processed before IRQ[1].

When the processor is executing an exception handler, the exception handler is preempted if a higher priority exception occurs. If an exception occurs with the same priority as the exception being handled, the handler is not preempted, irrespective of the exception number. However, the status of the new interrupt changes to pending.

2.5.6 Interrupt Priority Grouping

To increase priority control in systems with interrupts, the NVIC supports priority grouping. This grouping divides each interrupt priority register entry into two fields:

- An upper field that defines the group priority
- A lower field that defines a subpriority within the group

Only the group priority determines preemption of interrupt exceptions. When the processor is executing an interrupt exception handler, another interrupt with the same group priority as the interrupt being handled does not preempt the handler.

If multiple pending interrupts have the same group priority, the subpriority field determines the order in which they are processed. If multiple pending interrupts have the same group priority and subpriority, the interrupt with the lowest IRQ number is processed first.

For information about splitting the interrupt priority fields into group priority and subpriority, see page 143.

2.5.7 Exception Entry and Return

Descriptions of exception handling use the following terms:

- **Preemption.** When the processor is executing an exception handler, an exception can preempt the exception handler if its priority is higher than the priority of the exception being handled. See “Interrupt Priority Grouping” on page 99 for more information about preemption by an interrupt. When one exception preempts another, the exceptions are called nested exceptions. See “Exception Entry” on page 100 more information.
- **Return.** Return occurs when the exception handler is completed, and there is no pending exception with sufficient priority to be serviced and the completed exception handler was not handling a late-arriving exception. The processor pops the stack and restores the processor state to the state it had before the interrupt occurred. See “Exception Return” on page 100 for more information.
- **Tail-Chaining.** This mechanism speeds up exception servicing. On completion of an exception handler, if there is a pending exception that meets the requirements for exception entry, the stack pop is skipped and control transfers to the new exception handler.
- **Late-Arriving.** This mechanism speeds up preemption. If a higher priority exception occurs during state saving for a previous exception, the processor switches to handle the higher priority exception and initiates the vector fetch for that exception. State saving is not affected by late arrival because the state saved is the same for both exceptions. Therefore, the state saving continues uninterrupted. The processor can accept a late arriving exception until the first instruction of the exception handler of the original exception enters the execute stage of the processor. On

return from the exception handler of the late-arriving exception, the normal tail-chaining rules apply.

2.5.7.1 Exception Entry

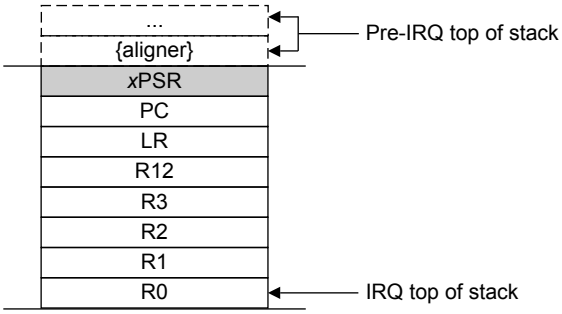
Exception entry occurs when there is a pending exception with sufficient priority and either the processor is in Thread mode or the new exception is of higher priority than the exception being handled, in which case the new exception preempts the original exception.

When one exception preempts another, the exceptions are nested.

Sufficient priority means the exception has more priority than any limits set by the mask registers (see **PRIMASK** on page 80, **FAULTMASK** on page 81, and **BASEPRI** on page 82). An exception with less priority than this is pending but is not handled by the processor.

When the processor takes an exception, unless the exception is a tail-chained or a late-arriving exception, the processor pushes information onto the current stack. This operation is referred to as *stacking* and the structure of eight data words is referred to as *stack frame*.

Figure 2-7. Exception Stack Frame



Immediately after stacking, the stack pointer indicates the lowest address in the stack frame.

The stack frame includes the return address, which is the address of the next instruction in the interrupted program. This value is restored to the **PC** at exception return so that the interrupted program resumes.

In parallel to the stacking operation, the processor performs a vector fetch that reads the exception handler start address from the vector table. When stacking is complete, the processor starts executing the exception handler. At the same time, the processor writes an EXC_RETURN value to the **LR**, indicating which stack pointer corresponds to the stack frame and what operation mode the processor was in before the entry occurred.

If no higher-priority exception occurs during exception entry, the processor starts executing the exception handler and automatically changes the status of the corresponding pending interrupt to active.

If another higher-priority exception occurs during exception entry, known as late arrival, the processor starts executing the exception handler for this exception and does not change the pending status of the earlier exception.

2.5.7.2 Exception Return

Exception return occurs when the processor is in Handler mode and executes one of the following instructions to load the EXC_RETURN value into the **PC**:

- An `LDM` or `POP` instruction that loads the **PC**

- A BX instruction using any register
- An LDR instruction with the **PC** as the destination

EXC_RETURN is the value loaded into the **LR** on exception entry. The exception mechanism relies on this value to detect when the processor has completed an exception handler. The lowest four bits of this value provide information on the return stack and processor mode. Table 2-10 on page 101 shows the EXC_RETURN values with a description of the exception return behavior.

EXC_RETURN bits 31:4 are all set. When this value is loaded into the **PC**, it indicates to the processor that the exception is complete, and the processor initiates the appropriate exception return sequence.

Table 2-10. Exception Return Behavior

EXC_RETURN[31:0]	Description
0xFFFF.FFF0	Reserved
0xFFFF.FFF1	Return to Handler mode. Exception return uses state from **MSP**. Execution uses **MSP** after return.
0xFFFF.FFF2 - 0xFFFF.FFF8	Reserved
0xFFFF.FFF9	Return to Thread mode. Exception return uses state from **MSP**. Execution uses **MSP** after return.
0xFFFF.FFFA - 0xFFFF.FFFC	Reserved
0xFFFF.FFFD	Return to Thread mode. Exception return uses state from **PSP**. Execution uses **PSP** after return.
0xFFFF.FFFE - 0xFFFF.FFFF	Reserved

2.6 Fault Handling

Faults are a subset of the exceptions (see “Exception Model” on page 93). The following conditions generate a fault:

- A bus error on an instruction fetch or vector table load or a data access.
- An internally detected error such as an undefined instruction or an attempt to change state with a BX instruction.
- Attempting to execute an instruction from a memory region marked as Non-Executable (XN).
- An MPU fault because of a privilege violation or an attempt to access an unmanaged region.

2.6.1 Fault Types

Table 2-11 on page 101 shows the types of fault, the handler used for the fault, the corresponding fault status register, and the register bit that indicates the fault has occurred. See page 156 for more information about the fault status registers.

Table 2-11. Faults

Fault	Handler	Fault Status Register	Bit Name
Bus error on a vector read	Hard fault	**Hard Fault Status (HFAULTSTAT)**	VECT

Table 2-11. Faults *(continued)*

Fault	Handler	Fault Status Register	Bit Name
Fault escalated to a hard fault	Hard fault	**Hard Fault Status (HFAULTSTAT)**	FORCED
MPU or default memory mismatch on instruction access	Memory management fault	**Memory Management Fault Status (MFAULTSTAT)**	IERR [a]
MPU or default memory mismatch on data access	Memory management fault	**Memory Management Fault Status (MFAULTSTAT)**	DERR
MPU or default memory mismatch on exception stacking	Memory management fault	**Memory Management Fault Status (MFAULTSTAT)**	MSTKE
MPU or default memory mismatch on exception unstacking	Memory management fault	**Memory Management Fault Status (MFAULTSTAT)**	MUSTKE
Bus error during exception stacking	Bus fault	**Bus Fault Status (BFAULTSTAT)**	BSTKE
Bus error during exception unstacking	Bus fault	**Bus Fault Status (BFAULTSTAT)**	BUSTKE
Bus error during instruction prefetch	Bus fault	**Bus Fault Status (BFAULTSTAT)**	IBUS
Precise data bus error	Bus fault	**Bus Fault Status (BFAULTSTAT)**	PRECISE
Imprecise data bus error	Bus fault	**Bus Fault Status (BFAULTSTAT)**	IMPRE
Attempt to access a coprocessor	Usage fault	**Usage Fault Status (UFAULTSTAT)**	NOCP
Undefined instruction	Usage fault	**Usage Fault Status (UFAULTSTAT)**	UNDEF
Attempt to enter an invalid instruction set state [b]	Usage fault	**Usage Fault Status (UFAULTSTAT)**	INVSTAT
Invalid EXC_RETURN value	Usage fault	**Usage Fault Status (UFAULTSTAT)**	INVPC
Illegal unaligned load or store	Usage fault	**Usage Fault Status (UFAULTSTAT)**	UNALIGN
Divide by 0	Usage fault	**Usage Fault Status (UFAULTSTAT)**	DIV0

a. Occurs on an access to an XN region even if the MPU is disabled.

b. Attempting to use an instruction set other than the Thumb instruction set, or returning to a non load-store-multiple instruction with ICI continuation.

2.6.2 Fault Escalation and Hard Faults

All fault exceptions except for hard fault have configurable exception priority (see **SYSPRI1** on page 149). Software can disable execution of the handlers for these faults (see **SYSHNDCTRL** on page 152).

Usually, the exception priority, together with the values of the exception mask registers, determines whether the processor enters the fault handler, and whether a fault handler can preempt another fault handler as described in “Exception Model” on page 93.

In some situations, a fault with configurable priority is treated as a hard fault. This process is called priority escalation, and the fault is described as *escalated to hard fault*. Escalation to hard fault occurs when:

- A fault handler causes the same kind of fault as the one it is servicing. This escalation to hard fault occurs because a fault handler cannot preempt itself because it must have the same priority as the current priority level.
- A fault handler causes a fault with the same or lower priority as the fault it is servicing. This situation happens because the handler for the new fault cannot preempt the currently executing fault handler.
- An exception handler causes a fault for which the priority is the same as or lower than the currently executing exception.

- A fault occurs and the handler for that fault is not enabled.

If a bus fault occurs during a stack push when entering a bus fault handler, the bus fault does not escalate to a hard fault. Thus if a corrupted stack causes a fault, the fault handler executes even though the stack push for the handler failed. The fault handler operates but the stack contents are corrupted.

Note: Only Reset and NMI can preempt the fixed priority hard fault. A hard fault can preempt any exception other than Reset, NMI, or another hard fault.

2.6.3 Fault Status Registers and Fault Address Registers

The fault status registers indicate the cause of a fault. For bus faults and memory management faults, the fault address register indicates the address accessed by the operation that caused the fault, as shown in Table 2-12 on page 103.

Table 2-12. Fault Status and Fault Address Registers

Handler	Status Register Name	Address Register Name	Register Description
Hard fault	**Hard Fault Status (HFAULTSTAT)**	-	page 162
Memory management fault	**Memory Management Fault Status (MFAULTSTAT)**	**Memory Management Fault Address (MMADDR)**	page 156 page 163
Bus fault	**Bus Fault Status (BFAULTSTAT)**	**Bus Fault Address (FAULTADDR)**	page 156 page 164
Usage fault	**Usage Fault Status (UFAULTSTAT)**	-	page 156

2.6.4 Lockup

The processor enters a lockup state if a hard fault occurs when executing the NMI or hard fault handlers. When the processor is in the lockup state, it does not execute any instructions. The processor remains in lockup state until it is reset, an NMI occurs, or it is halted by a debugger.

Note: If the lockup state occurs from the NMI handler, a subsequent NMI does not cause the processor to leave the lockup state.

2.7 Power Management

The Cortex-M3 processor sleep modes reduce power consumption:

- Sleep mode stops the processor clock.
- Deep-sleep mode stops the system clock and switches off the PLL and Flash memory.

The SLEEPDEEP bit of the **System Control (SYSCTRL)** register selects which sleep mode is used (see page 145). For more information about the behavior of the sleep modes, see “System Control” on page 199.

This section describes the mechanisms for entering sleep mode and the conditions for waking up from sleep mode, both of which apply to Sleep mode and Deep-sleep mode.

2.7.1 Entering Sleep Modes

This section describes the mechanisms software can use to put the processor into one of the sleep modes.

The system can generate spurious wake-up events, for example a debug operation wakes up the processor. Therefore, software must be able to put the processor back into sleep mode after such an event. A program might have an idle loop to put the processor back to sleep mode.

2.7.1.1 Wait for Interrupt

The wait for interrupt instruction, WFI, causes immediate entry to sleep mode unless the wake-up condition is true (see “Wake Up from WFI or Sleep-on-Exit” on page 104). When the processor executes a WFI instruction, it stops executing instructions and enters sleep mode. See the *Cortex™-M3/M4 Instruction Set Technical User's Manual* for more information.

2.7.1.2 Wait for Event

The wait for event instruction, WFE, causes entry to sleep mode conditional on the value of a one-bit event register. When the processor executes a WFE instruction, it checks the event register. If the register is 0, the processor stops executing instructions and enters sleep mode. If the register is 1, the processor clears the register and continues executing instructions without entering sleep mode.

If the event register is 1, the processor must not enter sleep mode on execution of a WFE instruction. Typically, this situation occurs if an SEV instruction has been executed. Software cannot access this register directly.

See the *Cortex™-M3/M4 Instruction Set Technical User's Manual* for more information.

2.7.1.3 Sleep-on-Exit

If the SLEEPEXIT bit of the **SYSCTRL** register is set, when the processor completes the execution of all exception handlers, it returns to Thread mode and immediately enters sleep mode. This mechanism can be used in applications that only require the processor to run when an exception occurs.

2.7.2 Wake Up from Sleep Mode

The conditions for the processor to wake up depend on the mechanism that cause it to enter sleep mode.

2.7.2.1 Wake Up from WFI or Sleep-on-Exit

Normally, the processor wakes up only when the NVIC detects an exception with sufficient priority to cause exception entry. Some embedded systems might have to execute system restore tasks after the processor wakes up and before executing an interrupt handler. Entry to the interrupt handler can be delayed by setting the PRIMASK bit and clearing the FAULTMASK bit. If an interrupt arrives that is enabled and has a higher priority than current exception priority, the processor wakes up but does not execute the interrupt handler until the processor clears PRIMASK. For more information about **PRIMASK** and **FAULTMASK**, see page 80 and page 81.

2.7.2.2 Wake Up from WFE

The processor wakes up if it detects an exception with sufficient priority to cause exception entry.

In addition, if the SEVONPEND bit in the **SYSCTRL** register is set, any new pending interrupt triggers an event and wakes up the processor, even if the interrupt is disabled or has insufficient priority to cause exception entry. For more information about **SYSCTRL**, see page 145.

2.8 Instruction Set Summary

The processor implements a version of the Thumb instruction set. Table 2-13 on page 105 lists the supported instructions.

Note: In Table 2-13 on page 105:

- Angle brackets, <>, enclose alternative forms of the operand
- Braces, {}, enclose optional operands
- The Operands column is not exhaustive
- Op2 is a flexible second operand that can be either a register or a constant
- Most instructions can use an optional condition code suffix

For more information on the instructions and operands, see the instruction descriptions in the *Cortex™-M3/M4 Instruction Set Technical User's Manual*.

Table 2-13. Cortex-M3 Instruction Summary

Mnemonic	Operands	Brief Description	Flags
ADC, ADCS	{Rd,} Rn, Op2	Add with carry	N,Z,C,V
ADD, ADDS	{Rd,} Rn, Op2	Add	N,Z,C,V
ADD, ADDW	{Rd,} Rn , #imm12	Add	N,Z,C,V
ADR	Rd, label	Load PC-relative address	-
AND, ANDS	{Rd,} Rn, Op2	Logical AND	N,Z,C
ASR, ASRS	Rd, Rm, <Rs|#n>	Arithmetic shift right	N,Z,C
B	label	Branch	-
BFC	Rd, #lsb, #width	Bit field clear	-
BFI	Rd, Rn, #lsb, #width	Bit field insert	-
BIC, BICS	{Rd,} Rn, Op2	Bit clear	N,Z,C
BKPT	#imm	Breakpoint	-
BL	label	Branch with link	-
BLX	Rm	Branch indirect with link	-
BX	Rm	Branch indirect	-
CBNZ	Rn, label	Compare and branch if non-zero	-
CBZ	Rn, label	Compare and branch if zero	-
CLREX	-	Clear exclusive	-
CLZ	Rd, Rm	Count leading zeros	-
CMN	Rn, Op2	Compare negative	N,Z,C,V
CMP	Rn, Op2	Compare	N,Z,C,V
CPSID	i	Change processor state, disable interrupts	-
CPSIE	i	Change processor state, enable interrupts	-
DMB	-	Data memory barrier	-
DSB	-	Data synchronization barrier	-
EOR, EORS	{Rd,} Rn, Op2	Exclusive OR	N,Z,C
ISB	-	Instruction synchronization barrier	-
IT	-	If-Then condition block	-

Table 2-13. Cortex-M3 Instruction Summary *(continued)*

Mnemonic	Operands	Brief Description	Flags
LDM	Rn{!}, reglist	Load multiple registers, increment after	-
LDMDB, LDMEA	Rn{!}, reglist	Load multiple registers, decrement before	-
LDMFD, LDMIA	Rn{!}, reglist	Load multiple registers, increment after	-
LDR	Rt, [Rn, #offset]	Load register with word	-
LDRB, LDRBT	Rt, [Rn, #offset]	Load register with byte	-
LDRD	Rt, Rt2, [Rn, #offset]	Load register with two bytes	-
LDREX	Rt, [Rn, #offset]	Load register exclusive	-
LDREXB	Rt, [Rn]	Load register exclusive with byte	-
LDREXH	Rt, [Rn]	Load register exclusive with halfword	-
LDRH, LDRHT	Rt, [Rn, #offset]	Load register with halfword	-
LDRSB, LDRSBT	Rt, [Rn, #offset]	Load register with signed byte	-
LDRSH, LDRSHT	Rt, [Rn, #offset]	Load register with signed halfword	-
LDRT	Rt, [Rn, #offset]	Load register with word	-
LSL, LSLS	Rd, Rm, <Rs|#n>	Logical shift left	N,Z,C
LSR, LSRS	Rd, Rm, <Rs|#n>	Logical shift right	N,Z,C
MLA	Rd, Rn, Rm, Ra	Multiply with accumulate, 32-bit result	-
MLS	Rd, Rn, Rm, Ra	Multiply and subtract, 32-bit result	-
MOV, MOVS	Rd, Op2	Move	N,Z,C
MOV, MOVW	Rd, #imm16	Move 16-bit constant	N,Z,C
MOVT	Rd, #imm16	Move top	-
MRS	Rd, spec_reg	Move from special register to general register	-
MSR	spec_reg, Rm	Move from general register to special register	N,Z,C,V
MUL, MULS	{Rd,} Rn, Rm	Multiply, 32-bit result	N,Z
MVN, MVNS	Rd, Op2	Move NOT	N,Z,C
NOP	-	No operation	-
ORN, ORNS	{Rd,} Rn, Op2	Logical OR NOT	N,Z,C
ORR, ORRS	{Rd,} Rn, Op2	Logical OR	N,Z,C
POP	reglist	Pop registers from stack	-
PUSH	reglist	Push registers onto stack	-
RBIT	Rd, Rn	Reverse bits	-
REV	Rd, Rn	Reverse byte order in a word	-
REV16	Rd, Rn	Reverse byte order in each halfword	-
REVSH	Rd, Rn	Reverse byte order in bottom halfword and sign extend	-
ROR, RORS	Rd, Rm, <Rs|#n>	Rotate right	N,Z,C
RRX, RRXS	Rd, Rm	Rotate right with extend	N,Z,C
RSB, RSBS	{Rd,} Rn, Op2	Reverse subtract	N,Z,C,V
SBC, SBCS	{Rd,} Rn, Op2	Subtract with carry	N,Z,C,V
SBFX	Rd, Rn, #lsb, #width	Signed bit field extract	-

Table 2-13. Cortex-M3 Instruction Summary *(continued)*

Mnemonic	Operands	Brief Description	Flags
SDIV	{Rd,} Rn, Rm	Signed divide	-
SEV	-	Send event	-
SMLAL	RdLo, RdHi, Rn, Rm	Signed multiply with accumulate (32x32+64), 64-bit result	-
SMULL	RdLo, RdHi, Rn, Rm	Signed multiply (32x32), 64-bit result	-
SSAT	Rd, #n, Rm {,shift #s}	Signed saturate	Q
STM	Rn{!}, reglist	Store multiple registers, increment after	-
STMDB, STMEA	Rn{!}, reglist	Store multiple registers, decrement before	-
STMFD, STMIA	Rn{!}, reglist	Store multiple registers, increment after	-
STR	Rt, [Rn {, #offset}]	Store register word	-
STRB, STRBT	Rt, [Rn {, #offset}]	Store register byte	-
STRD	Rt, Rt2, [Rn {, #offset}]	Store register two words	-
STREX	Rt, Rt, [Rn {, #offset}]	Store register exclusive	-
STREXB	Rd, Rt, [Rn]	Store register exclusive byte	-
STREXH	Rd, Rt, [Rn]	Store register exclusive halfword	-
STRH, STRHT	Rt, [Rn {, #offset}]	Store register halfword	-
STRSB, STRSBT	Rt, [Rn {, #offset}]	Store register signed byte	-
STRSH, STRSHT	Rt, [Rn {, #offset}]	Store register signed halfword	-
STRT	Rt, [Rn {, #offset}]	Store register word	-
SUB, SUBS	{Rd,} Rn, Op2	Subtract	N,Z,C,V
SUB, SUBW	{Rd,} Rn, #imm12	Subtract 12-bit constant	N,Z,C,V
SVC	#imm	Supervisor call	-
SXTB	{Rd,} Rm {,ROR #n}	Sign extend a byte	-
SXTH	{Rd,} Rm {,ROR #n}	Sign extend a halfword	-
TBB	[Rn, Rm]	Table branch byte	-
TBH	[Rn, Rm, LSL #1]	Table branch halfword	-
TEQ	Rn, Op2	Test equivalence	N,Z,C
TST	Rn, Op2	Test	N,Z,C
UBFX	Rd, Rn, #lsb, #width	Unsigned bit field extract	-
UDIV	{Rd,} Rn, Rm	Unsigned divide	-
UMLAL	RdLo, RdHi, Rn, Rm	Unsigned multiply with accumulate (32x32+32+32), 64-bit result	-
UMULL	RdLo, RdHi, Rn, Rm	Unsigned multiply (32x 2), 64-bit result	-
USAT	Rd, #n, Rm {,shift #s}	Unsigned Saturate	Q
UXTB	{Rd,} Rm, {,ROR #n}	Zero extend a Byte	-
UXTH	{Rd,} Rm, {,ROR #n}	Zero extend a Halfword	-
WFE	-	Wait for event	-
WFI	-	Wait for interrupt	-

3 Cortex-M3 Peripherals

This chapter provides information on the Stellaris® implementation of the Cortex-M3 processor peripherals, including:

- SysTick (see page 108)

  Provides a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism.

- Nested Vectored Interrupt Controller (NVIC) (see page 109)
  - Facilitates low-latency exception and interrupt handling
  - Controls power management
  - Implements system control registers

- System Control Block (SCB) (see page 111)

  Provides system implementation information and system control, including configuration, control, and reporting of system exceptions.

- Memory Protection Unit (MPU) (see page 111)

  Supports the standard ARMv7 Protected Memory System Architecture (PMSA) model. The MPU provides full support for protection regions, overlapping protection regions, access permissions, and exporting memory attributes to the system.

Table 3-1 on page 108 shows the address map of the Private Peripheral Bus (PPB). Some peripheral register regions are split into two address regions, as indicated by two addresses listed.

Table 3-1. Core Peripheral Register Regions

Address	Core Peripheral	Description (see page ...)
0xE000.E010-0xE000.E01F	System Timer	108
0xE000.E100-0xE000.E4EF 0xE000.EF00-0xE000.EF03	Nested Vectored Interrupt Controller	109
0xE000.E008-0xE000.E00F 0xE000.ED00-0xE000.ED3F	System Control Block	111
0xE000.ED90-0xE000.EDB8	Memory Protection Unit	111

3.1 Functional Description

This chapter provides information on the Stellaris implementation of the Cortex-M3 processor peripherals: SysTick, NVIC, SCB and MPU.

3.1.1 System Timer (SysTick)

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

- An RTOS tick timer that fires at a programmable rate (for example, 100 Hz) and invokes a SysTick routine.

- A high-speed alarm timer using the system clock.

- A variable rate alarm or signal timer—the duration is range-dependent on the reference clock used and the dynamic range of the counter.
- A simple counter used to measure time to completion and time used.
- An internal clock source control based on missing/meeting durations. The COUNT bit in the **STCTRL** control and status register can be used to determine if an action completed within a set duration, as part of a dynamic clock management control loop.

The timer consists of three registers:

- **SysTick Control and Status (STCTRL)**: A control and status counter to configure its clock, enable the counter, enable the SysTick interrupt, and determine counter status.
- **SysTick Reload Value (STRELOAD)**: The reload value for the counter, used to provide the counter's wrap value.
- **SysTick Current Value (STCURRENT)**: The current value of the counter.

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

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

The SysTick counter runs on the system clock. If this clock signal is stopped for low power mode, the SysTick counter stops. Ensure software uses aligned word accesses to access the SysTick registers.

Note: When the processor is halted for debugging, the counter does not decrement.

3.1.2 Nested Vectored Interrupt Controller (NVIC)

This section describes the Nested Vectored Interrupt Controller (NVIC) and the registers it uses. The NVIC supports:

- 47 interrupts.
- A programmable priority level of 0-7 for each interrupt. A higher level corresponds to a lower priority, so level 0 is the highest interrupt priority.
- Low-latency exception and interrupt handling.
- Level and pulse detection of interrupt signals.
- Dynamic reprioritization of interrupts.
- Grouping of priority values into group priority and subpriority fields.
- Interrupt tail-chaining.
- An external Non-maskable interrupt (NMI).

The processor automatically stacks its state on exception entry and unstacks this state on exception exit, with no instruction overhead, providing low latency exception handling.

3.1.2.1 Level-Sensitive and Pulse Interrupts

The processor supports both level-sensitive and pulse interrupts. Pulse interrupts are also described as edge-triggered interrupts.

A level-sensitive interrupt is held asserted until the peripheral deasserts the interrupt signal. Typically this happens because the ISR accesses the peripheral, causing it to clear the interrupt request. A pulse interrupt is an interrupt signal sampled synchronously on the rising edge of the processor clock. To ensure the NVIC detects the interrupt, the peripheral must assert the interrupt signal for at least one clock cycle, during which the NVIC detects the pulse and latches the interrupt.

When the processor enters the ISR, it automatically removes the pending state from the interrupt (see “Hardware and Software Control of Interrupts” on page 110 for more information). For a level-sensitive interrupt, if the signal is not deasserted before the processor returns from the ISR, the interrupt becomes pending again, and the processor must execute its ISR again. As a result, the peripheral can hold the interrupt signal asserted until it no longer needs servicing.

3.1.2.2 Hardware and Software Control of Interrupts

The Cortex-M3 latches all interrupts. A peripheral interrupt becomes pending for one of the following reasons:

- The NVIC detects that the interrupt signal is High and the interrupt is not active.
- The NVIC detects a rising edge on the interrupt signal.
- Software writes to the corresponding interrupt set-pending register bit, or to the **Software Trigger Interrupt (SWTRIG)** register to make a Software-Generated Interrupt pending. See the `INT` bit in the **PEND0** register on page 127 or **SWTRIG** on page 135.

A pending interrupt remains pending until one of the following:

- The processor enters the ISR for the interrupt, changing the state of the interrupt from pending to active. Then:
  - For a level-sensitive interrupt, when the processor returns from the ISR, the NVIC samples the interrupt signal. If the signal is asserted, the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR. Otherwise, the state of the interrupt changes to inactive.
  - For a pulse interrupt, the NVIC continues to monitor the interrupt signal, and if this is pulsed the state of the interrupt changes to pending and active. In this case, when the processor returns from the ISR the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR.

    If the interrupt signal is not pulsed while the processor is in the ISR, when the processor returns from the ISR the state of the interrupt changes to inactive.
- Software writes to the corresponding interrupt clear-pending register bit
  - For a level-sensitive interrupt, if the interrupt signal is still asserted, the state of the interrupt does not change. Otherwise, the state of the interrupt changes to inactive.

– For a pulse interrupt, the state of the interrupt changes to inactive, if the state was pending or to active, if the state was active and pending.

3.1.3 System Control Block (SCB)

The System Control Block (SCB) provides system implementation information and system control, including configuration, control, and reporting of the system exceptions.

3.1.4 Memory Protection Unit (MPU)

This section describes the Memory protection unit (MPU). The MPU divides the memory map into a number of regions and defines the location, size, access permissions, and memory attributes of each region. The MPU supports independent attribute settings for each region, overlapping regions, and export of memory attributes to the system.

The memory attributes affect the behavior of memory accesses to the region. The Cortex-M3 MPU defines eight separate memory regions, 0-7, and a background region.

When memory regions overlap, a memory access is affected by the attributes of the region with the highest number. For example, the attributes for region 7 take precedence over the attributes of any region that overlaps region 7.

The background region has the same memory access attributes as the default memory map, but is accessible from privileged software only.

The Cortex-M3 MPU memory map is unified, meaning that instruction accesses and data accesses have the same region settings.

If a program accesses a memory location that is prohibited by the MPU, the processor generates a memory management fault, causing a fault exception and possibly causing termination of the process in an OS environment. In an OS environment, the kernel can update the MPU region setting dynamically based on the process to be executed. Typically, an embedded OS uses the MPU for memory protection.

Configuration of MPU regions is based on memory types (see “Memory Regions, Types and Attributes” on page 86 for more information).

Table 3-2 on page 111 shows the possible MPU region attributes. See the section called “MPU Configuration for a Stellaris Microcontroller” on page 115 for guidelines for programming a microcontroller implementation.

Table 3-2. Memory Attributes Summary

Memory Type	Description
Strongly Ordered	All accesses to Strongly Ordered memory occur in program order.
Device	Memory-mapped peripherals
Normal	Normal memory

To avoid unexpected behavior, disable the interrupts before updating the attributes of a region that the interrupt handlers might access.

Ensure software uses aligned accesses of the correct size to access MPU registers:

- Except for the **MPU Region Attribute and Size (MPUATTR)** register, all MPU registers must be accessed with aligned word accesses.
- The **MPUATTR** register can be accessed with byte or aligned halfword or word accesses.

The processor does not support unaligned accesses to MPU registers.

When setting up the MPU, and if the MPU has previously been programmed, disable unused regions to prevent any previous region settings from affecting the new MPU setup.

3.1.4.1 Updating an MPU Region

To update the attributes for an MPU region, the **MPU Region Number (MPUNUMBER)**, **MPU Region Base Address (MPUBASE)** and **MPUATTR** registers must be updated. Each register can be programmed separately or with a multiple-word write to program all of these registers. You can use the **MPUBASEx** and **MPUATTRx** aliases to program up to four regions simultaneously using an STM instruction.

Updating an MPU Region Using Separate Words

This example simple code configures one region:

```
; R1 = region number
; R2 = size/enable
; R3 = attributes
; R4 = address
LDR R0,=MPUNUMBER        ; 0xE000ED98, MPU region number register
STR R1, [R0, #0x0]       ; Region Number
STR R4, [R0, #0x4]       ; Region Base Address
STRH R2, [R0, #0x8]      ; Region Size and Enable
STRH R3, [R0, #0xA]      ; Region Attribute
```

Disable a region before writing new region settings to the MPU if you have previously enabled the region being changed. For example:

```
; R1 = region number
; R2 = size/enable
; R3 = attributes
; R4 = address
LDR R0,=MPUNUMBER         ; 0xE000ED98, MPU region number register
STR R1, [R0, #0x0]        ; Region Number
BIC R2, R2, #1            ; Disable
STRH R2, [R0, #0x8]       ; Region Size and Enable
STR R4, [R0, #0x4]        ; Region Base Address
STRH R3, [R0, #0xA]       ; Region Attribute
ORR R2, #1                ; Enable
STRH R2, [R0, #0x8]       ; Region Size and Enable
```

Software must use memory barrier instructions:

- Before MPU setup, if there might be outstanding memory transfers, such as buffered writes, that might be affected by the change in MPU settings.
- After MPU setup, if it includes memory transfers that must use the new MPU settings.

However, memory barrier instructions are not required if the MPU setup process starts by entering an exception handler, or is followed by an exception return, because the exception entry and exception return mechanism cause memory barrier behavior.

Software does not need any memory barrier instructions during MPU setup, because it accesses the MPU through the Private Peripheral Bus (PPB), which is a Strongly Ordered memory region.

For example, if all of the memory access behavior is intended to take effect immediately after the programming sequence, then a DSB instruction and an ISB instruction should be used. A DSB is required after changing MPU settings, such as at the end of context switch. An ISB is required if the code that programs the MPU region or regions is entered using a branch or call. If the programming sequence is entered using a return from exception, or by taking an exception, then an ISB is not required.

Updating an MPU Region Using Multi-Word Writes

The MPU can be programmed directly using multi-word writes, depending how the information is divided. Consider the following reprogramming:

```
; R1 = region number
; R2 = address
; R3 = size, attributes in one
LDR R0, =MPUNUMBER  ; 0xE000ED98, MPU region number register
STR R1, [R0, #0x0]  ; Region Number
STR R2, [R0, #0x4]  ; Region Base Address
STR R3, [R0, #0x8]  ; Region Attribute, Size and Enable
```

An STM instruction can be used to optimize this:

```
; R1 = region number
; R2 = address
; R3 = size, attributes in one
LDR R0, =MPUNUMBER  ; 0xE000ED98, MPU region number register
STM R0, {R1-R3}     ; Region number, address, attribute, size and enable
```

This operation can be done in two words for pre-packed information, meaning that the **MPU Region Base Address (MPUBASE)** register (see page 169) contains the required region number and has the VALID bit set. This method can be used when the data is statically packed, for example in a boot loader:

```
; R1 = address and region number in one
; R2 = size and attributes in one
LDR R0, =MPUBASE    ; 0xE000ED9C, MPU Region Base register
STR R1, [R0, #0x0]  ; Region base address and region number combined
                    ; with VALID (bit 4) set
STR R2, [R0, #0x4]  ; Region Attribute, Size and Enable
```

Subregions

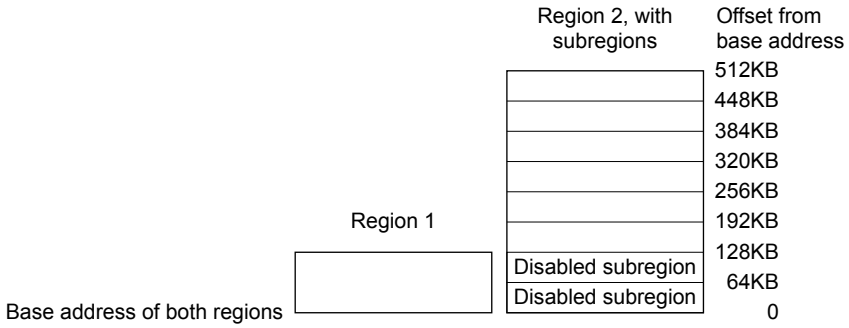
Regions of 256 bytes or more are divided into eight equal-sized subregions. Set the corresponding bit in the SRD field of the **MPU Region Attribute and Size (MPUATTR)** register (see page 171) to disable a subregion. The least-significant bit of the SRD field controls the first subregion, and the most-significant bit controls the last subregion. Disabling a subregion means another region overlapping the disabled range matches instead. If no other enabled region overlaps the disabled subregion, the MPU issues a fault.

Regions of 32, 64, and 128 bytes do not support subregions. With regions of these sizes, the SRD field must be configured to 0x00, otherwise the MPU behavior is unpredictable.

Example of SRD Use

Two regions with the same base address overlap. Region one is 128 KB, and region two is 512 KB. To ensure the attributes from region one apply to the first 128 KB region, configure the SRD field for region two to 0x03 to disable the first two subregions, as Figure 3-1 on page 114 shows.

Figure 3-1. SRD Use Example



3.1.4.2 MPU Access Permission Attributes

The access permission bits, TEX, S, C, B, AP, and XN of the **MPUATTR** register, control access to the corresponding memory region. If an access is made to an area of memory without the required permissions, then the MPU generates a permission fault.

Table 3-3 on page 114 shows the encodings for the TEX, C, B, and S access permission bits. All encodings are shown for completeness, however the current implementation of the Cortex-M3 does not support the concept of cacheability or shareability. Refer to the section called “MPU Configuration for a Stellaris Microcontroller” on page 115 for information on programming the MPU for Stellaris implementations.

Table 3-3. TEX, S, C, and B Bit Field Encoding

TEX	S	C	B	Memory Type	Shareability	Other Attributes
000b	x[a]	0	0	Strongly Ordered	Shareable	-
000	x[a]	0	1	Device	Shareable	-
000	0	1	0	Normal	Not shareable	Outer and inner write-through. No write allocate.
000	1	1	0	Normal	Shareable	
000	0	1	1	Normal	Not shareable	
000	1	1	1	Normal	Shareable	
001	0	0	0	Normal	Not shareable	Outer and inner noncacheable.
001	1	0	0	Normal	Shareable	
001	x[a]	0	1	Reserved encoding	-	-
001	x[a]	1	0	Reserved encoding	-	-
001	0	1	1	Normal	Not shareable	Outer and inner write-back. Write and read allocate.
001	1	1	1	Normal	Shareable	
010	x[a]	0	0	Device	Not shareable	Nonshared Device.
010	x[a]	0	1	Reserved encoding	-	-
010	x[a]	1	x[a]	Reserved encoding	-	-

Table 3-3. TEX, S, C, and B Bit Field Encoding *(continued)*

TEX	S	C	B	Memory Type	Shareability	Other Attributes
1BB	0	A	A	Normal	Not shareable	Cached memory (BB = outer policy, AA = inner policy). See Table 3-4 for the encoding of the AA and BB bits.
1BB	1	A	A	Normal	Shareable	

a. The MPU ignores the value of this bit.

Table 3-4 on page 115 shows the cache policy for memory attribute encodings with a TEX value in the range of 0x4-0x7.

Table 3-4. Cache Policy for Memory Attribute Encoding

Encoding, AA or BB	Corresponding Cache Policy
00	Non-cacheable
01	Write back, write and read allocate
10	Write through, no write allocate
11	Write back, no write allocate

Table 3-5 on page 115 shows the AP encodings in the **MPUATTR** register that define the access permissions for privileged and unprivileged software.

Table 3-5. AP Bit Field Encoding

AP Bit Field	Privileged Permissions	Unprivileged Permissions	Description
000	No access	No access	All accesses generate a permission fault.
001	R/W	No access	Access from privileged software only.
010	R/W	RO	Writes by unprivileged software generate a permission fault.
011	R/W	R/W	Full access.
100	Unpredictable	Unpredictable	Reserved.
101	RO	No access	Reads by privileged software only.
110	RO	RO	Read-only, by privileged or unprivileged software.
111	RO	RO	Read-only, by privileged or unprivileged software.

MPU Configuration for a Stellaris Microcontroller

Stellaris microcontrollers have only a single processor and no caches. As a result, the MPU should be programmed as shown in Table 3-6 on page 115.

Table 3-6. Memory Region Attributes for Stellaris Microcontrollers

Memory Region	TEX	S	C	B	Memory Type and Attributes
Flash memory	000b	0	1	0	Normal memory, non-shareable, write-through
Internal SRAM	000b	1	1	0	Normal memory, shareable, write-through
External SRAM	000b	1	1	1	Normal memory, shareable, write-back, write-allocate
Peripherals	000b	1	0	1	Device memory, shareable

In current Stellaris microcontroller implementations, the shareability and cache policy attributes do not affect the system behavior. However, using these settings for the MPU regions can make the application code more portable. The values given are for typical situations.

3.1.4.3 MPU Mismatch

When an access violates the MPU permissions, the processor generates a memory management fault (see “Exceptions and Interrupts” on page 84 for more information). The **MFAULTSTAT** register indicates the cause of the fault. See page 156 for more information.

3.2 Register Map

Table 3-7 on page 116 lists the Cortex-M3 Peripheral SysTick, NVIC, MPU and SCB registers. The offset listed is a hexadecimal increment to the register's address, relative to the Core Peripherals base address of 0xE000.E000.

Note: Register spaces that are not used are reserved for future or internal use. Software should not modify any reserved memory address.

Table 3-7. Peripherals Register Map

Offset	Name	Type	Reset	Description	See page
System Timer (SysTick) Registers					
0x010	STCTRL	R/W	0x0000.0004	SysTick Control and Status Register	119
0x014	STRELOAD	R/W	0x0000.0000	SysTick Reload Value Register	121
0x018	STCURRENT	R/WC	0x0000.0000	SysTick Current Value Register	122
Nested Vectored Interrupt Controller (NVIC) Registers					
0x100	EN0	R/W	0x0000.0000	Interrupt 0-31 Set Enable	123
0x104	EN1	R/W	0x0000.0000	Interrupt 32-54 Set Enable	124
0x180	DIS0	R/W	0x0000.0000	Interrupt 0-31 Clear Enable	125
0x184	DIS1	R/W	0x0000.0000	Interrupt 32-54 Clear Enable	126
0x200	PEND0	R/W	0x0000.0000	Interrupt 0-31 Set Pending	127
0x204	PEND1	R/W	0x0000.0000	Interrupt 32-54 Set Pending	128
0x280	UNPEND0	R/W	0x0000.0000	Interrupt 0-31 Clear Pending	129
0x284	UNPEND1	R/W	0x0000.0000	Interrupt 32-54 Clear Pending	130
0x300	ACTIVE0	RO	0x0000.0000	Interrupt 0-31 Active Bit	131
0x304	ACTIVE1	RO	0x0000.0000	Interrupt 32-54 Active Bit	132
0x400	PRI0	R/W	0x0000.0000	Interrupt 0-3 Priority	133
0x404	PRI1	R/W	0x0000.0000	Interrupt 4-7 Priority	133
0x408	PRI2	R/W	0x0000.0000	Interrupt 8-11 Priority	133
0x40C	PRI3	R/W	0x0000.0000	Interrupt 12-15 Priority	133
0x410	PRI4	R/W	0x0000.0000	Interrupt 16-19 Priority	133

Table 3-7. Peripherals Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x414	PRI5	R/W	0x0000.0000	Interrupt 20-23 Priority	133
0x418	PRI6	R/W	0x0000.0000	Interrupt 24-27 Priority	133
0x41C	PRI7	R/W	0x0000.0000	Interrupt 28-31 Priority	133
0x420	PRI8	R/W	0x0000.0000	Interrupt 32-35 Priority	133
0x424	PRI9	R/W	0x0000.0000	Interrupt 36-39 Priority	133
0x428	PRI10	R/W	0x0000.0000	Interrupt 40-43 Priority	133
0x42C	PRI11	R/W	0x0000.0000	Interrupt 44-47 Priority	133
0x430	PRI12	R/W	0x0000.0000	Interrupt 48-51 Priority	133
0x434	PRI13	R/W	0x0000.0000	Interrupt 52-54 Priority	133
0xF00	SWTRIG	WO	0x0000.0000	Software Trigger Interrupt	135
System Control Block (SCB) Registers					
0x008	ACTLR	R/W	0x0000.0000	Auxiliary Control	136
0xD00	CPUID	RO	0x412F.C230	CPU ID Base	138
0xD04	INTCTRL	R/W	0x0000.0000	Interrupt Control and State	139
0xD08	VTABLE	R/W	0x0000.0000	Vector Table Offset	142
0xD0C	APINT	R/W	0xFA05.0000	Application Interrupt and Reset Control	143
0xD10	SYSCTRL	R/W	0x0000.0000	System Control	145
0xD14	CFGCTRL	R/W	0x0000.0200	Configuration and Control	147
0xD18	SYSPRI1	R/W	0x0000.0000	System Handler Priority 1	149
0xD1C	SYSPRI2	R/W	0x0000.0000	System Handler Priority 2	150
0xD20	SYSPRI3	R/W	0x0000.0000	System Handler Priority 3	151
0xD24	SYSHNDCTRL	R/W	0x0000.0000	System Handler Control and State	152
0xD28	FAULTSTAT	R/W1C	0x0000.0000	Configurable Fault Status	156
0xD2C	HFAULTSTAT	R/W1C	0x0000.0000	Hard Fault Status	162
0xD34	MMADDR	R/W	-	Memory Management Fault Address	163
0xD38	FAULTADDR	R/W	-	Bus Fault Address	164
Memory Protection Unit (MPU) Registers					
0xD90	MPUTYPE	RO	0x0000.0800	MPU Type	165
0xD94	MPUCTRL	R/W	0x0000.0000	MPU Control	166
0xD98	MPUNUMBER	R/W	0x0000.0000	MPU Region Number	168
0xD9C	MPUBASE	R/W	0x0000.0000	MPU Region Base Address	169
0xDA0	MPUATTR	R/W	0x0000.0000	MPU Region Attribute and Size	171

Table 3-7. Peripherals Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0xDA4	MPUBASE1	R/W	0x0000.0000	MPU Region Base Address Alias 1	169
0xDA8	MPUATTR1	R/W	0x0000.0000	MPU Region Attribute and Size Alias 1	171
0xDAC	MPUBASE2	R/W	0x0000.0000	MPU Region Base Address Alias 2	169
0xDB0	MPUATTR2	R/W	0x0000.0000	MPU Region Attribute and Size Alias 2	171
0xDB4	MPUBASE3	R/W	0x0000.0000	MPU Region Base Address Alias 3	169
0xDB8	MPUATTR3	R/W	0x0000.0000	MPU Region Attribute and Size Alias 3	171

3.3 System Timer (SysTick) Register Descriptions

This section lists and describes the System Timer registers, in numerical order by address offset.

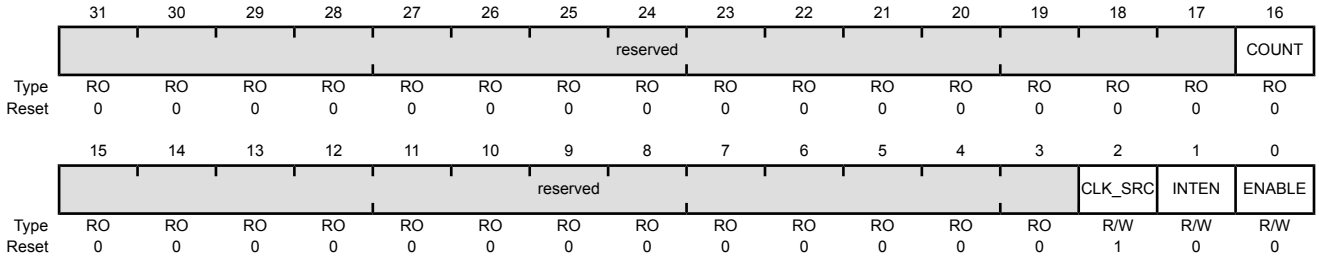
Register 1: SysTick Control and Status Register (STCTRL), offset 0x010

Note: This register can only be accessed from privileged mode.

The SysTick **STCTRL** register enables the SysTick features.

SysTick Control and Status Register (STCTRL)

Base 0xE000.E000
Offset 0x010
Type R/W, reset 0x0000.0004



Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	COUNT	RO	0	Count Flag Value Description 0 The SysTick timer has not counted to 0 since the last time this bit was read. 1 The SysTick timer has counted to 0 since the last time this bit was read. This bit is cleared by a read of the register or if the **STCURRENT** register is written with any value. If read by the debugger using the DAP, this bit is cleared only if the `MasterType` bit in the **AHB-AP Control Register** is clear. Otherwise, the `COUNT` bit is not changed by the debugger read. See the *ARM® Debug Interface V5 Architecture Specification* for more information on `MasterType`.
15:3	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	CLK_SRC	R/W	1	Clock Source Value Description 0 External reference clock. (Not implemented for most Stellaris microcontrollers.) 1 System clock Because an external reference clock is not implemented, this bit must be set in order for SysTick to operate.

Bit/Field	Name	Type	Reset	Description
1	INTEN	R/W	0	Interrupt Enable

Value	Description
0	Interrupt generation is disabled. Software can use the COUNT bit to determine if the counter has ever reached 0.
1	An interrupt is generated to the NVIC when SysTick counts to 0.

Bit/Field	Name	Type	Reset	Description
0	ENABLE	R/W	0	Enable

Value	Description
0	The counter is disabled.
1	Enables SysTick to operate in a multi-shot way. That is, the counter loads the RELOAD value and begins counting down. On reaching 0, the COUNT bit is set and an interrupt is generated if enabled by INTEN. The counter then loads the RELOAD value again and begins counting.

Register 2: SysTick Reload Value Register (STRELOAD), offset 0x014

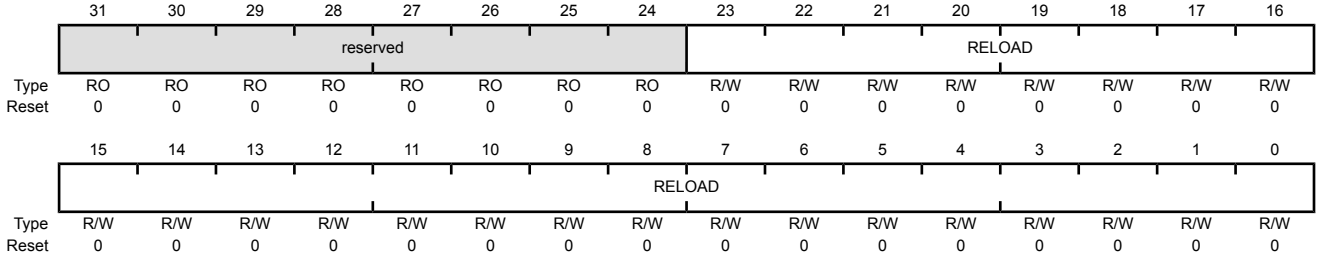
Note: This register can only be accessed from privileged mode.

The **STRELOAD** register specifies the start value to load into the **SysTick Current Value (STCURRENT)** register when the counter reaches 0. The start value can be between 0x1 and 0x00FF.FFFF. A start value of 0 is possible but has no effect because the SysTick interrupt and the COUNT bit are activated when counting from 1 to 0.

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

SysTick Reload Value Register (STRELOAD)

Base 0xE000.E000
Offset 0x014
Type R/W, reset 0x0000.0000



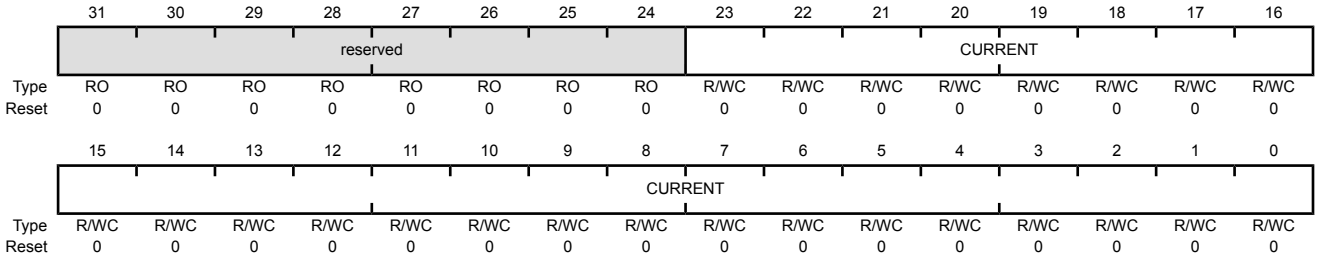
Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:0	RELOAD	R/W	0x00.0000	Reload Value Value to load into the **SysTick Current Value (STCURRENT)** register when the counter reaches 0.

Register 3: SysTick Current Value Register (STCURRENT), offset 0x018

Note: This register can only be accessed from privileged mode.

The **STCURRENT** register contains the current value of the SysTick counter.

SysTick Current Value Register (STCURRENT)
Base 0xE000.E000
Offset 0x018
Type R/WC, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:0	CURRENT	R/WC	0x00.0000	Current Value This field contains the current value at the time the register is accessed. No read-modify-write protection is provided, so change with care. This register is write-clear. Writing to it with any value clears the register. Clearing this register also clears the COUNT bit of the **STCTRL** register.

3.4 NVIC Register Descriptions

This section lists and describes the NVIC registers, in numerical order by address offset.

The NVIC registers can only be fully accessed from privileged mode, but interrupts can be pended while in unprivileged mode by enabling the **Configuration and Control (CFGCTRL)** register. Any other unprivileged mode access causes a bus fault.

Ensure software uses correctly aligned register accesses. The processor does not support unaligned accesses to NVIC registers.

An interrupt can enter the pending state even if it is disabled.

Before programming the **VTABLE** register to relocate the vector table, ensure the vector table entries of the new vector table are set up for fault handlers, NMI, and all enabled exceptions such as interrupts. For more information, see page 142.

Register 4: Interrupt 0-31 Set Enable (EN0), offset 0x100

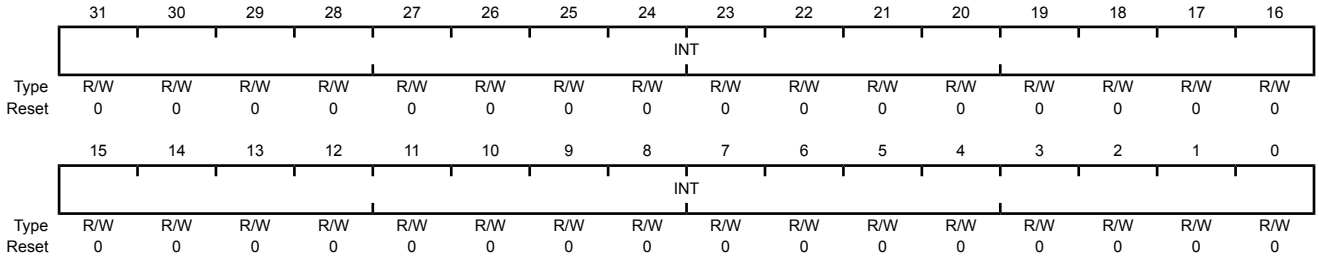
Note: This register can only be accessed from privileged mode.

See Table 2-9 on page 96 for interrupt assignments.

If a pending interrupt is enabled, the NVIC activates the interrupt based on its priority. If an interrupt is not enabled, asserting its interrupt signal changes the interrupt state to pending, but the NVIC never activates the interrupt, regardless of its priority.

Interrupt 0-31 Set Enable (EN0)

Base 0xE000.E000
Offset 0x100
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	INT	R/W	0x0000.0000	Interrupt Enable

Value	Description
0	On a read, indicates the interrupt is disabled. On a write, no effect.
1	On a read, indicates the interrupt is enabled. On a write, enables the interrupt.

A bit can only be cleared by setting the corresponding `INT[n]` bit in the **DISn** register.

Register 5: Interrupt 32-54 Set Enable (EN1), offset 0x104

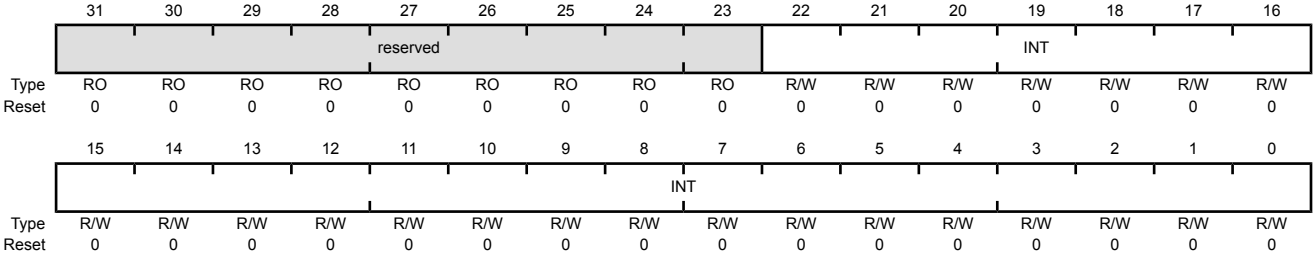
Note: This register can only be accessed from privileged mode.

The **EN1** register enables interrupts and shows which interrupts are enabled. Bit 0 corresponds to Interrupt 32; bit 22 corresponds to Interrupt 54. See Table 2-9 on page 96 for interrupt assignments.

If a pending interrupt is enabled, the NVIC activates the interrupt based on its priority. If an interrupt is not enabled, asserting its interrupt signal changes the interrupt state to pending, but the NVIC never activates the interrupt, regardless of its priority.

Interrupt 32-54 Set Enable (EN1)

Base 0xE000.E000
Offset 0x104
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:23	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22:0	INT	R/W	0x00.0000	Interrupt Enable

Value	Description
0	On a read, indicates the interrupt is disabled.
	On a write, no effect.
1	On a read, indicates the interrupt is enabled.
	On a write, enables the interrupt.

A bit can only be cleared by setting the corresponding `INT[n]` bit in the **DIS1** register.

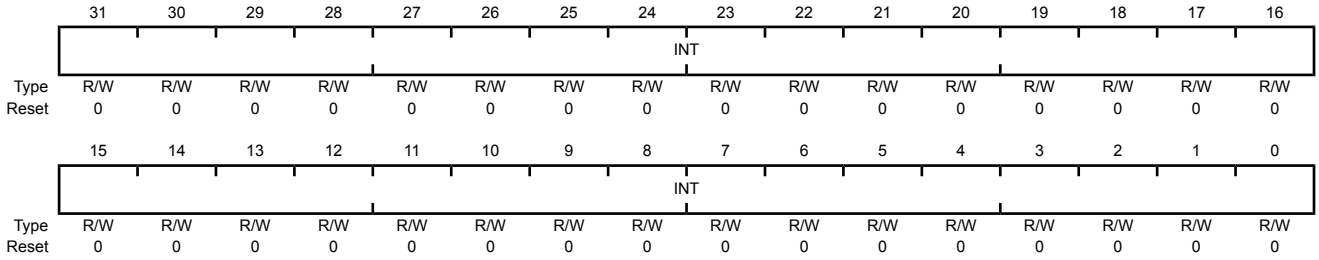
Register 6: Interrupt 0-31 Clear Enable (DIS0), offset 0x180

Note: This register can only be accessed from privileged mode.

See Table 2-9 on page 96 for interrupt assignments.

Interrupt 0-31 Clear Enable (DIS0)

Base 0xE000.E000
Offset 0x180
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	INT	R/W	0x0000.0000	Interrupt Disable

Value	Description
0	On a read, indicates the interrupt is disabled. On a write, no effect.
1	On a read, indicates the interrupt is enabled. On a write, clears the corresponding `INT[n]` bit in the **EN0** register, disabling interrupt [n].

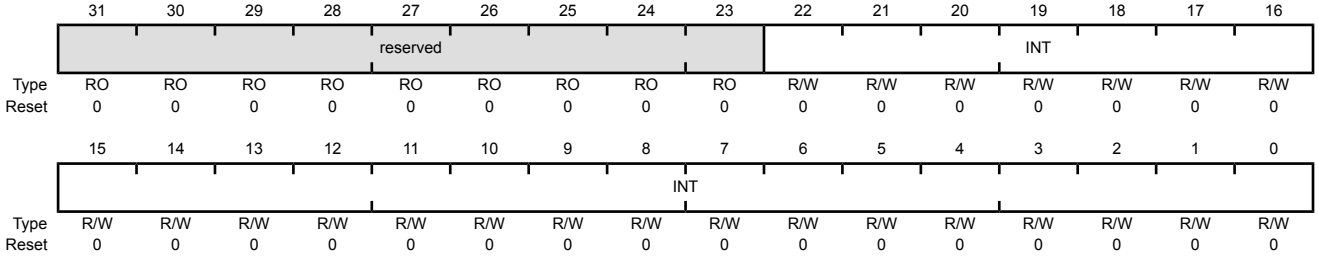
Register 7: Interrupt 32-54 Clear Enable (DIS1), offset 0x184

Note: This register can only be accessed from privileged mode.

The **DIS1** register disables interrupts. Bit 0 corresponds to Interrupt 32; bit 22 corresponds to Interrupt 54. See Table 2-9 on page 96 for interrupt assignments.

Interrupt 32-54 Clear Enable (DIS1)

Base 0xE000.E000
Offset 0x184
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:23	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22:0	INT	R/W	0x00.0000	Interrupt Disable

Value	Description
0	On a read, indicates the interrupt is disabled. On a write, no effect.
1	On a read, indicates the interrupt is enabled. On a write, clears the corresponding `INT[n]` bit in the **EN1** register, disabling interrupt [n].

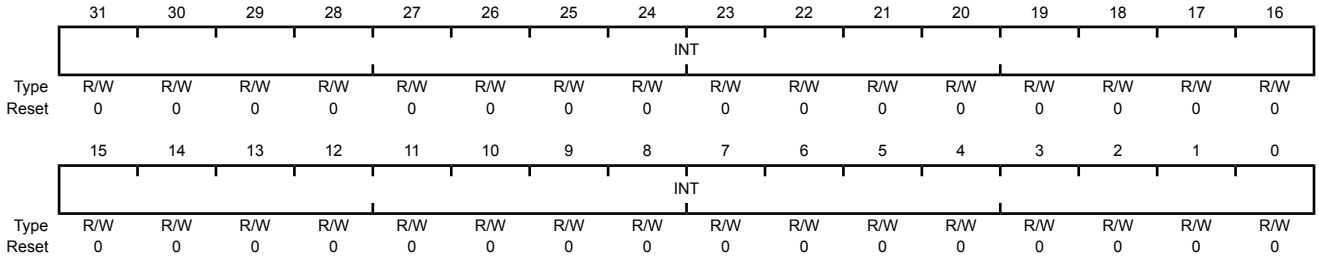
Register 8: Interrupt 0-31 Set Pending (PEND0), offset 0x200

Note: This register can only be accessed from privileged mode.

See Table 2-9 on page 96 for interrupt assignments.

Interrupt 0-31 Set Pending (PEND0)

Base 0xE000.E000
Offset 0x200
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	INT	R/W	0x0000.0000	Interrupt Set Pending

Value	Description
0	On a read, indicates that the interrupt is not pending. On a write, no effect.
1	On a read, indicates that the interrupt is pending. On a write, the corresponding interrupt is set to pending even if it is disabled.

If the corresponding interrupt is already pending, setting a bit has no effect.

A bit can only be cleared by setting the corresponding `INT[n]` bit in the **UNPEND0** register.

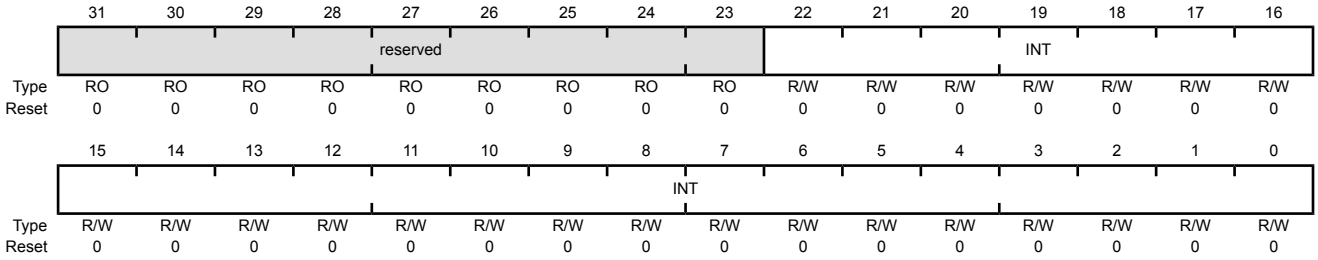
Register 9: Interrupt 32-54 Set Pending (PEND1), offset 0x204

Note: This register can only be accessed from privileged mode.

The **PEND1** register forces interrupts into the pending state and shows which interrupts are pending. Bit 0 corresponds to Interrupt 32; bit 22 corresponds to Interrupt 54. See Table 2-9 on page 96 for interrupt assignments.

Interrupt 32-54 Set Pending (PEND1)

Base 0xE000.E000
Offset 0x204
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:23	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22:0	INT	R/W	0x00.0000	Interrupt Set Pending

Value	Description
0	On a read, indicates that the interrupt is not pending. On a write, no effect.
1	On a read, indicates that the interrupt is pending. On a write, the corresponding interrupt is set to pending even if it is disabled.

If the corresponding interrupt is already pending, setting a bit has no effect.

A bit can only be cleared by setting the corresponding `INT[n]` bit in the **UNPEND1** register.

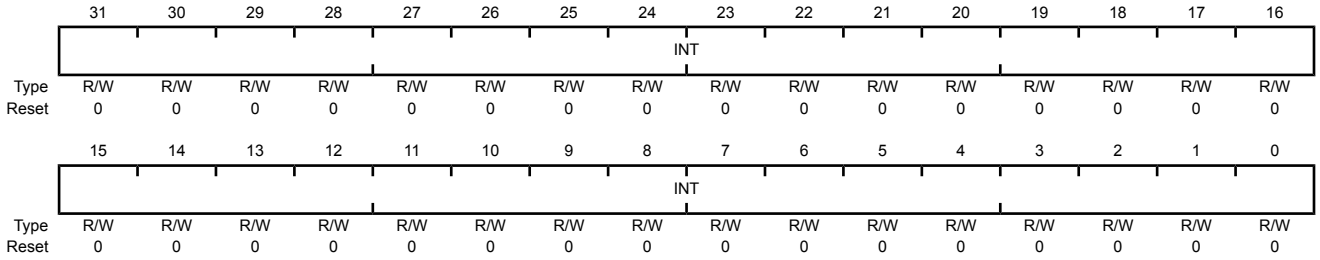
Register 10: Interrupt 0-31 Clear Pending (UNPEND0), offset 0x280

Note: This register can only be accessed from privileged mode.

See Table 2-9 on page 96 for interrupt assignments.

Interrupt 0-31 Clear Pending (UNPEND0)

Base 0xE000.E000
Offset 0x280
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	INT	R/W	0x0000.0000	Interrupt Clear Pending

Value	Description
0	On a read, indicates that the interrupt is not pending. On a write, no effect.
1	On a read, indicates that the interrupt is pending. On a write, clears the corresponding `INT[n]` bit in the **PEND0** register, so that interrupt [n] is no longer pending. Setting a bit does not affect the active state of the corresponding interrupt.

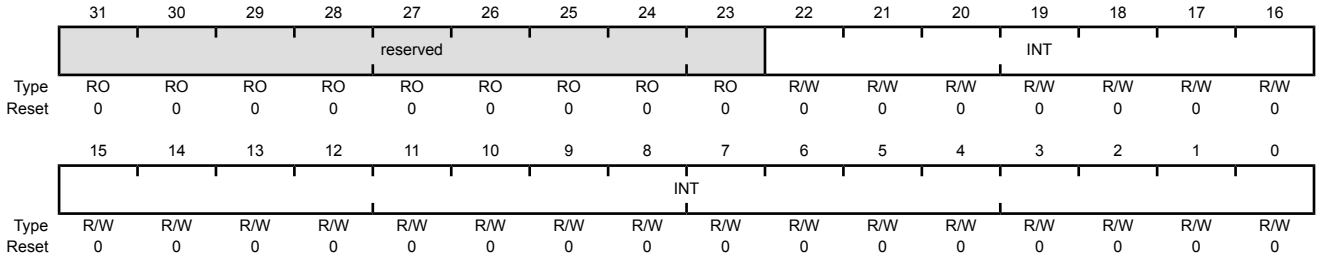
Register 11: Interrupt 32-54 Clear Pending (UNPEND1), offset 0x284

Note: This register can only be accessed from privileged mode.

The **UNPEND1** register shows which interrupts are pending and removes the pending state from interrupts. Bit 0 corresponds to Interrupt 32; bit 22 corresponds to Interrupt 54. See Table 2-9 on page 96 for interrupt assignments.

Interrupt 32-54 Clear Pending (UNPEND1)

Base 0xE000.E000
Offset 0x284
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:23	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22:0	INT	R/W	0x00.0000	Interrupt Clear Pending

Value	Description
0	On a read, indicates that the interrupt is not pending. On a write, no effect.
1	On a read, indicates that the interrupt is pending. On a write, clears the corresponding `INT[n]` bit in the **PEND1** register, so that interrupt [n] is no longer pending. Setting a bit does not affect the active state of the corresponding interrupt.

Register 12: Interrupt 0-31 Active Bit (ACTIVE0), offset 0x300

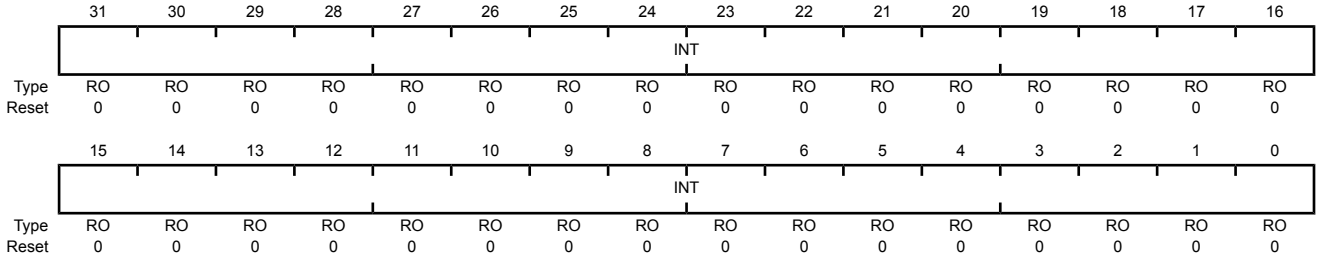
Note: This register can only be accessed from privileged mode.

See Table 2-9 on page 96 for interrupt assignments.

Caution – Do not manually set or clear the bits in this register.

Interrupt 0-31 Active Bit (ACTIVE0)

Base 0xE000.E000
Offset 0x300
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	INT	RO	0x0000.0000	Interrupt Active

Value	Description
0	The corresponding interrupt is not active.
1	The corresponding interrupt is active, or active and pending.

Register 13: Interrupt 32-54 Active Bit (ACTIVE1), offset 0x304

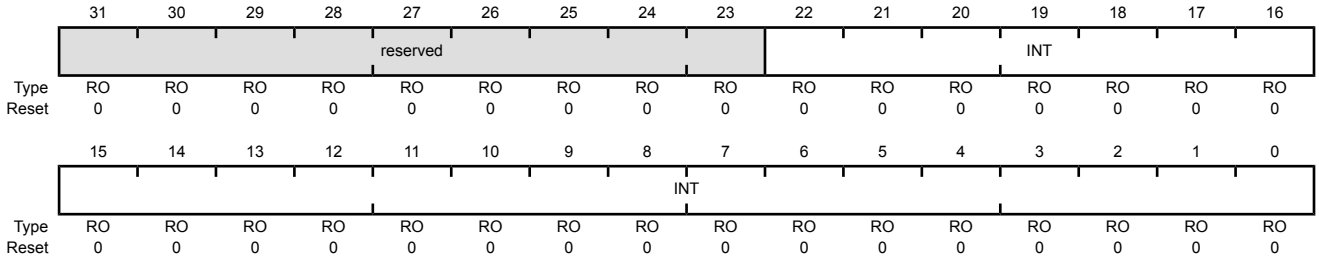
Note: This register can only be accessed from privileged mode.

The **ACTIVE1** register indicates which interrupts are active. Bit 0 corresponds to Interrupt 32; bit 22 corresponds to Interrupt 54. See Table 2-9 on page 96 for interrupt assignments.

Caution – Do not manually set or clear the bits in this register.

Interrupt 32-54 Active Bit (ACTIVE1)

Base 0xE000.E000
Offset 0x304
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:23	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22:0	INT	RO	0x00.0000	Interrupt Active

Value	Description
0	The corresponding interrupt is not active.
1	The corresponding interrupt is active, or active and pending.

Register 14: Interrupt 0-3 Priority (PRI0), offset 0x400

Register 15: Interrupt 4-7 Priority (PRI1), offset 0x404

Register 16: Interrupt 8-11 Priority (PRI2), offset 0x408

Register 17: Interrupt 12-15 Priority (PRI3), offset 0x40C

Register 18: Interrupt 16-19 Priority (PRI4), offset 0x410

Register 19: Interrupt 20-23 Priority (PRI5), offset 0x414

Register 20: Interrupt 24-27 Priority (PRI6), offset 0x418

Register 21: Interrupt 28-31 Priority (PRI7), offset 0x41C

Register 22: Interrupt 32-35 Priority (PRI8), offset 0x420

Register 23: Interrupt 36-39 Priority (PRI9), offset 0x424

Register 24: Interrupt 40-43 Priority (PRI10), offset 0x428

Register 25: Interrupt 44-47 Priority (PRI11), offset 0x42C

Register 26: Interrupt 48-51 Priority (PRI12), offset 0x430

Register 27: Interrupt 52-54 Priority (PRI13), offset 0x434

Note: This register can only be accessed from privileged mode.

The **PRIn** registers provide 3-bit priority fields for each interrupt. These registers are byte accessible. Each register holds four priority fields that are assigned to interrupts as follows:

PRIn Register Bit Field	Interrupt
Bits 31:29	Interrupt [4n+3]
Bits 23:21	Interrupt [4n+2]
Bits 15:13	Interrupt [4n+1]
Bits 7:5	Interrupt [4n]

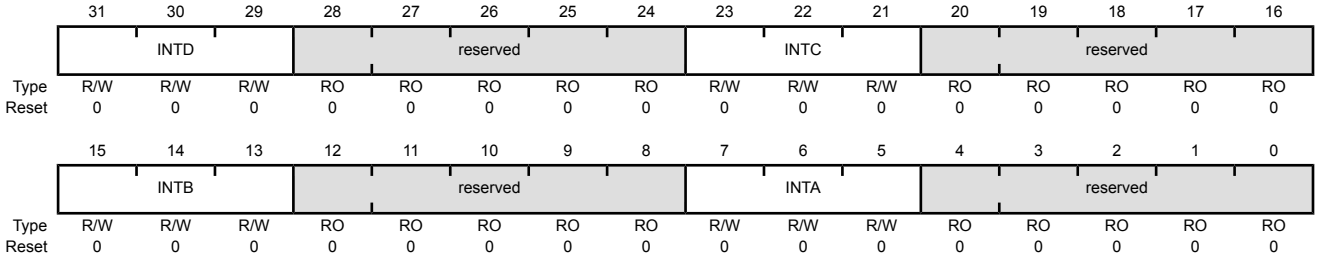
See Table 2-9 on page 96 for interrupt assignments.

Each priority level can be split into separate group priority and subpriority fields. The PRIGROUP field in the **Application Interrupt and Reset Control (APINT)** register (see page 143) indicates the position of the binary point that splits the priority and subpriority fields.

These registers can only be accessed from privileged mode.

Interrupt 0-3 Priority (PRI0)

Base 0xE000.E000
Offset 0x400
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:29	INTD	R/W	0x0	Interrupt Priority for Interrupt [4n+3] This field holds a priority value, 0-7, for the interrupt with the number [4n+3], where n is the number of the **Interrupt Priority** register (n=0 for **PRI0**, and so on). The lower the value, the greater the priority of the corresponding interrupt.
28:24	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:21	INTC	R/W	0x0	Interrupt Priority for Interrupt [4n+2] This field holds a priority value, 0-7, for the interrupt with the number [4n+2], where n is the number of the **Interrupt Priority** register (n=0 for **PRI0**, and so on). The lower the value, the greater the priority of the corresponding interrupt.
20:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:13	INTB	R/W	0x0	Interrupt Priority for Interrupt [4n+1] This field holds a priority value, 0-7, for the interrupt with the number [4n+1], where n is the number of the **Interrupt Priority** register (n=0 for **PRI0**, and so on). The lower the value, the greater the priority of the corresponding interrupt.
12:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:5	INTA	R/W	0x0	Interrupt Priority for Interrupt [4n] This field holds a priority value, 0-7, for the interrupt with the number [4n], where n is the number of the **Interrupt Priority** register (n=0 for **PRI0**, and so on). The lower the value, the greater the priority of the corresponding interrupt.
4:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 28: Software Trigger Interrupt (SWTRIG), offset 0xF00

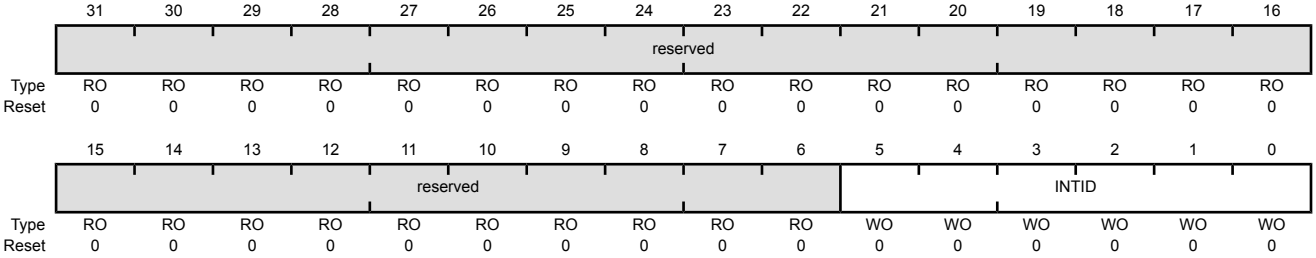
Note: Only privileged software can enable unprivileged access to the **SWTRIG** register.

Writing an interrupt number to the **SWTRIG** register generates a Software Generated Interrupt (SGI). See Table 2-9 on page 96 for interrupt assignments.

When the MAINPEND bit in the **Configuration and Control (CFGCTRL)** register (see page 147) is set, unprivileged software can access the **SWTRIG** register.

Software Trigger Interrupt (SWTRIG)

Base 0xE000.E000
Offset 0xF00
Type WO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	INTID	WO	0x00	Interrupt ID This field holds the interrupt ID of the required SGI. For example, a value of 0x3 generates an interrupt on IRQ3.

3.5 System Control Block (SCB) Register Descriptions

This section lists and describes the System Control Block (SCB) registers, in numerical order by address offset. The SCB registers can only be accessed from privileged mode.

All registers must be accessed with aligned word accesses except for the **FAULTSTAT** and **SYSPRI1**-**SYSPRI3** registers, which can be accessed with byte or aligned halfword or word accesses. The processor does not support unaligned accesses to system control block registers.

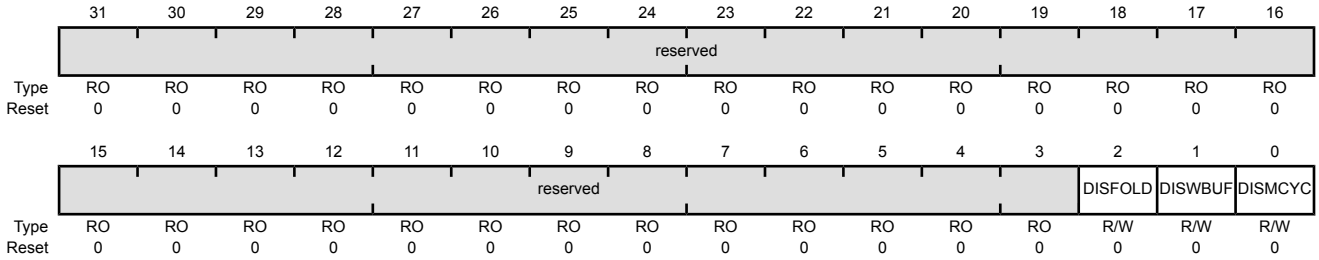
Register 29: Auxiliary Control (ACTLR), offset 0x008

Note: This register can only be accessed from privileged mode.

The **ACTLR** register provides disable bits for IT folding, write buffer use for accesses to the default memory map, and interruption of multi-cycle instructions. By default, this register is set to provide optimum performance from the Cortex-M3 processor and does not normally require modification.

Auxiliary Control (ACTLR)

Base 0xE000.E000
Offset 0x008
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	DISFOLD	R/W	0	Disable IT Folding Value Description 0 No effect. 1 Disables IT folding. In some situations, the processor can start executing the first instruction in an IT block while it is still executing the IT instruction. This behavior is called *IT folding*, and improves performance, However, IT folding can cause jitter in looping. If a task must avoid jitter, set the DISFOLD bit before executing the task, to disable IT folding.
1	DISWBUF	R/W	0	Disable Write Buffer Value Description 0 No effect. 1 Disables write buffer use during default memory map accesses. In this situation, all bus faults are precise bus faults but performance is decreased because any store to memory must complete before the processor can execute the next instruction. **Note:** This bit only affects write buffers implemented in the Cortex-M3 processor.

Bit/Field	Name	Type	Reset	Description
0	DISMCYC	R/W	0	Disable Interrupts of Multiple Cycle Instructions

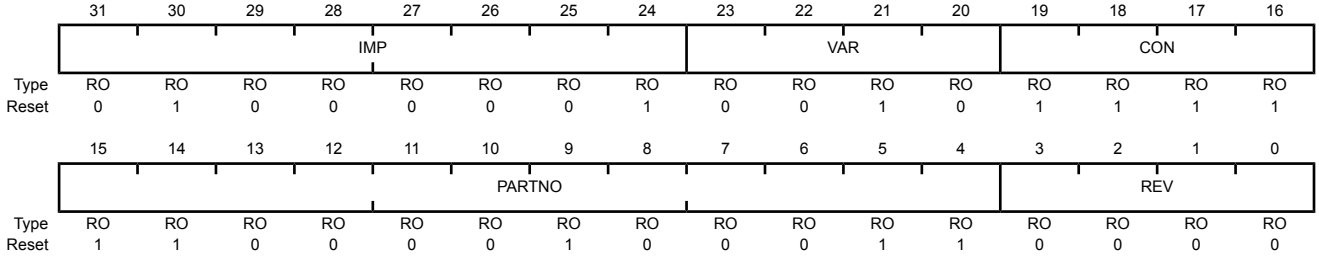
Value	Description
0	No effect.
1	Disables interruption of load multiple and store multiple instructions. In this situation, the interrupt latency of the processor is increased because any LDM or STM must complete before the processor can stack the current state and enter the interrupt handler.

Register 30: CPU ID Base (CPUID), offset 0xD00

Note: This register can only be accessed from privileged mode.

The **CPUID** register contains the ARM® Cortex™-M3 processor part number, version, and implementation information.

CPU ID Base (CPUID)
Base 0xE000.E000
Offset 0xD00
Type RO, reset 0x412F.C230



Bit/Field	Name	Type	Reset	Description
31:24	IMP	RO	0x41	Implementer Code Value Description 0x41 ARM
23:20	VAR	RO	0x2	Variant Number Value Description 0x2 The rn value in the rnpn product revision identifier, for example, the 2 in r2p0.
19:16	CON	RO	0xF	Constant Value Description 0xF Always reads as 0xF.
15:4	PARTNO	RO	0xC23	Part Number Value Description 0xC23 Cortex-M3 processor.
3:0	REV	RO	0x0	Revision Number Value Description 0x0 The pn value in the rnpn product revision identifier, for example, the 0 in r2p0.

Register 31: Interrupt Control and State (INTCTRL), offset 0xD04

Note: This register can only be accessed from privileged mode.

The **INCTRL** register provides a set-pending bit for the NMI exception, and set-pending and clear-pending bits for the PendSV and SysTick exceptions. In addition, bits in this register indicate the exception number of the exception being processed, whether there are preempted active exceptions, the exception number of the highest priority pending exception, and whether any interrupts are pending.

When writing to **INCTRL**, the effect is unpredictable when writing a 1 to both the PENDSV and UNPENDSV bits, or writing a 1 to both the PENDSTSET and PENDSTCLR bits.

Interrupt Control and State (INTCTRL)

Base 0xE000.E000
Offset 0xD04
Type R/W, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NMISET	reserved		PENDSV	UNPENDSV	PENDSTSET	PENDSTCLR	reserved	ISRPRE	ISRPEND	reserved			VECPEND		
Type R/W	RO	RO	R/W	WO	R/W	WO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit/Field	Name	Type	Reset	Description
31	NMISET	R/W	0	NMI Set Pending Value Description 0 On a read, indicates an NMI exception is not pending. On a write, no effect. 1 On a read, indicates an NMI exception is pending. On a write, changes the NMI exception state to pending. Because NMI is the highest-priority exception, normally the processor enters the NMI exception handler as soon as it registers the setting of this bit, and clears this bit on entering the interrupt handler. A read of this bit by the NMI exception handler returns 1 only if the NMI signal is reasserted while the processor is executing that handler.
30:29	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	PENDSV	R/W	0	PendSV Set Pending Value Description 0 On a read, indicates a PendSV exception is not pending. On a write, no effect. 1 On a read, indicates a PendSV exception is pending. On a write, changes the PendSV exception state to pending. Setting this bit is the only way to set the PendSV exception state to pending. This bit is cleared by writing a 1 to the UNPENDSV bit.

Bit/Field	Name	Type	Reset	Description
27	UNPENDSV	WO	0	PendSV Clear Pending Value Description 0 On a write, no effect. 1 On a write, removes the pending state from the PendSV exception. This bit is write only; on a register read, its value is unknown.
26	PENDSTSET	R/W	0	SysTick Set Pending Value Description 0 On a read, indicates a SysTick exception is not pending. On a write, no effect. 1 On a read, indicates a SysTick exception is pending. On a write, changes the SysTick exception state to pending. This bit is cleared by writing a 1 to the PENDSTCLR bit.
25	PENDSTCLR	WO	0	SysTick Clear Pending Value Description 0 On a write, no effect. 1 On a write, removes the pending state from the SysTick exception. This bit is write only; on a register read, its value is unknown.
24	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23	ISRPRE	RO	0	Debug Interrupt Handling Value Description 0 The release from halt does not take an interrupt. 1 The release from halt takes an interrupt. This bit is only meaningful in Debug mode and reads as zero when the processor is not in Debug mode.
22	ISRPEND	RO	0	Interrupt Pending Value Description 0 No interrupt is pending. 1 An interrupt is pending. This bit provides status for all interrupts excluding NMI and Faults.
21:19	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
18:12	VECPEND	RO	0x00	Interrupt Pending Vector Number This field contains the exception number of the highest priority pending enabled exception. The value indicated by this field includes the effect of the **BASEPRI** and **FAULTMASK** registers, but not any effect of the **PRIMASK** register. Value / Description 0x00 No exceptions are pending 0x01 Reserved 0x02 NMI 0x03 Hard fault 0x04 Memory management fault 0x05 Bus fault 0x06 Usage fault 0x07-0x0A Reserved 0x0B SVCall 0x0C Reserved for Debug 0x0D Reserved 0x0E PendSV 0x0F SysTick 0x10 Interrupt Vector 0 0x11 Interrupt Vector 1 ... ... 0x46 Interrupt Vector 54 0x47-0x7F Reserved
11	RETBASE	RO	0	Return to Base Value / Description 0 There are preempted active exceptions to execute. 1 There are no active exceptions, or the currently executing exception is the only active exception. This bit provides status for all interrupts excluding NMI and Faults. This bit only has meaning if the processor is currently executing an ISR (the **Interrupt Program Status (IPSR)** register is non-zero).
10:7	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	VECACT	RO	0x00	Interrupt Pending Vector Number This field contains the active exception number. The exception numbers can be found in the description for the VECPEND field. If this field is clear, the processor is in Thread mode. This field contains the same value as the ISRNUM field in the **IPSR** register. Subtract 16 from this value to obtain the IRQ number required to index into the **Interrupt Set Enable (ENn)**, **Interrupt Clear Enable (DISn**), **Interrupt Set Pending (PENDn)**, **Interrupt Clear Pending (UNPENDn)**, and **Interrupt Priority (PRIn)** registers (see page 76).

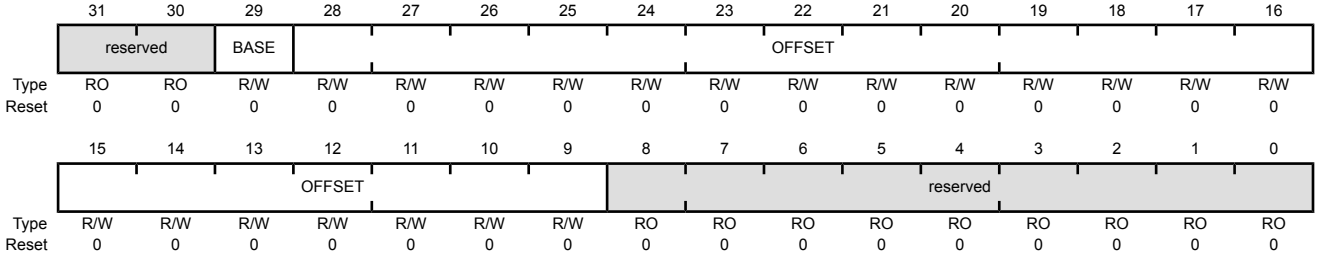
Register 32: Vector Table Offset (VTABLE), offset 0xD08

Note: This register can only be accessed from privileged mode.

The **VTABLE** register indicates the offset of the vector table base address from memory address 0x0000.0000.

Vector Table Offset (VTABLE)

Base 0xE000.E000
Offset 0xD08
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:30	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
29	BASE	R/W	0	Vector Table Base Value Description 0 The vector table is in the code memory region. 1 The vector table is in the SRAM memory region.
28:9	OFFSET	R/W	0x000.00	Vector Table Offset When configuring the OFFSET field, the offset must be aligned to the number of exception entries in the vector table. Because there are 54 interrupts, the offset must be aligned on a 512-byte boundary.
8:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 33: Application Interrupt and Reset Control (APINT), offset 0xD0C

Note: This register can only be accessed from privileged mode.

The **APINT** register provides priority grouping control for the exception model, endian status for data accesses, and reset control of the system. To write to this register, 0x05FA must be written to the VECTKEY field, otherwise the write is ignored.

The PRIGROUP field indicates the position of the binary point that splits the INTx fields in the **Interrupt Priority (PRIx)** registers into separate group priority and subpriority fields. Table 3-8 on page 143 shows how the PRIGROUP value controls this split. The bit numbers in the Group Priority Field and Subpriority Field columns in the table refer to the bits in the INTA field. For the INTB field, the corresponding bits are 15:13; for INTC, 23:21; and for INTD, 31:29.

Note: Determining preemption of an exception uses only the group priority field.

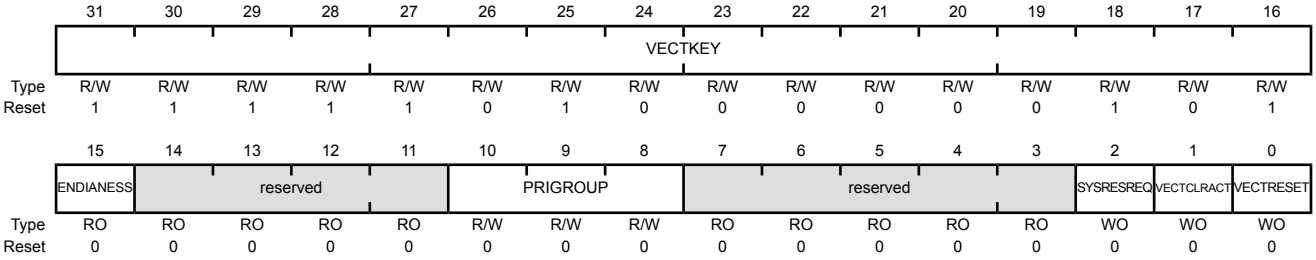
Table 3-8. Interrupt Priority Levels

PRIGROUP Bit Field	Binary Point[a]	Group Priority Field	Subpriority Field	Group Priorities	Subpriorities
0x0 - 0x4	bxxx.	[7:5]	None	8	1
0x5	bxx.y	[7:6]	[5]	4	2
0x6	bx.yy	[7]	[6:5]	2	4
0x7	b.yyy	None	[7:5]	1	8

a. INTx field showing the binary point. An x denotes a group priority field bit, and a y denotes a subpriority field bit.

Application Interrupt and Reset Control (APINT)

Base 0xE000.E000
Offset 0xD0C
Type R/W, reset 0xFA05.0000



Bit/Field	Name	Type	Reset	Description
31:16	VECTKEY	R/W	0xFA05	Register Key This field is used to guard against accidental writes to this register. 0x05FA must be written to this field in order to change the bits in this register. On a read, 0xFA05 is returned.
15	ENDIANESS	RO	0	Data Endianess The Stellaris implementation uses only little-endian mode so this is cleared to 0.
14:11	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

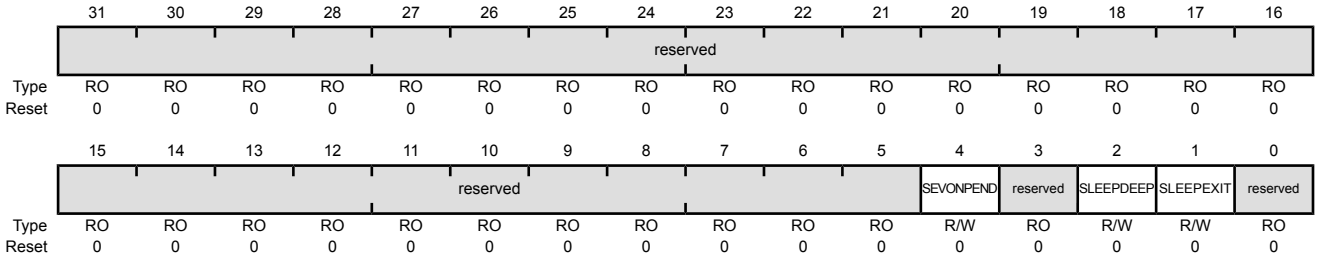
Bit/Field	Name	Type	Reset	Description
10:8	PRIGROUP	R/W	0x0	Interrupt Priority Grouping This field determines the split of group priority from subpriority (see Table 3-8 on page 143 for more information).
7:3	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	SYSRESREQ	WO	0	System Reset Request Value Description 0 No effect. 1 Resets the core and all on-chip peripherals except the Debug interface. This bit is automatically cleared during the reset of the core and reads as 0.
1	VECTCLRACT	WO	0	Clear Active NMI / Fault This bit is reserved for Debug use and reads as 0. This bit must be written as a 0, otherwise behavior is unpredictable.
0	VECTRESET	WO	0	System Reset This bit is reserved for Debug use and reads as 0. This bit must be written as a 0, otherwise behavior is unpredictable.

Register 34: System Control (SYSCTRL), offset 0xD10

Note: This register can only be accessed from privileged mode.

The **SYSCTRL** register controls features of entry to and exit from low-power state.

System Control (SYSCTRL)
Base 0xE000.E000
Offset 0xD10
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SEVONPEND	R/W	0	Wake Up on Pending Value Description 0 Only enabled interrupts or events can wake up the processor; disabled interrupts are excluded. 1 Enabled events and all interrupts, including disabled interrupts, can wake up the processor. When an event or interrupt enters the pending state, the event signal wakes up the processor from WFE. If the processor is not waiting for an event, the event is registered and affects the next WFE. The processor also wakes up on execution of a SEV instruction or an external event.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	SLEEPDEEP	R/W	0	Deep Sleep Enable Value Description 0 Use Sleep mode as the low power mode. 1 Use Deep-sleep mode as the low power mode.

Bit/Field	Name	Type	Reset	Description
1	SLEEPEXIT	R/W	0	Sleep on ISR Exit Value Description 0 When returning from Handler mode to Thread mode, do not sleep when returning to Thread mode. 1 When returning from Handler mode to Thread mode, enter sleep or deep sleep on return from an ISR. Setting this bit enables an interrupt-driven application to avoid returning to an empty main application.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

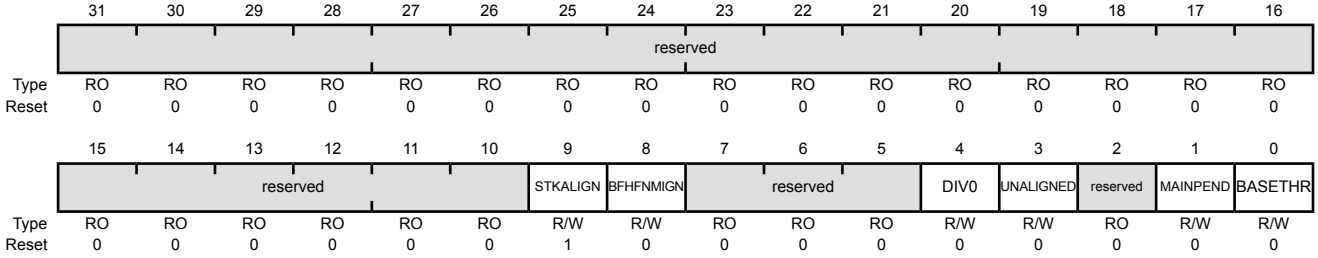
Register 35: Configuration and Control (CFGCTRL), offset 0xD14

Note: This register can only be accessed from privileged mode.

The **CFGCTRL** register controls entry to Thread mode and enables: the handlers for NMI, hard fault and faults escalated by the **FAULTMASK** register to ignore bus faults; trapping of divide by zero and unaligned accesses; and access to the **SWTRIG** register by unprivileged software (see page 135).

Configuration and Control (CFGCTRL)

Base 0xE000.E000
Offset 0xD14
Type R/W, reset 0x0000.0200



Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	STKALIGN	R/W	1	Stack Alignment on Exception Entry Value Description 0 The stack is 4-byte aligned. 1 The stack is 8-byte aligned. On exception entry, the processor uses bit 9 of the stacked **PSR** to indicate the stack alignment. On return from the exception, it uses this stacked bit to restore the correct stack alignment.
8	BFHFNMIGN	R/W	0	Ignore Bus Fault in NMI and Fault This bit enables handlers with priority -1 or -2 to ignore data bus faults caused by load and store instructions. The setting of this bit applies to the hard fault, NMI, and **FAULTMASK** escalated handlers. Value Description 0 Data bus faults caused by load and store instructions cause a lock-up. 1 Handlers running at priority -1 and -2 ignore data bus faults caused by load and store instructions. Set this bit only when the handler and its data are in absolutely safe memory. The normal use of this bit is to probe system devices and bridges to detect control path problems and fix them.
7:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
4	DIV0	R/W	0	Trap on Divide by 0 This bit enables faulting or halting when the processor executes an SDIV or UDIV instruction with a divisor of 0. Value Description 0 Do not trap on divide by 0. A divide by zero returns a quotient of 0. 1 Trap on divide by 0.
3	UNALIGNED	R/W	0	Trap on Unaligned Access Value Description 0 Do not trap on unaligned halfword and word accesses. 1 Trap on unaligned halfword and word accesses. An unaligned access generates a usage fault. Unaligned LDM, STM, LDRD, and STRD instructions always fault regardless of whether UNALIGNED is set.
2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	MAINPEND	R/W	0	Allow Main Interrupt Trigger Value Description 0 Disables unprivileged software access to the **SWTRIG** register. 1 Enables unprivileged software access to the **SWTRIG** register (see page 135).
0	BASETHR	R/W	0	Thread State Control Value Description 0 The processor can enter Thread mode only when no exception is active. 1 The processor can enter Thread mode from any level under the control of an EXC_RETURN value (see “Exception Return” on page 100 for more information).

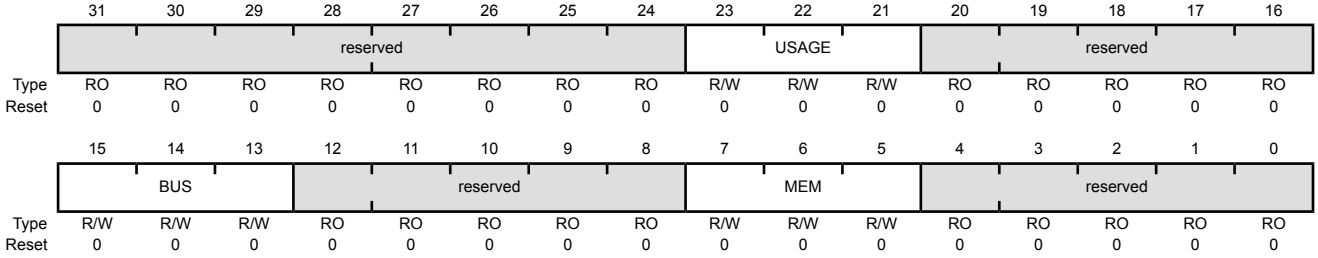
Register 36: System Handler Priority 1 (SYSPRI1), offset 0xD18

Note: This register can only be accessed from privileged mode.

The **SYSPRI1** register configures the priority level, 0 to 7 of the usage fault, bus fault, and memory management fault exception handlers. This register is byte-accessible.

System Handler Priority 1 (SYSPRI1)

Base 0xE000.E000
Offset 0xD18
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:21	USAGE	R/W	0x0	Usage Fault Priority This field configures the priority level of the usage fault. Configurable priority values are in the range 0-7, with lower values having higher priority.
20:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:13	BUS	R/W	0x0	Bus Fault Priority This field configures the priority level of the bus fault. Configurable priority values are in the range 0-7, with lower values having higher priority.
12:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:5	MEM	R/W	0x0	Memory Management Fault Priority This field configures the priority level of the memory management fault. Configurable priority values are in the range 0-7, with lower values having higher priority.
4:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

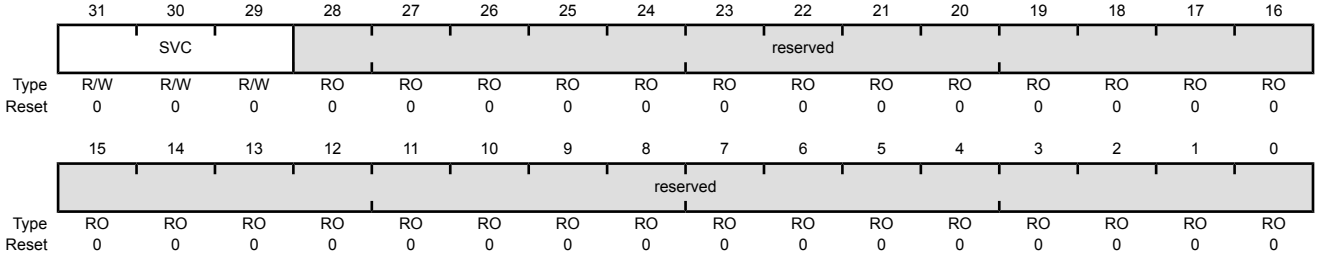
Register 37: System Handler Priority 2 (SYSPRI2), offset 0xD1C

Note: This register can only be accessed from privileged mode.

The **SYSPRI2** register configures the priority level, 0 to 7 of the SVCall handler. This register is byte-accessible.

System Handler Priority 2 (SYSPRI2)

Base 0xE000.E000
Offset 0xD1C
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:29	SVC	R/W	0x0	SVCall Priority This field configures the priority level of SVCall. Configurable priority values are in the range 0-7, with lower values having higher priority.
28:0	reserved	RO	0x000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

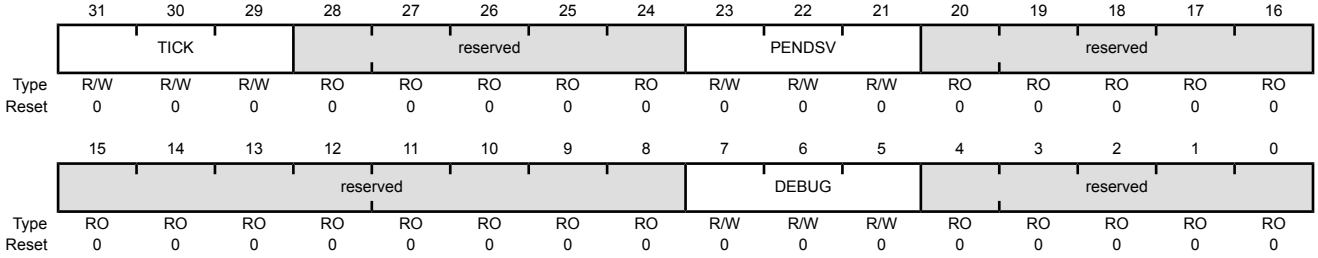
Register 38: System Handler Priority 3 (SYSPRI3), offset 0xD20

Note: This register can only be accessed from privileged mode.

The **SYSPRI3** register configures the priority level, 0 to 7 of the SysTick exception and PendSV handlers. This register is byte-accessible.

System Handler Priority 3 (SYSPRI3)

Base 0xE000.E000
Offset 0xD20
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:29	TICK	R/W	0x0	SysTick Exception Priority This field configures the priority level of the SysTick exception. Configurable priority values are in the range 0-7, with lower values having higher priority.
28:24	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:21	PENDSV	R/W	0x0	PendSV Priority This field configures the priority level of PendSV. Configurable priority values are in the range 0-7, with lower values having higher priority.
20:8	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:5	DEBUG	R/W	0x0	Debug Priority This field configures the priority level of Debug. Configurable priority values are in the range 0-7, with lower values having higher priority.
4:0	reserved	RO	0x0.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 39: System Handler Control and State (SYSHNDCTRL), offset 0xD24

Note: This register can only be accessed from privileged mode.

The **SYSHNDCTRL** register enables the system handlers, and indicates the pending status of the usage fault, bus fault, memory management fault, and SVC exceptions as well as the active status of the system handlers.

If a system handler is disabled and the corresponding fault occurs, the processor treats the fault as a hard fault.

This register can be modified to change the pending or active status of system exceptions. An OS kernel can write to the active bits to perform a context switch that changes the current exception type.

Caution – Software that changes the value of an active bit in this register without correct adjustment to the stacked content can cause the processor to generate a fault exception. Ensure software that writes to this register retains and subsequently restores the current active status.

If the value of a bit in this register must be modified after enabling the system handlers, a read-modify-write procedure must be used to ensure that only the required bit is modified.

System Handler Control and State (SYSHNDCTRL)

Base 0xE000.E000
Offset 0xD24
Type R/W, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved													USAGE	BUS	MEM
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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SVC	BUSP	MEMP	USAGEP	TICK	PNDSV	reserved	MON	SVCA	reserved			USGA	reserved	BUSA	MEMA
Type R/W	R/W	R/W	R/W	R/W	R/W	RO	R/W	R/W	RO	RO	RO	R/W	RO	R/W	R/W
Reset 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:19	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
18	USAGE	R/W	0	Usage Fault Enable Value Description 0 Disables the usage fault exception. 1 Enables the usage fault exception.
17	BUS	R/W	0	Bus Fault Enable Value Description 0 Disables the bus fault exception. 1 Enables the bus fault exception.

Bit/Field	Name	Type	Reset	Description
16	MEM	R/W	0	Memory Management Fault Enable Value Description 0 Disables the memory management fault exception. 1 Enables the memory management fault exception.
15	SVC	R/W	0	SVC Call Pending Value Description 0 An SVC call exception is not pending. 1 An SVC call exception is pending. This bit can be modified to change the pending status of the SVC call exception.
14	BUSP	R/W	0	Bus Fault Pending Value Description 0 A bus fault exception is not pending. 1 A bus fault exception is pending. This bit can be modified to change the pending status of the bus fault exception.
13	MEMP	R/W	0	Memory Management Fault Pending Value Description 0 A memory management fault exception is not pending. 1 A memory management fault exception is pending. This bit can be modified to change the pending status of the memory management fault exception.
12	USAGEP	R/W	0	Usage Fault Pending Value Description 0 A usage fault exception is not pending. 1 A usage fault exception is pending. This bit can be modified to change the pending status of the usage fault exception.
11	TICK	R/W	0	SysTick Exception Active Value Description 0 A SysTick exception is not active. 1 A SysTick exception is active. This bit can be modified to change the active status of the SysTick exception, however, see the Caution above before setting this bit.

Bit/Field	Name	Type	Reset	Description
10	PNDSV	R/W	0	PendSV Exception Active Value Description 0 A PendSV exception is not active. 1 A PendSV exception is active. This bit can be modified to change the active status of the PendSV exception, however, see the Caution above before setting this bit.
9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	MON	R/W	0	Debug Monitor Active Value Description 0 The Debug monitor is not active. 1 The Debug monitor is active.
7	SVCA	R/W	0	SVC Call Active Value Description 0 SVC call is not active. 1 SVC call is active. This bit can be modified to change the active status of the SVC call exception, however, see the Caution above before setting this bit.
6:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	USGA	R/W	0	Usage Fault Active Value Description 0 Usage fault is not active. 1 Usage fault is active. This bit can be modified to change the active status of the usage fault exception, however, see the Caution above before setting this bit.
2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BUSA	R/W	0	Bus Fault Active Value Description 0 Bus fault is not active. 1 Bus fault is active. This bit can be modified to change the active status of the bus fault exception, however, see the Caution above before setting this bit.

Bit/Field	Name	Type	Reset	Description
0	MEMA	R/W	0	Memory Management Fault Active

Value	Description
0	Memory management fault is not active.
1	Memory management fault is active.

This bit can be modified to change the active status of the memory management fault exception, however, see the Caution above before setting this bit.

Register 40: Configurable Fault Status (FAULTSTAT), offset 0xD28

Note: This register can only be accessed from privileged mode.

The **FAULTSTAT** register indicates the cause of a memory management fault, bus fault, or usage fault. Each of these functions is assigned to a subregister as follows:

- **Usage Fault Status (UFAULTSTAT)**, bits 31:16
- **Bus Fault Status (BFAULTSTAT)**, bits 15:8
- **Memory Management Fault Status (MFAULTSTAT)**, bits 7:0

FAULTSTAT is byte accessible. **FAULTSTAT** or its subregisters can be accessed as follows:

- The complete **FAULTSTAT** register, with a word access to offset 0xD28
- The **MFAULTSTAT**, with a byte access to offset 0xD28
- The **MFAULTSTAT** and **BFAULTSTAT**, with a halfword access to offset 0xD28
- The **BFAULTSTAT**, with a byte access to offset 0xD29
- The **UFAULTSTAT**, with a halfword access to offset 0xD2A

Bits are cleared by writing a 1 to them.

In a fault handler, the true faulting address can be determined by:

1. Read and save the **Memory Management Fault Address (MMADDR)** or **Bus Fault Address (FAULTADDR)** value.
2. Read the MMARV bit in **MFAULTSTAT**, or the BFARV bit in **BFAULTSTAT** to determine if the **MMADDR** or **FAULTADDR** contents are valid.

Software must follow this sequence because another higher priority exception might change the **MMADDR** or **FAULTADDR** value. For example, if a higher priority handler preempts the current fault handler, the other fault might change the **MMADDR** or **FAULTADDR** value.

Configurable Fault Status (FAULTSTAT)
Base 0xE000.E000
Offset 0xD28
Type R/W1C, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved						DIV0	UNALIGN	reserved				NOCP	INVPC	INVSTAT	UNDEF
Type RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C
Reset 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BFARV	reserved		BSTKE	BUSTKE	IMPRE	PRECISE	IBUS	MMARV	reserved		MSTKE	MUSTKE	reserved	DERR	IERR
Type R/W1C	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	RO	RO	R/W1C	R/W1C	RO	R/W1C	R/W1C
Reset 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
25	DIV0	R/W1C	0	Divide-by-Zero Usage Fault Value Description 0 No divide-by-zero fault has occurred, or divide-by-zero trapping is not enabled. 1 The processor has executed an SDIV or UDIV instruction with a divisor of 0. When this bit is set, the **PC** value stacked for the exception return points to the instruction that performed the divide by zero. Trapping on divide-by-zero is enabled by setting the DIV0 bit in the **Configuration and Control (CFGCTRL)** register (see page 147). This bit is cleared by writing a 1 to it.
24	UNALIGN	R/W1C	0	Unaligned Access Usage Fault Value Description 0 No unaligned access fault has occurred, or unaligned access trapping is not enabled. 1 The processor has made an unaligned memory access. Unaligned LDM, STM, LDRD, and STRD instructions always fault regardless of the configuration of this bit. Trapping on unaligned access is enabled by setting the UNALIGNED bit in the **CFGCTRL** register (see page 147). This bit is cleared by writing a 1 to it.
23:20	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	NOCP	R/W1C	0	No Coprocessor Usage Fault Value Description 0 A usage fault has not been caused by attempting to access a coprocessor. 1 The processor has attempted to access a coprocessor. This bit is cleared by writing a 1 to it.
18	INVPC	R/W1C	0	Invalid PC Load Usage Fault Value Description 0 A usage fault has not been caused by attempting to load an invalid **PC** value. 1 The processor has attempted an illegal load of EXC_RETURN to the **PC** as a result of an invalid context or an invalid EXC_RETURN value. When this bit is set, the **PC** value stacked for the exception return points to the instruction that tried to perform the illegal load of the **PC**. This bit is cleared by writing a 1 to it.

Bit/Field	Name	Type	Reset	Description
17	INVSTAT	R/W1C	0	Invalid State Usage Fault Value Description 0 A usage fault has not been caused by an invalid state. 1 The processor has attempted to execute an instruction that makes illegal use of the **EPSR** register. When this bit is set, the **PC** value stacked for the exception return points to the instruction that attempted the illegal use of the **Execution Program Status Register (EPSR)** register. This bit is not set if an undefined instruction uses the **EPSR** register. This bit is cleared by writing a 1 to it.
16	UNDEF	R/W1C	0	Undefined Instruction Usage Fault Value Description 0 A usage fault has not been caused by an undefined instruction. 1 The processor has attempted to execute an undefined instruction. When this bit is set, the **PC** value stacked for the exception return points to the undefined instruction. An undefined instruction is an instruction that the processor cannot decode. This bit is cleared by writing a 1 to it.
15	BFARV	R/W1C	0	Bus Fault Address Register Valid Value Description 0 The value in the **Bus Fault Address (FAULTADDR)** register is not a valid fault address. 1 The **FAULTADDR** register is holding a valid fault address. This bit is set after a bus fault, where the address is known. Other faults can clear this bit, such as a memory management fault occurring later. If a bus fault occurs and is escalated to a hard fault because of priority, the hard fault handler must clear this bit. This action prevents problems if returning to a stacked active bus fault handler whose **FAULTADDR** register value has been overwritten. This bit is cleared by writing a 1 to it.
14:13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
12	BSTKE	R/W1C	0	Stack Bus Fault Value Description 0 No bus fault has occurred on stacking for exception entry. 1 Stacking for an exception entry has caused one or more bus faults. When this bit is set, the **SP** is still adjusted but the values in the context area on the stack might be incorrect. A fault address is not written to the **FAULTADDR** register. This bit is cleared by writing a 1 to it.
11	BUSTKE	R/W1C	0	Unstack Bus Fault Value Description 0 No bus fault has occurred on unstacking for a return from exception. 1 Unstacking for a return from exception has caused one or more bus faults. This fault is chained to the handler. Thus, when this bit is set, the original return stack is still present. The **SP** is not adjusted from the failing return, a new save is not performed, and a fault address is not written to the **FAULTADDR** register. This bit is cleared by writing a 1 to it.
10	IMPRE	R/W1C	0	Imprecise Data Bus Error Value Description 0 An imprecise data bus error has not occurred. 1 A data bus error has occurred, but the return address in the stack frame is not related to the instruction that caused the error. When this bit is set, a fault address is not written to the **FAULTADDR** register. This fault is asynchronous. Therefore, if the fault is detected when the priority of the current process is higher than the bus fault priority, the bus fault becomes pending and becomes active only when the processor returns from all higher-priority processes. If a precise fault occurs before the processor enters the handler for the imprecise bus fault, the handler detects that both the IMPRE bit is set and one of the precise fault status bits is set. This bit is cleared by writing a 1 to it.
9	PRECISE	R/W1C	0	Precise Data Bus Error Value Description 0 A precise data bus error has not occurred. 1 A data bus error has occurred, and the **PC** value stacked for the exception return points to the instruction that caused the fault. When this bit is set, the fault address is written to the **FAULTADDR** register. This bit is cleared by writing a 1 to it.

Bit/Field	Name	Type	Reset	Description
8	IBUS	R/W1C	0	Instruction Bus Error Value Description 0 An instruction bus error has not occurred. 1 An instruction bus error has occurred. The processor detects the instruction bus error on prefetching an instruction, but sets this bit only if it attempts to issue the faulting instruction. When this bit is set, a fault address is not written to the **FAULTADDR** register. This bit is cleared by writing a 1 to it.
7	MMARV	R/W1C	0	Memory Management Fault Address Register Valid Value Description 0 The value in the **Memory Management Fault Address (MMADDR)** register is not a valid fault address. 1 The **MMADDR** register is holding a valid fault address. If a memory management fault occurs and is escalated to a hard fault because of priority, the hard fault handler must clear this bit. This action prevents problems if returning to a stacked active memory management fault handler whose **MMADDR** register value has been overwritten. This bit is cleared by writing a 1 to it.
6:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	MSTKE	R/W1C	0	Stack Access Violation Value Description 0 No memory management fault has occurred on stacking for exception entry. 1 Stacking for an exception entry has caused one or more access violations. When this bit is set, the **SP** is still adjusted but the values in the context area on the stack might be incorrect. A fault address is not written to the **MMADDR** register. This bit is cleared by writing a 1 to it.

Bit/Field	Name	Type	Reset	Description
3	MUSTKE	R/W1C	0	Unstack Access Violation Value Description 0 No memory management fault has occurred on unstacking for a return from exception. 1 Unstacking for a return from exception has caused one or more access violations. This fault is chained to the handler. Thus, when this bit is set, the original return stack is still present. The **SP** is not adjusted from the failing return, a new save is not performed, and a fault address is not written to the **MMADDR** register. This bit is cleared by writing a 1 to it.
2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	DERR	R/W1C	0	Data Access Violation Value Description 0 A data access violation has not occurred. 1 The processor attempted a load or store at a location that does not permit the operation. When this bit is set, the **PC** value stacked for the exception return points to the faulting instruction and the address of the attempted access is written to the **MMADDR** register. This bit is cleared by writing a 1 to it.
0	IERR	R/W1C	0	Instruction Access Violation Value Description 0 An instruction access violation has not occurred. 1 The processor attempted an instruction fetch from a location that does not permit execution. This fault occurs on any access to an XN region, even when the MPU is disabled or not present. When this bit is set, the **PC** value stacked for the exception return points to the faulting instruction and the address of the attempted access is not written to the **MMADDR** register. This bit is cleared by writing a 1 to it.

Register 41: Hard Fault Status (HFAULTSTAT), offset 0xD2C

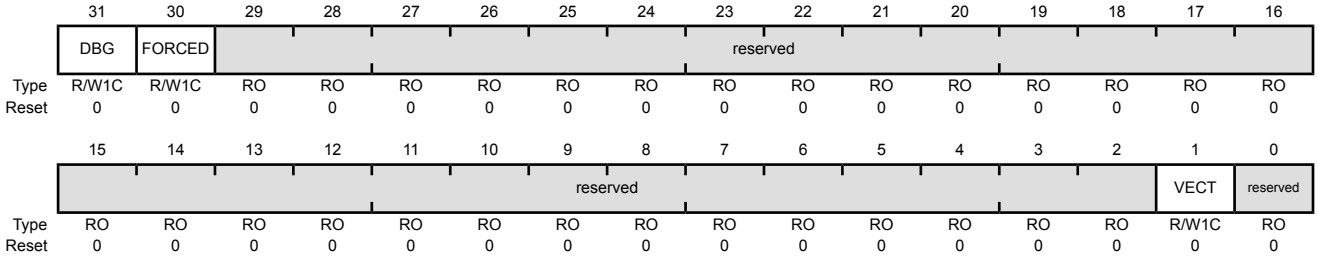
Note: This register can only be accessed from privileged mode.

The **HFAULTSTAT** register gives information about events that activate the hard fault handler.

Bits are cleared by writing a 1 to them.

Hard Fault Status (HFAULTSTAT)

Base 0xE000.E000
Offset 0xD2C
Type R/W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31	DBG	R/W1C	0	Debug Event This bit is reserved for Debug use. This bit must be written as a 0, otherwise behavior is unpredictable.
30	FORCED	R/W1C	0	Forced Hard Fault Value Description 0 No forced hard fault has occurred. 1 A forced hard fault has been generated by escalation of a fault with configurable priority that cannot be handled, either because of priority or because it is disabled. When this bit is set, the hard fault handler must read the other fault status registers to find the cause of the fault. This bit is cleared by writing a 1 to it.
29:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	VECT	R/W1C	0	Vector Table Read Fault Value Description 0 No bus fault has occurred on a vector table read. 1 A bus fault occurred on a vector table read. This error is always handled by the hard fault handler. When this bit is set, the **PC** value stacked for the exception return points to the instruction that was preempted by the exception. This bit is cleared by writing a 1 to it.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

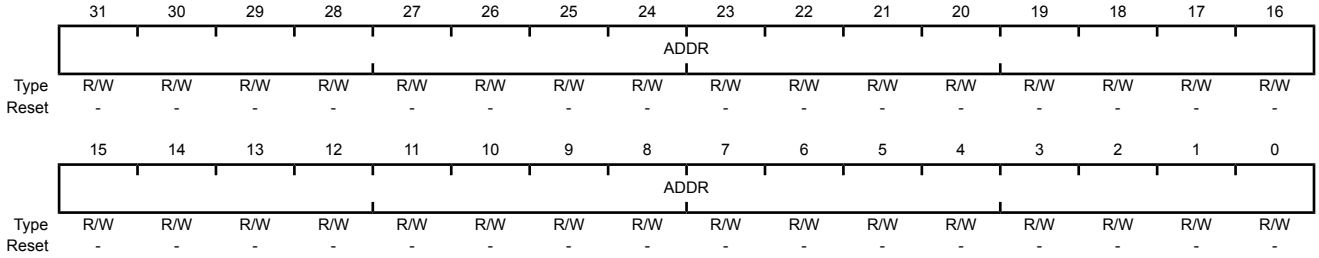
Register 42: Memory Management Fault Address (MMADDR), offset 0xD34

Note: This register can only be accessed from privileged mode.

The **MMADDR** register contains the address of the location that generated a memory management fault. When an unaligned access faults, the address in the **MMADDR** register is the actual address that faulted. Because a single read or write instruction can be split into multiple aligned accesses, the fault address can be any address in the range of the requested access size. Bits in the **Memory Management Fault Status (MFAULTSTAT)** register indicate the cause of the fault and whether the value in the **MMADDR** register is valid (see page 156).

Memory Management Fault Address (MMADDR)

Base 0xE000.E000
Offset 0xD34
Type R/W, reset -



Bit/Field	Name	Type	Reset	Description
31:0	ADDR	R/W	-	Fault Address When the MMARV bit of **MFAULTSTAT** is set, this field holds the address of the location that generated the memory management fault.

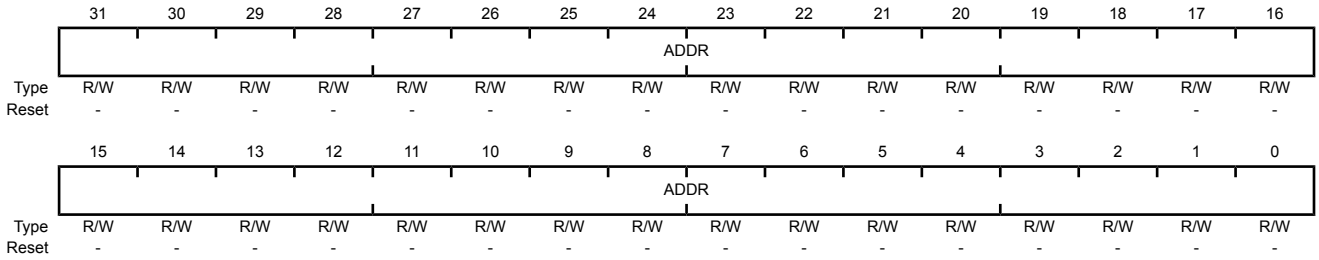
Register 43: Bus Fault Address (FAULTADDR), offset 0xD38

Note: This register can only be accessed from privileged mode.

The **FAULTADDR** register contains the address of the location that generated a bus fault. When an unaligned access faults, the address in the **FAULTADDR** register is the one requested by the instruction, even if it is not the address of the fault. Bits in the **Bus Fault Status (BFAULTSTAT)** register indicate the cause of the fault and whether the value in the **FAULTADDR** register is valid (see page 156).

Bus Fault Address (FAULTADDR)

Base 0xE000.E000
Offset 0xD38
Type R/W, reset -



Bit/Field	Name	Type	Reset	Description
31:0	ADDR	R/W	-	Fault Address When the FAULTADDRV bit of **BFAULTSTAT** is set, this field holds the address of the location that generated the bus fault.

3.6 Memory Protection Unit (MPU) Register Descriptions

This section lists and describes the Memory Protection Unit (MPU) registers, in numerical order by address offset.

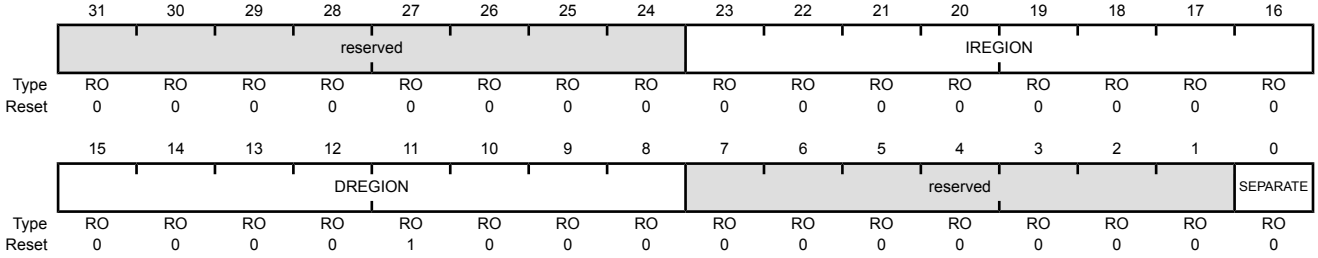
The MPU registers can only be accessed from privileged mode.

Register 44: MPU Type (MPUTYPE), offset 0xD90

Note: This register can only be accessed from privileged mode.

The **MPUTYPE** register indicates whether the MPU is present, and if so, how many regions it supports.

MPU Type (MPUTYPE)
Base 0xE000.E000
Offset 0xD90
Type RO, reset 0x0000.0800



Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:16	IREGION	RO	0x00	Number of I Regions This field indicates the number of supported MPU instruction regions. This field always contains 0x00. The MPU memory map is unified and is described by the `DREGION` field.
15:8	DREGION	RO	0x08	Number of D Regions Value Description 0x08 Indicates there are eight supported MPU data regions.
7:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	SEPARATE	RO	0	Separate or Unified MPU Value Description 0 Indicates the MPU is unified.

Register 45: MPU Control (MPUCTRL), offset 0xD94

Note: This register can only be accessed from privileged mode.

The **MPUCTRL** register enables the MPU, enables the default memory map background region, and enables use of the MPU when in the hard fault, Non-maskable Interrupt (NMI), and **Fault Mask Register (FAULTMASK)** escalated handlers.

When the ENABLE and PRIVDEFEN bits are both set:

- For privileged accesses, the default memory map is as described in "Memory Model" on page 84. Any access by privileged software that does not address an enabled memory region behaves as defined by the default memory map.
- Any access by unprivileged software that does not address an enabled memory region causes a memory management fault.

Execute Never (XN) and Strongly Ordered rules always apply to the System Control Space regardless of the value of the ENABLE bit.

When the ENABLE bit is set, at least one region of the memory map must be enabled for the system to function unless the PRIVDEFEN bit is set. If the PRIVDEFEN bit is set and no regions are enabled, then only privileged software can operate.

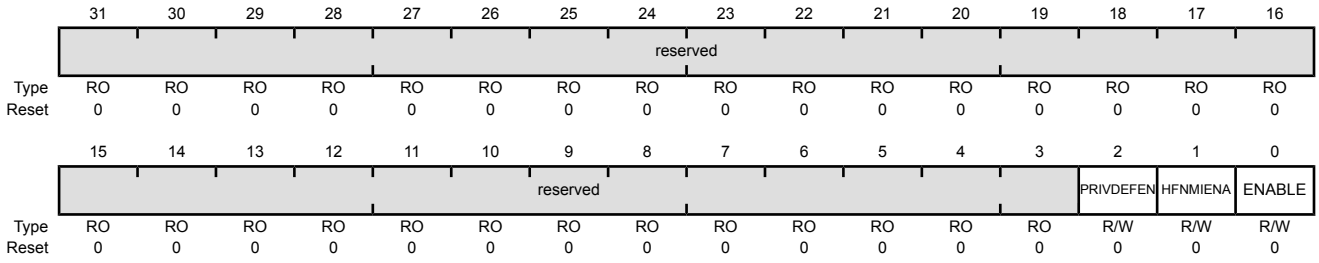
When the ENABLE bit is clear, the system uses the default memory map, which has the same memory attributes as if the MPU is not implemented (see Table 2-5 on page 87 for more information). The default memory map applies to accesses from both privileged and unprivileged software.

When the MPU is enabled, accesses to the System Control Space and vector table are always permitted. Other areas are accessible based on regions and whether PRIVDEFEN is set.

Unless HFNMIENA is set, the MPU is not enabled when the processor is executing the handler for an exception with priority –1 or –2. These priorities are only possible when handling a hard fault or NMI exception or when **FAULTMASK** is enabled. Setting the HFNMIENA bit enables the MPU when operating with these two priorities.

MPU Control (MPUCTRL)

Base 0xE000.E000
Offset 0xD94
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
2	PRIVDEFEN	R/W	0	MPU Default Region This bit enables privileged software access to the default memory map. Value Description 0 If the MPU is enabled, this bit disables use of the default memory map. Any memory access to a location not covered by any enabled region causes a fault. 1 If the MPU is enabled, this bit enables use of the default memory map as a background region for privileged software accesses. When this bit is set, the background region acts as if it is region number -1. Any region that is defined and enabled has priority over this default map. If the MPU is disabled, the processor ignores this bit.
1	HFNMIENA	R/W	0	MPU Enabled During Faults This bit controls the operation of the MPU during hard fault, NMI, and **FAULTMASK** handlers. Value Description 0 The MPU is disabled during hard fault, NMI, and **FAULTMASK** handlers, regardless of the value of the ENABLE bit. 1 The MPU is enabled during hard fault, NMI, and **FAULTMASK** handlers. When the MPU is disabled and this bit is set, the resulting behavior is unpredictable.
0	ENABLE	R/W	0	MPU Enable Value Description 0 The MPU is disabled. 1 The MPU is enabled. When the MPU is disabled and the HFNMIENA bit is set, the resulting behavior is unpredictable.

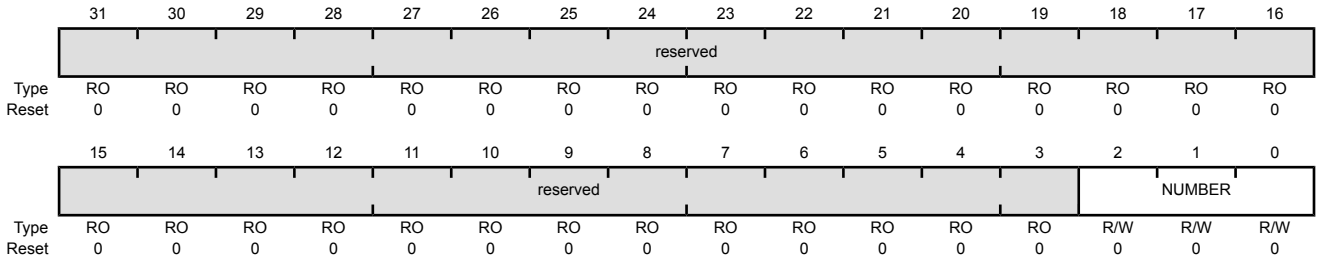
Register 46: MPU Region Number (MPUNUMBER), offset 0xD98

Note: This register can only be accessed from privileged mode.

The **MPUNUMBER** register selects which memory region is referenced by the **MPU Region Base Address (MPUBASE)** and **MPU Region Attribute and Size (MPUATTR)** registers. Normally, the required region number should be written to this register before accessing the **MPUBASE** or the **MPUATTR** register. However, the region number can be changed by writing to the **MPUBASE** register with the VALID bit set (see page 169). This write updates the value of the REGION field.

MPU Region Number (MPUNUMBER)

Base 0xE000.E000
Offset 0xD98
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	NUMBER	R/W	0x0	MPU Region to Access This field indicates the MPU region referenced by the **MPUBASE** and **MPUATTR** registers. The MPU supports eight memory regions.

Register 47: MPU Region Base Address (MPUBASE), offset 0xD9C

Register 48: MPU Region Base Address Alias 1 (MPUBASE1), offset 0xDA4

Register 49: MPU Region Base Address Alias 2 (MPUBASE2), offset 0xDAC

Register 50: MPU Region Base Address Alias 3 (MPUBASE3), offset 0xDB4

Note: This register can only be accessed from privileged mode.

The **MPUBASE** register defines the base address of the MPU region selected by the **MPU Region Number (MPUNUMBER)** register and can update the value of the **MPUNUMBER** register. To change the current region number and update the **MPUNUMBER** register, write the **MPUBASE** register with the VALID bit set.

The ADDR field is bits 31:*N* of the **MPUBASE** register. Bits (*N*-1):5 are reserved. The region size, as specified by the SIZE field in the **MPU Region Attribute and Size (MPUATTR)** register, defines the value of *N* where:

$N = Log_2(\text{Region size in bytes})$

If the region size is configured to 4 GB in the **MPUATTR** register, there is no valid ADDR field. In this case, the region occupies the complete memory map, and the base address is 0x0000.0000.

The base address is aligned to the size of the region. For example, a 64-KB region must be aligned on a multiple of 64 KB, for example, at 0x0001.0000 or 0x0002.0000.

MPU Region Base Address (MPUBASE)

Base 0xE000.E000
Offset 0xD9C
Type R/W, reset 0x0000.0000

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ADDR											VALID	reserved	REGION		
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	WO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:5	ADDR	R/W	0x0000.000	Base Address Mask Bits 31:*N* in this field contain the region base address. The value of *N* depends on the region size, as shown above. The remaining bits (*N*-1):5 are reserved. Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
4	VALID	WO	0	Region Number Valid Value Description 0 The **MPUNUMBER** register is not changed and the processor updates the base address for the region specified in the **MPUNUMBER** register and ignores the value of the REGION field. 1 The **MPUNUMBER** register is updated with the value of the REGION field and the base address is updated for the region specified in the REGION field. This bit is always read as 0.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	REGION	R/W	0x0	Region Number On a write, contains the value to be written to the **MPUNUMBER** register. On a read, returns the current region number in the **MPUNUMBER** register.

Register 51: MPU Region Attribute and Size (MPUATTR), offset 0xDA0

Register 52: MPU Region Attribute and Size Alias 1 (MPUATTR1), offset 0xDA8

Register 53: MPU Region Attribute and Size Alias 2 (MPUATTR2), offset 0xDB0

Register 54: MPU Region Attribute and Size Alias 3 (MPUATTR3), offset 0xDB8

Note: This register can only be accessed from privileged mode.

The **MPUATTR** register defines the region size and memory attributes of the MPU region specified by the **MPU Region Number (MPUNUMBER)** register and enables that region and any subregions.

The **MPUATTR** register is accessible using word or halfword accesses with the most-significant halfword holding the region attributes and the least-significant halfword holds the region size and the region and subregion enable bits.

The MPU access permission attribute bits, XN, AP, TEX, S, C, and B, control access to the corresponding memory region. If an access is made to an area of memory without the required permissions, then the MPU generates a permission fault.

The SIZE field defines the size of the MPU memory region specified by the **MPUNUMBER** register as follows:

(Region size in bytes) = $2^{(SIZE+1)}$

The smallest permitted region size is 32 bytes, corresponding to a SIZE value of 4. Table 3-9 on page 171 gives example SIZE values with the corresponding region size and value of N in the **MPU Region Base Address (MPUBASE)** register.

Table 3-9. Example SIZE Field Values

SIZE Encoding	Region Size	Value of N[a]	Note
00100b (0x4)	32 B	5	Minimum permitted size
01001b (0x9)	1 KB	10	-
10011b (0x13)	1 MB	20	-
11101b (0x1D)	1 GB	30	-
11111b (0x1F)	4 GB	No valid ADDR field in **MPUBASE**; the region occupies the complete memory map.	Maximum possible size

a. Refers to the N parameter in the **MPUBASE** register (see page 169).

MPU Region Attribute and Size (MPUATTR)

Base 0xE000.E000
Offset 0xDA0
Type R/W, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved			XN	reserved	AP			reserved		TEX			S	C	B
Type	RO	RO	RO	R/W	RO	R/W	R/W	R/W	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit/Field	Name	Type	Reset	Description
31:29	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	XN	R/W	0	Instruction Access Disable Value Description 0 Instruction fetches are enabled. 1 Instruction fetches are disabled.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26:24	AP	R/W	0	Access Privilege For information on using this bit field, see Table 3-5 on page 115.
23:22	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
21:19	TEX	R/W	0x0	Type Extension Mask For information on using this bit field, see Table 3-3 on page 114.
18	S	R/W	0	Shareable For information on using this bit, see Table 3-3 on page 114.
17	C	R/W	0	Cacheable For information on using this bit, see Table 3-3 on page 114.
16	B	R/W	0	Bufferable For information on using this bit, see Table 3-3 on page 114.
15:8	SRD	R/W	0x00	Subregion Disable Bits Value Description 0 The corresponding subregion is enabled. 1 The corresponding subregion is disabled. Region sizes of 128 bytes and less do not support subregions. When writing the attributes for such a region, configure the SRD field as 0x00. See the section called “Subregions” on page 113 for more information.
7:6	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:1	SIZE	R/W	0x0	Region Size Mask The SIZE field defines the size of the MPU memory region specified by the **MPUNUMBER** register. Refer to Table 3-9 on page 171 for more information.

Bit/Field	Name	Type	Reset	Description
0	ENABLE	R/W	0	Region Enable

Value	Description
0	The region is disabled.
1	The region is enabled.

4 JTAG Interface

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

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

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

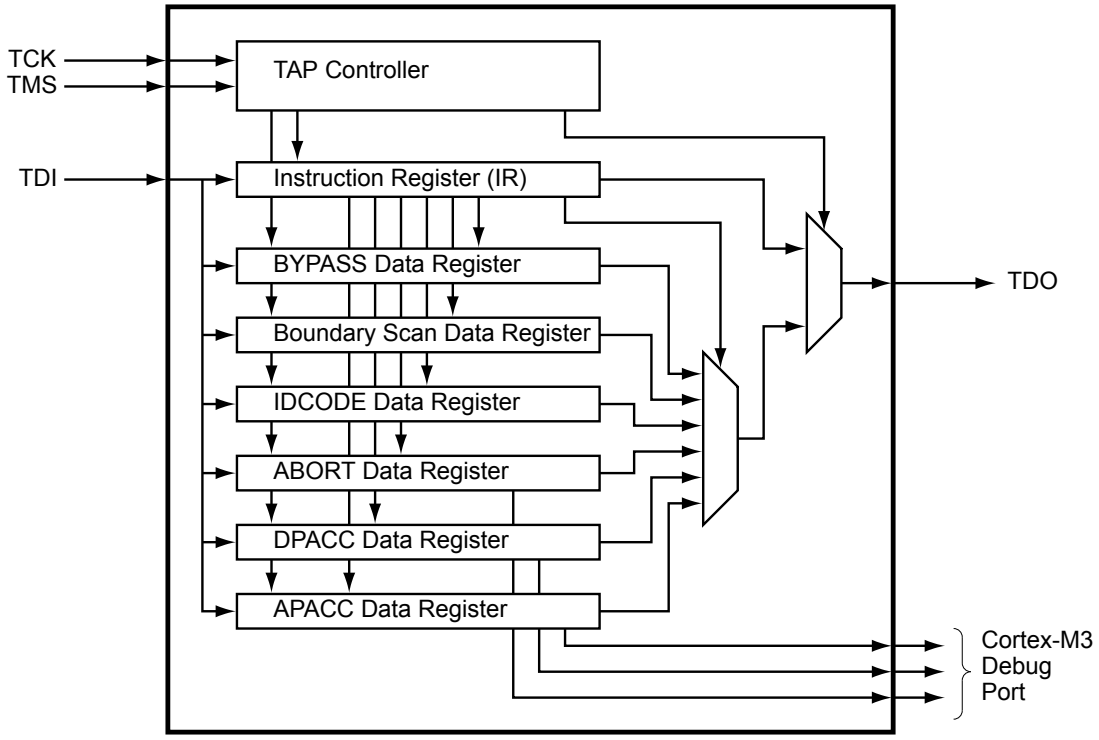
The Stellaris JTAG module has the following features:

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

See the *ARM® Debug Interface V5 Architecture Specification* for more information on the ARM JTAG controller.

4.1 Block Diagram

Figure 4-1. JTAG Module Block Diagram



4.2 Signal Description

The following table lists the external signals of the JTAG/SWD controller and describes the function of each. The JTAG/SWD controller signals are alternate functions for some GPIO signals, however note that the reset state of the pins is for the JTAG/SWD function. The JTAG/SWD controller signals are under commit protection and require a special process to be configured as GPIOs, see “Commit Control” on page 403. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placement for the JTAG/SWD controller signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 419) is set to choose the JTAG/SWD function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 437) to assign the JTAG/SWD controller signals to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 395.

Table 4-1. JTAG_SWD_SWO Signals (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
SWCLK	80	PC0 (3)	I	TTL	JTAG/SWD CLK.
SWDIO	79	PC1 (3)	I/O	TTL	JTAG TMS and SWDIO.
SWO	77	PC3 (3)	O	TTL	JTAG TDO and SWO.
TCK	80	PC0 (3)	I	TTL	JTAG/SWD CLK.
TDI	78	PC2 (3)	I	TTL	JTAG TDI.
TDO	77	PC3 (3)	O	TTL	JTAG TDO and SWO.

Table 4-1. JTAG_SWD_SWO Signals (100LQFP) *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
TMS	79	PC1 (3)	I	TTL	JTAG TMS and SWDIO.

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

Table 4-2. JTAG_SWD_SWO Signals (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
SWCLK	A9	PC0 (3)	I	TTL	JTAG/SWD CLK.
SWDIO	B9	PC1 (3)	I/O	TTL	JTAG TMS and SWDIO.
SWO	A10	PC3 (3)	O	TTL	JTAG TDO and SWO.
TCK	A9	PC0 (3)	I	TTL	JTAG/SWD CLK.
TDI	B8	PC2 (3)	I	TTL	JTAG TDI.
TDO	A10	PC3 (3)	O	TTL	JTAG TDO and SWO.
TMS	B9	PC1 (3)	I	TTL	JTAG TMS and SWDIO.

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

4.3 Functional Description

A high-level conceptual drawing of the JTAG module is shown in Figure 4-1 on page 175. The JTAG module is composed of the Test Access Port (TAP) controller and serial shift chains with parallel update registers. The TAP controller is a simple state machine controlled by the TCK and TMS inputs. The current state of the TAP controller depends on the sequence of values captured on TMS at the rising edge of TCK. The TAP controller determines when the serial shift chains capture new data, shift data from TDI towards TDO, and update the parallel load registers. The current state of the TAP controller also determines whether the Instruction Register (IR) chain or one of the Data Register (DR) chains is being accessed.

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

Some instructions, like EXTEST and INTEST, operate on data currently in a DR chain and do not capture, shift, or update any of the chains. Instructions that are not implemented decode to the BYPASS instruction to ensure that the serial path between TDI and TDO is always connected (see Table 4-4 on page 182 for a list of implemented instructions).

See “JTAG and Boundary Scan” on page 1193 for JTAG timing diagrams.

Note: Of all the possible reset sources, only Power-On reset (POR) and the assertion of the $\overline{\text{RST}}$ input have any effect on the JTAG module. The pin configurations are reset by both the $\overline{\text{RST}}$ input and POR, whereas the internal JTAG logic is only reset with POR. See “Reset Sources” on page 187 for more information on reset.

4.3.1 JTAG Interface Pins

The JTAG interface consists of four standard pins: TCK, TMS, TDI, and TDO. These pins and their associated state after a power-on reset or reset caused by the $\overline{\text{RST}}$ input are given in Table 4-3. Detailed information on each pin follows. Refer to “General-Purpose Input/Outputs (GPIOs)” on page 395 for information on how to reprogram the configuration of these pins.

Table 4-3. JTAG Port Pins State after Power-On Reset or $\overline{\text{RST}}$ assertion

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

4.3.1.1 Test Clock Input (TCK)

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

By default, the internal pull-up resistor on the TCK pin is enabled after reset, assuring that no clocking occurs if the pin is not driven from an external source. The internal pull-up and pull-down resistors can be turned off to save internal power as long as the TCK pin is constantly being driven by an external source (see page 425 and page 427).

4.3.1.2 Test Mode Select (TMS)

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

Holding TMS high for five consecutive TCK cycles drives the TAP controller state machine to the Test-Logic-Reset state. When the TAP controller enters the Test-Logic-Reset state, the JTAG module and associated registers are reset to their default values. This procedure should be performed to initialize the JTAG controller. The JTAG Test Access Port state machine can be seen in its entirety in Figure 4-2 on page 178.

By default, the internal pull-up resistor on the TMS pin is enabled after reset. Changes to the pull-up resistor settings on GPIO Port C should ensure that the internal pull-up resistor remains enabled on PC1/TMS; otherwise JTAG communication could be lost (see page 425).

4.3.1.3 Test Data Input (TDI)

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

By default, the internal pull-up resistor on the TDI pin is enabled after reset. Changes to the pull-up resistor settings on GPIO Port C should ensure that the internal pull-up resistor remains enabled on PC2/TDI; otherwise JTAG communication could be lost (see page 425).

4.3.1.4 Test Data Output (TDO)

The TDO pin provides an output stream of serial information from the IR chain or the DR chains. The value of TDO depends on the current TAP state, the current instruction, and the data in the

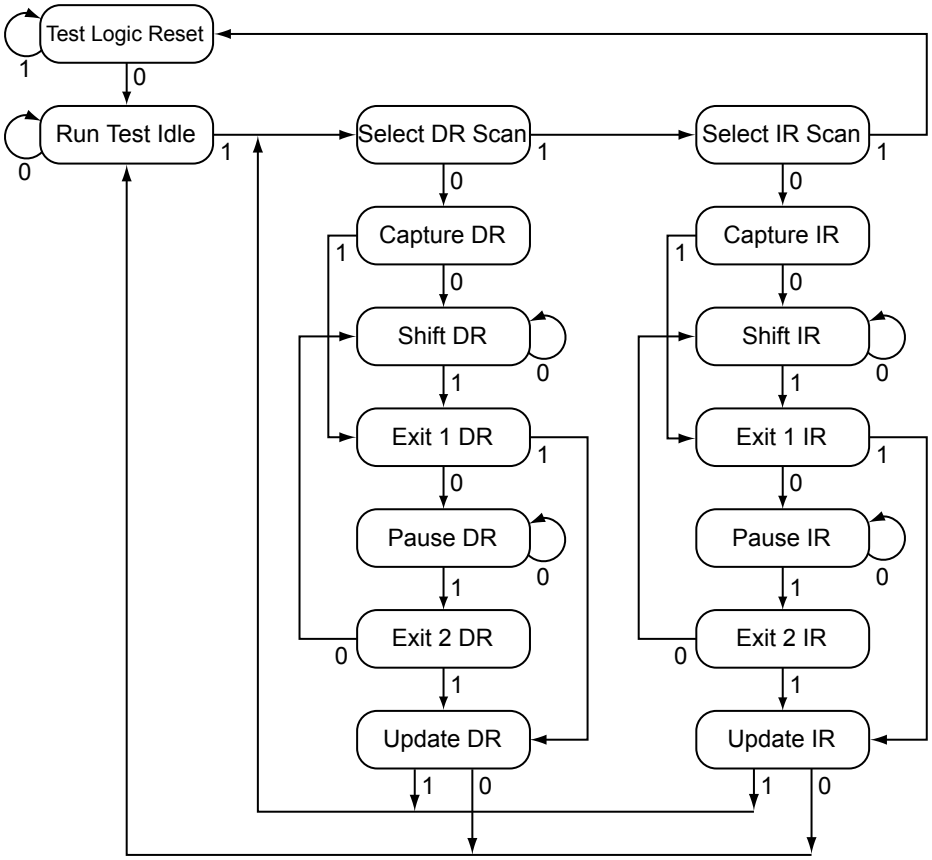
chain being accessed. In order to save power when the JTAG port is not being used, the TDO pin is placed in an inactive drive state when not actively shifting out data. Because TDO can be connected to the TDI of another controller in a daisy-chain configuration, the *IEEE Standard 1149.1* expects the value on TDO to change on the falling edge of TCK.

By default, the internal pull-up resistor on the TDO pin is enabled after reset, assuring that the pin remains at a constant logic level when the JTAG port is not being used. The internal pull-up and pull-down resistors can be turned off to save internal power if a High-Z output value is acceptable during certain TAP controller states (see page 425 and page 427).

4.3.2 JTAG TAP Controller

The JTAG TAP controller state machine is shown in Figure 4-2. The TAP controller state machine is reset to the Test-Logic-Reset state on the assertion of a Power-On-Reset (POR). In order to reset the JTAG module after the microcontroller has been powered on, the TMS input must be held HIGH for five TCK clock cycles, resetting the TAP controller and all associated JTAG chains. Asserting the correct sequence on the TMS pin allows the JTAG module to shift in new instructions, shift in data, or idle during extended testing sequences. For detailed information on the function of the TAP controller and the operations that occur in each state, please refer to *IEEE Standard 1149.1*.

Figure 4-2. Test Access Port State Machine



4.3.3 Shift Registers

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

this information to be shifted out on TDO during the TAP controller's SHIFT states. While the sampled data is being shifted out of the chain on TDO, new data is being shifted into the serial shift register on TDI. This new data is stored in the parallel load register during the TAP controller's UPDATE states. Each of the shift registers is discussed in detail in "Register Descriptions" on page 182.

4.3.4 Operational Considerations

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

4.3.4.1 GPIO Functionality

When the microcontroller is reset with either a POR or $\overline{\text{RST}}$, the JTAG/SWD port pins default to their JTAG/SWD configurations. The default configuration includes enabling digital functionality (DEN[3:0] set in the **Port C GPIO Digital Enable (GPIODEN)** register), enabling the pull-up resistors (PUE[3:0] set in the **Port C GPIO Pull-Up Select (GPIOPUR)** register), disabling the pull-down resistors (PDE[3:0] cleared in the **Port C GPIO Pull-Down Select (GPIOPDR)** register) and enabling the alternate hardware function (AFSEL[3:0] set in the **Port C GPIO Alternate Function Select (GPIOAFSEL)** register) on the JTAG/SWD pins. See page 419, page 425, page 427, and page 430.

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

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

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the NMI pin (PB7) and the four JTAG/SWD pins (PC[3:0]). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 419), **GPIO Pull Up Select (GPIOPUR)** register (see page 425), **GPIO Pull-Down Select (GPIOPDR)** register (see page 427), and **GPIO Digital Enable (GPIODEN)** register (see page 430) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 432) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 433) have been set.

4.3.4.2 Communication with JTAG/SWD

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

4.3.4.3 Recovering a "Locked" Microcontroller

Note: Performing the sequence below restores the non-volatile registers discussed in “Non-Volatile Register Programming” on page 295 to their factory default values. The mass erase of the Flash memory caused by the sequence below occurs prior to the non-volatile registers being restored.

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

1. Assert and hold the $\overline{\text{RST}}$ signal.
2. Apply power to the device.
3. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence on the section called “JTAG-to-SWD Switching” on page 181.
4. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence on the section called “SWD-to-JTAG Switching” on page 181.
5. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
6. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
7. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
8. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
9. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
10. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
11. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
12. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
13. Release the $\overline{\text{RST}}$ signal.
14. Wait 400 ms.
15. Power-cycle the microcontroller.

4.3.4.4 ARM Serial Wire Debug (SWD)

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

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

Stepping through this sequence of the TAP state machine enables the SWD interface and disables the JTAG interface. For more information on this operation and the SWD interface, see the *ARM® Debug Interface V5 Architecture Specification*.

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

JTAG-to-SWD Switching

To switch the operating mode of the Debug Access Port (DAP) from JTAG to SWD mode, the external debug hardware must send the switching preamble to the microcontroller. The 16-bit TMS command for switching to SWD mode is defined as b1110.0111.1001.1110, transmitted LSB first. This command can also be represented as 0xE79E when transmitted LSB first. The complete switch sequence should consist of the following transactions on the TCK/SWCLK and TMS/SWDIO signals:

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

SWD-to-JTAG Switching

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

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

4.4 Initialization and Configuration

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

4.5 Register Descriptions

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

4.5.1 Instruction Register (IR)

The JTAG TAP Instruction Register (IR) is a four-bit serial scan chain connected between the JTAG TDI and TDO pins with a parallel load register. When the TAP Controller is placed in the correct states, bits can be shifted into the IR. Once these bits have been shifted into the chain and updated, they are interpreted as the current instruction. The decode of the IR bits is shown in Table 4-4. A detailed explanation of each instruction, along with its associated Data Register, follows.

Table 4-4. JTAG Instruction Register Commands

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

4.5.1.1 EXTEST Instruction

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

4.5.1.2 INTEST Instruction

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

While the INTEST instruction is present in the Instruction Register, the Boundary Scan Data Register can be accessed to sample and shift out the current data and load new data into the Boundary Scan Data Register.

4.5.1.3 SAMPLE/PRELOAD Instruction

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

While these samples of the inputs, outputs, and output enables are being shifted out of the Boundary Scan Data Register, new data is being shifted into the Boundary Scan Data Register from TDI. Once the new data has been shifted into the Boundary Scan Data Register, the data is saved in the parallel load registers when the TAP controller enters the Update DR state. This update of the parallel load register preloads data into the Boundary Scan Data Register that is associated with each input, output, and output enable. This preloaded data can be used with the EXTEST and INTEST instructions to drive data into or out of the controller. See “Boundary Scan Data Register” on page 184 for more information.

4.5.1.4 ABORT Instruction

The ABORT instruction connects the associated ABORT Data Register chain between TDI and TDO. This instruction provides read and write access to the ABORT Register of the ARM Debug Access Port (DAP). Shifting the proper data into this Data Register clears various error bits or initiates a DAP abort of a previous request. See the “ABORT Data Register” on page 185 for more information.

4.5.1.5 DPACC Instruction

The DPACC instruction connects the associated DPACC Data Register chain between TDI and TDO. This instruction provides read and write access to the DPACC Register of the ARM Debug Access Port (DAP). Shifting the proper data into this register and reading the data output from this register allows read and write access to the ARM debug and status registers. See “DPACC Data Register” on page 185 for more information.

4.5.1.6 APACC Instruction

The APACC instruction connects the associated APACC Data Register chain between TDI and TDO. This instruction provides read and write access to the APACC Register of the ARM Debug Access Port (DAP). Shifting the proper data into this register and reading the data output from this register allows read and write access to internal components and buses through the Debug Port. See “APACC Data Register” on page 185 for more information.

4.5.1.7 IDCODE Instruction

The IDCODE instruction connects the associated IDCODE Data Register chain between TDI and TDO. This instruction provides information on the manufacturer, part number, and version of the ARM core. This information can be used by testing equipment and debuggers to automatically configure input and output data streams. IDCODE is the default instruction loaded into the JTAG Instruction Register when a Power-On-Reset (POR) is asserted, or the Test-Logic-Reset state is entered. See “IDCODE Data Register” on page 184 for more information.

4.5.1.8 BYPASS Instruction

The BYPASS instruction connects the associated BYPASS Data Register chain between TDI and TDO. This instruction is used to create a minimum length serial path between the TDI and TDO ports. The BYPASS Data Register is a single-bit shift register. This instruction improves test efficiency by allowing components that are not needed for a specific test to be bypassed in the JTAG scan chain by loading them with the BYPASS instruction. See “BYPASS Data Register” on page 184 for more information.

4.5.2 Data Registers

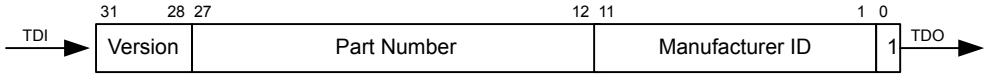
The JTAG module contains six Data Registers. These serial Data Register chains include: IDCODE, BYPASS, Boundary Scan, APACC, DPACC, and ABORT and are discussed in the following sections.

4.5.2.1 IDCODE Data Register

The format for the 32-bit IDCODE Data Register defined by the *IEEE Standard 1149.1* is shown in Figure 4-3. The standard requires that every JTAG-compliant microcontroller implement either the IDCODE instruction or the BYPASS instruction as the default instruction. The LSB of the IDCODE Data Register is defined to be a 1 to distinguish it from the BYPASS instruction, which has an LSB of 0. This definition allows auto-configuration test tools to determine which instruction is the default instruction.

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

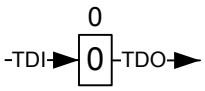
Figure 4-3. IDCODE Register Format



4.5.2.2 BYPASS Data Register

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

Figure 4-4. BYPASS Register Format



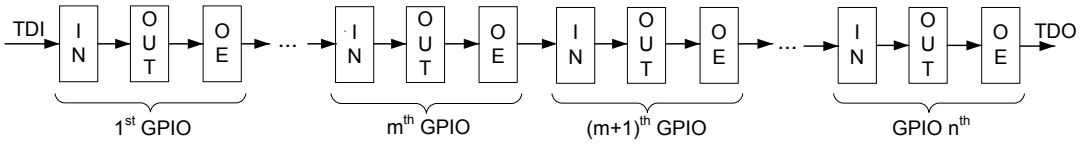
4.5.2.3 Boundary Scan Data Register

The format of the Boundary Scan Data Register is shown in Figure 4-5. Each GPIO pin, starting with a GPIO pin next to the JTAG port pins, is included in the Boundary Scan Data Register. Each

GPIO pin has three associated digital signals that are included in the chain. These signals are input, output, and output enable, and are arranged in that order as shown in the figure.

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

Figure 4-5. Boundary Scan Register Format



4.5.2.4 APACC Data Register

The format for the 35-bit APACC Data Register defined by ARM is described in the *ARM® Debug Interface V5 Architecture Specification*.

4.5.2.5 DPACC Data Register

The format for the 35-bit DPACC Data Register defined by ARM is described in the *ARM® Debug Interface V5 Architecture Specification*.

4.5.2.6 ABORT Data Register

The format for the 35-bit ABORT Data Register defined by ARM is described in the *ARM® Debug Interface V5 Architecture Specification*.

5 System Control

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

5.1 Signal Description

The following table lists the external signals of the System Control module and describes the function of each. The NMI signal is the alternate function for the GPIO PB7 signal and functions as a GPIO after reset. PB7 is under commit protection and requires a special process to be configured as any alternate function or to subsequently return to the GPIO function, see “Commit Control” on page 403. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placement for the NMI signal. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 419) should be set to choose the NMI function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 437) to assign the NMI signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 395. The remaining signals (with the word "fixed" in the Pin Mux/Pin Assignment column) have a fixed pin assignment and function.

Table 5-1. System Control & Clocks Signals (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
NMI	89	PB7 (4)	I	TTL	Non-maskable interrupt.
OSC0	48	fixed	I	Analog	Main oscillator crystal input or an external clock reference input.
OSC1	49	fixed	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
RST	64	fixed	I	TTL	System reset input.

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

Table 5-2. System Control & Clocks Signals (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
NMI	A8	PB7 (4)	I	TTL	Non-maskable interrupt.
OSC0	L11	fixed	I	Analog	Main oscillator crystal input or an external clock reference input.
OSC1	M11	fixed	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
RST	H11	fixed	I	TTL	System reset input.

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

5.2 Functional Description

The System Control module provides the following capabilities:

- Device identification, see “Device Identification” on page 187
- Local control, such as reset (see “Reset Control” on page 187), power (see “Power Control” on page 192) and clock control (see “Clock Control” on page 193)

- System control (Run, Sleep, and Deep-Sleep modes), see “System Control” on page 199

5.2.1 Device Identification

Several read-only registers provide software with information on the microcontroller, such as version, part number, SRAM size, Flash memory size, and other features. See the **DID0** (page 204), **DID1** (page 231), **DC0**-**DC9** (page 233) and **NVMSTAT** (page 254) registers.

5.2.2 Reset Control

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

5.2.2.1 Reset Sources

The LM3S9U81 microcontroller has six sources of reset:

1. Power-on reset (POR) (see page 188).
2. External reset input pin ($\overline{\text{RST}}$) assertion (see page 188).
3. Internal brown-out (BOR) detector (see page 190).
4. Software-initiated reset (with the software reset registers) (see page 190).
5. A watchdog timer reset condition violation (see page 191).
6. MOSC failure (see page 192).

Table 5-3 provides a summary of results of the various reset operations.

Table 5-3. Reset Sources

Reset Source	Core Reset?	JTAG Reset?	On-Chip Peripherals Reset?
Power-On Reset	Yes	Yes	Yes
$\overline{\text{RST}}$	Yes	Yes	Yes
Brown-Out Reset	Yes	Yes	Yes
Software System Request Reset using the SYSRESREQ bit in the **APINT** register.	Yes	Yes	Yes
Software System Request Reset using the VECTRESET bit in the **APINT** register.	Yes	No	No
Software Peripheral Reset	No	Yes	Yes[a]
Watchdog Reset	Yes	Yes	Yes
MOSC Failure Reset	Yes	Yes	Yes

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

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

At any reset that resets the core, the user has the opportunity to direct the core to execute the ROM Boot Loader or the application in Flash memory by using any GPIO signal as configured in the **Boot Configuration (BOOTCFG)** register.

At reset, the ROM is mapped over the Flash memory so that the ROM boot sequence is always executed. The boot sequence executed from ROM is as follows:

1. The BA bit (below) is cleared such that ROM is mapped to 0x01xx.xxxx and Flash memory is mapped to address 0x0.
2. The **BOOTCFG** register is read. If the EN bit is clear, the status of the specified GPIO pin is compared with the specified polarity. If the status matches the specified polarity, the ROM is mapped to address 0x0000.0000 and execution continues out of the ROM Boot Loader.
3. If the status doesn't match the specified polarity, the data at address 0x0000.0004 is read, and if the data at this address is 0xFFFF.FFFF, the ROM is mapped to address 0x0000.0000 and execution continues out of the ROM Boot Loader.
4. If there is valid data at address 0x0000.0004, the stack pointer (**SP**) is loaded from Flash memory at address 0x0000.0000 and the program counter (**PC**) is loaded from address 0x0000.0004. The user application begins executing.

For example, if the **BOOTCFG** register is written and committed with the value of 0x0000.3C01, then PB7 is examined at reset to determine if the ROM Boot Loader should be executed. If PB7 is Low, the core unconditionally begins executing the ROM boot loader. If PB7 is High, then the application in Flash memory is executed if the reset vector at location 0x0000.0004 is not 0xFFFF.FFFF. Otherwise, the ROM boot loader is executed.

5.2.2.2 Power-On Reset (POR)

The internal Power-On Reset (POR) circuit monitors the power supply voltage (V_{DD}) and generates a reset signal to all of the internal logic including JTAG when the power supply ramp reaches a threshold value (V_{TH}). The microcontroller must be operating within the specified operating parameters when the on-chip power-on reset pulse is complete (see “Power and Brown-Out” on page 1195). For applications that require the use of an external reset signal to hold the microcontroller in reset longer than the internal POR, the $\overline{\text{RST}}$ input may be used as discussed in “External $\overline{\text{RST}}$ Pin” on page 188.

The Power-On Reset sequence is as follows:

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

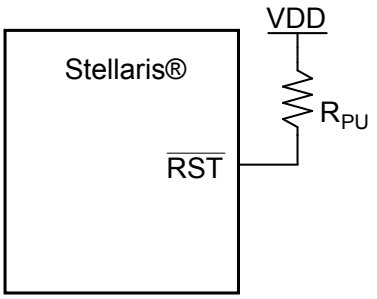
The internal POR is only active on the initial power-up of the microcontroller. The Power-On Reset timing is shown in Figure 24-4 on page 1195.

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

Note: It is recommended that the trace for the $\overline{\text{RST}}$ signal must be kept as short as possible. Be sure to place any components connected to the $\overline{\text{RST}}$ signal as close to the microcontroller as possible.

If the application only uses the internal POR circuit, the $\overline{\text{RST}}$ input must be connected to the power supply (V_{DD}) through an optional pull-up resistor (0 to 100K Ω) as shown in Figure 5-1 on page 189.

Figure 5-1. Basic $\overline{\text{RST}}$ Configuration



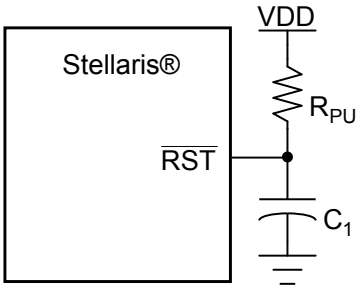
R_{PU} = 0 to 100 kΩ

The external reset pin ($\overline{\text{RST}}$) resets the microcontroller including the core and all the on-chip peripherals except the JTAG TAP controller (see "JTAG Interface" on page 174). The external reset sequence is as follows:

1. The external reset pin ($\overline{\text{RST}}$) is asserted for the duration specified by T_{MIN} and then de-asserted (see "Reset" on page 1196).
2. The internal reset is released and the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

To improve noise immunity and/or to delay reset at power up, the $\overline{\text{RST}}$ input may be connected to an RC network as shown in Figure 5-2 on page 189.

Figure 5-2. External Circuitry to Extend Power-On Reset

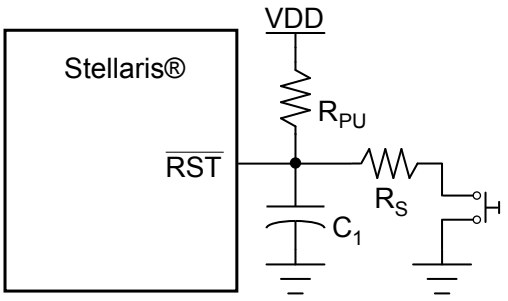


R_{PU} = 1 kΩ to 100 kΩ

C_1 = 1 nF to 10 µF

If the application requires the use of an external reset switch, Figure 5-3 on page 190 shows the proper circuitry to use.

Figure 5-3. Reset Circuit Controlled by Switch



Typical R_{PU} = 10 kΩ

Typical R_S = 470 Ω

C_1 = 10 nF

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

The external reset timing is shown in Figure 24-7 on page 1196.

5.2.2.4 Brown-Out Reset (BOR)

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

The brown-out reset sequence is as follows:

1. When V_{DD} drops below V_{BTH}, an internal BOR condition is set.
2. If the BOR condition exists, an internal reset is asserted.
3. The internal reset is released and the microcontroller fetches and loads the initial stack pointer, the initial program counter, the first instruction designated by the program counter, and begins execution.
4. The internal BOR condition is reset after 500 µs to prevent another BOR condition from being set before software has a chance to investigate the original cause.

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

The internal Brown-Out Reset timing is shown in Figure 24-5 on page 1195.

5.2.2.5 Software Reset

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

Peripherals can be individually reset by software via three registers that control reset signals to each on-chip peripheral (see the **SRCRn** registers, page 281). If the bit position corresponding to a peripheral is set and subsequently cleared, the peripheral is reset. The encoding of the reset registers is consistent with the encoding of the clock gating control for peripherals and on-chip functions (see "System Control" on page 199).

The entire microcontroller, including the core, can be reset by software by setting the SYSRESREQ bit in the **Application Interrupt and Reset Control (APINT)** register. The software-initiated system reset sequence is as follows:

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

The core only can be reset by software by setting the VECTRESET bit in the **APINT** register. The software-initiated core reset sequence is as follows:

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

The software-initiated system reset timing is shown in Figure 24-8 on page 1196.

5.2.2.6 Watchdog Timer Reset

The Watchdog Timer module's function is to prevent system hangs. The LM3S9U81 microcontroller has two Watchdog Timer modules in case one watchdog clock source fails. One watchdog is run off the system clock and the other is run off the Precision Internal Oscillator (PIOSC). Each module operates in the same manner except that because the PIOSC watchdog timer module is in a different clock domain, register accesses must have a time delay between them. The watchdog timer can be configured to generate an interrupt to the microcontroller on its first time-out and to generate a reset on its second time-out.

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

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

For more information on the Watchdog Timer module, see "Watchdog Timers" on page 573.

The watchdog reset timing is shown in Figure 24-9 on page 1197.

5.2.3 Non-Maskable Interrupt

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

- The assertion of the NMI signal
- A main oscillator verification error
- The NMISET bit in the **Interrupt Control and State (INTCTRL)** register in the Cortex™-M3 (see page 139).

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

5.2.3.1 NMI Pin

The NMI signal is the alternate function for GPIO port pin PB7. The alternate function must be enabled in the GPIO for the signal to be used as an interrupt, as described in "General-Purpose Input/Outputs (GPIOs)" on page 395. Note that enabling the NMI alternate function requires the use of the GPIO lock and commit function just like the GPIO port pins associated with JTAG/SWD functionality, see page 433. The active sense of the NMI signal is High; asserting the enabled NMI signal above V_{IH} initiates the NMI interrupt sequence.

5.2.3.2 Main Oscillator Verification Failure

The LM3S9U81 microcontroller provides a main oscillator verification circuit that generates an error condition if the oscillator is running too fast or too slow. If the main oscillator verification circuit is enabled and a failure occurs, a power-on reset is generated and control is transferred to the NMI handler. The NMI handler is used to address the main oscillator verification failure because the necessary code can be removed from the general reset handler, speeding up reset processing. The detection circuit is enabled by setting the CVAL bit in the **Main Oscillator Control (MOSCCTL)** register. The main oscillator verification error is indicated in the main oscillator fail status (MOSCFAIL) bit in the **Reset Cause (RESC)** register. The main oscillator verification circuit action is described in more detail in "Main Oscillator Verification Circuit" on page 199.

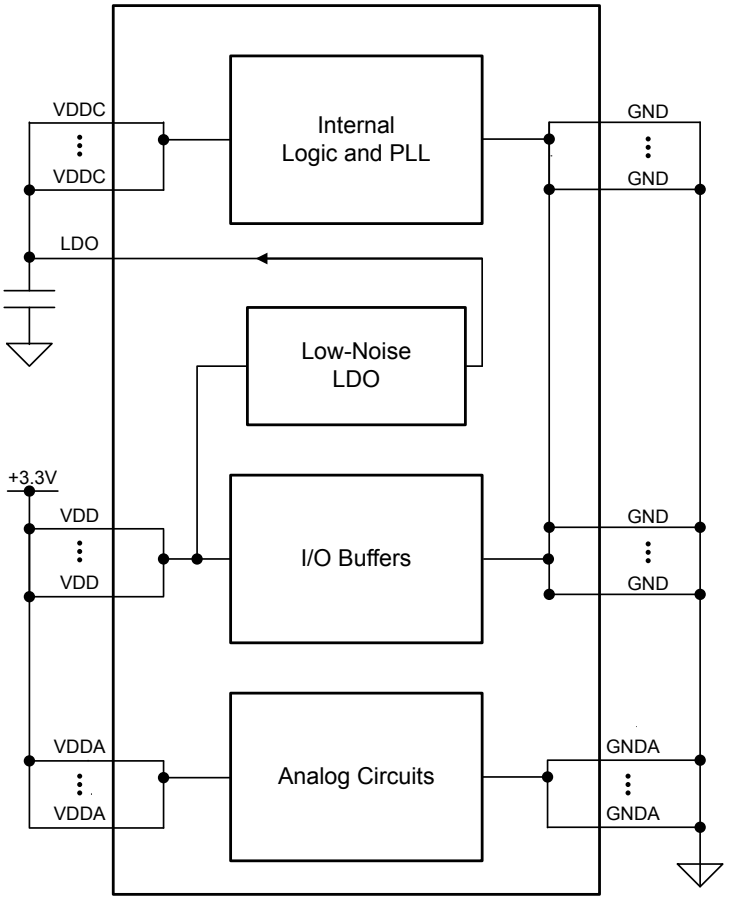
5.2.4 Power Control

The Stellaris® microcontroller provides an integrated LDO regulator that is used to provide power to the majority of the microcontroller's internal logic. Figure 5-4 shows the power architecture.

An external LDO may not be used.

Note: VDDA must be supplied with a voltage that meets the specification in Table 24-2 on page 1192, or the microcontroller does not function properly. VDDA is the supply for all of the analog circuitry on the device, including the clock circuitry.

Figure 5-4. Power Architecture



5.2.5 Clock Control

System control determines the control of clocks in this part.

5.2.5.1 Fundamental Clock Sources

There are multiple clock sources for use in the microcontroller:

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

- **Main Oscillator (MOSC).** The main oscillator provides a frequency-accurate clock source by one of two means: an external single-ended clock source is connected to the OSC0 input pin, or an external crystal is connected across the OSC0 input and OSC1 output pins. If the PLL is being used, the crystal value must be one of the supported frequencies between 3.579545 MHz to 16.384 MHz (inclusive). If the PLL is not being used, the crystal may be any one of the supported frequencies between 1 MHz to 16.384 MHz. The single-ended clock source range is from DC

through the specified speed of the microcontroller. The supported crystals are listed in the XTAL bit field in the **RCC** register (see page 215). Note that the MOSC provides the clock source for the USB PLL and must be connected to a crystal or an oscillator.

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

The internal system clock (SysClk), is derived from any of the above sources plus two others: the output of the main internal PLL and the precision internal oscillator divided by four (4 MHz ± 1%). The frequency of the PLL clock reference must be in the range of 3.579545 MHz to 16.384 MHz (inclusive). Table 5-4 on page 194 shows how the various clock sources can be used in a system.

Table 5-4. Clock Source Options

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

5.2.5.2 Clock Configuration

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

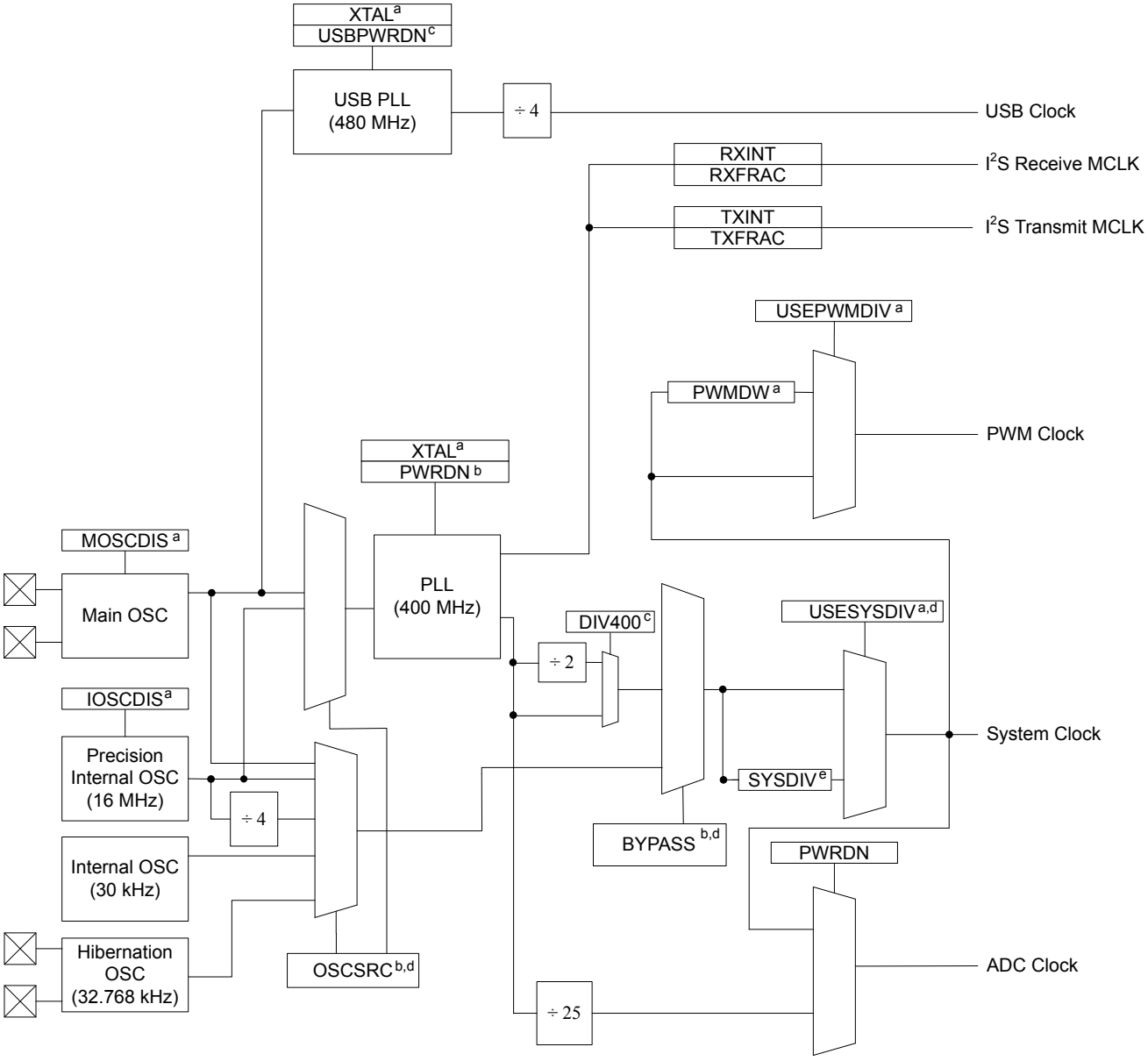
- Source of clocks in sleep and deep-sleep modes
- System clock derived from PLL or other clock source
- Enabling/disabling of oscillators and PLL
- Clock divisors
- Crystal input selection

Important: Write the **RCC** register prior to writing the **RCC2** register. If a subsequent write to the **RCC** register is required, include another register access after writing the **RCC** register and before writing the **RCC2** register.

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

Note: When the ADC module is in operation, the system clock must be at least 16 MHz. When the USB module is in operation, MOSC must be the clock source, either with or without using the PLL, and the system clock must be at least 30 MHz.

Figure 5-5. Main Clock Tree



a. Control provided by **RCC** register bit/field.
b. Control provided by **RCC** register bit/field or **RCC2** register bit/field, if overridden with **RCC2** register bit USERCC2.
c. Control provided by **RCC2** register bit/field.
d. Also may be controlled by **DSLPCLKCFG** when in deep sleep mode.
e. Control provided by **RCC** register SYSDIV field, **RCC2** register SYSDIV2 field if overridden with USERCC2 bit, or [SYSDIV2,SYSDIV2LSB] if both USERCC2 and DIV400 bits are set.

Note: The figure above shows all features available on all Stellaris® Firestorm-class microcontrollers. Not all peripherals may be available on this device.

Using the SYSDIV and SYSDIV2 Fields

In the **RCC** register, the SYSDIV field specifies which divisor is used to generate the system clock from either the PLL output or the oscillator source (depending on how the BYPASS bit in this register

is configured). When using the PLL, the VCO frequency of 400 MHz is predivided by 2 before the divisor is applied. Table 5-5 shows how the SYSDIV encoding affects the system clock frequency, depending on whether the PLL is used (BYPASS=0) or another clock source is used (BYPASS=1). The divisor is equivalent to the SYSDIV encoding plus 1. For a list of possible clock sources, see Table 5-4 on page 194.

Table 5-5. Possible System Clock Frequencies Using the SYSDIV Field

SYSDIV	Divisor	Frequency (BYPASS=0)	Frequency (BYPASS=1)	StellarisWare® Parameter[a]
0x0	/1	reserved	Clock source frequency/2	SYSCTL_SYSDIV_1[b]
0x1	/2	reserved	Clock source frequency/2	SYSCTL_SYSDIV_2
0x2	/3	66.67 MHz	Clock source frequency/3	SYSCTL_SYSDIV_3
0x3	/4	50 MHz	Clock source frequency/4	SYSCTL_SYSDIV_4
0x4	/5	40 MHz	Clock source frequency/5	SYSCTL_SYSDIV_5
0x5	/6	33.33 MHz	Clock source frequency/6	SYSCTL_SYSDIV_6
0x6	/7	28.57 MHz	Clock source frequency/7	SYSCTL_SYSDIV_7
0x7	/8	25 MHz	Clock source frequency/8	SYSCTL_SYSDIV_8
0x8	/9	22.22 MHz	Clock source frequency/9	SYSCTL_SYSDIV_9
0x9	/10	20 MHz	Clock source frequency/10	SYSCTL_SYSDIV_10
0xA	/11	18.18 MHz	Clock source frequency/11	SYSCTL_SYSDIV_11
0xB	/12	16.67 MHz	Clock source frequency/12	SYSCTL_SYSDIV_12
0xC	/13	15.38 MHz	Clock source frequency/13	SYSCTL_SYSDIV_13
0xD	/14	14.29 MHz	Clock source frequency/14	SYSCTL_SYSDIV_14
0xE	/15	13.33 MHz	Clock source frequency/15	SYSCTL_SYSDIV_15
0xF	/16	12.5 MHz (default)	Clock source frequency/16	SYSCTL_SYSDIV_16

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

b. SYSCTL_SYSDIV_1 does not set the USESYSDIV bit. As a result, using this parameter without enabling the PLL results in the system clock having the same frequency as the clock source.

The SYSDIV2 field in the **RCC2** register is 2 bits wider than the SYSDIV field in the **RCC** register so that additional larger divisors up to /64 are possible, allowing a lower system clock frequency for improved Deep Sleep power consumption. When using the PLL, the VCO frequency of 400 MHz is predivided by 2 before the divisor is applied. The divisor is equivalent to the SYSDIV2 encoding plus 1. Table 5-6 shows how the SYSDIV2 encoding affects the system clock frequency, depending on whether the PLL is used (BYPASS2=0) or another clock source is used (BYPASS2=1). For a list of possible clock sources, see Table 5-4 on page 194.

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

SYSDIV2	Divisor	Frequency (BYPASS2=0)	Frequency (BYPASS2=1)	StellarisWare Parameter[a]
0x00	/1	reserved	Clock source frequency/2	SYSCTL_SYSDIV_1[b]
0x01	/2	reserved	Clock source frequency/2	SYSCTL_SYSDIV_2
0x02	/3	66.67 MHz	Clock source frequency/3	SYSCTL_SYSDIV_3
0x03	/4	50 MHz	Clock source frequency/4	SYSCTL_SYSDIV_4
0x04	/5	40 MHz	Clock source frequency/5	SYSCTL_SYSDIV_5
...	...	...	...	...
0x09	/10	20 MHz	Clock source frequency/10	SYSCTL_SYSDIV_10
...	...	...	...	...

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

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

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

b. SYSCTL_SYSDIV_1 does not set the USESYSDIV bit. As a result, using this parameter without enabling the PLL results in the system clock having the same frequency as the clock source.

To allow for additional frequency choices when using the PLL, the DIV400 bit is provided along with the SYSDIV2LSB bit. When the DIV400 bit is set, bit 22 becomes the LSB for SYSDIV2. In this situation, the divisor is equivalent to the (SYSDIV2 encoding with SYSDIV2LSB appended) plus one. Table 5-7 shows the frequency choices when DIV400 is set. When the DIV400 bit is clear, SYSDIV2LSB is ignored, and the system clock frequency is determined as shown in Table 5-6 on page 196.

Table 5-7. Examples of Possible System Clock Frequencies with DIV400=1

SYSDIV2	SYSDIV2LSB	Divisor	Frequency (BYPASS2=0)[a]	StellarisWare Parameter[b]
0x00	reserved	/2	reserved	-
0x01	0	/3	reserved	-
	1	/4	reserved	-
0x02	0	/5	80 MHz	SYSCTL_SYSDIV_2_5
	1	/6	66.67 MHz	SYSCTL_SYSDIV_3
0x03	0	/7	reserved	-
	1	/8	50 MHz	SYSCTL_SYSDIV_4
0x04	0	/9	44.44 MHz	SYSCTL_SYSDIV_4_5
	1	/10	40 MHz	SYSCTL_SYSDIV_5
...	...	...	...	...
0x3F	0	/127	3.15 MHz	SYSCTL_SYSDIV_63_5
	1	/128	3.125 MHz	SYSCTL_SYSDIV_64

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

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

5.2.5.3 Precision Internal Oscillator Operation (PIOSC)

The microcontroller powers up with the PIOSC running. If another clock source is desired, the PIOSC must remain enabled as it is used for internal functions. The PIOSC can only be disabled during Deep-Sleep mode. It can be powered down by setting the IOSCDIS bit in the **RCC** register.

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

- Default calibration: clear the UTEN bit and set the UPDATE bit in the **Precision Internal Oscillator Calibration (PIOSCCAL)** register.
- User-defined calibration: The user can program the UT value to adjust the PIOSC frequency. As the UT value increases, the generated period increases. To commit a new UT value, first set the

UTEN bit, then program the UT field, and then set the UPDATE bit. The adjustment finishes within a few clock periods and is glitch free.

5.2.5.4 Crystal Configuration for the Main Oscillator (MOSC)

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

The XTAL bit in the **RCC** register (see page 215) describes the available crystal choices and default programming values.

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

5.2.5.5 Main PLL Frequency Configuration

The main PLL is disabled by default during power-on reset and is enabled later by software if required. Software specifies the output divisor to set the system clock frequency and enables the main PLL to drive the output. The PLL operates at 400 MHz, but is divided by two prior to the application of the output divisor, unless the DIV400 bit in the **RCC2** register is set.

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

If the main oscillator provides the clock reference to the main PLL, the translation provided by hardware and used to program the PLL is available for software in the **XTAL to PLL Translation (PLLCFG)** register (see page 219). The internal translation provides a translation within ± 1% of the targeted PLL VCO frequency. Table 24-8 on page 1198 shows the actual PLL frequency and error for a given crystal choice.

The Crystal Value field (XTAL) in the **Run-Mode Clock Configuration (RCC)** register (see page 215) describes the available crystal choices and default programming of the **PLLCFG** register. Any time the XTAL field changes, the new settings are translated and the internal PLL settings are updated.

5.2.5.6 USB PLL Frequency Configuration

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

5.2.5.7 PLL Modes

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

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

The modes are programmed using the **RCC**/**RCC2** register fields (see page 215 and page 222).

5.2.5.8 PLL Operation

If a PLL configuration is changed, the PLL output frequency is unstable until it reconverges (relocks) to the new setting. The time between the configuration change and relock is T_{READY} (see Table 24-7 on page 1197). During the relock time, the affected PLL is not usable as a clock reference.

Either PLL is changed by one of the following:

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

A counter clocked by the system clock is used to measure the T_{READY} requirement. If the system clock is the main oscillator and it is running off an 8.192 MHz or slower external oscillator clock, the down counter is set to 0x1200 (that is, ~600 µs at an 8.192 MHz). If the system clock is running off the PIOSC or an external oscillator clock that is faster than 8.192 MHz, the down counter is set to 0x2400. Hardware is provided to keep the PLL from being used as a system clock until the T_{READY} condition is met after one of the two changes above. It is the user's responsibility to have a stable clock source (like the main oscillator) before the **RCC**/**RCC2** register is switched to use the PLL.

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

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

5.2.5.9 Main Oscillator Verification Circuit

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

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

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

5.2.6 System Control

For power-savings purposes, the **RCGCn**, **SCGCn**, and **DCGCn** registers control the clock gating logic for each peripheral or block in the system while the microcontroller is in Run, Sleep, and Deep-Sleep mode, respectively. These registers are located in the System Control register map

starting at offsets 0x600, 0x700, and 0x800, respectively. There must be a delay of 3 system clocks after a peripheral module clock is enabled in the **RCGC** register before any module registers are accessed.

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

- Run mode
- Sleep mode
- Deep-Sleep mode

The following sections describe the different modes in detail.

Caution – If the Cortex-M3 Debug Access Port (DAP) has been enabled, and the device wakes from a low power sleep or deep-sleep mode, the core may start executing code before all clocks to peripherals have been restored to their Run mode configuration. The DAP is usually enabled by software tools accessing the JTAG or SWD interface when debugging or flash programming. If this condition occurs, a Hard Fault is triggered when software accesses a peripheral with an invalid clock.

A software delay loop can be used at the beginning of the interrupt routine that is used to wake up a system from a WFI (Wait For Interrupt) instruction. This stalls the execution of any code that accesses a peripheral register that might cause a fault. This loop can be removed for production software as the DAP is most likely not enabled during normal execution.

Because the DAP is disabled by default (power on reset), the user can also power cycle the device. The DAP is not enabled unless it is enabled through the JTAG or SWD interface.

5.2.6.1 Run Mode

In Run mode, the microcontroller actively executes code. Run mode provides normal operation of the processor and all of the peripherals that are currently enabled by the **RCGCn** registers. The system clock can be any of the available clock sources including the PLL.

5.2.6.2 Sleep Mode

In Sleep mode, the clock frequency of the active peripherals is unchanged, but the processor and the memory subsystem are not clocked and therefore no longer execute code. Sleep mode is entered by the Cortex-M3 core executing a `WFI` (Wait for Interrupt) instruction. Any properly configured interrupt event in the system brings the processor back into Run mode. See “Power Management” on page 103 for more details.

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

5.2.6.3 Deep-Sleep Mode

In Deep-Sleep mode, the clock frequency of the active peripherals may change (depending on the Run mode clock configuration) in addition to the processor clock being stopped. An interrupt returns the microcontroller to Run mode from one of the sleep modes; the sleep modes are entered on request from the code. Deep-Sleep mode is entered by first setting the SLEEPDEEP bit in the **System Control (SYSCTRL)** register (see page 145) and then executing a WFI instruction. Any properly configured interrupt event in the system brings the processor back into Run mode. See “Power Management” on page 103 for more details.

The Cortex-M3 processor core and the memory subsystem are not clocked in Deep-Sleep mode. Peripherals are clocked that are enabled in the **DCGCn** registers when auto-clock gating is enabled (see the **RCC** register) or the **RCGCn** registers when auto-clock gating is disabled. The system clock source is specified in the **DSLPCLKCFG** register. When the **DSLPCLKCFG** register is used, the internal oscillator source is powered up, if necessary, and other clocks are powered down. If the PLL is running at the time of the WFI instruction, hardware powers the PLL down and overrides the SYSDIV field of the active **RCC**/**RCC2** register, to be determined by the DSDIVORIDE setting in the **DSLPCLKCFG** register, up to /16 or /64 respectively. When the Deep-Sleep exit event occurs, hardware brings the system clock back to the source and frequency it had at the onset of Deep-Sleep mode before enabling the clocks that had been stopped during the Deep-Sleep duration. If the PIOSC is used as the PLL reference clock source, it may continue to provide the clock during Deep-Sleep. See page 226.

5.3 Initialization and Configuration

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

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

5.4 Register Map

Table 5-8 on page 201 lists the System Control registers, grouped by function. The offset listed is a hexadecimal increment to the register's address, relative to the System Control base address of 0x400F.E000.

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

Additional Flash and ROM registers defined in the System Control register space are described in the “Internal Memory” on page 288.

Table 5-8. System Control Register Map

Offset	Name	Type	Reset	Description	See page
0x000	DID0	RO	-	Device Identification 0	204
0x004	DID1	RO	-	Device Identification 1	231

Table 5-8. System Control Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x008	DC0	RO	0x017F.00BF	Device Capabilities 0	233
0x010	DC1	RO	-	Device Capabilities 1	234
0x014	DC2	RO	0x570F.5037	Device Capabilities 2	237
0x018	DC3	RO	0xBFFF.7FC0	Device Capabilities 3	239
0x01C	DC4	RO	0x5004.F1FF	Device Capabilities 4	241
0x020	DC5	RO	0x0000.0000	Device Capabilities 5	243
0x024	DC6	RO	0x0000.0013	Device Capabilities 6	244
0x028	DC7	RO	0xFFFF.FFFF	Device Capabilities 7	245
0x02C	DC8	RO	0xFFFF.FFFF	Device Capabilities 8 ADC Channels	249
0x030	PBORCTL	R/W	0x0000.0002	Brown-Out Reset Control	206
0x040	SRCR0	R/W	0x00000000	Software Reset Control 0	281
0x044	SRCR1	R/W	0x00000000	Software Reset Control 1	283
0x048	SRCR2	R/W	0x00000000	Software Reset Control 2	286
0x050	RIS	RO	0x0000.0000	Raw Interrupt Status	207
0x054	IMC	R/W	0x0000.0000	Interrupt Mask Control	209
0x058	MISC	R/W1C	0x0000.0000	Masked Interrupt Status and Clear	211
0x05C	RESC	R/W	-	Reset Cause	213
0x060	RCC	R/W	0x0780.3AD1	Run-Mode Clock Configuration	215
0x064	PLLCFG	RO	-	XTAL to PLL Translation	219
0x06C	GPIOHBCTL	R/W	0x0000.0000	GPIO High-Performance Bus Control	220
0x070	RCC2	R/W	0x07C0.6810	Run-Mode Clock Configuration 2	222
0x07C	MOSCCTL	R/W	0x0000.0000	Main Oscillator Control	225
0x100	RCGC0	R/W	0x00000040	Run Mode Clock Gating Control Register 0	255
0x104	RCGC1	R/W	0x00000000	Run Mode Clock Gating Control Register 1	263
0x108	RCGC2	R/W	0x00000000	Run Mode Clock Gating Control Register 2	272
0x110	SCGC0	R/W	0x00000040	Sleep Mode Clock Gating Control Register 0	258
0x114	SCGC1	R/W	0x00000000	Sleep Mode Clock Gating Control Register 1	266
0x118	SCGC2	R/W	0x00000000	Sleep Mode Clock Gating Control Register 2	275
0x120	DCGC0	R/W	0x00000040	Deep Sleep Mode Clock Gating Control Register 0	261
0x124	DCGC1	R/W	0x00000000	Deep-Sleep Mode Clock Gating Control Register 1	269
0x128	DCGC2	R/W	0x00000000	Deep Sleep Mode Clock Gating Control Register 2	278
0x144	DSLPCLKCFG	R/W	0x0780.0000	Deep Sleep Clock Configuration	226

Table 5-8. System Control Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x150	PIOSCCAL	R/W	0x0000.0000	Precision Internal Oscillator Calibration	228
0x170	I2SMCLKCFG	R/W	0x0000.0000	I2S MCLK Configuration	229
0x190	DC9	RO	0x00FF.00FF	Device Capabilities 9 ADC Digital Comparators	252
0x1A0	NVMSTAT	RO	0x0000.0001	Non-Volatile Memory Information	254

5.5 Register Descriptions

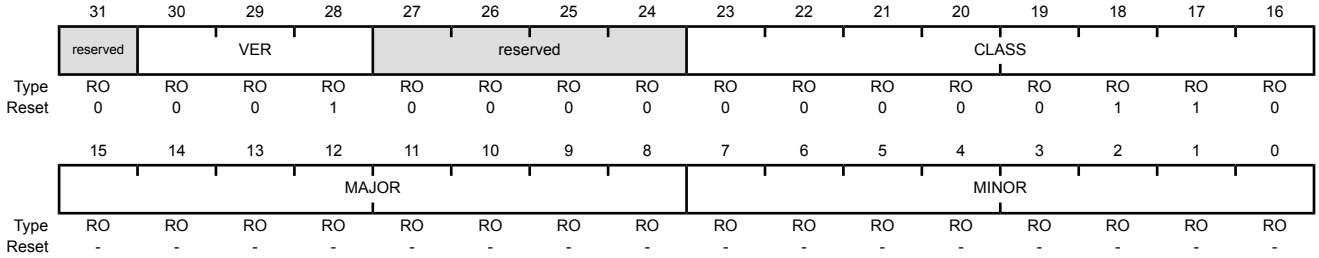
All addresses given are relative to the System Control base address of 0x400F.E000.

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

This register identifies the version of the microcontroller. Each microcontroller is uniquely identified by the combined values of the CLASS field in the **DID0** register and the PARTNO field in the **DID1** register.

Device Identification 0 (DID0)

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



Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30:28	VER	RO	0x1	DID0 Version This field defines the **DID0** register format version. The version number is numeric. The value of the VER field is encoded as follows (all other encodings are reserved): Value Description 0x1 Second version of the **DID0** register format.
27:24	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:16	CLASS	RO	0x06	Device Class The CLASS field value identifies the internal design from which all mask sets are generated for all microcontrollers in a particular product line. The CLASS field value is changed for new product lines, for changes in fab process (for example, a remap or shrink), or any case where the MAJOR or MINOR fields require differentiation from prior microcontrollers. The value of the CLASS field is encoded as follows (all other encodings are reserved): Value Description 0x06 Stellaris® Firestorm-class microcontrollers

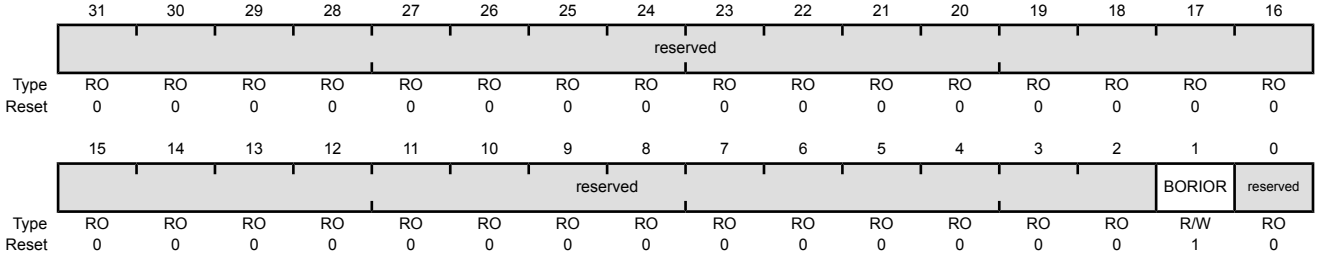
Bit/Field	Name	Type	Reset	Description
15:8	MAJOR	RO	-	Major Revision This field specifies the major revision number of the microcontroller. The major revision reflects changes to base layers of the design. The major revision number is indicated in the part number as a letter (A for first revision, B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

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

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

Brown-Out Reset Control (PBORCTL)

Base 0x400F.E000
Offset 0x030
Type R/W, reset 0x0000.0002



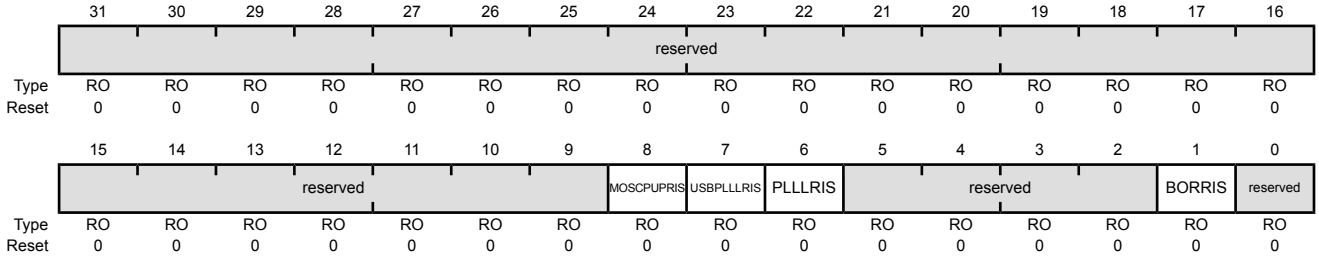
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BORIOR	R/W	1	BOR Interrupt or Reset Value Description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

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

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

Raw Interrupt Status (RIS)

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



Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	MOSCPUPRIS	RO	0	MOSC Power Up Raw Interrupt Status Value Description 1 Sufficient time has passed for the MOSC to reach the expected frequency. The value for this power-up time is indicated by T_{MOSC_START}. 0 Sufficient time has not passed for the MOSC to reach the expected frequency. This bit is cleared by writing a 1 to the `MOSCPUPMIS` bit in the **MISC** register.
7	USBPLLLRIS	RO	0	USB PLL Lock Raw Interrupt Status Value Description 1 The USB PLL timer has reached T_{READY} indicating that sufficient time has passed for the USB PLL to lock. 0 The USB PLL timer has not reached T_{READY}. This bit is cleared by writing a 1 to the `USBPLLLMIS` bit in the **MISC** register.
6	PLLLRIS	RO	0	PLL Lock Raw Interrupt Status Value Description 1 The PLL timer has reached T_{READY} indicating that sufficient time has passed for the PLL to lock. 0 The PLL timer has not reached T_{READY}. This bit is cleared by writing a 1 to the `PLLLMIS` bit in the **MISC** register.

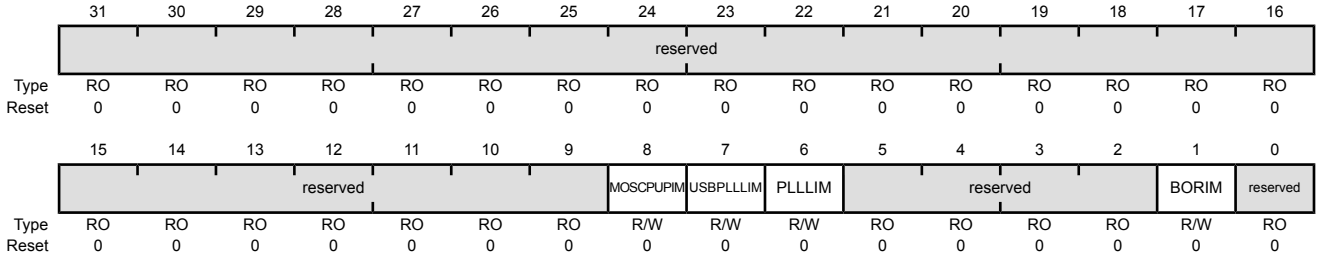
Bit/Field	Name	Type	Reset	Description
5:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BORRIS	RO	0	Brown-Out Reset Raw Interrupt Status Value Description 1 A brown-out condition is currently active. 0 A brown-out condition is not currently active. Note the BORIOR bit in the **PBORCTL** register must be cleared to cause an interrupt due to a Brown Out Event. This bit is cleared by writing a 1 to the BORMIS bit in the **MISC** register.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

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

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

Interrupt Mask Control (IMC)

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



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

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

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

On a read, this register gives the current masked status value of the corresponding interrupt in the **Raw Interrupt Status (RIS)** register. All of the bits are R/W1C, thus writing a 1 to a bit clears the corresponding raw interrupt bit in the **RIS** register (see page 207).

Masked Interrupt Status and Clear (MISC)

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

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

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

Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	MOSCPUPMIS	R/W1C	0	MOSC Power Up Masked Interrupt Status Value Description 1 When read, a 1 indicates that an unmasked interrupt was signaled because sufficient time has passed for the MOSC PLL to lock. Writing a 1 to this bit clears it and also the `MOSCPUPRIS` bit in the **RIS** register. 0 When read, a 0 indicates that sufficient time has not passed for the MOSC PLL to lock. A write of 0 has no effect on the state of this bit.
7	USBPLLLMIS	R/W1C	0	USB PLL Lock Masked Interrupt Status Value Description 1 When read, a 1 indicates that an unmasked interrupt was signaled because sufficient time has passed for the USB PLL to lock. Writing a 1 to this bit clears it and also the `USBPLLLRIS` bit in the **RIS** register. 0 When read, a 0 indicates that sufficient time has not passed for the USB PLL to lock. A write of 0 has no effect on the state of this bit.

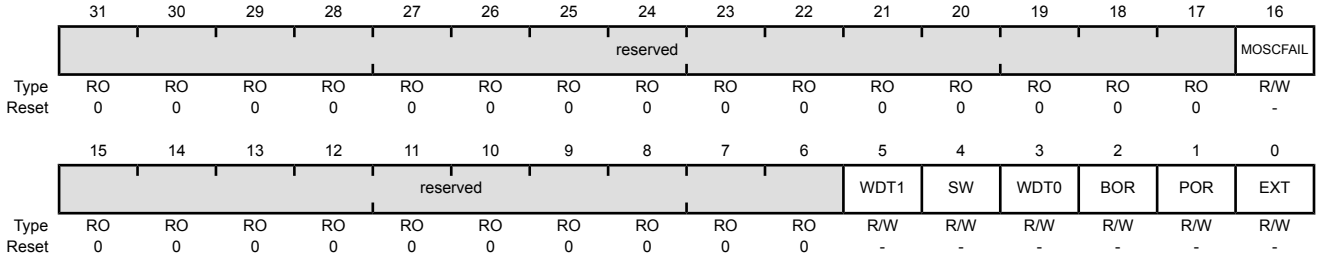
Bit/Field	Name	Type	Reset	Description
6	PLLLMIS	R/W1C	0	PLL Lock Masked Interrupt Status Value Description 1 When read, a 1 indicates that an unmasked interrupt was signaled because sufficient time has passed for the PLL to lock. Writing a 1 to this bit clears it and also the PLLLRIS bit in the **RIS**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

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

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

Reset Cause (RESC)

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



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

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

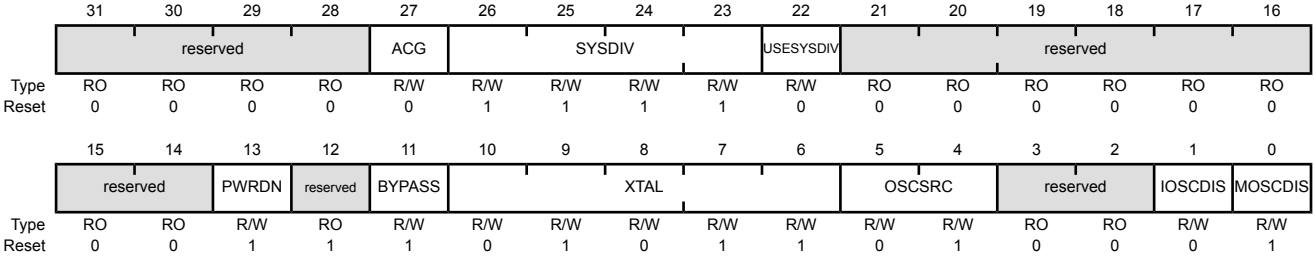
Register 7: Run-Mode Clock Configuration (RCC), offset 0x060

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

Important: Write the **RCC** register prior to writing the **RCC2** register. If a subsequent write to the **RCC** register is required, include another register access after writing the **RCC** register and before writing the **RCC2** register.

Run-Mode Clock Configuration (RCC)

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



Bit/Field	Name	Type	Reset	Description
31:28	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
27	ACG	R/W	0	Auto Clock Gating This bit specifies whether the system uses the **Sleep-Mode Clock Gating Control (SCGCn)** registers and **Deep-Sleep-Mode Clock Gating Control (DCGCn)** registers if the microcontroller enters a Sleep or Deep-Sleep mode (respectively). Value Description 1 The **SCGCn** or **DCGCn** registers are used to control the clocks distributed to the peripherals when the microcontroller is in a sleep mode. The **SCGCn** and **DCGCn** registers allow unused peripherals to consume less power when the microcontroller is in a sleep mode. 0 The **Run-Mode Clock Gating Control (RCGCn)** registers are used when the microcontroller enters a sleep mode. The **RCGCn** registers are always used to control the clocks in Run mode.
26:23	SYSDIV	R/W	0xF	System Clock Divisor Specifies which divisor is used to generate the system clock from either the PLL output or the oscillator source (depending on how the BYPASS bit in this register is configured). See Table 5-5 on page 196 for bit encodings. If the SYSDIV value is less than MINSYSDIV (see page 234), and the PLL is being used, then the MINSYSDIV value is used as the divisor. If the PLL is not being used, the SYSDIV value can be less than MINSYSDIV.

Bit/Field	Name	Type	Reset	Description
22	USESYSDIV	R/W	0	Enable System Clock Divider Value Description 1 The system clock divider is the source for the system clock. The system clock divider is forced to be used when the PLL is selected as the source. If the USERCC2 bit in the **RCC2** register is set, then the SYSDIV2 field in the **RCC2** register is used as the system clock divider rather than the SYSDIV field in this register. 0 The system clock is used undivided.
21:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	PWRDN	R/W	1	PLL Power Down Value Description 1 The PLL is powered down. Care must be taken to ensure that another clock source is functioning and that the BYPASS bit is set before setting this bit. 0 The PLL is operating normally.
12	reserved	RO	1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	BYPASS	R/W	1	PLL Bypass Value Description 1 The system clock is derived from the OSC source and divided by the divisor specified by SYSDIV. 0 The system clock is the PLL output clock divided by the divisor specified by SYSDIV. See Table 5-5 on page 196 for programming guidelines. **Note:** The ADC must be clocked from the PLL or directly from a 16-MHz clock source to operate properly.

Bit/Field	Name	Type	Reset	Description
10:6	XTAL	R/W	0x0B	Crystal Value This field specifies the crystal value attached to the main oscillator. The encoding for this field is provided below. Depending on the crystal used, the PLL frequency may not be exactly 400 MHz, see Table 24-8 on page 1198 for more information. Frequencies that may be used with the USB interface are indicated in the table. To function within the clocking requirements of the USB specification, a crystal of 4, 5, 6, 8, 10, 12, or 16 MHz must be used.

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

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

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

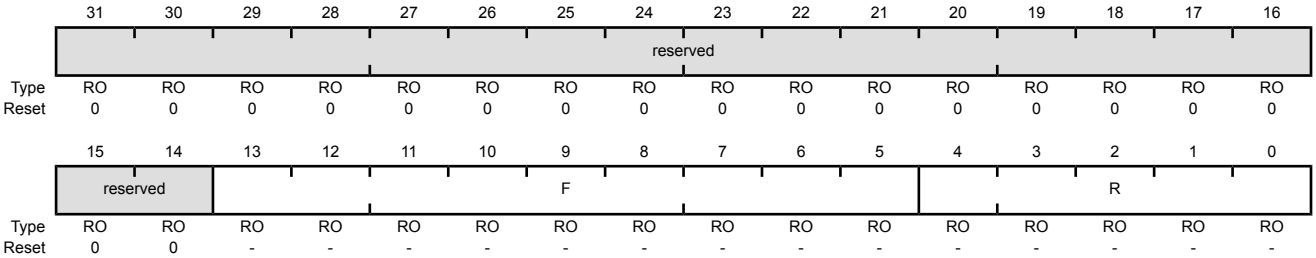
This register provides a means of translating external crystal frequencies into the appropriate PLL settings. This register is initialized during the reset sequence and updated anytime that the XTAL field changes in the **Run-Mode Clock Configuration (RCC)** register (see page 215).

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

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

XTAL to PLL Translation (PLLCFG)

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



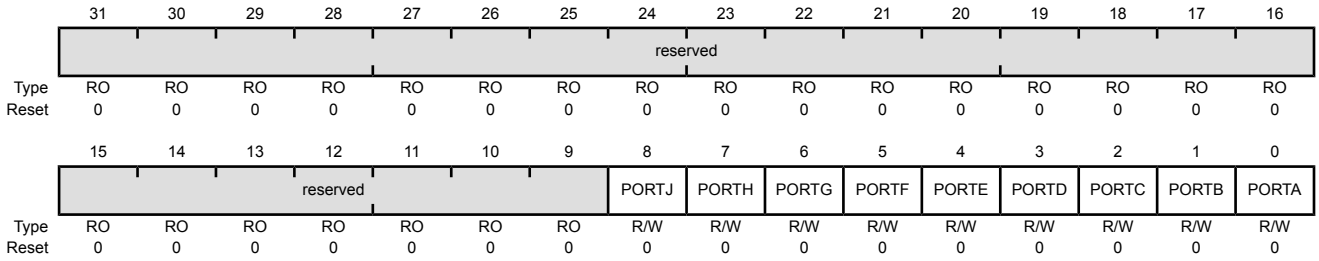
Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13:5	F	RO	-	PLL F Value This field specifies the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

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

This register controls which internal bus is used to access each GPIO port. When a bit is clear, the corresponding GPIO port is accessed across the legacy Advanced Peripheral Bus (APB) bus and through the APB memory aperture. When a bit is set, the corresponding port is accessed across the Advanced High-Performance Bus (AHB) bus and through the AHB memory aperture. Each GPIO port can be individually configured to use AHB or APB, but may be accessed only through one aperture. The AHB bus provides better back-to-back access performance than the APB bus. The address aperture in the memory map changes for the ports that are enabled for AHB access (see Table 8-7 on page 406).

GPIO High-Performance Bus Control (GPIOHBCTL)

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



Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	PORTJ	R/W	0	Port J Advanced High-Performance Bus This bit defines the memory aperture for Port J. Value Description 1 Advanced High-Performance Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.
7	PORTH	R/W	0	Port H Advanced High-Performance Bus This bit defines the memory aperture for Port H. Value Description 1 Advanced High-Performance Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.
6	PORTG	R/W	0	Port G Advanced High-Performance Bus This bit defines the memory aperture for Port G. Value Description 1 Advanced High-Performance Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.

Bit/Field	Name	Type	Reset	Description
5	PORTF	R/W	0	Port F Advanced High-Performance Bus This bit defines the memory aperture for Port F. Value Description 1 Advanced High-Performance Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.
4	PORTE	R/W	0	Port E Advanced High-Performance Bus This bit defines the memory aperture for Port E. Value Description 1 Advanced High-Performance Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.
3	PORTD	R/W	0	Port D Advanced High-Performance Bus This bit defines the memory aperture for Port D. Value Description 1 Advanced High-Performance Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.
2	PORTC	R/W	0	Port C Advanced High-Performance Bus This bit defines the memory aperture for Port C. Value Description 1 Advanced High-Performance Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.
1	PORTB	R/W	0	Port B Advanced High-Performance Bus This bit defines the memory aperture for Port B. Value Description 1 Advanced High-Performance Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.
0	PORTA	R/W	0	Port A Advanced High-Performance Bus This bit defines the memory aperture for Port A. Value Description 1 Advanced High-Performance Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.

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

This register overrides the **RCC** equivalent register fields, as shown in Table 5-9, when the USERCC2 bit is set, allowing the extended capabilities of the **RCC2** register to be used while also providing a means to be backward-compatible to previous parts. Each **RCC2** field that supersedes an **RCC** field is located at the same LSB bit position; however, some **RCC2** fields are larger than the corresponding **RCC** field.

Table 5-9. RCC2 Fields that Override RCC Fields

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

Important: Write the **RCC** register prior to writing the **RCC2** register. If a subsequent write to the **RCC** register is required, include another register access after writing the **RCC** register and before writing the **RCC2** register.

Run-Mode Clock Configuration 2 (RCC2)

Base 0x400F.E000
Offset 0x070
Type R/W, reset 0x07C0.6810

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved	USBPWRDN	PWRDN2	reserved	BYPASS2	reserved				OSCSRC2			reserved			
Type RO	R/W	R/W	RO	R/W	RO	RO	RO	RO	R/W	R/W	R/W	RO	RO	RO	RO
Reset 0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	USERCC2	R/W	0	Use **RCC2** Value Description 1 The **RCC2** register fields override the **RCC** register fields. 0 The **RCC** register fields are used, and the fields in **RCC2** are ignored.
30	DIV400	R/W	0	Divide PLL as 400 MHz vs. 200 MHz This bit, along with the SYSDIV2LSB bit, allows additional frequency choices. Value Description 1 Append the SYSDIV2LSB bit to the SYSDIV2 field to create a 7 bit divisor using the 400 MHz PLL output, see Table 5-7 on page 197. 0 Use SYSDIV2 as is and apply to 200 MHz predivided PLL output. See Table 5-6 on page 196 for programming guidelines.

Bit/Field	Name	Type	Reset	Description
29	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28:23	SYSDIV2	R/W	0x0F	System Clock Divisor 2 Specifies which divisor is used to generate the system clock from either the PLL output or the oscillator source (depending on how the BYPASS2 bit is configured). SYSDIV2 is used for the divisor when both the USESYSDIV bit in the **RCC** register and the USERCC2 bit in this register are set. See Table 5-6 on page 196 for programming guidelines.
22	SYSDIV2LSB	R/W	1	Additional LSB for SYSDIV2 When DIV400 is set, this bit becomes the LSB of SYSDIV2. If DIV400 is clear, this bit is not used. See Table 5-6 on page 196 for programming guidelines. This bit can only be set or cleared when DIV400 is set.
21:15	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	USBPWRDN	R/W	1	Power-Down USB PLL Value Description 1 The USB PLL is powered down. 0 The USB PLL operates normally.
13	PWRDN2	R/W	1	Power-Down PLL 2 Value Description 1 The PLL is powered down. 0 The PLL operates normally.
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	BYPASS2	R/W	1	PLL Bypass 2 Value Description 1 The system clock is derived from the OSC source and divided by the divisor specified by SYSDIV2. 0 The system clock is the PLL output clock divided by the divisor specified by SYSDIV2. See Table 5-6 on page 196 for programming guidelines. **Note:** The ADC must be clocked from the PLL or directly from a 16-MHz clock source to operate properly.
10:7	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

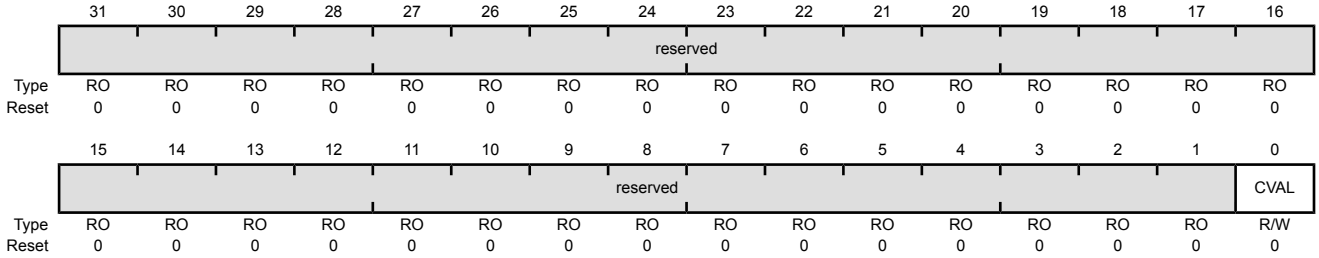
Bit/Field	Name	Type	Reset	Description
6:4	OSCSRC2	R/W	0x1	Oscillator Source 2 Selects the input source for the OSC. The values are: Value Description 0x0 MOSC Main oscillator 0x1 PIOSC Precision internal oscillator 0x2 PIOSC/4 Precision internal oscillator / 4 0x3 30 kHz 30-kHz internal oscillator 0x4-0x7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

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

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

Main Oscillator Control (MOSCCTL)

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



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

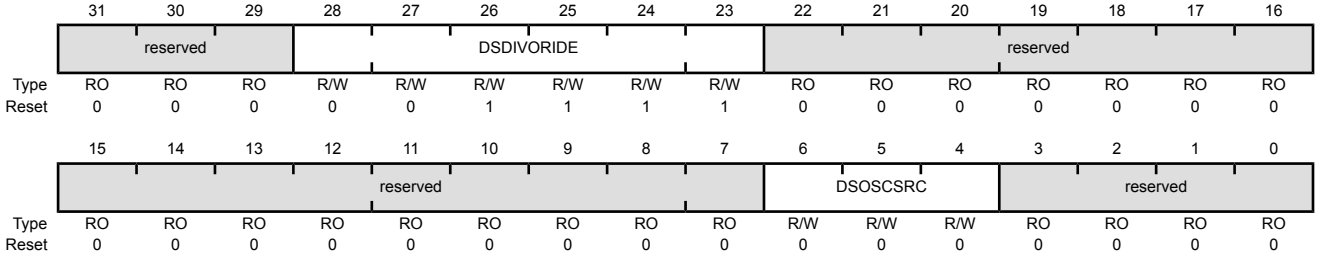
Value	Description
1	The MOSC monitor circuit is enabled.
0	The MOSC monitor circuit is disabled.

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

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

Deep Sleep Clock Configuration (DSLPCLKCFG)

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



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

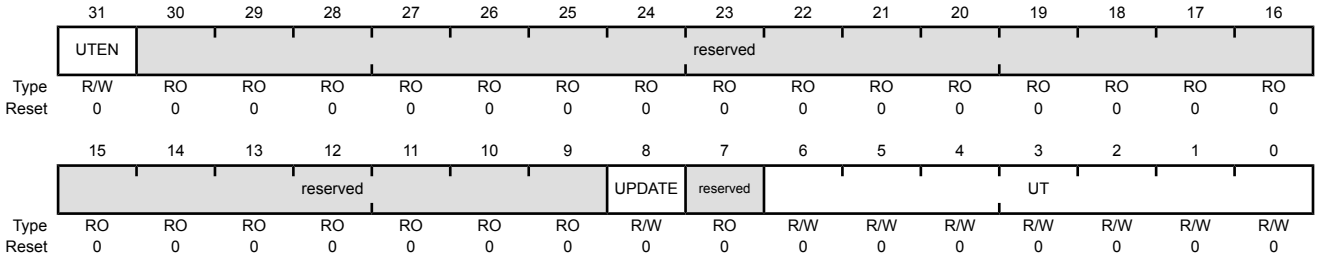
Bit/Field	Name	Type	Reset	Description
6:4	DSOSCSRC	R/W	0x0	Clock Source Specifies the clock source during Deep-Sleep mode. Value Description 0x0 MOSC Use the main oscillator as the source. **Note:** If the PIOSC is being used as the clock reference for the PLL, the PIOSC is the clock source instead of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

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

This register provides the ability to update or recalibrate the precision internal oscillator.

Precision Internal Oscillator Calibration (PIOSCCAL)

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



Bit/Field	Name	Type	Reset	Description
31	UTEN	R/W	0	Use User Trim Value Value Description 1 The trim value in bits[6:0] of this register are used for any update trim operation. 0 The factory calibration value is used for an update trim operation.
30:9	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	UPDATE	R/W	0	Update Trim Value Description 1 Updates the PIOSC trim value with the UT bit. Used with UTEN. 0 No action. This bit is auto-cleared after the update.
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	UT	R/W	0x0	User Trim Value User trim value that can be loaded into the PIOSC. Refer to “Main PLL Frequency Configuration” on page 198 for more information on calibrating the PIOSC.

Register 14: I²S MCLK Configuration (I2SMCLKCFG), offset 0x170

This register configures the receive and transmit fractional clock dividers for the for the I²S master transmit and receive clocks (I2S0TXMCLK and I2S0RXMCLK). Varying the integer and fractional inputs for the clocks allows greater accuracy in hitting the target I²S clock frequencies. Refer to "Clock Control" on page 827 for combinations of the TXI and TXF bits and the RXI and RXF bits that provide MCLK frequencies within acceptable error limits.

I2S MCLK Configuration (I2SMCLKCFG)

Base 0x400F.E000
Offset 0x170
Type R/W, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXEN	reserved	RXI										RXF			
Type R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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
TXEN	reserved	TXI										TXF			
Type R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	RXEN	R/W	0	RX Clock Enable Value Description 1 The I²S receive clock generator is enabled. 0 The I²S receive clock generator is disabled. If the RXSLV bit in the **I²S Module Configuration (I2SCFG)** register is set, then the I2S0RXMCLK must be externally generated.
30	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
29:20	RXI	R/W	0x0	RX Clock Integer Input This field contains the integer input for the receive clock generator.
19:16	RXF	R/W	0x0	RX Clock Fractional Input This field contains the fractional input for the receive clock generator.
15	TXEN	R/W	0	TX Clock Enable Value Description 1 The I²S transmit clock generator is enabled. 0 The I²S transmit clock generator is disabled. If the TXSLV bit in the **I²S Module Configuration (I2SCFG)** register is set, then the I2S0TXMCLK must be externally generated.

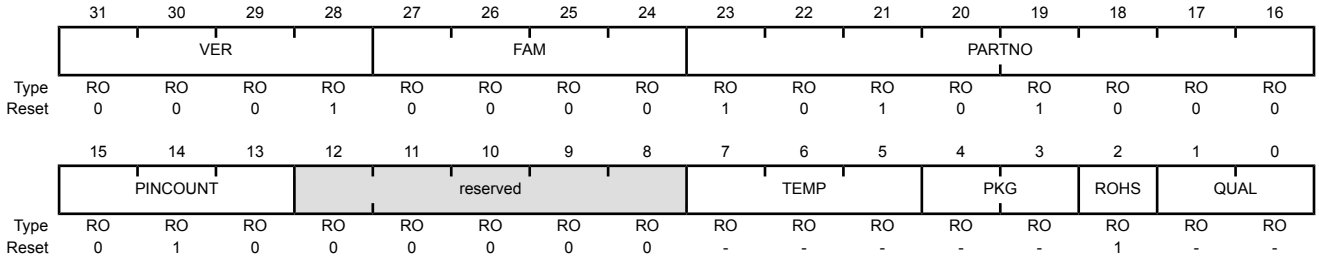
Bit/Field	Name	Type	Reset	Description
14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13:4	TXI	R/W	0x00	TX Clock Integer Input This field contains the integer input for the transmit clock generator.
3:0	TXF	R/W	0x0	TX Clock Fractional Input This field contains the fractional input for the transmit clock generator.

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

This register identifies the device family, part number, temperature range, pin count, and package type. Each microcontroller is uniquely identified by the combined values of the CLASS field in the **DID0** register and the PARTNO field in the **DID1** register.

Device Identification 1 (DID1)

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



Bit/Field	Name	Type	Reset	Description
31:28	VER	RO	0x1	DID1 Version This field defines the **DID1** register format version. The version number is numeric. The value of the VER field is encoded as follows (all other encodings are reserved): Value Description 0x1 Second version of the **DID1** register format.
27:24	FAM	RO	0x0	Family This field provides the family identification of the device within the Luminary Micro product portfolio. The value is encoded as follows (all other encodings are reserved): Value Description 0x0 Stellaris family of microcontollers, that is, all devices with external part numbers starting with LM3S.
23:16	PARTNO	RO	0xA8	Part Number This field provides the part number of the device within the family. The value is encoded as follows (all other encodings are reserved): Value Description 0xA8 LM3S9U81
15:13	PINCOUNT	RO	0x2	Package Pin Count This field specifies the number of pins on the device package. The value is encoded as follows (all other encodings are reserved): Value Description 0x2 100-pin package

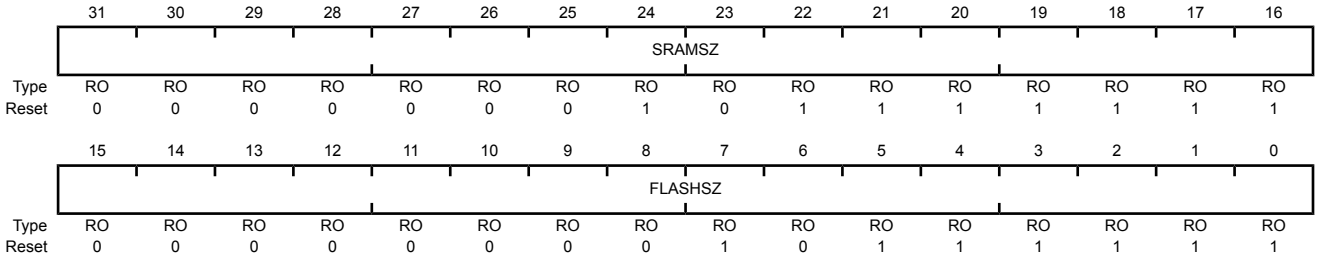
Bit/Field	Name	Type	Reset	Description
12:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:5	TEMP	RO	-	Temperature Range This field specifies the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

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

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

Device Capabilities 0 (DC0)

Base 0x400F.E000
Offset 0x008
Type RO, reset 0x017F.00BF



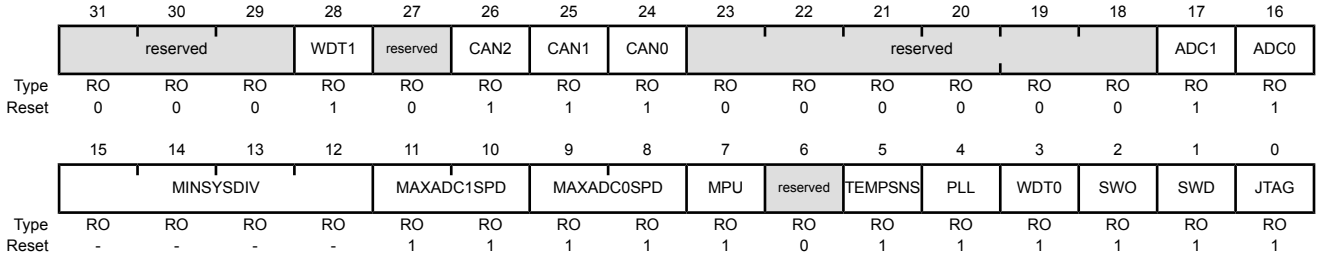
Bit/Field	Name	Type	Reset	Description
31:16	SRAMSZ	RO	0x017F	SRAM Size Indicates the size of the on-chip SRAM memory. Value Description 0x017F 96 KB of SRAM
15:0	FLASHSZ	RO	0x00BF	Flash Size Indicates the size of the on-chip flash memory. Value Description 0x00BF 384 KB of Flash

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

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

Device Capabilities 1 (DC1)

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



Bit/Field	Name	Type	Reset	Description
31:29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	WDT1	RO	1	Watchdog Timer 1 Present When set, indicates that watchdog timer 1 is present.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	CAN2	RO	1	CAN Module 2 Present When set, indicates that CAN unit 2 is present.
25	CAN1	RO	1	CAN Module 1 Present When set, indicates that CAN unit 1 is present.
24	CAN0	RO	1	CAN Module 0 Present When set, indicates that CAN unit 0 is present.
23:18	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
17	ADC1	RO	1	ADC Module 1 Present When set, indicates that ADC module 1 is present.
16	ADC0	RO	1	ADC Module 0 Present When set, indicates that ADC module 0 is present

Bit/Field	Name	Type	Reset	Description
15:12	MINSYSDIV	RO	-	System Clock Divider Minimum 4-bit divider value for system clock. The reset value is hardware-dependent. See the RCC register for how to change the system clock divisor using the SYSDIV bit. Value Description 0x1 Specifies an 80-MHz CPU clock with a PLL divider of 2.5. 0x2 Specifies a 66.67-MHz CPU clock with a PLL divider of 3. 0x3 Specifies a 50-MHz CPU clock with a PLL divider of 4. 0x7 Specifies a 25-MHz clock with a PLL divider of 8. 0x9 Specifies a 20-MHz clock with a PLL divider of 10.
11:10	MAXADC1SPD	RO	0x3	Max ADC1 Speed This field indicates the maximum rate at which the ADC samples data. Value Description 0x3 1M samples/second
9:8	MAXADC0SPD	RO	0x3	Max ADC0 Speed This field indicates the maximum rate at which the ADC samples data. Value Description 0x3 1M samples/second
7	MPU	RO	1	MPU Present When set, indicates that the Cortex-M3 Memory Protection Unit (MPU) module is present. See the "Cortex-M3 Peripherals" chapter for details on the MPU.
6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	TEMPSNS	RO	1	Temp Sensor Present When set, indicates that the on-chip temperature sensor is present.
4	PLL	RO	1	PLL Present When set, indicates that the on-chip Phase Locked Loop (PLL) is present.
3	WDT0	RO	1	Watchdog Timer 0 Present When set, indicates that watchdog timer 0 is present.
2	SWO	RO	1	SWO Trace Port Present When set, indicates that the Serial Wire Output (SWO) trace port is present.
1	SWD	RO	1	SWD Present When set, indicates that the Serial Wire Debugger (SWD) is present.

Bit/Field	Name	Type	Reset	Description
0	JTAG	RO	1	JTAG Present When set, indicates that the JTAG debugger interface is present.

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

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

Device Capabilities 2 (DC2)

Base 0x400F.E000
Offset 0x014
Type RO, reset 0x570F.5037

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved	EPI0	reserved	I2S0	reserved	COMP2	COMP1	COMP0	reserved				TIMER3	TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	1	0	1	0	1	1	1	0	0	0	0	1	1	1	1

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

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPI0	RO	1	EPI Module 0 Present When set, indicates that EPI module 0 is present.
29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	I2S0	RO	1	I2S Module 0 Present When set, indicates that I2S module 0 is present.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	COMP2	RO	1	Analog Comparator 2 Present When set, indicates that analog comparator 2 is present.
25	COMP1	RO	1	Analog Comparator 1 Present When set, indicates that analog comparator 1 is present.
24	COMP0	RO	1	Analog Comparator 0 Present When set, indicates that analog comparator 0 is present.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	RO	1	Timer Module 3 Present When set, indicates that General-Purpose Timer module 3 is present.
18	TIMER2	RO	1	Timer Module 2 Present When set, indicates that General-Purpose Timer module 2 is present.

Bit/Field	Name	Type	Reset	Description
17	TIMER1	RO	1	Timer Module 1 Present When set, indicates that General-Purpose Timer module 1 is present.
16	TIMER0	RO	1	Timer Module 0 Present When set, indicates that General-Purpose Timer module 0 is present.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	RO	1	I2C Module 1 Present When set, indicates that I2C module 1 is present.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	RO	1	I2C Module 0 Present When set, indicates that I2C module 0 is present.
11:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SSI1	RO	1	SSI Module 1 Present When set, indicates that SSI module 1 is present.
4	SSI0	RO	1	SSI Module 0 Present When set, indicates that SSI module 0 is present.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	RO	1	UART Module 2 Present When set, indicates that UART module 2 is present.
1	UART1	RO	1	UART Module 1 Present When set, indicates that UART module 1 is present.
0	UART0	RO	1	UART Module 0 Present When set, indicates that UART module 0 is present.

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

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

Device Capabilities 3 (DC3)

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

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

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

Bit/Field	Name	Type	Reset	Description
31	32KHZ	RO	1	32KHz Input Clock Available When set, indicates an even CCP pin is present and can be used as a 32-KHz input clock.
30	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
29	CCP5	RO	1	CCP5 Pin Present When set, indicates that Capture/Compare/PWM pin 5 is present.
28	CCP4	RO	1	CCP4 Pin Present When set, indicates that Capture/Compare/PWM pin 4 is present.
27	CCP3	RO	1	CCP3 Pin Present When set, indicates that Capture/Compare/PWM pin 3 is present.
26	CCP2	RO	1	CCP2 Pin Present When set, indicates that Capture/Compare/PWM pin 2 is present.
25	CCP1	RO	1	CCP1 Pin Present When set, indicates that Capture/Compare/PWM pin 1 is present.
24	CCP0	RO	1	CCP0 Pin Present When set, indicates that Capture/Compare/PWM pin 0 is present.
23	ADC0AIN7	RO	1	ADC Module 0 AIN7 Pin Present When set, indicates that ADC module 0 input pin 7 is present.
22	ADC0AIN6	RO	1	ADC Module 0 AIN6 Pin Present When set, indicates that ADC module 0 input pin 6 is present.
21	ADC0AIN5	RO	1	ADC Module 0 AIN5 Pin Present When set, indicates that ADC module 0 input pin 5 is present.

Bit/Field	Name	Type	Reset	Description
20	ADC0AIN4	RO	1	ADC Module 0 AIN4 Pin Present When set, indicates that ADC module 0 input pin 4 is present.
19	ADC0AIN3	RO	1	ADC Module 0 AIN3 Pin Present When set, indicates that ADC module 0 input pin 3 is present.
18	ADC0AIN2	RO	1	ADC Module 0 AIN2 Pin Present When set, indicates that ADC module 0 input pin 2 is present.
17	ADC0AIN1	RO	1	ADC Module 0 AIN1 Pin Present When set, indicates that ADC module 0 input pin 1 is present.
16	ADC0AIN0	RO	1	ADC Module 0 AIN0 Pin Present When set, indicates that ADC module 0 input pin 0 is present.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	C2O	RO	1	C2o Pin Present When set, indicates that the analog comparator 2 output pin is present.
13	C2PLUS	RO	1	C2+ Pin Present When set, indicates that the analog comparator 2 (+) input pin is present.
12	C2MINUS	RO	1	C2- Pin Present When set, indicates that the analog comparator 2 (-) input pin is present.
11	C1O	RO	1	C1o Pin Present When set, indicates that the analog comparator 1 output pin is present.
10	C1PLUS	RO	1	C1+ Pin Present When set, indicates that the analog comparator 1 (+) input pin is present.
9	C1MINUS	RO	1	C1- Pin Present When set, indicates that the analog comparator 1 (-) input pin is present.
8	C0O	RO	1	C0o Pin Present When set, indicates that the analog comparator 0 output pin is present.
7	C0PLUS	RO	1	C0+ Pin Present When set, indicates that the analog comparator 0 (+) input pin is present.
6	C0MINUS	RO	1	C0- Pin Present When set, indicates that the analog comparator 0 (-) input pin is present.
5:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 20: Device Capabilities 4 (DC4), offset 0x01C

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

Device Capabilities 4 (DC4)

Base 0x400F.E000
Offset 0x01C
Type RO, reset 0x5004.F1FF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved	EPHY0	reserved	EMAC0	reserved									PICAL	reserved	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CCP7	CCP6	UDMA	ROM	reserved			GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPHY0	RO	1	Ethernet PHY Layer 0 Present When set, indicates that Ethernet PHY layer 0 is present.
29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	EMAC0	RO	1	Ethernet MAC Layer 0 Present When set, indicates that Ethernet MAC layer 0 is present.
27:19	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
18	PICAL	RO	1	PIOSC Calibrate When set, indicates that the PIOSC can be calibrated.
17:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	CCP7	RO	1	CCP7 Pin Present When set, indicates that Capture/Compare/PWM pin 7 is present.
14	CCP6	RO	1	CCP6 Pin Present When set, indicates that Capture/Compare/PWM pin 6 is present.
13	UDMA	RO	1	Micro-DMA Module Present When set, indicates that the micro-DMA module present.
12	ROM	RO	1	Internal Code ROM Present When set, indicates that internal code ROM is present.

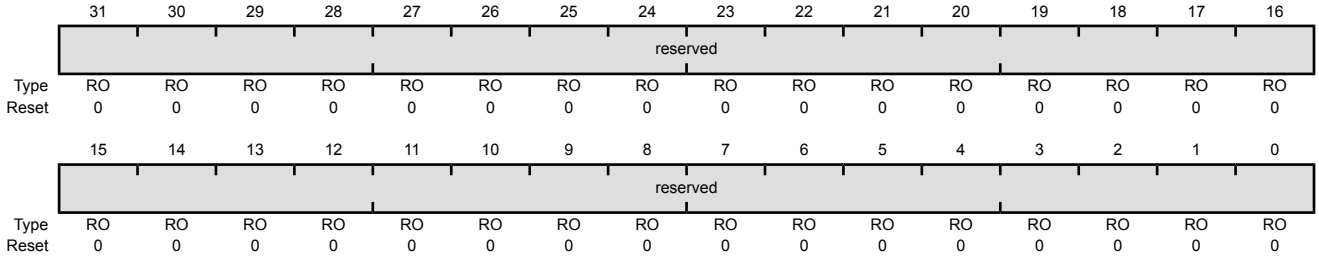
Bit/Field	Name	Type	Reset	Description
11:9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	GPIOJ	RO	1	GPIO Port J Present When set, indicates that GPIO Port J is present.
7	GPIOH	RO	1	GPIO Port H Present When set, indicates that GPIO Port H is present.
6	GPIOG	RO	1	GPIO Port G Present When set, indicates that GPIO Port G is present.
5	GPIOF	RO	1	GPIO Port F Present When set, indicates that GPIO Port F is present.
4	GPIOE	RO	1	GPIO Port E Present When set, indicates that GPIO Port E is present.
3	GPIOD	RO	1	GPIO Port D Present When set, indicates that GPIO Port D is present.
2	GPIOC	RO	1	GPIO Port C Present When set, indicates that GPIO Port C is present.
1	GPIOB	RO	1	GPIO Port B Present When set, indicates that GPIO Port B is present.
0	GPIOA	RO	1	GPIO Port A Present When set, indicates that GPIO Port A is present.

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

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

Device Capabilities 5 (DC5)

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



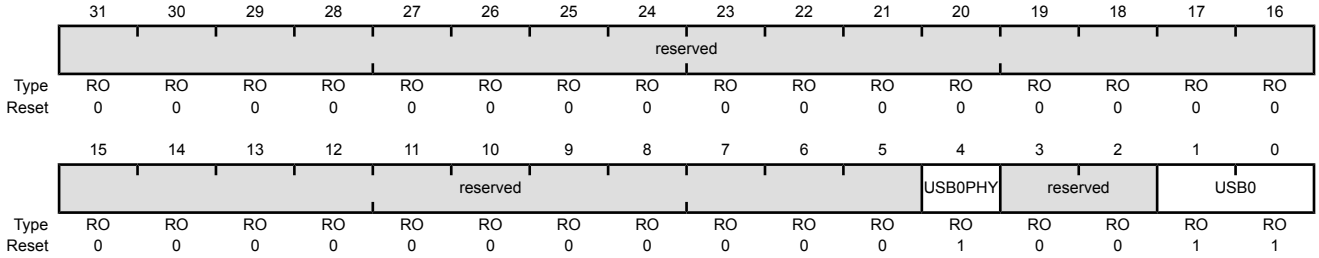
Bit/Field	Name	Type	Reset	Description
31:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

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

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

Device Capabilities 6 (DC6)

Base 0x400F.E000
Offset 0x024
Type RO, reset 0x0000.0013



Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	USB0PHY	RO	1	USB Module 0 PHY Present When set, indicates that the USB module 0 PHY is present.
3:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1:0	USB0	RO	0x3	USB Module 0 Present Thie field indicates that USB module 0 is present and specifies its capability. Value Description 0x3 USB0 is OTG.

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

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

Device Capabilities 7 (DC7)

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

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

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

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

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

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

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

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

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

Device Capabilities 8 ADC Channels (DC8)

Base 0x400F.E000
Offset 0x02C
Type RO, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ADC1AIN15	ADC1AIN14	ADC1AIN13	ADC1AIN12	ADC1AIN11	ADC1AIN10	ADC1AIN9	ADC1AIN8	ADC1AIN7	ADC1AIN6	ADC1AIN5	ADC1AIN4	ADC1AIN3	ADC1AIN2	ADC1AIN1	ADC1AIN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ADC0AIN15	ADC0AIN14	ADC0AIN13	ADC0AIN12	ADC0AIN11	ADC0AIN10	ADC0AIN9	ADC0AIN8	ADC0AIN7	ADC0AIN6	ADC0AIN5	ADC0AIN4	ADC0AIN3	ADC0AIN2	ADC0AIN1	ADC0AIN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31	ADC1AIN15	RO	1	ADC Module 1 AIN15 Pin Present When set, indicates that ADC module 1 input pin 15 is present.
30	ADC1AIN14	RO	1	ADC Module 1 AIN14 Pin Present When set, indicates that ADC module 1 input pin 14 is present.
29	ADC1AIN13	RO	1	ADC Module 1 AIN13 Pin Present When set, indicates that ADC module 1 input pin 13 is present.
28	ADC1AIN12	RO	1	ADC Module 1 AIN12 Pin Present When set, indicates that ADC module 1 input pin 12 is present.
27	ADC1AIN11	RO	1	ADC Module 1 AIN11 Pin Present When set, indicates that ADC module 1 input pin 11 is present.
26	ADC1AIN10	RO	1	ADC Module 1 AIN10 Pin Present When set, indicates that ADC module 1 input pin 10 is present.
25	ADC1AIN9	RO	1	ADC Module 1 AIN9 Pin Present When set, indicates that ADC module 1 input pin 9 is present.
24	ADC1AIN8	RO	1	ADC Module 1 AIN8 Pin Present When set, indicates that ADC module 1 input pin 8 is present.
23	ADC1AIN7	RO	1	ADC Module 1 AIN7 Pin Present When set, indicates that ADC module 1 input pin 7 is present.
22	ADC1AIN6	RO	1	ADC Module 1 AIN6 Pin Present When set, indicates that ADC module 1 input pin 6 is present.
21	ADC1AIN5	RO	1	ADC Module 1 AIN5 Pin Present When set, indicates that ADC module 1 input pin 5 is present.
20	ADC1AIN4	RO	1	ADC Module 1 AIN4 Pin Present When set, indicates that ADC module 1 input pin 4 is present.

Bit/Field	Name	Type	Reset	Description
19	ADC1AIN3	RO	1	ADC Module 1 AIN3 Pin Present When set, indicates that ADC module 1 input pin 3 is present.
18	ADC1AIN2	RO	1	ADC Module 1 AIN2 Pin Present When set, indicates that ADC module 1 input pin 2 is present.
17	ADC1AIN1	RO	1	ADC Module 1 AIN1 Pin Present When set, indicates that ADC module 1 input pin 1 is present.
16	ADC1AIN0	RO	1	ADC Module 1 AIN0 Pin Present When set, indicates that ADC module 1 input pin 0 is present.
15	ADC0AIN15	RO	1	ADC Module 0 AIN15 Pin Present When set, indicates that ADC module 0 input pin 15 is present.
14	ADC0AIN14	RO	1	ADC Module 0 AIN14 Pin Present When set, indicates that ADC module 0 input pin 14 is present.
13	ADC0AIN13	RO	1	ADC Module 0 AIN13 Pin Present When set, indicates that ADC module 0 input pin 13 is present.
12	ADC0AIN12	RO	1	ADC Module 0 AIN12 Pin Present When set, indicates that ADC module 0 input pin 12 is present.
11	ADC0AIN11	RO	1	ADC Module 0 AIN11 Pin Present When set, indicates that ADC module 0 input pin 11 is present.
10	ADC0AIN10	RO	1	ADC Module 0 AIN10 Pin Present When set, indicates that ADC module 0 input pin 10 is present.
9	ADC0AIN9	RO	1	ADC Module 0 AIN9 Pin Present When set, indicates that ADC module 0 input pin 9 is present.
8	ADC0AIN8	RO	1	ADC Module 0 AIN8 Pin Present When set, indicates that ADC module 0 input pin 8 is present.
7	ADC0AIN7	RO	1	ADC Module 0 AIN7 Pin Present When set, indicates that ADC module 0 input pin 7 is present.
6	ADC0AIN6	RO	1	ADC Module 0 AIN6 Pin Present When set, indicates that ADC module 0 input pin 6 is present.
5	ADC0AIN5	RO	1	ADC Module 0 AIN5 Pin Present When set, indicates that ADC module 0 input pin 5 is present.
4	ADC0AIN4	RO	1	ADC Module 0 AIN4 Pin Present When set, indicates that ADC module 0 input pin 4 is present.
3	ADC0AIN3	RO	1	ADC Module 0 AIN3 Pin Present When set, indicates that ADC module 0 input pin 3 is present.
2	ADC0AIN2	RO	1	ADC Module 0 AIN2 Pin Present When set, indicates that ADC module 0 input pin 2 is present.

Bit/Field	Name	Type	Reset	Description
1	ADC0AIN1	RO	1	ADC Module 0 AIN1 Pin Present When set, indicates that ADC module 0 input pin 1 is present.
0	ADC0AIN0	RO	1	ADC Module 0 AIN0 Pin Present When set, indicates that ADC module 0 input pin 0 is present.

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

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

Device Capabilities 9 ADC Digital Comparators (DC9)

Base 0x400F.E000
Offset 0x190
Type RO, reset 0x00FF.00FF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved								ADC1DC7	ADC1DC6	ADC1DC5	ADC1DC4	ADC1DC3	ADC1DC2	ADC1DC1	ADC1DC0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								ADC0DC7	ADC0DC6	ADC0DC5	ADC0DC4	ADC0DC3	ADC0DC2	ADC0DC1	ADC0DC0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23	ADC1DC7	RO	1	ADC1 DC7 Present When set, indicates that ADC module 1 Digital Comparator 7 is present.
22	ADC1DC6	RO	1	ADC1 DC6 Present When set, indicates that ADC module 1 Digital Comparator 6 is present.
21	ADC1DC5	RO	1	ADC1 DC5 Present When set, indicates that ADC module 1 Digital Comparator 5 is present.
20	ADC1DC4	RO	1	ADC1 DC4 Present When set, indicates that ADC module 1 Digital Comparator 4 is present.
19	ADC1DC3	RO	1	ADC1 DC3 Present When set, indicates that ADC module 1 Digital Comparator 3 is present.
18	ADC1DC2	RO	1	ADC1 DC2 Present When set, indicates that ADC module 1 Digital Comparator 2 is present.
17	ADC1DC1	RO	1	ADC1 DC1 Present When set, indicates that ADC module 1 Digital Comparator 1 is present.
16	ADC1DC0	RO	1	ADC1 DC0 Present When set, indicates that ADC module 1 Digital Comparator 0 is present.
15:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	ADC0DC7	RO	1	ADC0 DC7 Present When set, indicates that ADC module 0 Digital Comparator 7 is present.

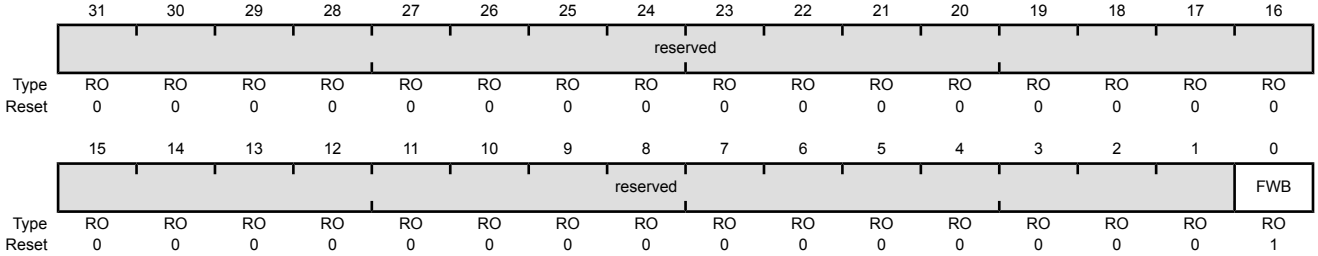
Bit/Field	Name	Type	Reset	Description
6	ADC0DC6	RO	1	ADC0 DC6 Present When set, indicates that ADC module 0 Digital Comparator 6 is present.
5	ADC0DC5	RO	1	ADC0 DC5 Present When set, indicates that ADC module 0 Digital Comparator 5 is present.
4	ADC0DC4	RO	1	ADC0 DC4 Present When set, indicates that ADC module 0 Digital Comparator 4 is present.
3	ADC0DC3	RO	1	ADC0 DC3 Present When set, indicates that ADC module 0 Digital Comparator 3 is present.
2	ADC0DC2	RO	1	ADC0 DC2 Present When set, indicates that ADC module 0 Digital Comparator 2 is present.
1	ADC0DC1	RO	1	ADC0 DC1 Present When set, indicates that ADC module 0 Digital Comparator 1 is present.
0	ADC0DC0	RO	1	ADC0 DC0 Present When set, indicates that ADC module 0 Digital Comparator 0 is present.

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

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

Non-Volatile Memory Information (NVMSTAT)

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



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

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

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

Run Mode Clock Gating Control Register 0 (RCGC0)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved			WDT1	reserved	CAN2	CAN1	CAN0	reserved						ADC1	ADC0
Type	RO	RO	RO	R/W	RO	R/W	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				MAXADC1SPD		MAXADC0SPD		reserved	reserved	reserved		WDT0	reserved		
Type	RO	RO	RO	RO	R/W	R/W	R/W	R/W	RO	RO	RO	RO	R/W	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	WDT1	R/W	0	WDT1 Clock Gating Control This bit controls the clock gating for the Watchdog Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	CAN2	R/W	0	CAN2 Clock Gating Control This bit controls the clock gating for CAN module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
25	CAN1	R/W	0	CAN1 Clock Gating Control This bit controls the clock gating for CAN module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
24	CAN0	R/W	0	CAN0 Clock Gating Control This bit controls the clock gating for CAN module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
23:18	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
17	ADC1	R/W	0	ADC1 Clock Gating Control This bit controls the clock gating for SAR ADC module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
16	ADC0	R/W	0	ADC0 Clock Gating Control This bit controls the clock gating for ADC module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11:10	MAXADC1SPD	R/W	0	ADC1 Sample Speed This field sets the rate at which ADC module 1 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC1SPD bit as follows (all other encodings are reserved): Value Description 0x3 1M samples/second 0x2 500K samples/second 0x1 250K samples/second 0x0 125K samples/second
9:8	MAXADC0SPD	R/W	0	ADC0 Sample Speed This field sets the rate at which ADC0 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC0SPD bit as follows (all other encodings are reserved): Value Description 0x3 1M samples/second 0x2 500K samples/second 0x1 250K samples/second 0x0 125K samples/second
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

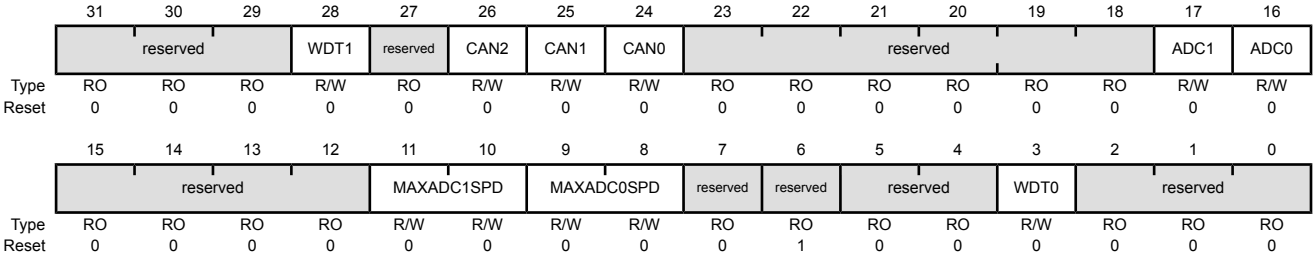
Bit/Field	Name	Type	Reset	Description
6	reserved	RO	1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:4	reserved	RO	0	Software should not rely on the value of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

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

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

Sleep Mode Clock Gating Control Register 0 (SCGC0)

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



Bit/Field	Name	Type	Reset	Description
31:29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	WDT1	R/W	0	WDT1 Clock Gating Control This bit controls the clock gating for Watchdog Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	CAN2	R/W	0	CAN2 Clock Gating Control This bit controls the clock gating for CAN module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
25	CAN1	R/W	0	CAN1 Clock Gating Control This bit controls the clock gating for CAN module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
24	CAN0	R/W	0	CAN0 Clock Gating Control This bit controls the clock gating for CAN module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
23:18	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
17	ADC1	R/W	0	ADC1 Clock Gating Control This bit controls the clock gating for ADC module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
16	ADC0	R/W	0	ADC0 Clock Gating Control This bit controls the clock gating for ADC module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11:10	MAXADC1SPD	R/W	0	ADC1 Sample Speed This field sets the rate at which ADC module 1 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC1SPD bit as follows (all other encodings are reserved): Value Description 0x3 1M samples/second 0x2 500K samples/second 0x1 250K samples/second 0x0 125K samples/second
9:8	MAXADC0SPD	R/W	0	ADC0 Sample Speed This field sets the rate at which ADC module 0 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC0SPD bit as follows (all other encodings are reserved): Value Description 0x3 1M samples/second 0x2 500K samples/second 0x1 250K samples/second 0x0 125K samples/second
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

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

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

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

Deep Sleep Mode Clock Gating Control Register 0 (DCGC0)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved			WDT1	reserved	CAN2	CAN1	CAN0	reserved						ADC1	ADC0
Type	RO	RO	RO	R/W	RO	R/W	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									reserved	reserved		WDT0	reserved		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	WDT1	R/W	0	WDT1 Clock Gating Control This bit controls the clock gating for the Watchdog Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	CAN2	R/W	0	CAN2 Clock Gating Control This bit controls the clock gating for CAN module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
25	CAN1	R/W	0	CAN1 Clock Gating Control This bit controls the clock gating for CAN module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
24	CAN0	R/W	0	CAN0 Clock Gating Control This bit controls the clock gating for CAN module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
23:18	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
17	ADC1	R/W	0	ADC1 Clock Gating Control This bit controls the clock gating for ADC module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
16	ADC0	R/W	0	ADC0 Clock Gating Control This bit controls the clock gating for ADC module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	reserved	RO	1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT0	R/W	0	WDT0 Clock Gating Control This bit controls the clock gating for the Watchdog Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

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

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

Run Mode Clock Gating Control Register 1 (RCGC1)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved	EPI0	reserved	I2S0	reserved	COMP2	COMP1	COMP0	reserved				TIMER3	TIMER2	TIMER1	TIMER0
Type	RO	R/W	RO	R/W	RO	R/W	R/W	R/W	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPI0	R/W	0	EPI0 Clock Gating This bit controls the clock gating for EPI module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	I2S0	R/W	0	I2S0 Clock Gating This bit controls the clock gating for I2S module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
26	COMP2	R/W	0	Analog Comparator 2 Clock Gating This bit controls the clock gating for analog comparator 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
25	COMP1	R/W	0	Analog Comparator 1 Clock Gating This bit controls the clock gating for analog comparator 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
24	COMP0	R/W	0	Analog Comparator 0 Clock Gating This bit controls the clock gating for analog comparator 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	R/W	0	Timer 3 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 3. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
18	TIMER2	R/W	0	Timer 2 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
17	TIMER1	R/W	0	Timer 1 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	R/W	0	I2C1 Clock Gating Control This bit controls the clock gating for I2C module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

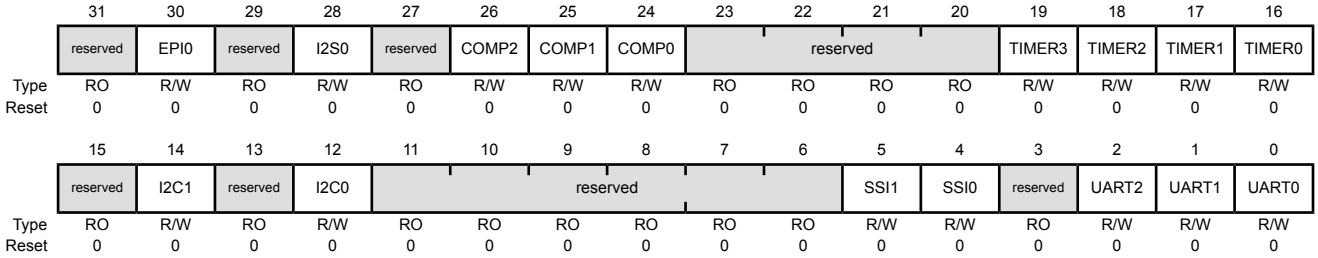
Bit/Field	Name	Type	Reset	Description
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	I2C0 Clock Gating Control This bit controls the clock gating for I2C module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
11:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SSI1	R/W	0	SSI1 Clock Gating Control This bit controls the clock gating for SSI module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
4	SSI0	R/W	0	SSI0 Clock Gating Control This bit controls the clock gating for SSI module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	R/W	0	UART2 Clock Gating Control This bit controls the clock gating for UART module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
1	UART1	R/W	0	UART1 Clock Gating Control This bit controls the clock gating for UART module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
0	UART0	R/W	0	UART0 Clock Gating Control This bit controls the clock gating for UART module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

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

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

Sleep Mode Clock Gating Control Register 1 (SCGC1)

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



Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPI0	R/W	0	EPI0 Clock Gating This bit controls the clock gating for EPI module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	I2S0	R/W	0	I2S0 Clock Gating This bit controls the clock gating for I2S module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
26	COMP2	R/W	0	Analog Comparator 2 Clock Gating This bit controls the clock gating for analog comparator 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
25	COMP1	R/W	0	Analog Comparator 1 Clock Gating This bit controls the clock gating for analog comparator 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
24	COMP0	R/W	0	Analog Comparator 0 Clock Gating This bit controls the clock gating for analog comparator 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	R/W	0	Timer 3 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 3. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
18	TIMER2	R/W	0	Timer 2 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
17	TIMER1	R/W	0	Timer 1 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	R/W	0	I2C1 Clock Gating Control This bit controls the clock gating for I2C module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	I2C0 Clock Gating Control This bit controls the clock gating for I2C module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
11:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SSI1	R/W	0	SSI1 Clock Gating Control This bit controls the clock gating for SSI module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
4	SSI0	R/W	0	SSI0 Clock Gating Control This bit controls the clock gating for SSI module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	R/W	0	UART2 Clock Gating Control This bit controls the clock gating for UART module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
1	UART1	R/W	0	UART1 Clock Gating Control This bit controls the clock gating for UART module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
0	UART0	R/W	0	UART0 Clock Gating Control This bit controls the clock gating for UART module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

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

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

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

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved	EPI0	reserved	I2S0	reserved	COMP2	COMP1	COMP0	reserved				TIMER3	TIMER2	TIMER1	TIMER0
Type	RO	R/W	RO	R/W	RO	R/W	R/W	R/W	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPI0	R/W	0	EPI0 Clock Gating This bit controls the clock gating for EPI module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	I2S0	R/W	0	I2S0 Clock Gating This bit controls the clock gating for I2S module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
26	COMP2	R/W	0	Analog Comparator 2 Clock Gating This bit controls the clock gating for analog comparator 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
25	COMP1	R/W	0	Analog Comparator 1 Clock Gating This bit controls the clock gating for analog comparator 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
24	COMP0	R/W	0	Analog Comparator 0 Clock Gating This bit controls the clock gating for analog comparator 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	R/W	0	Timer 3 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 3. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
18	TIMER2	R/W	0	Timer 2 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
17	TIMER1	R/W	0	Timer 1 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	R/W	0	I2C1 Clock Gating Control This bit controls the clock gating for I2C module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

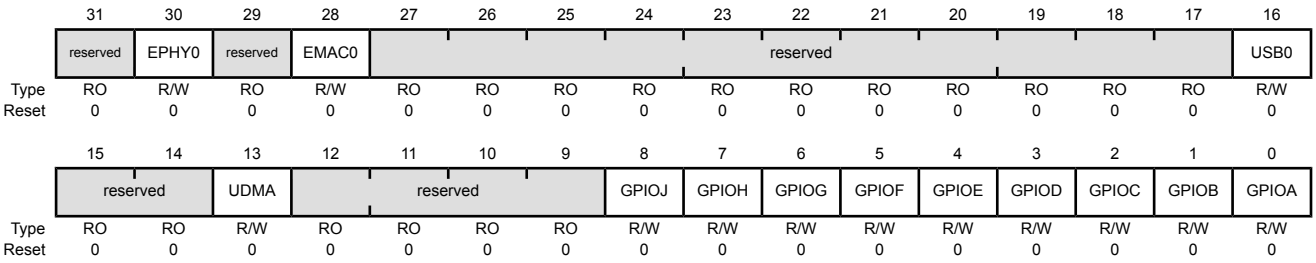
Bit/Field	Name	Type	Reset	Description
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	I2C0 Clock Gating Control This bit controls the clock gating for I2C module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
11:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SSI1	R/W	0	SSI1 Clock Gating Control This bit controls the clock gating for SSI module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
4	SSI0	R/W	0	SSI0 Clock Gating Control This bit controls the clock gating for SSI module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	R/W	0	UART2 Clock Gating Control This bit controls the clock gating for UART module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
1	UART1	R/W	0	UART1 Clock Gating Control This bit controls the clock gating for UART module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
0	UART0	R/W	0	UART0 Clock Gating Control This bit controls the clock gating for UART module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

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

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

Run Mode Clock Gating Control Register 2 (RCGC2)

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



Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPHY0	R/W	0	PHY0 Clock Gating Control This bit controls the clock gating for Ethernet PHY layer 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	EMAC0	R/W	0	MAC0 Clock Gating Control This bit controls the clock gating for Ethernet MAC layer 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
27:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
16	USB0	R/W	0	USB0 Clock Gating Control This bit controls the clock gating for USB module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	UDMA	R/W	0	Micro-DMA Clock Gating Control This bit controls the clock gating for micro-DMA. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
12:9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	GPIOJ	R/W	0	Port J Clock Gating Control This bit controls the clock gating for Port J. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
7	GPIOH	R/W	0	Port H Clock Gating Control This bit controls the clock gating for Port H. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
6	GPIOG	R/W	0	Port G Clock Gating Control This bit controls the clock gating for Port G. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
5	GPIOF	R/W	0	Port F Clock Gating Control This bit controls the clock gating for Port F. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
4	GPIOE	R/W	0	Port E Clock Gating Control Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
3	GPIOD	R/W	0	Port D Clock Gating Control Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

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

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

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

Sleep Mode Clock Gating Control Register 2 (SCGC2)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved	EPHY0	reserved	EMAC0	reserved											USB0
Type RO	R/W	RO	R/W	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved		UDMA	reserved				GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Type RO	RO	R/W	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPHY0	R/W	0	PHY0 Clock Gating Control This bit controls the clock gating for Ethernet PHY layer 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	EMAC0	R/W	0	MAC0 Clock Gating Control This bit controls the clock gating for Ethernet MAC layer 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
27:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
16	USB0	R/W	0	USB0 Clock Gating Control This bit controls the clock gating for USB module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	UDMA	R/W	0	Micro-DMA Clock Gating Control This bit controls the clock gating for micro-DMA. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
12:9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	GPIOJ	R/W	0	Port J Clock Gating Control This bit controls the clock gating for Port J. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
7	GPIOH	R/W	0	Port H Clock Gating Control This bit controls the clock gating for Port H. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
6	GPIOG	R/W	0	Port G Clock Gating Control This bit controls the clock gating for Port G. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
5	GPIOF	R/W	0	Port F Clock Gating Control This bit controls the clock gating for Port F. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
4	GPIOE	R/W	0	Port E Clock Gating Control Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
3	GPIOD	R/W	0	Port D Clock Gating Control Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

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

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

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

Deep Sleep Mode Clock Gating Control Register 2 (DCGC2)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved	EPHY0	reserved	EMAC0	reserved											USB0
Type	RO	R/W	RO	R/W	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		UDMA	reserved				GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	R/W	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPHY0	R/W	0	PHY0 Clock Gating Control This bit controls the clock gating for Ethernet PHY layer 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	EMAC0	R/W	0	MAC0 Clock Gating Control This bit controls the clock gating for Ethernet MAC layer 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
27:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
16	USB0	R/W	0	USB0 Clock Gating Control This bit controls the clock gating for USB module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	UDMA	R/W	0	Micro-DMA Clock Gating Control This bit controls the clock gating for micro-DMA. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
12:9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	GPIOJ	R/W	0	Port J Clock Gating Control This bit controls the clock gating for Port J. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
7	GPIOH	R/W	0	Port H Clock Gating Control This bit controls the clock gating for Port H. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
6	GPIOG	R/W	0	Port G Clock Gating Control This bit controls the clock gating for Port G. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
5	GPIOF	R/W	0	Port F Clock Gating Control This bit controls the clock gating for Port F. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
4	GPIOE	R/W	0	Port E Clock Gating Control Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
3	GPIOD	R/W	0	Port D Clock Gating Control Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

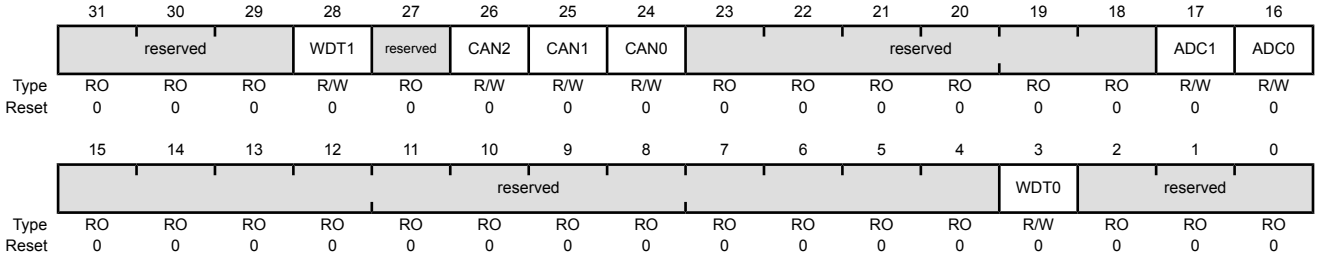
Bit/Field	Name	Type	Reset	Description
2	GPIOC	R/W	0	Port C Clock Gating Control This bit controls the clock gating for Port C. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus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

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

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

Software Reset Control 0 (SRCR0)

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



Bit/Field	Name	Type	Reset	Description
31:29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	WDT1	R/W	0	WDT1 Reset Control When this bit is set, Watchdog Timer module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	CAN2	R/W	0	CAN2 Reset Control When this bit is set, CAN module 2 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
25	CAN1	R/W	0	CAN1 Reset Control When this bit is set, CAN module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
24	CAN0	R/W	0	CAN0 Reset Control When this bit is set, CAN module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
23:18	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
17	ADC1	R/W	0	ADC1 Reset Control When this bit is set, ADC module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.

Bit/Field	Name	Type	Reset	Description
16	ADC0	R/W	0	ADC0 Reset Control When this bit is set, ADC module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
15:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT0	R/W	0	WDT0 Reset Control When this bit is set, Watchdog Timer module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 37: Software Reset Control 1 (SRCR1), offset 0x044

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

Software Reset Control 1 (SRCR1)
Base 0x400F.E000
Offset 0x044
Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved	EPI0	reserved	I2S0	reserved	COMP2	COMP1	COMP0	reserved				TIMER3	TIMER2	TIMER1	TIMER0
Type	RO	R/W	RO	R/W	RO	R/W	R/W	R/W	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPI0	R/W	0	EPI0 Reset Control When this bit is set, EPI module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	I2S0	R/W	0	I2S0 Reset Control When this bit is set, I2S module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	COMP2	R/W	0	Analog Comp 2 Reset Control When this bit is set, Analog Comparator module 2 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
25	COMP1	R/W	0	Analog Comp 1 Reset Control When this bit is set, Analog Comparator module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
24	COMP0	R/W	0	Analog Comp 0 Reset Control When this bit is set, Analog Comparator module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.

Bit/Field	Name	Type	Reset	Description
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	R/W	0	Timer 3 Reset Control Timer 3 Reset Control. When this bit is set, General-Purpose Timer module 3 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
18	TIMER2	R/W	0	Timer 2 Reset Control When this bit is set, General-Purpose Timer module 2 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
17	TIMER1	R/W	0	Timer 1 Reset Control When this bit is set, General-Purpose Timer module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
16	TIMER0	R/W	0	Timer 0 Reset Control When this bit is set, General-Purpose Timer module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	R/W	0	I2C1 Reset Control When this bit is set, I2C module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	I2C0 Reset Control When this bit is set, I2C module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
11:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SSI1	R/W	0	SSI1 Reset Control When this bit is set, SSI module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
4	SSI0	R/W	0	SSI0 Reset Control When this bit is set, SSI module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.

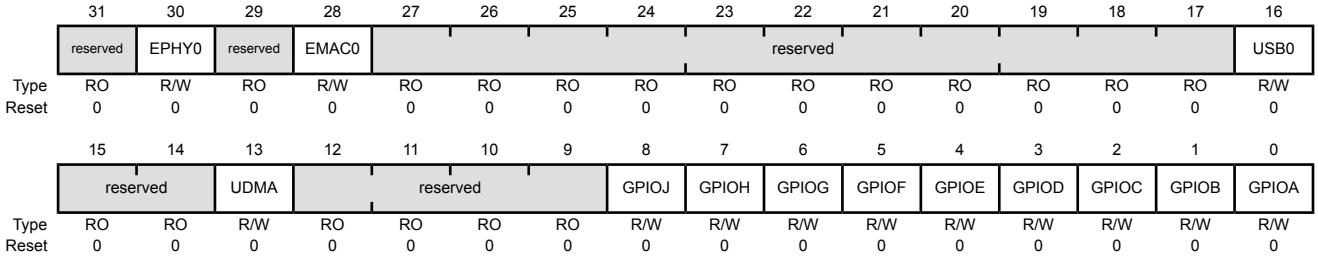
Bit/Field	Name	Type	Reset	Description
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	R/W	0	UART2 Reset Control When this bit is set, UART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

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

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

Software Reset Control 2 (SRCR2)

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



Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPHY0	R/W	0	PHY0 Reset Control When this bit is set, Ethernet PHY layer 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	EMAC0	R/W	0	MAC0 Reset Control When this bit is set, Ethernet MAC layer 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
27:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	USB0	R/W	0	USB0 Reset Control When this bit is set, USB module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
15:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	UDMA	R/W	0	Micro-DMA Reset Control When this bit is set, uDMA module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.

Bit/Field	Name	Type	Reset	Description
12:9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	GPIOJ	R/W	0	Port J Reset Control When this bit is set, Port J module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
7	GPIOH	R/W	0	Port H Reset Control When this bit is set, Port H module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
6	GPIOG	R/W	0	Port G Reset Control When this bit is set, Port G module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
5	GPIOF	R/W	0	Port F Reset Control When this bit is set, Port F module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
4	GPIOE	R/W	0	Port E Reset Control When this bit is set, Port E module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
3	GPIOD	R/W	0	Port D Reset Control When this bit is set, Port D module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
2	GPIOC	R/W	0	Port C Reset Control When this bit is set, Port C module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
1	GPIOB	R/W	0	Port B Reset Control When this bit is set, Port B module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
0	GPIOA	R/W	0	Port A Reset Control When this bit is set, Port A module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.

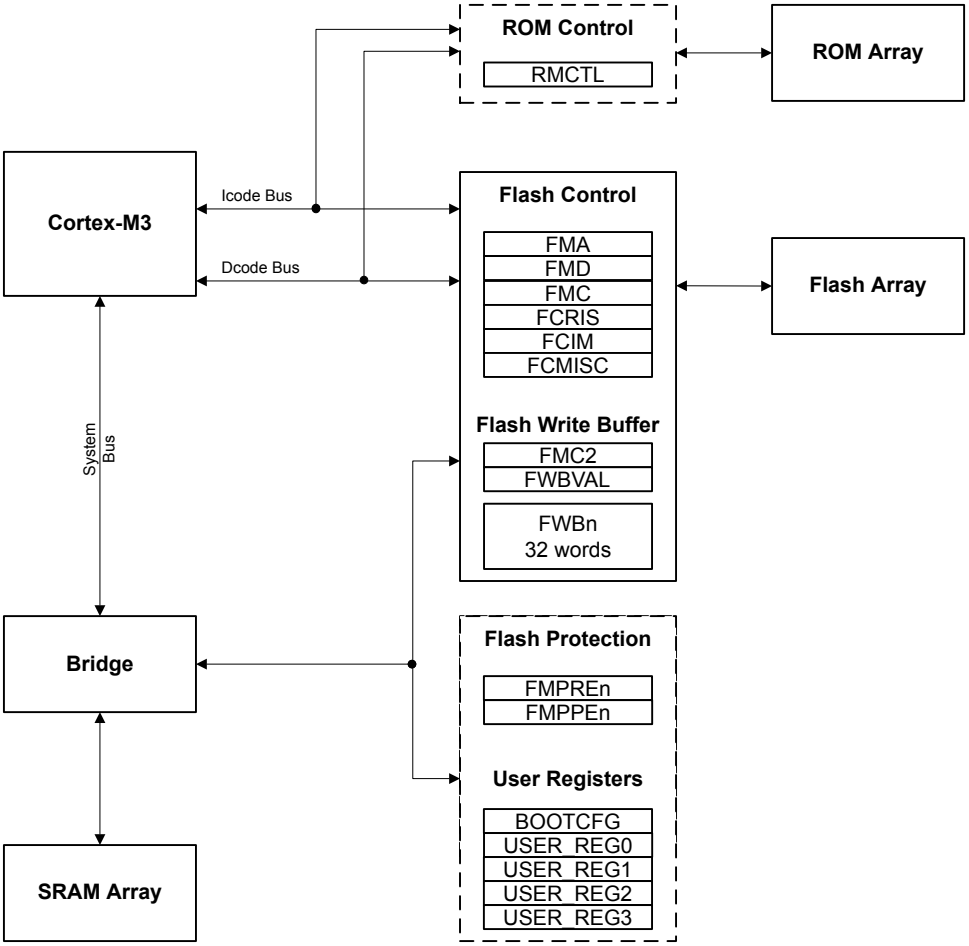
6 Internal Memory

The LM3S9U81 microcontroller comes with 96 KB of bit-banded SRAM, internal ROM,and 384 KB of Flash memory. The Flash memory controller provides a user-friendly interface, making Flash memory programming a simple task. Flash memory protection can be applied to the Flash memory on a 2-KB block basis.

6.1 Block Diagram

Figure 6-1 on page 288 illustrates the internal memory blocks and control logic. The dashed boxes in the figure indicate registers residing in the System Control module.

Figure 6-1. Internal Memory Block Diagram



6.2 Functional Description

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

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

6.2.1 SRAM

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

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

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

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

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

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

For details about bit-banding, see “Bit-Banding” on page 89.

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

6.2.2 ROM

The internal ROM of the Stellaris device is located at address 0x0100.0000 of the device memory map. Detailed information on the ROM contents can be found in the *Stellaris® ROM User's Guide*.

The ROM contains the following components:

- Stellaris Boot Loader and vector table
- Stellaris Peripheral Driver Library (DriverLib) release for product-specific peripherals and interfaces
- Advanced Encryption Standard (AES) cryptography tables
- Cyclic Redundancy Check (CRC) error detection functionality

The boot loader is used as an initial program loader (when the Flash memory is empty) as well as an application-initiated firmware upgrade mechanism (by calling back to the boot loader). The Peripheral Driver Library APIs in ROM can be called by applications, reducing Flash memory requirements and freeing the Flash memory to be used for other purposes (such as additional features in the application). Advance Encryption Standard (AES) is a publicly defined encryption standard used by the U.S. Government and Cyclic Redundancy Check (CRC) is a technique to validate a span of data has the same contents as when previously checked.

6.2.2.1 Boot Loader Overview

The Stellaris Boot Loader is used to download code to the Flash memory of a device without the use of a debug interface. When the core is reset, the user has the opportunity to direct the core to execute the ROM Boot Loader or the application in Flash memory by using any GPIO signal in Ports A-H as configured in the **Boot Configuration (BOOTCFG)** register.

At reset, the ROM is mapped over the Flash memory so that the ROM boot sequence is always executed. The boot sequence executed from ROM is as follows:

1. The BA bit (below) is cleared such that ROM is mapped to 0x01xx.xxxx and Flash memory is mapped to address 0x0.
2. The **BOOTCFG** register is read. If the EN bit is clear, the status of the specified GPIO pin is compared with the specified polarity. If the status matches the specified polarity, the ROM is mapped to address 0x0000.0000 and execution continues out of the ROM Boot Loader.
3. If the status doesn't match the specified polarity, the data at address 0x0000.0004 is read, and if the data at this address is 0xFFFF.FFFF, the ROM is mapped to address 0x0000.0000 and execution continues out of the ROM Boot Loader.
4. If there is data at address 0x0000.0004 that is not 0xFFFF.FFFF, the stack pointer (**SP**) is loaded from Flash memory at address 0x0000.0000 and the program counter (**PC**) is loaded from address 0x0000.0004. The user application begins executing.

The boot loader uses a simple packet interface to provide synchronous communication with the device. The speed of the boot loader is determined by the internal oscillator (PIOSC) frequency as it does not enable the PLL. The following serial interfaces can be used:

- UART0
- SSI0
- I^2C0
- Ethernet

For simplicity, both the data format and communication protocol are identical for all serial interfaces.

Note: The Flash-memory-resident version of the Boot Loader also supports CAN and USB.

See the *Stellaris® Boot Loader User's Guide* for information on the boot loader software.

6.2.2.2 Stellaris Peripheral Driver Library

The Stellaris Peripheral Driver Library contains a file called `driverlib/rom.h` that assists with calling the peripheral driver library functions in the ROM. The detailed description of each function is available in the *Stellaris® ROM User's Guide*. See the "Using the ROM" chapter of the *Stellaris® Peripheral Driver Library User's Guide* for more details on calling the ROM functions and using `driverlib/rom.h`.

A table at the beginning of the ROM points to the entry points for the APIs that are provided in the ROM. Accessing the API through these tables provides scalability; while the API locations may change in future versions of the ROM, the API tables will not. The tables are split into two levels; the main table contains one pointer per peripheral which points to a secondary table that contains one pointer per API that is associated with that peripheral. The main table is located at 0x0100.0010, right after the Cortex-M3 vector table in the ROM.

DriverLib functions are described in detail in the *Stellaris® Peripheral Driver Library User's Guide*.

Additional APIs are available for graphics and USB functions, but are not preloaded into ROM. The Stellaris Graphics Library provides a set of graphics primitives and a widget set for creating graphical user interfaces on Stellaris microcontroller-based boards that have a graphical display (for more information, see the *Stellaris® Graphics Library User's Guide*). The Stellaris USB Library is a set

of data types and functions for creating USB Device, Host or On-The-Go (OTG) applications on Stellaris microcontroller-based boards (for more information, see the *Stellaris® USB Library User's Guide*).

6.2.2.3 Advanced Encryption Standard (AES) Cryptography Tables

AES is a strong encryption method with reasonable performance and size. AES is fast in both hardware and software, is fairly easy to implement, and requires little memory. AES is ideal for applications that can use pre-arranged keys, such as setup during manufacturing or configuration. Four data tables used by the XySSL AES implementation are provided in the ROM. The first is the forward S-box substitution table, the second is the reverse S-box substitution table, the third is the forward polynomial table, and the final is the reverse polynomial table. See the *Stellaris® ROM User's Guide* for more information on AES.

6.2.2.4 Cyclic Redundancy Check (CRC) Error Detection

The CRC technique can be used to validate correct receipt of messages (nothing lost or modified in transit), to validate data after decompression, to validate that Flash memory contents have not been changed, and for other cases where the data needs to be validated. A CRC is preferred over a simple checksum (e.g. XOR all bits) because it catches changes more readily. See the *Stellaris® ROM User's Guide* for more information on CRC.

6.2.3 Flash Memory

At system clock speeds of 50 MHz and below, the Flash memory is read in a single cycle. The Flash memory is organized as a set of 1-KB blocks that can be individually erased. An individual 32-bit word can be programmed to change bits from 1 to 0. In addition, a write buffer provides the ability to concurrently program 32 continuous words in Flash memory. Erasing a block causes the entire contents of the block to be reset to all 1s. The 1-KB blocks are paired into sets of 2-KB blocks that can be individually protected. The protection allows blocks to be marked as read-only or execute-only, providing different levels of code protection. Read-only blocks cannot be erased or programmed, protecting the contents of those blocks from being modified. Execute-only blocks cannot be erased or programmed and can only be read by the controller instruction fetch mechanism, protecting the contents of those blocks from being read by either the controller or by a debugger.

Caution – The Stellaris Flash memory array has ECC which uses a test port into the Flash memory to continually scan the array for ECC errors and to correct any that are detected. This operation is transparent to the microcontroller. The BIST must scan the entire memory array occasionally to ensure integrity, taking about five minutes to do so. In systems where the microcontroller is frequently powered for less than five minutes, power should be removed from the microcontroller in a controlled manner to ensure proper operation. Software can request permission to power down the part using the USDREQ bit in the Flash Control (FCTL) register and wait to receive an acknowledge from the USDACK bit prior to removing power. If the microcontroller is powered down using this controlled method, the BIST engine keeps track of where it was in the memory array and it always scans the complete array after any aggregate of five minutes powered-on, regardless of the number of intervening power cycles. If the microcontroller is powered down before five minutes of being powered up, BIST starts again from wherever it left off before the last controlled power-down or from 0 if there never was a controlled power down. An occasional short power down is not a concern, but the microcontroller should not always be powered down frequently in an uncontrolled manner. The microcontroller can be power-cycled as frequently as necessary if it is powered-down in a controlled manner.

6.2.3.1 Prefetch Buffer

The Flash memory controller has a prefetch buffer that is automatically used when the CPU frequency is greater than 50 MHz. In this mode, the Flash memory operates at half of the system clock. The prefetch buffer fetches two 32-bit words per clock allowing instructions to be fetched with no wait states while code is executing linearly. The fetch buffer includes a branch speculation mechanism that recognizes a branch and avoids extra wait states by not reading the next word pair. Also, short loop branches often stay in the buffer. As a result, some branches can be executed with no wait states. Other branches incur a single wait state.

6.2.3.2 Flash Memory Protection

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

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

The policies may be combined as shown in Table 6-1 on page 292.

Table 6-1. Flash Memory Protection Policy Combinations

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

A Flash memory access that attempts to read a read-protected block (**FMPREn** bit is set) is prohibited and generates a bus fault. A Flash memory access that attempts to program or erase a program-protected block (**FMPPEn** bit is set) is prohibited and can optionally generate an interrupt (by setting the AMASK bit in the **Flash Controller Interrupt Mask (FCIM)** register) to alert software developers of poorly behaving software during the development and debug phases. Note that if a **FMPREn** bit is cleared, all read accesses to the Flash memory block are disallowed, including any data accesses. Care must be taken not to store required data in a Flash memory block that has the associated **FMPREn** bit cleared.

The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. These settings create a policy of open access and programmability. The register bits may be changed by clearing the specific register bit. The changes are effective immediately, but 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 any type of reset sequence. The changes are committed using the **Flash Memory Control (FMC)** register. Details on programming these bits are discussed in “Non-Volatile Register Programming” on page 295.

6.2.3.3 Interrupts

The Flash memory controller can generate interrupts when the following conditions are observed:

- Programming Interrupt - signals when a program or erase action is complete.
- Access Interrupt - signals when a program or erase action has been attempted on a 2-kB block of memory that is protected by its corresponding **FMPPEn** bit.

The interrupt events that can trigger a controller-level interrupt are defined in the **Flash Controller Masked Interrupt Status (FCMIS)** register (see page 305) by setting the corresponding MASK bits. If interrupts are not used, the raw interrupt status is always visible via the **Flash Controller Raw Interrupt Status (FCRIS)** register (see page 304).

Interrupts are always cleared (for both the **FCMIS** and **FCRIS** registers) by writing a 1 to the corresponding bit in the **Flash Controller Masked Interrupt Status and Clear (FCMISC)** register (see page 306).

6.2.3.4 Flash Memory Programming

The Stellaris devices provide a user-friendly interface for Flash memory programming. All erase/program operations are handled via three registers: **Flash Memory Address (FMA)**, **Flash Memory Data (FMD)**, and **Flash Memory Control (FMC)**. Note that if the debug capabilities of the microcontroller have been deactivated, resulting in a "locked" state, a recovery sequence must be performed in order to reactivate the debug module. See “Recovering a "Locked" Microcontroller” on page 180.

During a Flash memory operation (write, page erase, or mass erase) access to the Flash memory is inhibited. As a result, instruction and literal fetches are held off until the Flash memory operation is complete. If instruction execution is required during a Flash memory operation, the code that is executing must be placed in SRAM and executed from there while the flash operation is in progress.

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

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

To program a 32-bit word

1. Write source data to the **FMD** register.
2. Write the target address to the **FMA** register.

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

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

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

The following sequence is NOT allowed:

- ERASE -> PROGRAM value -> PROGRAM value

To perform an erase of a 1-KB page

1. Write the page address to the **FMA** register.
2. Write the Flash memory write key and the ERASE bit (a value of 0xA442.0002) to the **FMC** register.
3. Poll the **FMC** register until the ERASE bit is cleared or, alternatively, enable the programming interrupt using the PMASK bit in the **FCIM** register.

To perform a mass erase of the Flash memory

1. Write the Flash memory write key and the MERASE bit (a value of 0xA442.0004) to the **FMC** register.
2. Poll the **FMC** register until the MERASE bit is cleared or, alternatively, enable the programming interrupt using the PMASK bit in the **FCIM** register.

6.2.3.5 32-Word Flash Memory Write Buffer

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

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

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

1. Write the source data to the **FWBn** registers.

2. Write the target address to the **FMA** register. This must be a 32-word aligned address (that is, bits [6:0] in **FMA** must be 0s).

3. Write the Flash memory write key and the WRBUF bit (a value of 0xA442.0001) to the **FMC2** register.

4. Poll the **FMC2** register until the WRBUF bit is cleared or wait for the PMIS interrupt to be signaled.

6.2.3.6 Non-Volatile Register Programming

This section discusses how to update the registers shown in Table 6-2 on page 296 that are resident within the Flash memory itself. These registers exist in a separate space from the main Flash memory array and are not affected by an ERASE or MASS ERASE operation. With the exception of the **Boot Configuration (BOOTCFG)** register, the settings in these registers can be written, their functions verified, and their values read back before they are committed, at which point they become non-volatile. If a value in one of these registers has not been committed, any type of reset restores the last committed value or the default value if the register has never been committed. Once the register contents are committed, the only way to restore the factory default values is to perform the sequence described in “Recovering a "Locked" Microcontroller” on page 180.

To write to a non-volatile register:

- Bits can only be changed from 1 to 0.
- For all registers except the **BOOTCFG** register, write the data to the register address provided in the register description. For the **BOOTCFG** register, write the data to the **FMD** register.
- The registers can be read to verify their contents. To verify what is to be stored in the **BOOTCFG** register, read the **FMD** register. Reading the **BOOTCFG** register returns the previously committed value or the default value if the register has never been committed.
- The new values are effectively immediately for all registers except **BOOTCFG**, as the new value for the register is not stored in the register until it has been committed.
- Prior to committing the register value, any type of reset restores the last committed value or the default value if the register has never been committed.

To commit a new value to a non-volatile register:

- Write the data as described above.
- Write to the **FMA** register the value shown in Table 6-2 on page 296.
- Write the Flash memory write key and set the COMT bit in the **FMC** register. These values must be written to the **FMC** register at the same time.
- Committing a non-volatile register has the same timing as a write to regular Flash memory, defined by T_{PROG}, as shown in Table 24-16 on page 1200. Software can poll the COMT bit in the **FMC** register to determine when the operation is complete, or an interrupt can be enabled by setting the PMASK bit in the **FCIM** register.
- When committing the **BOOTCFG** register, the INVDRIS bit in the **FCRIS** register is set if a bit that has already been committed as a 0 is attempted to be committed as a 1.
- Once the value has been committed, any type of reset has no effect on the register contents.

- Changes to the **BOOTCFG** register are effective after the next reset.
- The NW bit in the **USER_REG0**, **USER_REG1**, **USER_REG2**, **USER_REG3**, and **BOOTCFG** registers is cleared when the register is committed. Once this bit is cleared, additional changes to the register are not allowed.

Important: After being committed, these registers can only be restored to their factory default values by performing the sequence described in “Recovering a "Locked" Microcontroller” on page 180. The mass erase of the main Flash memory array caused by the sequence is performed prior to restoring these registers.

Table 6-2. User-Programmable Flash Memory Resident Registers

Register to be Committed	FMA Value	Data Source
FMPRE0	0x0000.0000	FMPRE0
FMPRE1	0x0000.0002	FMPRE1
FMPRE2	0x0000.0004	FMPRE2
FMPRE3	0x0000.0006	FMPRE3
FMPRE4	0x0000.0008	FMPRE4
FMPRE5	0x0000.000A	FMPRE5
FMPPE0	0x0000.0001	FMPPE0
FMPPE1	0x0000.0003	FMPPE1
FMPPE2	0x0000.0005	FMPPE2
FMPPE3	0x0000.0007	FMPPE3
FMPRE4	0x0000.0009	FMPRE4
FMPRE5	0x0000.000B	FMPRE5
USER_REG0	0x8000.0000	USER_REG0
USER_REG1	0x8000.0001	USER_REG1
USER_REG2	0x8000.0002	USER_REG2
USER_REG3	0x8000.0003	USER_REG3
BOOTCFG	0x7510.0000	FMD

6.3 Register Map

Table 6-3 on page 296 lists the ROM Controller register and the Flash memory and control registers. The offset listed is a hexadecimal increment to the register's address. The Flash memory register offsets are relative to the Flash memory control base address of 0x400F.D000. The ROM and Flash memory protection register offsets are relative to the System Control base address of 0x400F.E000.

Table 6-3. Flash Register Map

Offset	Name	Type	Reset	Description	See page
Flash Memory Registers (Flash Control Offset)					
0x000	FMA	R/W	0x0000.0000	Flash Memory Address	299
0x004	FMD	R/W	0x0000.0000	Flash Memory Data	300
0x008	FMC	R/W	0x0000.0000	Flash Memory Control	301

Table 6-3. Flash Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x00C	FCRIS	RO	0x0000.0000	Flash Controller Raw Interrupt Status	304
0x010	FCIM	R/W	0x0000.0000	Flash Controller Interrupt Mask	305
0x014	FCMISC	R/W1C	0x0000.0000	Flash Controller Masked Interrupt Status and Clear	306
0x020	FMC2	R/W	0x0000.0000	Flash Memory Control 2	307
0x030	FWBVAL	R/W	0x0000.0000	Flash Write Buffer Valid	308
0x0F8	FCTL	R/W	0x0000.0000	Flash Control	309
0x100 - 0x17C	FWBn	R/W	0x0000.0000	Flash Write Buffer n	310
Memory Registers (System Control Offset)					
0x0F0	RMCTL	R/W1C	-	ROM Control	311
0x130	FMPRE0	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 0	312
0x200	FMPRE0	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 0	312
0x134	FMPPE0	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 0	313
0x400	FMPPE0	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 0	313
0x1D0	BOOTCFG	R/W	0xFFFF.FFFE	Boot Configuration	314
0x1E0	USER_REG0	R/W	0xFFFF.FFFF	User Register 0	316
0x1E4	USER_REG1	R/W	0xFFFF.FFFF	User Register 1	317
0x1E8	USER_REG2	R/W	0xFFFF.FFFF	User Register 2	318
0x1EC	USER_REG3	R/W	0xFFFF.FFFF	User Register 3	319
0x204	FMPRE1	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 1	320
0x208	FMPRE2	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 2	321
0x20C	FMPRE3	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 3	322
0x210	FMPRE4	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 4	323
0x214	FMPRE5	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 5	324
0x218	FMPRE6	R/W	0x0000.0000	Flash Memory Protection Read Enable 6	325
0x21C	FMPRE7	R/W	0x0000.0000	Flash Memory Protection Read Enable 7	326
0x404	FMPPE1	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 1	327
0x408	FMPPE2	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 2	328
0x40C	FMPPE3	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 3	329
0x410	FMPPE4	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 4	330
0x414	FMPPE5	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 5	331
0x418	FMPPE6	R/W	0x0000.0000	Flash Memory Protection Program Enable 6	332

Table 6-3. Flash Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x41C	FMPPE7	R/W	0x0000.0000	Flash Memory Protection Program Enable 7	333

6.4 Flash Memory Register Descriptions (Flash Control Offset)

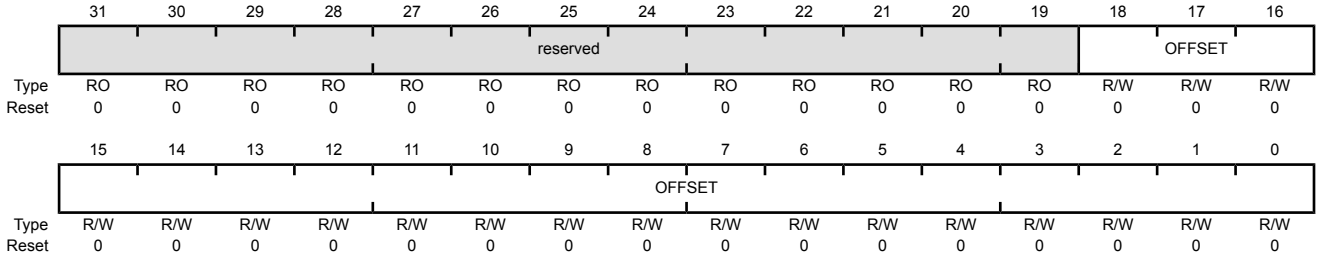
This section lists and describes the Flash Memory registers, in numerical order by address offset. Registers in this section are relative to the Flash control base address of 0x400F.D000.

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

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

Flash Memory Address (FMA)

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



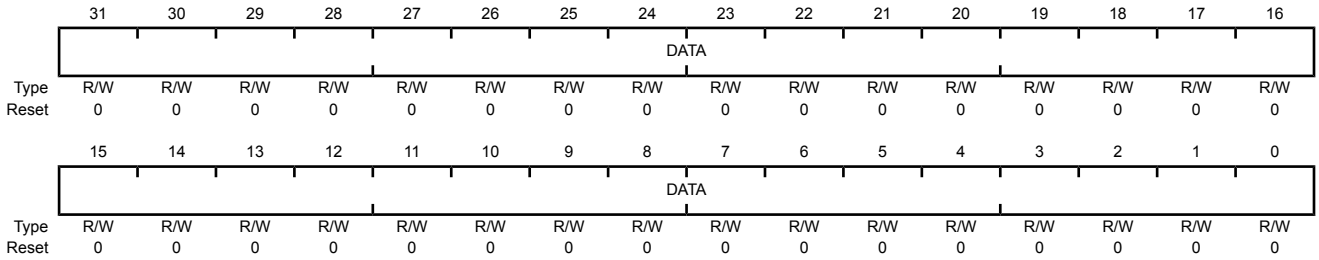
Bit/Field	Name	Type	Reset	Description
31:19	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
18:0	OFFSET	R/W	0x0	Address Offset Address offset in Flash memory where operation is performed, except for non-volatile registers (see “Non-Volatile Register Programming” on page 295 for details on values for this field).

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

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

Flash Memory Data (FMD)

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



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

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

When this register is written, the Flash memory controller initiates the appropriate access cycle for the location specified by the **Flash Memory Address (FMA)** register (see page 299). If the access is a write access, the data contained in the **Flash Memory Data (FMD)** register (see page 300) is written to the specified address.

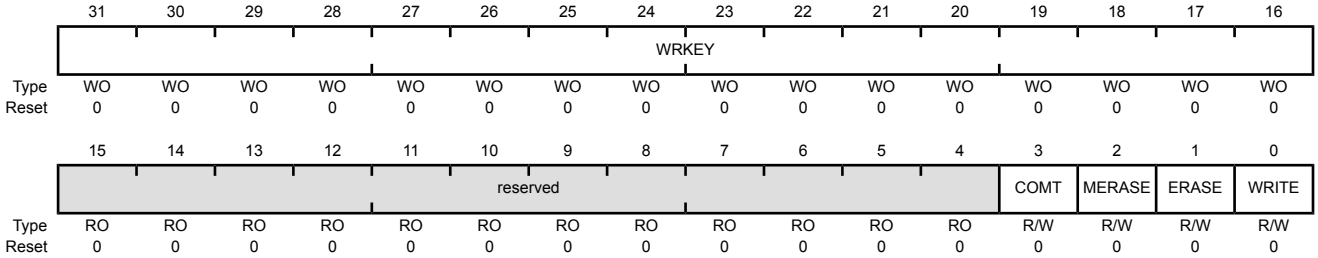
This register must be the final register written and initiates the memory operation. The four control bits in the lower byte of this register are used to initiate memory operations.

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

Caution – If any of bits [15:4] are written to 1, the device may become inoperable. These bits should always be written to 0. In all registers, the value of a reserved bit should be preserved across a read-modify-write operation.

Flash Memory Control (FMC)

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



Bit/Field	Name	Type	Reset	Description
31:16	WRKEY	WO	0x0000	Flash Memory Write Key This field contains a write key, which is used to minimize the incidence of accidental Flash memory writes. The value 0xA442 must be written into this field for a Flash memory write to occur. Writes to the **FMC** register without this WRKEY value are ignored. A read of this field returns the value 0.
15:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
3	COMT	R/W	0	Commit Register Value This bit is used to commit writes to Flash-memory-resident registers and to monitor the progress of that process. Value Description 1 Set this bit to commit (write) the register value to a Flash-memory-resident register. When read, a 1 indicates that the previous commit access is not complete. 0 A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous commit access is complete. See “Non-Volatile Register Programming” on page 295 for more information on programming Flash-memory-resident registers.
2	MERASE	R/W	0	Mass Erase Flash Memory This bit is used to mass erase the Flash main memory and to monitor the progress of that process. Value Description 1 Set this bit to erase the Flash main memory. When read, a 1 indicates that the previous mass erase access is not complete. 0 A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous mass erase access is complete. For information on erase time, see “Flash Memory” on page 1200.
1	ERASE	R/W	0	Erase a Page of Flash Memory This bit is used to erase a page of Flash memory and to monitor the progress of that process. Value Description 1 Set this bit to erase the Flash memory page specified by the contents of the **FMA** register. When read, a 1 indicates that the previous page erase access is not complete. 0 A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous page erase access is complete. For information on erase time, see “Flash Memory” on page 1200.

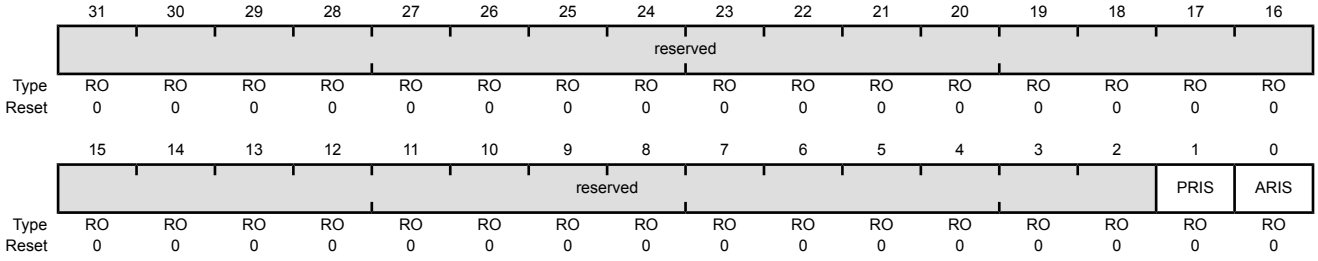
Bit/Field	Name	Type	Reset	Description
0	WRITE	R/W	0	Write a Word into Flash Memory This bit is used to write a word into Flash memory and to monitor the progress of that process. Value Description 1 Set this bit to write the data stored in the **FMD** register into the Flash memory location specified by the contents of the **FMA** register. When read, a 1 indicates that the write update access is not complete. 0 A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous write update access is complete. For information on programming time, see “Flash Memory” on page 1200.

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

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

Flash Controller Raw Interrupt Status (FCRIS)

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



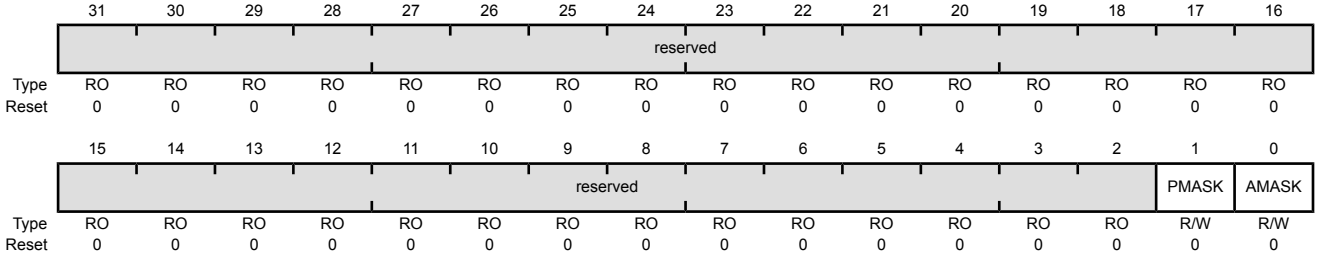
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PRIS	RO	0	Programming Raw Interrupt Status This bit provides status on programming cycles which are write or erase actions generated through the **FMC** or **FMC2** register bits (see page 301 and page 307). Value Description 1 The programming or erase cycle has completed. 0 The programming or erase cycle has not completed. This status is sent to the interrupt controller when the PMASK bit in the **FCIM** register is set. This bit is cleared by writing a 1 to the PMISC bit in the **FCMISC** register.
0	ARIS	RO	0	Access Raw Interrupt Status Value Description 1 A program or erase action was attempted on a block of Flash memory that contradicts the protection policy for that block as set in the **FMPPEn** registers. 0 No access has tried to improperly program or erase the Flash memory. This status is sent to the interrupt controller when the AMASK bit in the **FCIM** register is set. This bit is cleared by writing a 1 to the AMISC bit in the **FCMISC** register.

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

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

Flash Controller Interrupt Mask (FCIM)

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



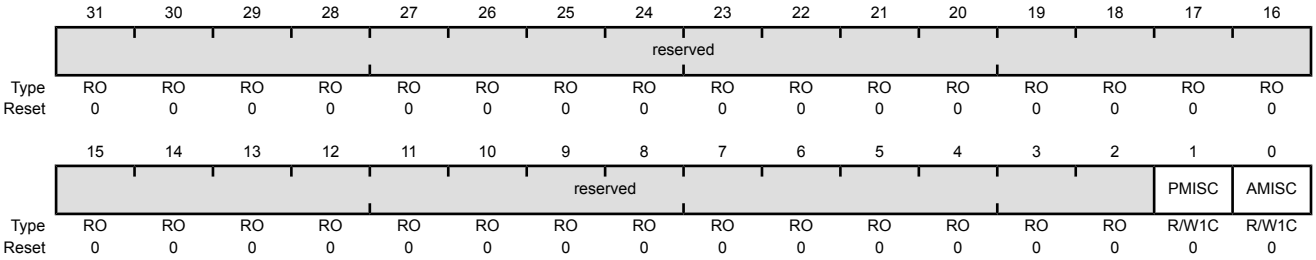
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PMASK	R/W	0	Programming Interrupt Mask This bit controls the reporting of the programming raw interrupt status to the interrupt controller. Value Description 1 An interrupt is sent to the interrupt controller when the PRIS bit is set. 0 The PRIS interrupt is suppressed and not sent to the interrupt controller.
0	AMASK	R/W	0	Access Interrupt Mask This bit controls the reporting of the access raw interrupt status to the interrupt controller. Value Description 1 An interrupt is sent to the interrupt controller when the ARIS bit is set. 0 The ARIS interrupt is suppressed and not sent to the interrupt controller.

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

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

Flash Controller Masked Interrupt Status and Clear (FCMISC)

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



Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PMISC	R/W1C	0	Programming Masked Interrupt Status and Clear Value Description 1 When read, a 1 indicates that an unmasked interrupt was signaled because a programming cycle completed. Writing a 1 to this bit clears PMISC and also the PRIS bit in the **FCRIS** register (see page 304). 0 When read, a 0 indicates that a programming cycle complete interrupt has not occurred. A write of 0 has no effect on the state of this bit.
0	AMISC	R/W1C	0	Access Masked Interrupt Status and Clear Value Description 1 When read, a 1 indicates that an unmasked interrupt was signaled because a program or erase action was attempted on a block of Flash memory that contradicts the protection policy for that block as set in the **FMPPEn** registers. Writing a 1 to this bit clears AMISC and also the ARIS bit in the **FCRIS** register (see page 304). 0 When read, a 0 indicates that no improper accesses have occurred. A write of 0 has no effect on the state of this bit.

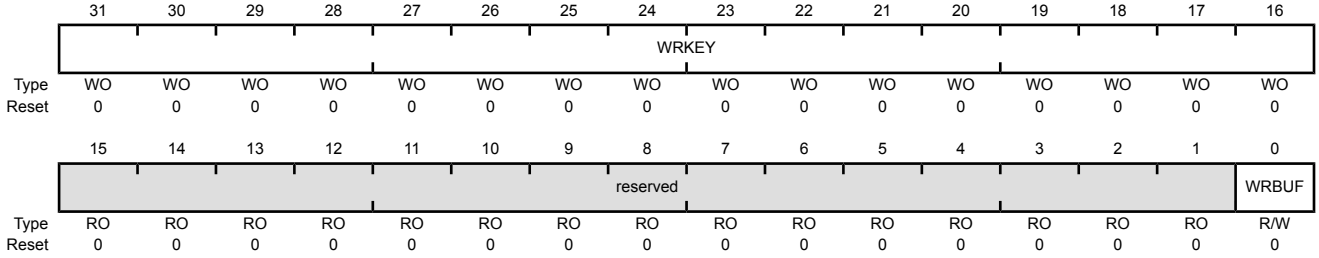
Register 7: Flash Memory Control 2 (FMC2), offset 0x020

When this register is written, the Flash memory controller initiates the appropriate access cycle for the location specified by the **Flash Memory Address (FMA)** register (see page 299). If the access is a write access, the data contained in the **Flash Write Buffer (FWB)** registers is written.

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

Flash Memory Control 2 (FMC2)

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



Bit/Field	Name	Type	Reset	Description
31:16	WRKEY	WO	0x0000	Flash Memory Write Key This field contains a write key, which is used to minimize the incidence of accidental Flash memory writes. The value 0xA442 must be written into this field for a write to occur. Writes to the **FMC2** register without this WRKEY value are ignored. A read of this field returns the value 0.
15:1	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	WRBUF	R/W	0	Buffered Flash Memory Write This bit is used to start a buffered write to Flash memory. Value Description 1 Set this bit to write the data stored in the **FWBn** registers to the location specified by the contents of the **FMA** register. When read, a 1 indicates that the previous buffered Flash memory write access is not complete. 0 A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous buffered Flash memory write access is complete. For information on programming time, see “Flash Memory” on page 1200.

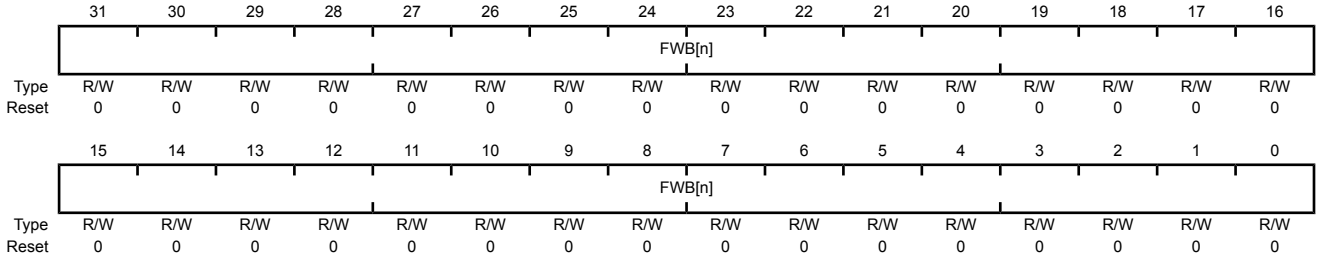
Register 8: Flash Write Buffer Valid (FWBVAL), offset 0x030

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

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

Flash Write Buffer Valid (FWBVAL)

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



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

Value	Description
1	The corresponding **FWBn** register has been updated since the last buffer write operation and is ready to be written to Flash memory.
0	The corresponding **FWBn** register has no new data to be written.

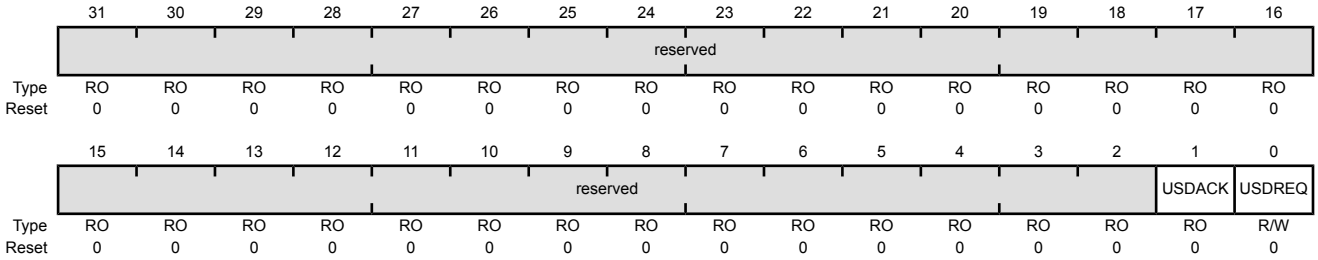
Bit 0 corresponds to **FWB0**, offset 0x100, and bit 31 corresponds to **FWB31**, offset 0x13C.

Register 9: Flash Control (FCTL), offset 0x0F8

This register is used to ensure that the microcontroller is powered down in a controlled fashion in systems where power is cycled more frequently than once every five minutes. The USDREQ bit should be set to indicate that power is going to be turned off. Software should poll the USDACK bit to determine when it is acceptable to power down.

Flash Control (FCTL)

Base 0x400F.D000
Offset 0x0F8
Type R/W, reset 0x0000.0000



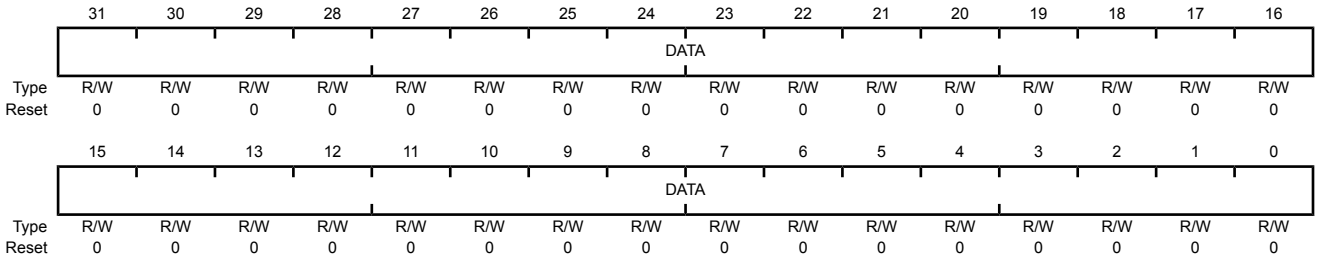
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	USDACK	RO	0	User Shut Down Acknowledge Value Description 1 The microcontroller can be powered down. 0 The microcontroller cannot yet be powered down. This bit should be set within 50 ms of setting the USDREQ bit.
0	USDREQ	R/W	0	User Shut Down Request Value Description 1 Requests permission to power down the microcontroller. 0 No effect.

Register 10: Flash Write Buffer n (FWBn), offset 0x100 - 0x17C

These 32 registers hold the contents of the data to be written into the Flash memory on a buffered Flash memory write operation. The offset selects one of the 32-bit registers. Only **FWBn** registers that have been updated since the preceding buffered Flash memory write operation are written into the Flash memory, so it is not necessary to write the entire bank of registers in order to write 1 or 2 words. The **FWBn** registers are written into the Flash memory with the **FWB0** register corresponding to the address contained in **FMA**. **FWB1** is written to the address **FMA**+0x4 etc. Note that only data bits that are 0 result in the Flash memory being modified. A data bit that is 1 leaves the content of the Flash memory bit at its previous value.

Flash Write Buffer n (FWBn)

Base 0x400F.D000
Offset 0x100 - 0x17C
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	DATA	R/W	0x0000.0000	Data Data to be written into the Flash memory.

6.5 Memory Register Descriptions (System Control Offset)

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

Register 11: ROM Control (RMCTL), offset 0x0F0

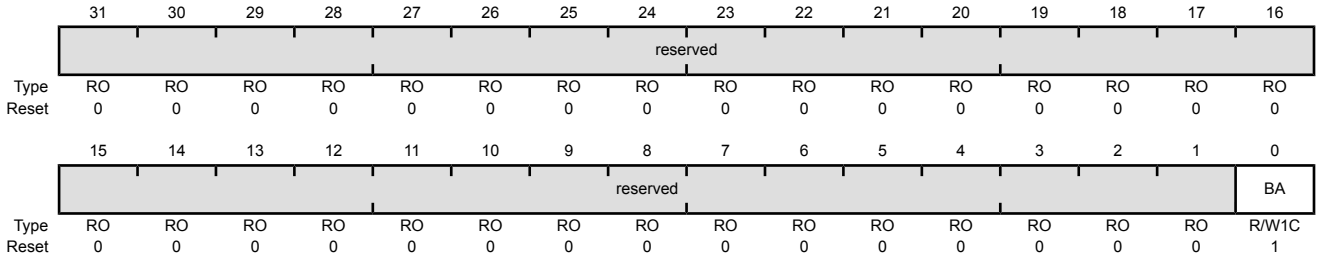
This register provides control of the ROM controller state. This register offset is relative to the System Control base address of 0x400F.E000.

At reset, the ROM is mapped over the Flash memory so that the ROM boot sequence is always executed. The boot sequence executed from ROM is as follows:

1. The BA bit (below) is cleared such that ROM is mapped to 0x01xx.xxxx and Flash memory is mapped to address 0x0.
2. The **BOOTCFG** register is read. If the EN bit is clear, the status of the specified GPIO pin is compared with the specified polarity. If the status matches the specified polarity, the ROM is mapped to address 0x0000.0000 and execution continues out of the ROM Boot Loader.
3. If the status doesn't match the specified polarity, the data at address 0x0000.0004 is read, and if the data at this address is 0xFFFF.FFFF, the ROM is mapped to address 0x0000.0000 and execution continues out of the ROM Boot Loader.
4. If there is data at address 0x0000.0004 that is not 0xFFFF.FFFF, the stack pointer (**SP**) is loaded from Flash memory at address 0x0000.0000 and the program counter (**PC**) is loaded from address 0x0000.0004. The user application begins executing.

ROM Control (RMCTL)

Base 0x400F.E000
Offset 0x0F0
Type R/W1C, reset -



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	BA	R/W1C	1	Boot Alias Value Description 1 The microcontroller's ROM appears at address 0x0. 0 The Flash memory is at address 0x0. This bit is cleared by writing a 1 to this bit position.

Register 12: Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x130 and 0x200

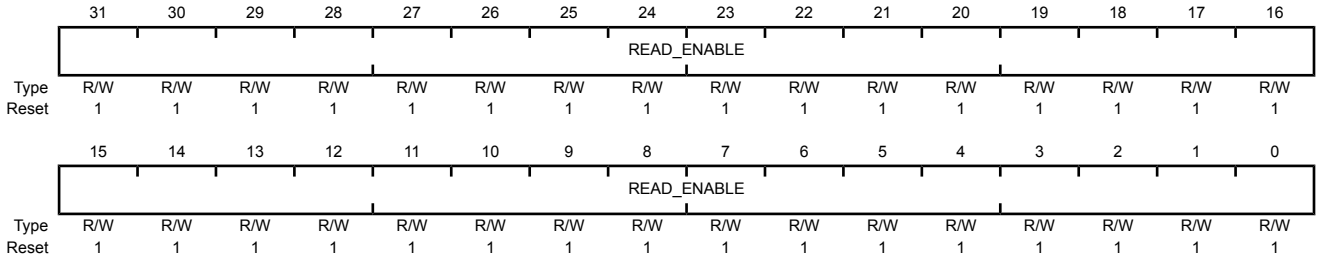
Note: This register is aliased for backwards compatability.

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

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 180. For additional information, see “Flash Memory Protection” on page 292.

Flash Memory Protection Read Enable 0 (FMPRE0)

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



Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0xFFFFFFFF	Flash Read Enable Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table "Flash Protection Policy Combinations". Value: 0xFFFFFFFF — Description: Bits [31:0] each enable protection on a 2-KB block of Flash memory up to the total of 64 KB.

Register 13: Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x134 and 0x400

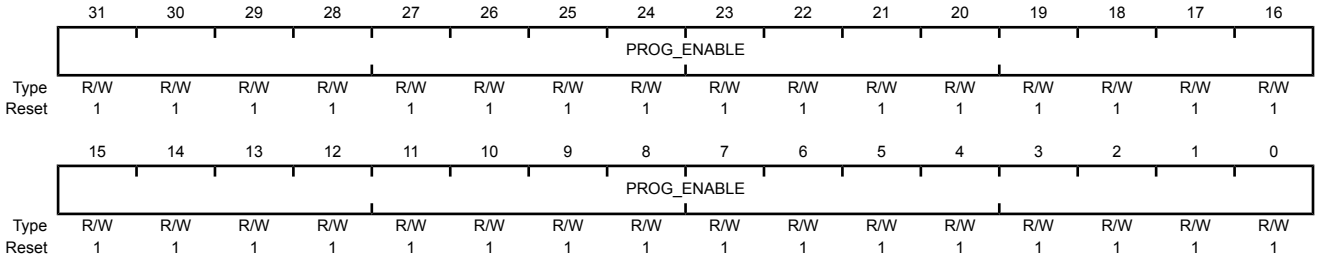
Note: This register is aliased for backwards compatability.

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

This register stores the execute-only protection bits for each 2-KB flash block (**FMPREn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 180. For additional information, see “Flash Memory Protection” on page 292.

Flash Memory Protection Program Enable 0 (FMPPE0)

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



Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0xFFFFFFFF	Flash Programming Enable Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table "Flash Protection Policy Combinations". Value Description 0xFFFFFFFF Bits [31:0] each enable protection on a 2-KB block of Flash memory up to the total of 64 KB.

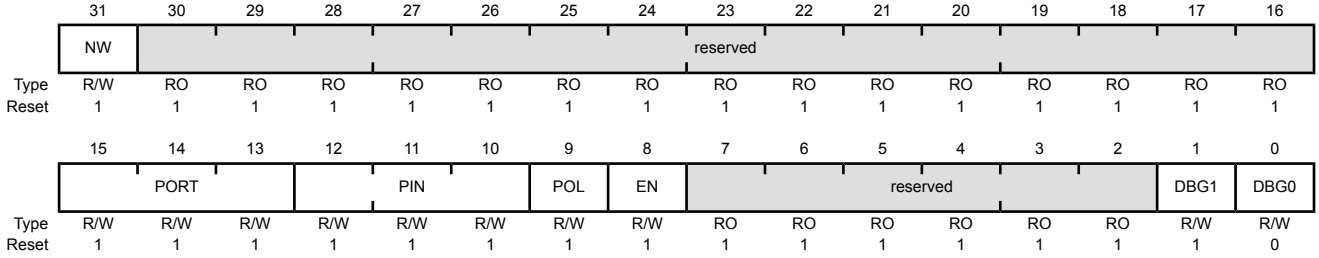
Register 14: Boot Configuration (BOOTCFG), offset 0x1D0

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

This register provides configuration of a GPIO pin to enable the ROM Boot Loader as well as a write-once mechanism to disable external debugger access to the device. Upon reset, the user has the opportunity to direct the core to execute the ROM Boot Loader or the application in Flash memory by using any GPIO signal from Ports A-H as configured by the bits in this register. If the EN bit is set or the specified pin does not have the required polarity, the system control module checks address 0x000.0004 to see if the Flash memory has a valid reset vector. If the data at address 0x0000.0004 is 0xFFFF.FFFF, then it is assumed that the Flash memory has not yet been programmed, and the core executes the ROM Boot Loader. The DBG0 bit (bit 0) is set to 0 from the factory and the DBG1 bit (bit 1) is set to 1, which enables external debuggers. Clearing the DBG1 bit disables any external debugger access to the device permanently, starting with the next power-up cycle of the device. The NW bit (bit 31) indicates that the register has not yet been committed and is controlled through hardware to ensure that the register is only committed once. Prior to being committed, bits can only be changed from 1 to 0. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 180.

Boot Configuration (BOOTCFG)

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



Bit/Field	Name	Type	Reset	Description
31	NW	R/W	1	Not Written When set, this bit indicates that this 32-bit register has not been committed. When clear, this bit specifies that this register has been committed and may not be committed again.
30:16	reserved	RO	0x7FFF	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
15:13	PORT	R/W	0x7	Boot GPIO Port This field selects the port of the GPIO port pin that enables the ROM boot loader at reset. Value Description 0x0 Port A 0x1 Port B 0x2 Port C 0x3 Port D 0x4 Port E 0x5 Port F 0x6 Port G 0x7 Port H
12:10	PIN	R/W	0x7	Boot GPIO Pin This field selects the pin number of the GPIO port pin that enables the ROM boot loader at reset. Value Description 0x0 Pin 0 0x1 Pin 1 0x2 Pin 2 0x3 Pin 3 0x4 Pin 4 0x5 Pin 5 0x6 Pin 6 0x7 Pin 7
9	POL	R/W	0x1	Boot GPIO Polarity When set, this bit selects a high level for the GPIO port pin to enable the ROM boot loader at reset. When clear, this bit selects a low level for the GPIO port pin.
8	EN	R/W	0x1	Boot GPIO Enable Clearing this bit enables the use of a GPIO pin to enable the ROM Boot Loader at reset. When this bit is set, the contents of address 0x0000.0004 are checked to see if the Flash memory has been programmed. If the contents are not 0xFFFF.FFFF, the core executes out of Flash memory. If the Flash has not been programmed, the core executes out of ROM.
7:2	reserved	RO	0x3F	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	DBG1	R/W	1	Debug Control 1 The DBG1 bit must be 1 and DBG0 must be 0 for debug to be available.
0	DBG0	R/W	0x0	Debug Control 0 The DBG1 bit must be 1 and DBG0 must be 0 for debug to be available.

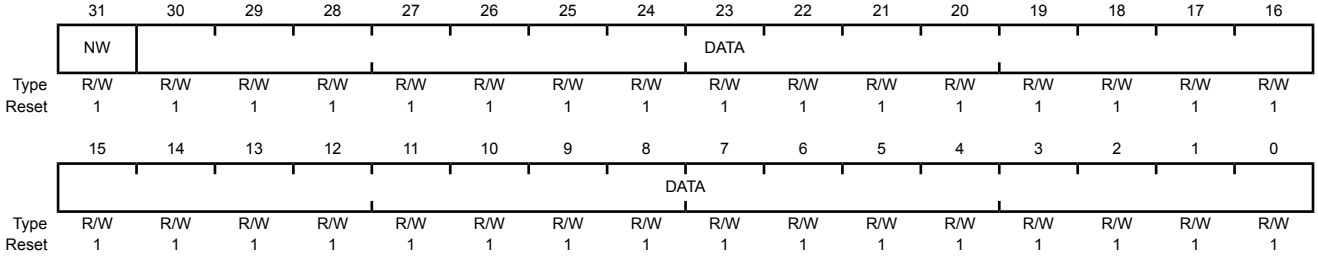
Register 15: User Register 0 (USER_REG0), offset 0x1E0

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

This register provides 31 bits of user-defined data that is non-volatile and can only be committed once. Bit 31 indicates that the register is available to be committed and is controlled through hardware to ensure that the register is only committed once. Prior to being committed, bits can only be changed from 1 to 0. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. The write-once characteristics of this register are useful for keeping static information like communication addresses that need to be unique per part and would otherwise require an external EEPROM or other non-volatile device. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 180.

User Register 0 (USER_REG0)

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



Bit/Field	Name	Type	Reset	Description
31	NW	R/W	1	Not Written When set, this bit indicates that this 32-bit register has not been committed. When clear, this bit specifies that this register has been committed and may not be committed again.
30:0	DATA	R/W	0x7FFFFFFF	User Data Contains the user data value. This field is initialized to all 1s and can only be committed once.

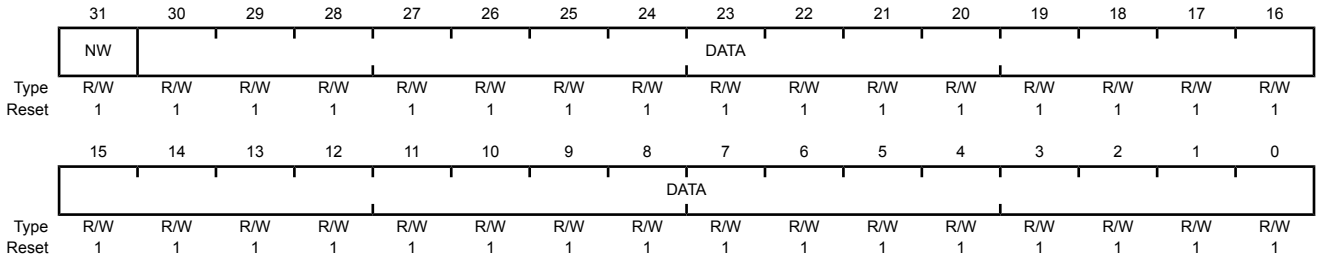
Register 16: User Register 1 (USER_REG1), offset 0x1E4

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

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

User Register 1 (USER_REG1)

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



Bit/Field	Name	Type	Reset	Description
31	NW	R/W	1	Not Written When set, this bit indicates that this 32-bit register has not been committed. When clear, this bit specifies that this register has been committed and may not be committed again.
30:0	DATA	R/W	0x7FFFFFFF	User Data Contains the user data value. This field is initialized to all 1s and can only be committed once.

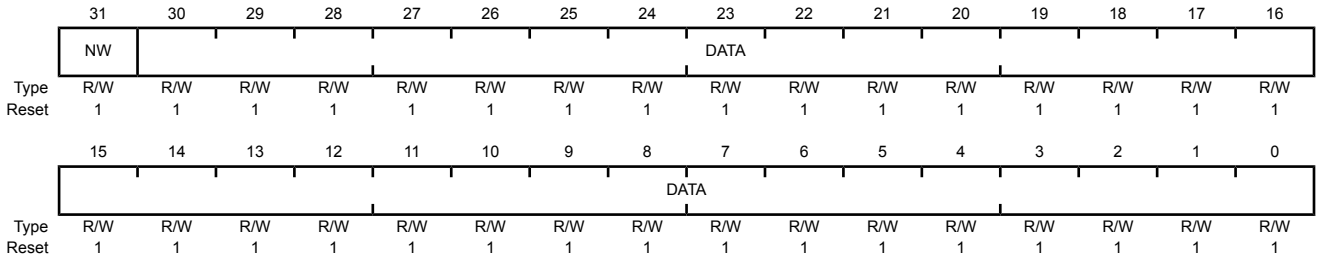
Register 17: User Register 2 (USER_REG2), offset 0x1E8

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

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

User Register 2 (USER_REG2)

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



Bit/Field	Name	Type	Reset	Description
31	NW	R/W	1	Not Written When set, this bit indicates that this 32-bit register has not been committed. When clear, this bit specifies that this register has been committed and may not be committed again.
30:0	DATA	R/W	0x7FFFFFFF	User Data Contains the user data value. This field is initialized to all 1s and can only be committed once.

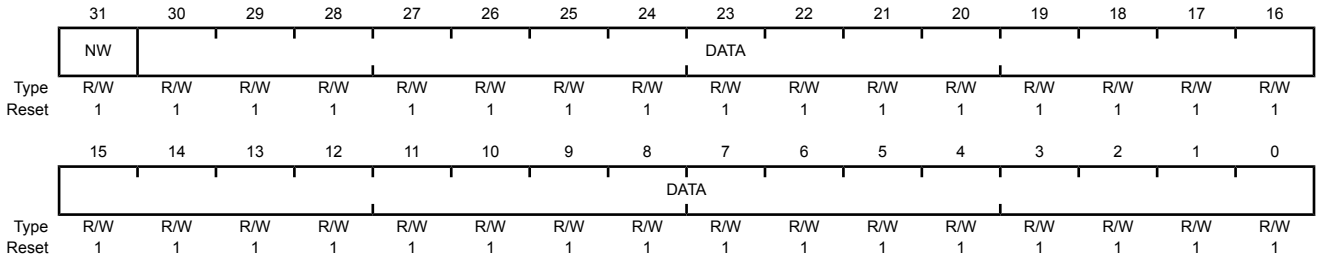
Register 18: User Register 3 (USER_REG3), offset 0x1EC

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

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

User Register 3 (USER_REG3)

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



Bit/Field	Name	Type	Reset	Description
31	NW	R/W	1	Not Written When set, this bit indicates that this 32-bit register has not been committed. When clear, this bit specifies that this register has been committed and may not be committed again.
30:0	DATA	R/W	0x7FFFFFFF	User Data Contains the user data value. This field is initialized to all 1s and can only be committed once.

Register 19: Flash Memory Protection Read Enable 1 (FMPRE1), offset 0x204

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

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 180. If the Flash memory size on the device is less than 64 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see “Flash Memory Protection” on page 292.

Flash Memory Protection Read Enable 1 (FMPRE1)

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

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

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

Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0xFFFFFFFF	Flash Read Enable Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table "Flash Protection Policy Combinations". Value: 0xFFFFFFFF — Description: Bits [31:0] each enable protection on a 2-KB block of Flash memory in memory range from 65 to 128 KB.

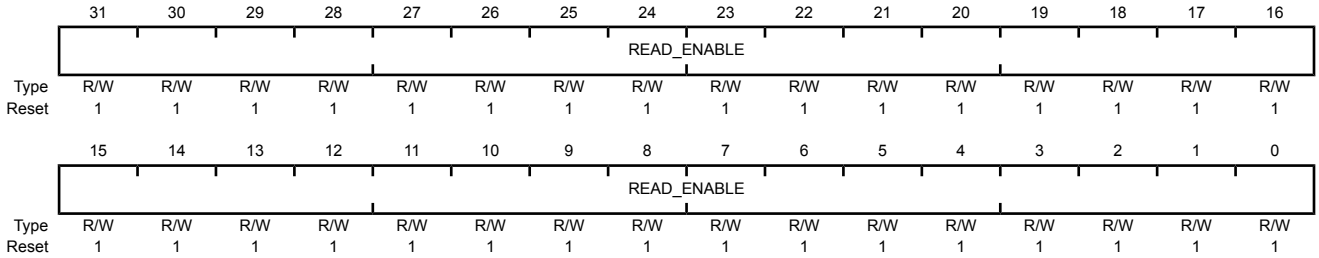
Register 20: Flash Memory Protection Read Enable 2 (FMPRE2), offset 0x208

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

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 180. If the Flash memory size on the device is less than 128 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see “Flash Memory Protection” on page 292.

Flash Memory Protection Read Enable 2 (FMPRE2)

Base 0x400F.E000
Offset 0x208
Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0xFFFFFFFF	Flash Read Enable Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table "Flash Protection Policy Combinations". Value: 0xFFFFFFFF — Description: Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 129 to 192 KB.

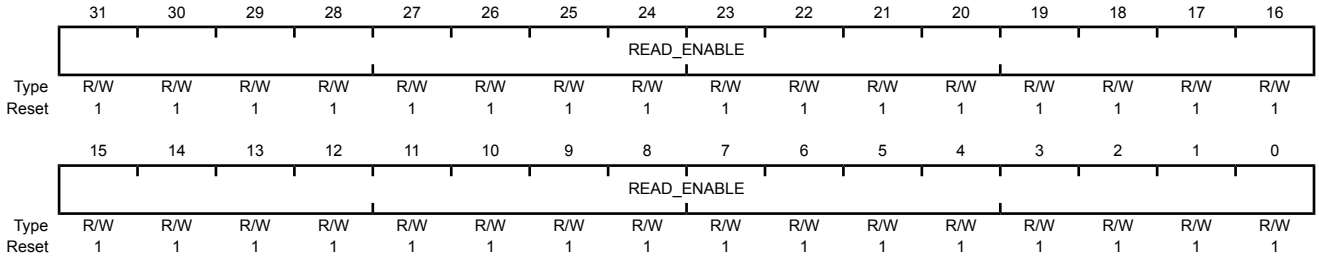
Register 21: Flash Memory Protection Read Enable 3 (FMPRE3), offset 0x20C

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

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 180. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see “Flash Memory Protection” on page 292.

Flash Memory Protection Read Enable 3 (FMPRE3)

Base 0x400F.E000
Offset 0x20C
Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0xFFFFFFFF	Flash Read Enable Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table "Flash Protection Policy Combinations". Value: 0xFFFFFFFF Description: Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 193 to 256 KB.

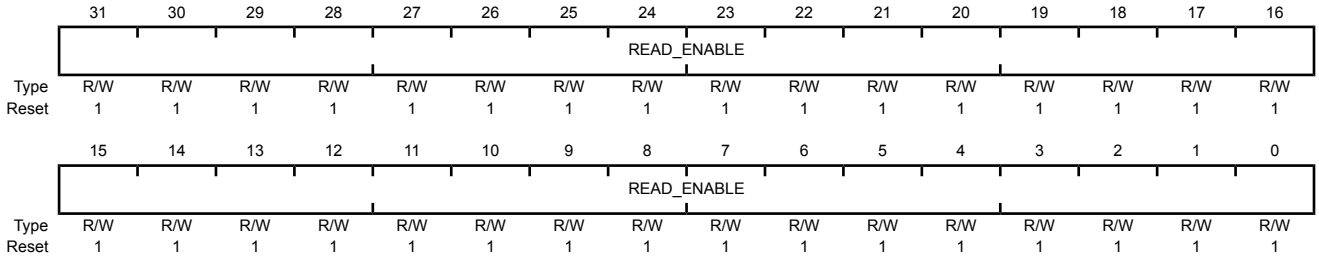
Register 22: Flash Memory Protection Read Enable 4 (FMPRE4), offset 0x210

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

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 180. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see “Flash Memory Protection” on page 292.

Flash Memory Protection Read Enable 4 (FMPRE4)

Base 0x400F.E000
Offset 0x210
Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0xFFFFFFFF	Flash Read Enable Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table "Flash Protection Policy Combinations". Value: 0xFFFFFFFF Description: Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 257 to 320 KB.

Register 23: Flash Memory Protection Read Enable 5 (FMPRE5), offset 0x214

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

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 180. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see “Flash Memory Protection” on page 292.

Flash Memory Protection Read Enable 5 (FMPRE5)

Base 0x400F.E000
Offset 0x214
Type R/W, reset 0xFFFF.FFFF

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

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

Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0xFFFFFFFF	Flash Read Enable Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table "Flash Protection Policy Combinations". Value: 0xFFFFFFFF — Description: Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 321 to 384 KB.

Register 24: Flash Memory Protection Read Enable 6 (FMPRE6), offset 0x218

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

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 180. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see “Flash Memory Protection” on page 292.

Flash Memory Protection Read Enable 6 (FMPRE6)

Base 0x400F.E000
Offset 0x218
Type R/W, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0x00000000	Flash Read Enable Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table "Flash Protection Policy Combinations". Value Description 0x00000000 Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 385 to 448 KB.

Register 25: Flash Memory Protection Read Enable 7 (FMPRE7), offset 0x21C

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

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 180. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see “Flash Memory Protection” on page 292.

Flash Memory Protection Read Enable 7 (FMPRE7)

Base 0x400F.E000
Offset 0x21C
Type R/W, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0x00000000	Flash Read Enable Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table "Flash Protection Policy Combinations". Value Description 0x00000000 Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 449 to 512 KB.

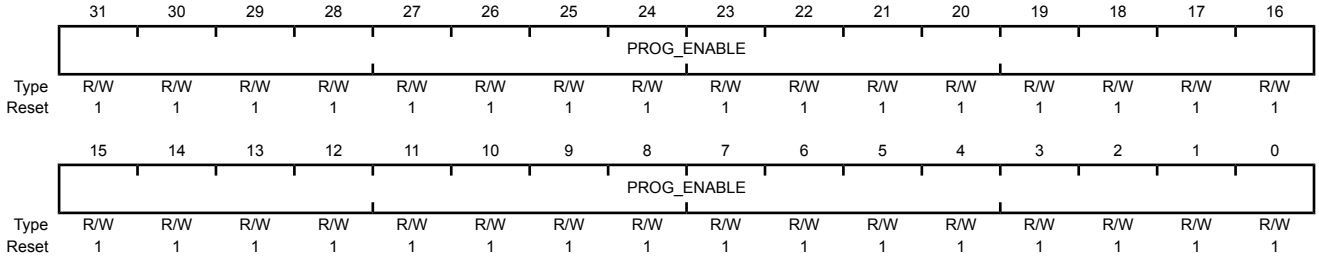
Register 26: Flash Memory Protection Program Enable 1 (FMPPE1), offset 0x404

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

This register stores the execute-only protection bits for each 2-KB flash block (**FMPREn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 180. If the Flash memory size on the device is less than 64 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see “Flash Memory Protection” on page 292.

Flash Memory Protection Program Enable 1 (FMPPE1)

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



Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0xFFFFFFFF	Flash Programming Enable Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table "Flash Protection Policy Combinations". Value Description 0xFFFFFFFF Bits [31:0] each enable protection on a 2-KB block of Flash memory in memory range from 65 to 128 KB.

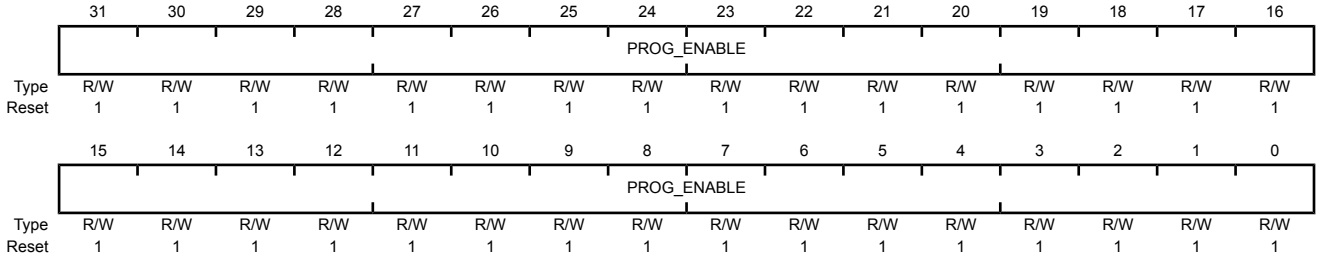
Register 27: Flash Memory Protection Program Enable 2 (FMPPE2), offset 0x408

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

This register stores the execute-only protection bits for each 2-KB flash block (**FMPREn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 180. If the Flash memory size on the device is less than 128 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see “Flash Memory Protection” on page 292.

Flash Memory Protection Program Enable 2 (FMPPE2)

Base 0x400F.E000
Offset 0x408
Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0xFFFFFFFF	Flash Programming Enable Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table "Flash Protection Policy Combinations". Value Description 0xFFFFFFFF Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 129 to 192 KB.

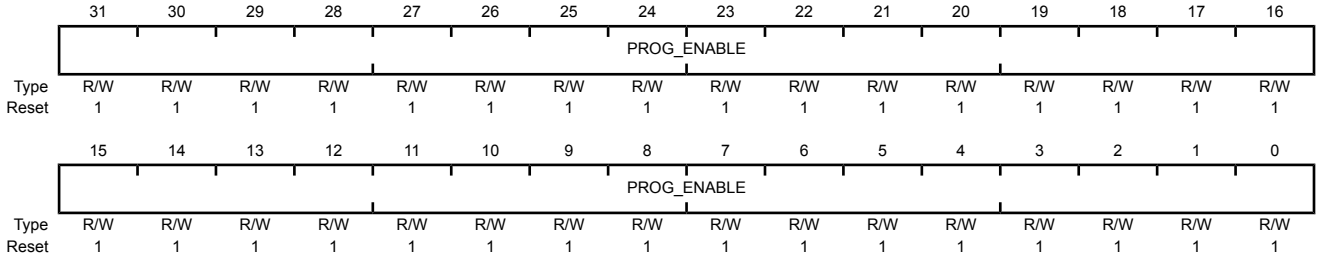
Register 28: Flash Memory Protection Program Enable 3 (FMPPE3), offset 0x40C

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

This register stores the execute-only protection bits for each 2-KB flash block (**FMPREn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 180. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see “Flash Memory Protection” on page 292.

Flash Memory Protection Program Enable 3 (FMPPE3)

Base 0x400F.E000
Offset 0x40C
Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0xFFFFFFFF	Flash Programming Enable Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table "Flash Protection Policy Combinations". Value Description 0xFFFFFFFF Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 193 to 256 KB.

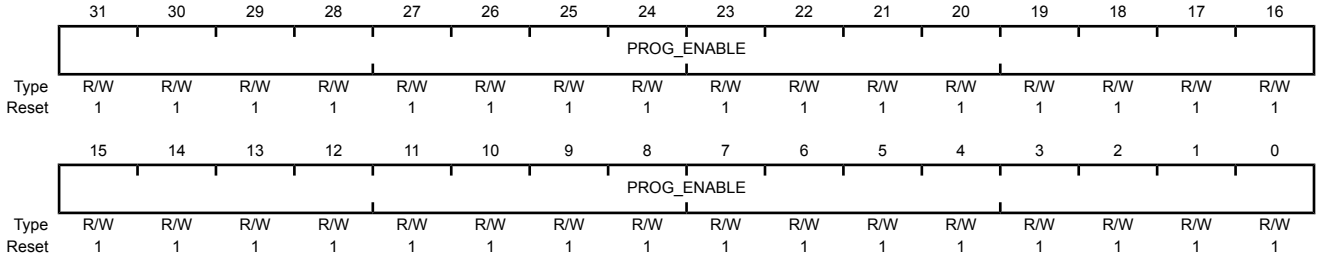
Register 29: Flash Memory Protection Program Enable 4 (FMPPE4), offset 0x410

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

This register stores the execute-only protection bits for each 2-KB flash block (**FMPREn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 180. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see “Flash Memory Protection” on page 292.

Flash Memory Protection Program Enable 4 (FMPPE4)

Base 0x400F.E000
Offset 0x410
Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0xFFFFFFFF	Flash Programming Enable Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table "Flash Protection Policy Combinations". Value Description 0xFFFFFFFF Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 257 to 320 KB.

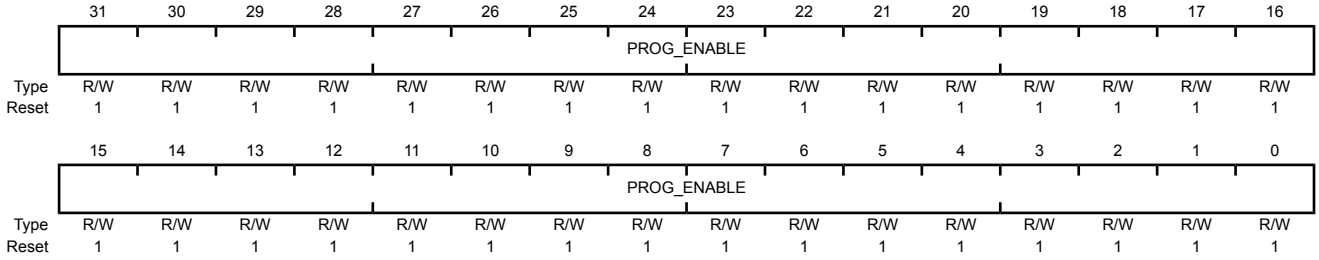
Register 30: Flash Memory Protection Program Enable 5 (FMPPE5), offset 0x414

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

This register stores the execute-only protection bits for each 2-KB flash block (**FMPREn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 180. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see “Flash Memory Protection” on page 292.

Flash Memory Protection Program Enable 5 (FMPPE5)

Base 0x400F.E000
Offset 0x414
Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0xFFFFFFFF	Flash Programming Enable Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table "Flash Protection Policy Combinations". Value Description 0xFFFFFFFF Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 321 to 384 KB.

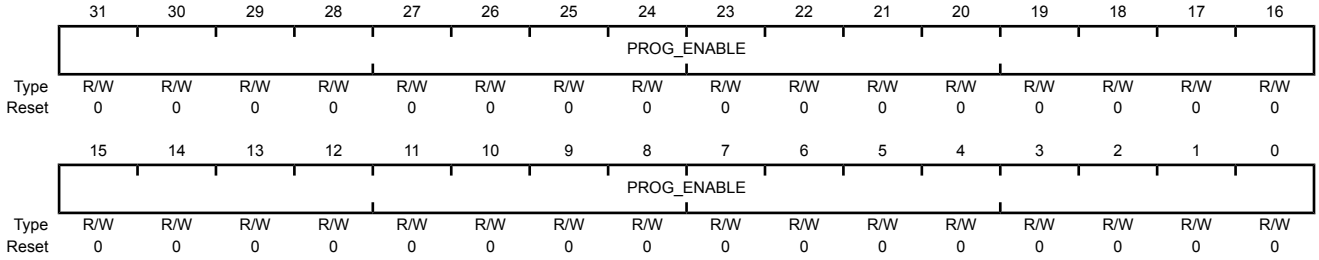
Register 31: Flash Memory Protection Program Enable 6 (FMPPE6), offset 0x418

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

This register stores the execute-only protection bits for each 2-KB flash block (**FMPREn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 180. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see “Flash Memory Protection” on page 292.

Flash Memory Protection Program Enable 6 (FMPPE6)

Base 0x400F.E000
Offset 0x418
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0x00000000	Flash Programming Enable Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table "Flash Protection Policy Combinations". Value Description 0x00000000 Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 385 to 448 KB.

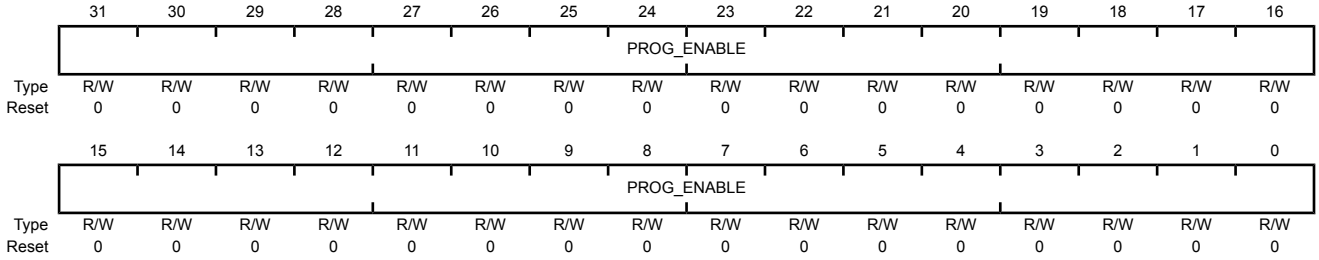
Register 32: Flash Memory Protection Program Enable 7 (FMPPE7), offset 0x41C

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

This register stores the execute-only protection bits for each 2-KB flash block (**FMPREn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 180. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see “Flash Memory Protection” on page 292.

Flash Memory Protection Program Enable 7 (FMPPE7)

Base 0x400F.E000
Offset 0x41C
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0x00000000	Flash Programming Enable Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table "Flash Protection Policy Combinations". Value Description 0x00000000 Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 449 to 512 KB.

7 Micro Direct Memory Access (µDMA)

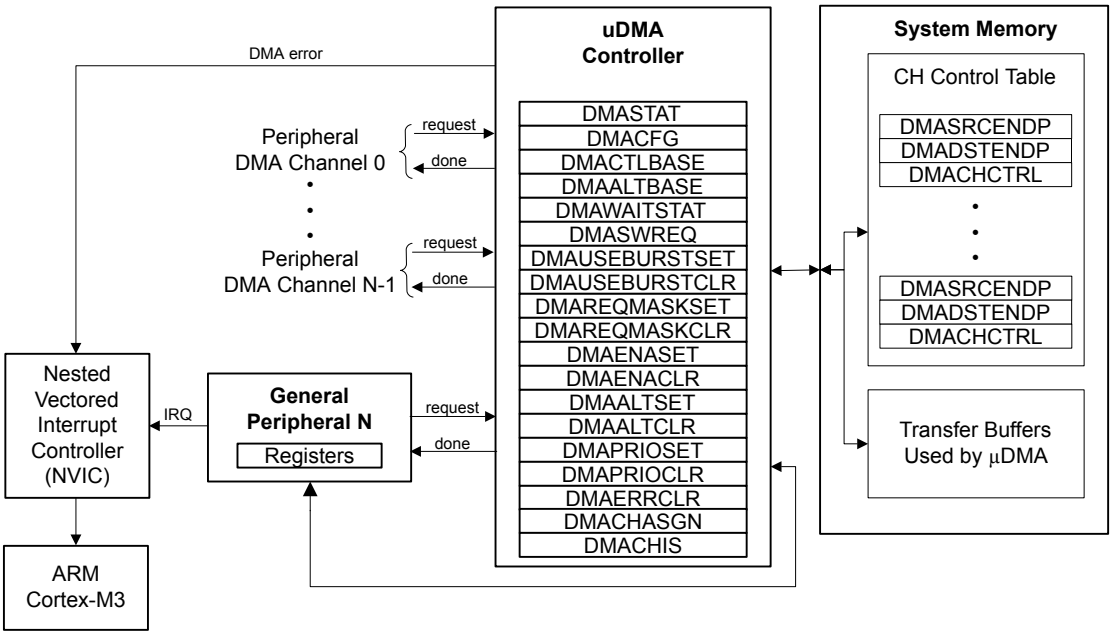
The LM3S9U81 microcontroller includes a Direct Memory Access (DMA) controller, known as micro-DMA (µDMA). The µDMA controller provides a way to offload data transfer tasks from the Cortex™-M3 processor, allowing for more efficient use of the processor and the available bus bandwidth. The µDMA controller can perform transfers between memory and peripherals. It has dedicated channels for each supported on-chip module and can be programmed to automatically perform transfers between peripherals and memory as the peripheral is ready to transfer more data. The µDMA controller provides the following features:

- ARM® PrimeCell® 32-channel configurable µDMA controller
- Support for memory-to-memory, memory-to-peripheral, and peripheral-to-memory in multiple transfer modes
  - Basic for simple transfer scenarios
  - Ping-pong for continuous data flow
  - Scatter-gather for a programmable list of arbitrary transfers initiated from a single request
- Highly flexible and configurable channel operation
  - Independently configured and operated channels
  - Dedicated channels for supported on-chip modules
  - Primary and secondary channel assignments
  - One channel each for receive and transmit path for bidirectional modules
  - Dedicated channel for software-initiated transfers
  - Per-channel configurable priority scheme
  - Optional software-initiated requests for any channel
- Two levels of priority
- Design optimizations for improved bus access performance between µDMA controller and the processor core
  - µDMA controller access is subordinate to core access
  - RAM striping
  - Peripheral bus segmentation
- Data sizes of 8, 16, and 32 bits
- Transfer size is programmable in binary steps from 1 to 1024
- Source and destination address increment size of byte, half-word, word, or no increment
- Maskable peripheral requests

- Interrupt on transfer completion, with a separate interrupt per channel

7.1 Block Diagram

Figure 7-1. µDMA Block Diagram



7.2 Functional Description

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

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

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

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

7.2.1 Channel Assignments

µDMA channels 0-31 are assigned to peripherals according to the following table. The **DMA Channel Assignment (DMACHASGN)** register (see page 384) can be used to specify the primary or secondary assignment. If the primary function is not available on this microcontroller, the secondary function becomes the primary function. If the secondary function is not available, the primary function is the only option.

Note: Channels noted in the table as "Available for software" may be assigned to peripherals in the future. However, they are currently available for software use. Channel 30 is dedicated for software use.

The USB endpoints mapped to µDMA channels 0-3 can be changed with the **USBDMASEL** register (see page 1106).

Because of the way the µDMA controller interacts with peripherals, the µDMA channel for the peripheral must be enabled in order for the µDMA controller to be able to read and write the peripheral registers, even if a different µDMA channel is used to perform the µDMA transfer. To minimize confusion and chance of software errors, it is best practice to use a peripheral's µDMA channel for performing all µDMA transfers for that peripheral, even if it is processor-triggered and using AUTO mode, which could be considered a software transfer. Note that if the software channel is used, interrupts occur on the dedicated µDMA interrupt vector. If the peripheral channel is used, then the interrupt occurs on the interrupt vector for the peripheral.

Table 7-1. µDMA Channel Assignments

µDMA Channel	Primary Assignment	Secondary Assignment
0	USB Endpoint 1 Receive	UART2 Receive
1	USB Endpoint 1 Transmit	UART2 Transmit
2	USB Endpoint 2 Receive	General-Purpose Timer 3A
3	USB Endpoint 2 Transmit	General-Purpose Timer 3B
4	USB Endpoint 3 Receive	General-Purpose Timer 2A
5	USB Endpoint 3 Transmit	General-Purpose Timer 2B
6	Ethernet Receive	General-Purpose Timer 2A
7	Ethernet Transmit	General-Purpose Timer 2B
8	UART0 Receive	UART1 Receive
9	UART0 Transmit	UART1 Transmit
10	SSI0 Receive	SSI1 Receive
11	SSI0 Transmit	SSI1 Transmit
12	Available for software	UART2 Receive
13	Available for software	UART2 Transmit
14	ADC0 Sample Sequencer 0	General-Purpose Timer 2A
15	ADC0 Sample Sequencer 1	General-Purpose Timer 2B
16	ADC0 Sample Sequencer 2	Available for software
17	ADC0 Sample Sequencer 3	Available for software

Table 7-1. µDMA Channel Assignments *(continued)*

µDMA Channel	Primary Assignment	Secondary Assignment
18	General-Purpose Timer 0A	General-Purpose Timer 1A
19	General-Purpose Timer 0B	General-Purpose Timer 1B
20	General-Purpose Timer 1A	EPI0 NBRFIFO
21	General-Purpose Timer 1B	EPI0 WFIFO
22	UART1 Receive	Available for software
23	UART1 Transmit	Available for software
24	SSI1 Receive	ADC1 Sample Sequencer 0
25	SSI1 Transmit	ADC1 Sample Sequencer 1
26	Available for software	ADC1 Sample Sequencer 2
27	Available for software	ADC1 Sample Sequencer 3
28	I^2S0 Receive	Available for software
29	I^2S0 Transmit	Available for software
30	Dedicated for software use	
31	Reserved	

7.2.2 Priority

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

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

7.2.3 Arbitration Size

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

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

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

7.2.4 Request Types

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

The µDMA controller responds differently depending on whether the peripheral is making a single request or a burst request. If both are asserted, and the µDMA channel has been set up for a burst transfer, then the burst request takes precedence. See Table 7-2 on page 338, which shows how each peripheral supports the two request types.

Table 7-2. Request Type Support

Peripheral	Single Request Signal	Burst Request Signal
ADC	None	Sequencer IE bit
EPI WFIFO	None	WFIFO Level (configurable)
EPI NBRFIFO	None	NBRFIFO Level (configurable)
Ethernet TX	TX FIFO empty	None
Ethernet RX	RX packet received	None
General-Purpose Timer	Raw interrupt pulse	None
I^2S TX	None	FIFO service request
I^2S RX	None	FIFO service request
SSI TX	TX FIFO Not Full	TX FIFO Level (fixed at 4)
SSI RX	RX FIFO Not Empty	RX FIFO Level (fixed at 4)
UART TX	TX FIFO Not Full	TX FIFO Level (configurable)
UART RX	RX FIFO Not Empty	RX FIFO Level (configurable)
USB TX	None	FIFO TXRDY
USB RX	None	FIFO RXRDY

7.2.4.1 Single Request

When a single request is detected, and not a burst request, the µDMA controller transfers one item and then stops to wait for another request.

7.2.4.2 Burst Request

When a burst request is detected, the µDMA controller transfers the number of items that is the lesser of the arbitration size or the number of items remaining in the transfer. Therefore, the arbitration size should be the same as the number of data items that the peripheral can accommodate when making a burst request. For example, the UART generates a burst request based on the FIFO trigger level. In this case, the arbitration size should be set to the amount of data that the FIFO can transfer when the trigger level is reached. A burst transfer runs to completion once it is started, and cannot be interrupted, even by a higher priority channel. Burst transfers complete in a shorter time than the same number of non-burst transfers.

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

7.2.5 Channel Configuration

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

Table 7-3 on page 339 shows the layout in memory of the channel control table. Each channel may have one or two control structures in the control table: a primary control structure and an optional alternate control structure. The table is organized so that all of the primary entries are in the first half of the table, and all the alternate structures are in the second half of the table. The primary entry is used for simple transfer modes where transfers can be reconfigured and restarted after each transfer is complete. In this case, the alternate control structures are not used and therefore only the first half of the table must be allocated in memory; the second half of the control table is not necessary, and that memory can be used for something else. If a more complex transfer mode is used such as ping-pong or scatter-gather, then the alternate control structure is also used and memory space should be allocated for the entire table.

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

Table 7-3. Control Structure Memory Map

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

Table 7-4 shows an individual control structure entry in the control table. Each entry is aligned on a 16-byte boundary. The entry contains four long words: the source end pointer, the destination end pointer, the control word, and an unused entry. The end pointers point to the ending address of the transfer and are inclusive. If the source or destination is non-incrementing (as for a peripheral register), then the pointer should point to the transfer address.

Table 7-4. Channel Control Structure

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

The control word contains the following fields:

- Source and destination data sizes

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

The control word and each field are described in detail in “µDMA Channel Control Structure” on page 358. The µDMA controller updates the transfer size and transfer mode fields as the transfer is performed. At the end of a transfer, the transfer size indicates 0, and the transfer mode indicates "stopped." Because the control word is modified by the µDMA controller, it must be reconfigured before each new transfer. The source and destination end pointers are not modified, so they can be left unchanged if the source or destination addresses remain the same.

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

7.2.6 Transfer Modes

The µDMA controller supports several transfer modes. Two of the modes support simple one-time transfers. Several complex modes support a continuous flow of data.

7.2.6.1 Stop Mode

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

7.2.6.2 Basic Mode

In Basic mode, the µDMA controller performs transfers as long as there are more items to transfer, and a transfer request is present. This mode is used with peripherals that assert a µDMA request signal whenever the peripheral is ready for a data transfer. Basic mode should not be used in any situation where the request is momentary even though the entire transfer should be completed. For example, a software-initiated transfer creates a momentary request, and in Basic mode, only the number of transfers specified by the ARBSIZE field in the **DMA Channel Control Word (DMACHCTL)** register is transferred on a software request, even if there is more data to transfer.

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

7.2.6.3 Auto Mode

Auto mode is similar to Basic mode, except that once a transfer request is received, the transfer runs to completion, even if the µDMA request is removed. This mode is suitable for software-triggered transfers. Generally, Auto mode is not used with a peripheral.

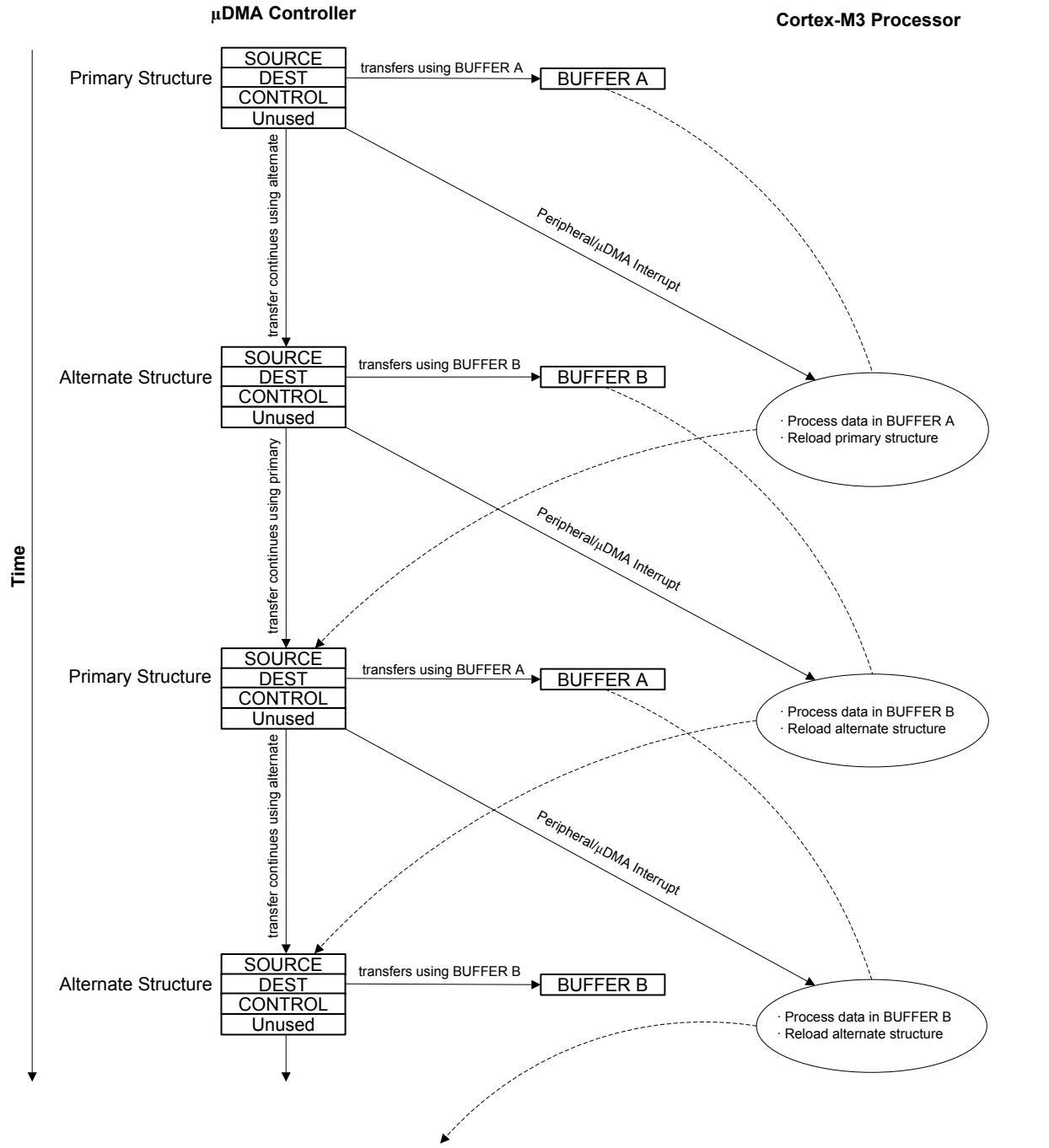
When all the items have been transferred using Auto mode, the µDMA controller sets the mode for that channel to Stop.

7.2.6.4 Ping-Pong

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

Refer to Figure 7-2 on page 342 for an example showing operation in Ping-Pong mode.

Figure 7-2. Example of Ping-Pong µDMA Transaction



7.2.6.5 Memory Scatter-Gather

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

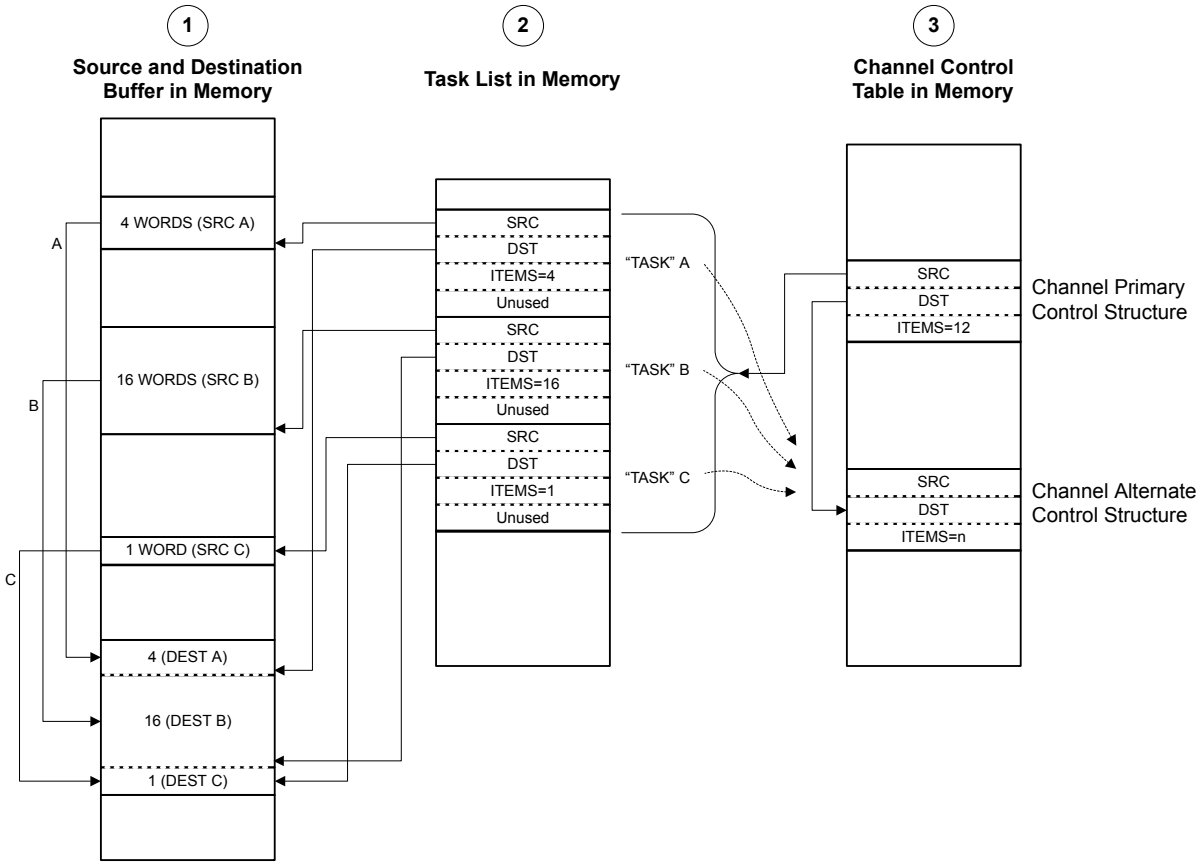
In Memory Scatter-Gather mode, the primary control structure is used to program the alternate control structure from a table in memory. The table is set up by the processor software and contains a list of control structures, each containing the source and destination end pointers, and the control word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

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

Refer to Figure 7-3 on page 344 and Figure 7-4 on page 345, which show an example of operation in Memory Scatter-Gather mode. This example shows a *gather* operation, where data in three separate buffers in memory is copied together into one buffer. Figure 7-3 on page 344 shows how the application sets up a µDMA task list in memory that is used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel that is used for the operation is configured to copy from the task list to the alternate control structure.

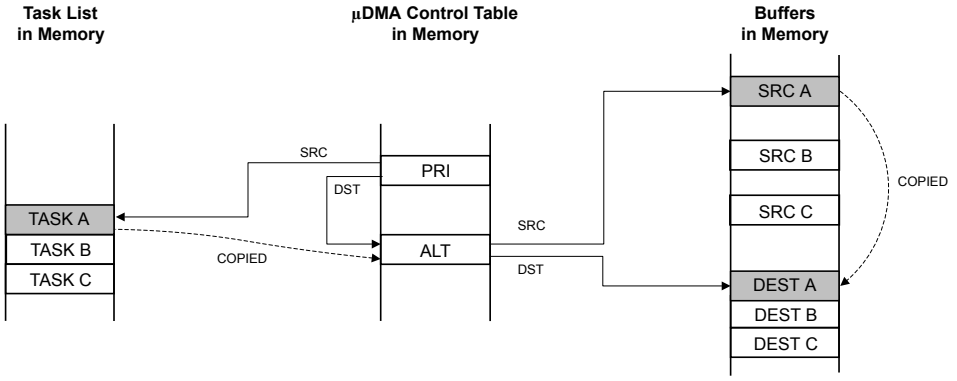
Figure 7-4 on page 345 shows the sequence as the µDMA controller performs the three sets of copy operations. First, using the primary control structure, the µDMA controller loads the alternate control structure with task A. It then performs the copy operation specified by task A, copying the data from the source buffer A to the destination buffer. Next, the µDMA controller again uses the primary control structure to load task B into the alternate control structure, and then performs the B operation with the alternate control structure. The process is repeated for task C.

Figure 7-3. Memory Scatter-Gather, Setup and Configuration



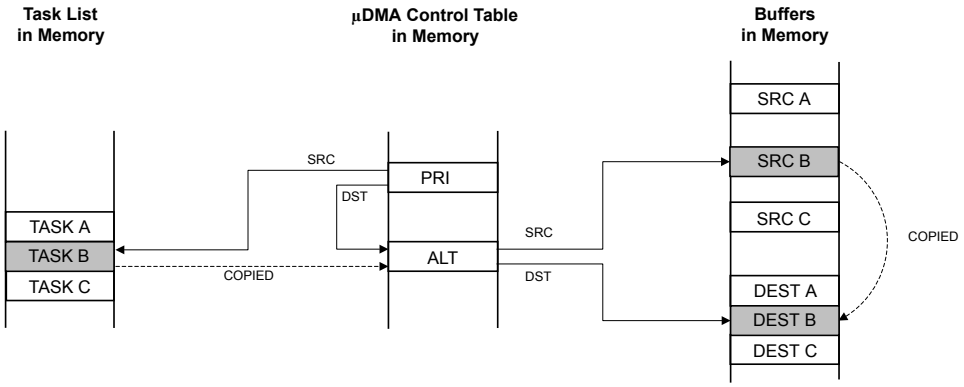
NOTES:
1. Application has a need to copy data items from three separate locations in memory into one combined buffer.
2. Application sets up µDMA "task list" in memory, which contains the pointers and control configuration for three µDMA copy "tasks."
3. Application sets up the channel primary control structure to copy each task configuration, one at a time, to the alternate control structure, where it is executed by the µDMA controller.
4. The SRC and DST pointers in the task list must point to the last location in the corresponding buffer.

Figure 7-4. Memory Scatter-Gather, µDMA Copy Sequence



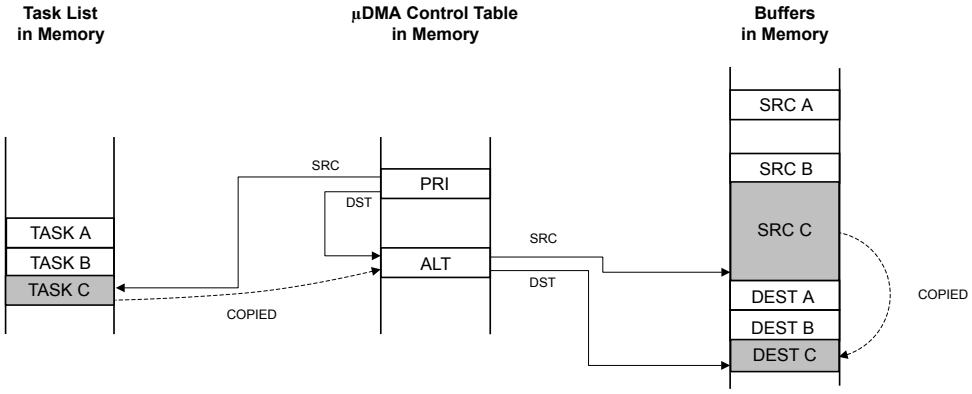
Using the channel's primary control structure, the µDMA controller copies task A configuration to the channel's alternate control structure.

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



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

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



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

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

7.2.6.6 Peripheral Scatter-Gather

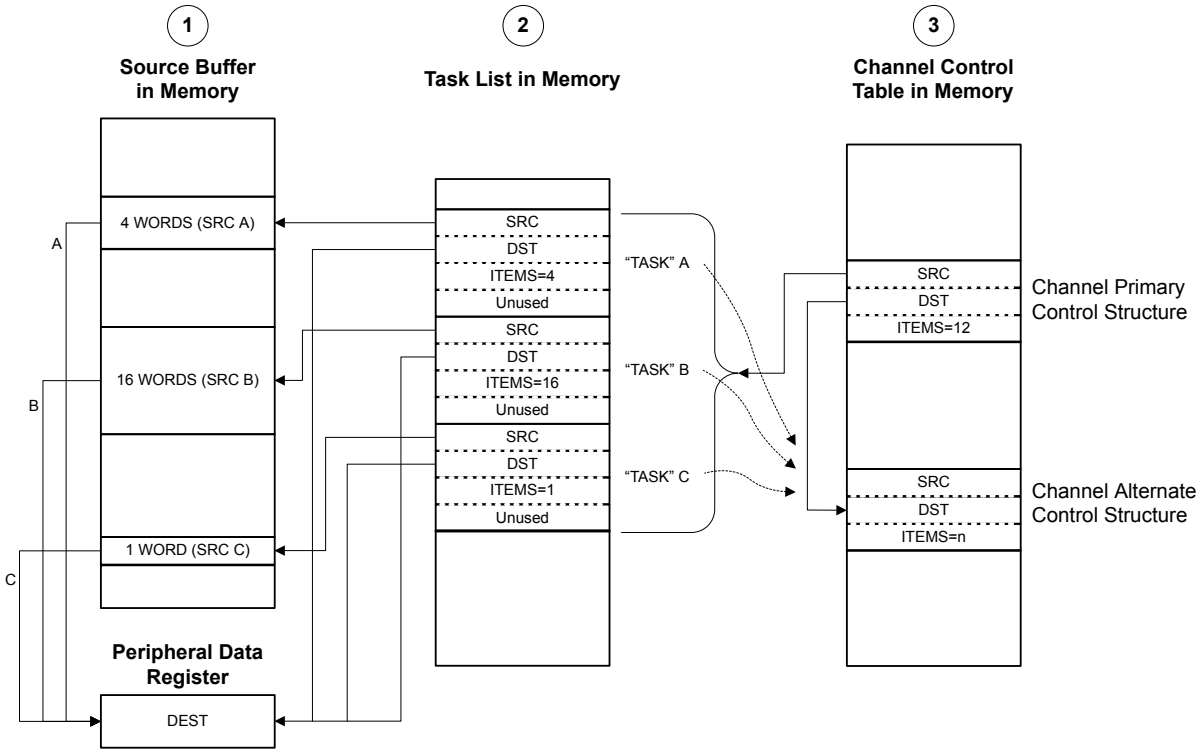
Peripheral Scatter-Gather mode is very similar to Memory Scatter-Gather, except that the transfers are controlled by a peripheral making a µDMA request. Upon detecting a request from the peripheral, the µDMA controller uses the primary control structure to copy one entry from the list to the alternate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

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

Refer to Figure 7-5 on page 347 and Figure 7-6 on page 348, which show an example of operation in Peripheral Scatter-Gather mode. This example shows a gather operation, where data from three separate buffers in memory is copied to a single peripheral data register. Figure 7-5 on page 347 shows how the application sets up a µDMA task list in memory that is used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel that is used for the operation is configured to copy from the task list to the alternate control structure.

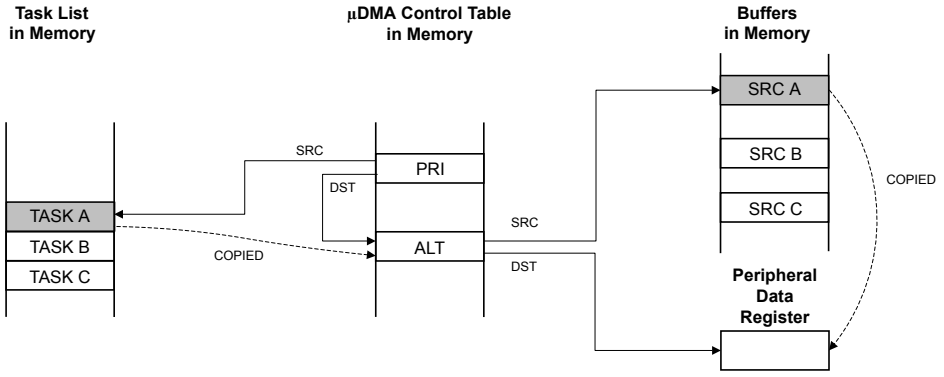
Figure 7-6 on page 348 shows the sequence as the µDMA controller performs the three sets of copy operations. First, using the primary control structure, the µDMA controller loads the alternate control structure with task A. It then performs the copy operation specified by task A, copying the data from the source buffer A to the peripheral data register. Next, the µDMA controller again uses the primary control structure to load task B into the alternate control structure, and then performs the B operation with the alternate control structure. The process is repeated for task C.

Figure 7-5. Peripheral Scatter-Gather, Setup and Configuration



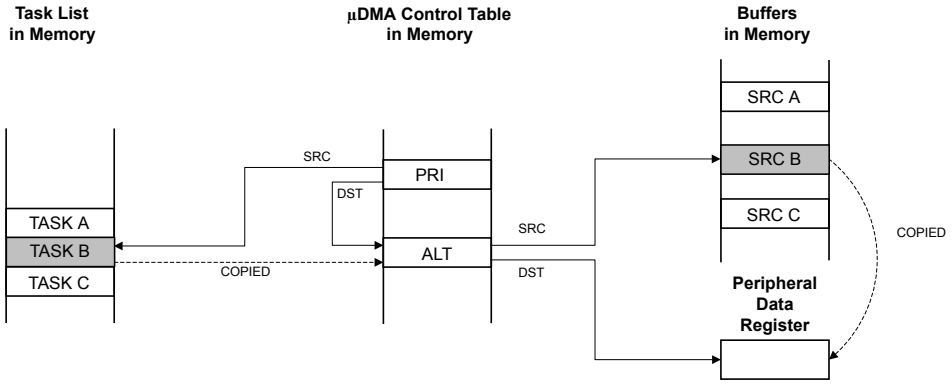
NOTES:
1. Application has a need to copy data items from three separate locations in memory into a peripheral data register.
2. Application sets up µDMA "task list" in memory, which contains the pointers and control configuration for three µDMA copy "tasks."
3. Application sets up the channel primary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

Figure 7-6. Peripheral Scatter-Gather, µDMA Copy Sequence



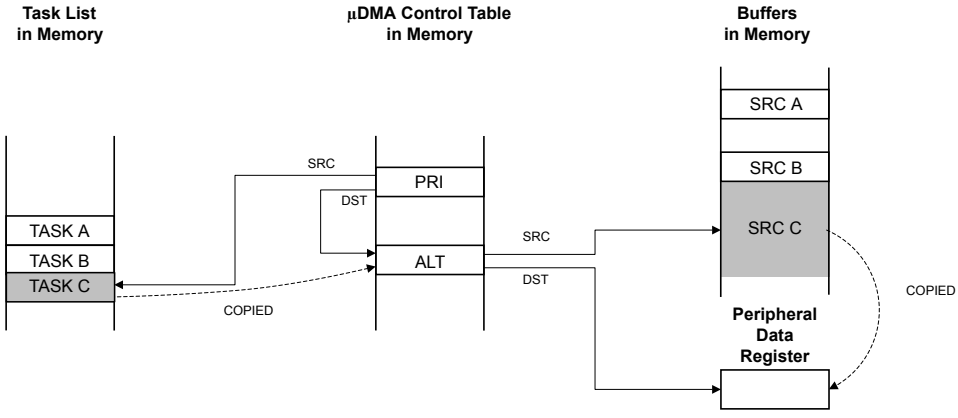
Using the channel's primary control structure, the µDMA controller copies task A configuration to the channel's alternate control structure.

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



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

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



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

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

7.2.7 Transfer Size and Increment

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

Table 7-5 shows the configuration to read from a peripheral that supplies 8-bit data.

Table 7-5. µDMA Read Example: 8-Bit Peripheral

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

7.2.8 Peripheral Interface

Each peripheral that supports µDMA has a single request and/or burst request signal that is asserted when the peripheral is ready to transfer data (see Table 7-2 on page 338). The request signal can be disabled or enabled using the **DMA Channel Request Mask Set (DMAREQMASKSET)** and **DMA Channel Request Mask Clear (DMAREQMASKCLR)** registers. The µDMA request signal is disabled, or masked, when the channel request mask bit is set. When the request is not masked, the µDMA channel is configured correctly and enabled, and the peripheral asserts the request signal, the µDMA controller begins the transfer.

Note: When using µDMA to transfer data to and from a peripheral, the peripheral must disable all interrupts to the NVIC.

When a µDMA transfer is complete, the µDMA controller generates an interrupt, see “Interrupts and Errors” on page 350 for more information.

For more information on how a specific peripheral interacts with the µDMA controller, refer to the DMA Operation section in the chapter that discusses that peripheral.

7.2.9 Software Request

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

It is possible to initiate a transfer on any channel using the **DMASWREQ** register. If a request is initiated by software using a peripheral µDMA channel, then the completion interrupt occurs on the interrupt vector for the peripheral instead of the software interrupt vector. Any channel may be used for software requests as long as the corresponding peripheral is not using µDMA for data transfer.

7.2.10 Interrupts and Errors

When a µDMA transfer is complete, the µDMA controller generates a completion interrupt on the interrupt vector of the peripheral. Therefore, if µDMA is used to transfer data for a peripheral and interrupts are used, then the interrupt handler for that peripheral must be designed to handle the µDMA transfer completion interrupt. If the transfer uses the software µDMA channel, then the completion interrupt occurs on the dedicated software µDMA interrupt vector (see Table 7-6 on page 350).

When µDMA is enabled for a peripheral, the µDMA controller stops the normal transfer interrupts for a peripheral from reaching the interrupt controller (the interrupts are still reported in the peripheral's interrupt registers). Thus, when a large amount of data is transferred using µDMA, instead of receiving multiple interrupts from the peripheral as data flows, the interrupt controller receives only one interrupt when the transfer is complete. Unmasked peripheral error interrupts continue to be sent to the interrupt controller.

When a µDMA channel generates a completion interrupt, the CHIS bit corresponding to the peripheral channel is set in the **DMA Channel Interrupt Status (DMACHIS)** register (see page 385). This register can be used by the peripheral interrupt handler code to determine if the interrupt was caused by the µDMA channel or an error event reported by the peripheral's interrupt registers. The completion interrupt request from the µDMA controller is automatically cleared when the interrupt handler is activated.

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

Table 7-6 shows the dedicated interrupt assignments for the µDMA controller.

Table 7-6. µDMA Interrupt Assignments

Interrupt	Assignment
46	µDMA Software Channel Transfer
47	µDMA Error

7.3 Initialization and Configuration

7.3.1 Module Initialization

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

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

1. The µDMA peripheral must be enabled in the System Control block. To do this, set the UDMA bit of the System Control **RCGC2** register (see page 272).
2. Enable the µDMA controller by setting the MASTEREN bit of the **DMA Configuration (DMACFG)** register.
3. Program the location of the channel control table by writing the base address of the table to the **DMA Channel Control Base Pointer (DMACTLBASE)** register. The base address must be aligned on a 1024-byte boundary.

7.3.2 Configuring a Memory-to-Memory Transfer

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

7.3.2.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Program bit 30 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.
2. Set bit 30 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.
3. Set bit 30 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the µDMA controller to respond to single and burst requests.
4. Set bit 30 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the µDMA controller to recognize requests for this channel.

7.3.2.2 Configure the Channel Control Structure

Now the channel control structure must be configured.

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

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

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

Configure the Source and Destination

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

1. Program the source end pointer at offset 0x1E0 to the address of the source buffer + 0x3FC.
2. Program the destination end pointer at offset 0x1E4 to the address of the destination buffer + 0x3FC.

The control word at offset 0x1E8 must be programmed according to Table 7-8.

Table 7-8. Channel Control Word Configuration for Memory Transfer Example

Field in DMACHCTL	Bits	Value	Description
DSTINC	31:30	2	32-bit destination address increment
DSTSIZE	29:28	2	32-bit destination data size
SRCINC	27:26	2	32-bit source address increment
SRCSIZE	25:24	2	32-bit source data size
reserved	23:18	0	Reserved

Table 7-8. Channel Control Word Configuration for Memory Transfer Example *(continued)*

Field in DMACHCTL	Bits	Value	Description
ARBSIZE	17:14	3	Arbitrates after 8 transfers
XFERSIZE	13:4	255	Transfer 256 items
NXTUSEBURST	3	0	N/A for this transfer type
XFERMODE	2:0	2	Use Auto-request transfer mode

7.3.2.3 Start the Transfer

Now the channel is configured and is ready to start.

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

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

7.3.3 Configuring a Peripheral for Simple Transmit

This example configures the µDMA controller to transmit a buffer of data to a peripheral. The peripheral has a transmit FIFO with a trigger level of 4. The example peripheral uses µDMA channel 7.

7.3.3.1 Configure the Channel Attributes

First, configure the channel attributes:

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

7.3.3.2 Configure the Channel Control Structure

This example transfers 64 bytes from a memory buffer to the peripheral's transmit FIFO register using µDMA channel 7. The control structure for channel 7 is at offset 0x070 of the channel control table. The channel control structure for channel 7 is located at the offsets shown in Table 7-9.

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

Offset	Description
Control Table Base + 0x070	Channel 7 Source End Pointer

Table 7-9. Channel Control Structure Offsets for Channel 7 *(continued)*

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

Configure the Source and Destination

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

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

The control word at offset 0x078 must be programmed according to Table 7-10.

Table 7-10. Channel Control Word Configuration for Peripheral Transmit Example

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

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

7.3.3.3 Start the Transfer

Now the channel is configured and is ready to start.

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

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

If peripheral interrupts are enabled, then the peripheral interrupt handler receives an interrupt when the entire transfer is complete.

7.3.4 Configuring a Peripheral for Ping-Pong Receive

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

7.3.4.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Configure bit 8 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.
2. Set bit 8 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.
3. Set bit 8 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the µDMA controller to respond to single and burst requests.
4. Set bit 8 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the µDMA controller to recognize requests for this channel.

7.3.4.2 Configure the Channel Control Structure

This example transfers bytes from the peripheral's receive FIFO register into two memory buffers of 64 bytes each. As data is received, when one buffer is full, the µDMA controller switches to use the other.

To use Ping-Pong buffering, both primary and alternate channel control structures must be used. The primary control structure for channel 8 is at offset 0x080 of the channel control table, and the alternate channel control structure is at offset 0x280. The channel control structures for channel 8 are located at the offsets shown in Table 7-11.

Table 7-11. Primary and Alternate Channel Control Structure Offsets for Channel 8

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

Configure the Source and Destination

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

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

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

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

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

The primary control word at offset 0x088 and the alternate control word at offset 0x288 are initially programmed the same way.

1. Program the primary channel control word at offset 0x088 according to Table 7-12.

2. Program the alternate channel control word at offset 0x288 according to Table 7-12.

Table 7-12. Channel Control Word Configuration for Peripheral Ping-Pong Receive Example

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

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

7.3.4.3 Configure the Peripheral Interrupt

An interrupt handler should be configured when using µDMA Ping-Pong mode, it is best to use an interrupt handler. However, the Ping-Pong mode can be configured without interrupts by polling. The interrupt handler is triggered after each buffer is complete.

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

7.3.4.4 Enable the µDMA Channel

Now the channel is configured and is ready to start.

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

7.3.4.5 Process Interrupts

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

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

In the interrupt handler:

1. Read the primary channel control word at offset 0x088 and check the XFERMODE field. If the field is 0, this means buffer A is complete. If buffer A is complete, then:
   a. Process the newly received data in buffer A or signal the buffer processing code that buffer A has data available.
   b. Reprogram the primary channel control word at offset 0x88 according to Table 7-12 on page 355.
2. Read the alternate channel control word at offset 0x288 and check the XFERMODE field. If the field is 0, this means buffer B is complete. If buffer B is complete, then:
   a. Process the newly received data in buffer B or signal the buffer processing code that buffer B has data available.
   b. Reprogram the alternate channel control word at offset 0x288 according to Table 7-12 on page 355.

7.3.5 Configuring Channel Assignments

Channel assignments for each µDMA channel can be changed using the **DMACHASGN** register. Each bit represents a µDMA channel. If the bit is set, then the secondary function is used for the channel.

Refer to Table 7-1 on page 336 for channel assignments.

For example, to use SSI1 Receive on channel 8 instead of UART0, set bit 8 of the **DMACHASGN** register.

7.4 Register Map

Table 7-13 on page 357 lists the µDMA channel control structures and registers. The channel control structure shows the layout of one entry in the channel control table. The channel control table is located in system memory, and the location is determined by the application, that is, the base address is n/a (not applicable). In the table below, the offset for the channel control structures is the offset from the entry in the channel control table. See “Channel Configuration” on page 339 and Table 7-3 on page 339 for a description of how the entries in the channel control table are located in memory. The µDMA register addresses are given as a hexadecimal increment, relative to the µDMA base address of 0x400F.F000. Note that the µDMA module clock must be enabled before the registers can be programmed (see page 272). There must be a delay of 3 system clocks after the µDMA module clock is enabled before any µDMA module registers are accessed.

Table 7-13. µDMA Register Map

Offset	Name	Type	Reset	Description	See page
µDMA Channel Control Structure (Offset from Channel Control Table Base)					
0x000	DMASRCENDP	R/W	-	DMA Channel Source Address End Pointer	359
0x004	DMADSTENDP	R/W	-	DMA Channel Destination Address End Pointer	360
0x008	DMACHCTL	R/W	-	DMA Channel Control Word	361
µDMA Registers (Offset from µDMA Base Address)					
0x000	DMASTAT	RO	0x001F.0000	DMA Status	366
0x004	DMACFG	WO	-	DMA Configuration	368
0x008	DMACTLBASE	R/W	0x0000.0000	DMA Channel Control Base Pointer	369
0x00C	DMAALTBASE	RO	0x0000.0200	DMA Alternate Channel Control Base Pointer	370
0x010	DMAWAITSTAT	RO	0xFFFF.FFC0	DMA Channel Wait-on-Request Status	371
0x014	DMASWREQ	WO	-	DMA Channel Software Request	372
0x018	DMAUSEBURSTSET	R/W	0x0000.0000	DMA Channel Useburst Set	373
0x01C	DMAUSEBURSTCLR	WO	-	DMA Channel Useburst Clear	374
0x020	DMAREQMASKSET	R/W	0x0000.0000	DMA Channel Request Mask Set	375
0x024	DMAREQMASKCLR	WO	-	DMA Channel Request Mask Clear	376
0x028	DMAENASET	R/W	0x0000.0000	DMA Channel Enable Set	377
0x02C	DMAENACLR	WO	-	DMA Channel Enable Clear	378
0x030	DMAALTSET	R/W	0x0000.0000	DMA Channel Primary Alternate Set	379
0x034	DMAALTCLR	WO	-	DMA Channel Primary Alternate Clear	380
0x038	DMAPRIOSET	R/W	0x0000.0000	DMA Channel Priority Set	381
0x03C	DMAPRIOCLR	WO	-	DMA Channel Priority Clear	382
0x04C	DMAERRCLR	R/W	0x0000.0000	DMA Bus Error Clear	383
0x500	DMACHASGN	R/W	0x0000.0000	DMA Channel Assignment	384
0x504	DMACHIS	R/W1C	0x0000.0000	DMA Channel Interrupt Status	385
0xFD0	DMAPeriphID4	RO	0x0000.0004	DMA Peripheral Identification 4	390
0xFE0	DMAPeriphID0	RO	0x0000.0030	DMA Peripheral Identification 0	386
0xFE4	DMAPeriphID1	RO	0x0000.00B2	DMA Peripheral Identification 1	387
0xFE8	DMAPeriphID2	RO	0x0000.000B	DMA Peripheral Identification 2	388
0xFEC	DMAPeriphID3	RO	0x0000.0000	DMA Peripheral Identification 3	389
0xFF0	DMAPCellID0	RO	0x0000.000D	DMA PrimeCell Identification 0	391
0xFF4	DMAPCellID1	RO	0x0000.00F0	DMA PrimeCell Identification 1	392
0xFF8	DMAPCellID2	RO	0x0000.0005	DMA PrimeCell Identification 2	393

Table 7-13. µDMA Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0xFFC	DMAPCellID3	RO	0x0000.00B1	DMA PrimeCell Identification 3	394

7.5 µDMA Channel Control Structure

The µDMA Channel Control Structure holds the transfer settings for a µDMA channel. Each channel has two control structures, which are located in a table in system memory. Refer to “Channel Configuration” on page 339 for an explanation of the Channel Control Table and the Channel Control Structure.

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

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

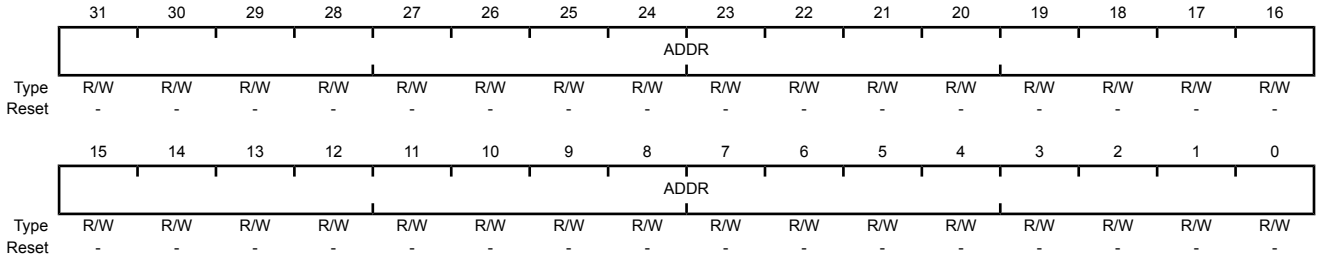
DMA Channel Source Address End Pointer (DMASRCENDP) is part of the Channel Control Structure and is used to specify the source address for a µDMA transfer.

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

Note: The offset specified is from the base address of the control structure in system memory, not the µDMA module base address.

DMA Channel Source Address End Pointer (DMASRCENDP)

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



Bit/Field	Name	Type	Reset	Description
31:0	ADDR	R/W	-	Source Address End Pointer This field points to the last address of the µDMA transfer source (inclusive). If the source address is not incrementing (the SRCINC field in the **DMACHCTL** register is 0x3), then this field points at the source location itself (such as a peripheral data register).

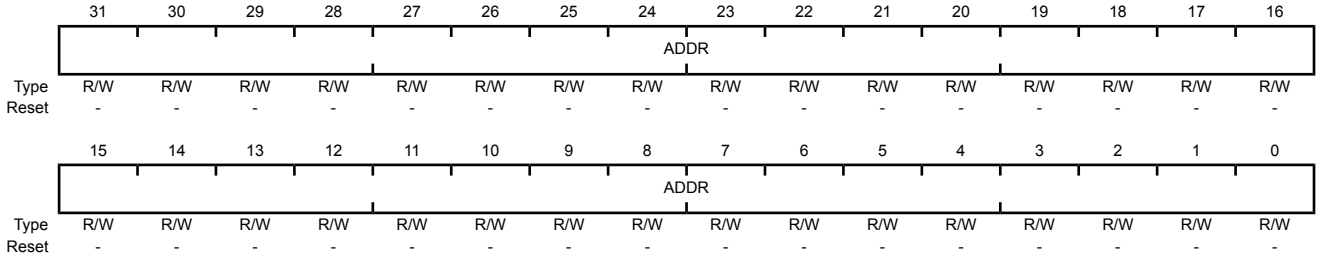
Register 2: DMA Channel Destination Address End Pointer (DMADSTENDP), offset 0x004

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

Note: The offset specified is from the base address of the control structure in system memory, not the µDMA module base address.

DMA Channel Destination Address End Pointer (DMADSTENDP)

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



Bit/Field	Name	Type	Reset	Description
31:0	ADDR	R/W	-	Destination Address End Pointer This field points to the last address of the µDMA transfer destination (inclusive). If the destination address is not incrementing (the DSTINC field in the **DMACHCTL** register is 0x3), then this field points at the destination location itself (such as a peripheral data register).

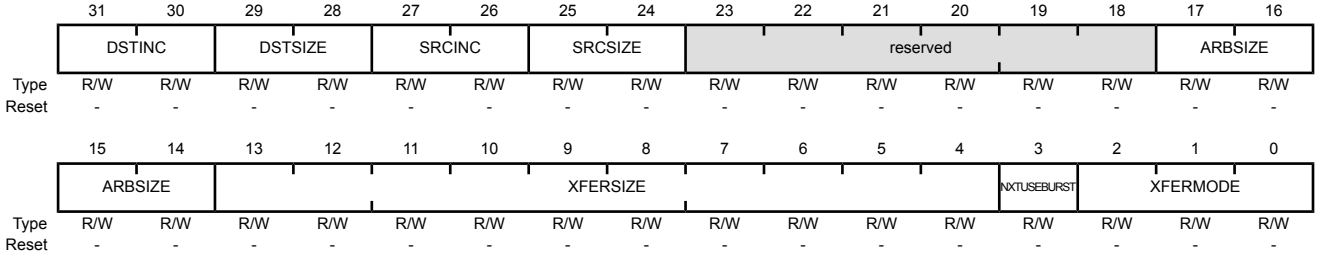
Register 3: DMA Channel Control Word (DMACHCTL), offset 0x008

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

Note: The offset specified is from the base address of the control structure in system memory, not the µDMA module base address.

DMA Channel Control Word (DMACHCTL)

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



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

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

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

Bit/Field	Name	Type	Reset	Description
2:0	XFERMODE	R/W	-	µDMA Transfer Mode This field configures the operating mode of the µDMA cycle. Refer to "Transfer Modes" on page 340 for a detailed explanation of transfer modes. Because this register is in system RAM, it has no reset value. Therefore, this field should be initialized to 0 before the channel is enabled. Value Description 0x0 Stop 0x1 Basic 0x2 Auto-Request 0x3 Ping-Pong 0x4 Memory Scatter-Gather 0x5 Alternate Memory Scatter-Gather 0x6 Peripheral Scatter-Gather 0x7 Alternate Peripheral Scatter-Gather

XFERMODE Bit Field Values.

Stop
Channel is stopped or configuration data is invalid. No more transfers can occur.

Basic
For each trigger (whether from a peripheral or a software request), the µDMA controller performs the number of transfers specified by the ARBSIZE field.

Auto-Request
The initial request (software- or peripheral-initiated) is sufficient to complete the entire transfer of XFERSIZE items without any further requests.

Ping-Pong
This mode uses both the primary and alternate control structures for this channel. When the number of transfers specified by the XFERSIZE field have completed for the current control structure (primary or alternate), the µDMA controller switches to the other one. These switches continue until one of the control structures is not set to ping-pong mode. At that point, the µDMA controller stops. An interrupt is generated on completion of the transfers configured by each control structure. See "Ping-Pong" on page 341.

Memory Scatter-Gather
When using this mode, the primary control structure for the channel is configured to allow a list of operations (tasks) to be performed. The source address pointer specifies the start of a table of tasks to be copied to the alternate control structure for this channel. The XFERMODE field for the alternate control structure should be configured to 0x5 (Alternate memory scatter-gather) to perform the task. When the task completes, the µDMA switches back to the primary channel control structure, which then copies the next task to the alternate control structure. This process continues until the table of tasks is empty. The last task must have an XFERMODE value other than 0x5. Note that for continuous operation, the last task can update the primary channel control structure back to the start of the list or to another list. See "Memory Scatter-Gather" on page 342.

Alternate Memory Scatter-Gather
: This value must be used in the alternate channel control data structure when the µDMA controller operates in Memory Scatter-Gather mode.

Peripheral Scatter-Gather
: This value must be used in the primary channel control data structure when the µDMA controller operates in Peripheral Scatter-Gather mode. In this mode, the µDMA controller operates exactly the same as in Memory Scatter-Gather mode, except that instead of performing the number of transfers specified by the XFERSIZE field in the alternate control structure at one time, the µDMA controller only performs the number of transfers specified by the ARBSIZE field per trigger; see Basic mode for details. See “Peripheral Scatter-Gather” on page 346.

Alternate Peripheral Scatter-Gather
: This value must be used in the alternate channel control data structure when the µDMA controller operates in Peripheral Scatter-Gather mode.

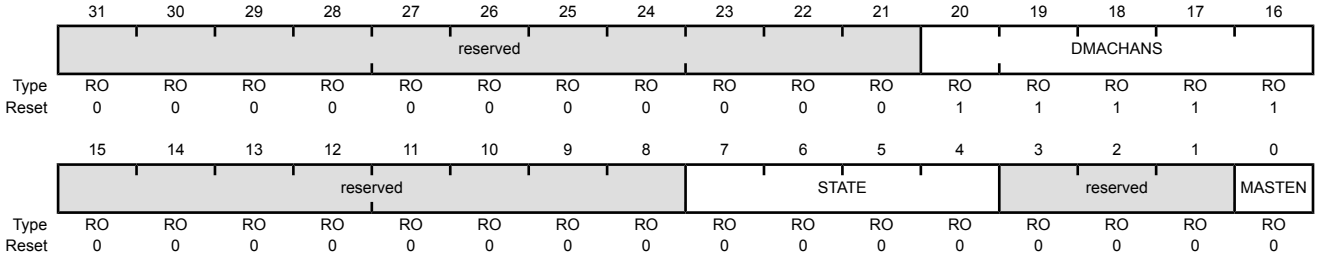
7.6 µDMA Register Descriptions

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

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

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

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



Bit/Field	Name	Type	Reset	Description
31:21	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
20:16	DMACHANS	RO	0x1F	Available µDMA Channels Minus 1 This field contains a value equal to the number of µDMA channels the µDMA controller is configured to use, minus one. The value of 0x1F corresponds to 32 µDMA channels.
15:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:4	STATE	RO	0x0	Control State Machine Status This field shows the current status of the control state machine. Status can be one of the following.
3:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Value	Description
0x0	Idle
0x1	Reading channel controller data.
0x2	Reading source end pointer.
0x3	Reading destination end pointer.
0x4	Reading source data.
0x5	Writing destination data.
0x6	Waiting for µDMA request to clear.
0x7	Writing channel controller data.
0x8	Stalled
0x9	Done
0xA-0xF	Undefined

Bit/Field	Name	Type	Reset	Description
0	MASTEN	RO	0	Master Enable Status

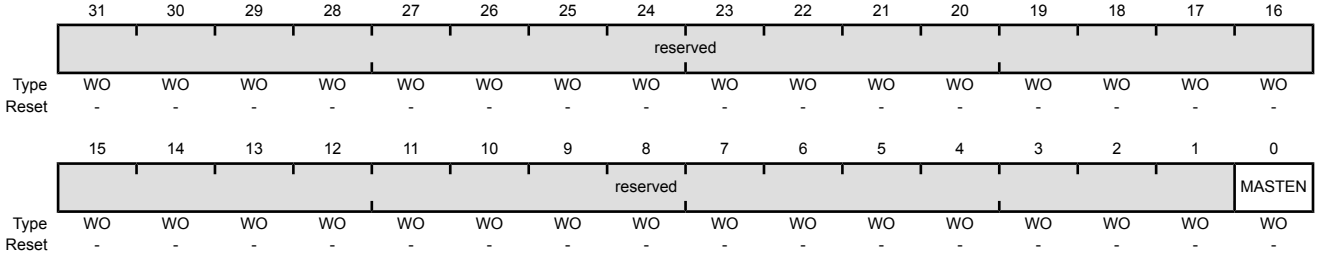
Value	Description
0	The µDMA controller is disabled.
1	The µDMA controller is enabled.

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

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

DMA Configuration (DMACFG)

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



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

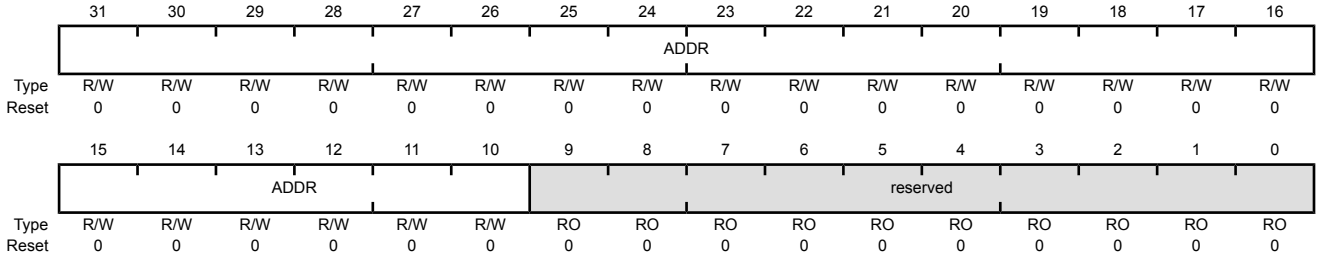
Register 6: DMA Channel Control Base Pointer (DMACTLBASE), offset 0x008

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

The amount of system memory that must be assigned to the µDMA controller depends on the number of µDMA channels used and whether the alternate channel control data structure is used. See “Channel Configuration” on page 339 for details about the Channel Control Table. The base address must be aligned on a 1024-byte boundary. This register cannot be read when the µDMA controller is in the reset state.

DMA Channel Control Base Pointer (DMACTLBASE)

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



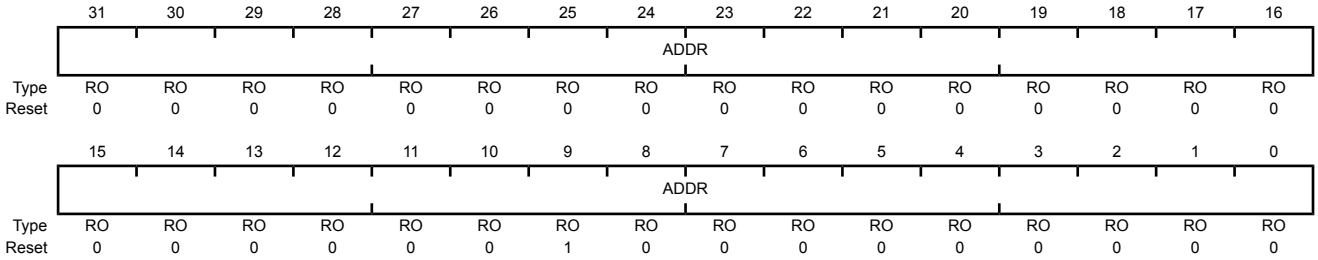
Bit/Field	Name	Type	Reset	Description
31:10	ADDR	R/W	0x0000.00	Channel Control Base Address This field contains the pointer to the base address of the channel control table. The base address must be 1024-byte aligned.
9:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

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

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

DMA Alternate Channel Control Base Pointer (DMAALTBASE)

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



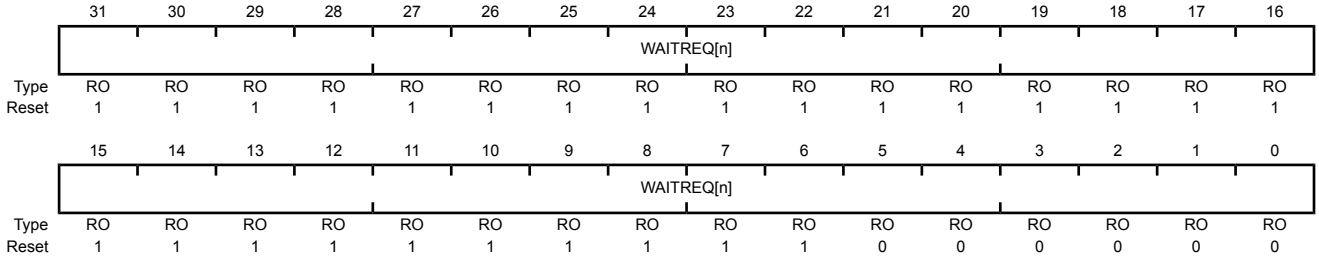
Bit/Field	Name	Type	Reset	Description
31:0	ADDR	RO	0x0000.0200	Alternate Channel Address Pointer This field provides the base address of the alternate channel control structures.

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

This read-only register indicates that the µDMA channel is waiting on a request. A peripheral can hold off the µDMA from performing a single request until the peripheral is ready for a burst request to enhance the µDMA performance. The use of this feature is dependent on the design of the peripheral and is not controllable by software in any way. This register cannot be read when the µDMA controller is in the reset state.

DMA Channel Wait-on-Request Status (DMAWAITSTAT)

Base 0x400F.F000
Offset 0x010
Type RO, reset 0xFFFF.FFC0



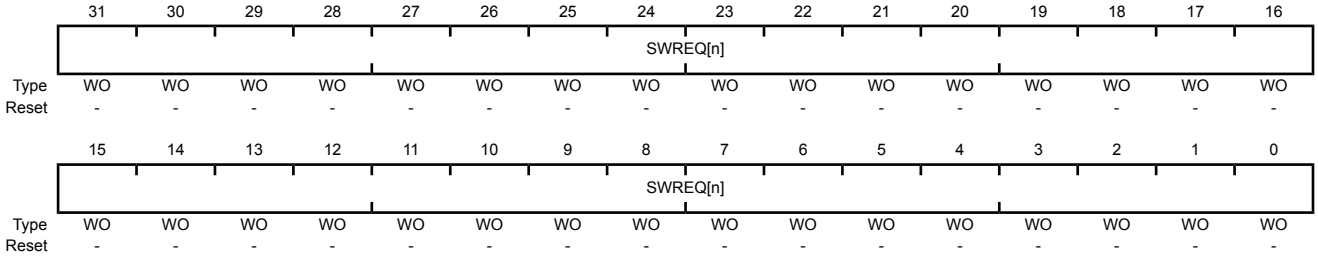
Bit/Field	Name	Type	Reset	Description
31:0	WAITREQ[n]	RO	0xFFFF.FFC0	Channel [n] Wait Status These bits provide the channel wait-on-request status. Bit 0 corresponds to channel 0. Value Description 1 The corresponding channel is waiting on a request. 0 The corresponding channel is not waiting on a request.

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

Each bit of the **DMASWREQ** register represents the corresponding µDMA channel. Setting a bit generates a request for the specified µDMA channel.

DMA Channel Software Request (DMASWREQ)

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



Bit/Field	Name	Type	Reset	Description
31:0	SWREQ[n]	WO	-	Channel [n] Software Request These bits generate software requests. Bit 0 corresponds to channel 0. Value Description 1 Generate a software request for the corresponding channel. 0 No request generated. These bits are automatically cleared when the software request has been completed.

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

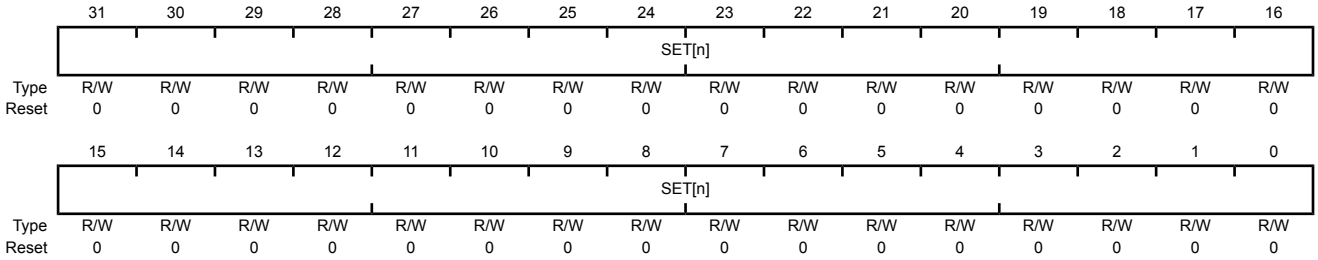
Each bit of the **DMAUSEBURSTSET** register represents the corresponding µDMA channel. Setting a bit disables the channel's single request input from generating requests, configuring the channel to only accept burst requests. Reading the register returns the status of USEBURST.

If the amount of data to transfer is a multiple of the arbitration (burst) size, the corresponding SET[n] bit is cleared after completing the final transfer. If there are fewer items remaining to transfer than the arbitration (burst) size, the µDMA controller automatically clears the corresponding SET[n] bit, allowing the remaining items to transfer using single requests. In order to resume transfers using burst requests, the corresponding bit must be set again. A bit should not be set if the corresponding peripheral does not support the burst request model.

Refer to “Request Types” on page 338 for more details about request types.

DMA Channel Useburst Set (DMAUSEBURSTSET)

Base 0x400F.F000
Offset 0x018
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R/W	0x0000.0000	Channel [n] Useburst Set

Value	Description
0	µDMA channel [n] responds to single or burst requests.
1	µDMA channel [n] responds only to burst requests.

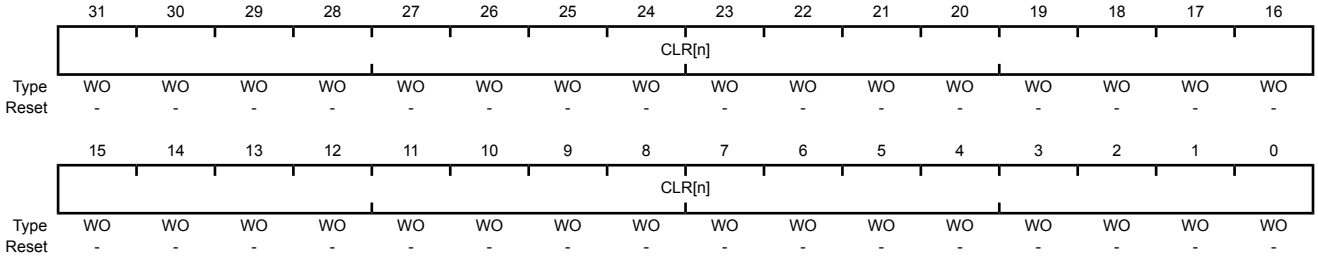
Bit 0 corresponds to channel 0. This bit is automatically cleared as described above. A bit can also be manually cleared by setting the corresponding CLR[n] bit in the **DMAUSEBURSTCLR** register.

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

Each bit of the **DMAUSEBURSTCLR** register represents the corresponding µDMA channel. Setting a bit clears the corresponding SET[n] bit in the **DMAUSEBURSTSET** register.

DMA Channel Useburst Clear (DMAUSEBURSTCLR)

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



Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Channel [n] Useburst Clear

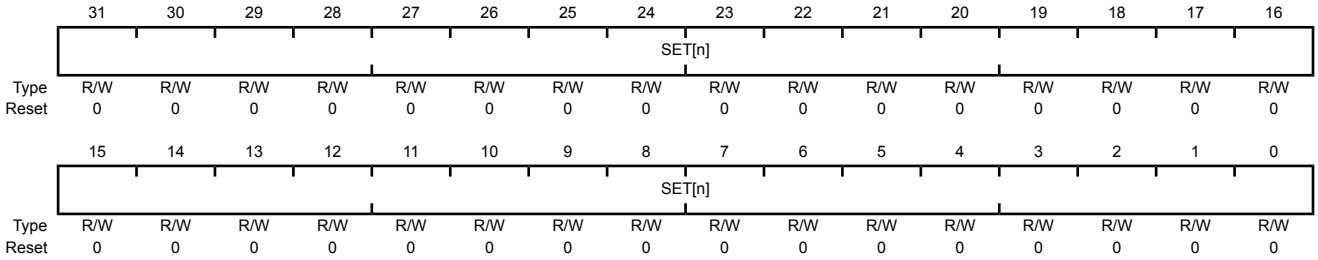
Value	Description
0	No effect.
1	Setting a bit clears the corresponding SET[n] bit in the **DMAUSEBURSTSET** register meaning that µDMA channel [n] responds to single and burst requests.

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

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

DMA Channel Request Mask Set (DMAREQMASKSET)

Base 0x400F.F000
Offset 0x020
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R/W	0x0000.0000	Channel [n] Request Mask Set

Value	Description
0	The peripheral associated with channel [n] is enabled to request µDMA transfers.
1	The peripheral associated with channel [n] is not able to request µDMA transfers. Channel [n] may be used for software-initiated transfers.

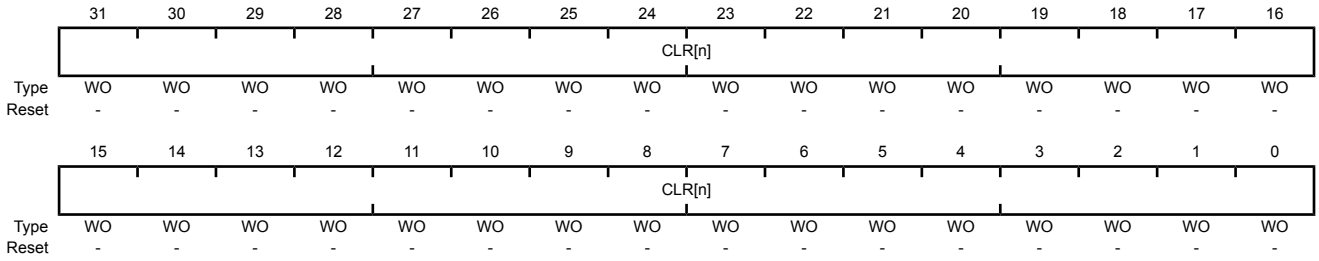
Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding `CLR[n]` bit in the **DMAREQMASKCLR** register.

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

Each bit of the **DMAREQMASKCLR** register represents the corresponding µDMA channel. Setting a bit clears the corresponding SET[n] bit in the **DMAREQMASKSET** register.

DMA Channel Request Mask Clear (DMAREQMASKCLR)

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



Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Channel [n] Request Mask Clear

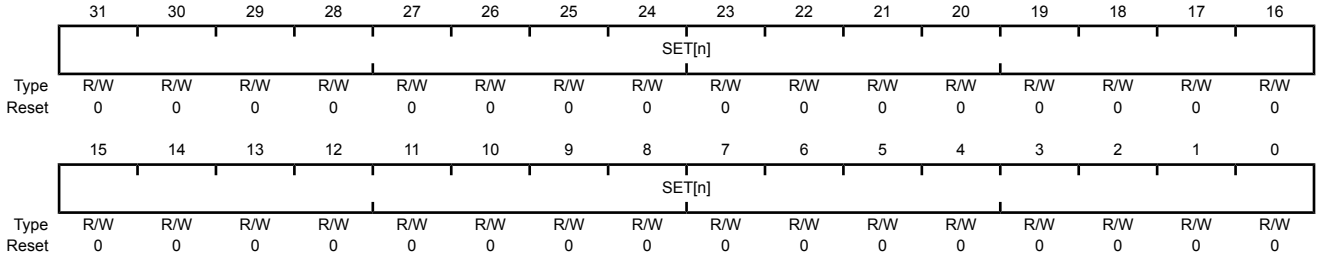
Value	Description
0	No effect.
1	Setting a bit clears the corresponding SET[n] bit in the **DMAREQMASKSET** register meaning that the peripheral associated with channel [n] is enabled to request µDMA transfers.

Register 14: DMA Channel Enable Set (DMAENASET), offset 0x028

Each bit of the **DMAENASET** register represents the corresponding µDMA channel. Setting a bit enables the corresponding µDMA channel. Reading the register returns the enable status of the channels. If a channel is enabled but the request mask is set (**DMAREQMASKSET**), then the channel can be used for software-initiated transfers.

DMA Channel Enable Set (DMAENASET)

Base 0x400F.F000
Offset 0x028
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R/W	0x0000.0000	Channel [n] Enable Set

Value	Description
0	µDMA Channel [n] is disabled.
1	µDMA Channel [n] is enabled.

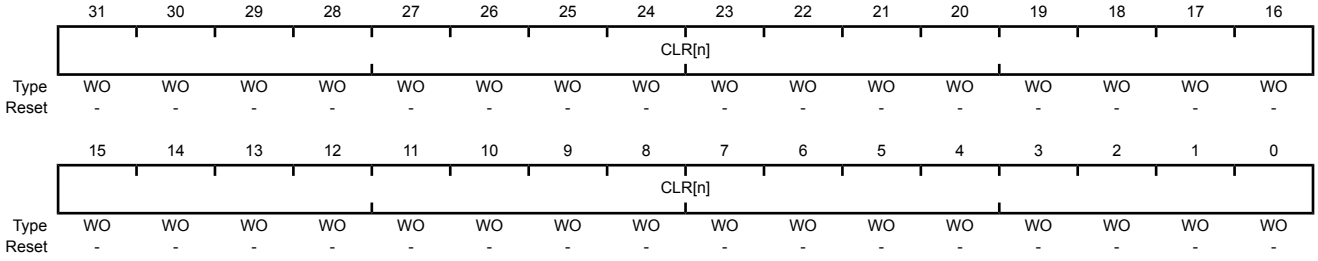
Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding `CLR[n]` bit in the **DMAENACLR** register.

Register 15: DMA Channel Enable Clear (DMAENACLR), offset 0x02C

Each bit of the **DMAENACLR** register represents the corresponding µDMA channel. Setting a bit clears the corresponding SET[n] bit in the **DMAENASET** register.

DMA Channel Enable Clear (DMAENACLR)

Base 0x400F.F000
Offset 0x02C
Type WO, reset -



Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Clear Channel [n] Enable Clear

Value	Description
0	No effect.
1	Setting a bit clears the corresponding SET[n] bit in the **DMAENASET** register meaning that channel [n] is disabled for µDMA transfers.

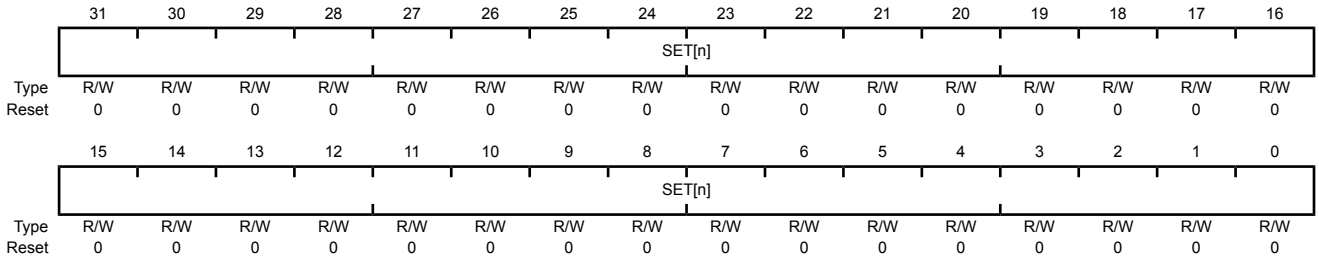
Note: The controller disables a channel when it completes the µDMA cycle.

Register 16: DMA Channel Primary Alternate Set (DMAALTSET), offset 0x030

Each bit of the **DMAALTSET** register represents the corresponding µDMA channel. Setting a bit configures the µDMA channel to use the alternate control data structure. Reading the register returns the status of which control data structure is in use for the corresponding µDMA channel.

DMA Channel Primary Alternate Set (DMAALTSET)

Base 0x400F.F000
Offset 0x030
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R/W	0x0000.0000	Channel [n] Alternate Set

Value	Description
0	µDMA channel [n] is using the primary control structure.
1	µDMA channel [n] is using the alternate control structure.

Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding `CLR[n]` bit in the **DMAALTCLR** register.

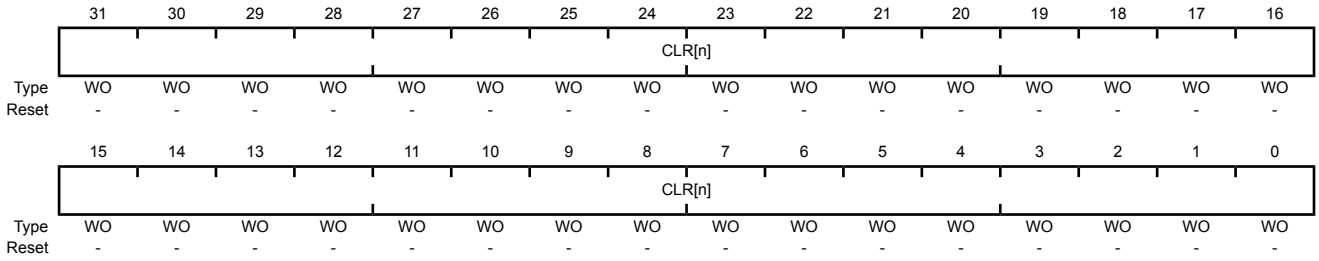
Note: For Ping-Pong and Scatter-Gather cycle types, the µDMA controller automatically sets these bits to select the alternate channel control data structure.

Register 17: DMA Channel Primary Alternate Clear (DMAALTCLR), offset 0x034

Each bit of the **DMAALTCLR** register represents the corresponding µDMA channel. Setting a bit clears the corresponding SET[n] bit in the **DMAALTSET** register.

DMA Channel Primary Alternate Clear (DMAALTCLR)

Base 0x400F.F000
Offset 0x034
Type WO, reset -



Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Channel [n] Alternate Clear

Value	Description
0	No effect.
1	Setting a bit clears the corresponding SET[n] bit in the **DMAALTSET** register meaning that channel [n] is using the primary control structure.

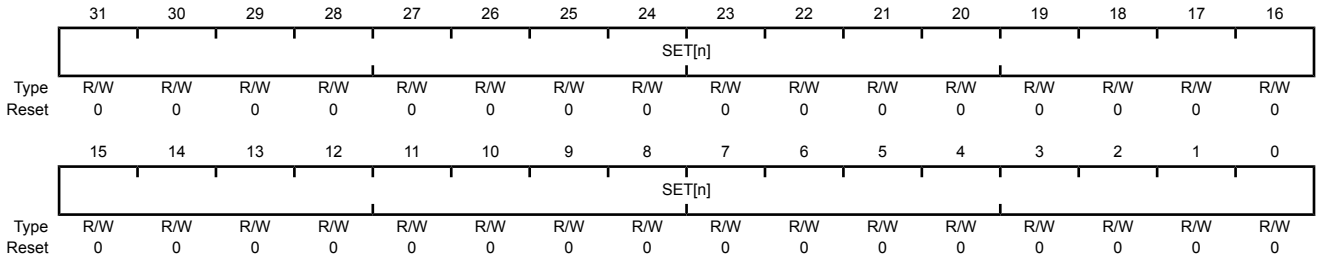
Note: For Ping-Pong and Scatter-Gather cycle types, the µDMA controller automatically sets these bits to select the alternate channel control data structure.

Register 18: DMA Channel Priority Set (DMAPRIOSET), offset 0x038

Each bit of the **DMAPRIOSET** register represents the corresponding µDMA channel. Setting a bit configures the µDMA channel to have a high priority level. Reading the register returns the status of the channel priority mask.

DMA Channel Priority Set (DMAPRIOSET)

Base 0x400F.F000
Offset 0x038
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R/W	0x0000.0000	Channel [n] Priority Set

Value	Description
0	µDMA channel [n] is using the default priority level.
1	µDMA channel [n] is using a high priority level.

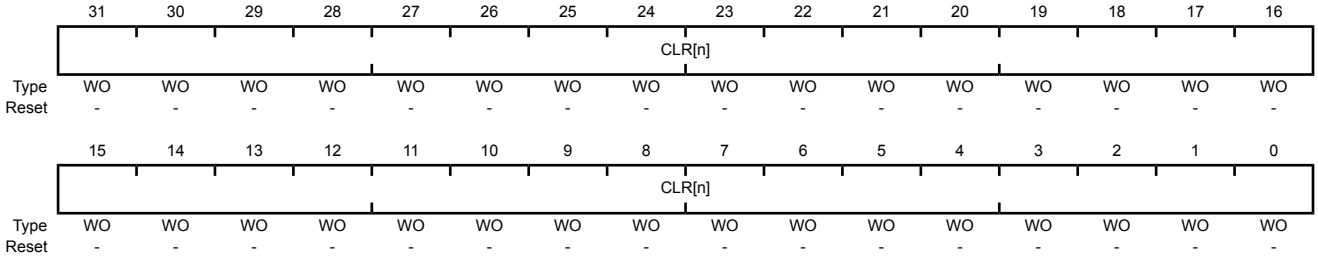
Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding `CLR[n]` bit in the **DMAPRIOCLR** register.

Register 19: DMA Channel Priority Clear (DMAPRIOCLR), offset 0x03C

Each bit of the **DMAPRIOCLR** register represents the corresponding µDMA channel. Setting a bit clears the corresponding SET[n] bit in the **DMAPRIOSET** register.

DMA Channel Priority Clear (DMAPRIOCLR)

Base 0x400F.F000
Offset 0x03C
Type WO, reset -



Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Channel [n] Priority Clear

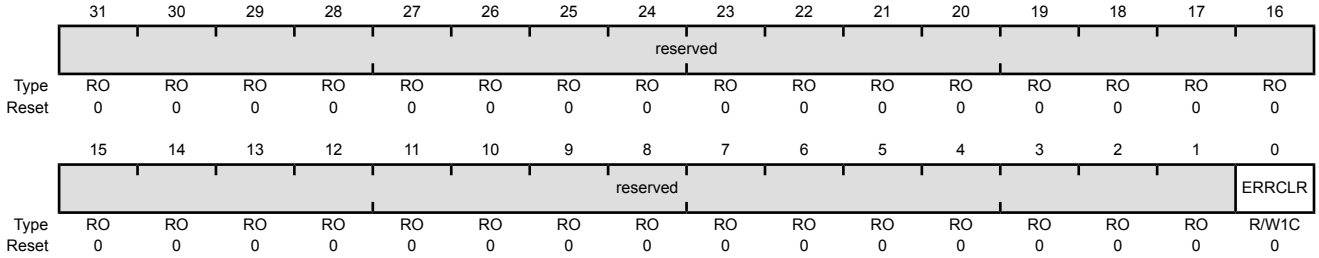
Value	Description
0	No effect.
1	Setting a bit clears the corresponding SET[n] bit in the **DMAPRIOSET** register meaning that channel [n] is using the default priority level.

Register 20: DMA Bus Error Clear (DMAERRCLR), offset 0x04C

The **DMAERRCLR** register is used to read and clear the µDMA bus error status. The error status is set if the µDMA controller encountered a bus error while performing a transfer. If a bus error occurs on a channel, that channel is automatically disabled by the µDMA controller. The other channels are unaffected.

DMA Bus Error Clear (DMAERRCLR)

Base 0x400F.F000
Offset 0x04C
Type R/W, reset 0x0000.0000



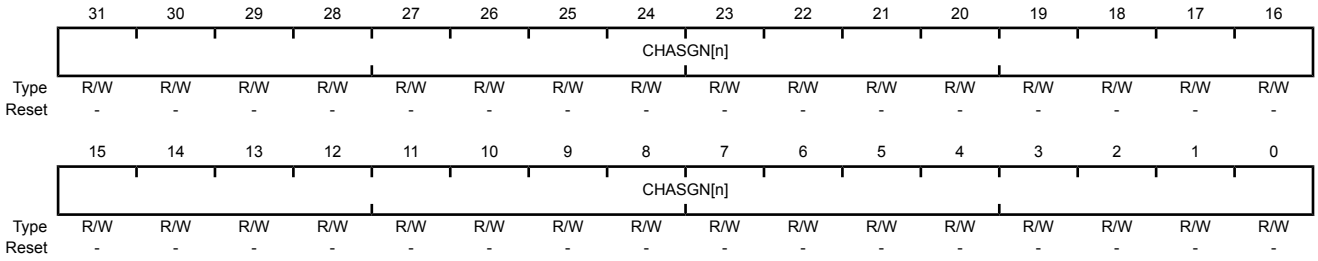
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	ERRCLR	R/W1C	0	µDMA Bus Error Status Value Description 0 No bus error is pending. 1 A bus error is pending. This bit is cleared by writing a 1 to it.

Register 21: DMA Channel Assignment (DMACHASGN), offset 0x500

Each bit of the **DMACHASGN** register represents the corresponding µDMA channel. Setting a bit selects the secondary channel assignment as specified in Table 7-1 on page 336.

DMA Channel Assignment (DMACHASGN)

Base 0x400F.F000
Offset 0x500
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	CHASGN[n]	R/W	-	Channel [n] Assignment Select

Value	Description
0	Use the primary channel assignment.
1	Use the secondary channel assignment.

Register 22: DMA Channel Interrupt Status (DMACHIS), offset 0x504

Each bit of the **DMACHIS** register represents the corresponding µDMA channel. A bit is set when that µDMA channel causes a completion interrupt. The bits are cleared by a writing a 1.

DMA Channel Interrupt Status (DMACHIS)

Base 0x400F.F000
Offset 0x504
Type R/W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CHIS[n]															
Type	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CHIS[n]															
Type	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	CHIS[n]	R/W1C	0x0000.0000	Channel [n] Interrupt Status

Value	Description
1	The corresponding µDMA channel caused an interrupt.
0	The corresponding µDMA channel has not caused an interrupt.

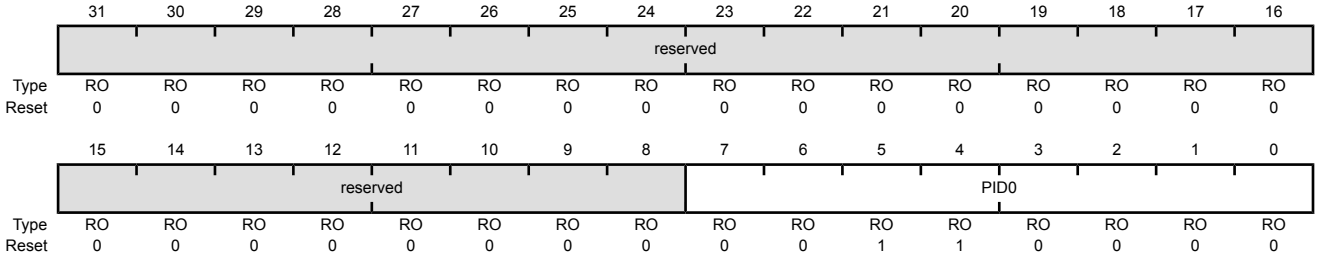
This bit is cleared by writing a 1 to it.

Register 23: DMA Peripheral Identification 0 (DMAPeriphID0), offset 0xFE0

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

DMA Peripheral Identification 0 (DMAPeriphID0)

Base 0x400F.F000
Offset 0xFE0
Type RO, reset 0x0000.0030



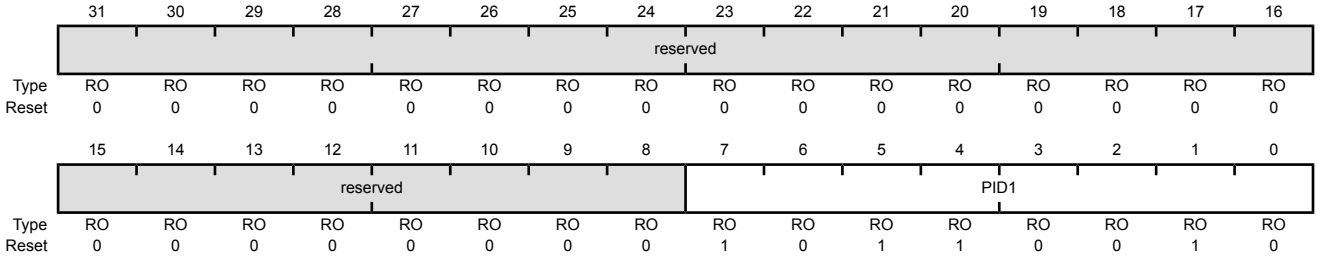
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x30	µDMA Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

Register 24: DMA Peripheral Identification 1 (DMAPeriphID1), offset 0xFE4

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

DMA Peripheral Identification 1 (DMAPeriphID1)

Base 0x400F.F000
Offset 0xFE4
Type RO, reset 0x0000.00B2



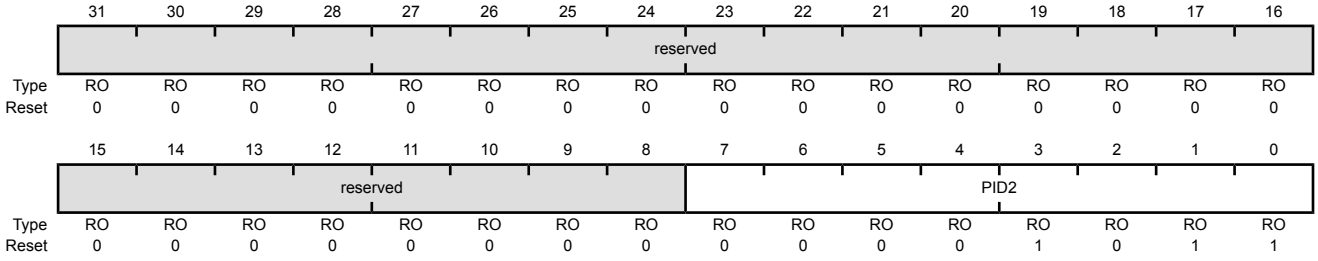
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0xB2	µDMA Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Register 25: DMA Peripheral Identification 2 (DMAPeriphID2), offset 0xFE8

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

DMA Peripheral Identification 2 (DMAPeriphID2)

Base 0x400F.F000
Offset 0xFE8
Type RO, reset 0x0000.000B



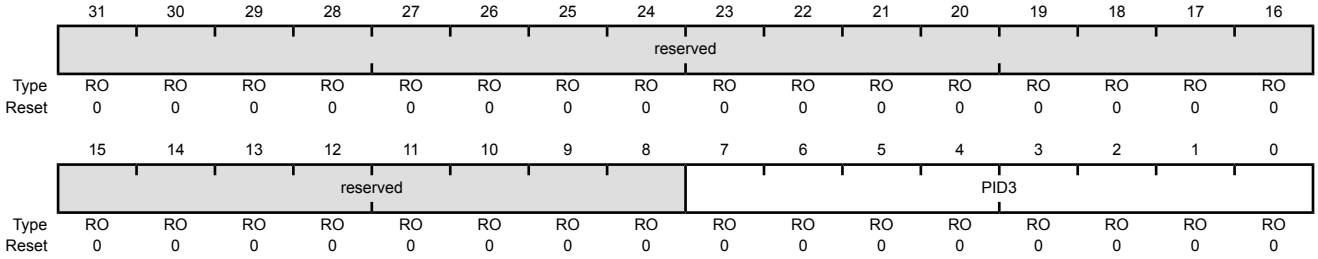
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x0B	µDMA Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 26: DMA Peripheral Identification 3 (DMAPeriphID3), offset 0xFEC

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

DMA Peripheral Identification 3 (DMAPeriphID3)

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



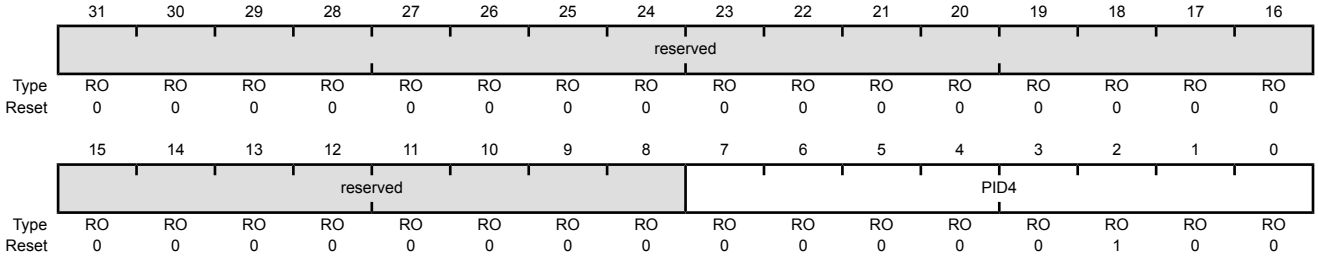
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x00	µDMA Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

Register 27: DMA Peripheral Identification 4 (DMAPeriphID4), offset 0xFD0

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

DMA Peripheral Identification 4 (DMAPeriphID4)

Base 0x400F.F000
Offset 0xFD0
Type RO, reset 0x0000.0004



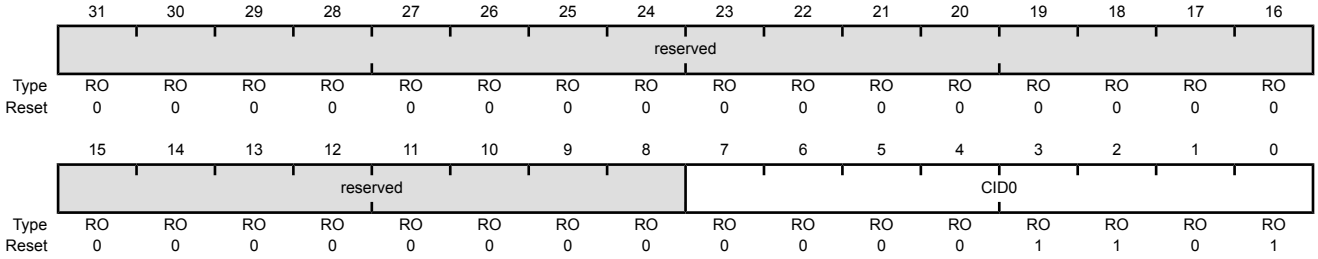
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x04	µDMA Peripheral ID Register Can be used by software to identify the presence of this peripheral.

Register 28: DMA PrimeCell Identification 0 (DMAPCellID0), offset 0xFF0

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

DMA PrimeCell Identification 0 (DMAPCellID0)

Base 0x400F.F000
Offset 0xFF0
Type RO, reset 0x0000.000D



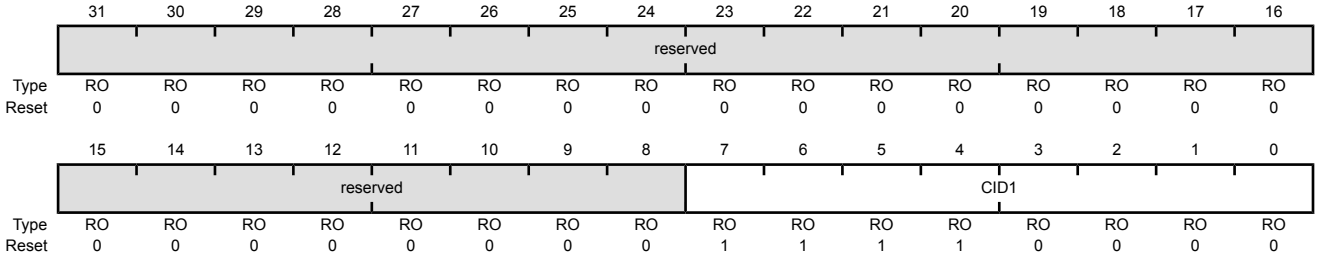
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	µDMA PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

Register 29: DMA PrimeCell Identification 1 (DMAPCellID1), offset 0xFF4

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

DMA PrimeCell Identification 1 (DMAPCellID1)

Base 0x400F.F000
Offset 0xFF4
Type RO, reset 0x0000.00F0



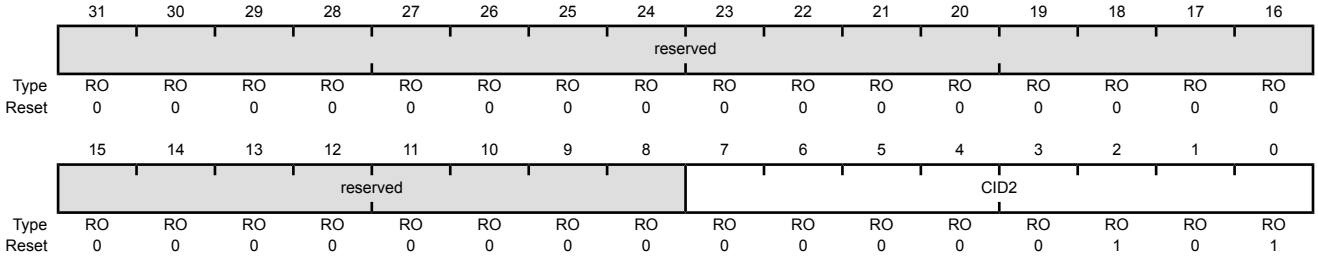
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	µDMA PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system.

Register 30: DMA PrimeCell Identification 2 (DMAPCellID2), offset 0xFF8

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

DMA PrimeCell Identification 2 (DMAPCellID2)

Base 0x400F.F000
Offset 0xFF8
Type RO, reset 0x0000.0005



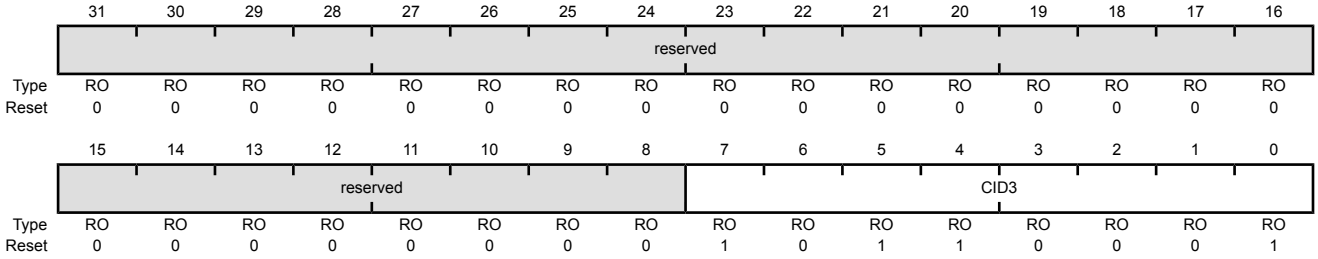
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	µDMA PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system.

Register 31: DMA PrimeCell Identification 3 (DMAPCellID3), offset 0xFFC

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

DMA PrimeCell Identification 3 (DMAPCellID3)

Base 0x400F.F000
Offset 0xFFC
Type RO, reset 0x0000.00B1



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	µDMA PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system.

8 General-Purpose Input/Outputs (GPIOs)

The GPIO module is composed of nine physical GPIO blocks, each corresponding to an individual GPIO port (Port A, Port B, Port C, Port D, Port E, Port F, Port G, Port H, Port J). The GPIO module supports up to 65 programmable input/output pins, depending on the peripherals being used.

The GPIO module has the following features:

- Up to 65 GPIOs, depending on configuration
- Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
- 5-V-tolerant in input configuration
- Two means of port access: either Advanced High-Performance Bus (AHB) with better back-to-back access performance, or the legacy Advanced Peripheral Bus (APB) for backwards-compatibility with existing code
- Fast toggle capable of a change every clock cycle for ports on AHB, every two clock cycles for ports on APB
- Programmable control for GPIO interrupts
  - Interrupt generation masking
  - Edge-triggered on rising, falling, or both
  - Level-sensitive on High or Low values
- Bit masking in both read and write operations through address lines
- Can be used to initiate an ADC sample sequence
- Pins configured as digital inputs are Schmitt-triggered
- Programmable control for GPIO pad configuration
  - Weak pull-up or pull-down resistors
  - 2-mA, 4-mA, and 8-mA pad drive for digital communication; up to four pads can sink 18-mA for high-current applications
  - Slew rate control for the 8-mA drive
  - Open drain enables
  - Digital input enables

8.1 Signal Description

GPIO signals have alternate hardware functions. The following table lists the GPIO pins and their analog and digital alternate functions. The AINx and VREFA analog signals are not 5-V tolerant and go through an isolation circuit before reaching their circuitry. These signals are configured by clearing the corresponding DEN bit in the **GPIO Digital Enable (GPIODEN)** register and setting the corresponding AMSEL bit in the **GPIO Analog Mode Select (GPIOAMSEL)** register. Other analog

signals are 5-V tolerant and are connected directly to their circuitry (C0-, C0+, C1-, C1+, C2-, C2+, USB0VBUS, USB0ID). These signals are configured by clearing the DEN bit in the **GPIO Digital Enable (GPIODEN)** register. All GPIO signals are 5-V tolerant when configured as inputs except for PB0 and PB1, which are limited to 3.6 V. The digital alternate hardware functions are enabled by setting the appropriate bit in the **GPIO Alternate Function Select (GPIOAFSEL)** and **GPIODEN** registers and configuring the PMCx bit field in the **GPIO Port Control (GPIOPCTL)** register to the numeric encoding shown in the table below. Note that each pin must be programmed individually; no type of grouping is implied by the columns in the table. Table entries that are shaded gray are the default values for the corresponding GPIO pin.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{POR}$) or asserting $\overline{RST}$ puts the pins back to their default state.

Table 8-1. GPIO Pins With Non-Zero Reset Values

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	0	0	0	0	0x1
PA[5:2]	SSI0	0	0	0	0	0x2
PB[3:2]	I^2C0	0	0	0	0	0x3
PC[3:0]	JTAG/SWD	1	1	0	1	0x1

Table 8-2. GPIO Pins and Alternate Functions (100LQFP)

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	10	11
PA0	26	-	U0Rx	-	-	-	-	-	-	I2C1SCL	U1Rx	-	-
PA1	27	-	U0Tx	-	-	-	-	-	-	I2C1SDA	U1Tx	-	-
PA2	28	-	SSI0Clk	-	-	-	-	-	-	-	I2S0RXSD	-	-
PA3	29	-	SSI0Fss	-	-	-	-	-	-	-	I2S0RXMCLK	-	-
PA4	30	-	SSI0Rx	-	-	-	CAN0Rx	-	-	-	I2S0TXSCK	-	-
PA5	31	-	SSI0Tx	-	-	-	CAN0Tx	-	-	-	I2S0TXWS	-	-
PA6	34	-	I2C1SCL	CCP1	-	-	-	CAN0Rx	-	USB0EPEN	U1CTS	-	-
PA7	35	-	I2C1SDA	CCP4	-	-	-	CAN0Tx	CCP3	USB0PFLT	U1DCD	-	-
PB0	66	USB0ID	CCP0	-	-	-	U1Rx	-	-	-	-	-	-
PB1	67	USB0VBUS	CCP2	-	-	CCP1	U1Tx	-	-	-	-	-	-
PB2	72	-	I2C0SCL	-	-	CCP3	CCP0	-	-	USB0EPEN	-	-	-
PB3	65	-	I2C0SDA	-	-	-	-	-	-	USB0PFLT	-	-	-
PB4	92	AIN10 C0-	-	-	-	U2Rx	CAN0Rx	-	U1Rx	EPI0S23	-	-	-
PB5	91	AIN11 C1-	C0o	CCP5	CCP6	CCP0	CAN0Tx	CCP2	U1Tx	EPI0S22	-	-	-
PB6	90	VREFA C0+	CCP1	CCP7	C0o	-	-	CCP5	-	-	I2S0TXSCK	-	-
PB7	89	-	-	-	-	NMI	-	-	-	-	-	-	-
PC0	80	-	-	-	TCK SWCLK	-	-	-	-	-	-	-	-

Table 8-2. GPIO Pins and Alternate Functions (100LQFP) *(continued)*

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	10	11
PC1	79	-	-	-	TMS SWDIO	-	-	-	-	-	-	-	-
PC2	78	-	-	-	TDI	-	-	-	-	-	-	-	-
PC3	77	-	-	-	TDO SWO	-	-	-	-	-	-	-	-
PC4	25	-	CCP5	-	-	-	CCP2	CCP4	-	EPI0S2	CCP1	-	-
PC5	24	C1+	CCP1	C1o	C0o	-	CCP3	USB0EPEN	-	EPI0S3	-	-	-
PC6	23	C2+	CCP3	-	C2o	-	U1Rx	CCP0	USB0PFLT	EPI0S4	-	-	-
PC7	22	C2-	CCP4	-	-	CCP0	U1Tx	USB0PFLT	C1o	EPI0S5	-	-	-
PD0	10	AIN15	-	CAN0Rx	-	U2Rx	U1Rx	CCP6	-	I2S0RXSCK	U1CTS	-	-
PD1	11	AIN14	-	CAN0Tx	-	U2Tx	U1Tx	CCP7	-	I2S0RXWS	U1DCD	CCP2	-
PD2	12	AIN13	U1Rx	CCP6	-	CCP5	-	-	-	EPI0S20	-	-	-
PD3	13	AIN12	U1Tx	CCP7	-	CCP0	-	-	-	EPI0S21	-	-	-
PD4	97	AIN7	CCP0	CCP3	-	-	-	-	-	I2S0RXSD	U1RI	EPI0S19	-
PD5	98	AIN6	CCP2	CCP4	-	-	-	-	-	I2S0RXMCLK	U2Rx	EPI0S28	-
PD6	99	AIN5	-	-	-	-	-	-	-	I2S0TXSCK	U2Tx	EPI0S29	-
PD7	100	AIN4	-	C0o	CCP1	-	-	-	-	I2S0TXWS	U1DTR	EPI0S30	-
PE0	74	-	-	SSI1Clk	CCP3	-	-	-	-	EPI0S8	USB0PFLT	-	-
PE1	75	-	-	SSI1Fss	-	CCP2	CCP6	-	-	EPI0S9	-	-	-
PE2	95	AIN9	CCP4	SSI1Rx	-	-	CCP2	-	-	EPI0S24	-	-	-
PE3	96	AIN8	CCP1	SSI1Tx	-	-	CCP7	-	-	EPI0S25	-	-	-
PE4	6	AIN3	CCP3	CAN2Rx	-	-	U2Tx	CCP2	-	-	I2S0TXWS	-	-
PE5	5	AIN2	CCP5	CAN2Tx	-	-	-	-	-	-	I2S0TXSD	-	-
PE6	2	AIN1	-	C1o	-	-	-	-	-	-	U1CTS	-	-
PE7	1	AIN0	-	C2o	-	-	-	-	-	-	U1DCD	-	-
PF0	47	-	CAN1Rx	-	-	-	-	-	-	I2S0TXSD	U1DSR	-	-
PF1	61	-	CAN1Tx	-	-	-	-	-	-	I2S0TXMCLK	U1RTS	CCP3	-
PF2	60	-	LED1	-	-	-	-	-	-	-	SSI1Clk	-	-
PF3	59	-	LED0	-	-	-	-	-	-	-	SSI1Fss	-	-
PF4	42	-	CCP0	C0o	-	-	-	-	-	EPI0S12	SSI1Rx	-	-
PF5	41	-	CCP2	C1o	-	-	-	-	-	EPI0S15	SSI1Tx	-	-
PG0	19	-	U2Rx	-	I2C1SCL	-	-	-	USB0EPEN	EPI0S13	-	-	-
PG1	18	-	U2Tx	-	I2C1SDA	-	-	-	-	EPI0S14	-	-	-
PG7	36	-	-	-	-	-	-	-	-	CCP5	EPI0S31	-	-
PH0	86	-	CCP6	-	-	-	-	-	-	EPI0S6	-	-	-
PH1	85	-	CCP7	-	-	-	-	-	-	EPI0S7	-	-	-
PH2	84	-	-	C1o	-	-	-	-	-	EPI0S1	-	-	-
PH3	83	-	-	-	-	USB0EPEN	-	-	-	EPI0S0	-	-	-
PH4	76	-	-	-	-	USB0PFLT	-	-	-	EPI0S10	-	-	SSI1Clk
PH5	63	-	-	-	-	-	-	-	-	EPI0S11	-	-	SSI1Fss
PH6	62	-	-	-	-	-	-	-	-	EPI0S26	-	-	SSI1Rx

Table 8-2. GPIO Pins and Alternate Functions (100LQFP) *(continued)*

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	10	11
PH7	15	-	-	-	-	-	-	-	-	EPI0S27	-	-	SSI1Tx
PJ0	14	-	-	-	-	-	-	-	-	EPI0S16	-	-	I2C1SCL
PJ1	87	-	-	-	-	-	-	-	-	EPI0S17	USB0PFLT	-	I2C1SDA
PJ2	39	-	-	-	-	-	-	-	-	EPI0S18	CCP0	-	-
PJ3	50	-	-	-	-	-	-	-	-	EPI0S19	U1CTS	CCP6	-
PJ4	52	-	-	-	-	-	-	-	-	EPI0S28	U1DCD	CCP4	-
PJ5	53	-	-	-	-	-	-	-	-	EPI0S29	U1DSR	CCP2	-
PJ6	54	-	-	-	-	-	-	-	-	EPI0S30	U1RTS	CCP1	-
PJ7	55	-	-	-	-	-	-	-	-	-	U1DTR	CCP0	-

a. The digital signals that are shaded gray are the power-on default values for the corresponding GPIO pin.

Table 8-3. GPIO Pins and Alternate Functions (108BGA)

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	10	11
PA0	L3	-	U0Rx	-	-	-	-	-	-	I2C1SCL	U1Rx	-	-
PA1	M3	-	U0Tx	-	-	-	-	-	-	I2C1SDA	U1Tx	-	-
PA2	M4	-	SSI0Clk	-	-	-	-	-	-	-	I2S0RXSD	-	-
PA3	L4	-	SSI0Fss	-	-	-	-	-	-	-	I2S0RXMCLK	-	-
PA4	L5	-	SSI0Rx	-	-	-	CAN0Rx	-	-	-	I2S0TXSCK	-	-
PA5	M5	-	SSI0Tx	-	-	-	CAN0Tx	-	-	-	I2S0TXWS	-	-
PA6	L6	-	I2C1SCL	CCP1	-	-	-	CAN0Rx	-	USB0EPEN	U1CTS	-	-
PA7	M6	-	I2C1SDA	CCP4	-	-	-	CAN0Tx	CCP3	USB0PFLT	U1DCD	-	-
PB0	E12	USB0ID	CCP0	-	-	-	U1Rx	-	-	-	-	-	-
PB1	D12	USB0VBUS	CCP2	-	-	CCP1	U1Tx	-	-	-	-	-	-
PB2	A11	-	I2C0SCL	-	-	CCP3	CCP0	-	-	USB0EPEN	-	-	-
PB3	E11	-	I2C0SDA	-	-	-	-	-	-	USB0PFLT	-	-	-
PB4	A6	AIN10 C0-	-	-	-	U2Rx	CAN0Rx	-	U1Rx	EPI0S23	-	-	-
PB5	B7	AIN11 C1-	C0o	CCP5	CCP6	CCP0	CAN0Tx	CCP2	U1Tx	EPI0S22	-	-	-
PB6	A7	VREFA C0+	CCP1	CCP7	C0o	-	-	CCP5	-	-	I2S0TXSCK	-	-
PB7	A8	-	-	-	-	NMI	-	-	-	-	-	-	-
PC0	A9	-	-	-	TCK SWCLK	-	-	-	-	-	-	-	-
PC1	B9	-	-	-	TMS SWDIO	-	-	-	-	-	-	-	-
PC2	B8	-	-	-	TDI	-	-	-	-	-	-	-	-
PC3	A10	-	-	-	TDO SWO	-	-	-	-	-	-	-	-
PC4	L1	-	CCP5	-	-	-	CCP2	CCP4	-	EPI0S2	CCP1	-	-
PC5	M1	C1+	CCP1	C1o	C0o	-	CCP3	USB0EPEN	-	EPI0S3	-	-	-

Table 8-3. GPIO Pins and Alternate Functions (108BGA) *(continued)*

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	10	11
PC6	M2	C2+	CCP3	-	C2o	-	U1Rx	CCP0	USB0PFLT	EPI0S4	-	-	-
PC7	L2	C2-	CCP4	-	-	CCP0	U1Tx	USB0PFLT	C1o	EPI0S5	-	-	-
PD0	G1	AIN15	-	CAN0Rx	-	U2Rx	U1Rx	CCP6	-	I2S0RXSCK	U1CTS	-	-
PD1	G2	AIN14	-	CAN0Tx	-	U2Tx	U1Tx	CCP7	-	I2S0RXWS	U1DCD	CCP2	-
PD2	H2	AIN13	U1Rx	CCP6	-	CCP5	-	-	-	EPI0S20	-	-	-
PD3	H1	AIN12	U1Tx	CCP7	-	CCP0	-	-	-	EPI0S21	-	-	-
PD4	B5	AIN7	CCP0	CCP3	-	-	-	-	-	I2S0RXSD	U1RI	EPI0S19	-
PD5	C6	AIN6	CCP2	CCP4	-	-	-	-	-	I2S0RXMCLK	U2Rx	EPI0S28	-
PD6	A3	AIN5	-	-	-	-	-	-	-	I2S0TXSCK	U2Tx	EPI0S29	-
PD7	A2	AIN4	-	C0o	CCP1	-	-	-	-	I2S0TXWS	U1DTR	EPI0S30	-
PE0	B11	-	-	SSI1Clk	CCP3	-	-	-	-	EPI0S8	USB0PFLT	-	-
PE1	A12	-	-	SSI1Fss	-	CCP2	CCP6	-	-	EPI0S9	-	-	-
PE2	A4	AIN9	CCP4	SSI1Rx	-	-	CCP2	-	-	EPI0S24	-	-	-
PE3	B4	AIN8	CCP1	SSI1Tx	-	-	CCP7	-	-	EPI0S25	-	-	-
PE4	B2	AIN3	CCP3	CAN2Rx	-	-	U2Tx	CCP2	-	-	I2S0TXWS	-	-
PE5	B3	AIN2	CCP5	CAN2Tx	-	-	-	-	-	-	I2S0TXSD	-	-
PE6	A1	AIN1	-	C1o	-	-	-	-	-	-	U1CTS	-	-
PE7	B1	AIN0	-	C2o	-	-	-	-	-	-	U1DCD	-	-
PF0	M9	-	CAN1Rx	-	-	-	-	-	-	I2S0TXSD	U1DSR	-	-
PF1	H12	-	CAN1Tx	-	-	-	-	-	-	I2S0TXMCLK	U1RTS	CCP3	-
PF2	J11	-	LED1	-	-	-	-	-	-	-	SSI1Clk	-	-
PF3	J12	-	LED0	-	-	-	-	-	-	-	SSI1Fss	-	-
PF4	K4	-	CCP0	C0o	-	-	-	-	-	EPI0S12	SSI1Rx	-	-
PF5	K3	-	CCP2	C1o	-	-	-	-	-	EPI0S15	SSI1Tx	-	-
PG0	K1	-	U2Rx	-	I2C1SCL	-	-	-	USB0EPEN	EPI0S13	-	-	-
PG1	K2	-	U2Tx	-	I2C1SDA	-	-	-	-	EPI0S14	-	-	-
PG7	C10	-	-	-	-	-	-	-	-	CCP5	EPI0S31	-	-
PH0	C9	-	CCP6	-	-	-	-	-	-	EPI0S6	-	-	-
PH1	C8	-	CCP7	-	-	-	-	-	-	EPI0S7	-	-	-
PH2	D11	-	-	C1o	-	-	-	-	-	EPI0S1	-	-	-
PH3	D10	-	-	-	-	USB0EPEN	-	-	-	EPI0S0	-	-	-
PH4	B10	-	-	-	-	USB0PFLT	-	-	-	EPI0S10	-	-	SSI1Clk
PH5	F10	-	-	-	-	-	-	-	-	EPI0S11	-	-	SSI1Fss
PH6	G3	-	-	-	-	-	-	-	-	EPI0S26	-	-	SSI1Rx
PH7	H3	-	-	-	-	-	-	-	-	EPI0S27	-	-	SSI1Tx
PJ0	F3	-	-	-	-	-	-	-	-	EPI0S16	-	-	I2C1SCL
PJ1	B6	-	-	-	-	-	-	-	-	EPI0S17	USB0PFLT	-	I2C1SDA
PJ2	K6	-	-	-	-	-	-	-	-	EPI0S18	CCP0	-	-
PJ3	M10	-	-	-	-	-	-	-	-	EPI0S19	U1CTS	CCP6	-
PJ4	K11	-	-	-	-	-	-	-	-	EPI0S28	U1DCD	CCP4	-

Table 8-3. GPIO Pins and Alternate Functions (108BGA) *(continued)*

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	10	11
PJ5	K12	-	-	-	-	-	-	-	-	EPI0S29	U1DSR	CCP2	-
PJ6	L10	-	-	-	-	-	-	-	-	EPI0S30	U1RTS	CCP1	-
PJ7	L12	-	-	-	-	-	-	-	-	-	U1DTR	CCP0	-

a. The digital signals that are shaded gray are the power-on default values for the corresponding GPIO pin.

8.2 Functional Description

Each GPIO port is a separate hardware instantiation of the same physical block (see Figure 8-1 on page 400 and Figure 8-2 on page 401). The LM3S9U81 microcontroller contains nine ports and thus nine of these physical GPIO blocks. Note that not all pins may be implemented on every block. Some GPIO pins can function as I/O signals for the on-chip peripheral modules. For information on which GPIO pins are used for alternate hardware functions, refer to Table 22-5 on page 1151.

Figure 8-1. Digital I/O Pads

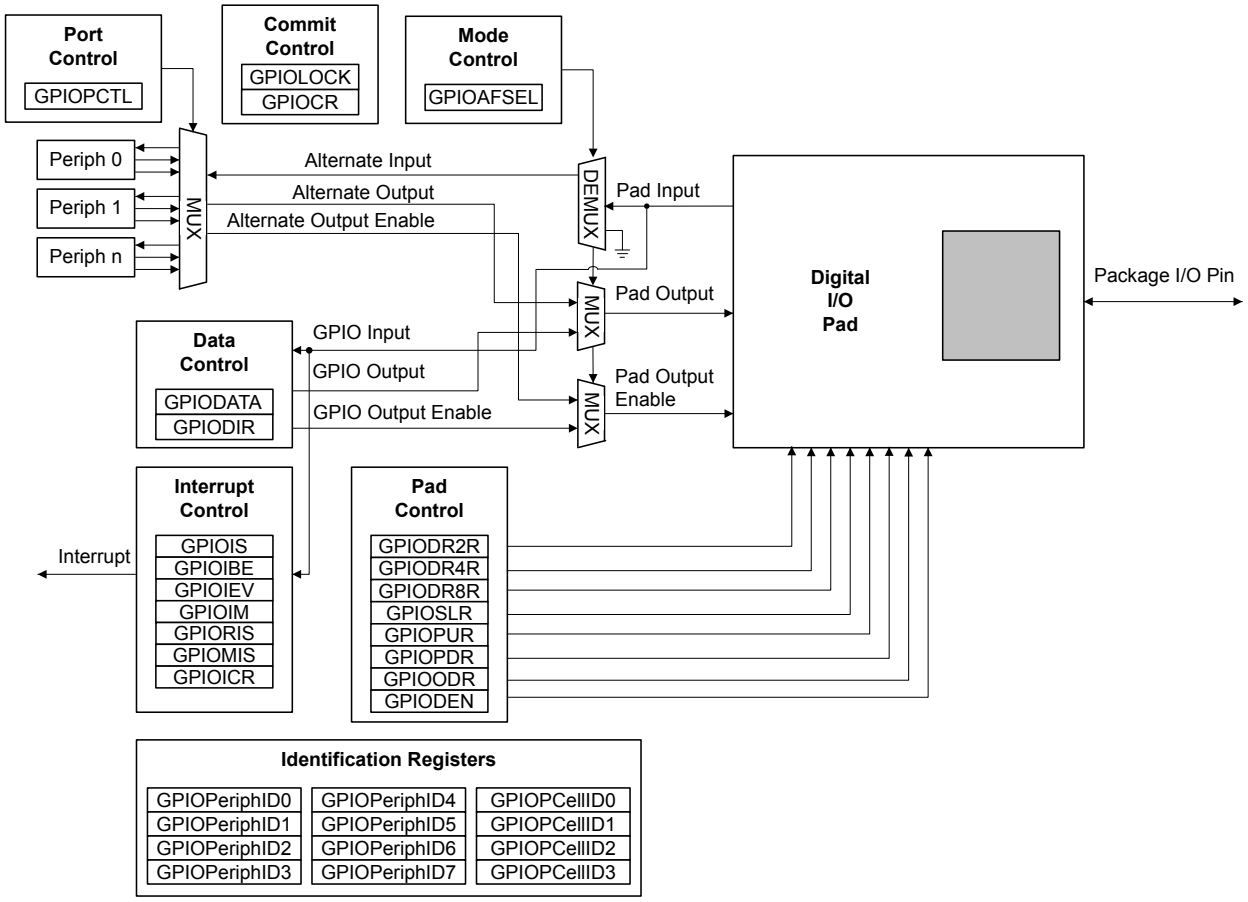
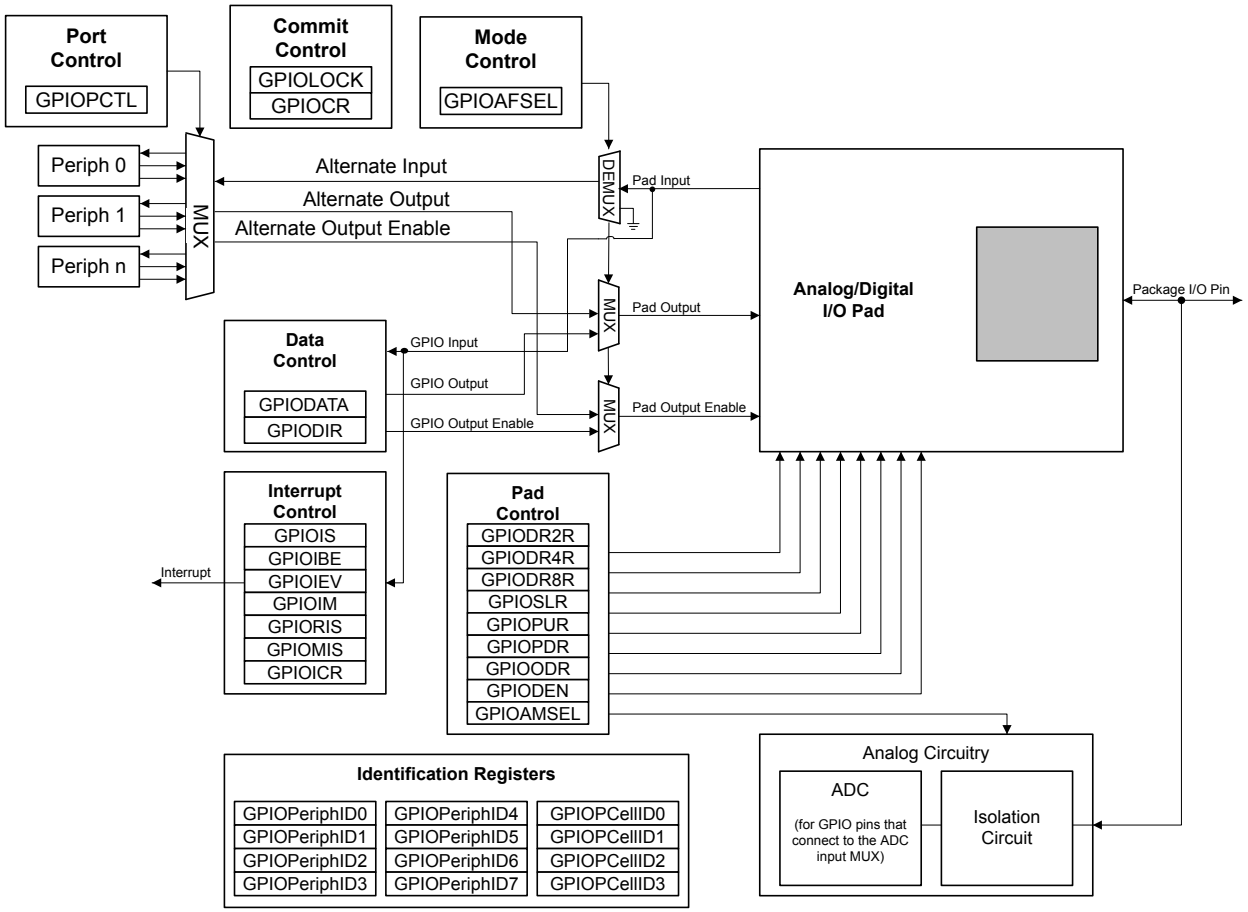


Figure 8-2. Analog/Digital I/O Pads



8.2.1 Data Control

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

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

8.2.1.1 Data Direction Operation

The **GPIO Direction (GPIODIR)** register (see page 410) is used to configure each individual pin as an input or output. When the data direction bit is cleared, the GPIO is configured as an input, and the corresponding data register bit captures and stores the value on the GPIO port. When the data direction bit is set, the GPIO is configured as an output, and the corresponding data register bit is driven out on the GPIO port.

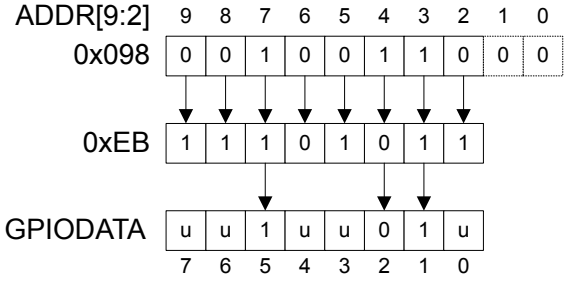
8.2.1.2 Data Register Operation

To aid in the efficiency of software, the GPIO ports allow for the modification of individual bits in the **GPIO Data (GPIODATA)** register (see page 409) by using bits [9:2] of the address bus as a mask. In this manner, software drivers can modify individual GPIO pins in a single instruction without affecting the state of the other pins. This method is more efficient than the conventional method of performing a read-modify-write operation to set or clear an individual GPIO pin. To implement this feature, the **GPIODATA** register covers 256 locations in the memory map.

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

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

Figure 8-3. GPIODATA Write Example



During a read, if the address bit associated with the data bit is set, the value is read. If the address bit associated with the data bit is cleared, the data bit is read as a zero, regardless of its actual value. For example, reading address GPIODATA + 0x0C4 yields as shown in Figure 8-4.

Figure 8-4. GPIODATA Read Example

8.2.2 Interrupt Control

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

Three registers define the edge or sense that causes interrupts:

- **GPIO Interrupt Sense (GPIOIS)** register (see page 411)

- **GPIO Interrupt Both Edges (GPIOIBE)** register (see page 412)
- **GPIO Interrupt Event (GPIOIEV)** register (see page 413)

Interrupts are enabled/disabled via the **GPIO Interrupt Mask (GPIOIM)** register (see page 414).

When an interrupt condition occurs, the state of the interrupt signal can be viewed in two locations: the **GPIO Raw Interrupt Status (GPIORIS)** and **GPIO Masked Interrupt Status (GPIOMIS)** registers (see page 415 and page 416). As the name implies, the **GPIOMIS** register only shows interrupt conditions that are allowed to be passed to the interrupt controller. The **GPIORIS** register indicates that a GPIO pin meets the conditions for an interrupt, but has not necessarily been sent to the interrupt controller.

Interrupts are cleared by writing a 1 to the appropriate bit of the **GPIO Interrupt Clear (GPIOICR)** register (see page 418).

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

8.2.2.1 ADC Trigger Source

In addition to providing GPIO functionality, PB4 can also be used as an external trigger for the ADC. If PB4 is configured as a non-masked interrupt pin (the appropriate bit of GPIOIM is set), an interrupt for Port B is generated, and an external trigger signal is sent to the ADC. If the **ADC Event Multiplexer Select (ADCEMUX)** register is configured to use the external trigger, an ADC conversion is initiated. See page 631.

If no other Port B pins are being used to generate interrupts, the **Interrupt 0-31 Set Enable (EN0)** register can disable the Port B interrupts, and the ADC interrupt can be used to read back the converted data. Otherwise, the Port B interrupt handler must ignore and clear interrupts on PB4 and wait for the ADC interrupt, or the ADC interrupt must be disabled in the **EN0** register and the Port B interrupt handler must poll the ADC registers until the conversion is completed. See page 123 for more information.

8.2.3 Mode Control

The GPIO pins can be controlled by either software or hardware. Software control is the default for most signals and corresponds to the GPIO mode, where the **GPIODATA** register is used to read or write the corresponding pins. When hardware control is enabled via the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 419), the pin state is controlled by its alternate function (that is, the peripheral).

Further pin muxing options are provided through the **GPIO Port Control (GPIOPCTL)** register which selects one of several peripheral functions for each GPIO. For information on the configuration options, refer to Table 22-5 on page 1151.

Note: If any pin is to be used as an ADC input, the appropriate bit in the **GPIOAMSEL** register must be set to disable the analog isolation circuit.

8.2.4 Commit Control

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the NMI pin (PB7) and the four JTAG/SWD pins (PC[3:0]). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 419), **GPIO Pull Up Select (GPIOPUR)** register (see page 425), **GPIO Pull-Down Select (GPIOPDR)** register (see page 427), and **GPIO Digital Enable (GPIODEN)** register (see

page 430) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 432) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 433) have been set.

8.2.5 Pad Control

The pad control registers allow software to configure the GPIO pads based on the application requirements. The pad control registers include the **GPIODR2R**, **GPIODR4R**, **GPIODR8R**, **GPIOODR**, **GPIOPUR**, **GPIOPDR**, **GPIOSLR**, and **GPIODEN** registers. These registers control drive strength, open-drain configuration, pull-up and pull-down resistors, slew-rate control and digital input enable for each GPIO.

For special high-current applications, the GPIO output buffers may be used with the following restrictions. With the GPIO pins configured as 8-mA output drivers, a total of four GPIO outputs may be used to sink current loads up to 18 mA each. At 18-mA sink current loading, the V_{OL} value is specified as 1.2 V. The high-current GPIO package pins must be selected such that there are only a maximum of two per side of the physical package or BGA pin group with the total number of high-current GPIO outputs not exceeding four for the entire package.

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

The GPIO modules may be accessed via two different memory apertures. The legacy aperture, the Advanced Peripheral Bus (APB), is backwards-compatible with previous Stellaris parts. The other aperture, the Advanced High-Performance Bus (AHB), offers the same register map but provides better back-to-back access performance than the APB bus. These apertures are mutually exclusive. The aperture enabled for a given GPIO port is controlled by the appropriate bit in the **GPIOHBCTL** register (see page 220).

To use the pins in a particular GPIO port, the clock for the port must be enabled by setting the appropriate GPIO Port bit field (`GPIOn`) in the **RCGC2** register (see page 272).

When the internal POR signal is asserted and until otherwise configured, all GPIO pins are configured to be undriven (tristate): **GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, and **GPIOPUR**=0, except for the pins shown in Table 8-1 on page 396. Table 8-4 on page 404 shows all possible configurations of the GPIO pads and the control register settings required to achieve them. Table 8-5 on page 405 shows how a rising edge interrupt is configured for pin 2 of a GPIO port.

Table 8-4. GPIO Pad Configuration Examples

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

Table 8-4. GPIO Pad Configuration Examples *(continued)*

Configuration	GPIO Register Bit Value[a]									
	AFSEL	DIR	ODR	DEN	PUR	PDR	DR2R	DR4R	DR8R	SLR
Digital Output (Timer PWM)	1	X	0	1	?	?	?	?	?	?
Digital Input/Output (SSI)	1	X	0	1	?	?	?	?	?	?
Digital Input/Output (UART)	1	X	0	1	?	?	?	?	?	?
Analog Input (Comparator)	0	0	0	0	0	0	X	X	X	X
Digital Output (Comparator)	1	X	0	1	?	?	?	?	?	?

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

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

Table 8-5. GPIO Interrupt Configuration Example

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

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

8.4 Register Map

Table 8-7 on page 406 lists the GPIO registers. Each GPIO port can be accessed through one of two bus apertures. The legacy aperture, the Advanced Peripheral Bus (APB), is backwards-compatible with previous Stellaris parts. The other aperture, the Advanced High-Performance Bus (AHB), offers the same register map but provides better back-to-back access performance than the APB bus.

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

The offset listed is a hexadecimal increment to the register's address, relative to that GPIO port's base address:

- GPIO Port A (APB): 0x4000.4000
- GPIO Port A (AHB): 0x4005.8000
- GPIO Port B (APB): 0x4000.5000
- GPIO Port B (AHB): 0x4005.9000

- GPIO Port C (APB): 0x4000.6000
- GPIO Port C (AHB): 0x4005.A000
- GPIO Port D (APB): 0x4000.7000
- GPIO Port D (AHB): 0x4005.B000
- GPIO Port E (APB): 0x4002.4000
- GPIO Port E (AHB): 0x4005.C000
- GPIO Port F (APB): 0x4002.5000
- GPIO Port F (AHB): 0x4005.D000
- GPIO Port G (APB): 0x4002.6000
- GPIO Port G (AHB): 0x4005.E000
- GPIO Port H (APB): 0x4002.7000
- GPIO Port H (AHB): 0x4005.F000
- GPIO Port J (APB): 0x4003.D000
- GPIO Port J (AHB): 0x4006.0000

Note that each GPIO module clock must be enabled before the registers can be programmed (see page 272). There must be a delay of 3 system clocks after the GPIO module clock is enabled before any GPIO module registers are accessed.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{POR}$) or asserting $\overline{RST}$ puts the pins back to their default state.

Table 8-6. GPIO Pins With Non-Zero Reset Values

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	0	0	0	0	0x1
PA[5:2]	SSI0	0	0	0	0	0x2
PB[3:2]	I^2C0	0	0	0	0	0x3
PC[3:0]	JTAG/SWD	1	1	0	1	0x1

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

The default reset value for the **GPIOCR** register is 0x0000.00FF for all GPIO pins, with the exception of the NMI pin and the four JTAG/SWD pins (PB7 and PC[3:0]). To ensure that the JTAG port is not accidentally programmed as GPIO pins, the PC[3:0] pins default to non-committable. Similarly, to ensure that the NMI pin is not accidentally programmed as a GPIO pin, the PB7 pin defaults to non-committable. Because of this, the default reset value of **GPIOCR** for GPIO Port B is 0x0000.007F while the default reset value of **GPIOCR** for Port C is 0x0000.00F0.

Table 8-7. GPIO Register Map

Offset	Name	Type	Reset	Description	See page
0x000	GPIODATA	R/W	0x0000.0000	GPIO Data	409
0x400	GPIODIR	R/W	0x0000.0000	GPIO Direction	410
0x404	GPIOIS	R/W	0x0000.0000	GPIO Interrupt Sense	411

Table 8-7. GPIO Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x408	GPIOIBE	R/W	0x0000.0000	GPIO Interrupt Both Edges	412
0x40C	GPIOIEV	R/W	0x0000.0000	GPIO Interrupt Event	413
0x410	GPIOIM	R/W	0x0000.0000	GPIO Interrupt Mask	414
0x414	GPIORIS	RO	0x0000.0000	GPIO Raw Interrupt Status	415
0x418	GPIOMIS	RO	0x0000.0000	GPIO Masked Interrupt Status	416
0x41C	GPIOICR	W1C	0x0000.0000	GPIO Interrupt Clear	418
0x420	GPIOAFSEL	R/W	-	GPIO Alternate Function Select	419
0x500	GPIODR2R	R/W	0x0000.00FF	GPIO 2-mA Drive Select	421
0x504	GPIODR4R	R/W	0x0000.0000	GPIO 4-mA Drive Select	422
0x508	GPIODR8R	R/W	0x0000.0000	GPIO 8-mA Drive Select	423
0x50C	GPIOODR	R/W	0x0000.0000	GPIO Open Drain Select	424
0x510	GPIOPUR	R/W	-	GPIO Pull-Up Select	425
0x514	GPIOPDR	R/W	0x0000.0000	GPIO Pull-Down Select	427
0x518	GPIOSLR	R/W	0x0000.0000	GPIO Slew Rate Control Select	429
0x51C	GPIODEN	R/W	-	GPIO Digital Enable	430
0x520	GPIOLOCK	R/W	0x0000.0001	GPIO Lock	432
0x524	GPIOCR	-	-	GPIO Commit	433
0x528	GPIOAMSEL	R/W	0x0000.0000	GPIO Analog Mode Select	435
0x52C	GPIOPCTL	R/W	-	GPIO Port Control	437
0xFD0	GPIOPeriphID4	RO	0x0000.0000	GPIO Peripheral Identification 4	439
0xFD4	GPIOPeriphID5	RO	0x0000.0000	GPIO Peripheral Identification 5	440
0xFD8	GPIOPeriphID6	RO	0x0000.0000	GPIO Peripheral Identification 6	441
0xFDC	GPIOPeriphID7	RO	0x0000.0000	GPIO Peripheral Identification 7	442
0xFE0	GPIOPeriphID0	RO	0x0000.0061	GPIO Peripheral Identification 0	443
0xFE4	GPIOPeriphID1	RO	0x0000.0000	GPIO Peripheral Identification 1	444
0xFE8	GPIOPeriphID2	RO	0x0000.0018	GPIO Peripheral Identification 2	445
0xFEC	GPIOPeriphID3	RO	0x0000.0001	GPIO Peripheral Identification 3	446
0xFF0	GPIOPCellID0	RO	0x0000.000D	GPIO PrimeCell Identification 0	447
0xFF4	GPIOPCellID1	RO	0x0000.00F0	GPIO PrimeCell Identification 1	448
0xFF8	GPIOPCellID2	RO	0x0000.0005	GPIO PrimeCell Identification 2	449
0xFFC	GPIOPCellID3	RO	0x0000.00B1	GPIO PrimeCell Identification 3	450

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 410).

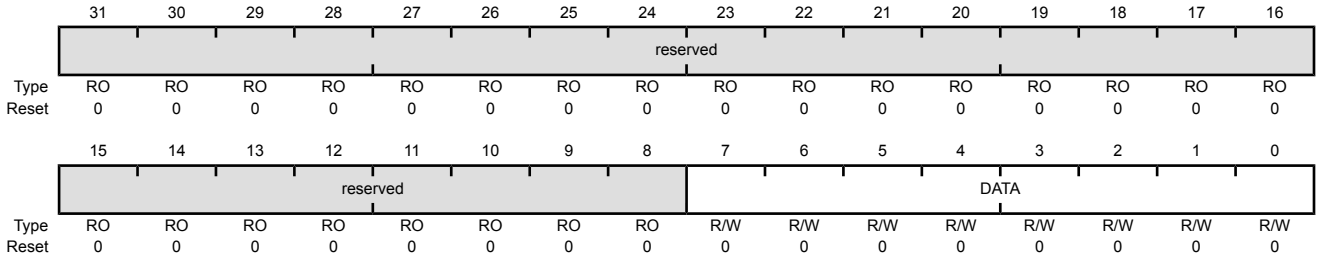
In order to write to **GPIODATA**, the corresponding bits in the mask, resulting from the address bus bits [9:2], must be set. Otherwise, the bit values remain unchanged by the write.

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

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

GPIO Data (GPIODATA)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x000
Type R/W, reset 0x0000.0000



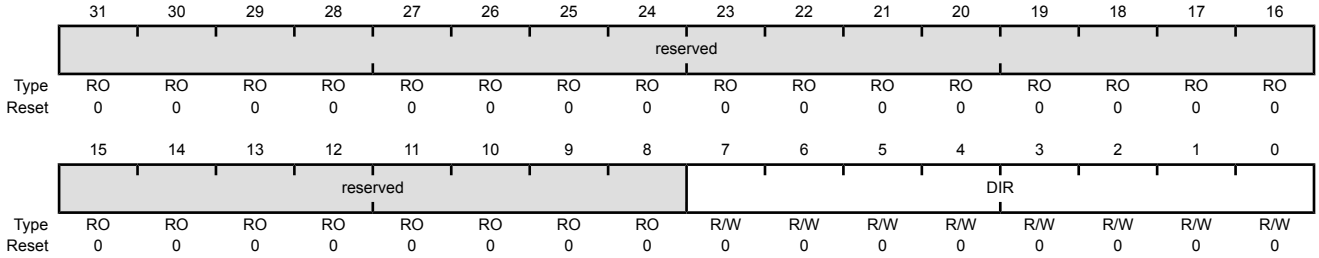
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x00	GPIO Data This register is virtually mapped to 256 locations in the address space. To facilitate the reading and writing of data to these registers by independent drivers, the data read from and written to the registers are masked by the eight address lines [9:2]. Reads from this register return its current state. Writes to this register only affect bits that are not masked by ADDR[9:2] and are configured as outputs. See “Data Register Operation” on page 402 for examples of reads and writes.

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

The **GPIODIR** register is the data direction register. Setting a bit in the **GPIODIR** register configures the corresponding pin to be an output, while clearing a bit configures the corresponding pin to be an input. All bits are cleared by a reset, meaning all GPIO pins are inputs by default.

GPIO Direction (GPIODIR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x400
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DIR	R/W	0x00	GPIO Data Direction

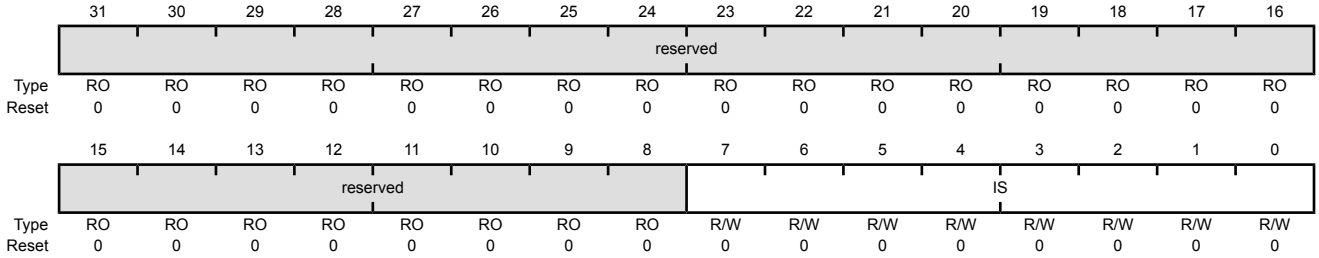
Value	Description
0	Corresponding pin is an input.
1	Corresponding pins is an output.

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

The **GPIOIS** register is the interrupt sense register. Setting a bit in the **GPIOIS** register configures the corresponding pin to detect levels, while clearing a bit configures the corresponding pin to detect edges. All bits are cleared by a reset.

GPIO Interrupt Sense (GPIOIS)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x404
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IS	R/W	0x00	GPIO Interrupt Sense

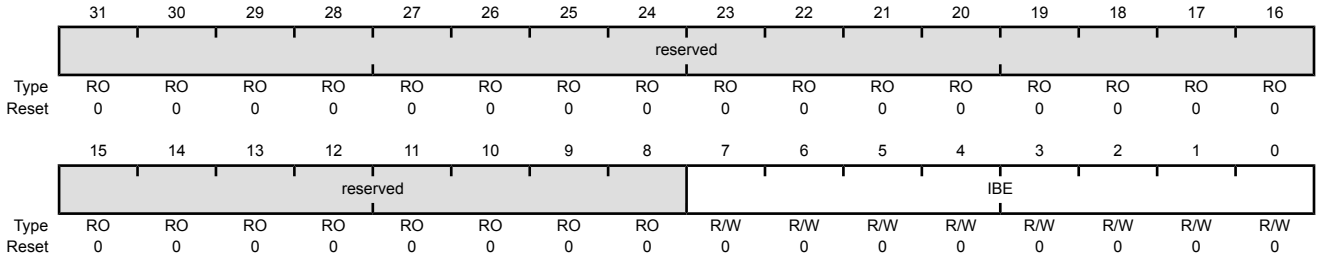
Value	Description
0	The edge on the corresponding pin is detected (edge-sensitive).
1	The level on the corresponding pin is detected (level-sensitive).

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

The **GPIOIBE** register allows both edges to cause interrupts. When the corresponding bit in the **GPIO Interrupt Sense (GPIOIS)** register (see page 411) is set to detect edges, setting a bit in the **GPIOIBE** register configures the corresponding pin to detect both rising and falling edges, regardless of the corresponding bit in the **GPIO Interrupt Event (GPIOIEV)** register (see page 413). Clearing a bit configures the pin to be controlled by the **GPIOIEV** register. All bits are cleared by a reset.

GPIO Interrupt Both Edges (GPIOIBE)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x408
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IBE	R/W	0x00	GPIO Interrupt Both Edges

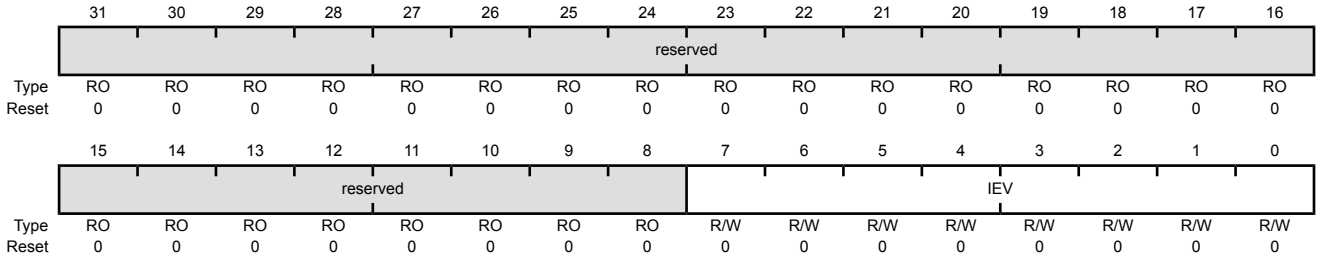
Value	Description
0	Interrupt generation is controlled by the **GPIO Interrupt Event (GPIOIEV)** register (see page 413).
1	Both edges on the corresponding pin trigger an interrupt.

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

The **GPIOIEV** register is the interrupt event register. Setting a bit in the **GPIOIEV** register configures the corresponding pin to detect rising edges or high levels, depending on the corresponding bit value in the **GPIO Interrupt Sense (GPIOIS)** register (see page 411). Clearing a bit configures the pin to detect falling edges or low levels, depending on the corresponding bit value in the **GPIOIS** register. All bits are cleared by a reset.

GPIO Interrupt Event (GPIOIEV)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x40C
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IEV	R/W	0x00	GPIO Interrupt Event

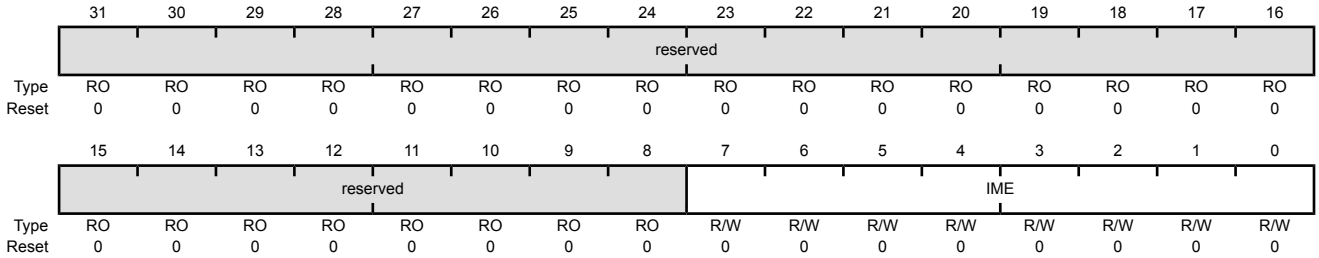
Value	Description
0	A falling edge or a Low level on the corresponding pin triggers an interrupt.
1	A rising edge or a High level on the corresponding pin triggers an interrupt.

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

The **GPIOIM** register is the interrupt mask register. Setting a bit in the **GPIOIM** register allows interrupts that are generated by the corresponding pin to be sent to the interrupt controller on the combined interrupt signal. Clearing a bit prevents an interrupt on the corresponding pin from being sent to the interrupt controller. All bits are cleared by a reset.

GPIO Interrupt Mask (GPIOIM)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x410
Type R/W, reset 0x0000.0000



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

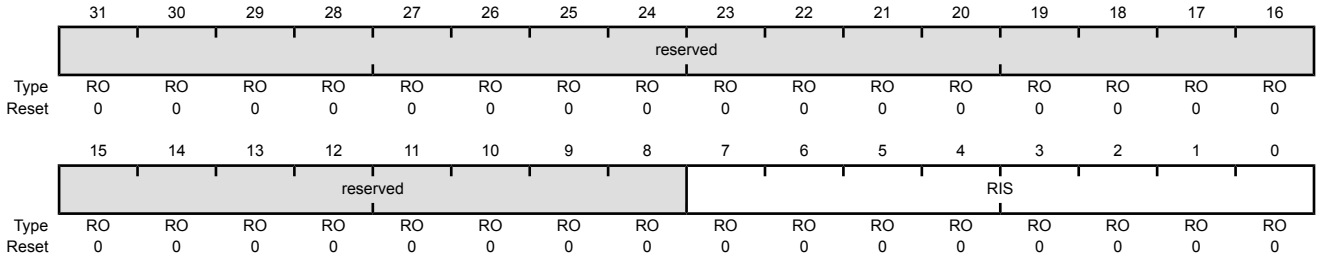
Value	Description
0	The interrupt from the corresponding pin is masked.
1	The interrupt from the corresponding pin is sent to the interrupt controller.

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

The **GPIORIS** register is the raw interrupt status register. A bit in this register is set when an interrupt condition occurs on the corresponding GPIO pin. If the corresponding bit in the **GPIO Interrupt Mask (GPIOIM)** register (see page 414) is set, the interrupt is sent to the interrupt controller. Bits read as zero indicate that corresponding input pins have not initiated an interrupt. A bit in this register can be cleared by writing a 1 to the corresponding bit in the **GPIO Interrupt Clear (GPIOICR)** register.

GPIO Raw Interrupt Status (GPIORIS)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x414
Type RO, reset 0x0000.0000



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

Value	Description
1	An interrupt condition has occurred on the corresponding pin.
0	An interrupt condition has not occurred on the corresponding pin.

A bit is cleared by writing a 1 to the corresponding bit in the **GPIOICR** register.

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

The **GPIOMIS** register is the masked interrupt status register. If a bit is set in this register, the corresponding interrupt has triggered an interrupt to the interrupt controller. If a bit is clear, either no interrupt has been generated, or the interrupt is masked.

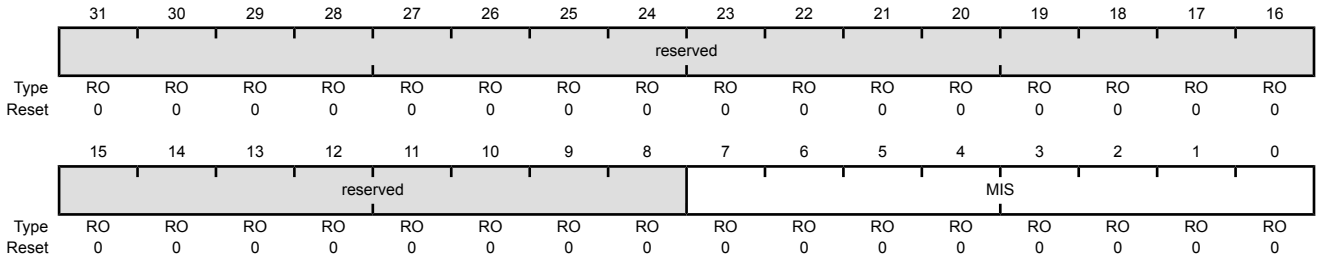
In addition to providing GPIO functionality, PB4 can also be used as an external trigger for the ADC. If PB4 is configured as a non-masked interrupt pin (the appropriate bit of GPIOIM is set), an interrupt for Port B is generated, and an external trigger signal is sent to the ADC. If the **ADC Event Multiplexer Select (ADCEMUX)** register is configured to use the external trigger, an ADC conversion is initiated. See page 631.

If no other Port B pins are being used to generate interrupts, the **Interrupt 0-31 Set Enable (EN0)** register can disable the Port B interrupts, and the ADC interrupt can be used to read back the converted data. Otherwise, the Port B interrupt handler must ignore and clear interrupts on PB4 and wait for the ADC interrupt, or the ADC interrupt must be disabled in the **EN0** register and the Port B interrupt handler must poll the ADC registers until the conversion is completed. See page 123 for more information.

GPIOMIS is the state of the interrupt after masking.

GPIO Masked Interrupt Status (GPIOMIS)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x418
Type RO, reset 0x0000.0000



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

Bit/Field	Name	Type	Reset	Description
7:0	MIS	RO	0x00	GPIO Masked Interrupt Status

Value	Description
1	An interrupt condition on the corresponding pin has triggered an interrupt to the interrupt controller.
0	An interrupt condition on the corresponding pin is masked or has not occurred.

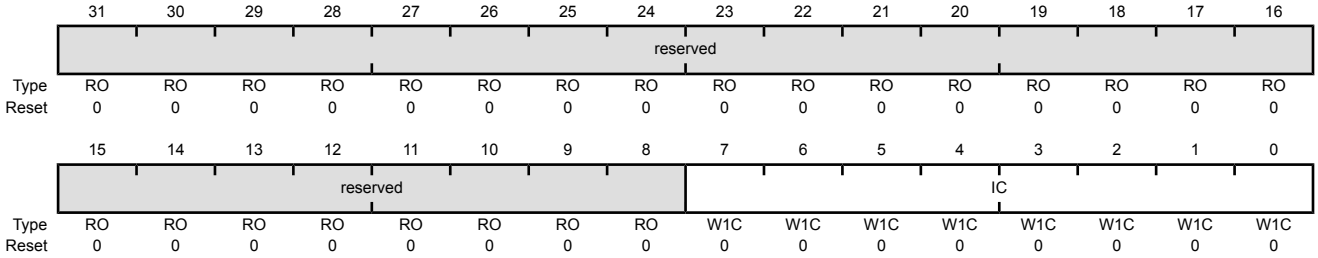
A bit is cleared by writing a 1 to the corresponding bit in the **GPIOICR** register.

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

The **GPIOICR** register is the interrupt clear register. Writing a 1 to a bit in this register clears the corresponding interrupt bit in the **GPIORIS** and **GPIOMIS** registers. Writing a 0 has no effect.

GPIO Interrupt Clear (GPIOICR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x41C
Type W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IC	W1C	0x00	GPIO Interrupt Clear Value Description 1 The corresponding interrupt is cleared. 0 The corresponding interrupt is unaffected.

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

The **GPIOAFSEL** register is the mode control select register. If a bit is clear, the pin is used as a GPIO and is controlled by the GPIO registers. Setting a bit in this register configures the corresponding GPIO line to be controlled by an associated peripheral. Several possible peripheral functions are multiplexed on each GPIO. The **GPIO Port Control (GPIOPCTL)** register is used to select one of the possible functions. Table 22-5 on page 1151 details which functions are muxed on each GPIO pin. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in the table below.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{POR}$) or asserting $\overline{RST}$ puts the pins back to their default state.

Table 8-8. GPIO Pins With Non-Zero Reset Values

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	0	0	0	0	0x1
PA[5:2]	SSI0	0	0	0	0	0x2
PB[3:2]	I^2C0	0	0	0	0	0x3
PC[3:0]	JTAG/SWD	1	1	0	1	0x1

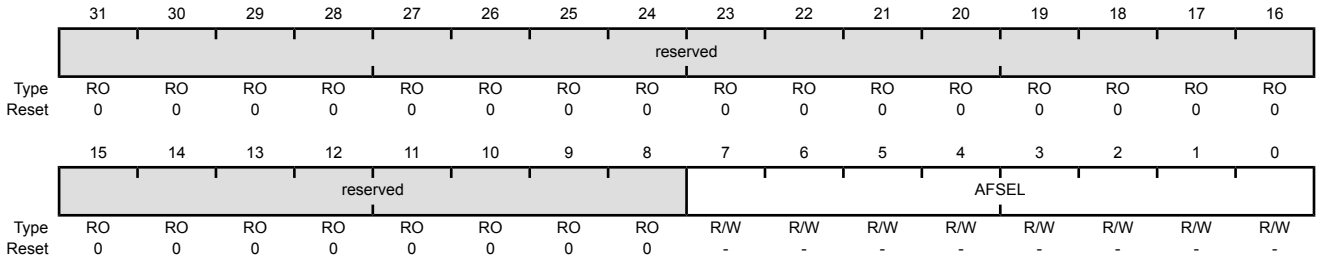
Caution – It is possible to create a software sequence that prevents the debugger from connecting to the Stellaris microcontroller. If the program code loaded into flash immediately changes the JTAG pins to their GPIO functionality, the debugger may not have enough time to connect and halt the controller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

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the `NMI` pin (`PB7`) and the four JTAG/SWD pins (`PC[3:0]`). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 419), **GPIO Pull Up Select (GPIOPUR)** register (see page 425), **GPIO Pull-Down Select (GPIOPDR)** register (see page 427), and **GPIO Digital Enable (GPIODEN)** register (see page 430) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 432) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 433) have been set.

When using the I^2C module, in addition to setting the **GPIOAFSEL** register bits for the I^2C clock and data pins, the data pins should be set to open drain using the **GPIO Open Drain Select (GPIOODR)** register (see examples in “Initialization and Configuration” on page 404).

GPIO Alternate Function Select (GPIOAFSEL)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x420
Type R/W, reset -



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	AFSEL	R/W	-	GPIO Alternate Function Select

Value	Description
0	The associated pin functions as a GPIO and is controlled by the GPIO registers.
1	The associated pin functions as a peripheral signal and is controlled by the alternate hardware function.

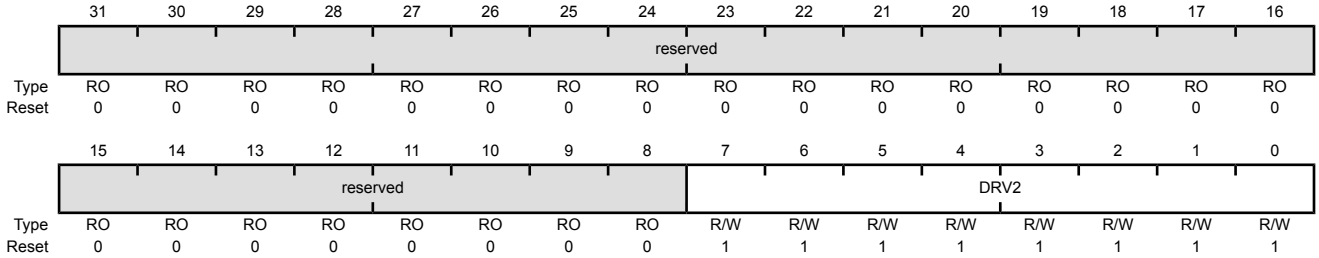
The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 8-1 on page 396.

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

The **GPIODR2R** register is the 2-mA drive control register. Each GPIO signal in the port can be individually configured without affecting the other pads. When setting the DRV2 bit for a GPIO signal, the corresponding DRV4 bit in the **GPIODR4R** register and DRV8 bit in the **GPIODR8R** register are automatically cleared by hardware. By default, all GPIO pins have 2-mA drive.

GPIO 2-mA Drive Select (GPIODR2R)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x500
Type R/W, reset 0x0000.00FF



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV2	R/W	0xFF	Output Pad 2-mA Drive Enable

Value	Description
1	The corresponding GPIO pin has 2-mA drive.
0	The drive for the corresponding GPIO pin is controlled by the **GPIODR4R** or **GPIODR8R** register.

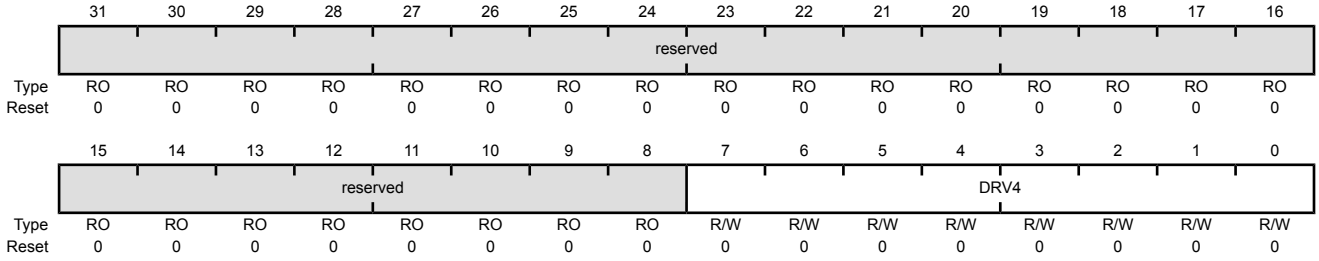
Setting a bit in either the **GPIODR4** register or the **GPIODR8** register clears the corresponding 2-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.

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

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

GPIO 4-mA Drive Select (GPIODR4R)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x504
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV4	R/W	0x00	Output Pad 4-mA Drive Enable

Value	Description
1	The corresponding GPIO pin has 4-mA drive.
0	The drive for the corresponding GPIO pin is controlled by the **GPIODR2R** or **GPIODR8R** register.

Setting a bit in either the **GPIODR2** register or the **GPIODR8** register clears the corresponding 4-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.

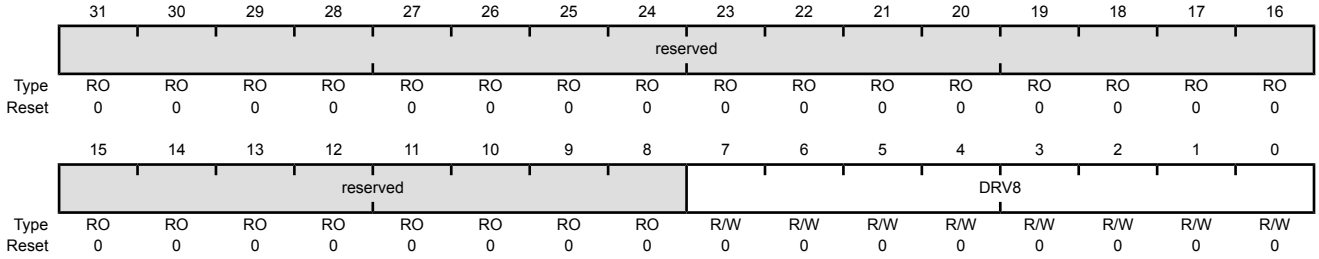
Register 13: GPIO 8-mA Drive Select (GPIODR8R), offset 0x508

The **GPIODR8R** register is the 8-mA drive control register. Each GPIO signal in the port can be individually configured without affecting the other pads. When setting the DRV8 bit for a GPIO signal, the corresponding DRV2 bit in the **GPIODR2R** register and DRV4 bit in the **GPIODR4R** register are automatically cleared by hardware. The 8-mA setting is also used for high-current operation.

Note: There is no configuration difference between 8-mA and high-current operation. The additional current capacity results from a shift in the V_{OH}/V_{OL} levels. See “Recommended Operating Conditions” on page 1192 for further information.

GPIO 8-mA Drive Select (GPIODR8R)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x508
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV8	R/W	0x00	Output Pad 8-mA Drive Enable

Value	Description
1	The corresponding GPIO pin has 8-mA drive.
0	The drive for the corresponding GPIO pin is controlled by the **GPIODR2R** or **GPIODR4R** register.

Setting a bit in either the **GPIODR2** register or the **GPIODR4** register clears the corresponding 8-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.

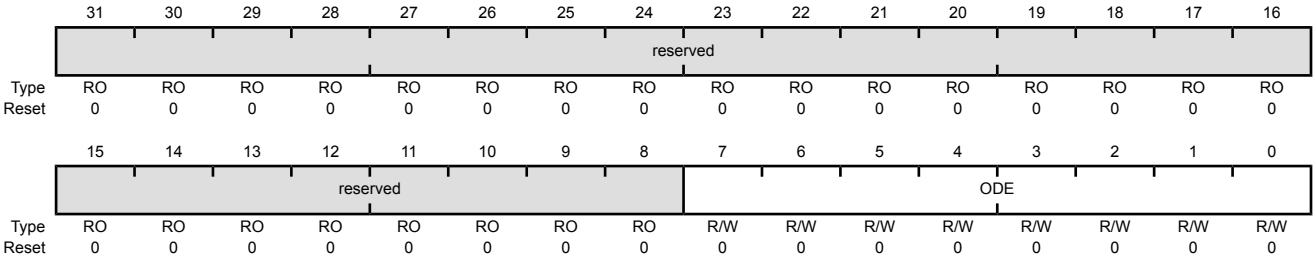
Register 14: GPIO Open Drain Select (GPIOODR), offset 0x50C

The **GPIOODR** register is the open drain control register. Setting a bit in this register enables the open-drain configuration of the corresponding GPIO pad. When open-drain mode is enabled, the corresponding bit should also be set in the **GPIO Digital Enable (GPIODEN)** register (see page 430). Corresponding bits in the drive strength and slew rate control registers (**GPIODR2R**, **GPIODR4R**, **GPIODR8R**, and **GPIOSLR**) can be set to achieve the desired rise and fall times. The GPIO acts as an input if the corresponding bit in the **GPIODIR** register is cleared. If open drain is selected while the GPIO is configured as an input, the GPIO will remain an input and the open-drain selection has no effect until the GPIO is changed to an output.

When using the I^2C module, in addition to configuring the pin to open drain, the **GPIO Alternate Function Select (GPIOAFSEL)** register bits for the I^2C clock and data pins should be set (see examples in “Initialization and Configuration” on page 404).

GPIO Open Drain Select (GPIOODR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x50C
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ODE	R/W	0x00	Output Pad Open Drain Enable Value Description 1 The corresponding pin is configured as open drain. 0 The corresponding pin is not configured as open drain.

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

The **GPIOPUR** register is the pull-up control register. When a bit is set, a weak pull-up resistor on the corresponding GPIO signal is enabled. Setting a bit in **GPIOPUR** automatically clears the corresponding bit in the **GPIO Pull-Down Select (GPIOPDR)** register (see page 427). Write access to this register is protected with the **GPIOCR** register. Bits in **GPIOCR** that are cleared prevent writes to the equivalent bit in this register.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ puts the pins back to their default state.

Table 8-9. GPIO Pins With Non-Zero Reset Values

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	0	0	0	0	0x1
PA[5:2]	SSI0	0	0	0	0	0x2
PB[3:2]	I^2C0	0	0	0	0	0x3
PC[3:0]	JTAG/SWD	1	1	0	1	0x1

Note: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the `NMI` pin (`PB7`) and the four JTAG/SWD pins (`PC[3:0]`). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 419), **GPIO Pull Up Select (GPIOPUR)** register (see page 425), **GPIO Pull-Down Select (GPIOPDR)** register (see page 427), and **GPIO Digital Enable (GPIODEN)** register (see page 430) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 432) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 433) have been set.

GPIO Pull-Up Select (GPIOPUR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x510
Type R/W, reset -

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

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PUE	R/W	-	Pad Weak Pull-Up Enable Value Description 1 The corresponding pin has a weak pull-up resistor. 0 The corresponding pin is not affected. Setting a bit in the **GPIOPDR** register clears the corresponding bit in the **GPIOPUR** register. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 8-1 on page 396.

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

The **GPIOPDR** register is the pull-down control register. When a bit is set, a weak pull-down resistor on the corresponding GPIO signal is enabled. Setting a bit in **GPIOPDR** automatically clears the corresponding bit in the **GPIO Pull-Up Select (GPIOPUR)** register (see page 425).

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{POR}$) or asserting $\overline{RST}$ puts the pins back to their default state.

Table 8-10. GPIO Pins With Non-Zero Reset Values

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	0	0	0	0	0x1
PA[5:2]	SSI0	0	0	0	0	0x2
PB[3:2]	I^2C0	0	0	0	0	0x3
PC[3:0]	JTAG/SWD	1	1	0	1	0x1

Note: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the `NMI` pin (`PB7`) and the four JTAG/SWD pins (`PC[3:0]`). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 419), **GPIO Pull Up Select (GPIOPUR)** register (see page 425), **GPIO Pull-Down Select (GPIOPDR)** register (see page 427), and **GPIO Digital Enable (GPIODEN)** register (see page 430) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 432) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 433) have been set.

GPIO Pull-Down Select (GPIOPDR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x514
Type R/W, reset 0x0000.0000

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

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

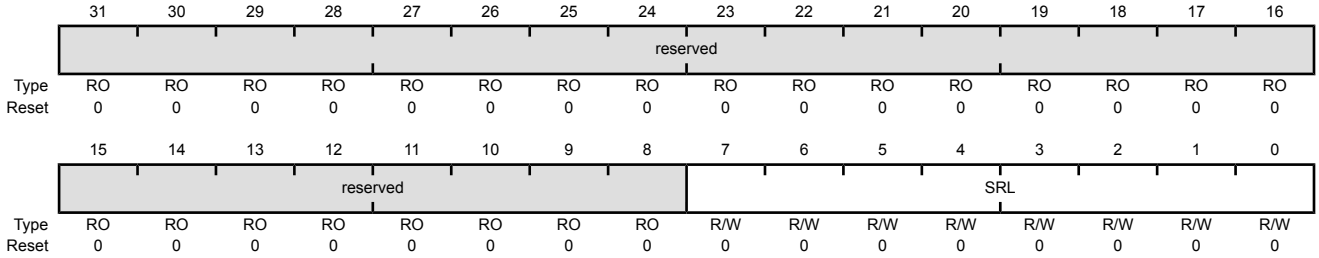
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PDE	R/W	0x00	Pad Weak Pull-Down Enable Value Description 1 The corresponding pin has a weak pull-down resistor. 0 The corresponding pin is not affected. Setting a bit in the **GPIOPUR** register clears the corresponding bit in the **GPIOPDR** register. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.

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

The **GPIOSLR** register is the slew rate control register. Slew rate control is only available when using the 8-mA drive strength option via the **GPIO 8-mA Drive Select (GPIODR8R)** register (see page 423).

GPIO Slew Rate Control Select (GPIOSLR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x518
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	SRL	R/W	0x00	Slew Rate Limit Enable (8-mA drive only) Value Description 1 Slew rate control is enabled for the corresponding pin. 0 Slew rate control is disabled for the corresponding pin.

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

Note: Pins configured as digital inputs are Schmitt-triggered.

The **GPIODEN** register is the digital enable register. By default, all GPIO signals except those listed below are configured out of reset to be undriven (tristate). Their digital function is disabled; they do not drive a logic value on the pin and they do not allow the pin voltage into the GPIO receiver. To use the pin as a digital input or output (either GPIO or alternate function), the corresponding GPIODEN bit must be set.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{POR}$) or asserting $\overline{RST}$ puts the pins back to their default state.

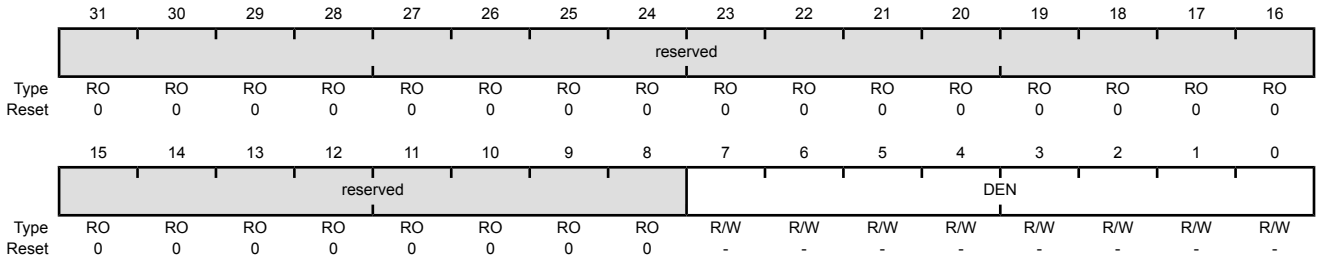
Table 8-11. GPIO Pins With Non-Zero Reset Values

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	0	0	0	0	0x1
PA[5:2]	SSI0	0	0	0	0	0x2
PB[3:2]	I^2C0	0	0	0	0	0x3
PC[3:0]	JTAG/SWD	1	1	0	1	0x1

Note: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the NMI pin (PB7) and the four JTAG/SWD pins (PC[3:0]). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 419), **GPIO Pull Up Select (GPIOPUR)** register (see page 425), **GPIO Pull-Down Select (GPIOPDR)** register (see page 427), and **GPIO Digital Enable (GPIODEN)** register (see page 430) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 432) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 433) have been set.

GPIO Digital Enable (GPIODEN)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x51C
Type R/W, reset -



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DEN	R/W	-	Digital Enable

Value	Description
0	The digital functions for the corresponding pin are disabled.
1	The digital functions for the corresponding pin are enabled.

The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 8-1 on page 396.

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

The **GPIOLOCK** register enables write access to the **GPIOCR** register (see page 433). Writing 0x4C4F.434B to the **GPIOLOCK** register unlocks the **GPIOCR** register. Writing any other value to the **GPIOLOCK** register re-enables the locked state. Reading the **GPIOLOCK** register returns the lock status rather than the 32-bit value that was previously written. Therefore, when write accesses are disabled, or locked, reading the **GPIOLOCK** register returns 0x0000.0001. When write accesses are enabled, or unlocked, reading the **GPIOLOCK** register returns 0x0000.0000.

GPIO Lock (GPIOLOCK)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x520
Type R/W, reset 0x0000.0001

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

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

Bit/Field	Name	Type	Reset	Description
31:0	LOCK	R/W	0x0000.0001	GPIO Lock A write of the value 0x4C4F.434B unlocks the **GPIO Commit (GPIOCR)** register for write access.A write of any other value or a write to the **GPIOCR** register reapplies the lock, preventing any register updates. A read of this register returns the following values: Value Description 0x1 The **GPIOCR** register is locked and may not be modified. 0x0 The **GPIOCR** register is unlocked and may be modified.

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

The **GPIOCR** register is the commit register. The value of the **GPIOCR** register determines which bits of the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, and **GPIODEN** registers are committed when a write to these registers is performed. If a bit in the **GPIOCR** register is cleared, the data being written to the corresponding bit in the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** registers cannot be committed and retains its previous value. If a bit in the **GPIOCR** register is set, the data being written to the corresponding bit of the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** registers is committed to the register and reflects the new value.

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

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

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

GPIO Commit (GPIOCR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x524
Type -, reset -

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

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CR	-	-	GPIO Commit Value Description 1 The corresponding **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** bits can be written. 0 The corresponding **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** bits cannot be written. **Note:** The default register type for the **GPIOCR** register is RO for all GPIO pins with the exception of the NMI pin and the four JTAG/SWD pins (PB7 and PC[3:0]). These five pins are the only GPIOs that are protected by the **GPIOCR** register. Because of this, the register type for GPIO Port B7 and GPIO Port C[3:0] is R/W. The default reset value for the **GPIOCR** register is 0x0000.00FF for all GPIO pins, with the exception of the NMI pin and the four JTAG/SWD pins (PB7 and PC[3:0]). To ensure that the JTAG port is not accidentally programmed as GPIO pins, the PC[3:0] pins default to non-committable. Similarly, to ensure that the NMI pin is not accidentally programmed as a GPIO pin, the PB7 pin defaults to non-committable. Because of this, the default reset value of **GPIOCR** for GPIO Port B is 0x0000.007F while the default reset value of **GPIOCR** for Port C is 0x0000.00F0.

Register 21: GPIO Analog Mode Select (GPIOAMSEL), offset 0x528

Important: This register is only valid for ports D and E; the corresponding base addresses for the remaining ports are not valid.

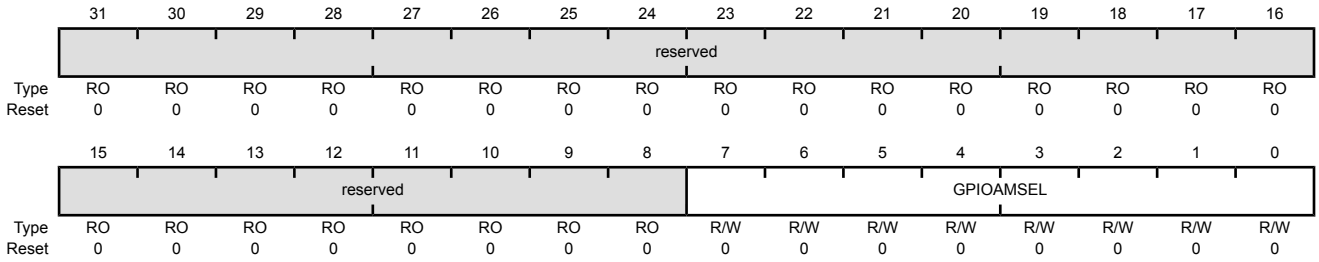
If any pin is to be used as an ADC input, the appropriate bit in **GPIOAMSEL** must be set to disable the analog isolation circuit.

The **GPIOAMSEL** register controls isolation circuits to the analog side of a unified I/O pad. Because the GPIOs may be driven by a 5-V source and affect analog operation, analog circuitry requires isolation from the pins when they are not used in their analog function.

Each bit of this register controls the isolation circuitry for the corresponding GPIO signal. For information on which GPIO pins can be used for ADC functions, refer to Table 22-5 on page 1151.

GPIO Analog Mode Select (GPIOAMSEL)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x528
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
7:0	GPIOAMSEL	R/W	0x00	GPIO Analog Mode Select

Value	Description
1	The analog function of the pin is enabled, the isolation is disabled, and the pin is capable of analog functions.
0	The analog function of the pin is disabled, the isolation is enabled, and the pin is capable of digital functions as specified by the other GPIO configuration registers.

Note: This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad.

The reset state of this register is 0 for all signals.

Register 22: GPIO Port Control (GPIOPCTL), offset 0x52C

The **GPIOPCTL** register is used in conjunction with the **GPIOAFSEL** register and selects the specific peripheral signal for each GPIO pin when using the alternate function mode. Most bits in the **GPIOAFSEL** register are cleared on reset, therefore most GPIO pins are configured as GPIOs by default. When a bit is set in the **GPIOAFSEL** register, the corresponding GPIO signal is controlled by an associated peripheral. The **GPIOPCTL** register selects one out of a set of peripheral functions for each GPIO, providing additional flexibility in signal definition. For information on the defined encodings for the bit fields in this register, refer to Table 22-5 on page 1151. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in the table below.

Note: If the same signal is assigned to two different GPIO port pins, the signal is assigned to the port with the lowest letter and the assignment to the higher letter port is ignored.

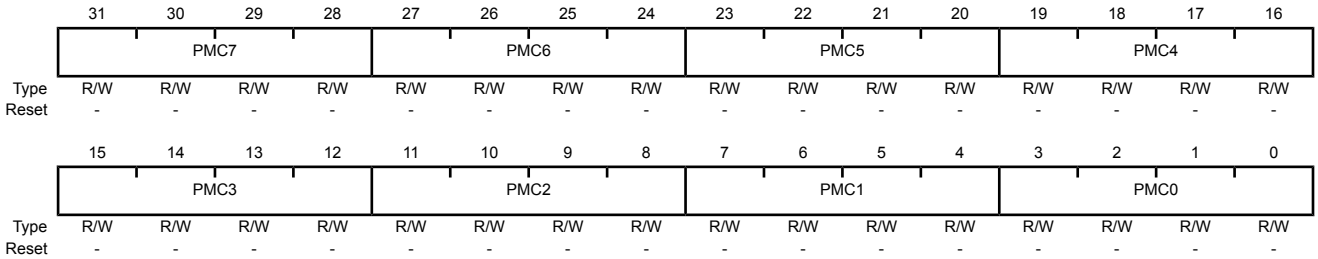
Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{POR}$) or asserting $\overline{RST}$ puts the pins back to their default state.

Table 8-12. GPIO Pins With Non-Zero Reset Values

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	0	0	0	0	0x1
PA[5:2]	SSI0	0	0	0	0	0x2
PB[3:2]	I^2C0	0	0	0	0	0x3
PC[3:0]	JTAG/SWD	1	1	0	1	0x1

GPIO Port Control (GPIOPCTL)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0x52C
Type R/W, reset -



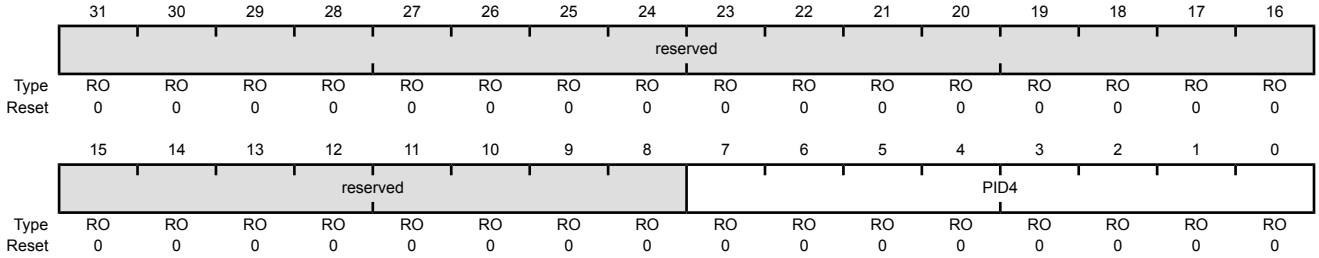
Bit/Field	Name	Type	Reset	Description
31:28	PMC7	R/W	-	Port Mux Control 7 This field controls the configuration for GPIO pin 7.
27:24	PMC6	R/W	-	Port Mux Control 6 This field controls the configuration for GPIO pin 6.
23:20	PMC5	R/W	-	Port Mux Control 5 This field controls the configuration for GPIO pin 5.
19:16	PMC4	R/W	-	Port Mux Control 4 This field controls the configuration for GPIO pin 4.
15:12	PMC3	R/W	-	Port Mux Control 3 This field controls the configuration for GPIO pin 3.
11:8	PMC2	R/W	-	Port Mux Control 2 This field controls the configuration for GPIO pin 2.
7:4	PMC1	R/W	-	Port Mux Control 1 This field controls the configuration for GPIO pin 1.
3:0	PMC0	R/W	-	Port Mux Control 0 This field controls the configuration for GPIO pin 0.

Register 23: GPIO Peripheral Identification 4 (GPIOPeriphID4), offset 0xFD0

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

GPIO Peripheral Identification 4 (GPIOPeriphID4)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0xFD0
Type RO, reset 0x0000.0000



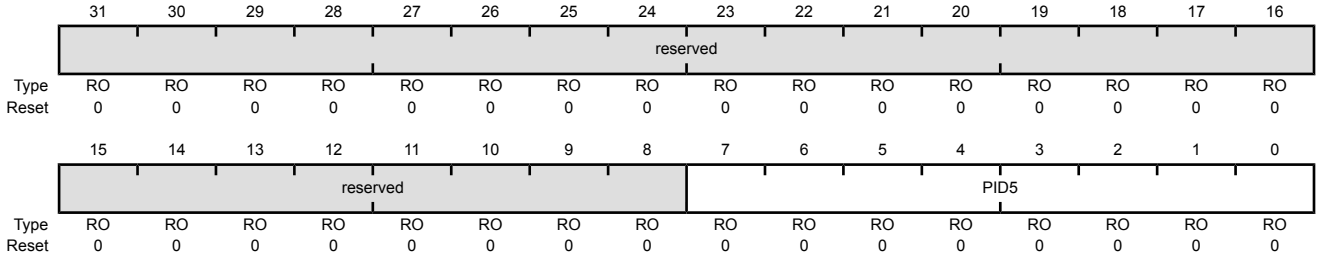
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	GPIO Peripheral ID Register [7:0]

Register 24: GPIO Peripheral Identification 5 (GPIOPeriphID5), offset 0xFD4

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

GPIO Peripheral Identification 5 (GPIOPeriphID5)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0xFD4
Type RO, reset 0x0000.0000



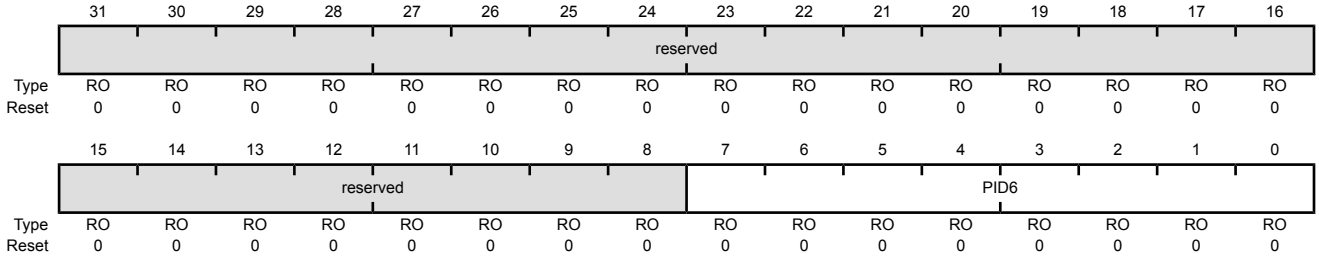
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	GPIO Peripheral ID Register [15:8]

Register 25: GPIO Peripheral Identification 6 (GPIOPeriphID6), offset 0xFD8

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

GPIO Peripheral Identification 6 (GPIOPeriphID6)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0xFD8
Type RO, reset 0x0000.0000



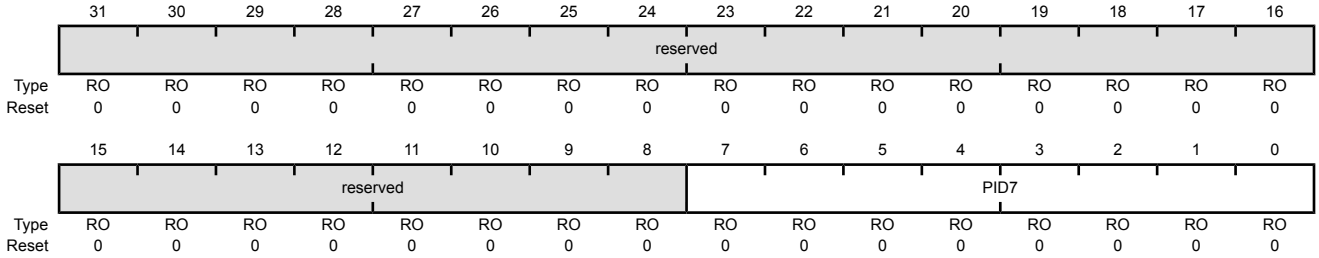
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	GPIO Peripheral ID Register [23:16]

Register 26: GPIO Peripheral Identification 7 (GPIOPeriphID7), offset 0xFDC

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

GPIO Peripheral Identification 7 (GPIOPeriphID7)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0xFDC
Type RO, reset 0x0000.0000



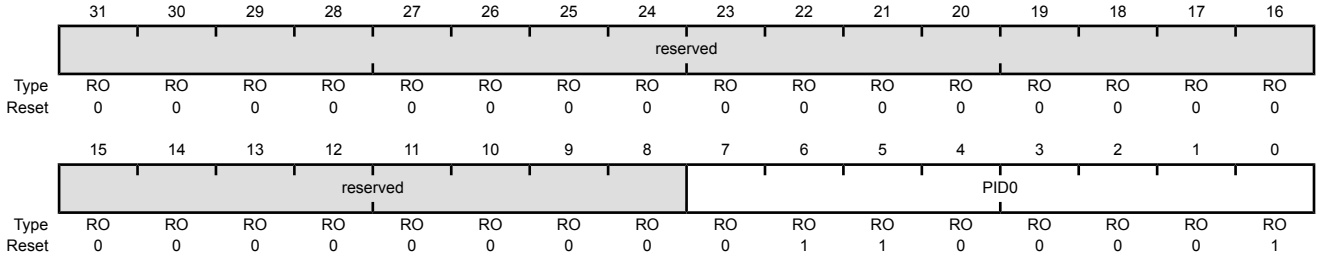
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	GPIO Peripheral ID Register [31:24]

Register 27: GPIO Peripheral Identification 0 (GPIOPeriphID0), offset 0xFE0

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

GPIO Peripheral Identification 0 (GPIOPeriphID0)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0xFE0
Type RO, reset 0x0000.0061



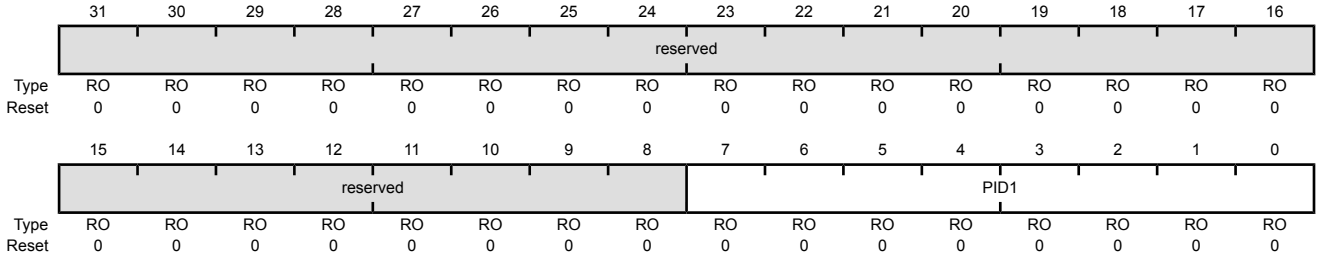
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x61	GPIO Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

Register 28: GPIO Peripheral Identification 1 (GPIOPeriphID1), offset 0xFE4

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

GPIO Peripheral Identification 1 (GPIOPeriphID1)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0xFE4
Type RO, reset 0x0000.0000



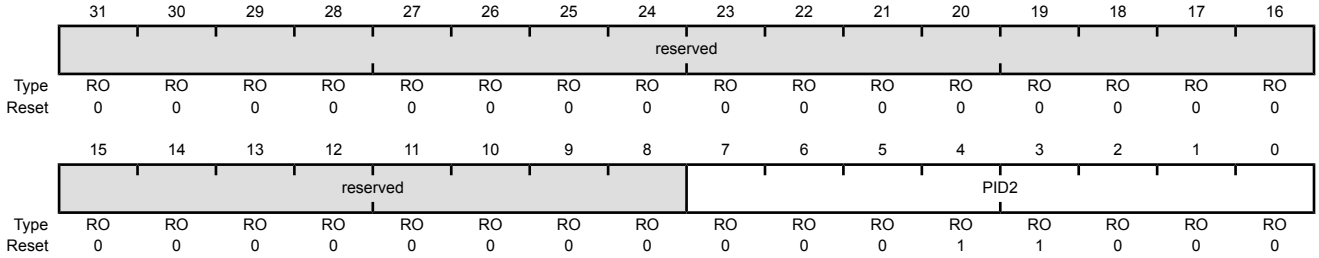
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	GPIO Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Register 29: GPIO Peripheral Identification 2 (GPIOPeriphID2), offset 0xFE8

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

GPIO Peripheral Identification 2 (GPIOPeriphID2)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0xFE8
Type RO, reset 0x0000.0018



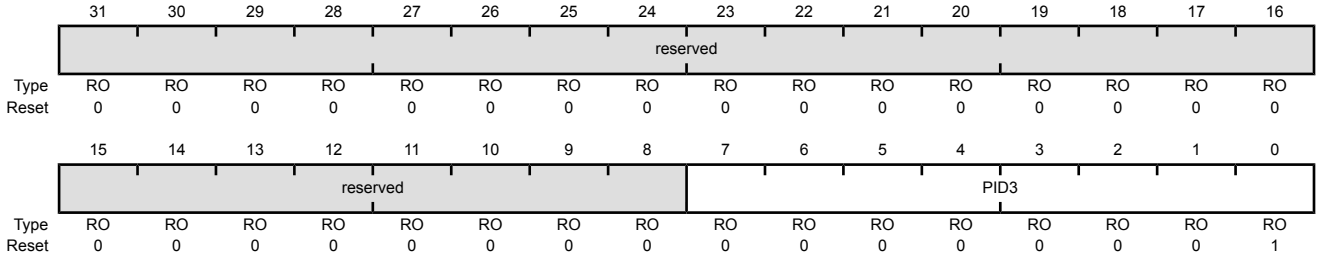
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	GPIO Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 30: GPIO Peripheral Identification 3 (GPIOPeriphID3), offset 0xFEC

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

GPIO Peripheral Identification 3 (GPIOPeriphID3)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0xFEC
Type RO, reset 0x0000.0001



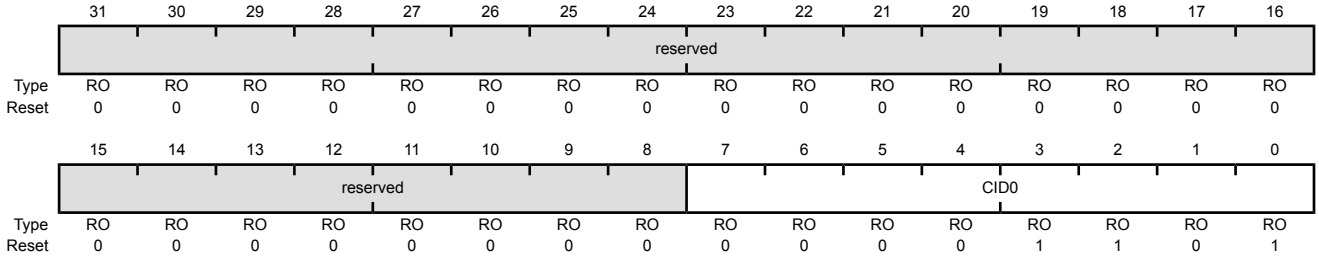
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	GPIO Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

Register 31: GPIO PrimeCell Identification 0 (GPIOPCellID0), offset 0xFF0

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

GPIO PrimeCell Identification 0 (GPIOPCellID0)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0xFF0
Type RO, reset 0x0000.000D



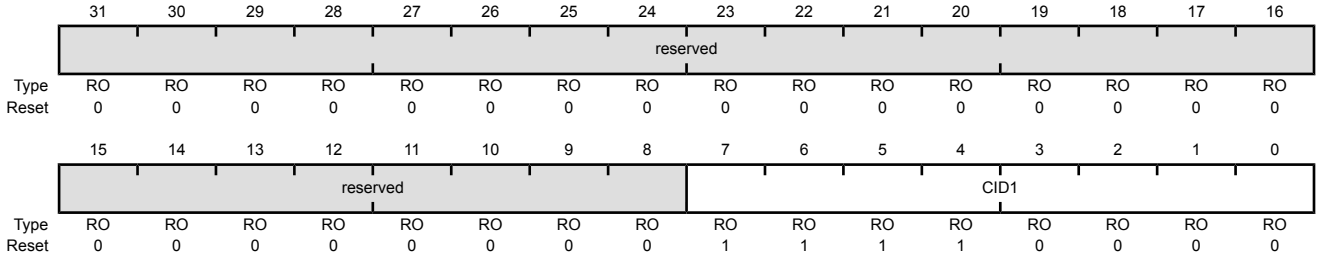
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	GPIO PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

Register 32: GPIO PrimeCell Identification 1 (GPIOPCellID1), offset 0xFF4

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

GPIO PrimeCell Identification 1 (GPIOPCellID1)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0xFF4
Type RO, reset 0x0000.00F0



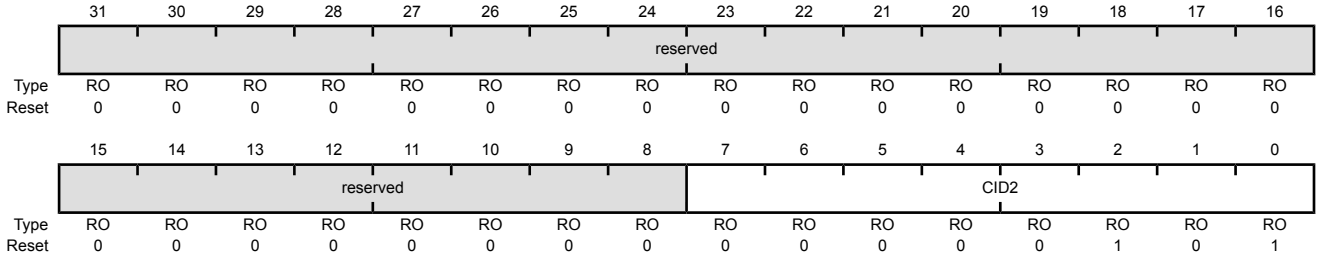
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	GPIO PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system.

Register 33: GPIO PrimeCell Identification 2 (GPIOPCellID2), offset 0xFF8

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

GPIO PrimeCell Identification 2 (GPIOPCellID2)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0xFF8
Type RO, reset 0x0000.0005



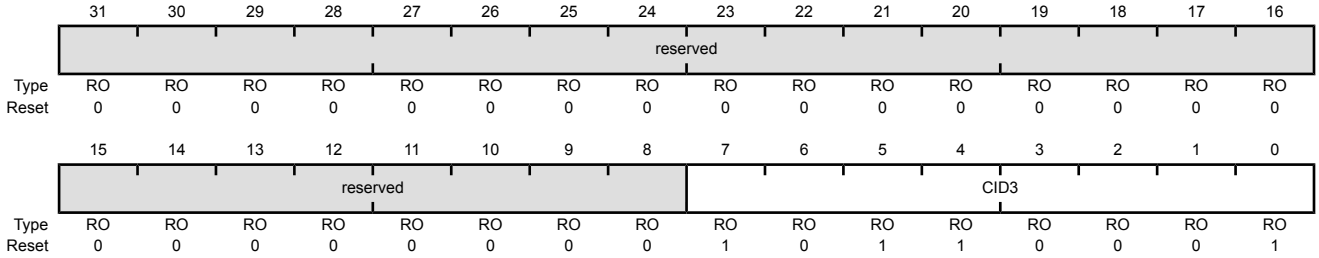
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	GPIO PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system.

Register 34: GPIO PrimeCell Identification 3 (GPIOPCellID3), offset 0xFFC

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

GPIO PrimeCell Identification 3 (GPIOPCellID3)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (APB) base: 0x4003.D000
GPIO Port J (AHB) base: 0x4006.0000
Offset 0xFFC
Type RO, reset 0x0000.00B1



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	GPIO PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system.

9 External Peripheral Interface (EPI)

The External Peripheral Interface is a high-speed parallel bus for external peripherals or memory. It has several modes of operation to interface gluelessly to many types of external devices. The External Peripheral Interface is similar to a standard microprocessor address/data bus, except that it must typically be connected to just one type of external device. Enhanced capabilities include µDMA support, clocking control and support for external FIFO buffers.

The EPI has the following features:

- 8/16/32-bit dedicated parallel bus for external peripherals and memory
- Memory interface supports contiguous memory access independent of data bus width, thus enabling code execution directly from SDRAM, SRAM and Flash memory
- Blocking and non-blocking reads
- Separates processor from timing details through use of an internal write FIFO
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Separate channels for read and write
  - Read channel request asserted by programmable levels on the internal non-blocking read FIFO (NBRFIFO)
  - Write channel request asserted by empty on the internal write FIFO (WFIFO)

The EPI supports three primary functional modes: Synchronous Dynamic Random Access Memory (SDRAM) mode, Traditional Host-Bus mode, and General-Purpose mode. The EPI module also provides custom GPIOs; however, unlike regular GPIOs, the EPI module uses a FIFO in the same way as a communication mechanism and is speed-controlled using clocking.

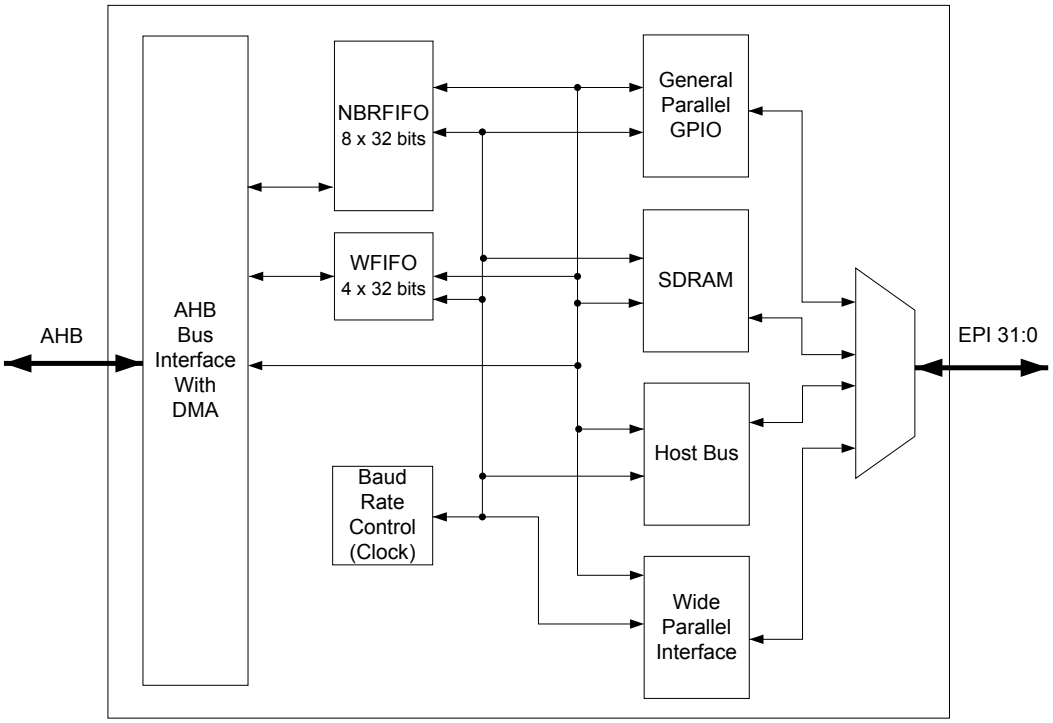
- Synchronous Dynamic Random Access Memory (SDRAM) mode
  - Supports x16 (single data rate) SDRAM at up to 50 MHz
  - Supports low-cost SDRAMs up to 64 MB (512 megabits)
  - Includes automatic refresh and access to all banks/rows
  - Includes a Sleep/Standby mode to keep contents active with minimal power draw
  - Multiplexed address/data interface for reduced pin count
- Host-Bus mode
  - Traditional x8 and x16 MCU bus interface capabilities
  - Similar device compatibility options as PIC, ATmega, 8051, and others
  - Access to SRAM, NOR Flash memory, and other devices, with up to 1 MB of addressing in unmultiplexed mode and 256 MB in multiplexed mode (512 MB in Host-Bus 16 mode with no byte selects)

  - Support of both muxed and de-muxed address and data
  - Access to a range of devices supporting the non-address FIFO x8 and x16 interface variant, with support for external FIFO (XFIFO) EMPTY and FULL signals
  - Speed controlled, with read and write data wait-state counters
  - Chip select modes include ALE, CSn, Dual CSn and ALE with dual CSn
  - Manual chip-enable (or use extra address pins)
- General-Purpose mode
  - Wide parallel interfaces for fast communications with CPLDs and FPGAs
  - Data widths up to 32 bits
  - Data rates up to 150 MB/second
  - Optional "address" sizes from 4 bits to 20 bits
  - Optional clock output, read/write strobes, framing (with counter-based size), and clock-enable input
- General parallel GPIO
  - 1 to 32 bits, FIFOed with speed control
  - Useful for custom peripherals or for digital data acquisition and actuator controls

9.1 EPI Block Diagram

Figure 9-1 on page 453 provides a block diagram of a Stellaris® EPI module.

Figure 9-1. EPI Block Diagram



9.2 Signal Description

The following table lists the external signals of the EPI controller and describes the function of each. The EPI controller signals are alternate functions for GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placement for the EPI signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 419) should be set to choose the EPI controller function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 437) to assign the EPI signals to the specified GPIO port pins. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 395.

Table 9-1. External Peripheral Interface Signals (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
EPI0S0	83	PH3 (8)	I/O	TTL	EPI module 0 signal 0.
EPI0S1	84	PH2 (8)	I/O	TTL	EPI module 0 signal 1.
EPI0S2	25	PC4 (8)	I/O	TTL	EPI module 0 signal 2.
EPI0S3	24	PC5 (8)	I/O	TTL	EPI module 0 signal 3.
EPI0S4	23	PC6 (8)	I/O	TTL	EPI module 0 signal 4.
EPI0S5	22	PC7 (8)	I/O	TTL	EPI module 0 signal 5.
EPI0S6	86	PH0 (8)	I/O	TTL	EPI module 0 signal 6.
EPI0S7	85	PH1 (8)	I/O	TTL	EPI module 0 signal 7.
EPI0S8	74	PE0 (8)	I/O	TTL	EPI module 0 signal 8.
EPI0S9	75	PE1 (8)	I/O	TTL	EPI module 0 signal 9.
EPI0S10	76	PH4 (8)	I/O	TTL	EPI module 0 signal 10.

Table 9-1. External Peripheral Interface Signals (100LQFP) *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
EPI0S11	63	PH5 (8)	I/O	TTL	EPI module 0 signal 11.
EPI0S12	42	PF4 (8)	I/O	TTL	EPI module 0 signal 12.
EPI0S13	19	PG0 (8)	I/O	TTL	EPI module 0 signal 13.
EPI0S14	18	PG1 (8)	I/O	TTL	EPI module 0 signal 14.
EPI0S15	41	PF5 (8)	I/O	TTL	EPI module 0 signal 15.
EPI0S16	14	PJ0 (8)	I/O	TTL	EPI module 0 signal 16.
EPI0S17	87	PJ1 (8)	I/O	TTL	EPI module 0 signal 17.
EPI0S18	39	PJ2 (8)	I/O	TTL	EPI module 0 signal 18.
EPI0S19	50 97	PJ3 (8) PD4 (10)	I/O	TTL	EPI module 0 signal 19.
EPI0S20	12	PD2 (8)	I/O	TTL	EPI module 0 signal 20.
EPI0S21	13	PD3 (8)	I/O	TTL	EPI module 0 signal 21.
EPI0S22	91	PB5 (8)	I/O	TTL	EPI module 0 signal 22.
EPI0S23	92	PB4 (8)	I/O	TTL	EPI module 0 signal 23.
EPI0S24	95	PE2 (8)	I/O	TTL	EPI module 0 signal 24.
EPI0S25	96	PE3 (8)	I/O	TTL	EPI module 0 signal 25.
EPI0S26	62	PH6 (8)	I/O	TTL	EPI module 0 signal 26.
EPI0S27	15	PH7 (8)	I/O	TTL	EPI module 0 signal 27.
EPI0S28	52 98	PJ4 (8) PD5 (10)	I/O	TTL	EPI module 0 signal 28.
EPI0S29	53 99	PJ5 (8) PD6 (10)	I/O	TTL	EPI module 0 signal 29.
EPI0S30	54 100	PJ6 (8) PD7 (10)	I/O	TTL	EPI module 0 signal 30.
EPI0S31	36	PG7 (9)	I/O	TTL	EPI module 0 signal 31.

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

Table 9-2. External Peripheral Interface Signals (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
EPI0S0	D10	PH3 (8)	I/O	TTL	EPI module 0 signal 0.
EPI0S1	D11	PH2 (8)	I/O	TTL	EPI module 0 signal 1.
EPI0S2	L1	PC4 (8)	I/O	TTL	EPI module 0 signal 2.
EPI0S3	M1	PC5 (8)	I/O	TTL	EPI module 0 signal 3.
EPI0S4	M2	PC6 (8)	I/O	TTL	EPI module 0 signal 4.
EPI0S5	L2	PC7 (8)	I/O	TTL	EPI module 0 signal 5.
EPI0S6	C9	PH0 (8)	I/O	TTL	EPI module 0 signal 6.
EPI0S7	C8	PH1 (8)	I/O	TTL	EPI module 0 signal 7.
EPI0S8	B11	PE0 (8)	I/O	TTL	EPI module 0 signal 8.
EPI0S9	A12	PE1 (8)	I/O	TTL	EPI module 0 signal 9.
EPI0S10	B10	PH4 (8)	I/O	TTL	EPI module 0 signal 10.
EPI0S11	F10	PH5 (8)	I/O	TTL	EPI module 0 signal 11.

Table 9-2. External Peripheral Interface Signals (108BGA) *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
EPI0S12	K4	PF4 (8)	I/O	TTL	EPI module 0 signal 12.
EPI0S13	K1	PG0 (8)	I/O	TTL	EPI module 0 signal 13.
EPI0S14	K2	PG1 (8)	I/O	TTL	EPI module 0 signal 14.
EPI0S15	K3	PF5 (8)	I/O	TTL	EPI module 0 signal 15.
EPI0S16	F3	PJ0 (8)	I/O	TTL	EPI module 0 signal 16.
EPI0S17	B6	PJ1 (8)	I/O	TTL	EPI module 0 signal 17.
EPI0S18	K6	PJ2 (8)	I/O	TTL	EPI module 0 signal 18.
EPI0S19	M10 B5	PJ3 (8) PD4 (10)	I/O	TTL	EPI module 0 signal 19.
EPI0S20	H2	PD2 (8)	I/O	TTL	EPI module 0 signal 20.
EPI0S21	H1	PD3 (8)	I/O	TTL	EPI module 0 signal 21.
EPI0S22	B7	PB5 (8)	I/O	TTL	EPI module 0 signal 22.
EPI0S23	A6	PB4 (8)	I/O	TTL	EPI module 0 signal 23.
EPI0S24	A4	PE2 (8)	I/O	TTL	EPI module 0 signal 24.
EPI0S25	B4	PE3 (8)	I/O	TTL	EPI module 0 signal 25.
EPI0S26	G3	PH6 (8)	I/O	TTL	EPI module 0 signal 26.
EPI0S27	H3	PH7 (8)	I/O	TTL	EPI module 0 signal 27.
EPI0S28	K11 C6	PJ4 (8) PD5 (10)	I/O	TTL	EPI module 0 signal 28.
EPI0S29	K12 A3	PJ5 (8) PD6 (10)	I/O	TTL	EPI module 0 signal 29.
EPI0S30	L10 A2	PJ6 (8) PD7 (10)	I/O	TTL	EPI module 0 signal 30.
EPI0S31	C10	PG7 (9)	I/O	TTL	EPI module 0 signal 31.

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

9.3 Functional Description

The EPI controller provides a glueless, programmable interface to a variety of common external peripherals such as SDRAM x 16, Host Bus x8 and x16 devices, RAM, NOR Flash memory, CPLDs and FPGAs. In addition, the EPI controller provides custom GPIO that can use a FIFO with speed control by using either the internal write FIFO (WFIFO) or the non-blocking read FIFO (NBRFIFO). The WFIFO can hold 4 words of data that are written to the external interface at the rate controlled by the **EPI Main Baud Rate (EPIBAUD)** register. The NBRFIFO can hold 8 words of data and samples at the rate controlled by the **EPIBAUD** register. The EPI controller provides predictable operation and thus has an advantage over regular GPIO which has more variable timing due to on-chip bus arbitration and delays across bus bridges. Blocking reads stall the CPU until the transaction completes. Non-blocking reads are performed in the background and allow the processor to continue operation. In addition, write data can also be stored in the WFIFO to allow multiple writes with no stalls.

Note: Both the WTAV bit field in the **EPIWFIFOCNT** register and the WBUSY bit in the **EPISTAT** register must be polled to determine if there is a current write transaction from the WFIFO. If both of these bits are clear, then a new bus access may begin.

Main read and write operations can be performed in subsets of the range 0x6000.0000 to 0xDFFF.FFFF. A read from an address mapped location uses the offset and size to control the

address and size of the external operation. When performing a multi-value load, the read is done as a burst (when available) to maximize performance. A write to an address mapped location uses the offset and size to control the address and size of the external operation. When performing a multi-value store, the write is done as a burst (when available) to maximize performance.

NAND Flash memory (x8) can be read natively. Automatic programming support is not provided; programming must be done by the user following the manufacturer's protocol. Automatic page ECC is also not supported, but can be performed in software.

9.3.1 Non-Blocking Reads

The EPI Controller supports a special kind of read called a non-blocking read, also referred to as a posted read. Where a normal read stalls the processor or µDMA until the data is returned, a non-blocking read is performed in the background.

A non-blocking read is configured by writing the start address into a **EPIRADDRn** register, the size per transaction into a **EPIRSIZEn** register, and then the count of operations into a **EPIRPSTDn** register. After each read is completed, the result is written into the NBRFIFO and the **EPIRADDRn** register is incremented by the size (1, 2, or 4).

If the NBRFIFO is filled, then the reads pause until space is made available. The NBRFIFO can be configured to interrupt the processor or trigger the µDMA based on fullness using the **EPIFIFOLVL** register. By using the trigger/interrupt method, the µDMA (or processor) can keep space available in the NBRFIFO and allow the reads to continue unimpeded.

When performing non-blocking reads, the SDRAM controller issues two additional read transactions after the burst request is terminated. The data for these additional transfers is discarded. This situation is transparent to the user other than the additional EPI bus activity and can safely be ignored.

Two non-blocking read register sets are available to allow sequencing and ping-pong use. When one completes, the other then activates. So, for example, if 20 words are to be read from 0x100 and 10 words from 0x200, the **EPIRPSTD0** register can be set up with the read from 0x100 (with a count of 20), and the **EPIRPSTD1** register can be set up with the read from 0x200 (with a count of 10). When **EPIRPSTD0** finishes (count goes to 0), the **EPIRPSTD1** register then starts its operation. The NBRFIFO has then passed 30 values. When used with the µDMA, it may transfer 30 values (simple sequence), or the primary/alternate model may be used to handle the first 20 in one way and the second 10 in another. It is also possible to reload the **EPIRPSTD0** register when it is finished (and the **EPIRPSTD1** register is active); thereby, keeping the interface constantly busy.

To cancel a non-blocking read, the **EPIRPSTDn** register is cleared. Care must be taken, however if the register set was active to drain away any values read into the NBRFIFO and ensure that any read in progress is allowed to complete.

To ensure that the cancel is complete, the following algorithm is used (using the **EPIRPSTD0** register for example):

EPIRPSTD0 = 0;

while ((**EPISTAT** & 0x11) == 0x10)

; // we are active and busy

// if here, then other one is active or interface no longer busy

cnt = (**EPIRADDR0** – original_address) / **EPIRSIZE0**; // count of values read

cnt -= values_read_so_far;

```
// cnt is now number left in FIFO
while (cnt--)
value = EPIREADFIFO; // drain
```

The above algorithm can be optimized in code; however, the important point is to wait for the cancel to complete because the external interface could have been in the process of reading a value when the cancel came in, and it must be allowed to complete.

9.3.2 DMA Operation

The µDMA can be used to achieve maximum transfer rates on the EPI through the NBRFIFO and the WFIFO. The µDMA has one channel for write and one for read. The write channel copies values to the WFIFO when the WFIFO is at the level specified by the **EPI FIFO Level Selects (EPIFIFOLVL)** register. The non-blocking read channel copies values from the NBRFIFO when the NBRFIFO is at the level specified by the **EPIFIFOLVL** register. For non-blocking reads, the start address, the size per transaction, and the count of elements must be programmed in the µDMA. Note that both non-blocking read register sets can be used, and they fill the NBRFIFO such that one runs to completion, then the next one starts (they do not interleave). Using the NBRFIFO provides the best possible transfer rate.

For blocking reads, the µDMA software channel (or another unused channel) is used for memory-to-memory transfers (or memory to peripheral, where some other peripheral is used). In this situation, the µDMA stalls until the read is complete and is not able to service another channel until the read is done. As a result, the arbitration size should normally be programmed to one access at a time. The µDMA controller can also transfer from and to the NBRFIFO and the WFIFO using the µDMA software channel in memory mode, however, the µDMA is stalled once the NBRFIFO is empty or the WFIFO is full. Note that when the µDMA controller is stalled, the core continues operation. See “Micro Direct Memory Access (µDMA)” on page 334 for more information on configuring the µDMA.

The size of the FIFOs must be taken into consideration when configuring the µDMA to transfer data to and from the EPI. The arbitration size should be 4 or less when writing to EPI address space and 8 or less when reading from EPI address space.

9.4 Initialization and Configuration

To enable and initialize the EPI controller, the following steps are necessary:

1. Enable the EPI module using the **RCGC1** register. See page 263.
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register. See page 272. To find out which GPIO port to enable, refer to “Signal Description” on page 453.
3. Set the GPIO AFSEL bits for the appropriate pins. See page 419. To determine which GPIOs to configure, see Table 22-4 on page 1144.
4. Configure the GPIO current level and/or slew rate as specified for the mode selected. See page 421 and page 429.
5. Configure the PMCn fields in the **GPIOPCTL** register to assign the EPI signals to the appropriate pins. See page 437 and Table 22-5 on page 1151.
6. Select the mode for the EPI block to SDRAM, HB8, HB16, or general parallel use, using the MODE field in the **EPI Configuration (EPICFG)** register. Set the mode-specific details (if needed)

using the appropriate mode configuration **EPI Host Bus Configuration (EPIHBnCFGn)** registers for the desired chip-select configuration. Set the **EPI Main Baud Rate (EPIBAUD)** register if the baud rate must be slower than the system clock rate.

7. Configure the address mapping using the **EPI Address Map (EPIADDRMAP)** register. The selected start address and range is dependent on the type of external device and maximum address (as appropriate). For example, for a 512-megabit SDRAM, program the ERADR field to 0x1 for address 0x6000.0000 or 0x2 for address 0x8000.0000; and program the ERSZ field to 0x3 for 256 MB. If using General-Purpose mode and no address at all, program the EPADR field to 0x1 for address 0xA000.0000 or 0x2 for address 0xC000.0000; and program the EPSZ field to 0x0 for 256 bytes.

8. To read or write directly, use the mapped address area (configured with the **EPIADDRMAP** register). Up to 4 or 5 writes can be performed at once without blocking. Each read is blocked until the value is retrieved.

9. To perform a non-blocking read, see “Non-Blocking Reads” on page 456.

The following sub-sections describe the initialization and configuration for each of the modes of operation. Care must be taken to initialize everything properly to ensure correct operation. Control of the GPIO states is also important, as changes may cause the external device to interpret pin states as actions or commands (see “Register Descriptions” on page 408). Normally, a pull-up or pull-down is needed on the board to at least control the chip-select or chip-enable as the Stellaris GPIOs come out of reset in tri-state.

9.4.1 SDRAM Mode

When activating the SDRAM mode, it is important to consider a few points:

1. Generally, it takes over 100 µs from when the mode is activated to when the first operation is allowed. The SDRAM controller begins the SDRAM initialization sequence as soon as the mode is selected and enabled via the **EPICFG** register. It is important that the GPIOs are properly configured before the SDRAM mode is enabled, as the EPI controller is relying on the GPIO block's ability to drive the pins immediately. As part of the initialization sequence, the LOAD MODE REGISTER command is automatically sent to the SDRAM with a value of 0x27, which sets a CAS latency of 2 and a full page burst length.

2. The INITSEQ bit in the **EPI Status (EPISTAT)** register can be checked to determine when the initialization sequence is complete.

3. When using a frequency range and/or refresh value other than the default value, it is important to configure the FREQ and RFSH fields in the **EPI SDRAM Configuration (EPISDRAMCFG)** register shortly after activating the mode. After the 100-µs startup time, the EPI block must be configured properly to keep the SDRAM contents stable.

4. The SLEEP bit in the **EPISDRAMCFG** register may be configured to put the SDRAM into a low-power self-refreshing state. It is important to note that the SDRAM mode must not be disabled once enabled, or else the SDRAM is no longer clocked and the contents are lost.

5. Before entering SLEEP mode, make sure all non-blocking reads and normal reads and writes have completed. If the system is running at 30 to 50 MHz, wait 2 EPI clocks after clearing the SLEEP bit before executing non-blocking reads, or normal reads and writes. If the system is configured to greater than 50 MHz, wait 5 EPI clocks before read and write transactions. For all other configurations, wait 1 EPI clock.

The SIZE field of the **EPISDRAMCFG** register must be configured correctly based on the amount of SDRAM in the system.

The FREQ field must be configured according to the value that represents the range being used. Based on the range selected, the number of external clocks used between certain operations (for example, PRECHARGE or ACTIVATE) is determined. If a higher frequency is given than is used, then the only downside is that the peripheral is slower (uses more cycles for these delays). If a lower frequency is given, incorrect operation occurs.

See "External Peripheral Interface (EPI)" on page 1201 for timing details for the SDRAM mode.

9.4.1.1 External Signal Connections

Table 9-3 on page 459 defines how EPI module signals should be connected to SDRAMs. The table applies when using a SDRAM up to 512 megabits. Note that the EPI signals must use 8-mA drive when interfacing to SDRAM, see page 423. Any unused EPI controller signals can be used as GPIOs or another alternate function.

Table 9-3. EPI SDRAM Signal Connections

EPI Signal	SDRAM Signal[a]	
EPI0S0	A0	D0
EPI0S1	A1	D1
EPI0S2	A2	D2
EPI0S3	A3	D3
EPI0S4	A4	D4
EPI0S5	A5	D5
EPI0S6	A6	D6
EPI0S7	A7	D7
EPI0S8	A8	D8
EPI0S9	A9	D9
EPI0S10	A10	D10
EPI0S11	A11	D11
EPI0S12	A12[b]	D12
EPI0S13	BA0	D13
EPI0S14	BA1	D14
EPI0S15	D15	
EPI0S16	DQML	
EPI0S17	DQMH	
EPI0S18	CASn	
EPI0S19	RASn	
EPI0S20-EPI0S27	not used	
EPI0S28	WEn	
EPI0S29	CSn	
EPI0S30	CKE	
EPI0S31	CLK	

a. If 2 signals are listed, connect the EPI signal to both pins.

b. Only for 256/512 megabit SDRAMs

9.4.1.2 Refresh Configuration

The refresh count is based on the external clock speed and the number of rows per bank as well as the refresh period. The RFSH field represents how many external clock cycles remain before an AUTO-REFRESH is required. The normal formula is:

RFSH = ($t_{Refresh\_us}$ / number_rows) / ext_clock_period

A refresh period is normally 64 ms, or 64000 µs. The number of rows is normally 4096 or 8192. The ext_clock_period is a value expressed in µsec and is derived by dividing 1000 by the clock speed expressed in MHz. So, 50 MHz is 1000/50=20 ns, or 0.02 µs. A typical SDRAM is 4096 rows per bank if the system clock is running at 50 MHz with an **EPIBAUD** register value of 0:

RFSH = (64000/4096) / 0.02 = 15.625 µs / 0.02 µs = 781.25

The default value in the RFSH field is 750 decimal or 0x2EE to allow for a margin of safety and providing 15 µs per refresh. It is important to note that this number should always be smaller or equal to what is required by the above equation. For example, if running the external clock at 25 MHz (40 ns per clock period), 390 is the highest number that may be used. Note that the external clock may be 25 MHz when the system clock is 25 MHz or when the system clock is 50 MHz and configuring the COUNT0 field in the **EPIBAUD** register to 1 (divide by 2).

If a number larger than allowed is used, the SDRAM is not refreshed often enough, and data is lost.

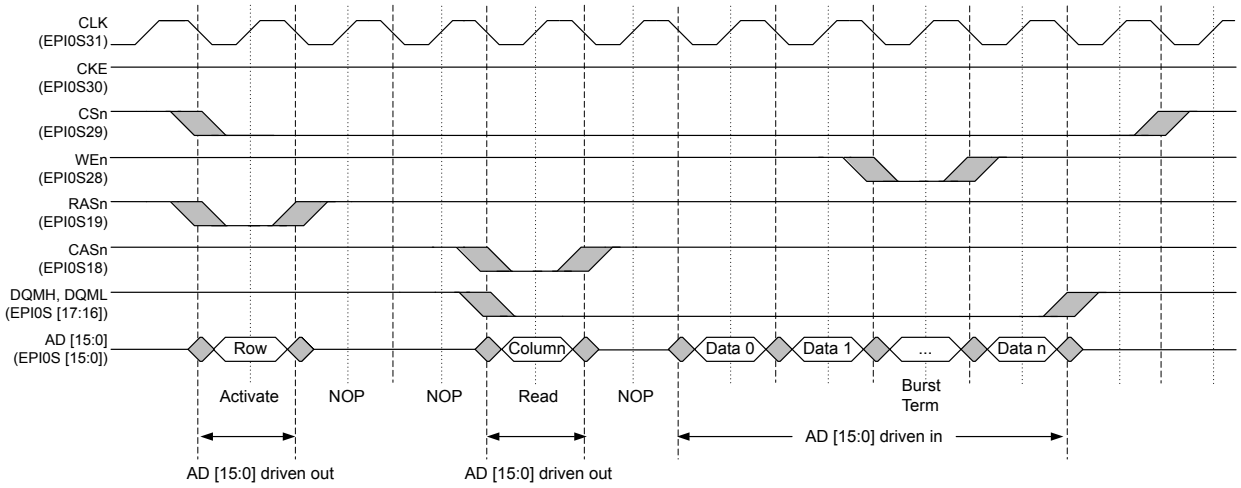
9.4.1.3 Bus Interface Speed

The EPI Controller SDRAM interface can operate up to 50 MHz. The COUNT0 field in the **EPIBAUD** register configures the speed of the EPI clock. For system clock (SysClk) speeds up to 50 MHz, the COUNT0 field can be 0x0000, and the SDRAM interface can run at the same speed as SysClk. However, if SysClk is running at higher speeds, the bus interface can run only as fast as half speed, and the COUNT0 field must be configured to at least 0x0001.

9.4.1.4 Non-Blocking Read Cycle

Figure 9-2 on page 461 shows a non-blocking read cycle of n halfwords; n can be any number greater than or equal to 1. The cycle begins with the Activate command and the row address on the EPI0S[15:0] signals. With the programmed CAS latency of 2, the Read command with the column address on the EPI0S[15:0] signals follows after 2 clock cycles. Following one more NOP cycle, data is read in on the EPI0S[15:0] signals on every rising clock edge. The Burst Terminate command is issued during the cycle when the next-to-last halfword is read in. The DQMH and DQML signals are deasserted after the last halfword of data is received; the CSn signal deasserts on the following clock cycle, signaling the end of the read cycle. At least one clock period of inactivity separates any two SDRAM cycles.

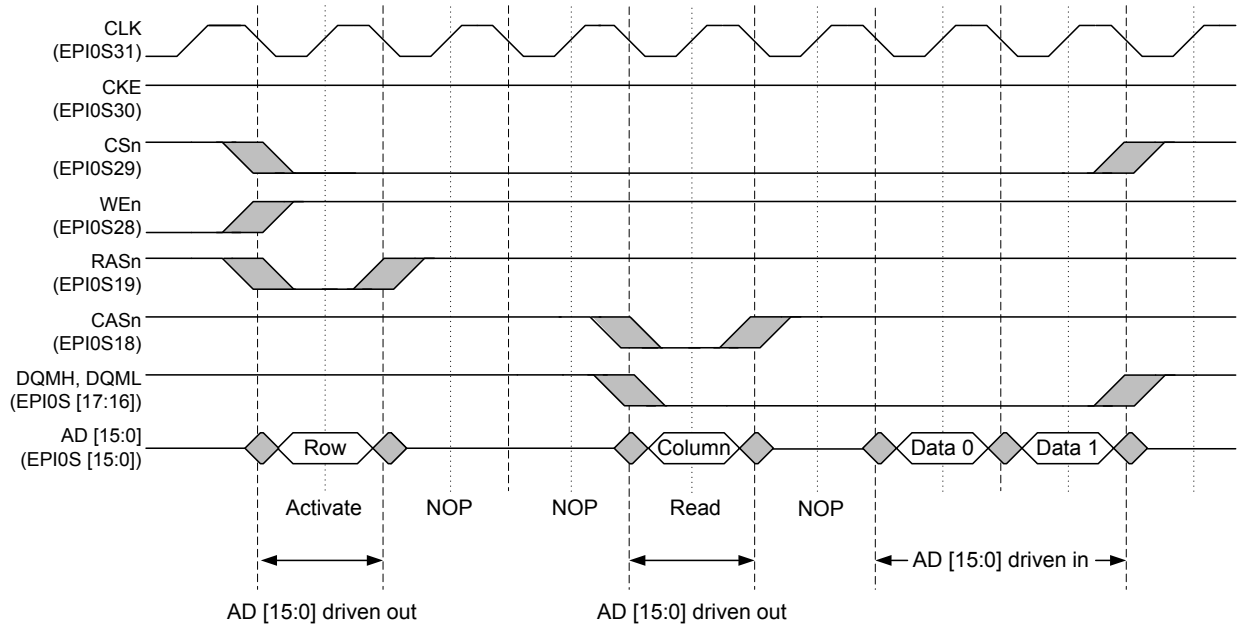
Figure 9-2. SDRAM Non-Blocking Read Cycle



9.4.1.5 Normal Read Cycle

Figure 9-3 on page 461 shows a normal read cycle of n halfwords; n can be 1 or 2. The cycle begins with the Activate command and the row address on the EPI0S[15:0] signals. With the programmed CAS latency of 2, the Read command with the column address on the EPI0S[15:0] signals follows after 2 clock cycles. Following one more NOP cycle, data is read in on the EPI0S[15:0] signals on every rising clock edge. The DQMH, DQML, and CSn signals are deasserted after the last halfword of data is received, signaling the end of the cycle. At least one clock period of inactivity separates any two SDRAM cycles.

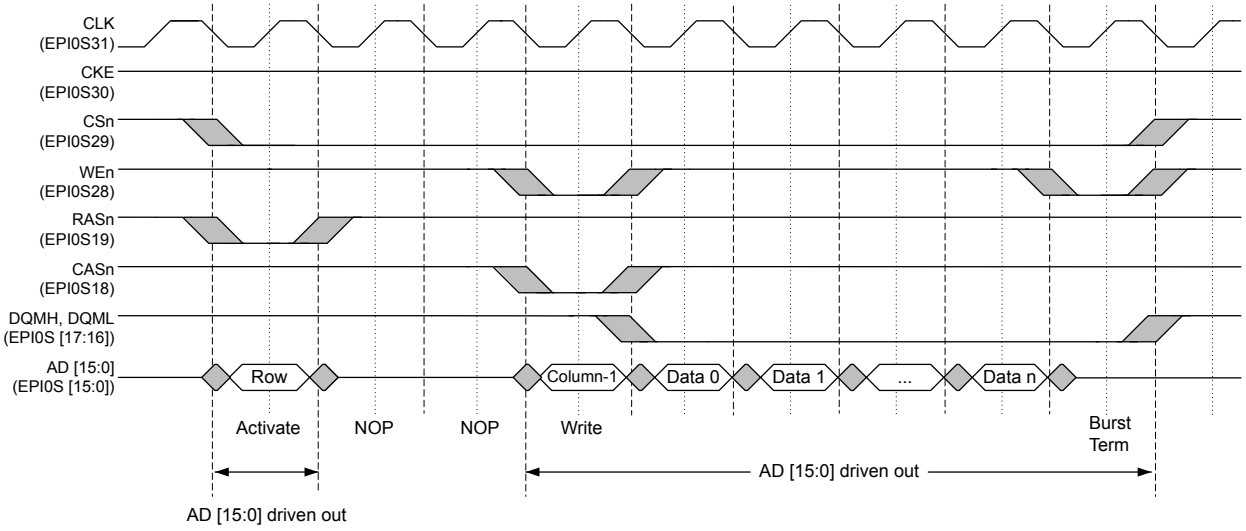
Figure 9-3. SDRAM Normal Read Cycle



9.4.1.6 Write Cycle

Figure 9-4 on page 462 shows a write cycle of n halfwords; n can be any number greater than or equal to 1. The cycle begins with the Activate command and the row address on the EPI0S[15:0] signals. With the programmed CAS latency of 2, the Write command with the column address on the EPI0S[15:0] signals follows after 2 clock cycles. When writing to SDRAMs, the Write command is presented with the first halfword of data. Because the address lines and the data lines are multiplexed, the column address is modified to be (programmed address -1). During the Write command, the DQMH and DQML signals are high, so no data is written to the SDRAM. On the next clock, the DQMH and DQML signals are asserted, and the data associated with the programmed address is written. The Burst Terminate command occurs during the clock cycle following the write of the last halfword of data. The WEn, DQMH, DQML, and CSn signals are deasserted after the last halfword of data is received, signaling the end of the access. At least one clock period of inactivity separates any two SDRAM cycles.

Figure 9-4. SDRAM Write Cycle



9.4.2 Host Bus Mode

Host Bus supports the traditional 8-bit and 16-bit interfaces popularized by the 8051 devices and SRAM devices. This interface is asynchronous and uses strobe pins to control activity. Addressable memory can be doubled using Host Bus-16 mode as it performs half-word accesses. The EPI0S0 is the LSB of the address and is equivalent to the internal Cortex-M3 A1 address. EPI0S0 should be connected to A0 of 16-bit memories.

9.4.2.1 Control Pins

The main three strobes are Address Latch Enable (ALE), Write (WRn), and Read (RDn, sometimes called OEn). Note that the timings are designed for older logic and so are hold-time vs. setup-time specific. The polarity of the read and write strobes can be active High or active Low by clearing or setting the RDHIGH and WRHIGH bits in the **EPI Host-Bus n Configuration 2 (EPIHBnCFG2)** register.

The ALE can be changed to an active-low chip select signal, CSn, through the **EPIHBnCFG2** register. The ALE is best used for Host-Bus muxed mode in which EPI address and data pins are shared. All Host-Bus accesses have an address phase followed by a data phase. The ALE indicates to an

external latch to capture the address then hold it until the data phase. CSn is best used for Host-Bus unmuxed mode in which EPI address and data pins are separate. The CSn indicates when the address and data phases of a read or write access are occurring. Both the ALE and the CSn modes can be enhanced to access external devices using settings in the **EPIHBnCFG2** register. Wait states can be added to the data phase of the access using the WRWS and RDWS bits in the **EPIHBnCFG2** register.

For FIFO mode, the ALE is not used, and two input holds are optionally supported to gate input and output to what the XFIFO can handle.

Host-Bus 8 and Host-Bus 16 modes are very configurable. The user has the ability to connect external devices to the EPI signals, as well as control whether byte select signals are provided in HB16 mode. These capabilities depend on the configuration of the MODE field in the **EPIHBnCFG** register and the CSCFG fieldin the **EPIHBnCFG2** register, and the BSEL bit in the **EPIHB16CFG** register. The CSCFGEXT bit extends the chip select configuration possibilities by providing the most significant bit of the CSCFG field.

If one of the Dual-Chip-Select modes is selected (CSCFG is 0x2 or 0x3 in the **EPIHBnCFG2** register), both chip selects can share the peripheral or the memory space, or one chip select can use the peripheral space and the other can use the memory space. In the **EPIADDRMAP** register, if the EPADR field is not 0x0 and the ERADR field is 0x0, then the address specified by EPADR is used for both chip selects, with CS0n being asserted when the MSB of the address range is 0 and CS1n being asserted when the MSB of the address range is 1. If the ERADR field is not 0x0 and the EPADR field is 0x0, then the address specified by ERADR is used for both chip selects, with the MSB performing the same delineation. If both the EPADR and the ERADR are not 0x0, then CS0n is asserted for either address range defined by EPADR and CS1n is asserted for either address range defined by ERADR.

If the CSBAUD bit in the **EPIHBnCFG2** register is set in Dual-chip select mode, the 2 chip selects can use different clock frequencies, wait states and strobe polarity. If the CSBAUD bit is clear, both chip selects use the clock frequency, wait states, and strobe polarity defined for CS0n.

When BSEL=1 in the **EPIHB16CFG** register, byte select signals are provided, so byte-sized data can be read and written at any address, however these signals reduce the available address width by 2 pins. The byte select signals are active Low. BSEL0n corresponds to the LSB of the halfword, and BSEL1n corresponds to the MSB of the halfword.

When BSEL=0, byte reads and writes at odd addresses only act on the even byte, and byte writes at even addresses write invalid values into the odd byte. As a result, accesses should be made as half-words (16-bits) or words (32-bits). In C/C++, programmers should use only short int and long int for accesses. Also, because data accesses in HB16 mode with no byte selects are on 2-byte boundaries, the available address space is doubled. For example, 28 bits of address accesses 512 MB in this mode. Table 9-4 on page 463 shows the capabilities of the HB8 and HB16 modes as well as the available address bits with the possible combinations of these bits.

Although the EPI0S31 signal can be configured for the EPI clock signal in Host-Bus mode, it is not required and should be configured as a GPIO to reduce EMI in the system.

Table 9-4. Capabilities of Host Bus 8 and Host Bus 16 Modes

Host Bus Type	MODE	CSCFG	Max # of External Devices	BSEL	Byte Access	Available Address	Addressable Memory
HB8	0x0	0x0, 0x1	1	N/A	Always	28 bits	256 MB
HB8	0x0	0x2	2	N/A	Always	27 bits	128 MB
HB8	0x0	0x3	2	N/A	Always	26 bits	64 MB

Table 9-4. Capabilities of Host Bus 8 and Host Bus 16 Modes *(continued)*

Host Bus Type	MODE	CSCFG	Max # of External Devices	BSEL	Byte Access	Available Address	Addressable Memory
HB8	0x1	0x0, 0x1	1	N/A	Always	20 bits	1 MB
HB8	0x1	0x2	2	N/A	Always	19 bits	512 kB
HB8	0x1	0x3	2	N/A	Always	18 bits	256 kB
HB8	0x3	0x1	1	N/A	Always	none	-
HB8	0x3	0x3	2	N/A	Always	none	-
HB16	0x0	0x0, 0x1	1	0	No	28 bits[a]	512 MB
HB16	0x0	0x0, 0x1	1	1	Yes	26 bits[b]	128 MB
HB16	0x0	0x2	2	0	No	27 bits[a]	256 MB
HB16	0x0	0x2	2	1	Yes	25 bits[b]	64 MB
HB16	0x0	0x3	2	0	No	26 bites[a]	128 MB
HB16	0x0	0x3	2	1	Yes	24 bits[b]	32 MB
HB16	0x1	0x0, 0x1	1	0	No	12 bits[a]	8 kB
HB16	0x1	0x0, 0x1	1	1	Yes	10 bits[b]	2 kB
HB16	0x1	0x2	2	0	No	11 bits[a]	4 kB
HB16	0x1	0x2	2	1	Yes	9 bits[b]	1 kB
HB16	0x1	0x3	2	0	No	10 bits[a]	2 kB
HB16	0x1	0x3	2	1	Yes	8 bits[b]	512 B
HB16	0x3	0x1	1	0	No	none	-
HB16	0x3	0x1	1	1	Yes	none	-
HB16	0x3	0x3	2	0	No	none	-
HB16	0x3	0x3	2	1	Yes	none	-

a. If byte selects are not used, data accesses are on 2-byte boundaries. As a result, the available address space is doubled.

b. Two EPI signals are used for byte selects, reducing the available address space by two bits.

Table 9-5 on page 464 shows how the EPI[31:0] signals function while in Host-Bus 8 mode. Notice that the signal configuration changes based on the address/data mode selected by the MODE field in the **EPIHB8CFG2** register and on the chip select configuration selected by the CSCFG field in the same register.

Although the EPI0S31 signal can be configured for the EPI clock signal in Host-Bus mode, it is not required and should be configured as a GPIO to reduce EMI in the system. Any unused EPI controller signals can be used as GPIOs or another alternate function.

Table 9-5. EPI Host-Bus 8 Signal Connections

EPI Signal	CSCFG	HB8 Signal (MODE =ADMUX)	HB8 Signal (MODE =ADNOMUX (Cont. Read))	HB8 Signal (MODE =XFIFO)
EPI0S0	X[a]	AD0	D0	D0
EPI0S1	X	AD1	D1	D1
EPI0S2	X	AD2	D2	D2
EPI0S3	X	AD3	D3	D3
EPI0S4	X	AD4	D4	D4
EPI0S5	X	AD5	D5	D5

Table 9-5. EPI Host-Bus 8 Signal Connections *(continued)*

EPI Signal	CSCFG	HB8 Signal (MODE =ADMUX)	HB8 Signal (MODE =ADNOMUX (Cont. Read))	HB8 Signal (MODE =XFIFO)
EPI0S6	X	AD6	D6	D6
EPI0S7	X	AD7	D7	D7
EPI0S8	X	A8	A0	-
EPI0S9	X	A9	A1	-
EPI0S10	X	A10	A2	-
EPI0S11	X	A11	A3	-
EPI0S12	X	A12	A4	-
EPI0S13	X	A13	A5	-
EPI0S14	X	A14	A6	-
EPI0S15	X	A15	A7	-
EPI0S16	X	A16	A8	-
EPI0S17	X	A17	A9	-
EPI0S18	X	A18	A10	-
EPI0S19	X	A19	A11	-
EPI0S20	X	A20	A12	-
EPI0S21	X	A21	A13	-
EPI0S22	X	A22	A14	-
EPI0S23	X	A23	A15	-
EPI0S24	X	A24	A16	-
EPI0S25	0x0	A25[b]	A17	-
	0x1			
	0x2			CS1n
	0x3			-
EPI0S26	0x0	A26	A18	FEMPTY
	0x1			
	0x2			
	0x3	CS0n	CS0n	
EPI0S27	0x0	A27	A19	FFULL
	0x1			
	0x2	CS1n	CS1n	
	0x3			
EPI0S28	X	RDn/OEn	RDn/OEn	RDn
EPI0S29	X	WRn	WRn	WRn
EPI0S30	0x0	ALE	ALE	-
	0x1	CSn	CSn	CSn
	0x2	CS0n	CS0n	CS0n
	0x3	ALE	ALE	-
EPI0S31	X	Clock[c]	Clock[c]	Clock[c]

a. "X" indicates the state of this field is a don't care.

b. When an entry straddles several row, the signal configuration is the same for all rows.

c. The clock signal is not required for this mode and has unspecified timing relationships to other signals.

Table 9-6 on page 466 shows how the EPI[31:0] signals function while in Host-Bus 16 mode. Notice that the signal configuration changes based on the address/data mode selected by the MODE field in the **EPIHB16CFG2** register, on the chip select configuration selected by the CSCFG field in the same register, and on whether byte selects are used as configured by the BSEL bit in the **EPIHB16CFG** register.

Although the EPI0S31 signal can be configured for the EPI clock signal in Host-Bus mode, it is not required and should be configured as a GPIO to reduce EMI in the system. Any unused EPI controller signals can be used as GPIOs or another alternate function.

Table 9-6. EPI Host-Bus 16 Signal Connections

EPI Signal	CSCFG	BSEL	HB16 Signal (MODE =ADMUX)	HB16 Signal (MODE =ADNOMUX (Cont. Read))	HB16 Signal (MODE =XFIFO)
EPI0S0	X[a]	X	AD0[b]	D0	D0
EPI0S1	X	X	AD1	D1	D1
EPI0S2	X	X	AD2	D2	D2
EPI0S3	X	X	AD3	D3	D3
EPI0S4	X	X	AD4	D4	D4
EPI0S5	X	X	AD5	D5	D5
EPI0S6	X	X	AD6	D6	D6
EPI0S7	X	X	AD7	D7	D7
EPI0S8	X	X	AD8	D8	D8
EPI0S9	X	X	AD9	D9	D9
EPI0S10	X	X	AD10	D10	D10
EPI0S11	X	X	AD11	D11	D11
EPI0S12	X	X	AD12	D12	D12
EPI0S13	X	X	AD13	D13	D13
EPI0S14	X	X	AD14	D14	D14
EPI0S15	X	X	AD15	D15	D15
EPI0S16	X	X	A16	A0[b]	-
EPI0S17	X	X	A17	A1	-
EPI0S18	X	X	A18	A2	-
EPI0S19	X	X	A19	A3	-
EPI0S20	X	X	A20	A4	-
EPI0S21	X	X	A21	A5	-
EPI0S22	X	X	A22	A6	-
EPI0S23	X[c]	0	A23	A7	-
		1			

Table 9-6. EPI Host-Bus 16 Signal Connections *(continued)*

EPI Signal	CSCFG	BSEL	HB16 Signal (MODE =ADMUX)	HB16 Signal (MODE =ADNOMUX (Cont. Read))	HB16 Signal (MODE =XFIFO)
EPI0S24	0x0	0	A24	A8	-
		1			
	0x1	0			
		1			
	0x2	0			
		1			
	0x3	0			
		1	BSEL0n	BSEL0n	
EPI0S25	0x0	X	A25	A9	-
	0x1				
	0x2	0	A25	A9	CS1n
		1	BSEL0n	BSEL0n	
	0x3	0	A25	A9	--
		1	BSEL1n	BSEL1n	
EPI0S26	0x0	0	A26	A10	FEMPTY
		1	BSEL0n	BSEL0n	
	0x1	0	A26	A10	
		1	BSEL0n	BSEL0n	
	0x2	0	A26	A10	
		1	BSEL1n	BSEL1n	
	0x3	X	CS0n	CS0n	
EPI0S27	0x0	0	A27	A11	FFULL
		1	BSEL1n	BSEL1n	
	0x1	0	A27	A11	
		1	BSEL1n	BSEL1n	
	0x2	X	CS1n	CS1n	
	0x3	X			
EPI0S28	X	X	RDn/OEn	RDn/OEn	RDn
EPI0S29	X	X	WRn	WRn	WRn
EPI0S30	0x0	X	ALE	ALE	-
	0x1	X	CSn	CSn	CSn
	0x2	X	CS0n	CS0n	CS0n
	0x3	X	ALE	ALE	-
EPI0S31	X	X	Clock[d]	Clock[d]	Clock[d]

a. "X" indicates the state of this field is a don't care.

b. In this mode, half-word accesses are used. A0 is the LSB of the address and is equivalent to the internal Cortex-M3 A1 address. This pin should be connected to A0 of 16-bit memories.

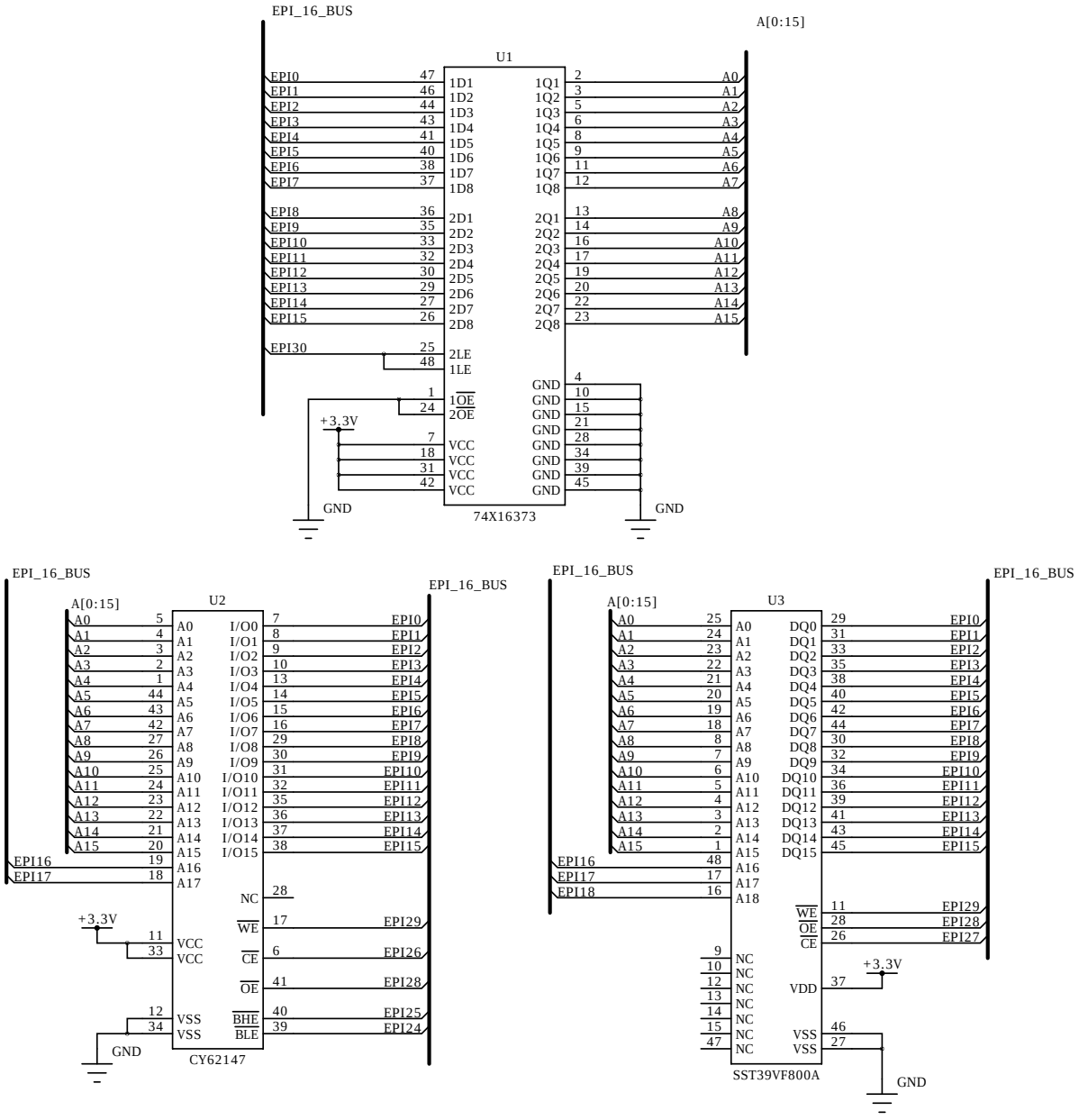
c. When an entry straddles several row, the signal configuration is the same for all rows.

d. The clock signal is not required for this mode and has unspecified timing relationships to other signals.

9.4.2.2 SRAM support

Figure 9-5 on page 468 shows how to connect the EPI signals to a 16-bit SRAM and a 16-bit Flash memory with muxed address and memory using byte selects and dual chip selects with ALE. This schematic is just an example of how to connect the signals; timing and loading have not been analyzed. In addition, not all bypass capacitors are shown.

Figure 9-5. Example Schematic for Muxed Host-Bus 16 Mode



9.4.2.3 Speed of Transactions

The COUNT0 field in the **EPIBAUD** register must be configured to set the main transaction rate based on what the slave device can support (including wiring considerations). The main control

transitions are normally ½ the baud rate (COUNT0 = 1) because the EPI block forces data vs. control to change on alternating clocks. When using dual chip selects, each chip select can access the bus using differing baud rates by setting the CSBAUD bit in the **EPIHBnCFG2** register. In this case, the COUNT0 field controls the CS0n transactions, and the COUNT1 field controls the CS1n transactions.

Additionally, the Host-Bus mode provides read and write wait states for the data portion to support different classes of device. These wait states stretch the data period (hold the rising edge of data strobe) and may be used in all four sub-modes. The wait states are set using the WRWS and RDWS bits in the **EPI Host-Bus n Configuration (EPIHBnCFG)** register.

9.4.2.4 Sub-Modes of Host Bus 8/16

The EPI controller supports four variants of the Host-Bus model using 8 or 16 bits of data in all four cases. The four sub-modes are selected using the MODE bits in the **EPIHBnCFG** register, and are:

1. Address and data are muxed. This scheme is used by many 8051 devices, some Microchip PIC parts, and some ATmega parts. When used for standard SRAMs, a latch must be used between the microcontroller and the SRAM. This sub-mode is provided for compatibility with existing devices that support data transfers without a latch (that is, CPLDs). In general, the de-muxed sub-mode should normally be used. The ALE configuration should be used in this mode, as all Host-Bus accesses have an address phase followed by a data phase. The ALE indicates to an external latch to capture the address then hold until the data phase. The ALE configuration is controlled by configuring the CSCFG field to be 0x0 in the **EPIHBnCFG2** register. The ALE can be enhanced to access two external devices with two separate CSn signals. By configuring the CSCFG field to be 0x3 in the **EPIHBnCFG2** register, EPI0S30 functions as ALE, EPI0S27 functions as CS1n, and EPI0S26 functions as CS0n. The CSn is best used for Host-Bus unmuxed mode, in which EPI address and data pins are separate. The CSn indicates when the address and data phases of a read or write access are occurring.

2. Address and data are separate with 8 or 16 bits of data and up to 20 bits of address (1 MB). This scheme is used by more modern 8051 devices, as well as some PIC and ATmega parts. This mode is generally used with real SRAMs, many EEPROMs, and many NOR Flash memory devices. Note that there is no hardware command write support for Flash memory devices; this mode should only be used for Flash memory devices programmed at manufacturing time. If a Flash memory device must be written and does not support a direct programming model, the command mechanism must be performed in software. The CSn configuration should be used in this mode. The CSn signal indicates when the address and data phases of a read or write access is occurring. The CSn configuration is controlled by configuring the CSCFG field to be 0x1 in the **EPIHBnCFG2** register.

3. Continuous read mode where address and data are separate. This sub-mode is used for real SRAMs which can be read more quickly by only changing the address (and not using RDn/OEn strobing). In this sub-mode, reads are performed by keeping the read mode selected (output enable is asserted) and then changing the address pins. The data pins are changed by the SRAM after the address pins change. For example, to read data from address 0x100 and then 0x101, the EPI controller asserts the output-enable signal and then configures the address pins to 0x100; the EPI controller then captures what is on the data pins and increments A0 to 1 (so the address is now 0x101); the EPI controller then captures what is on the data pins. Note that this mode consumes higher power because the SRAM must continuously drive the data pins. This mode is not practical in HB16 mode for normal SRAMs because there are generally not enough address bits available. Writes are not permitted in this mode.

4. FIFO mode uses 8 or 16 bits of data, removes ALE and address pins and optionally adds external XFIFO FULL/EMPTY flag inputs. This scheme is used by many devices, such as radios,

communication devices (including USB2 devices), and some FPGA configurations (FIFO through block RAM). This sub-mode provides the data side of the normal Host-Bus interface, but is paced by the FIFO control signals. It is important to consider that the XFIFO FULL/EMPTY control signals may stall the interface and could have an impact on blocking read latency from the processor or µDMA.

The WORD bit in the **EPIHBnCFG2** register can be set to use memory more efficiently. By default, the EPI controller uses data bits [7:0] for Host-Bus 8 accesses or bits [15:0] for Host-Bus 16 accesses. When the WORD bit is set, the EPI controller can automatically route bytes of data onto the correct byte lanes such that bytes or words of data can be transferred on the correct byte or half-word bits on the entire bus. For example, the most significant byte of data will be transferred on bits [31:28] in host-bus 8 mode and the most significant word of data will be transferred on bits [31:16] of Host-Bus 16 mode. In addition, for the three modes above (1, 2, 4) that the Host-Bus 16 mode supports, byte select signals can be optionally implemented by setting the BSEL bit in the **EPIHB16CFG** register.

Note: Byte accesses should not be attempted if the BSEL bit has not been enabled in Host-Bus 16 Mode.

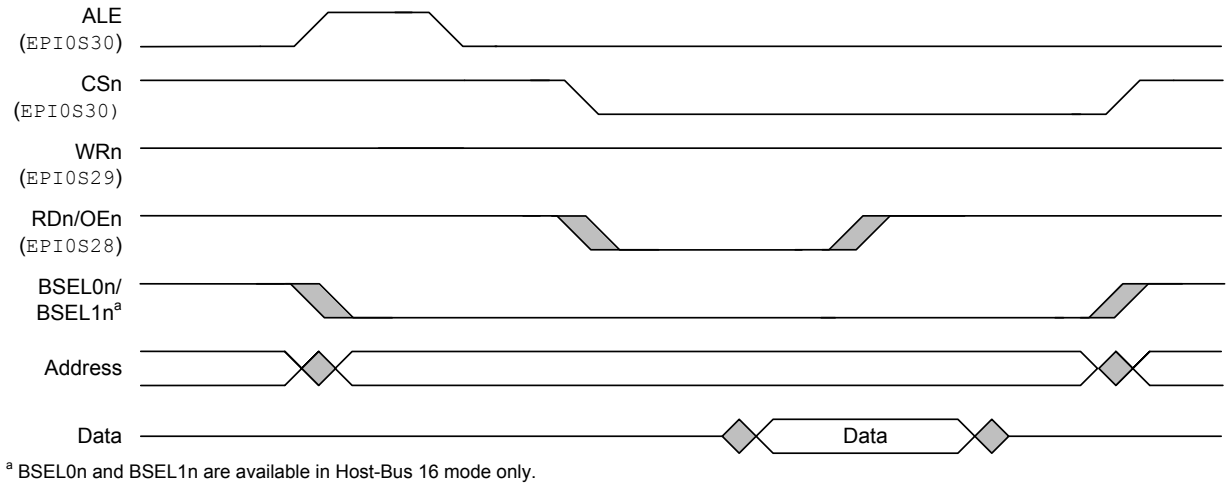
See “External Peripheral Interface (EPI)” on page 1201 for timing details for the Host-Bus mode.

9.4.2.5 Bus Operation

Bus operation is the same in Host-Bus 8 and Host-Bus 16 modes and is asynchronous. Timing diagrams show both ALE and CSn operation, but only one signal or the other is used in all modes except for ALE with dual chip selects mode (CSCFG field is 0x3 in the **EPIHBnCFG2** register). Address and data on write cycles are held after the CSn signal is deasserted. The optional HB16 byte select signals have the same timing as the address signals. If wait states are required in the bus access, they can be inserted during the data phase of the access using the WRWS and RDWS bits in the **EPIHBnCFG2** register. Each wait state adds 2 EPI clock cycles to the duration of the WRn or RDn strobe. During idle cycles, the address and muxed address data signals maintain the state of the last cycle.

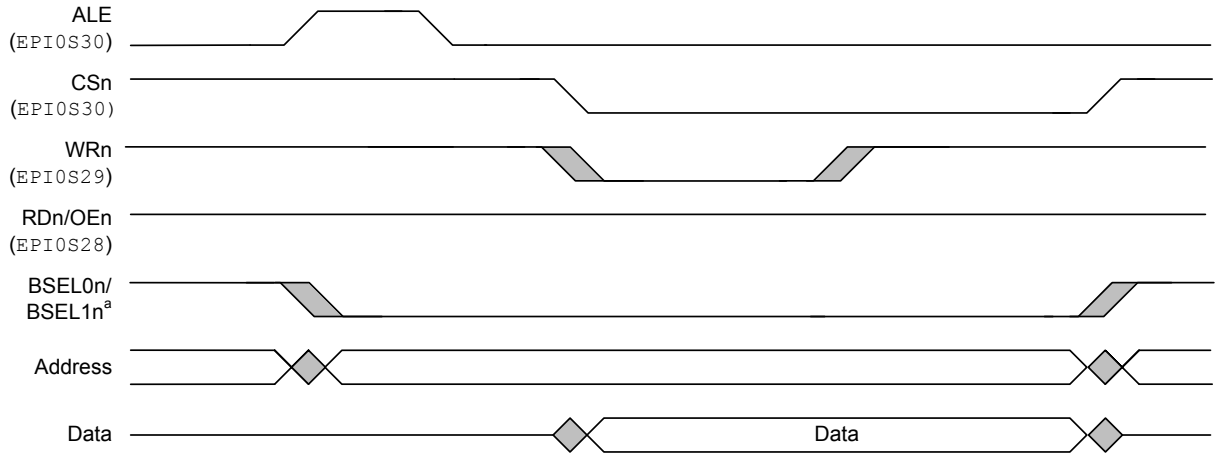
Figure 9-6 on page 470 shows a basic Host-Bus read cycle. Figure 9-7 on page 471 shows a basic Host-Bus write cycle. Both of these figures show address and data signals in the non-multiplexed mode (MODE field ix 0x1 in the **EPIHBnCFG** register).

Figure 9-6. Host-Bus Read Cycle, MODE = 0x1, WRHIGH = 0, RDHIGH = 0



[a] BSEL0n and BSEL1n are available in Host-Bus 16 mode only.

Figure 9-7. Host-Bus Write Cycle, MODE = 0x1, WRHIGH = 0, RDHIGH = 0



[a] BSEL0n and BSEL1n are available in Host-Bus 16 mode only.

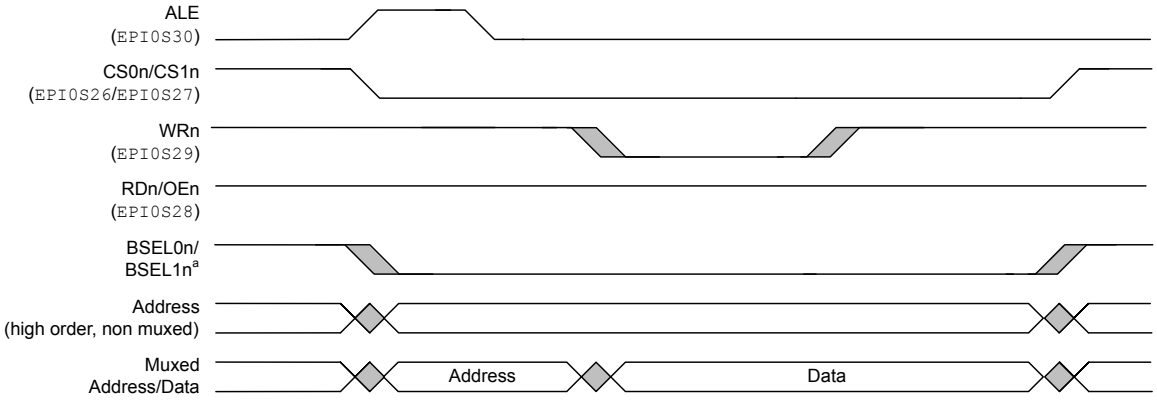
Figure 9-8 on page 471 shows a write cycle with the address and data signals multiplexed (MODE field is 0x0 in the **EPIHBnCFG** register). A read cycle would look similar, with the RDn strobe being asserted along with CSn and data being latched on the rising edge of RDn.

Figure 9-8. Host-Bus Write Cycle with Multiplexed Address and Data, MODE = 0x0, WRHIGH = 0, RDHIGH = 0

[a] BSEL0n and BSEL1n are available in Host-Bus 16 mode only.

When using ALE with dual CSn configuration (CSCFG field is 0x3 in the **EPIHBnCFG2** register), the appropriate CSn signal is asserted at the same time as ALE, as shown in Figure 9-9 on page 472.

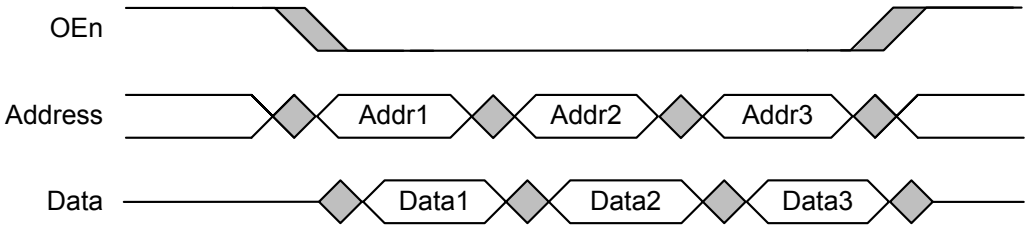
Figure 9-9. Host-Bus Write Cycle with Multiplexed Address and Data and ALE with Dual CSn



[a] BSEL0n and BSEL1n are available in Host-Bus 16 mode only.

Figure 9-10 on page 472 shows continuous read mode accesses. In this mode, reads are performed by keeping the read mode selected (output enable is asserted) and then changing the address pins. The data pins are changed by the SRAM after the address pins change.

Figure 9-10. Continuous Read Mode Accesses



FIFO mode accesses are the same as normal read and write accesses, except that the ALE signal and address pins are not present. Two input signals can be used to indicate when the XFIFO is full or empty to gate transactions and avoid overruns and underruns. The FFULL and FEMPTY signals are synchronized and must be recognized as asserted by the microcontroller for 2 system clocks before they affect transaction status. The MAXWAIT field in the **EPIHBnCFG** register defines the maximum number of EPI clocks to wait while the FEMPTY or FFULL signal is holding off a transaction. Figure 9-11 on page 473 shows how the FEMPTY signal should respond to a write and read from the XFIFO. Figure 9-12 on page 473 shows how the FEMPTY and FFULL signals should respond to 2 writes and 1 read from an external FIFO that contains two entries.

Figure 9-11. Write Followed by Read to External FIFO

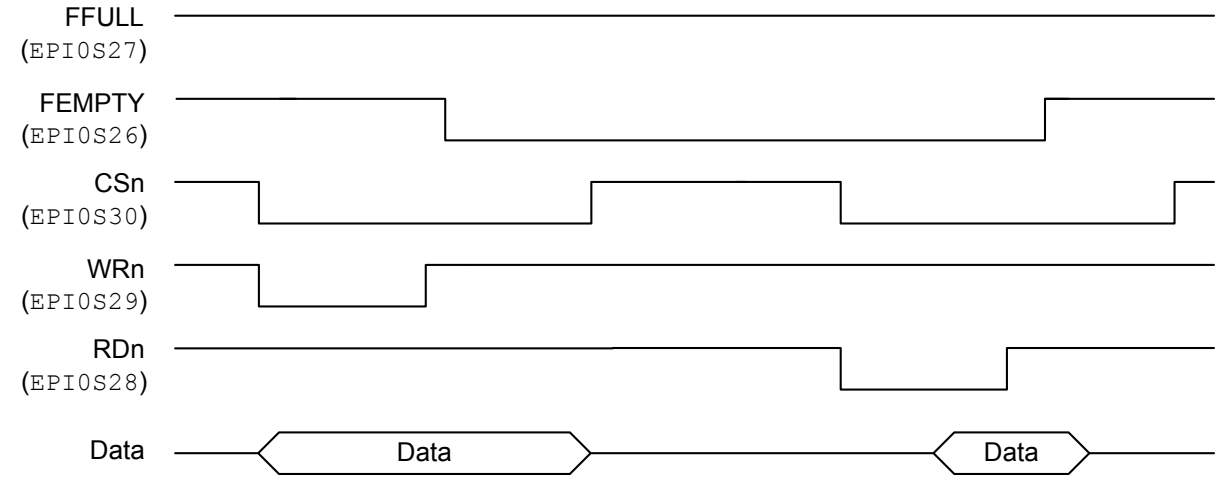
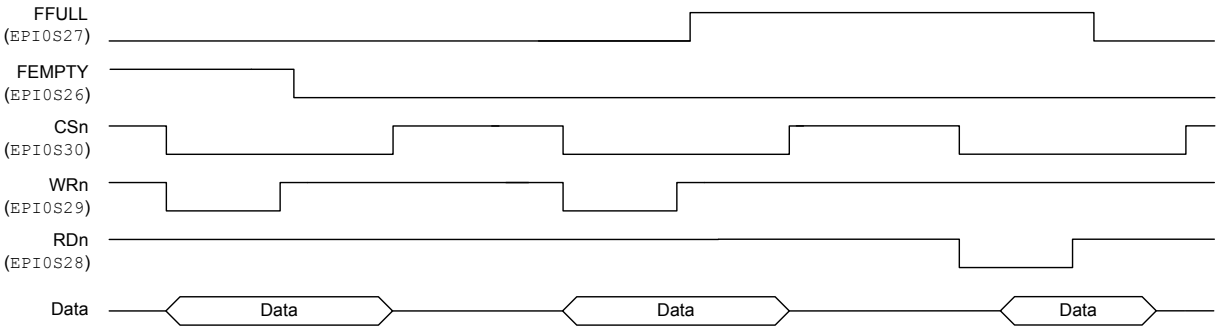


Figure 9-12. Two-Entry FIFO



9.4.3 General-Purpose Mode

The **General-Purpose Mode Configuration (EPIGPCFG)** register is used to configure the control, data, and address pins, if used. Any unused EPI controller signals can be used as GPIOs or another alternate function. The general-purpose configuration can be used for custom interfaces with FPGAs, CPLDs, and digital data acquisition and actuator control.

Important: The RD2CYC bit in the **EPIGPCFG** register must be set at all times in General-Purpose mode to ensure proper operation.

General-Purpose mode is designed for three general types of use:

- Extremely high-speed clocked interfaces to FPGAs and CPLDs. Three sizes of data and optional address are supported. Framing and clock-enable functions permit more optimized interfaces.
- General parallel GPIO. From 1 to 32 pins may be written or read, with the speed precisely controlled by the **EPIBAUD** register baud rate (when used with the WFIFO and/or the NBRFIFO) or by the rate of accesses from software or µDMA. Examples of this type of use include:
  - Reading 20 sensors at fixed time periods by configuring 20 pins to be inputs, configuring the COUNT0 field in the **EPIBAUD** register to some divider, and then using non-blocking reads.

  - Implementing a very wide ganged PWM/PCM with fixed frequency for driving actuators, LEDs, etc.
  - Implementing SDIO 4-bit mode where commands are driven or captured on 6 pins with fixed timing, fed by the µDMA.

- General custom interfaces of any speed.

The configuration allows for choice of an output clock (free-running or gated), a framing signal (with frame size), a ready input (to stretch transactions), a read and write strobe, an address (of varying sizes), and data (of varying sizes). Additionally, provisions are made for separating data and address phases.

The interface has the following optional features:

- Use of the EPI clock output is controlled by the CLKPIN bit in the **EPIGPCFG** register. Unclocked uses include general-purpose I/O and asynchronous interfaces (optionally using RD and WR strobes). Clocked interfaces allow for higher speeds and are much easier to connect to FPGAs and CPLDs (which usually include input clocks).

- EPI clock, if used, may be free running or gated depending on the CLKGATE bit in the **EPIGPCFG** register. A free-running EPI clock requires another method for determining when data is live, such as the frame pin or RD/WR strobes. A gated clock approach uses a setup-time model in which the EPI clock controls when transactions are starting and stopping. The gated clock is held high until a new transaction is started and goes high at the end of the cycle where RD/WR/FRAME and address (and data if write) are emitted.

- Use of the ready input (iRDY) from the external device is controlled by the RDYEN bit in the **EPIGPCFG** register. The iRDY signal uses EPI0S27 and may only be used with a free-running clock. iRDY gates transactions, no matter what state they are in. When iRDY is deasserted, the transaction is held off from completing.

- Use of the frame output (FRAME) is controlled by the FRMPIN bit in the **EPIGPCFG** register. The frame pin may be used whether the clock is output or not, and whether the clock is free running or not. It may also be used along with the iRDY signal. The frame may be a pulse (one clock) or may be 50/50 split across the frame size (controlled by the FRM50 bit in the **EPIGPCFG** register). The frame count (the size of the frame as specified by the FRMCNT field in the **EPIGPCFG** register) may be between 1 and 15 clocks for pulsed and between 2 and 30 clocks for 50/50. The frame pin counts transactions and not clocks; a transaction is any clock where the RD or WR strobe is high (if used). So, if the FRMCNT bit is set, then the frame pin pulses every other transaction; if 2-cycle reads and writes are used, it pulses every other address phase. FRM50 must be used with this in mind as it may hold state for many clocks waiting for the next transaction.

- Use of the RD and WR outputs is controlled by the RW bit in the **EPIGPCFG** register. For interfaces where the direction is known (in advance, related to frame size, or other means), these strobes are not needed. For most other interfaces, RD and WR are used so the external peripheral knows what transaction is taking place, and if any transaction is taking place.

- Separation of address/request and data phases may be used on writes using the WR2CYC bit in the **EPIGPCFG** register. This configuration allows the external peripheral extra time to act. Address and data phases must be separated on reads, and the RD2CYC bit in the **EPIGPCFG** register must be set. When configured to use an address as specified by the ASIZE field in the **EPIGPCFG** register, the address is emitted on the with the RD strobe (first cycle) and data is

expected to be returned on the next cycle (when RD is not asserted). If no address is used, then RD is asserted on the first cycle and data is captured on the second cycle (when RD is not asserted), allowing more setup time for data.

For writes, the output may be in one or two cycles. In the two-cycle case, the address (if any) is emitted on the first cycle with the WR strobe and the data is emitted on the second cycle (with WR not asserted). Although split address and write data phases are not normally needed for logic reasons, it may be useful to make read and write timings match. If 2-cycle reads or writes are used, the RW bit is automatically set.

- Address may be emitted (controlled by the ASIZE field in the **EPIGPCFG** register). The address may be up to 4 bits (16 possible values), up to 12 bits (4096 possible values), or up to 20 bits (1 M possible values). Size of address limits size of data, for example, 4 bits of address support up to 24 bits data. 4-bit address uses EPI0S[27:24]; 12-bit address uses EPI0S[27:16]; 20-bit address uses EPI0S[27:8]. The address signals may be used by the external peripheral as an address, code (command), or for other unrelated uses (such as a chip enable). If the chosen address/data combination does not use all of the EPI signals, the unused pins can be used as GPIOs or for other functions. For example, when using a 4-bit address with an 8-bit data, the pins assigned to EPIS0[23:8] can be assigned to other functions.

- Data may be 8 bits, 16 bits, 24 bits, or 32 bits (controlled by the DSIZE field in the **EPIGPCFG** register). 32-bit data cannot be used with address or EPI clock or any other signal. 24-bit data can only be used with 4-bit address or no address. 32-bit data requires that either the WR2CYC bit or the RD2CYC bit in the **EPIGPCFG** register is set.

- Memory can be used more efficiently by using the Word Access Mode. By default, the EPI controller uses data bits [7:0] when the DSIZE field in the **EPIGPCFG** register is 0x0; data bits [15:0] when the DSIZE field is 0x1; data bits [23:0] when the DSIZE field is 0x2; and data bits [31:0] when the DSIZE field is 0x3. When the WORD bit in the **EPIGPCFG2** register is set, the EPI controller automatically routes bytes of data onto the correct byte lanes such that data can be stored in bits [31:8] for DSIZE=0x0 and bits [31:16] for DSIZE=0x1.

- When using the EPI controller as a GPIO interface, writes are FIFOed (up to 4 can be held at any time), and up to 32 pins are changed using the **EPIBAUD** clock rate specified by COUNT0. As a result, output pin control can be very precisely controlled as a function of time. By contrast, when writing to normal GPIOs, writes can only occur 8-bits at a time and take up to two clock cycles to complete. In addition, the write itself may be further delayed by the bus due to µDMA or draining of a previous write. With both GPIO and the EPI controller, reads may be performed directly, in which case the current pin states are read back. With the EPI controller, the non-blocking interface may also be used to perform reads based on a fixed time rule via the **EPIBAUD** clock rate.

Table 9-7 on page 475 shows how the EPI0S[31:0] signals function while in General-Purpose mode. Notice that the address connections vary depending on the data-width restrictions of the external peripheral.

Table 9-7. EPI General Purpose Signal Connections

EPI Signal	General-Purpose Signal (D8, A20)	General- Purpose Signal (D16, A12)	General- Purpose Signal (D24, A4)	General- Purpose Signal (D32)
EPI0S0	D0	D0	D0	D0
EPI0S1	D1	D1	D1	D1
EPI0S2	D2	D2	D2	D2

Table 9-7. EPI General Purpose Signal Connections *(continued)*

EPI Signal	General-Purpose Signal (D8, A20)	General- Purpose Signal (D16, A12)	General- Purpose Signal (D24, A4)	General- Purpose Signal (D32)
EPI0S3	D3	D3	D3	D3
EPI0S4	D4	D4	D4	D4
EPI0S5	D5	D5	D5	D5
EPI0S6	D6	D6	D6	D6
EPI0S7	D7	D7	D7	D7
EPI0S8	A0	D8	D8	D8
EPI0S9	A1	D9	D9	D9
EPI0S10	A2	D10	D10	D10
EPI0S11	A3	D11	D11	D11
EPI0S12	A4	D12	D12	D12
EPI0S13	A5	D13	D13	D13
EPI0S14	A6	D14	D14	D14
EPI0S15	A7	D15	D15	D15
EPI0S16	A8	A0[a]	D16	D16
EPI0S17	A9	A1	D17	D17
EPI0S18	A10	A2	D18	D18
EPI0S19	A11	A3	D19	D19
EPI0S20	A12	A4	D20	D20
EPI0S21	A13	A5	D21	D21
EPI0S22	A14	A6	D22	D22
EPI0S23	A15	A7	D23	D23
EPI0S24	A16	A8	A0[b]	D24
EPI0S25	A17	A9	A1	D25
EPI0S26	A18	A10	A2	D26
EPI0S27	A19/iRDY[c]	A11/iRDY[c]	A3/iRDY[c]	D27
EPI0S28	WR	WR	WR	D28
EPI0S29	RD	RD	RD	D29
EPI0S30	Frame	Frame	Frame	D30
EPI0S31	Clock	Clock	Clock	D31

a. In this mode, half-word accesses are used. AO is the LSB of the address and is equivalent to the system A1 address.

b. In this mode, word accesses are used. AO is the LSB of the address and is equivalent to the system A2 address.

c. This signal is iRDY if the RDYEN bit in the **EPIGPCFG** register is set.

9.4.3.1 Bus Operation

A basic access is 1 EPI clock for write cycles and 2 EPI clocks for read cycles. An additional EPI clock can be inserted into a write cycle by setting the WR2CYC bit in the **EPIGPCFG** register. Note that the RD2CYC bit must always be set in the **EPIGPCFG** register.

Figure 9-13. Single-Cycle Write Access, FRM50=0, FRMCNT=0, WRCYC=0

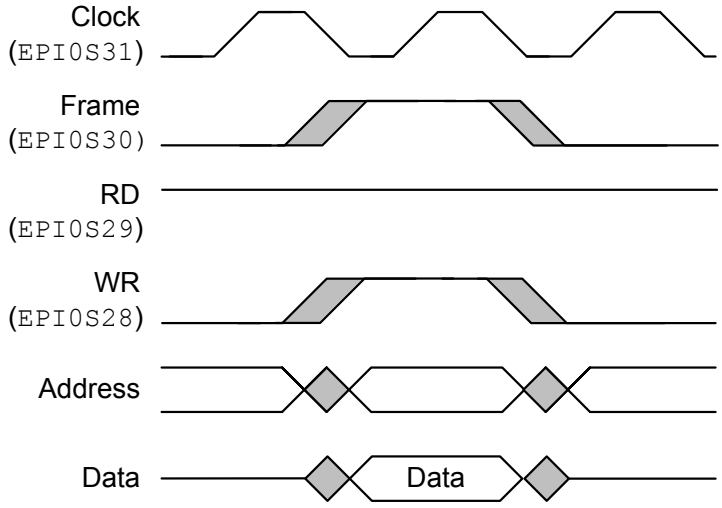


Figure 9-14. Two-Cycle Read, Write Accesses, FRM50=0, FRMCNT=0, RDCYC=1, WRCYC=1

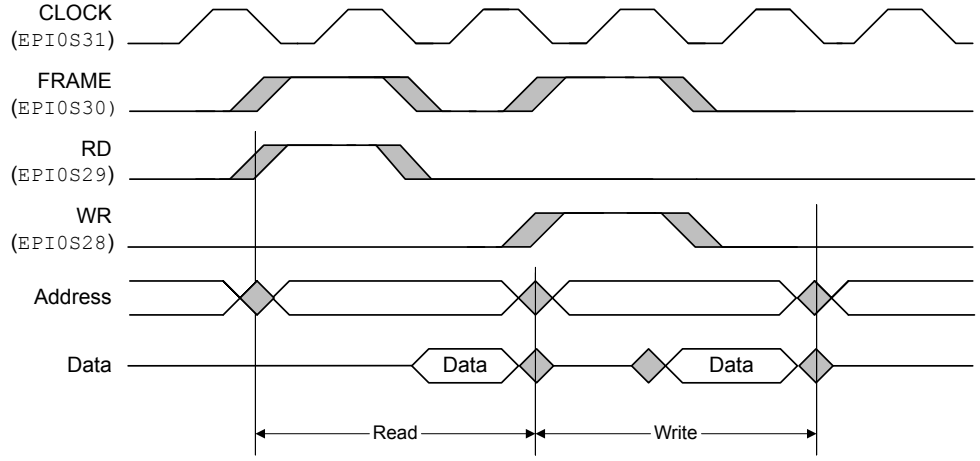
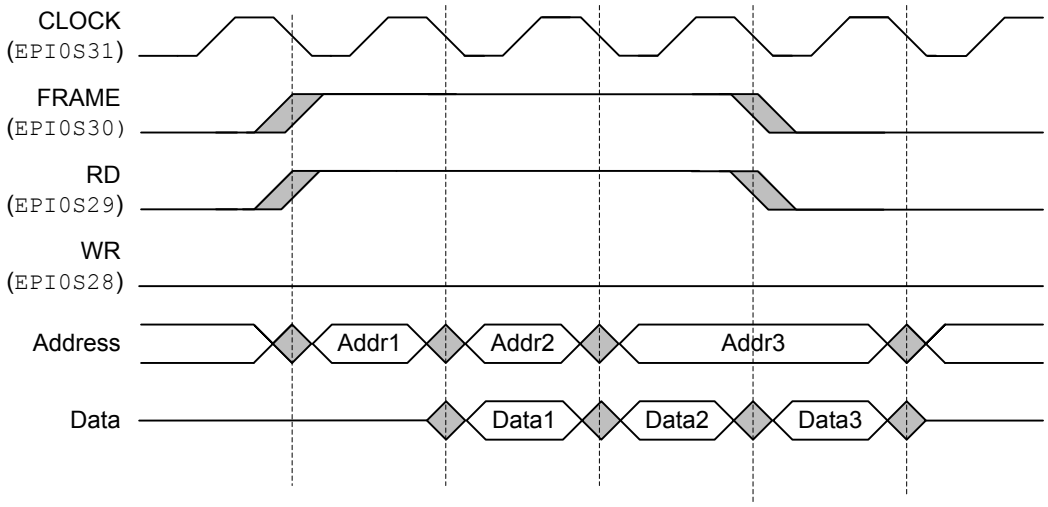


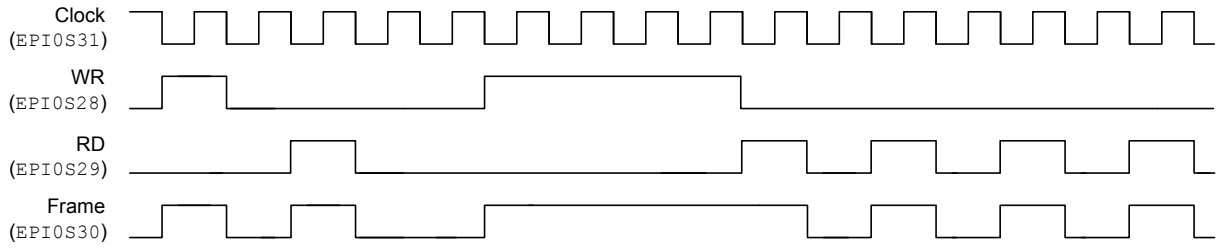
Figure 9-15. Read Accesses, FRM50=0, FRMCNT=0, RDCYC=1



FRAME Signal Operation

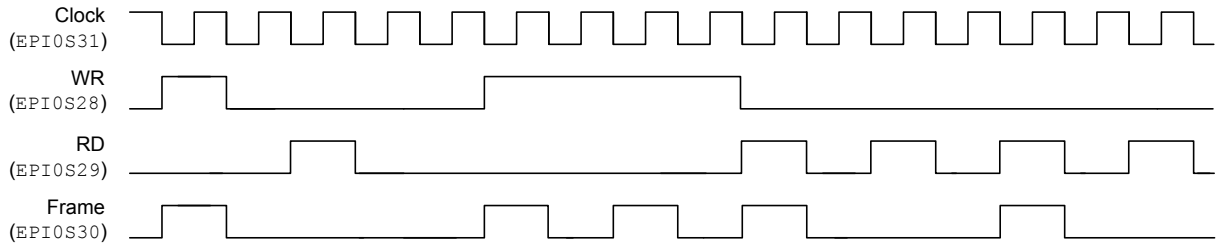
The operation of the FRAME signal is controlled by the FRMCNT and FRM50 bits. When FRM50 is clear, the FRAME signal is high whenever the WR or RD strobe is high. When FRMCNT is clear, the FRAME signal is simply the logical OR of the WR and RD strobes so the FRAME signal is high during every read or write access, see Figure 9-16 on page 478.

Figure 9-16. FRAME Signal Operation, FRM50=0 and FRMCNT=0



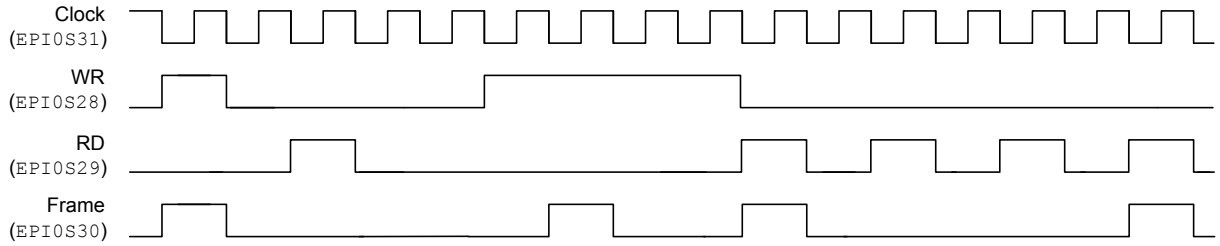
If the FRMCNT field is 0x1, then the FRAME signal pulses high during every other read or write access, see Figure 9-17 on page 478.

Figure 9-17. FRAME Signal Operation, FRM50=0 and FRMCNT=1



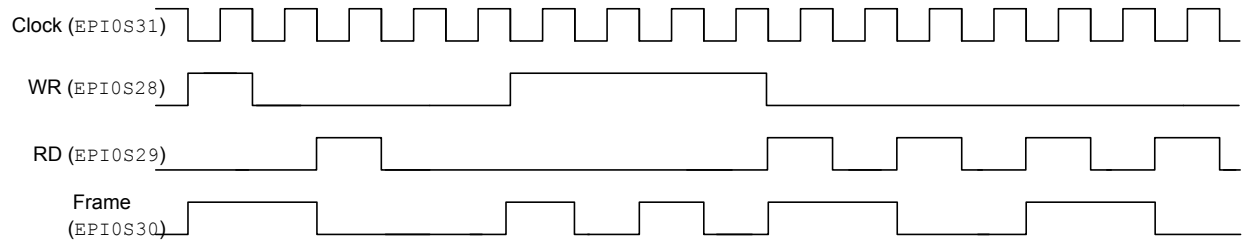
If the FRMCNT field is 0x2 and FRM50 is clear, then the FRAME signal pulses high during every third access, and so on for every value of FRMCNT, see Figure 9-18 on page 479.

Figure 9-18. FRAME Signal Operation, FRM50=0 and FRMCNT=2



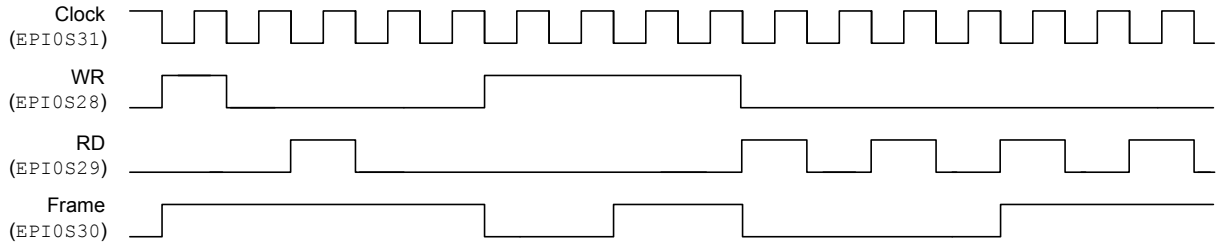
When FRM50 is set, the FRAME signal transitions on the rising edge of either the WR or RD strobes. When FRMCNT=0, the FRAME signal transitions on the rising edge of WR or RD for every access, see Figure 9-19 on page 479.

Figure 9-19. FRAME Signal Operation, FRM50=1 and FRMCNT=0



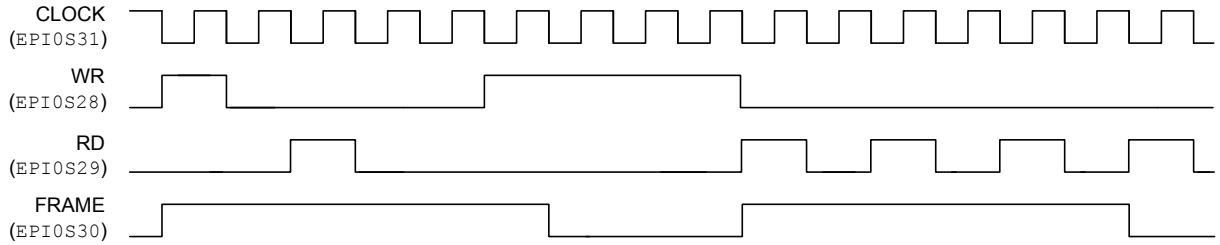
When FRMCNT=1, the FRAME signal transitions on the rising edge of the WR or RD strobes for every other access, see Figure 9-20 on page 479.

Figure 9-20. FRAME Signal Operation, FRM50=1 and FRMCNT=1



When FRMCNT=2, the FRAME signal transitions the rising edge of the WR or RD strobes for every third access, and so on for every value of FRMCNT, see Figure 9-21 on page 479.

Figure 9-21. FRAME Signal Operation, FRM50=1 and FRMCNT=2



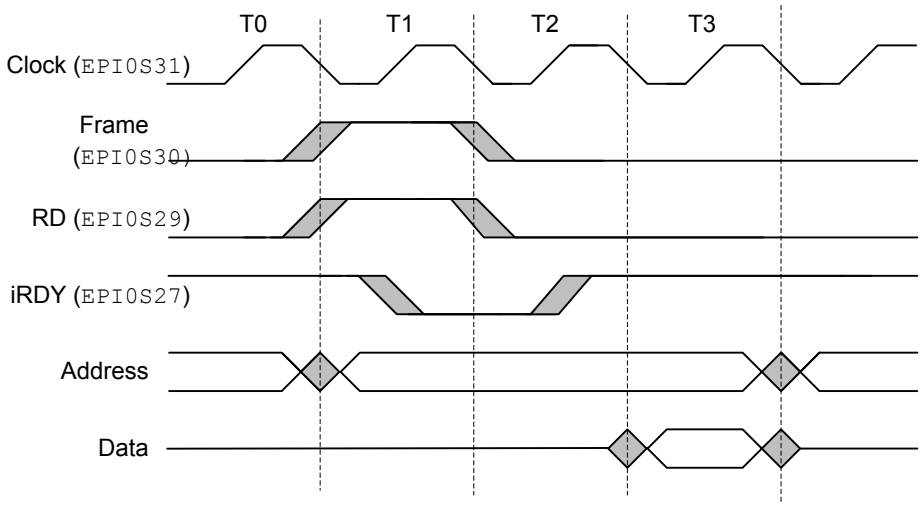
iRDY Signal Operation

The ready input (iRDY) signal can be used to lengthen bus cycles and is enabled by the RDYEN bit in the **EPIGPCFG** register. iRDY is input on EPI0S27 and may only be used with a free-running clock (CLKGATE is clear). If iRDY is deasserted, further transactions are held off until the iRDY signal is asserted again. iRDY is sampled on the falling edge of the EPI clock and gates transactions, no matter what state they are in.

A two-cycle access has two phases in the bus cycle. The first clock is the address phase, and the second clock is the data phase. If iRDY is sampled Low at the start of the address phase, as shown in Figure 24-19 on page 1206, then the address phase is extended (FRAME, RD, and Address are all asserted) until after iRDY has been sampled High again. Data is sampled on the subsequent rising edge.

If iRDY is sampled Low at the start of the data phase, as shown in Figure 9-22 on page 480, the FRAME, RD, Address, and Data signals behave as they would during a normal transaction in T1. The data phase (T2) is extended with only Address being asserted until iRDY is recognized as asserted again. Data is latched on the subsequent rising edge.

Figure 9-22. iRDY Signal Operation, FRM50=0, FRMCNT=0, and RD2CYC=1



EPI Clock Operation

If the CLKGATE bit in the **EPIGPCFG** register is clear, the EPI clock always toggles when General-purpose mode is enabled. If CLKGATE is set, the clock is output only when a transaction is occurring, otherwise the clock is held high. If the WR2CYC bit is clear, the EPI clock begins toggling 1 cycle before the WR strobe goes high. If the WR2CYC bit is set, the EPI clock begins toggling when the WR strobe goes high. The clock stops toggling after the first rising edge after the WR strobe is deasserted. The RD strobe operates in the same manner as the WR strobe when the WR2CYC bit is set, as the RD2CYC bit must always be set. See Figure 9-23 on page 481 and Figure 9-24 on page 481.

Figure 9-23. EPI Clock Operation, CLKGATE=1, WR2CYC=0

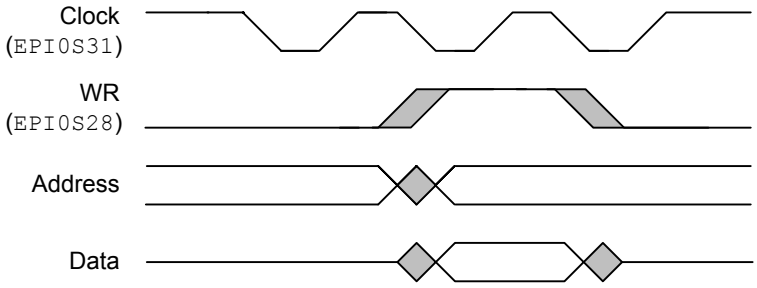
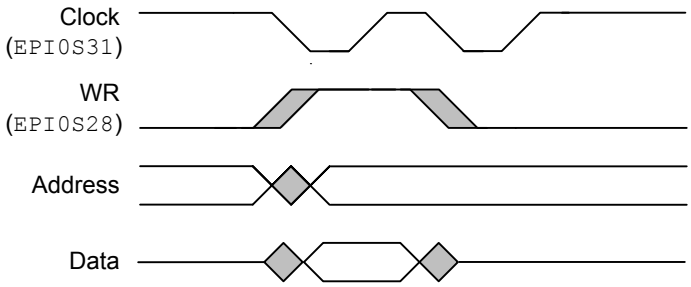


Figure 9-24. EPI Clock Operation, CLKGATE=1, WR2CYC=1



9.5 Register Map

Table 9-8 on page 481 lists the EPI registers. The offset listed is a hexadecimal increment to the register's address, relative to the base address of 0x400D.0000. Note that the EPI controller clock must be enabled before the registers can be programmed (see page 263). There must be a delay of 3 system clocks after the EPI module clock is enabled before any EPI module registers are accessed.

Note: A back-to-back write followed by a read of the same register reads the value that written by the first write access, not the value from the second write access. (This situation only occurs when the processor core attempts this action, the µDMA does not do this.). To read back what was just written, another instruction must be generated between the write and read. Read-write does not have this issue, so use of read-write for clear of error interrupt cause is not affected.

Table 9-8. External Peripheral Interface (EPI) Register Map

Offset	Name	Type	Reset	Description	See page
0x000	EPICFG	R/W	0x0000.0000	EPI Configuration	483
0x004	EPIBAUD	R/W	0x0000.0000	EPI Main Baud Rate	484
0x010	EPISDRAMCFG	R/W	0x82EE.0000	EPI SDRAM Configuration	486
0x010	EPIHB8CFG	R/W	0x0000.FF00	EPI Host-Bus 8 Configuration	488
0x010	EPIHB16CFG	R/W	0x0000.FF00	EPI Host-Bus 16 Configuration	491
0x010	EPIGPCFG	R/W	0x0000.0000	EPI General-Purpose Configuration	495
0x014	EPIHB8CFG2	R/W	0x0000.0000	EPI Host-Bus 8 Configuration 2	500

Table 9-8. External Peripheral Interface (EPI) Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x014	EPIHB16CFG2	R/W	0x0000.0000	EPI Host-Bus 16 Configuration 2	503
0x014	EPIGPCFG2	R/W	0x0000.0000	EPI General-Purpose Configuration 2	506
0x01C	EPIADDRMAP	R/W	0x0000.0000	EPI Address Map	507
0x020	EPIRSIZE0	R/W	0x0000.0003	EPI Read Size 0	509
0x024	EPIRADDR0	R/W	0x0000.0000	EPI Read Address 0	510
0x028	EPIRPSTD0	R/W	0x0000.0000	EPI Non-Blocking Read Data 0	511
0x030	EPIRSIZE1	R/W	0x0000.0003	EPI Read Size 1	509
0x034	EPIRADDR1	R/W	0x0000.0000	EPI Read Address 1	510
0x038	EPIRPSTD1	R/W	0x0000.0000	EPI Non-Blocking Read Data 1	511
0x060	EPISTAT	RO	0x0000.0000	EPI Status	513
0x06C	EPIRFIFOCNT	RO	-	EPI Read FIFO Count	515
0x070	EPIREADFIFO	RO	-	EPI Read FIFO	516
0x074	EPIREADFIFO1	RO	-	EPI Read FIFO Alias 1	516
0x078	EPIREADFIFO2	RO	-	EPI Read FIFO Alias 2	516
0x07C	EPIREADFIFO3	RO	-	EPI Read FIFO Alias 3	516
0x080	EPIREADFIFO4	RO	-	EPI Read FIFO Alias 4	516
0x084	EPIREADFIFO5	RO	-	EPI Read FIFO Alias 5	516
0x088	EPIREADFIFO6	RO	-	EPI Read FIFO Alias 6	516
0x08C	EPIREADFIFO7	RO	-	EPI Read FIFO Alias 7	516
0x200	EPIFIFOLVL	R/W	0x0000.0033	EPI FIFO Level Selects	517
0x204	EPIWFIFOCNT	RO	0x0000.0004	EPI Write FIFO Count	519
0x210	EPIIM	R/W	0x0000.0000	EPI Interrupt Mask	520
0x214	EPIRIS	RO	0x0000.0004	EPI Raw Interrupt Status	521
0x218	EPIMIS	RO	0x0000.0000	EPI Masked Interrupt Status	523
0x21C	EPIEISC	R/W1C	0x0000.0000	EPI Error and Interrupt Status and Clear	524

9.6 Register Descriptions

This section lists and describes the EPI registers, in numerical order by address offset.

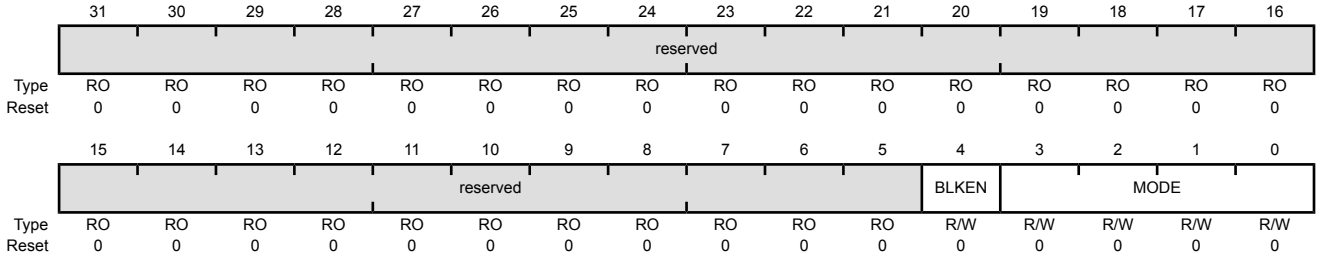
Register 1: EPI Configuration (EPICFG), offset 0x000

Important: The MODE field determines which configuration register is accessed for offsets 0x010 and 0x014. Any write to the **EPICFG** register resets the register contents at offsets 0x010 and 0x014.

The configuration register is used to enable the block, select a mode, and select the basic pin use (based on the mode). Note that attempting to program an undefined MODE field clears the BLKEN bit and disables the EPI controller.

EPI Configuration (EPICFG)

Base 0x400D.0000
Offset 0x000
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	BLKEN	R/W	0	Block Enable Value Description 0 The EPI controller is disabled. 1 The EPI controller is enabled.
3:0	MODE	R/W	0x0	Mode Select Value Description 0x0 General Purpose General-Purpose mode. Control, address, and data pins are configured using the **EPIGPCFG** and **EPIGPCFG2** registers. 0x1 SDRAM Supports SDR SDRAM. Control, address, and data pins are configured using the **EPISDRAMCFG** register. 0x2 8-Bit Host-Bus (HB8) Host-bus 8-bit interface (also known as the MCU interface). Control, address, and data pins are configured using the **EPIHB8CFG** and **EPIHB8CFG2** registers. 0x3 16-Bit Host-Bus (HB16) Host-bus 16-bit interface (standard SRAM). Control, address, and data pins are configured using the **EPIHB16CFG** and **EPIHB16CFG2** registers. 0x3-0xF Reserved

Register 2: EPI Main Baud Rate (EPIBAUD), offset 0x004

The system clock is used internally to the EPI Controller. The baud rate counter can be used to divide the system clock down to control the speed on the external interface. If the mode selected emits an external EPI clock, this register defines the EPI clock emitted. If the mode selected does not use an EPI clock, this register controls the speed of changes on the external interface. Care must be taken to program this register properly so that the speed of the external bus corresponds to the speed of the external peripheral and puts acceptable current load on the pins. COUNT0 is the bit field used in all modes except in HB8 and HB16 modes with dual chip selects when different baud rates are selected, see page 500 and page 503. If different baud rates are used, COUNT0 is associated with the address range specified by CS0n and COUNT1 is associated with the address range specified by CS1.

The COUNTn field is not a straight divider or count. The EPI Clock on EPI0S31 is related to the COUNTn field and the system clock as follows:

If COUNTn = 0,

$$EPIClockFreq = SystemClockFreq$$

otherwise:

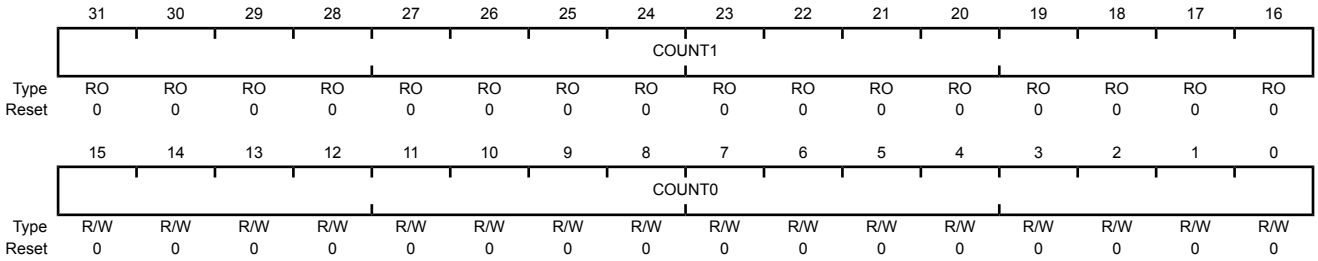
$$EPIClockFreq = \frac{SystemClockFreq}{\left(\left\lfloor \frac{COUNTn}{2} \right\rfloor + 1\right) \times 2}$$

where the symbol around COUNTn/2 is the floor operator, meaning the largest integer less than or equal to COUNTn/2.

So, for example, a COUNTn of 0x0001 results in a clock rate of ½(system clock); a COUNTn of 0x0002 or 0x0003 results in a clock rate of ¼(system clock).

EPI Main Baud Rate (EPIBAUD)

Base 0x400D.0000
Offset 0x004
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	COUNT1	RO	0x0000	Baud Rate Counter 1 This bit field is only valid with multiple chip selects which are enabled when the CSCFG field is 0x2 or 0x3 and the CSBAUD bit is set in the **EPIHBnCFG2** register. This bit field contains a counter used to divide the system clock by the count. A count of 0 means the system clock is used as is.

Bit/Field	Name	Type	Reset	Description
15:0	COUNT0	R/W	0x0000	Baud Rate Counter 0 This bit field contains a counter used to divide the system clock by the count. A count of 0 means the system clock is used as is.

Register 3: EPI SDRAM Configuration (EPISDRAMCFG), offset 0x010

Important: The MODE field in the **EPICFG** register determines which configuration register is accessed for offsets 0x010 and 0x014.

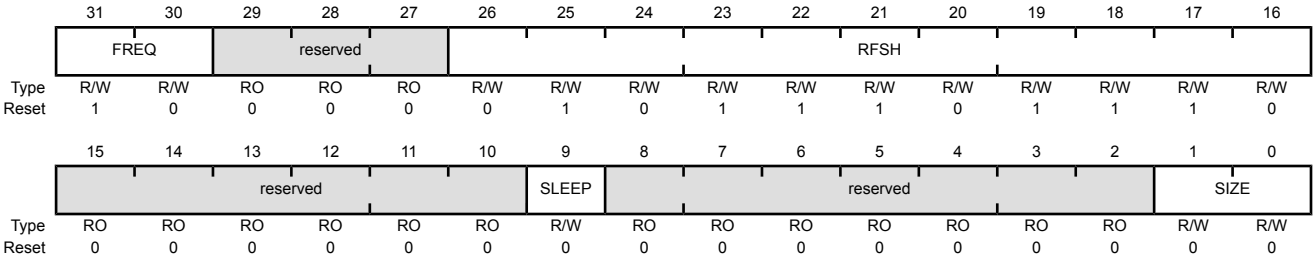
To access **EPISDRAMCFG**, the MODE field must be 0x1.

The SDRAM Configuration register is used to specify several parameters for the SDRAM controller. Note that this register is reset when the MODE field in the **EPICFG** register is changed. If another mode is selected and the SDRAM mode is selected again, the values must be reinitialized.

The SDRAM interface is designed to interface to x16 SDR SDRAMs of 64 MHz or higher, with the address and data pins overlapped (wire ORed on the board). See Table 9-3 on page 459 for pin assignments.

EPI SDRAM Configuration (EPISDRAMCFG)

Base 0x400D.0000
Offset 0x010
Type R/W, reset 0x82EE.0000



Bit/Field	Name	Type	Reset	Description
31:30	FREQ	R/W	0x2	EPI Frequency Range This field configures the frequency range used for delay references by internal counters. This EPI frequency is the system frequency with the divider programmed by the COUNT0 bit in the **EPIBAUDn** register bit. This field affects the power up, precharge, and auto refresh delays. This field does not affect the refresh counting, which is configured separately using the RFSH field (and is based on system clock rate and number of rows per bank). The ranges are: Value Description 0x0 0 - 15 MHz 0x1 15 - 30 MHz 0x2 30 - 50 MHz 0x3 50 - 100 MHz
29:27	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26:16	RFSH	R/W	0x2EE	Refresh Counter This field contains the refresh counter in system clocks. The reset value of 0x2EE provides a refresh period of 64 ms when using a 50 MHz clock.

Bit/Field	Name	Type	Reset	Description
15:10	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	SLEEP	R/W	0	Sleep Mode Value Description 0 No effect. 1 The SDRAM is put into low power state, but is self-refreshed.
8:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1:0	SIZE	R/W	0x0	Size of SDRAM The value of this field affects address pins and behavior. Value Description 0x0 64 megabits (8MB) 0x1 128 megabits (16MB) 0x2 256 megabits (32MB) 0x3 512 megabits (64MB)

Register 4: EPI Host-Bus 8 Configuration (EPIHB8CFG), offset 0x010

Important: The MODE field in the **EPICFG** register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access **EPIHB8CFG**, the MODE field must be 0x2.

The Host Bus 8 Configuration register is activated when the HB8 mode is selected. The HB8 mode supports muxed address/data (overlay of lower 8 address and all 8 data pins), separate address/data, and address-less FIFO mode. Note that this register is reset when the MODE field in the **EPICFG** register is changed. If another mode is selected and the HB8 mode is selected again, the values must be reinitialized.

This mode is intended to support SRAMs, Flash memory (read), FIFOs, CPLDs/FPGAs, and devices with an MCU/HostBus slave or 8-bit FIFO interface support.

Refer to Table 9-5 on page 464 for information on signal configuration controlled by this register and the **EPIHB8CFG2** register.

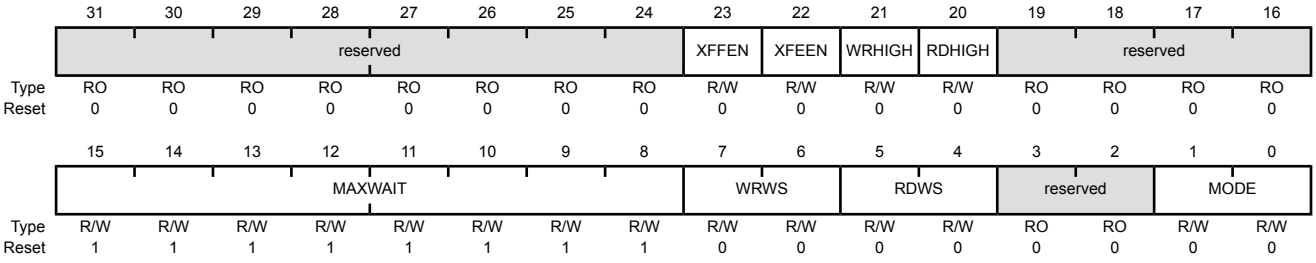
If less address pins are required, the corresponding AFSEL bit (page 419) should not be enabled so the EPI controller does not drive those pins, and they are available as standard GPIOs.

EPI Host-Bus 8 Mode can be configured to use one chip select with and without the use of ALE. If an alternative to chip selects are required, a chip enable can be handled in one of three ways:

1. Manually control via GPIOs.
2. Associate one or more upper address pins to CE. Because CE is normally CEn, lower addresses are not used. For example, if pins EPI0S27 and EPI0S26 are used for Device 1 and 0 respectively, then address 0x6800.0000 accesses Device 0 (Device 1 has its CEn high), and 0x6400.0000 accesses Device 1 (Device 0 has its CEn high). The pull-up behavior on the corresponding GPIOs must be properly configured to ensure that the pins are disabled when the interface is not in use.
3. With certain SRAMs, the ALE can be used as CEn because the address remains stable after the ALE strobe. The subsequent WRn or RDn signals write or read when ALE is low thus providing CEn functionality.

EPI Host-Bus 8 Configuration (EPIHB8CFG)

Base 0x400D.0000
Offset 0x010
Type R/W, reset 0x0000.FF00



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

Bit/Field	Name	Type	Reset	Description
23	XFFEN	R/W	0	External FIFO FULL Enable Value Description 0 No effect. 1 An external FIFO full signal can be used to control write cycles. If this bit is set and the FFULL full signal is high, XFIFO writes are stalled.
22	XFEEN	R/W	0	External FIFO EMPTY Enable Value Description 0 No effect. 1 An external FIFO empty signal can be used to control read cycles. If this bit is set and the FEMPTY signal is high, XFIFO reads are stalled.
21	WRHIGH	R/W	0	WRITE Strobe Polarity Value Description 0 The WRITE strobe for CS0n is WRn (active Low). 1 The WRITE strobe for CS0n is WR (active High).
20	RDHIGH	R/W	0	READ Strobe Polarity Value Description 0 The READ strobe for CS0n is RDn (active Low). 1 The READ strobe for CS0n is RD (active High).
19:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	MAXWAIT	R/W	0xFF	Maximum Wait This field defines the maximum number of external clocks to wait while an external FIFO ready signal is holding off a transaction (FFULL and FEMPTY). When the MAXWAIT value is reached the ERRRIS interrupt status bit is set in the **EPIRIS** register. When this field is clear, the transaction can be held off forever without a system interrupt. **Note:** When the MODE field is configured to be 0x2 and the **BLKEN** bit is set in the **EPICFG** register, enabling HB8 mode, this field defaults to 0xFF.

Bit/Field	Name	Type	Reset	Description
7:6	WRWS	R/W	0x0	Write Wait States This field adds wait states to the data phase of CS0n (the address phase is not affected). The effect is to delay the rising edge of WRn (or the falling edge of WR). Each wait state adds 2 EPI clock cycles to the access time. Value Description 0x0 Active WRn is 2 EPI clocks. 0x1 Active WRn is 4 EPI clocks. 0x2 Active WRn is 6 EPI clocks. 0x3 Active WRn is 8 EPI clocks. This field is used in conjunction with the **EPIBAUD** register.
5:4	RDWS	R/W	0x0	Read Wait States This field adds wait states to the data phase of CS0n (the address phase is not affected). The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD). Each wait state adds 2 EPI clock cycles to the access time. Value Description 0x0 Active RDn is 2 EPI clocks. 0x1 Active RDn is 4 EPI clocks. 0x2 Active RDn is 6 EPI clocks. 0x3 Active RDn is 8 EPI clocks. This field is used in conjunction with the **EPIBAUD** register
3:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1:0	MODE	R/W	0x0	Host Bus Sub-Mode This field determines which of four Host Bus 8 sub-modes to use. Sub-mode use is determined by the connected external peripheral. See Table 9-5 on page 464 for information on how this bit field affects the operation of the EPI signals. Value Description 0x0 ADMUX – AD[7:0] Data and Address are muxed. 0x1 ADNONMUX – D[7:0] Data and address are separate. 0x2 Continuous Read - D[7:0] This mode is the same as ADNONMUX, but uses address switch for multiple reads instead of OEn strobing. 0x3 XFIFO – D[7:0] This mode adds XFIFO controls with sense of XFIFO full and XFIFO empty. This mode uses no address or ALE.

Register 5: EPI Host-Bus 16 Configuration (EPIHB16CFG), offset 0x010

Important: The MODE field in the **EPICFG** register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access **EPIHB16CFG**, the MODE field must be 0x3.

The Host Bus 16 sub-configuration register is activated when the HB16 mode is selected. The HB16 mode supports muxed address/data (overlay of lower 16 address and all 16 data pins), separated address/data, and address-less FIFO mode. Note that this register is reset when the MODE field in the **EPICFG** register is changed. If another mode is selected and the HB16 mode is selected again, the values must be reinitialized.

This mode is intended to support SRAMs, Flash memory (read), FIFOs, and CPLDs/FPGAs, and devices with an MCU/HostBus slave or 16-bit FIFO interface support.

Refer to Table 9-6 on page 466 for information on signal configuration controlled by this register and the **EPIHB16CFG2** register.

If less address pins are required, the corresponding AFSEL bit (page 419) should not be enabled so the EPI controller does not drive those pins, and they are available as standard GPIOs.

EPI Host-Bus 16 Mode can be configured to use one to four chip selects with and without the use of ALE. If an alternative to chip selects are required, a chip enable can be handled in one of three ways:

1. Manually control via GPIOs.
2. Associate one or more upper address pins to CE. Because CE is normally CEn, lower addresses are not used. For example, if pins EPI0S27 and EPI0S26 are used for Device 1 and 0 respectively, then address 0x6800.0000 accesses Device 0 (Device 1 has its CEn high), and 0x6400.0000 accesses Device 1 (Device 0 has its CEn high). The pull-up behavior on the corresponding GPIOs must be properly configured to ensure that the pins are disabled when the interface is not in use.
3. With certain SRAMs, the ALE can be used as CEn because the address remains stable after the ALE strobe. The subsequent WRn or RDn signals write or read when ALE is low thus providing CEn functionality.

EPI Host-Bus 16 Configuration (EPIHB16CFG)

Base 0x400D.0000
Offset 0x010
Type R/W, reset 0x0000.FF00

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAXWAIT								WRWS		RDWS		reserved	BSEL	MODE	
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RO	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23	XFFEN	R/W	0	External FIFO FULL Enable Value Description 0 No effect. 1 An external FIFO full signal can be used to control write cycles. If this bit is set and the FFULL signal is high, XFIFO writes are stalled.
22	XFEEN	R/W	0	External FIFO EMPTY Enable Value Description 1 An external FIFO empty signal can be used to control read cycles. If this bit is set and the FEMPTY signal is high, XFIFO reads are stalled. 0 No effect.
21	WRHIGH	R/W	0	WRITE Strobe Polarity Value Description 0 The WRITE strobe for CS0n is WRn (active Low). 1 The WRITE strobe for CS0n is WR (active High).
20	RDHIGH	R/W	0	READ Strobe Polarity Value Description 0 The READ strobe for CS0n is RDn (active Low). 1 The READ strobe is RD (active High).
19:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	MAXWAIT	R/W	0xFF	Maximum Wait This field defines the maximum number of external clocks to wait while an external FIFO ready signal is holding off a transaction (FFULL and FEMPTY). When this field is clear, the transaction can be held off forever without a system interrupt. **Note:** When the MODE field is configured to be 0x3 and the **BLKEN** bit is set in the **EPICFG** register, enabling HB16 mode, this field defaults to 0xFF.

Bit/Field	Name	Type	Reset	Description
7:6	WRWS	R/W	0x0	Write Wait States This field adds wait states to the data phase of CS0n (the address phase is not affected). The effect is to delay the rising edge of WRn (or the falling edge of WR). Each wait state adds 2 EPI clock cycles to the access time. Value Description 0x0 Active WRn is 2 EPI clocks. 0x1 Active WRn is 4 EPI clocks. 0x2 Active WRn is 6 EPI clocks. 0x3 Active WRn is 8 EPI clocks. This field is used in conjunction with the **EPIBAUD** register.
5:4	RDWS	R/W	0x0	Read Wait States This field adds wait states to the data phase of CS0n (the address phase is not affected). The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD). Each wait state adds 2 EPI clock cycles to the access time. Value Description 0x0 Active RDn is 2 EPI clocks. 0x1 Active RDn is 4 EPI clocks. 0x2 Active RDn is 6 EPI clocks. 0x3 Active RDn is 8 EPI clocks. This field is used in conjunction with the **EPIBAUD** register
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	BSEL	R/W	0	Byte Select Configuration This bit enables byte select operation. Value Description 0 No Byte Selects Data is read and written as 16 bits. 1 Enable Byte Selects Two EPI signals function as byte select signals to allow 8-bit transfers. See Table 9-6 on page 466 for details on which EPI signals are used.

Bit/Field	Name	Type	Reset	Description
1:0	MODE	R/W	0x0	Host Bus Sub-Mode This field determines which of three Host Bus 16 sub-modes to use. Sub-mode use is determined by the connected external peripheral. See Table 9-6 on page 466 for information on how this bit field affects the operation of the EPI signals. Value Description 0x0 ADMUX – AD[15:0] Data and Address are muxed. 0x1 ADNONMUX – D[15:0] Data and address are separate. This mode is not practical in HB16 mode for normal peripherals because there are generally not enough address bits available. 0x2 Continuous Read - D[15:0] This mode is the same as ADNONMUX, but uses address switch for multiple reads instead of OEn strobing. This mode is not practical in HB16 mode for normal SRAMs because there are generally not enough address bits available. 0x3 XFIFO – D[15:0] This mode adds XFIFO controls with sense of XFIFO full and XFIFO empty. This mode uses no address or ALE.

Register 6: EPI General-Purpose Configuration (EPIGPCFG), offset 0x010

Important: The MODE field in the **EPICFG** register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access **EPIGPCFG**, the MODE field must be 0x0.

The RD2CYC bit must be set at all times in General-Purpose mode to ensure proper operation.

The General-Purpose configuration register is used to configure the control, data, and address pins. This mode can be used for custom interfaces with FPGAs, CPLDs, and for digital data acquisition and actuator control. Note that this register is reset when the MODE field in the **EPICFG** register is changed. If another mode is selected and the General-purpose mode is selected again, the register the values must be reinitialized.

This mode is designed for 3 general types of use:

- Extremely high-speed clocked interfaces to FPGAs and CPLDs, with 3 sizes of data and optional address. Framing and clock-enable permit more optimized interfaces.
- General parallel GPIO. From 1 to 32 pins may be written or read, with the speed precisely controlled by the baud rate in the **EPIBAUD** register (when used with the NBRFIFO and/or the WFIFO) or by rate of accesses from software or µDMA.
- General custom interfaces of any speed.

The configuration allows for choice of an output clock (free running or gated), a framing signal (with frame size), a ready input (to stretch transactions), read and write strobes, address of varying sizes, and data of varying sizes. Additionally, provisions are made for splitting address and data phases on the external interface.

EPI General-Purpose Configuration (EPIGPCFG)

Base 0x400D.0000
Offset 0x010
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CLKPIN	CLKGATE	reserved	RDYEN	FRMPIN	FRM50	FRMCNT				RW	reserved	WR2CYC	RD2CYC	reserved	
Type	R/W	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RO	R/W	R/W	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit/Field	Name	Type	Reset	Description
31	CLKPIN	R/W	0	Clock Pin

Value	Description
0	No clock output.
1	EPI0S31 functions as the EPI clock output.

The EPI clock is generated from the COUNT0 field in the **EPIBAUD** register (as is the system clock which is divided down from it).

Bit/Field	Name	Type	Reset	Description
30	CLKGATE	R/W	0	Clock Gated Value Description 0 The EPI clock is free running. 1 The EPI clock is output only when there is data to write or read (current transaction); otherwise the EPI clock is held low. Note that EPI0S27 is an iRDY signal if RDYEN is set. CLKGATE is ignored if CLKPIN is 0 or if the COUNT0 field in the **EPIBAUD** register is cleared.
29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	RDYEN	R/W	0	Ready Enable Value Description 0 The external peripheral does not drive an iRDY signal and is assumed to be ready always. 1 The external peripheral drives an iRDY signal into pin EPI0S27. The ready enable signal may only be used with a free-running EPI clock (CLKGATE=0). The external iRDY signal is sampled on the falling edge of the EPI clock. Setup and hold times must be met to ensure registration on the next falling EPI clock edge. This bit is ignored if CLKPIN is 0 or CLKGATE is 1.
27	FRMPIN	R/W	0	Framing Pin Value Description 0 No framing signal is output. 1 A framing signal is output on EPI0S30. Framing has no impact on data itself, but forms a context for the external peripheral. When used with a free-running EPI clock, the FRAME signal forms the valid signal. When used with a gated EPI clock, it is usually used to form a frame size.
26	FRM50	R/W	0	50/50 Frame Value Description 0 The FRAME signal is output as a single pulse, and then held low for the count. 1 The FRAME signal is output as 50/50 duty cycle using count (see FRMCNT). This bit is ignored if FRMPIN is 0.

Bit/Field	Name	Type	Reset	Description
25:22	FRMCNT	R/W	0x0	Frame Count This field specifies the size of the frame in EPI clocks. The frame counter is used to determine the frame size. The count is FRMCNT+1. So, a FRMCNT of 0 forms a pure transaction valid signal (held high during transactions, low otherwise). A FRMCNT of 0 with FRM50 set inverts the FRAME signal on each transaction. A FRMCNT of 1 means the FRAME signal is inverted every other transaction; a value of 15 means every sixteenth transaction. If FRM50 is set, the frame is held high for FRMCNT+1 transactions, then held low for that many transactions, and so on. If FRM50 is clear, the frame is pulsed high for one EPI clock and then low for FRMCNT EPI clocks. This field is ignored if FRMPIN is 0.
21	RW	R/W	0	Read and Write Value Description 0 RD and WR strobes are not output. 1 RD and WR strobes are asserted on EPI0S29 and EPI0S28. RD is asserted high on the rising edge of the EPI clock when a read is being performed. WR is asserted high on the rising edge of the EPI clock when a write is being performed This bit is forced to 1 when RD2CYC and/or WR2CYC is 1.
20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	WR2CYC	R/W	0	2-Cycle Writes Value Description 0 Data is output on the same EPI clock cycle as the address. 1 Writes are two EPI clock cycles long, with address on one EPI clock cycle (with the WR strobe asserted) and data written on the following EPI clock cycle (with WR strobe de-asserted). The next address (if any) is in the cycle following. When this bit is set, then the RW bit is forced to be set.
18	RD2CYC	R/W	0	2-Cycle Reads Value Description 0 Data is captured on the EPI clock cycle with READ strobe asserted. 1 Reads are two EPI clock cycles, with address on one EPI clock cycle (with the RD strobe asserted) and data captured on the following EPI clock cycle (with the RD strobe de-asserted). The next address (if any) is in the cycle following. When this bit is set, then the RW bit is forced to be set. **Caution – This bit must be set at all times in General-Purpose mode to ensure proper operation.**

Bit/Field	Name	Type	Reset	Description
17:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	MAXWAIT	R/W	0x00	Maximum Wait This field defines the maximum number of EPI clocks to wait while the iRDY signal (see RDYEN) is holding off a transaction. If this field is 0, the transaction is held forever. If the maximum wait of 255 clocks (MAXWAIT=0xFF) is exceeded, an error interrupt occurs and the transaction is aborted/ignored. **Note:** When the MODE field is configured to be 0x0 and the **BLKEN** bit is set in the **EPICFG** register, enabling General-Purpose mode, this field defaults to 0xFF.
7:6	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:4	ASIZE	R/W	0x0	Address Bus Size This field defines the size of the address bus. The address can be up to 4-bits wide with a 24-bit data bus, up to 12-bits wide with a 16-bit data bus, and up to 20-bits wide with an 8-bit data bus. If the full address bus is not used, use the least significant address bits. Any unused address bits can be used as GPIOs by clearing the AFSEL bit for the corresponding GPIOs. Also, if RDYEN is 1, then the address sizes are 1 smaller (3, 11, 19). The values are: Value Description 0x0 No address 0x1 Up to 4 bits wide. 0x2 Up to 12 bits wide. This size cannot be used with 24-bit data. 0x3 Up to 20 bits wide. This size cannot be used with data sizes other than 8.
3:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
1:0	DSIZE	R/W	0x0	Size of Data Bus This field defines the size of the data bus (starting at EPI0S0). Subsets of these numbers can be created by clearing the AFSEL bit for the corresponding GPIOs. Note that size 32 may not be used with clock, frame, address, or other control. The values are: Value Description 0x0 8 Bits Wide (EPI0S0 to EPI0S7) 0x1 16 Bits Wide (EPI0S0 to EPI0S15) 0x2 24 Bits Wide (EPI0S0 to EPI0S23) 0x3 32 Bits Wide (EPI0S0 to EPI0S31) This size may not be used with an EPI clock. This value is normally used for acquisition input and actuator control as well as other general-purpose uses that require 32 bits per direction.

Register 7: EPI Host-Bus 8 Configuration 2 (EPIHB8CFG2), offset 0x014

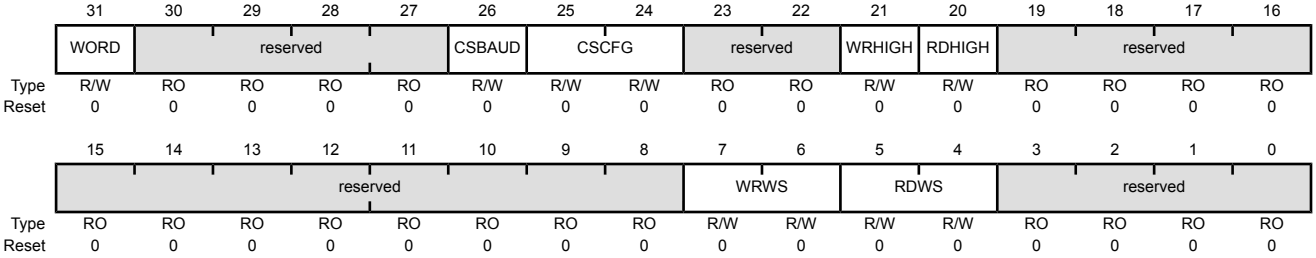
Important: The MODE field in the **EPICFG** register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access **EPIHB8CFG2**, the MODE field must be 0x2.

This register is used to configure operation while in Host-Bus 8 mode. Note that this register is reset when the MODE field in the **EPICFG** register is changed. If another mode is selected and the Host-Bus 8 mode is selected again, the values must be reinitialized.

EPI Host-Bus 8 Configuration 2 (EPIHB8CFG2)

Base 0x400D.0000
Offset 0x014
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31	WORD	R/W	0	Word Access Mode By default, the EPI controller uses data bits [7:0] for Host-Bus 8 accesses. When using Word Access mode, the EPI controller can automatically route bytes of data onto the correct byte lanes such that data can be stored in bits [31:8]. When WORD is set, short and long variables can be used in C programs. Value Description 0 Word Access mode is disabled. 1 Word Access mode is enabled.
30:27	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	CSBAUD	R/W	0	Chip Select Baud Rate Value Description 0 Same Baud Rate Both CS0n and CS1n use the baud rate for the external bus that is defined by the COUNT0 field in the **EPIBAUD** register. 1 Different Baud Rates CS0n uses the baud rate for the external bus that is defined by the COUNT0 field in the **EPIBAUD** register. CS1n uses the baud rate defined by the COUNT1 field in the **EPIBAUD** register.

Bit/Field	Name	Type	Reset	Description
25:24	CSCFG	R/W	0x0	Chip Select Configuration This field controls the chip select options, including an ALE format, a single chip select, two chip selects, and an ALE combined with two chip selects. Value Description 0x0 ALE Configuration EPI0S30 is used as an address latch (ALE). The ALE signal is generally used when the address and data are muxed (HB8MODE field in the **EPIHB8CFG** register is 0x0). The ALE signal is used by an external latch to hold the address through the bus cycle. 0x1 CSn Configuration EPI0S30 is used as a Chip Select (CSn). When using this mode, the address and data are generally not muxed (HB8MODE field in the **EPIHB8CFG** register is 0x1). However, if address and data muxing is needed, the WR signal (EPI0S29) and the RD signal (EPI0S28) can be used to latch the address when CSn is low. 0x2 Dual CSn Configuration EPI0S30 is used as CS0n and EPI0S27 is used as CS1n. Whether CS0n or CS1n is asserted is determined by two methods. If only external RAM or external PER is enabled in the address map, the most significant address bit for a respective external address map controls CS0n or CS1n. If both external RAM and external PER is enabled, CS0n is mapped to PER and CS1n is mapped to RAM. This configuration can be used for a RAM bank split between 2 devices as well as when using both an external RAM and an external peripheral. 0x3 ALE with Dual CSn Configuration EPI0S30 is used as address latch (ALE), EPI0S27 is used as CS1n, and EPI0S26 is used as CS0n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map.
23:22	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
21	WRHIGH	R/W	0	CS1n WRITE Strobe Polarity This field is used if the CSBAUD bit in the **EPIHBnCFG2** register is enabled. Value Description 0 The WRITE strobe for CS1n accesses is WRn (active Low). 1 The WRITE strobe for CS1n accesses is WR (active High).

Bit/Field	Name	Type	Reset	Description
20	RDHIGH	R/W	0	CS1n READ Strobe Polarity This field is used if the CSBAUD bit in the **EPIHBnCFG2** register is enabled. Value Description 0 The READ strobe for CS1n accesses is RDn (active Low). 1 The READ strobe for CS1n accesses is RD (active High).
19:8	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:6	WRWS	R/W	0x0	CS1n Write Wait States This field adds wait states to the data phase of CS1n accesses (the address phase is not affected). The effect is to delay the rising edge of WRn (or the falling edge of WR). Each wait state encoding adds 2 EPI clock cycles to the access time. Value Description 0x0 Active WRn is 2 EPI clocks. 0x1 Active WRn is 4 EPI clocks 0x2 Active WRn is 6 EPI clocks 0x3 Active WRn is 8 EPI clocks
5:4	RDWS	R/W	0x0	CS1n Read Wait States This field adds wait states to the data phase of CS1n accesses (the address phase is not affected). The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD). Each wait state encoding adds 2 EPI clock cycles to the access time. Value Description 0x0 Active RDn is 2 EPI clocks 0x1 Active RDn is 4 EPI clocks 0x2 Active RDn is 6 EPI clocks 0x3 Active RDn is 8 EPI clocks
3:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 8: EPI Host-Bus 16 Configuration 2 (EPIHB16CFG2), offset 0x014

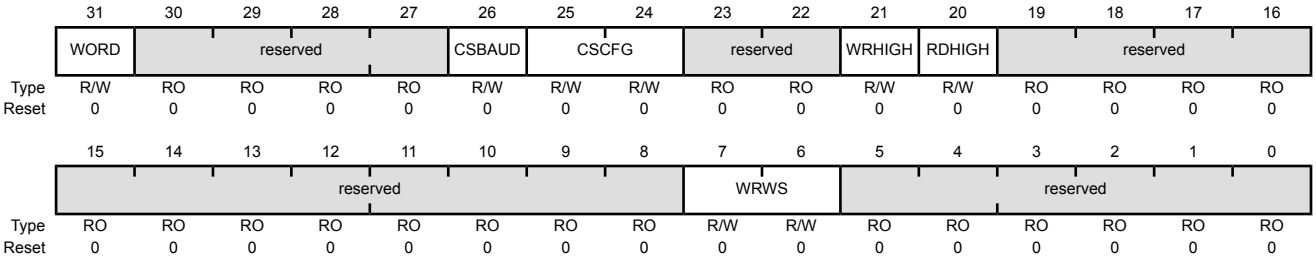
Important: The MODE field in the **EPICFG** register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access **EPIHB16CFG2**, the MODE field must be 0x3.

This register is used to configure operation while in Host-Bus 16 mode. Note that this register is reset when the MODE field in the **EPICFG** register is changed. If another mode is selected and the Host-Bus 16 mode is selected again, the values must be reinitialized.

EPI Host-Bus 16 Configuration 2 (EPIHB16CFG2)

Base 0x400D.0000
Offset 0x014
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31	WORD	R/W	0	Word Access Mode By default, the EPI controller uses data bits [15:0] for Host-Bus 16 accesses. When using Word Access mode, the EPI controller can automatically route bytes of data onto the correct byte lanes such that data can be stored in bits [31:16]. When WORD is set, long variables can be used in C programs. Value Description 0 Word Access mode is disabled. 1 Word Access mode is enabled.
30:27	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	CSBAUD	R/W	0	Chip Select Baud Rate Value Description 0 Same Baud Rate All CSn use the baud rate for the external bus that is defined by the COUNT0 field in the **EPIBAUD** register. 1 Different Baud Rates CS0n uses the baud rate for the external bus that is defined by the COUNT0 field in the **EPIBAUD** register. CS1n uses the baud rate defined by the COUNT1 field in the **EPIBAUD** register.

Bit/Field	Name	Type	Reset	Description
25:24	CSCFG	R/W	0x0	Chip Select Configuration This field controls the chip select options, including an ALE format, a single chip select, two chip selects, and an ALE combined with two chip selects. Value Description 0x0 ALE Configuration EPI0S30 is used as an address latch (ALE). When using this mode, the address and data should be muxed (HB16MODE field in the **EPIHB16CFG** register should be configured to 0x0). If needed, the address can be latched by external logic. 0x1 CSn Configuration EPI0S30 is used as a Chip Select (CSn). When using this mode, the address and data should not be muxed (MODE field in the **EPIHB16CFG** register should be configured to 0x1). In this mode, the WR signal (EPI0S29) and the RD signal (EPI0S28) are used to latch the address when CSn is low. 0x2 Dual CSn Configuration EPI0S30 is used as CS0n and EPI0S27 is used as CS1n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map. This configuration can be used for a RAM bank split between 2 devices as well as when using both an external RAM and an external peripheral. 0x3 ALE with Dual CSn Configuration EPI0S30 is used as address latch (ALE), EPI0S27 is used as CS1n, and EPI0S26 is used as CS0n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map.
23:22	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
21	WRHIGH	R/W	0	CS1n WRITE Strobe Polarity This field is used if CSBAUD bit of the **EPIHBnCFG2** register is enabled. Value Description 0 The WRITE strobe for CS1n accesses is WRn (active Low). 1 The WRITE strobe for CS1n accesses is WR (active High).
20	RDHIGH	R/W	0	CS1n READ Strobe Polarity This field is used if CSBAUD bit of the **EPIHBnCFG2** register is enabled. Value Description 0 The READ strobe for CS1n accesses is RDn (active Low). 1 The READ strobe for CS1n accesses is RD (active High).
19:8	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
7:6	WRWS	R/W	0x0	CS1n Write Wait States This field adds wait states to the data phase of CS1n accesses (the address phase is not affected). The effect is to delay the rising edge of WRn (or the falling edge of WR). Each wait state encoding adds 2 EPI clock cycles to the access time. Value Description 0x0 Active WRn is 2 EPI clocks 0x1 Active WRn is 4 EPI clocks. 0x2 Active WRn is 6 EPI clocks 0x3 Active WRn is 8 EPI clocks
5:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 9: EPI General-Purpose Configuration 2 (EPIGPCFG2), offset 0x014

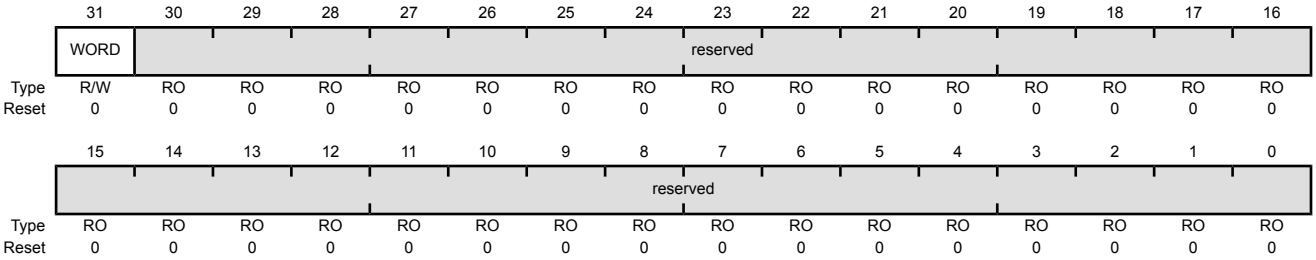
Important: The MODE field in the **EPICFG** register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access **EPIGPCFG2**, the MODE field must be 0x0.

This register is used to configure operation while in General-Purpose mode. Note that this register is reset when the MODE field in the **EPICFG** register is changed. If another mode is selected and the General-Purpose mode is selected again, the values must be reinitialized.

EPI General-Purpose Configuration 2 (EPIGPCFG2)

Base 0x400D.0000
Offset 0x014
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31	WORD	R/W	0	Word Access Mode By default, the EPI controller uses data bits [7:0] when the DSIZE field in the **EPIGPCFG** register is 0x0; data bits [15:0] when the DSIZE field is 0x1; data bits [23:0] when the DSIZE field is 0x2; and data bits [31:0] when the DSIZE field is 0x3. When using Word Access mode, the EPI controller can automatically route bytes of data onto the correct byte lanes such that data can be stored in bits [31:8] for DSIZE=0x0 and bits [31:16] for DSIZE=0x1. For DSIZE=0x2 or 0x3, this bit must be clear. Value Description 0 Word Access mode is disabled. 1 Word Access mode is enabled.
30:0	reserved	RO	0x000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 10: EPI Address Map (EPIADDRMAP), offset 0x01C

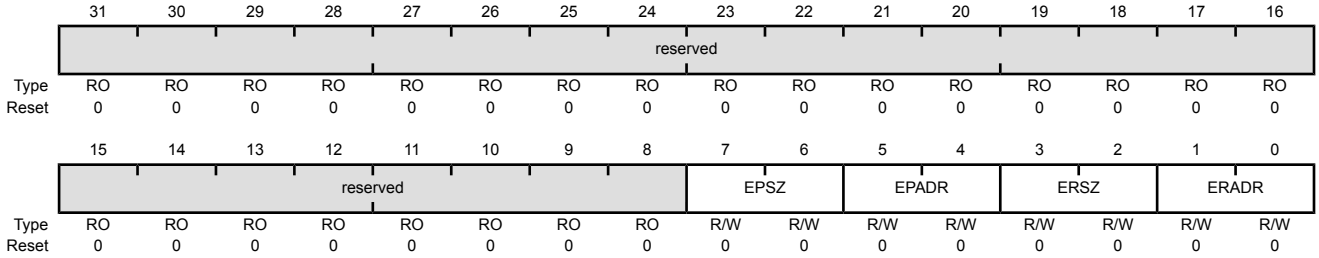
This register enables address mapping. The EPI controller can directly address memory and peripherals. In addition, the EPI controller supports address mapping to allow indirect accesses in the External RAM and External Peripheral areas.

If the external device is a peripheral, including a FIFO or a directly addressable device, the EPSZ and EPADR bit fields should be configured for the address space. If the external device is SDRAM, SRAM, or NOR Flash memory, the ERADR and ERSZ bit fields should be configured for the address space.

If one of the dual chip select modes is selected (CSCFG is 0x2 or 0x3 in the **EPIHBnCFG2** register), both chip selects can share the peripheral or the memory space, or one chip select can use the peripheral space and the other can use the memory space. In the **EPIADDRMAP** register, if the EPADR field is not 0x0 and the ERADR field is 0x0, then the address specified by EPADR is used for both chip selects, with CS0n being asserted when the MSB of the address range is 0 and CS1n being asserted when the MSB of the address range is 1. If the ERADR field is not 0x0 and the EPADR field is 0x0, then the address specified by ERADR is used for both chip selects, with the MSB performing the same delineation. If both the EPADR and the ERADR are not 0x0 , then CS0n is asserted for either address range defined by EPADR and CS1n is asserted for either address range defined by ERADR.

EPI Address Map (EPIADDRMAP)

Base 0x400D.0000
Offset 0x01C
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:6	EPSZ	R/W	0x0	External Peripheral Size This field selects the size of the external peripheral. If the size of the external peripheral is larger, a bus fault occurs. If the size of the external peripheral is smaller, it wraps (upper address bits unused). **Note:** When not using byte selects in Host-Bus 16, data is accessed on 2-byte boundaries. As a result, the available address space is double the amount shown below. Value Description 0x0 256 bytes; lower address range: 0x00 to 0xFF 0x1 64 KB; lower address range: 0x0000 to 0xFFFF 0x2 16 MB; lower address range: 0x00.0000 to 0xFF.FFFF 0x3 512 MB; lower address range: 0x000.0000 to 0x1FFF.FFFF

Bit/Field	Name	Type	Reset	Description
5:4	EPADR	R/W	0x0	External Peripheral Address This field selects address mapping for the external peripheral area. Value Description 0x0 Not mapped 0x1 At 0xA000.0000 0x2 At 0xC000.0000 0x3 reserved
3:2	ERSZ	R/W	0x0	External RAM Size This field selects the size of mapped RAM. If the size of the external memory is larger, a bus fault occurs. If the size of the external memory is smaller, it wraps (upper address bits unused): Value Description 0x0 256 bytes; lower address range: 0x00 to 0xFF 0x1 64 KB; lower address range: 0x0000 to 0xFFFF 0x2 16 MB; lower address range: 0x00.0000 to 0xFF.FFFF 0x3 512 MB; lower address range: 0x000.0000 to 0x1FFF.FFFF
1:0	ERADR	R/W	0x0	External RAM Address Selects address mapping for external RAM area: Value Description 0x0 Not mapped 0x1 At 0x6000.0000 0x2 At 0x8000.0000 0x3 reserved

Register 11: EPI Read Size 0 (EPIRSIZE0), offset 0x020

Register 12: EPI Read Size 1 (EPIRSIZE1), offset 0x030

This register selects the size of transactions when performing non-blocking reads with the **EPIRPSTDn** registers. This size affects how the external address is incremented.

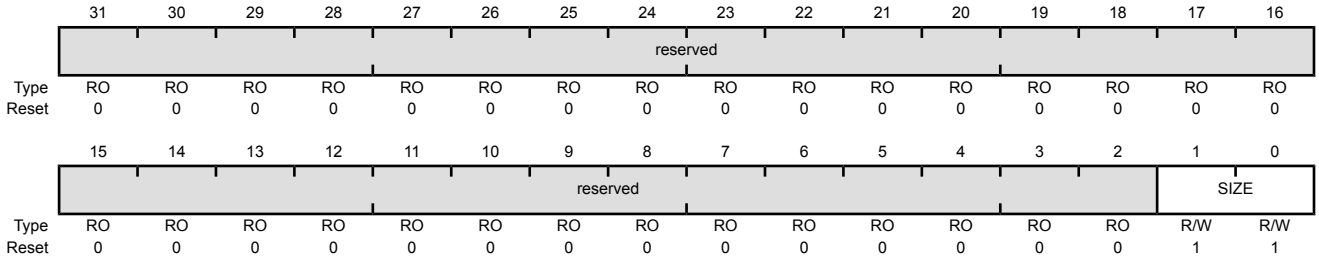
The SIZE field must match the external data width as configured in the **EPIHBnCFG** or **EPIGPCFG** register if the WORD bit is clear in the **EPIHBnCFG2** or **EPIGPCFG2** register. If the WORD bit is set, the SIZE field must be greater than or equal to the external data width.

SDRAM mode uses a 16-bit data interface. If SIZE is 0x1, data is returned on the least significant bits (D[7:0]), and the remaining bits D[31:8] are all zeros, therefore the data on bits D[15:8] is lost. If SIZE is 0x2, data is returned on the least significant bits (D[15:0]), and the remaining bits D[31:16] are all zeros.

Note that changing this register while a read is active has an unpredictable effect.

EPI Read Size 0 (EPIRSIZE0)

Base 0x400D.0000
Offset 0x020
Type R/W, reset 0x0000.0003



Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1:0	SIZE	R/W	0x3	Current Size

Value	Description
0x0	reserved
0x1	Byte (8 bits)
0x2	Half-word (16 bits)
0x3	Word (32 bits)

Register 13: EPI Read Address 0 (EPIRADDR0), offset 0x024

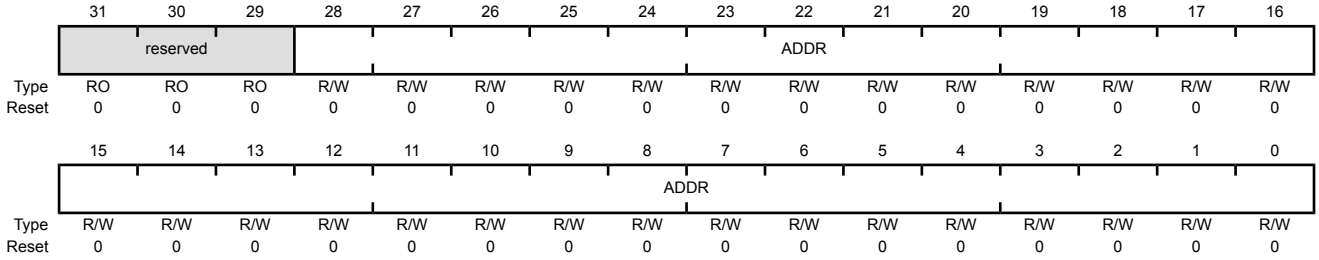
Register 14: EPI Read Address 1 (EPIRADDR1), offset 0x034

This register holds the current address value. When performing non-blocking reads via the **EPIRPSTDn** registers, this register's value forms the address (when used by the mode). That is, when an **EPIRPSTDn** register is written with a non-0 value, this register is used as the first address. After each read, it is incremented by the size specified by the corresponding **EPIRSIZEn** register. Thus at the end of a read, this register contains the next address for the next read. For example, if the last read was 0x20, and the size is word, then the register contains 0x24. When a non-blocking read is cancelled, this register contains the next address that would have been read had it not been cancelled. For example, if reading by bytes and 0x103 had been read but not 0x104, this register contains 0x104. In this manner, the system can determine the number of values in the NBRFIFO to drain.

Note that changing this register while a read is active has an unpredictable effect due to race condition.

EPI Read Address 0 (EPIRADDR0)

Base 0x400D.0000
Offset 0x024
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:29	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28:0	ADDR	R/W	0x000.0000	Current Address Next address to read.

Register 15: EPI Non-Blocking Read Data 0 (EPIRPSTD0), offset 0x028

Register 16: EPI Non-Blocking Read Data 1 (EPIRPSTD1), offset 0x038

This register sets up a non-blocking read via the external interface. A non-blocking read is started by writing to this register with the count (other than 0). Clearing this register terminates an active non-blocking read as well as cancelling any that are pending. This register should always be cleared before writing a value other than 0; failure to do so can cause improper operation. Note that both NBR channels can be enabled at the same time, but NBR channel 0 has the highest priority and channel 1 does not start until channel 0 is finished.

The first address is based on the corresponding **EPIRADDRn** register. The address register is incremented by the size specified by the **EPIRSIZEn** register after each read. If the size is less than a word, only the least significant bits of data are filled into the NBRFIFO; the most significant bits are cleared.

Note that all three registers may be written using one STM instruction, such as with a structure copy in C/C++.

The data may be read from the **EPIREADFIFO** register after the read cycle is completed. The interrupt mechanism is normally used to trigger the FIFO reads via ISR or µDMA.

If the countdown has not reached 0 and the NBRFIFO is full, the external interface waits until a NBRFIFO entry becomes available to continue.

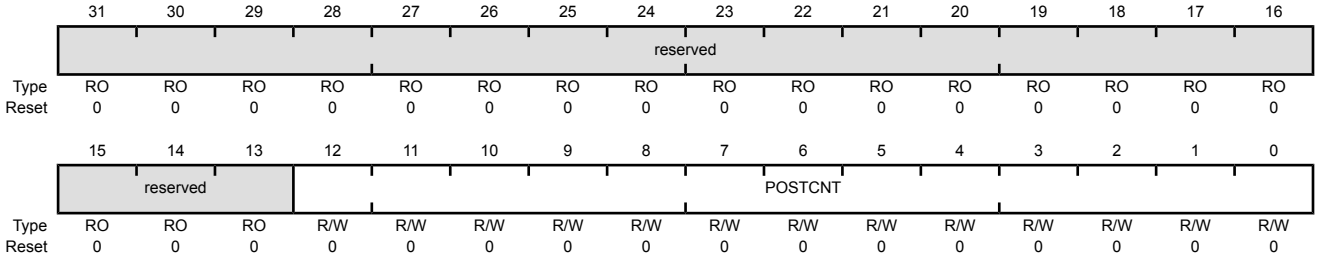
Note: if a blocking read or write is performed through the address mapped area (at 0x6000.0000 through 0xDFFF.FFFF), any current non-blocking read is paused (at the next safe boundary), and the blocking request is inserted. After completion of any blocking reads or writes, the non-blocking reads continue from where they were paused.

The other way to read data is via the address mapped locations (see the **EPIADDRMAP** register), but this method is blocking (core or µDMA waits until result is returned).

To cancel a non-blocking read, clear this register. To make sure that all values read are drained from the NBRFIFO, the **EPISTAT** register must be consulted to be certain that bits NBRBUSY and ACTIVE are cleared. One of these registers should not be cleared until either the other **EPIRPSTDn** register becomes active or the external interface is not busy. At that point, the corresponding **EPIRADDRn** register indicates how many values were read.

EPI Non-Blocking Read Data 0 (EPIRPSTD0)

Base 0x400D.0000
Offset 0x028
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:13	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
12:0	POSTCNT	R/W	0x000	Post Count A write of a non-zero value starts a read operation for that count. Note that it is the software's responsibility to handle address wrap-around. Reading this register provides the current count. A write of 0 cancels a non-blocking read (whether active now or pending). Prior to writing a non-zero value, this register must first be cleared.

Register 17: EPI Status (EPISTAT), offset 0x060

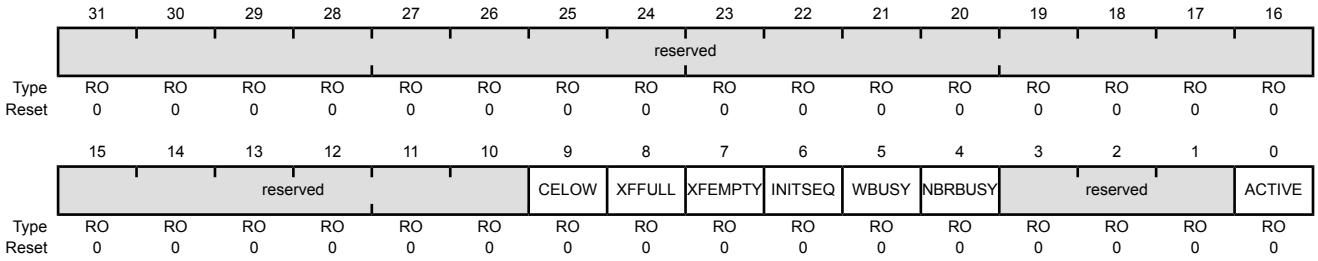
This register indicates which non-blocking read register is currently active; it also indicates whether the external interface is busy performing a write or non-blocking read (it cannot be performing a blocking read, as the bus would be blocked and as a result, this register could not be accessed).

This register is useful to determining which non-blocking read register is active when both are loaded with values and when implementing sequencing or sharing.

This register is also useful when canceling non-blocking reads, as it shows how many values were read by the canceled side.

EPI Status (EPISTAT)

Base 0x400D.0000
Offset 0x060
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	CELOW	RO	0	Clock Enable Low This bit provides information on the clock status when in general-purpose mode and the RDYEN bit is set. Value Description 0 The external device is not gating the clock.
8	XFFULL	RO	0	External FIFO Full This bit provides information on the XFIFO when in the FIFO sub-mode of the Host Bus n mode with the XFFEN bit set in the **EPIHBnCFG** register. The EPI0S26 signal reflects the status of this bit. Value Description 0 The external device is not gating the clock. 1 The XFIFO is signaling as full (the FIFO full signal is high). Attempts to write in this case are stalled until the XFIFO full signal goes low or the counter times out as specified by the MAXWAIT field.

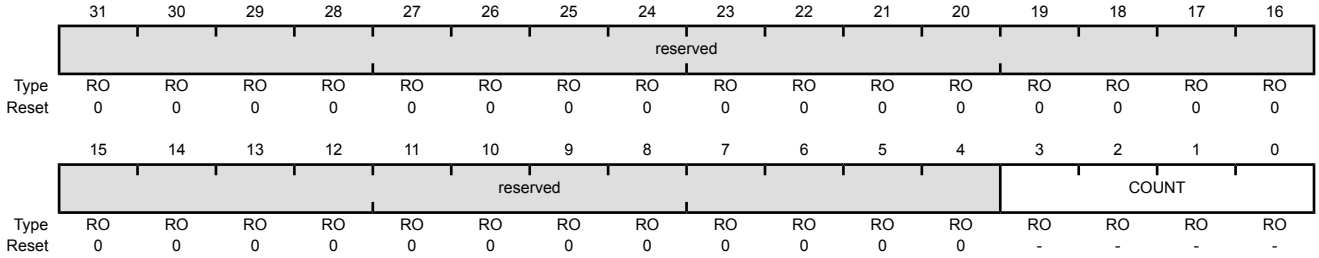
Bit/Field	Name	Type	Reset	Description
7	XFEMPTY	RO	0	External FIFO Empty This bit provides information on the XFIFO when in the FIFO sub-mode of the Host Bus n mode with the XFEEN bit set in the **EPIHBnCFG** register. The EPI0S27 signal reflects the status of this bit. Value Description 0 The external device is not gating the clock. 1 The XFIFO is signaling as empty (the FIFO empty signal is high). Attempts to read in this case are stalled until the XFIFO empty signal goes low or the counter times out as specified by the MAXWAIT field.
6	INITSEQ	RO	0	Initialization Sequence Value Description 0 The SDRAM interface is not in the wakeup period. 1 The SDRAM interface is running through the wakeup period (greater than 100 µs). If an attempt is made to read or write the SDRAM during this period, the access is held off until the wakeup period is complete.
5	WBUSY	RO	0	Write Busy Value Description 0 The external interface is not performing a write. 1 The external interface is performing a write.
4	NBRBUSY	RO	0	Non-Blocking Read Busy Value Description 0 The external interface is not performing a non-blocking read. 1 The external interface is performing a non-blocking read, or if the non-blocking read is paused due to a write.
3:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	ACTIVE	RO	0	Register Active Value Description 0 If NBRBUSY is set, the **EPIRPSTD0** register is active. If the NBRBUSY bit is clear, then neither **EPIRPSTDx** register is active. 1 The **EPIRPSTD1** register is active.

Register 18: EPI Read FIFO Count (EPIRFIFOCNT), offset 0x06C

This register returns the number of values in the NBRFIFO (the data in the NBRFIFO can be read via the **EPIREADFIFO** register). A race is possible, but that only means that more values may come in after this register has been read.

EPI Read FIFO Count (EPIRFIFOCNT)

Base 0x400D.0000
Offset 0x06C
Type RO, reset -



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	COUNT	RO	-	FIFO Count Number of filled entries in the NBRFIFO.

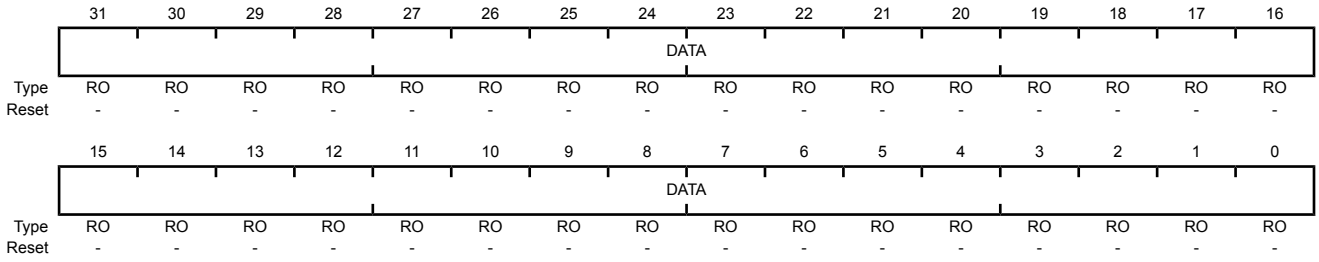
Register 19: EPI Read FIFO (EPIREADFIFO), offset 0x070
Register 20: EPI Read FIFO Alias 1 (EPIREADFIFO1), offset 0x074
Register 21: EPI Read FIFO Alias 2 (EPIREADFIFO2), offset 0x078
Register 22: EPI Read FIFO Alias 3 (EPIREADFIFO3), offset 0x07C
Register 23: EPI Read FIFO Alias 4 (EPIREADFIFO4), offset 0x080
Register 24: EPI Read FIFO Alias 5 (EPIREADFIFO5), offset 0x084
Register 25: EPI Read FIFO Alias 6 (EPIREADFIFO6), offset 0x088
Register 26: EPI Read FIFO Alias 7 (EPIREADFIFO7), offset 0x08C

Important: This register is read-sensitive. See the register description for details.

This register returns the contents of the NBRFIFO or 0 if the NBRFIFO is empty. Each read returns the data that is at the top of the NBRFIFO, and then empties that value from the NBRFIFO. The alias registers can be used with the LDMIA instruction for more efficient operation (for up to 8 registers). See *Cortex™-M3/M4 Instruction Set Technical User's Manual* for more information on the LDMIA instruction.

EPI Read FIFO (EPIREADFIFO)

Base 0x400D.0000
Offset 0x070
Type RO, reset -



Bit/Field	Name	Type	Reset	Description
31:0	DATA	RO	-	Reads Data This field contains the data that is at the top of the NBRFIFO. After being read, the NBRFIFO entry is removed.

Register 27: EPI FIFO Level Selects (EPIFIFOLVL), offset 0x200

This register allows selection of the FIFO levels which trigger an interrupt to the interrupt controller or, more efficiently, a DMA request to the µDMA. The NBRFIFO select triggers on fullness such that it triggers on match or above (more full). The WFIFO triggers on emptiness such that it triggers on match or below (less entries).

It should be noted that the FIFO triggers are not identical to other such FIFOs in Stellaris peripherals. In particular, empty and full triggers are provided to avoid wait states when using blocking operations.

The settings in this register are only meaningful if the µDMA is active or the interrupt is enabled.

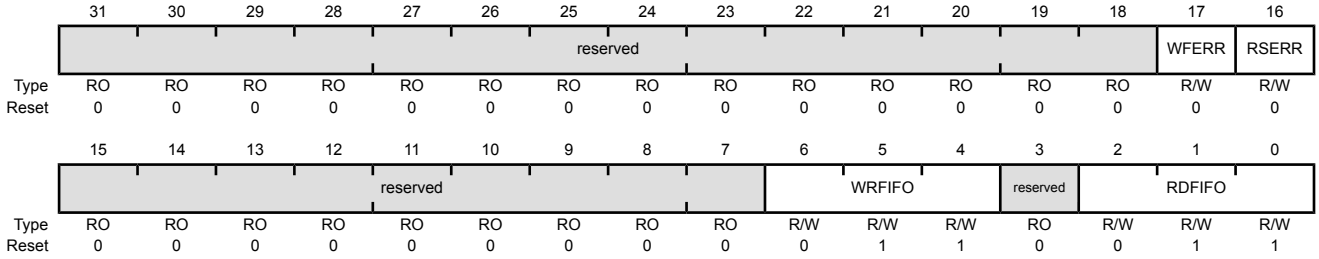
Additionally, this register allows protection against writes stalling and notification of performing blocking reads which stall for extra time due to preceding writes. The two functions behave in a non-orthogonal way because read and write are not orthogonal.

The write error bit configures the system such that an attempted write to an already full WFIFO abandons the write and signals an error interrupt to prevent accidental latencies due to stalling writes.

The read error bit configures the system such that after a read has been stalled due to any preceding writes in the WFIFO, the error interrupt is generated. Note that the excess stall is not prevented, but an interrupt is generated after the fact to notify that it has happened.

EPI FIFO Level Selects (EPIFIFOLVL)

Base 0x400D.0000
Offset 0x200
Type R/W, reset 0x0000.0033



Bit/Field	Name	Type	Reset	Description
31:18	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
17	WFERR	R/W	0	Write Full Error

Value	Description
0	The Write Full error interrupt is disabled. Writes are stalled when the WFIFO is full until a space becomes available but an error is not generated. Note that the Cortex-M3 write buffer may hide that stall if no other memory transactions are attempted during that time.
1	This bit enables the Write Full error interrupt (WTFULL in the **EPIEISC** register) to be generated when a write is attempted and the WFIFO is full. The write stalls until a WFIFO entry becomes available.

Bit/Field	Name	Type	Reset	Description
16	RSERR	R/W	0	Read Stall Error Value Description 0 The Read Stalled error interrupt is disabled. Reads behave as normal and are stalled until any preceding writes have completed and the read has returned a result. 1 This bit enables the Read Stalled error interrupt (RSTALL in the **EPIEISC** register) to be generated when a read is attempted and the WFIFO is not empty. The read is still stalled during the time the WFIFO drains, but this error notifies the application that this excess delay has occurred. Note that the configuration of this bit has no effect on non-blocking reads.
15:7	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:4	WRFIFO	R/W	0x3	Write FIFO This field configures the trigger point for the WFIFO. Value Description 0x0 Trigger when there are any spaces available in the WFIFO. 0x1 reserved 0x2 Trigger when there are up to 3 spaces available in the WFIFO. 0x3 Trigger when there are up to 2 spaces available in the WFIFO. 0x4 Trigger when there is 1 space available in the WFIFO. 0x5-0x7 reserved
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	RDFIFO	R/W	0x3	Read FIFO This field configures the trigger point for the NBRFIFO. Value Description 0x0 reserved 0x1 Trigger when there are 1 or more entries in the NBRFIFO. 0x2 Trigger when there are 2 or more entries in the NBRFIFO. 0x3 Trigger when there are 4 or more entries in the NBRFIFO. 0x4 Trigger when there are 6 or more entries in the NBRFIFO. 0x5 Trigger when there are 7 or more entries in the NBRFIFO. 0x6 Trigger when there are 8 entries in the NBRFIFO. 0x7 reserved

Register 28: EPI Write FIFO Count (EPIWFIFOCNT), offset 0x204

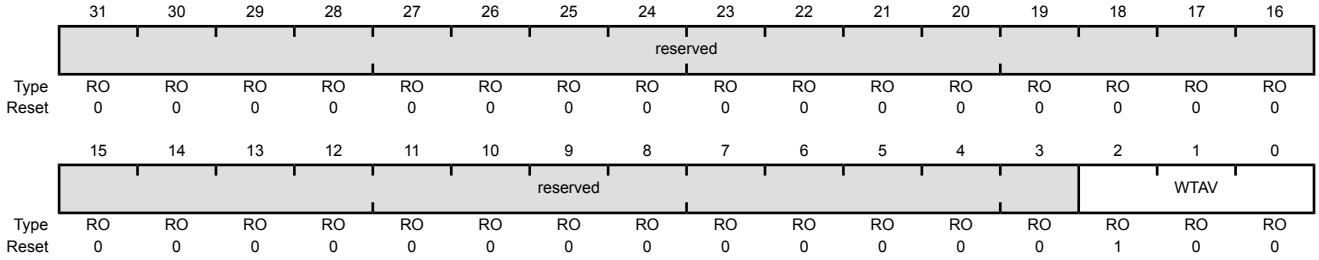
This register contains the number of slots currently available in the WFIFO. This register may be used for polled writes to avoid stalling and for blocking reads to avoid excess stalling (due to undrained writes). An example use for writes may be:

```
for (idx = 0; idx < cnt; idx++) {
while (EPIWFIFOCNT == 0) ;
*ext_ram = *mydata++;
}
```

The above code ensures that writes to the address mapped location do not occur unless the WFIFO has room. Although polling makes the code wait (spinning in the loop), it does not prevent interrupts being serviced due to bus stalling.

EPI Write FIFO Count (EPIWFIFOCNT)

Base 0x400D.0000
Offset 0x204
Type RO, reset 0x0000.0004



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	WTAV	RO	0x4	Available Write Transactions The number of write transactions available in the WFIFO. When clear, a write is stalled waiting for a slot to become free (from a preceding write completing).

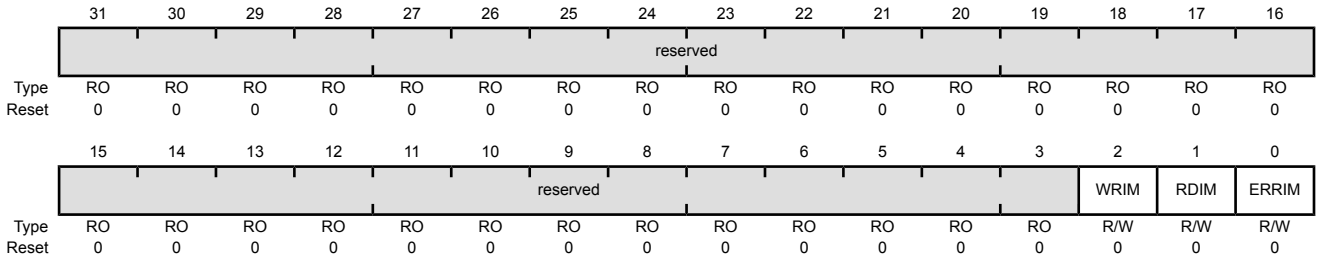
Register 29: EPI Interrupt Mask (EPIIM), offset 0x210

This register is the interrupt mask set or clear register. For each interrupt source (read, write, and error), a mask value of 1 allows the interrupt source to trigger an interrupt to the interrupt controller; a mask value of 0 prevents the interrupt source from triggering an interrupt.

Note that interrupt masking has no effect on µDMA, which operates off the raw source of the read and write interrupts.

EPI Interrupt Mask (EPIIM)

Base 0x400D.0000
Offset 0x210
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	WRIM	R/W	0	Write FIFO Empty Interrupt Mask Value Description 0 WRRIS in the **EPIRIS** register is masked and does not cause an interrupt. 1 WRRIS in the **EPIRIS** register is not masked and can trigger an interrupt to the interrupt controller.
1	RDIM	R/W	0	Read FIFO Full Interrupt Mask Value Description 0 RDRIS in the **EPIRIS** register is masked and does not cause an interrupt. 1 RDRIS in the **EPIRIS** register is not masked and can trigger an interrupt to the interrupt controller.
0	ERRIM	R/W	0	Error Interrupt Mask Value Description 0 ERRIS in the **EPIRIS** register is masked and does not cause an interrupt. 1 ERRIS in the **EPIRIS** register is not masked and can trigger an interrupt to the interrupt controller.

Register 30: EPI Raw Interrupt Status (EPIRIS), offset 0x214

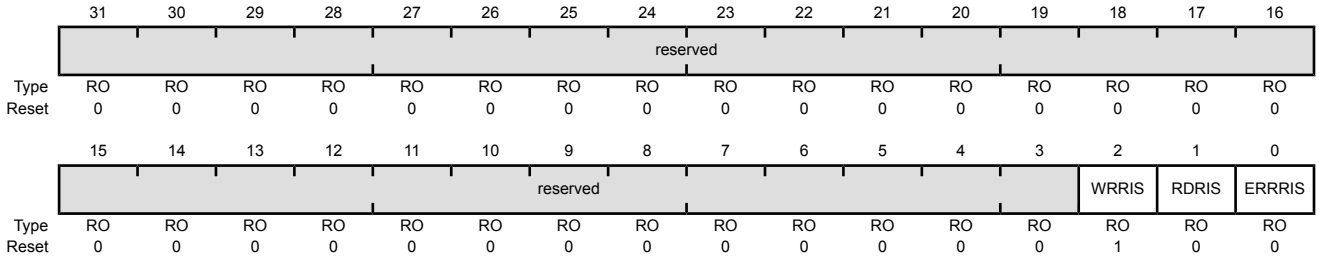
This register is the raw interrupt status register. On a read, it gives the current state of each interrupt source. A write has no effect.

Note that raw status for read and write is set or cleared based on FIFO fullness as controlled by **EPIFIFOLVL**.

Raw status for error is held until the error is cleared by writing to the **EPIEISC** register.

EPI Raw Interrupt Status (EPIRIS)

Base 0x400D.0000
Offset 0x214
Type RO, reset 0x0000.0004



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	WRRIS	RO	1	Write Raw Interrupt Status Value Description 0 The number of available entries in the WFIFO is above the range specified by the WRFIFO field in the **EPIFIFOLVL** register. 1 The number of available entries in the WFIFO is within the trigger range specified by the WRFIFO field in the **EPIFIFOLVL** register. This bit is cleared when the level in the WFIFO is above the trigger point programmed by the WRFIFO field.
1	RDRIS	RO	0	Read Raw Interrupt Status Value Description 0 The number of valid entries in the NBRFIFO is below the trigger range specified by the RDFIFO field in the **EPIFIFOLVL** register. 1 The number of valid entries in the NBRFIFO is in the trigger range specified by the RDFIFO field in the **EPIFIFOLVL** register. This bit is cleared when the level in the NBRFIFO is below the trigger point programmed by the RDFIFO field.

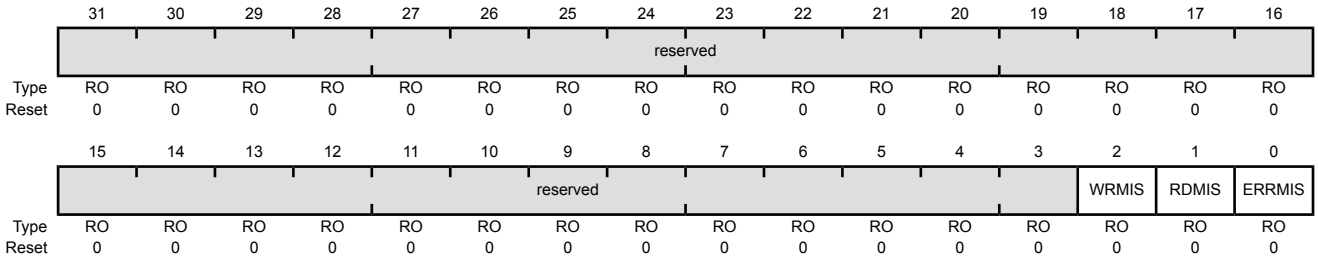
Bit/Field	Name	Type	Reset	Description
0	ERRRIS	RO	0	Error Raw Interrupt Status The error interrupt occurs in the following situations: ■ WFIFO Full. For a full WFIFO to generate an error interrupt, the WFERR bit in the **EPIFIFOLVL** register must be set. ■ Read Stalled. For a stalled read to generate an error interrupt, the RSERR bit in the **EPIFIFOLVL** register must be set. ■ Timeout. If the MAXWAIT field in the **EPIGPCFG** register is configured to a value other than 0, a timeout error occurs when iRDY or XFIFO not-ready signals hold a transaction for more than the count in the MAXWAIT field. Value Description 0 An error has not occurred. 1 A WFIFO Full, a Read Stalled, or a Timeout error has occurred. To determine which error occurred, read the status of the **EPI Error Interrupt Status and Clear (EPIEISC)** register. This bit is cleared by writing a 1 to the bit in the **EPIEISC** register that caused the interrupt.

Register 31: EPI Masked Interrupt Status (EPIMIS), offset 0x218

This register is the masked interrupt status register. On read, it gives the current state of each interrupt source (read, write, and error) after being masked via the **EPIIM** register. A write has no effect.

The values returned are the ANDing of the **EPIIM** and **EPIRIS** registers. If a bit is set in this register, the interrupt is sent to the interrupt controller.

EPI Masked Interrupt Status (EPIMIS)
Base 0x400D.0000
Offset 0x218
Type RO, reset 0x0000.0000



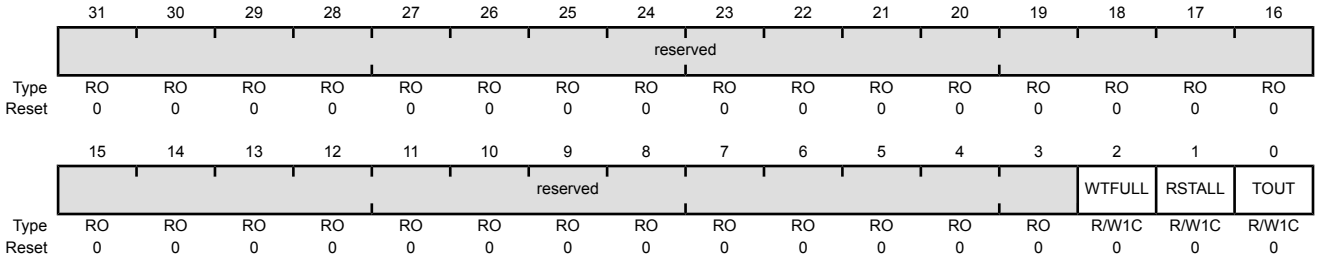
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	WRMIS	RO	0	Write Masked Interrupt Status Value Description 0 The number of available entries in the WFIFO is above the range specified by the trigger level or the interrupt is masked. 1 The number of available entries in the WFIFO is within the range specified by the trigger level (the `WRFIFO` field in the **EPIFIFOLVL** register) and the `WRIM` bit in the **EPIIM** register is set, triggering an interrupt to the interrupt controller.
1	RDMIS	RO	0	Read Masked Interrupt Status Value Description 0 The number of valid entries in the NBRFIFO is below the range specified by the trigger level or the interrupt is masked. 1 The number of valid entries in the NBRFIFO is within the range specified by the trigger level (the `RDFIFO` field in the **EPIFIFOLVL** register) and the `RDIM` bit in the **EPIIM** register is set, triggering an interrupt to the interrupt controller.
0	ERRMIS	RO	0	Error Masked Interrupt Status Value Description 0 An error has not occurred or the interrupt is masked. 1 A WFIFO Full, a Read Stalled, or a Timeout error has occurred and the `ERIM` bit in the **EPIIM** register is set, triggering an interrupt to the interrupt controller.

Register 32: EPI Error and Interrupt Status and Clear (EPIEISC), offset 0x21C

This register is used to clear a pending error interrupt. Clearing any defined bit in the **EPIEISC** has no effect; setting a bit clears the error source and the raw error returns to 0. When any of these bits are read as set it indicates that the ERRRIS bit in the **EPIRIS** register is set and an EPI controller error is sent to the interrupt controller if the ERIM bit in the **EPIIM** register is set. If any of bits [2:0] are written as 1, the register bit being written to, as well as the ERRIS bit in the **EPIRIS** register and the ERIM bit in the **EPIIM** register are cleared. Note that writing to this register and reading back immediately (pipelined by the processor) returns the old register contents. One cycle is needed between write and read.

EPI Error and Interrupt Status and Clear (EPIEISC)

Base 0x400D.0000
Offset 0x21C
Type R/W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	WTFULL	R/W1C	0	Write FIFO Full Error Value Description 0 The WFERR bit is not enabled or no writes are stalled. 1 The WFERR bit is enabled and a write is stalled due to the WFIFO being full. Writing a 1 to this bit clears it, as well as as the ERRRIS and ERIM bits.
1	RSTALL	R/W1C	0	Read Stalled Error Value Description 0 The RSERR bit is not enabled or no pending reads are stalled. 1 The RSERR bit is enabled and a pending read is stalled due to writes in the WFIFO. Writing a 1 to this bit clears it, as well as as the ERRRIS and ERIM bits.

Bit/Field	Name	Type	Reset	Description
0	TOUT	R/W1C	0	Timeout Error This bit is the timeout error source. The timeout error occurs when the iRDY or XFIFO not-ready signals hold a transaction for more than the count in the MAXWAIT field (when not 0). Value Description 0 No timeout error has occurred. 1 A timeout error has occurred. Writing a 1 to this bit clears it, as well as as the ERRRIS and ERIM bits.

10 General-Purpose Timers

Programmable timers can be used to count or time external events that drive the Timer input pins. The Stellaris® General-Purpose Timer Module (GPTM) contains four GPTM blocks. Each GPTM block provides two 16-bit timers/counters (referred to as Timer A and Timer B) that can be configured to operate independently as timers or event counters, or concatenated to operate as one 32-bit timer or one 32-bit Real-Time Clock (RTC). Timers can also be used to trigger µDMA transfers.

In addition, timers can be used to trigger analog-to-digital conversions (ADC). The ADC trigger signals from all of the general-purpose timers are ORed together before reaching the ADC module, so only one timer should be used to trigger ADC events.

The GPT Module is one timing resource available on the Stellaris microcontrollers. Other timer resources include the System Timer (SysTick) (see 108).

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

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

10.1 Block Diagram

In the block diagram, the specific Capture Compare PWM (CCP) pins available depend on the Stellaris device. See Table 10-1 on page 527 for the available CCP pins and their timer assignments.

Figure 10-1. GPTM Module Block Diagram

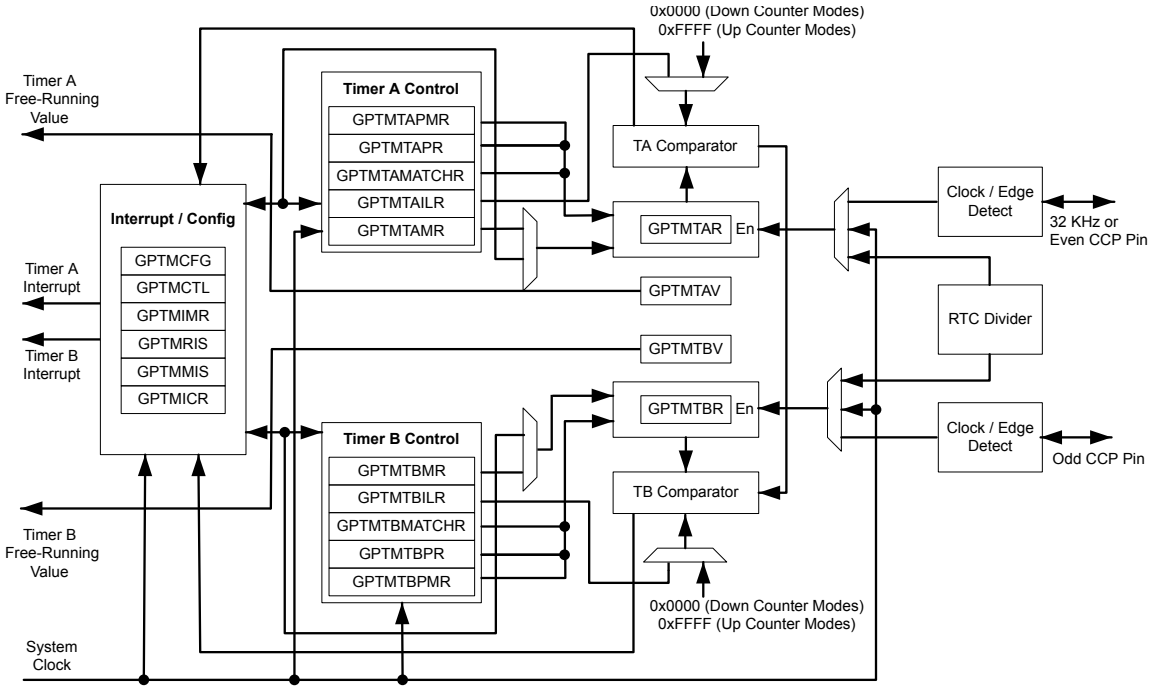


Table 10-1. Available CCP Pins

Timer	16-Bit Up/Down Counter	Even CCP Pin	Odd CCP Pin
Timer 0	TimerA	CCP0	-
	TimerB	-	CCP1
Timer 1	TimerA	CCP2	-
	TimerB	-	CCP3
Timer 2	TimerA	CCP4	-
	TimerB	-	CCP5
Timer 3	TimerA	CCP6	-
	TimerB	-	CCP7

10.2 Signal Description

The following table lists the external signals of the GP Timer module and describes the function of each. The GP Timer signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for these GP Timer signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 419) should be set to choose the GP Timer function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 437) to assign the GP Timer signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 395.

Table 10-2. General-Purpose Timers Signals (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
CCP0	13 22 23 39 42 55 66 72 91 97	PD3 (4) PC7 (4) PC6 (6) PJ2 (9) PF4 (1) PJ7 (10) PB0 (1) PB2 (5) PB5 (4) PD4 (1)	I/O	TTL	Capture/Compare/PWM 0.
CCP1	24 25 34 54 67 90 96 100	PC5 (1) PC4 (9) PA6 (2) PJ6 (10) PB1 (4) PB6 (1) PE3 (1) PD7 (3)	I/O	TTL	Capture/Compare/PWM 1.
CCP2	6 11 25 41 53 67 75 91 95 98	PE4 (6) PD1 (10) PC4 (5) PF5 (1) PJ5 (10) PB1 (1) PE1 (4) PB5 (6) PE2 (5) PD5 (1)	I/O	TTL	Capture/Compare/PWM 2.
CCP3	6 23 24 35 61 72 74 97	PE4 (1) PC6 (1) PC5 (5) PA7 (7) PF1 (10) PB2 (4) PE0 (3) PD4 (2)	I/O	TTL	Capture/Compare/PWM 3.
CCP4	22 25 35 52 95 98	PC7 (1) PC4 (6) PA7 (2) PJ4 (10) PE2 (1) PD5 (2)	I/O	TTL	Capture/Compare/PWM 4.
CCP5	5 12 25 36 90 91	PE5 (1) PD2 (4) PC4 (1) PG7 (8) PB6 (6) PB5 (2)	I/O	TTL	Capture/Compare/PWM 5.
CCP6	10 12 50 75 86 91	PD0 (6) PD2 (2) PJ3 (10) PE1 (5) PH0 (1) PB5 (3)	I/O	TTL	Capture/Compare/PWM 6.

Table 10-2. General-Purpose Timers Signals (100LQFP) *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
CCP7	11 13 85 90 96	PD1 (6) PD3 (2) PH1 (1) PB6 (2) PE3 (5)	I/O	TTL	Capture/Compare/PWM 7.

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

Table 10-3. General-Purpose Timers Signals (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
CCP0	H1 L2 M2 K6 K4 L12 E12 A11 B7 B5	PD3 (4) PC7 (4) PC6 (6) PJ2 (9) PF4 (1) PJ7 (10) PB0 (1) PB2 (5) PB5 (4) PD4 (1)	I/O	TTL	Capture/Compare/PWM 0.
CCP1	M1 L1 L6 L10 D12 A7 B4 A2	PC5 (1) PC4 (9) PA6 (2) PJ6 (10) PB1 (4) PB6 (1) PE3 (1) PD7 (3)	I/O	TTL	Capture/Compare/PWM 1.
CCP2	B2 G2 L1 K3 K12 D12 A12 B7 A4 C6	PE4 (6) PD1 (10) PC4 (5) PF5 (1) PJ5 (10) PB1 (1) PE1 (4) PB5 (6) PE2 (5) PD5 (1)	I/O	TTL	Capture/Compare/PWM 2.
CCP3	B2 M2 M1 M6 H12 A11 B11 B5	PE4 (1) PC6 (1) PC5 (5) PA7 (7) PF1 (10) PB2 (4) PE0 (3) PD4 (2)	I/O	TTL	Capture/Compare/PWM 3.
CCP4	L2 L1 M6 K11 A4 C6	PC7 (1) PC4 (6) PA7 (2) PJ4 (10) PE2 (1) PD5 (2)	I/O	TTL	Capture/Compare/PWM 4.

Table 10-3. General-Purpose Timers Signals (108BGA) *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
CCP5	B3 H2 L1 C10 A7 B7	PE5 (1) PD2 (4) PC4 (1) PG7 (8) PB6 (6) PB5 (2)	I/O	TTL	Capture/Compare/PWM 5.
CCP6	G1 H2 M10 A12 C9 B7	PD0 (6) PD2 (2) PJ3 (10) PE1 (5) PH0 (1) PB5 (3)	I/O	TTL	Capture/Compare/PWM 6.
CCP7	G2 H1 C8 A7 B4	PD1 (6) PD3 (2) PH1 (1) PB6 (2) PE3 (5)	I/O	TTL	Capture/Compare/PWM 7.

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

10.3 Functional Description

The main components of each GPTM block are two free-running up/down counters (referred to as Timer A and Timer B), two match registers, two prescaler match registers, two shadow registers, and two load/initialization registers and their associated control functions. The exact functionality of each GPTM is controlled by software and configured through the register interface. Timer A and Timer B can be used individually, in which case they have a 16-bit counting range. In addition, Timer A and Timer B can be concatenated to provide a 32-bit counting range. Note that the prescaler can only be used when the timers are used individually.

The available modes for each GPTM block are shown in Table 10-4 on page 530. Note that when counting down in one-shot or periodic modes, the prescaler acts as a true prescaler and contains the least-significant bits of the count. When counting up in one-shot or periodic modes, the prescaler acts as a timer extension and holds the most-significant bits of the count. In input edge count mode, the prescaler always acts as a timer extension, regardless of the count direction.

Table 10-4. General-Purpose Timer Capabilities

Mode	Timer Use	Count Direction	Counter Size	Prescaler Size[a]
One-shot	Individual	Up or Down	16-bit	8-bit
	Concatenated	Up or Down	32-bit	-
Periodic	Individual	Up or Down	16-bit	8-bit
	Concatenated	Up or Down	32-bit	-
RTC	Concatenated	Up	32-bit	-
Edge Count	Individual	Down	16-bit	8-bit
Edge Time	Individual	Down	16-bit	-
PWM	Individual	Down	16-bit	-

a. The prescaler is only available when the timers are used individually

Software configures the GPTM using the **GPTM Configuration (GPTMCFG)** register (see page 543), the **GPTM Timer A Mode (GPTMTAMR)** register (see page 544), and the **GPTM Timer B Mode**

(GPTMTBMR) register (see page 546). When in one of the concatentated modes, Timer A and Timer B can only operate in one mode. However, when configured in an individual mode, Timer A and Timer B can be independently configured in any combination of the individual modes.

10.3.1 GPTM Reset Conditions

After reset has been applied to the GPTM module, the module is in an inactive state, and all control registers are cleared and in their default states. Counters Timer A and Timer B are initialized to all 1s, along with their corresponding load registers: the **GPTM Timer A Interval Load (GPTMTAILR)** register (see page 561) and the **GPTM Timer B Interval Load (GPTMTBILR)** register (see page 562) and shadow registers: the **GPTM Timer A Value (GPTMTAV)** register (see page 571) and the **GPTM Timer B Value (GPTMTBV)** register (see page 572). The prescale counters are initialized to 0x00: the **GPTM Timer A Prescale (GPTMTAPR)** register (see page 565) and the **GPTM Timer B Prescale (GPTMTBPR)** register (see page 566).

10.3.2 Timer Modes

This section describes the operation of the various timer modes. When using Timer A and Timer B in concatenated mode, only the Timer A control and status bits must be used; there is no need to use Timer B control and status bits. The GPTM is placed into individual/split mode by writing a value of 0x4 to the **GPTM Configuration (GPTMCFG)** register (see page 543). In the following sections, the variable "n" is used in bit field and register names to imply either a Timer A function or a Timer B function. Throughout this section, the timeout event in down-count mode is 0x0 and in up-count mode is the value in the **GPTM Timer n Interval Load (GPTMTnILR)** and the optional **GPTM Timer n Prescale (GPTMTnPR)** registers.

10.3.2.1 One-Shot/Periodic Timer Mode

The selection of one-shot or periodic mode is determined by the value written to the TnMR field of the **GPTM Timer n Mode (GPTMTnMR)** register (see page 544). The timer is configured to count up or down using the TnCDIR bit in the **GPTMTnMR** register.

When software sets the TnEN bit in the **GPTM Control (GPTMCTL)** register (see page 548), the timer begins counting up from 0x0 or down from its preloaded value. Alternatively, if the TnWOT bit is set in the **GPTMTnMR** register, once the TnEN bit is set, the timer waits for a trigger to begin counting (see the section called “Wait-for-Trigger Mode” on page 532). Table 10-5 on page 531 shows the values that are loaded into the timer registers when the timer is enabled.

Table 10-5. Counter Values When the Timer is Enabled in Periodic or One-Shot Modes

Register	Count Down Mode	Count Up Mode
TnR	**GPTMTnILR**	0x0
TnV	**GPTMTnILR**	0x0

When the timer is counting down and it reaches the timeout event (0x0), the timer reloads its start value from the **GPTMTnILR** and the **GPTMTnPR** registers on the next cycle. When the timer is counting up and it reaches the timeout event (the value in the **GPTMTnILR** and the optional **GPTMTnPR** registers), the timer reloads with 0x0. 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, the timer starts counting again on the next cycle.

In periodic, snap-shot mode (TnMR field is 0x2 and the TnSNAPS bit is set in the **GPTMTnMR** register), the value of the timer at the time-out event is loaded into the **GPTMTnR** register. The free-running counter value is shown in the **GPTMTnV** register. In this manner, software can determine the time elapsed from the interrupt assertion to the ISR entry by examining the snapshot values

and the current value of the free-running timer. Snapshot mode is not available when the timer is configured in one-shot mode.

In addition to reloading the count value, the GPTM generates interrupts and triggers when it reaches the time-out event. The GPTM sets the TnTORIS bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register (see page 553), and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register (see page 559). If the time-out interrupt is enabled in the **GPTM Interrupt Mask (GPTMIMR)** register (see page 551), the GPTM also sets the TnTOMIS bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register (see page 556). By setting the TnMIE bit in the **GPTMTnMR** register, an interrupt condition can also be generated when the Timer value equals the value loaded into the **GPTM Timer n Match (GPTMTnMATCHR)** and **GPTM Timer n Prescale Match (GPTMTnPMR)** registers. This interrupt has the same status, masking, and clearing functions as the time-out interrupt, but uses the match interrupt bits instead (for example, the raw interrupt status is monitored via TnMRIS bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register). Note that the interrupt status bits are not updated by the hardware unless the TnMIE bit in the **GPTMTnMR** register is set, which is different than the behavior for the time-out interrupt. The ADC trigger is enabled by setting the TnOTE bit in **GPTMCTL**. The µDMA trigger is enabled by configuring and enabling the appropriate µDMA channel. See “Channel Configuration” on page 339.

If software updates the **GPTMTnILR** register while the counter is counting down, the counter loads the new value on the next clock cycle and continues counting from the new value. If software updates the **GPTMTnILR** register while the counter is counting up, the timeout event is changed on the next cycle to the new value. If software updates the **GPTM Timer n Value (GPTMTnV)** register while the counter is counting up or down, the counter loads the new value on the next clock cycle and continues counting from the new value..

If the TnSTALL bit in the **GPTMCTL** register is set, the timer freezes counting while the processor is halted by the debugger. The timer resumes counting when the processor resumes execution.

The following table shows a variety of configurations for a 16-bit free-running timer while using the prescaler. All values assume an 80-MHz clock with Tc=12.5 ns (clock period). The prescaler can only be used when a 16/32-bit timer is configured in 16-bit mode.

Table 10-6. 16-Bit Timer With Prescaler Configurations

Prescale (8-bit value)	# of Timer Clocks (Tc)[a]	Max Time	Units
00000000	1	0.8192	ms
00000001	2	1.6384	ms
00000010	3	2.4576	ms
----------	--	--	--
11111101	254	208.0768	ms
11111110	255	208.896	ms
11111111	256	209.7152	ms

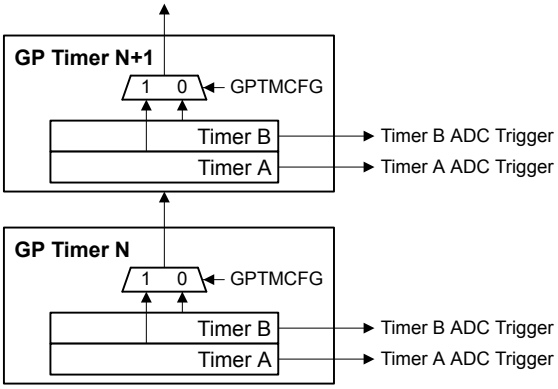
a. Tc is the clock period.

Wait-for-Trigger Mode

The Wait-for-Trigger mode allows daisy chaining of the timer modules such that once configured, a single timer can initiate mulitple timing events using the Timer triggers. Wait-for-Trigger mode is enabled by setting the TnWOT bit in the **GPTMTnMR** register. When the TnWOT bit is set, Timer N+1 does not begin counting until the timer in the previous position in the daisy chain (Timer N) reaches its time-out event. The daisy chain is configured such that GPTM1 always follows GPTM0, GPTM2 follows GPTM1, and so on. If Timer A is in 32-bit mode (controlled by the GPTMCFG bit in the **GPTMCFG** register), it triggers Timer A in the next module. If Timer A is in 16-bit mode, it triggers

Timer B in the same module, and Timer B triggers Timer A in the next module. Care must be taken that the TAWOT bit is never set in GPTM0. Figure 10-2 on page 533 shows how the GPTMCFG bit affects the daisy chain. This function is valid for both one-shot and periodic modes.

Figure 10-2. Timer Daisy Chain



10.3.2.2 Real-Time Clock Timer Mode

In Real-Time Clock (RTC) mode, the concatenated versions of the Timer A and Timer B registers are configured as an up-counter. When RTC mode is selected for the first time after reset, the counter is loaded with a value of 0x1. All subsequent load values must be written to the **GPTM Timer A Interval Load (GPTMTAILR)** register (see page 561). Table 10-7 on page 533 shows the values that are loaded into the timer registers when the timer is enabled.

Table 10-7. Counter Values When the Timer is Enabled in RTC Mode

Register	Count Down Mode	Count Up Mode
TnR	Not available	0x1
TnV	Not available	0x1

The input clock on a CCP input is required to be 32.768 KHz in RTC mode. The clock signal is then divided down to a 1-Hz rate and is passed along to the input of the counter.

When software writes the TAEN bit in the **GPTMCTL** register, the counter starts counting up from its preloaded value of 0x1. When the current count value matches the preloaded value in the **GPTMTAMATCHR** register, the GPTM asserts the RTCRIS bit in **GPTMRIS** and continues counting until either a hardware reset, or it is disabled by software (clearing the TAEN bit). When the timer value reaches the terminal count, the timer rolls over and continues counting up from 0x0. If the RTC interrupt is enabled in **GPTMIMR**, the GPTM also sets the RTCMIS bit in **GPTMMIS** and generates a controller interrupt. The status flags are cleared by writing the RTCCINT bit in **GPTMICR**.

In this mode, the **GPTMTnR** and **GPTMTnV** registers always have the same value.

In addition to generating interrupts, a µDMA trigger can be generated. The µDMA trigger is enabled by configuring and enabling the appropriate µDMA channel. See “Channel Configuration” on page 339.

If the TASTALL bit in the **GPTMCTL** register is set, the timer does not freeze when the processor is halted by the debugger if the RTCEN bit is set in **GPTMCTL**.

10.3.2.3 Input Edge-Count Mode

Note: For rising-edge detection, the input signal must be High for at least two system clock periods following the rising edge. Similarly, for falling-edge detection, the input signal must be Low for at least two system clock periods following the falling edge. Based on this criteria, the maximum input frequency for edge detection is 1/4 of the system frequency.

In Edge-Count mode, the timer is configured as a 24-bit down-counter including the optional prescaler with the upper count value stored in the **GPTM Timer n Prescale (GPTMTnPR)** register and the lower bits in the **GPTMTnR** register. In this mode, the timer is capable of capturing three types of events: rising edge, falling edge, or both. To place the timer in Edge-Count mode, the TnCMR bit of the **GPTMTnMR** register must be cleared. The type of edge that the timer counts is determined by the TnEVENT fields of the **GPTMCTL** register. During initialization, the **GPTMTnMATCHR** and **GPTMTnPMR** registers are configured so that the difference between the value in the **GPTMTnILR** and **GPTMTnPR** registers and the **GPTMTnMATCHR** and **GPTMTnPMR** registers equals the number of edge events that must be counted. Table 10-8 on page 534 shows the values that are loaded into the timer registers when the timer is enabled.

Table 10-8. Counter Values When the Timer is Enabled in Input Edge-Count Mode

Register	Count Down Mode	Count Up Mode
TnR	**GPTMTnILR**	Not available
TnV	**GPTMTnILR**	Not available

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** and **GPTMTnPMR**. When the counts match, the GPTM asserts the CnMRIS bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register, and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register. If the capture mode match interrupt is enabled in the **GPTM Interrupt Mask (GPTMIMR)** register, the GPTM also sets the CnMMIS bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register. In this mode, the **GPTMTnR** register holds the count of the input events while the **GPTMTnV** register holds the free-running timer value.

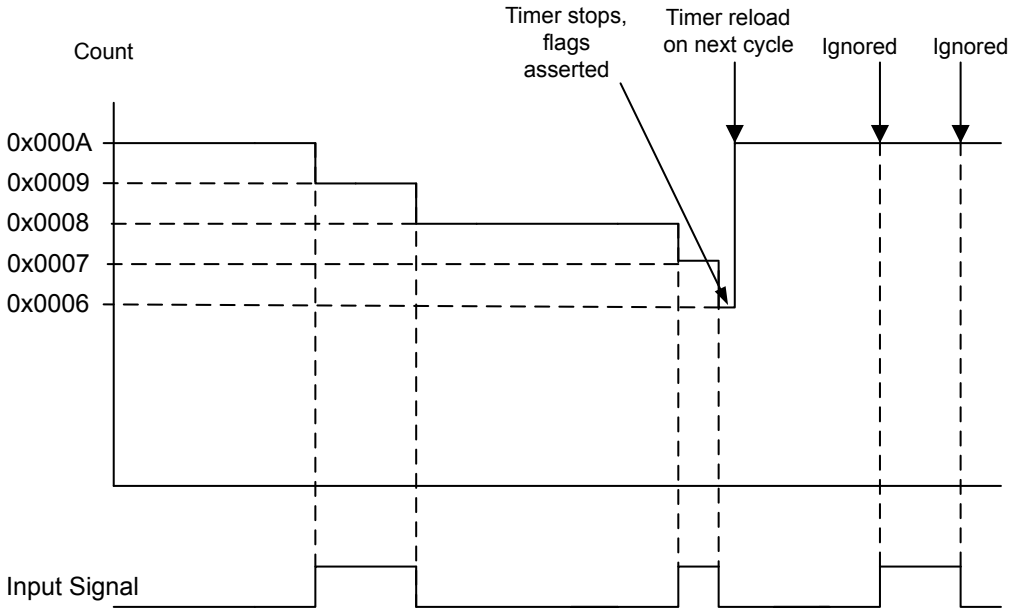
In addition to generating interrupts, an ADC and/or a µDMA trigger can be generated. The ADC trigger is enabled by setting the TnOTE bit in **GPTMCTL**.The µDMA trigger is enabled by configuring and enabling the appropriate µDMA channel. See “Channel Configuration” on page 339.

After the match value is reached, the counter is then reloaded using the value in **GPTMTnILR** and **GPTMTnPR** registers, and stopped because the GPTM automatically clears the TnEN bit in the **GPTMCTL** register. Once the event count has been reached, all further events are ignored until TnEN is re-enabled by software.

Figure 10-3 on page 535 shows how Input Edge-Count mode works. In this case, the timer start value is set to **GPTMTnILR** =0x000A and the match value is set to **GPTMTnMATCHR** =0x0006 so that four edge events are counted. The counter is configured to detect both edges of the input signal.

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

Figure 10-3. Input Edge-Count Mode Example



10.3.2.4 Input Edge-Time Mode

Note: For rising-edge detection, the input signal must be High for at least two system clock periods following the rising edge. Similarly, for falling edge detection, the input signal must be Low for at least two system clock periods following the falling edge. Based on this criteria, the maximum input frequency for edge detection is 1/4 of the system frequency.

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

In Edge-Time mode, the timer is configured as a 16-bit down-counter. In this mode, the timer is initialized to the value loaded in the **GPTMTnILR**register. The timer is capable of capturing three types of events: rising edge, falling edge, or both. The timer is placed into Edge-Time mode by setting the TnCMR bit in the **GPTMTnMR** register, and the type of event that the timer captures is determined by the TnEVENT fields of the **GPTMCTL** register. Table 10-9 on page 535 shows the values that are loaded into the timer registers when the timer is enabled.

Table 10-9. Counter Values When the Timer is Enabled in Input Event-Count Mode

Register	Count Down Mode	Count Up Mode
TnR	**GPTMTnILR**	Not available
TnV	**GPTMTnILR**	Not available

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 timer counter value is captured in the **GPTMTnR** register and is available to be read by the microcontroller. The GPTM then asserts the CnERIS bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register, and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register. If the capture mode event interrupt is enabled in the **GPTM Interrupt Mask (GPTMIMR)** register, the GPTM also sets the CnEMIS bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register. In this mode, the **GPTMTnR** register holds the time at which the selected input event occurred while the **GPTMTnV** register holds the free-running timer value. These registers can be read to determine the time that elapsed between the interrupt assertion and the entry into the ISR.

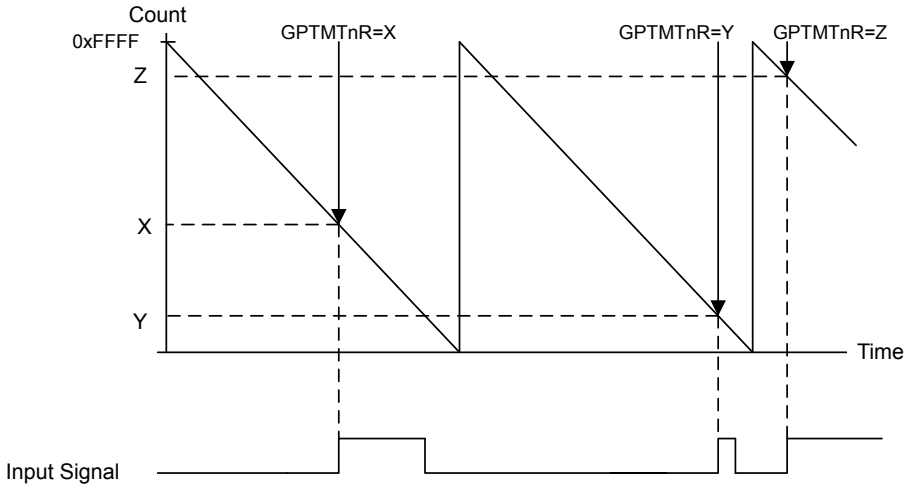
In addition to generating interrupts, an ADC and/or a µDMA trigger can be generated. The ADC trigger is enabled by setting the TnOTE bit in **GPTMCTL**.The µDMA trigger is enabled by configuring and enabling the appropriate µDMA channel. See “Channel Configuration” on page 339.

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 timeout value, it is reloaded with the value from the **GPTMTnILR** register.

Figure 10-4 on page 536 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 the **GPTMTnR** register).

Figure 10-4. 16-Bit Input Edge-Time Mode Example



10.3.2.5 PWM Mode

Note: The prescaler is not available in 16-Bit PWM mode.

The GPTM supports a simple PWM generation mode. In PWM mode, the timer is configured as a 16-bit down-counter with a start value (and thus period) defined by the **GPTMTnILR** register. In this mode, the PWM frequency and period are synchronous events and therefore guaranteed to be glitch free. PWM mode is enabled with the **GPTMTnMR** register by setting the TnAMS bit to 0x1, the TnCMR bit to 0x0, and the TnMR field to 0x1 or 0x2. Table 10-10 on page 536 shows the values that are loaded into the timer registers when the timer is enabled.

Table 10-10. Counter Values When the Timer is Enabled in PWM Mode

Register	Count Down Mode	Count Up Mode
GPTMTnR	**GPTMTnILR**	Not available
GPTMTnV	**GPTMTnILR**	Not available

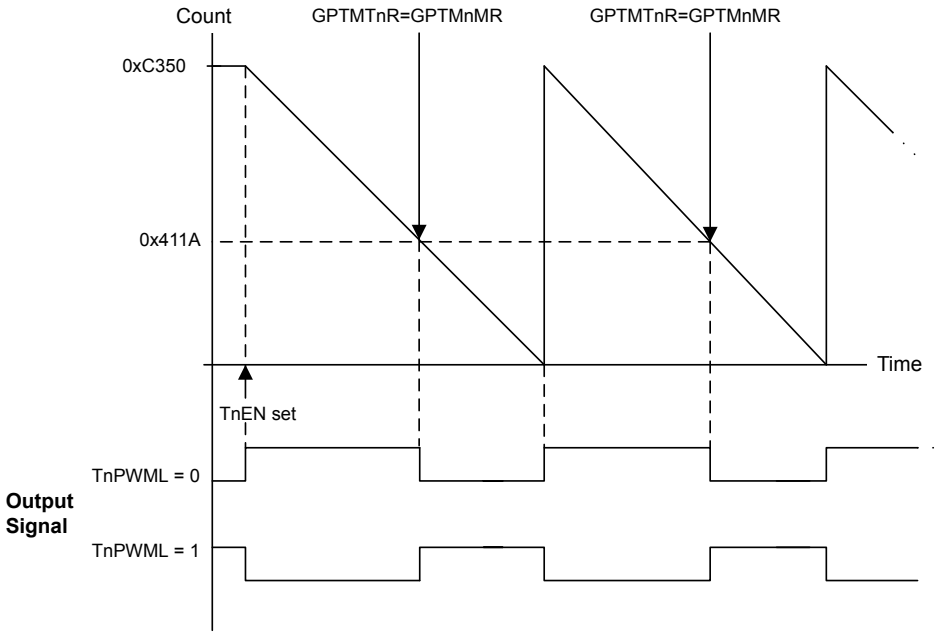
When software writes the TnEN bit in the **GPTMCTL** register, the counter begins counting down until it reaches the 0x0 state. On the next counter cycle in periodic mode, the counter reloads its start value from the **GPTMTnILR** register and continues counting until disabled by software clearing the TnEN bit in the **GPTMCTL** register. No interrupts or status bits are asserted in PWM mode.

In this mode, the **GPTMTnR** and **GPTMTnV** registers always have the same value.

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 **GPTMTnMATCHR** register. Software has the capability of inverting the output PWM signal by setting the TnPWML bit in the **GPTMCTL** register.

Figure 10-5 on page 537 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 **GPTMTnILR**=0xC350 and the match value is **GPTMTnMATCHR**=0x411A.

Figure 10-5. 16-Bit PWM Mode Example



10.3.3 DMA Operation

The timers each have a dedicated µDMA channel and can provide a request signal to the µDMA controller. The request is a burst type and occurs whenever a timer raw interrupt condition occurs. The arbitration size of the µDMA transfer should be set to the amount of data that should be transferred whenever a timer event occurs.

For example, to transfer 256 items, 8 items at a time every 10 ms, configure a timer to generate a periodic timeout at 10 ms. Configure the µDMA transfer for a total of 256 items, with a burst size of 8 items. Each time the timer times out, the µDMA controller transfers 8 items, until all 256 items have been transferred.

No other special steps are needed to enable Timers for µDMA operation. Refer to “Micro Direct Memory Access (µDMA)” on page 334 for more details about programming the µDMA controller.

10.3.4 Accessing Concatenated Register Values

The GPTM is placed into concatenated mode by writing a 0x0 or a 0x1 to the GPTMCFG bit field in the **GPTM Configuration (GPTMCFG)** register. In both configurations, certain registers are concatenated to form pseudo 32-bit registers. These registers include:

- **GPTM Timer A Interval Load (GPTMTAILR)** register [15:0], see page 561
- **GPTM Timer B Interval Load (GPTMTBILR)** register [15:0], see page 562
- **GPTM Timer A (GPTMTAR)** register [15:0], see page 569
- **GPTM Timer B (GPTMTBR)** register [15:0], see page 570
- **GPTM Timer A Value (GPTMTAV)** register [15:0], see page 571
- **GPTM Timer B Value (GPTMTBV)** register [15:0], see page 572
- **GPTM Timer A Match (GPTMTAMATCHR)** register [15:0], see page 563
- **GPTM Timer B Match (GPTMTBMATCHR)** register [15:0], see page 564

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

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

Likewise, a 32-bit read access to **GPTMTAR** returns the value:

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

A 32-bit read access to **GPTMTAV** returns the value:

```
GPTMTBV[15:0]:GPTMTAV[15:0]
```

10.4 Initialization and Configuration

To use a GPTM, the appropriate TIMERn bit must be set in the **RCGC1** register (see page 263). If using any CCP pins, the clock to the appropriate GPIO module must be enabled via the **RCGC1** register (see page 263). To find out which GPIO port to enable, refer to Table 22-4 on page 1144. Configure the PMCn fields in the **GPIOPCTL** register to assign the CCP signals to the appropriate pins (see page 437 and Table 22-5 on page 1151).

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

10.4.1 One-Shot/Periodic Timer Mode

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

1. Ensure the timer is disabled (the TnEN bit in the **GPTMCTL** register is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x0000.0000.

3. Configure the TnMR field in the **GPTM Timer n Mode Register (GPTMTnMR)**:

   a. Write a value of 0x1 for One-Shot mode.

   b. Write a value of 0x2 for Periodic mode.

4. Optionally configure the TnSNAPS, TnWOT, TnMTE, and TnCDIR bits in the **GPTMTnMR** register to select whether to capture the value of the free-running timer at time-out, use an external trigger to start counting, configure an additional trigger or interrupt, and count up or down.

5. Load the start value into the **GPTM Timer n Interval Load Register (GPTMTnILR)**.

6. If interrupts are required, set the appropriate bits in the **GPTM Interrupt Mask Register (GPTMIMR)**.

7. Set the TnEN bit in the **GPTMCTL** register to enable the timer and start counting.

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

If the TnMIE bit in the **GPTMTnMR** register is set, the RTCRIS bit in the **GPTMRIS** register is set, and the timer continues counting. In One-Shot mode, the timer stops counting after the time-out event. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode reloads the timer and continues counting after the time-out event.

10.4.2 Real-Time Clock (RTC) Mode

To use the RTC mode, the timer must have a 32.768-KHz input signal on an even CCP input. To enable the RTC feature, follow these steps:

1. Ensure the timer is disabled (the TAEN bit is cleared) before making any changes.

2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x0000.0001.

3. Write the match value to the **GPTM Timer n Match Register (GPTMTnMATCHR)**.

4. Set/clear the RTCEN bit in the **GPTM Control Register (GPTMCTL)** as needed.

5. If interrupts are required, set the RTCIM bit in the **GPTM Interrupt Mask Register (GPTMIMR)**.

6. Set the TAEN bit in the **GPTMCTL** register to enable the timer and start counting.

When the timer count equals the value in the **GPTMTnMATCHR** register, the GPTM asserts the RTCRIS bit in the **GPTMRIS** register and continues counting until Timer A is disabled or a hardware reset. The interrupt is cleared by writing the RTCCINT bit in the **GPTMICR** register.

10.4.3 Input Edge-Count Mode

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

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.

2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x0000.0004.

3. In the **GPTM Timer Mode (GPTMTnMR)** register, write the TnCMR field to 0x0 and the TnMR field to 0x3.
4. Configure the type of event(s) that the timer captures by writing the TnEVENT field of the **GPTM Control (GPTMCTL)** register.
5. If a prescaler is to be used, write the prescale value to the **GPTM Timer n Prescale Register (GPTMTnPR)**.
6. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** register.
7. Load the event count into the **GPTM Timer n Match (GPTMTnMATCHR)** register.
8. If interrupts are required, set the CnMIM bit in the **GPTM Interrupt Mask (GPTMIMR)** register.
9. Set the TnEN bit in the **GPTMCTL** register to enable the timer and begin waiting for edge events.
10. Poll the CnMRIS bit in the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the CnMCINT bit of the **GPTM Interrupt Clear (GPTMICR)** register.

When counting down in Input Edge-Count Mode, the timer stops after the programmed number of edge events has been detected. To re-enable the timer, ensure that the TnEN bit is cleared and repeat #4 on page 540 through #9 on page 540.

10.4.4 Input Edge Timing Mode

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

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x0000.0004.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, write the TnCMR field to 0x1 and the TnMR field to 0x3.
4. Configure the type of event that the timer captures by writing the TnEVENT field of the **GPTM Control (GPTMCTL)** register.
5. Load the timer start value into the **GPTM Timer n 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 Timer n (GPTMTnR)** register.

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

10.4.5 PWM Mode

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

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x0000.0004.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, set the TnAMS bit to 0x1, the TnCMR bit to 0x0, and the TnMR field to 0x2.
4. Configure the output state of the PWM signal (whether or not it is inverted) in the TnPWML field of the **GPTM Control (GPTMCTL)** register.
5. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** register.
6. Load the **GPTM Timer n Match (GPTMTnMATCHR)** register with the match value.
7. Set the TnEN bit in the **GPTM Control (GPTMCTL)** register to enable the timer and begin generation of the output PWM signal.

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

10.5 Register Map

Table 10-11 on page 541 lists the GPTM registers. The offset listed is a hexadecimal increment to the register's address, relative to that timer's base address:

- Timer 0: 0x4003.0000
- Timer 1: 0x4003.1000
- Timer 2: 0x4003.2000
- Timer 3: 0x4003.3000

Note that the GP Timer module clock must be enabled before the registers can be programmed (see page 263). There must be a delay of 3 system clocks after the Timer module clock is enabled before any Timer module registers are accessed.

Table 10-11. Timers Register Map

Offset	Name	Type	Reset	Description	See page
0x000	GPTMCFG	R/W	0x0000.0000	GPTM Configuration	543
0x004	GPTMTAMR	R/W	0x0000.0000	GPTM Timer A Mode	544
0x008	GPTMTBMR	R/W	0x0000.0000	GPTM Timer B Mode	546
0x00C	GPTMCTL	R/W	0x0000.0000	GPTM Control	548
0x018	GPTMIMR	R/W	0x0000.0000	GPTM Interrupt Mask	551
0x01C	GPTMRIS	RO	0x0000.0000	GPTM Raw Interrupt Status	553
0x020	GPTMMIS	RO	0x0000.0000	GPTM Masked Interrupt Status	556
0x024	GPTMICR	W1C	0x0000.0000	GPTM Interrupt Clear	559

Table 10-11. Timers Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x028	GPTMTAILR	R/W	0xFFFF.FFFF	GPTM Timer A Interval Load	561
0x02C	GPTMTBILR	R/W	0x0000.FFFF	GPTM Timer B Interval Load	562
0x030	GPTMTAMATCHR	R/W	0xFFFF.FFFF	GPTM Timer A Match	563
0x034	GPTMTBMATCHR	R/W	0x0000.FFFF	GPTM Timer B Match	564
0x038	GPTMTAPR	R/W	0x0000.0000	GPTM Timer A Prescale	565
0x03C	GPTMTBPR	R/W	0x0000.0000	GPTM Timer B Prescale	566
0x040	GPTMTAPMR	R/W	0x0000.0000	GPTM TimerA Prescale Match	567
0x044	GPTMTBPMR	R/W	0x0000.0000	GPTM TimerB Prescale Match	568
0x048	GPTMTAR	RO	0xFFFF.FFFF	GPTM Timer A	569
0x04C	GPTMTBR	RO	0x0000.FFFF	GPTM Timer B	570
0x050	GPTMTAV	RW	0xFFFF.FFFF	GPTM Timer A Value	571
0x054	GPTMTBV	RW	0x0000.FFFF	GPTM Timer B Value	572

10.6 Register Descriptions

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

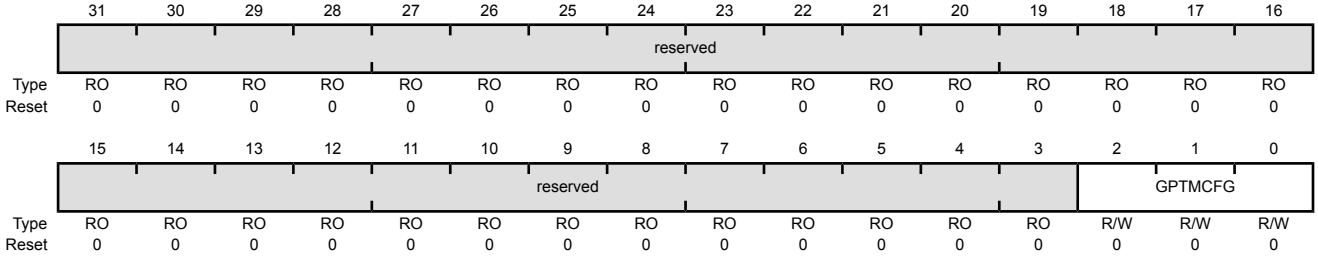
Register 1: GPTM Configuration (GPTMCFG), offset 0x000

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

Important: Bits in this register should only be changed when the TAEN and TBEN bits in the **GPTMCTL** register are cleared.

GPTM Configuration (GPTMCFG)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x000
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	GPTMCFG	R/W	0x0	GPTM Configuration The GPTMCFG values are defined as follows: Value Description 0x0 32-bit timer configuration. 0x1 32-bit real-time clock (RTC) counter configuration. 0x2-0x3 Reserved 0x4 16-bit timer configuration. The function is controlled by bits 1:0 of **GPTMTAMR** and **GPTMTBMR**. 0x5-0x7 Reserved

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

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

This register controls the modes for Timer A when it is used individually. When Timer A and Timer B are concatenated, this register controls the modes for both Timer A and Timer B, and the contents of **GPTMTBMR** are ignored.

Important: Bits in this register should only be changed when the TAEN bit in the **GPTMCTL** register is cleared.

GPTM Timer A Mode (GPTMTAMR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x004
Type R/W, reset 0x0000.0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved								TASNAPS	TAWOT	TAMIE	TACDIR	TAAMS	TACMR	TAMR	
RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TASNAPS	R/W	0	GPTM Timer A Snap-Shot Mode Value Description 0 Snap-shot mode is disabled. 1 If Timer A is configured in the periodic mode, the actual free-running value of Timer A is loaded at the time-out event into the **GPTM Timer A (GPTMTAR)** register. If the timer prescaler is used, the prescaler snapshot is loaded into the **GPTM Timer A (GPTMTAPR).**
6	TAWOT	R/W	0	GPTM Timer A Wait-on-Trigger Value Description 0 Timer A begins counting as soon as it is enabled. 1 If Timer A is enabled (TAEN is set in the **GPTMCTL** register), Timer A does not begin counting until it receives a trigger from the timer in the previous position in the daisy chain, see Figure 10-2 on page 533. This function is valid for both one-shot and periodic modes. This bit must be clear for GP Timer Module 0, Timer A.

Bit/Field	Name	Type	Reset	Description
5	TAMIE	R/W	0	GPTM Timer A Match Interrupt Enable Value Description 0 The match interrupt is disabled. 1 An interrupt is generated when the match value in the **GPTMTAMATCHR** register is reached in the one-shot and periodic modes.
4	TACDIR	R/W	0	GPTM Timer A Count Direction Value Description 0 The timer counts down. 1 When in one-shot or periodic mode, the timer counts up. When counting up, the timer starts from a value of 0x0. When in PWM or RTC mode, the status of this bit is ignored. PWM mode always counts down and RTC mode always counts up.
3	TAAMS	R/W	0	GPTM Timer A Alternate Mode Select The TAAMS values are defined as follows: Value Description 0 Capture mode is enabled. 1 PWM mode is enabled. **Note:** To enable PWM mode, you must also clear the TACMR bit and configure the TAMR field to 0x1 or 0x2.
2	TACMR	R/W	0	GPTM Timer A Capture Mode The TACMR values are defined as follows: Value Description 0 Edge-Count mode 1 Edge-Time mode
1:0	TAMR	R/W	0x0	GPTM Timer A Mode The TAMR values are defined as follows: Value Description 0x0 Reserved 0x1 One-Shot Timer mode 0x2 Periodic Timer mode 0x3 Capture mode The Timer mode is based on the timer configuration defined by bits 2:0 in the **GPTMCFG** register.

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

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

This register controls the modes for Timer B when it is used individually. When Timer A and Timer B are concatenated, this register is ignored and **GPTMTBMR** controls the modes for both Timer A and Timer B.

Important: Bits in this register should only be changed when the TBEN bit in the **GPTMCTL** register is cleared.

GPTM Timer B Mode (GPTMTBMR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x008
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								TBSNAPS	TBWOT	TBMIE	TBCDIR	TBAMS	TBCMR	TBMR	
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TBSNAPS	R/W	0	GPTM Timer B Snap-Shot Mode Value Description 0 Snap-shot mode is disabled. 1 If Timer B is configured in the periodic mode, the actual free-running value of Timer B is loaded at the time-out event into the **GPTM Timer B (GPTMTBR)** register. If the timer prescaler is used, the prescaler snapshot is loaded into the **GPTM Timer B (GPTMTBPR).**
6	TBWOT	R/W	0	GPTM Timer B Wait-on-Trigger Value Description 0 Timer B begins counting as soon as it is enabled. 1 If Timer B is enabled (TBEN is set in the **GPTMCTL** register), Timer B does not begin counting until it receives an it receives a trigger from the timer in the previous position in the daisy chain, see Figure 10-2 on page 533. This function is valid for both one-shot and periodic modes.

Bit/Field	Name	Type	Reset	Description
5	TBMIE	R/W	0	GPTM Timer B Match Interrupt Enable Value Description 0 The match interrupt is disabled. 1 An interrupt is generated when the match value in the **GPTMTBMATCHR** register is reached in the one-shot and periodic modes.
4	TBCDIR	R/W	0	GPTM Timer B Count Direction Value Description 0 The timer counts down. 1 When in one-shot or periodic mode, the timer counts up. When counting up, the timer starts from a value of 0x0. When in PWM or RTC mode, the status of this bit is ignored. PWM mode always counts down and RTC mode always counts up.
3	TBAMS	R/W	0	GPTM Timer B Alternate Mode Select The TBAMS values are defined as follows: Value Description 0 Capture mode is enabled. 1 PWM mode is enabled. **Note:** To enable PWM mode, you must also clear the TBCMR bit and configure the TBMR field to 0x1 or 0x2.
2	TBCMR	R/W	0	GPTM Timer B Capture Mode The TBCMR values are defined as follows: Value Description 0 Edge-Count mode 1 Edge-Time mode
1:0	TBMR	R/W	0x0	GPTM Timer B Mode The TBMR values are defined as follows: Value Description 0x0 Reserved 0x1 One-Shot Timer mode 0x2 Periodic Timer mode 0x3 Capture mode The timer mode is based on the timer configuration defined by bits 2:0 in the **GPTMCFG** register.

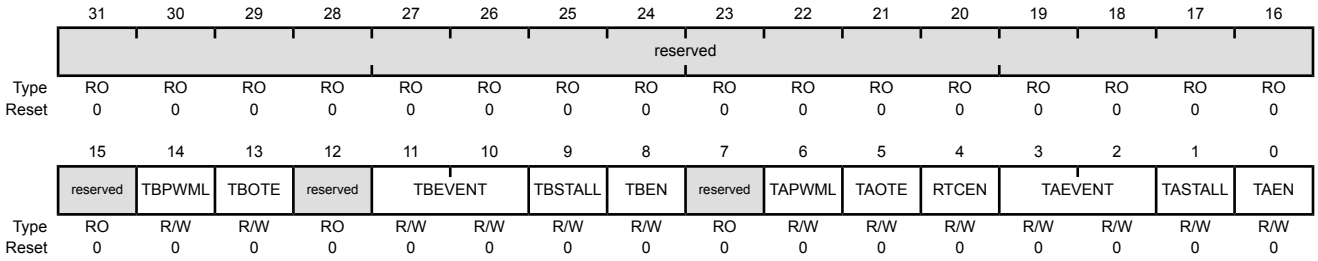
Register 4: GPTM Control (GPTMCTL), offset 0x00C

This register is used alongside the **GPTMCFG** and **GMTMTnMR** registers to fine-tune the timer configuration, and to enable other features such as timer stall and the output trigger. The output trigger can be used to initiate transfers on the ADC module.

Important: Bits in this register should only be changed when the TnEN bit for the respective timer is cleared.

GPTM Control (GPTMCTL)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x00C
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:15	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	TBPWML	R/W	0	GPTM Timer B PWM Output Level The TBPWML values are defined as follows: Value Description 0 Output is unaffected. 1 Output is inverted.
13	TBOTE	R/W	0	GPTM Timer B Output Trigger Enable The TBOTE values are defined as follows: Value Description 0 The output Timer B ADC trigger is disabled. 1 The output Timer B ADC trigger is enabled. In addition, the ADC must be enabled and the timer selected as a trigger source with the EMn bit in the **ADCEMUX** register (see page 631).
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
11:10	TBEVENT	R/W	0x0	GPTM Timer B Event Mode The TBEVENT values are defined as follows: Value Description 0x0 Positive edge 0x1 Negative edge 0x2 Reserved 0x3 Both edges
9	TBSTALL	R/W	0	GPTM Timer B Stall Enable The TBSTALL values are defined as follows: Value Description 0 Timer B continues counting while the processor is halted by the debugger. 1 Timer B freezes counting while the processor is halted by the debugger. If the processor is executing normally, the TBSTALL bit is ignored.
8	TBEN	R/W	0	GPTM Timer B Enable The TBEN values are defined as follows: Value Description 0 Timer B is disabled. 1 Timer B is enabled and begins counting or the capture logic is enabled based on the **GPTMCFG** register.
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	TAPWML	R/W	0	GPTM Timer A PWM Output Level The TAPWML values are defined as follows: Value Description 0 Output is unaffected. 1 Output is inverted.
5	TAOTE	R/W	0	GPTM Timer A Output Trigger Enable The TAOTE values are defined as follows: Value Description 0 The output Timer A ADC trigger is disabled. 1 The output Timer A ADC trigger is enabled. In addition, the ADC must be enabled and the timer selected as a trigger source with the EMn bit in the **ADCEMUX** register (see page 631).

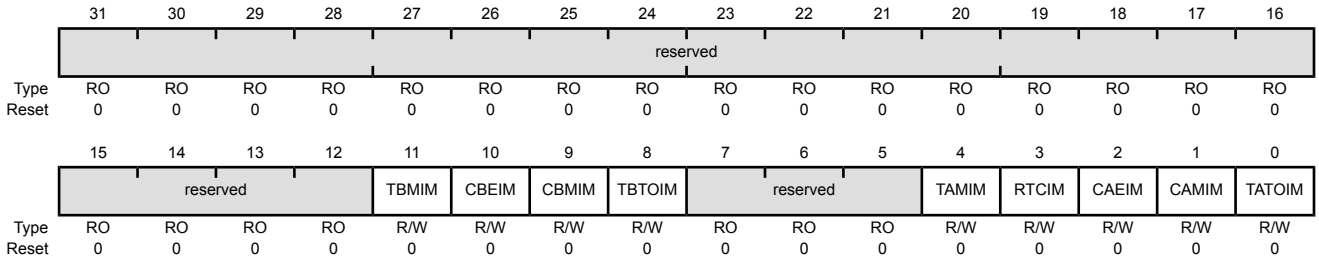
Bit/Field	Name	Type	Reset	Description
4	RTCEN	R/W	0	GPTM RTC Stall Enable The RTCEN values are defined as follows: Value Description 0 RTC counting freezes while the processor is halted by the debugger. 1 RTC counting continues while the processor is halted by the debugger. If the RTCEN bit is set, it prevents the timer from stalling in all operating modes, even if TnSTALL is set.
3:2	TAEVENT	R/W	0x0	GPTM Timer A Event Mode The TAEVENT values are defined as follows: Value Description 0x0 Positive edge 0x1 Negative edge 0x2 Reserved 0x3 Both edges
1	TASTALL	R/W	0	GPTM Timer A Stall Enable The TASTALL values are defined as follows: Value Description 0 Timer A continues counting while the processor is halted by the debugger. 1 Timer A freezes counting while the processor is halted by the debugger. If the processor is executing normally, the TASTALL bit is ignored.
0	TAEN	R/W	0	GPTM Timer A Enable The TAEN values are defined as follows: Value Description 0 Timer A is disabled. 1 Timer A is enabled and begins counting or the capture logic is enabled based on the **GPTMCFG** register.

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

This register allows software to enable/disable GPTM controller-level interrupts. Setting a bit enables the corresponding interrupt, while clearing a bit disables it.

GPTM Interrupt Mask (GPTMIMR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x018
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	TBMIM	R/W	0	GPTM Timer B Match Interrupt Mask The TBMIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
10	CBEIM	R/W	0	GPTM Timer B Capture Mode Event Interrupt Mask The CBEIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
9	CBMIM	R/W	0	GPTM Timer B Capture Mode Match Interrupt Mask The CBMIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.

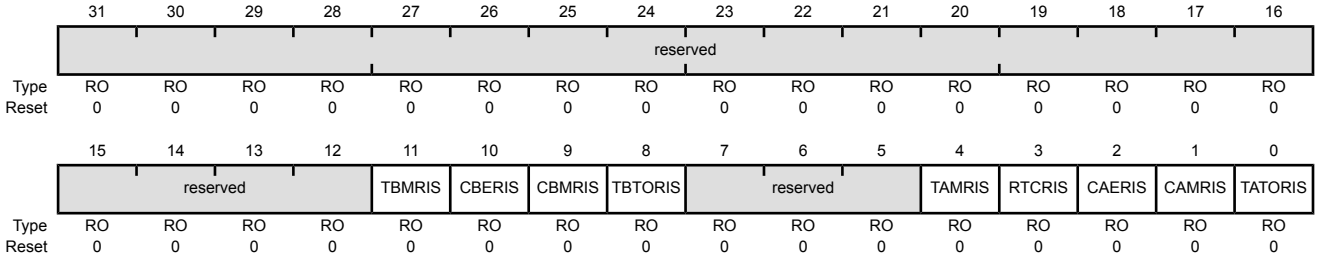
Bit/Field	Name	Type	Reset	Description
8	TBTOIM	R/W	0	GPTM Timer B Time-Out Interrupt Mask The TBTOIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
7:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	TAMIM	R/W	0	GPTM Timer A Match Interrupt Mask The TAMIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
3	RTCIM	R/W	0	GPTM RTC Interrupt Mask The RTCIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
2	CAEIM	R/W	0	GPTM Timer A Capture Mode Event Interrupt Mask The CAEIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
1	CAMIM	R/W	0	GPTM Timer A Capture Mode Match Interrupt Mask The CAMIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
0	TATOIM	R/W	0	GPTM Timer A Time-Out Interrupt Mask The TATOIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.

Register 6: GPTM Raw Interrupt Status (GPTMRIS), offset 0x01C

This register shows the state of the GPTM's internal interrupt signal. These bits are set whether or not the interrupt is masked in the **GPTMIMR** register. Each bit can be cleared by writing a 1 to its corresponding bit in **GPTMICR**.

GPTM Raw Interrupt Status (GPTMRIS)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x01C
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	TBMRIS	RO	0	GPTM Timer B Match Raw Interrupt Value Description 1 The TBMIE bit is set in the **GPTMTBMR** register, and the match values in the **GPTMTBMATCHR** and (optionally) **GPTMTBPMR** registers have been reached when configured in one-shot or periodic mode. 0 The match value has not been reached. This bit is cleared by writing a 1 to the TBMCINT bit in the **GPTMICR** register.
10	CBERIS	RO	0	GPTM Timer B Capture Mode Event Raw Interrupt Value Description 1 A capture mode event has occurred for Timer B. This interrupt asserts when the subtimer is configured in Input Edge-Time mode. 0 The capture mode event for Timer B has not occurred. This bit is cleared by writing a 1 to the CBECINT bit in the **GPTMICR** register.

Bit/Field	Name	Type	Reset	Description
9	CBMRIS	RO	0	GPTM Timer B Capture Mode Match Raw Interrupt Value Description 1 The capture mode match has occurred for Timer B. This interrupt asserts when the values in the **GPTMTBR** and **GPTMTBPR** match the values in the **GPTMTBMATCHR** and **GPTMTBPMR** when configured in Input Edge-Time mode. 0 The capture mode match for Timer B has not occurred. This bit is cleared by writing a 1 to the CBMCINT bit in the **GPTMICR** register.
8	TBTORIS	RO	0	GPTM Timer B Time-Out Raw Interrupt Value Description 1 Timer B has timed out. This interrupt is asserted when a one-shot or periodic mode timer reaches it's count limit (0 or the value loaded into **GPTMTBILR**, depending on the count direction). 0 Timer B has not timed out. This bit is cleared by writing a 1 to the TBTOCINT bit in the **GPTMICR** register.
7:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	TAMRIS	RO	0	GPTM Timer A Match Raw Interrupt Value Description 1 The TAMIE bit is set in the **GPTMTAMR** register, and the match value in the **GPTMTAMATCHR** and (optionally) **GPTMTAPMR** registers have been reached when configured in one-shot or periodic mode. 0 The match value has not been reached. This bit is cleared by writing a 1 to the TAMCINT bit in the **GPTMICR** register.
3	RTCRIS	RO	0	GPTM RTC Raw Interrupt Value Description 1 The RTC event has occurred. 0 The RTC event has not occurred. This bit is cleared by writing a 1 to the RTCCINT bit in the **GPTMICR** register.

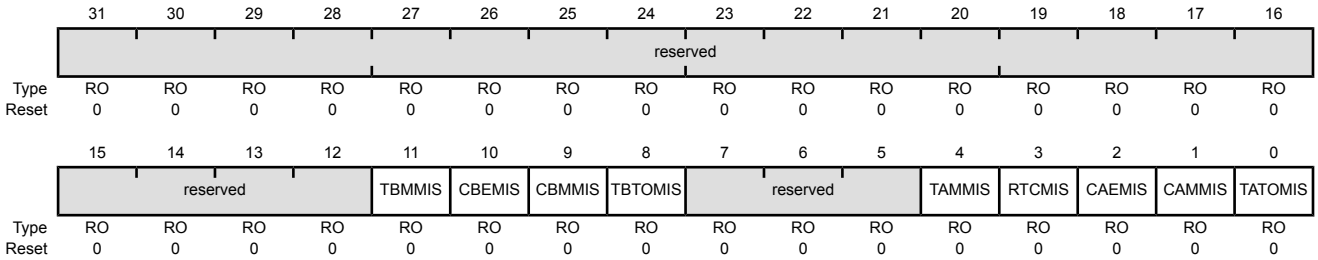
Bit/Field	Name	Type	Reset	Description
2	CAERIS	RO	0	GPTM Timer A Capture Mode Event Raw Interrupt Value Description 1 A capture mode event has occurred for Timer A. This interrupt asserts when the subtimer is configured in Input Edge-Time mode. 0 The capture mode event for Timer A has not occurred. This bit is cleared by writing a 1 to the CAECINT bit in the **GPTMICR** register.
1	CAMRIS	RO	0	GPTM Timer A Capture Mode Match Raw Interrupt Value Description 1 A capture mode match has occurred for Timer A. This interrupt asserts when the values in the **GPTMTAR** and **GPTMTAPR** match the values in the **GPTMTAMATCHR** and **GPTMTAPMR** when configured in Input Edge-Time mode. 0 The capture mode match for Timer A has not occurred. This bit is cleared by writing a 1 to the CAMCINT bit in the **GPTMICR** register.
0	TATORIS	RO	0	GPTM Timer A Time-Out Raw Interrupt Value Description 1 Timer A has timed out. This interrupt is asserted when a one-shot or periodic mode timer reaches it's count limit (0 or the value loaded into **GPTMTAILR**, depending on the count direction). 0 Timer A has not timed out. This bit is cleared by writing a 1 to the TATOCINT bit in the **GPTMICR** register.

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

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

GPTM Masked Interrupt Status (GPTMMIS)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x020
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	TBMMIS	RO	0	GPTM Timer B Match Masked Interrupt Value Description 1 An unmasked Timer B Mode Match interrupt has occurred. 0 A Timer B Mode Match interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the TBMCINT bit in the **GPTMICR** register.
10	CBEMIS	RO	0	GPTM Timer B Capture Mode Event Masked Interrupt Value Description 1 An unmasked Capture B event interrupt has occurred. 0 A Capture B event interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the CBECINT bit in the **GPTMICR** register.

Bit/Field	Name	Type	Reset	Description
9	CBMMIS	RO	0	GPTM Timer B Capture Mode Match Masked Interrupt Value Description 1 An unmasked Capture B Match interrupt has occurred. 0 A Capture B Mode Match interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the CBMCINT bit in the **GPTMICR** register.
8	TBTOMIS	RO	0	GPTM Timer B Time-Out Masked Interrupt Value Description 1 An unmasked Timer B Time-Out interrupt has occurred. 0 A Timer B Time-Out interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the TBTOCINT bit in the **GPTMICR** register.
7:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	TAMMIS	RO	0	GPTM Timer A Match Masked Interrupt Value Description 1 An unmasked Timer A Mode Match interrupt has occurred. 0 A Timer A Mode Match interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the TAMCINT bit in the **GPTMICR** register.
3	RTCMIS	RO	0	GPTM RTC Masked Interrupt Value Description 1 An unmasked RTC event interrupt has occurred. 0 An RTC event interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the RTCCINT bit in the **GPTMICR** register.
2	CAEMIS	RO	0	GPTM Timer A Capture Mode Event Masked Interrupt Value Description 1 An unmasked Capture A event interrupt has occurred. 0 A Capture A event interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the CAECINT bit in the **GPTMICR** register.

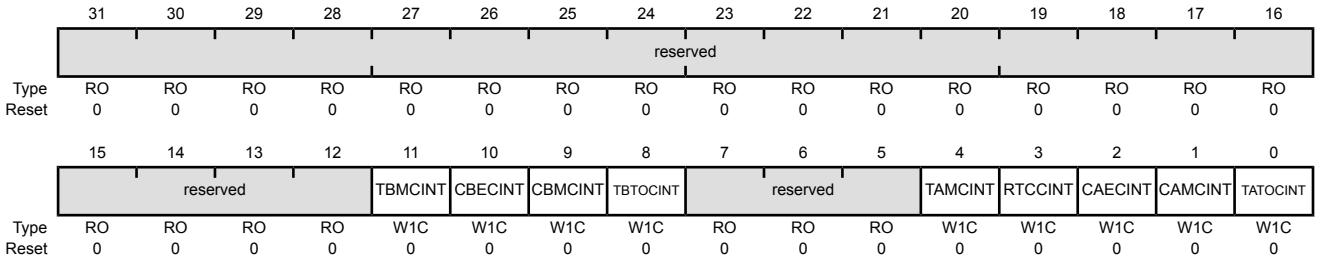
Bit/Field	Name	Type	Reset	Description
1	CAMMIS	RO	0	GPTM Timer A Capture Mode Match Masked Interrupt Value Description 1 An unmasked Capture A Match interrupt has occurred. 0 A Capture A Mode Match interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the CAMCINT bit in the **GPTMICR** register.
0	TATOMIS	RO	0	GPTM Timer A Time-Out Masked Interrupt Value Description 1 An unmasked Timer A Time-Out interrupt has occurred. 0 A Timer A Time-Out interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the TATOCINT bit in the **GPTMICR** register.

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

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

GPTM Interrupt Clear (GPTMICR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x024
Type W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	TBMCINT	W1C	0	GPTM Timer B Match Interrupt Clear Writing a 1 to this bit clears the TBMRIS bit in the **GPTMRIS** register and the TBMMIS bit in the **GPTMMIS** register.
10	CBECINT	W1C	0	GPTM Timer B Capture Mode Event Interrupt Clear Writing a 1 to this bit clears the CBERIS bit in the **GPTMRIS** register and the CBEMIS bit in the **GPTMMIS** register.
9	CBMCINT	W1C	0	GPTM Timer B Capture Mode Match Interrupt Clear Writing a 1 to this bit clears the CBMRIS bit in the **GPTMRIS** register and the CBMMIS bit in the **GPTMMIS** register.
8	TBTOCINT	W1C	0	GPTM Timer B Time-Out Interrupt Clear Writing a 1 to this bit clears the TBTORIS bit in the **GPTMRIS** register and the TBTOMIS bit in the **GPTMMIS** register.
7:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	TAMCINT	W1C	0	GPTM Timer A Match Interrupt Clear Writing a 1 to this bit clears the TAMRIS bit in the **GPTMRIS** register and the TAMMIS bit in the **GPTMMIS** register.
3	RTCCINT	W1C	0	GPTM RTC Interrupt Clear Writing a 1 to this bit clears the RTCRIS bit in the **GPTMRIS** register and the RTCMIS bit in the **GPTMMIS** register.
2	CAECINT	W1C	0	GPTM Timer A Capture Mode Event Interrupt Clear Writing a 1 to this bit clears the CAERIS bit in the **GPTMRIS** register and the CAEMIS bit in the **GPTMMIS** register.

Bit/Field	Name	Type	Reset	Description
1	CAMCINT	W1C	0	GPTM Timer A Capture Mode Match Interrupt Clear Writing a 1 to this bit clears the CAMRIS bit in the **GPTMRIS** register and the CAMMIS bit in the **GPTMMIS** register.
0	TATOCINT	W1C	0	GPTM Timer A Time-Out Raw Interrupt Writing a 1 to this bit clears the TATORIS bit in the **GPTMRIS** register and the TATOMIS bit in the **GPTMMIS** register.

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

When the timer is counting down, this register is used to load the starting count value into the timer. When the timer is counting up, this register sets the upper bound for the timeout event.

When a GPTM is configured to one of the 32-bit modes, **GPTMTAILR** appears as a 32-bit register (the upper 16-bits correspond to the contents of the **GPTM Timer B Interval Load (GPTMTBILR)** register). In a 16-bit mode, the upper 16 bits of this register read as 0s and have no effect on the state of **GPTMTBILR**.

GPTM Timer A Interval Load (GPTMTAILR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x028
Type R/W, reset 0xFFFF.FFFF

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

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

Bit/Field	Name	Type	Reset	Description
31:0	TAILR	R/W	0xFFFF.FFFF	GPTM Timer A Interval Load Register Writing this field loads the counter for Timer A. A read returns the current value of **GPTMTAILR**.

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

When the timer is counting down, this register is used to load the starting count value into the timer. When the timer is counting up, this register sets the upper bound for the timeout event.

When a GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the **GPTMTAILR** register. Reads from this register return the current value of Timer B and writes are ignored. In a 16-bit mode, bits 15:0 are used for the load value. Bits 31:16 are reserved in both cases.

GPTM Timer B Interval Load (GPTMTBILR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x02C
Type R/W, reset 0x0000.FFFF

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

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

Bit/Field	Name	Type	Reset	Description
31:0	TBILR	R/W	0x0000.FFFF	GPTM Timer B Interval Load Register Writing this field loads the counter for Timer B. A read returns the current value of **GPTMTBILR**. When a GPTM is in 32-bit mode, writes are ignored, and reads return the current value of **GPTMTBILR**.

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

This register is loaded with a match value. Interrupts can be generated when the timer value is equal to the value in this register in one-shot or periodic mode.

In Edge-Count mode, this register along with **GPTMTAILR**, determines how many edge events are counted. The total number of edge events counted is equal to the value in **GPTMTAILR** minus this value.

In PWM mode, this value along with **GPTMTAILR**, determines the duty cycle of the output PWM signal.

When a GPTM is configured to one of the 32-bit modes, **GPTMTAMATCHR** appears as a 32-bit register (the upper 16-bits correspond to the contents of the **GPTM Timer B Match (GPTMTBMATCHR)** register). In a 16-bit mode, the upper 16 bits of this register read as 0s and have no effect on the state of **GPTMTBMATCHR**.

GPTM Timer A Match (GPTMTAMATCHR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x030
Type R/W, reset 0xFFFF.FFFF

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

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

Bit/Field	Name	Type	Reset	Description
31:0	TAMR	R/W	0xFFFF.FFFF	GPTM Timer A Match Register This value is compared to the **GPTMTAR** register to determine match events.

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

This register is loaded with a match value. Interrupts can be generated when the timer value is equal to the value in this register in one-shot or periodic mode.

In Edge-Count mode, this register 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.

In PWM mode, this value along with **GPTMTBILR**, determines the duty cycle of the output PWM signal.

When a GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the **GPTMTAMATCHR** register. Reads from this register return the current match value of Timer B and writes are ignored. In a 16-bit mode, bits 15:0 are used for the match value. Bits 31:16 are reserved in both cases.

GPTM Timer B Match (GPTMTBMATCHR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x034
Type R/W, reset 0x0000.FFFF

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

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

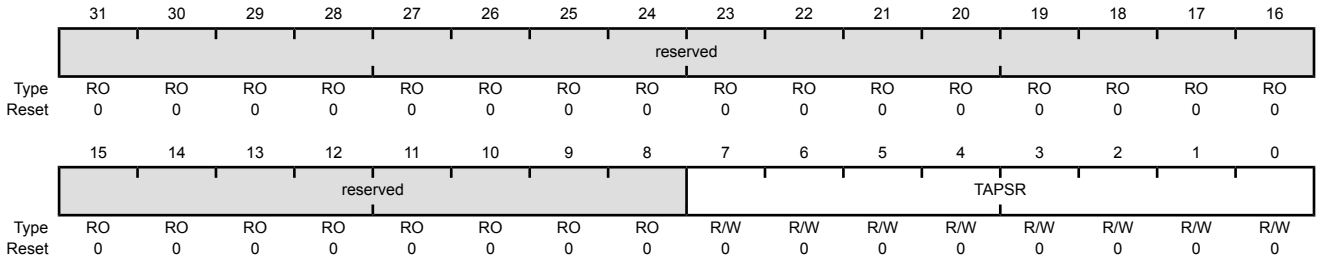
Bit/Field	Name	Type	Reset	Description
31:0	TBMR	R/W	0x0000.FFFF	GPTM Timer B Match Register This value is compared to the **GPTMTBR** register to determine match events.

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

This register allows software to extend the range of the 16-bit timers in periodic and one-shot modes. In Edge-Count mode, this register is the MSB of the 24-bit count value.

GPTM Timer A Prescale (GPTMTAPR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x038
Type R/W, reset 0x0000.0000



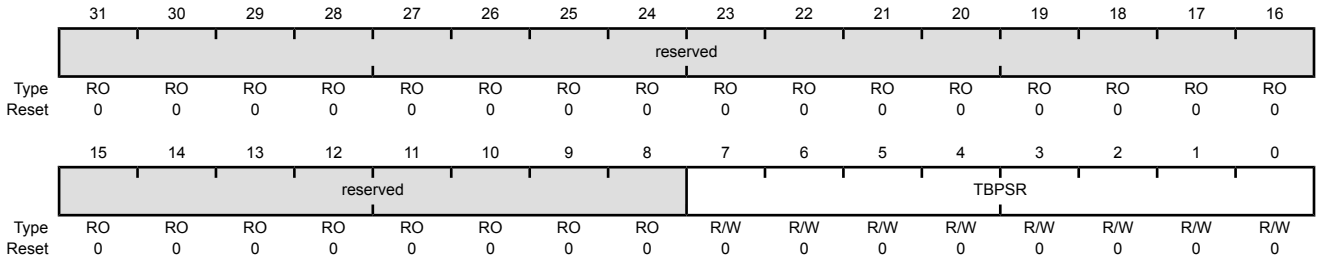
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TAPSR	R/W	0x00	GPTM Timer A Prescale The register loads this value on a write. A read returns the current value of the register. Refer to Table 10-6 on page 532 for more details and an example.

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

This register allows software to extend the range of the 16-bit timers in periodic and one-shot modes. In Edge-Count mode, this register is the MSB of the 24-bit count value.

GPTM Timer B Prescale (GPTMTBPR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x03C
Type R/W, reset 0x0000.0000



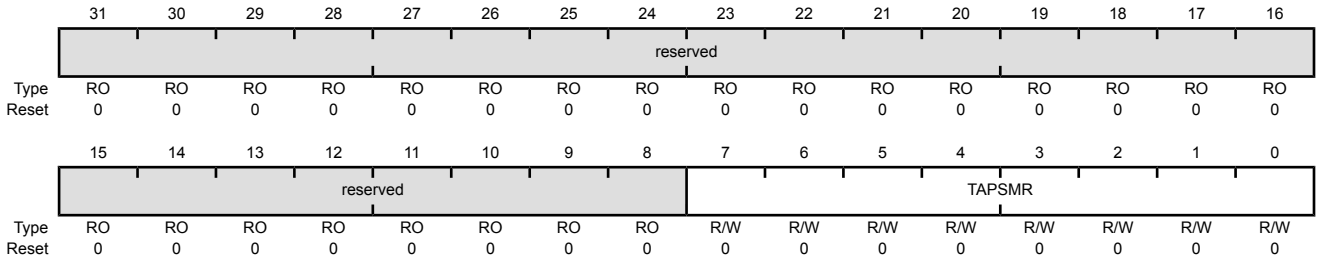
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TBPSR	R/W	0x00	GPTM Timer B Prescale The register loads this value on a write. A read returns the current value of this register. Refer to Table 10-6 on page 532 for more details and an example.

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

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

GPTM TimerA Prescale Match (GPTMTAPMR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x040
Type R/W, reset 0x0000.0000



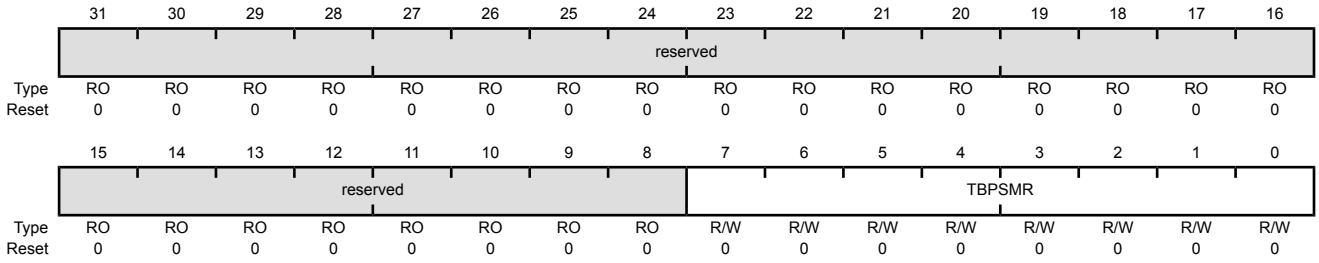
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TAPSMR	R/W	0x00	GPTM TimerA Prescale Match This value is used alongside **GPTMTAMATCHR** to detect timer match events while using a prescaler.

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

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

GPTM TimerB Prescale Match (GPTMTBPMR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x044
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TBPSMR	R/W	0x00	GPTM TimerB Prescale Match This value is used alongside **GPTMTBMATCHR** to detect timer match events while using a prescaler.

Register 17: GPTM Timer A (GPTMTAR), offset 0x048

This register shows the current value of the Timer A counter in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place. Also in Input Edge-Count mode, bits 23:16 contain the upper 8 bits of the count.

When a GPTM is configured to one of the 32-bit modes, **GPTMTAR** appears as a 32-bit register (the upper 16-bits correspond to the contents of the **GPTM Timer B (GPTMTBR)** register). In the16-bit Input Edge Count, Input Edge Time, and PWM modes, bits 15:0 contain the value of the counter and bits 23:16 contain the value of the prescaler, which is the upper 8 bits of the count. Bits 31:24 always read as 0. To read the value of the prescaler in 16-bit One-Shot and Periodic modes, read bits [23:16] in the **GPTMTAV** register.

GPTM Timer A (GPTMTAR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x048
Type RO, reset 0xFFFF.FFFF

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

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

Bit/Field	Name	Type	Reset	Description
31:0	TAR	RO	0xFFFF.FFFF	GPTM Timer A Register A read returns the current value of the **GPTM Timer A Count Register**, in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place.

Register 18: GPTM Timer B (GPTMTBR), offset 0x04C

This register shows the current value of the Timer B counter in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place. Also in Input Edge-Count mode, bits 23:16 contain the upper 8 bits of the count.

When a GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the **GPTMTAR** register. Reads from this register return the current value of Timer B. In a 16-bit mode, bits 15:0 contain the value of the counter and bits 23:16 contain the value of the prescaler in Input Edge Count, Input Edge Time, and PWM modes, which is the upper 8 bits of the count. Bits 31:24 always read as 0. To read the value of the prescaler in 16-bit One-Shot and Periodic modes, read bits [23:16] in the **GPTMTBV** register.

GPTM Timer B (GPTMTBR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x04C
Type RO, reset 0x0000.FFFF

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

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

Bit/Field	Name	Type	Reset	Description
31:0	TBR	RO	0x0000.FFFF	GPTM Timer B Register A read returns the current value of the **GPTM Timer B Count Register**, in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place.

Register 19: GPTM Timer A Value (GPTMTAV), offset 0x050

When read, this register shows the current, free-running value of Timer A in all modes. Software can use this value to determine the time elapsed between an interrupt and the ISR entry when using the snapshot feature with the periodic operating mode. When written, the value written into this register is loaded into the **GPTMTAR** register on the next clock cycle. In Input Edge-Count mode, bits 23:16 contain the upper 8 bits of the count.

When a GPTM is configured to one of the 32-bit modes, **GPTMTAV** appears as a 32-bit register (the upper 16-bits correspond to the contents of the **GPTM Timer B Value (GPTMTBV)** register). In a 16-bit mode, bits 15:0 contain the value of the counter and bits 23:16 contain the current, free-running value of the prescaler, which is the upper 8 bits of the count in Input Edge Count, Input Edge Time, PWM and one-shot or periodic up count modes. In one-shot or periodic down count modes, the prescaler stored in 23:16 is a true prescaler, meaning bits 23:16 count down before decrementing the value in bits 15:0. The prescaler in bits 31:24 always reads as 0.

GPTM Timer A Value (GPTMTAV)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x050
Type RW, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TAV															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

Bit/Field	Name	Type	Reset	Description
31:0	TAV	RW	0xFFFF.FFFF	GPTM Timer A Value A read returns the current, free-running value of Timer A in all modes. When written, the value written into this register is loaded into the **GPTMTAR** register on the next clock cycle.

Register 20: GPTM Timer B Value (GPTMTBV), offset 0x054

When read, this register shows the current, free-running value of Timer B in all modes. Software can use this value to determine the time elapsed between an interrupt and the ISR entry. When written, the value written into this register is loaded into the **GPTMTBR** register on the next clock cycle. In Input Edge-Count mode, bits 23:16 contain the upper 8 bits of the count.

When a GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the **GPTMTAV** register. Reads from this register return the current free-running value of Timer B. In a 16-bit mode, bits 15:0 contain the value of the counter and bits 23:16 contain the current, free-running value of the prescaler, which is the upper 8 bits of the count in Input Edge Count, Input Edge Time, PWM and one-shot or periodic up count modes. In one-shot or periodic down count modes, the prescaler stored in 23:16 is a true prescaler, meaning bits 23:16 count down before decrementing the value in bits 15:0. The prescaler in bits 31:24 always reads as 0.

GPTM Timer B Value (GPTMTBV)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x054
Type RW, reset 0x0000.FFFF

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

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

Bit/Field	Name	Type	Reset	Description
31:0	TBV	RW	0x0000.FFFF	GPTM Timer B Value A read returns the current, free-running value of Timer A in all modes. When written, the value written into this register is loaded into the **GPTMTAR** register on the next clock cycle.

11 Watchdog Timers

A watchdog timer can generate an interrupt or a reset when a time-out value is reached. The watchdog timer is used to regain control when a system has failed due to a software error or due to the failure of an external device to respond in the expected way. The LM3S9U81 microcontroller has two Watchdog Timer Modules, one module is clocked by the system clock (Watchdog Timer 0) and the other is clocked by the PIOSC (Watchdog Timer 1). The two modules are identical except that WDT1 is in a different clock domain, and therefore requires synchronizers. As a result, WDT1 has a bit defined in the **Watchdog Timer Control (WDTCTL)** register to indicate when a write to a WDT1 register is complete. Software can use this bit to ensure that the previous access has completed before starting the next access.

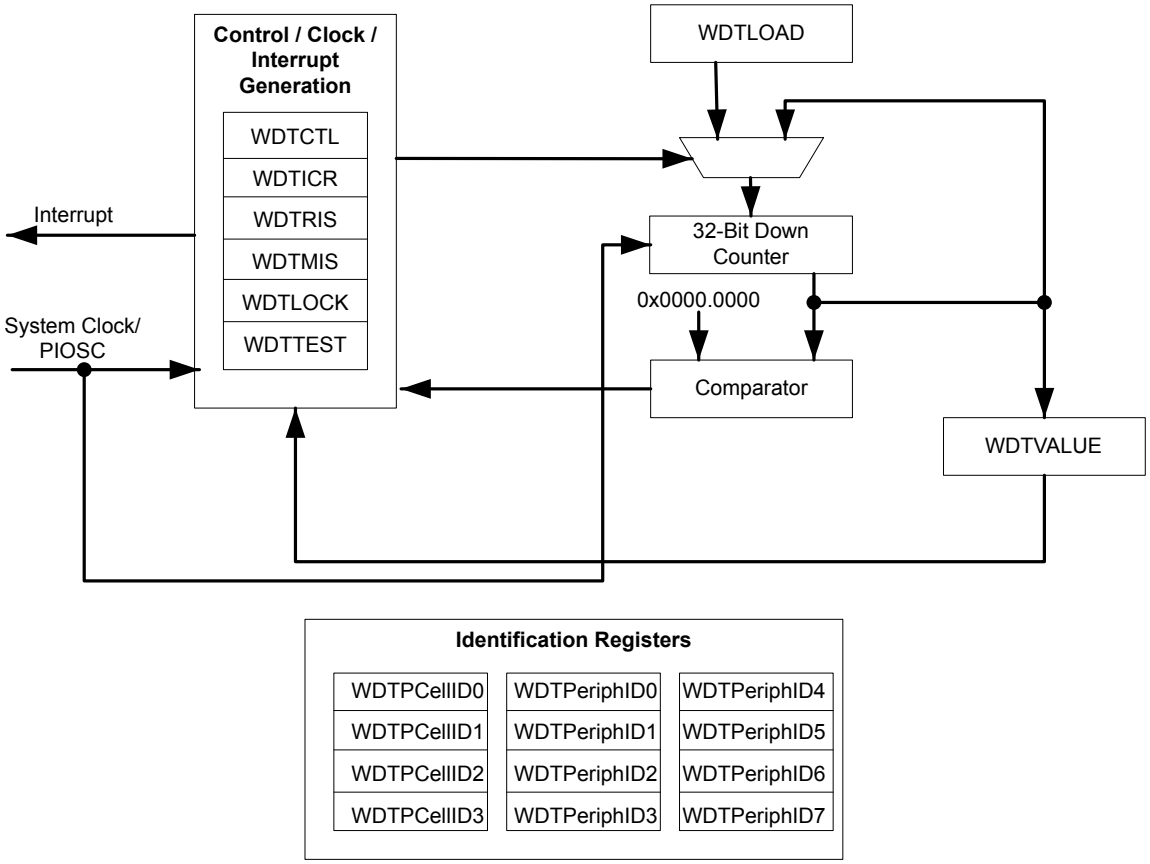
The Stellaris® LM3S9U81 controller has two Watchdog Timer modules with the following features:

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

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

11.1 Block Diagram

Figure 11-1. WDT Module Block Diagram



11.2 Functional Description

The Watchdog Timer module generates the first time-out signal when the 32-bit counter reaches the zero state after being enabled; enabling the counter also enables the watchdog timer interrupt. After the first time-out event, the 32-bit counter is re-loaded with the value of the **Watchdog Timer Load (WDTLOAD)** register, and the timer resumes counting down from that value. Once the Watchdog Timer has been configured, the **Watchdog Timer Lock (WDTLOCK)** register is written, which prevents the timer configuration from being inadvertently altered by software.

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

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

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

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

11.2.1 Register Access Timing

Because the Watchdog Timer 1 module has an independent clocking domain, its registers must be written with a timing gap between accesses. Software must guarantee that this delay is inserted between back-to-back writes to WDT1 registers or between a write followed by a read to the registers. The timing for back-to-back reads from the WDT1 module has no restrictions. The WRC bit in the **Watchdog Control (WDTCTL)** register for WDT1 indicates that the required timing gap has elapsed. This bit is cleared on a write operation and set once the write completes, indicating to software that another write or read may be started safely. Software should poll **WDTCTL** for WRC=1 prior to accessing another register. Note that WDT0 does not have this restriction as it runs off the system clock.

11.3 Initialization and Configuration

To use the WDT, its peripheral clock must be enabled by setting the WDT bit in the **RCGC0** register, see page 255.

The Watchdog Timer is configured using the following sequence:

1. Load the **WDTLOAD** register with the desired timer load value.
2. If WDT1, wait for the WRC bit in the **WDTCTL** register to be set.
3. If the Watchdog is configured to trigger system resets, set the RESEN bit in the **WDTCTL** register.
4. If WDT1, wait for the WRC bit in the **WDTCTL** register to be set.
5. Set the INTEN bit in the **WDTCTL** register to enable the Watchdog and lock the control register.

If software requires that all of the watchdog registers are locked, the Watchdog Timer module can be fully locked by writing any value to the **WDTLOCK** register. To unlock the Watchdog Timer, write a value of 0x1ACC.E551.

11.4 Register Map

Table 11-1 on page 576 lists the Watchdog registers. The offset listed is a hexadecimal increment to the register's address, relative to the Watchdog Timer base address:

- WDT0: 0x4000.0000
- WDT1: 0x4000.1000

Note that the Watchdog Timer module clock must be enabled before the registers can be programmed (see page 255).

Table 11-1. Watchdog Timers Register Map

Offset	Name	Type	Reset	Description	See page
0x000	WDTLOAD	R/W	0xFFFF.FFFF	Watchdog Load	577
0x004	WDTVALUE	RO	0xFFFF.FFFF	Watchdog Value	578
0x008	WDTCTL	R/W	0x0000.0000 (WDT0) 0x8000.0000 (WDT1)	Watchdog Control	579
0x00C	WDTICR	WO	-	Watchdog Interrupt Clear	581
0x010	WDTRIS	RO	0x0000.0000	Watchdog Raw Interrupt Status	582
0x014	WDTMIS	RO	0x0000.0000	Watchdog Masked Interrupt Status	583
0x418	WDTTEST	R/W	0x0000.0000	Watchdog Test	584
0xC00	WDTLOCK	R/W	0x0000.0000	Watchdog Lock	585
0xFD0	WDTPeriphID4	RO	0x0000.0000	Watchdog Peripheral Identification 4	586
0xFD4	WDTPeriphID5	RO	0x0000.0000	Watchdog Peripheral Identification 5	587
0xFD8	WDTPeriphID6	RO	0x0000.0000	Watchdog Peripheral Identification 6	588
0xFDC	WDTPeriphID7	RO	0x0000.0000	Watchdog Peripheral Identification 7	589
0xFE0	WDTPeriphID0	RO	0x0000.0005	Watchdog Peripheral Identification 0	590
0xFE4	WDTPeriphID1	RO	0x0000.0018	Watchdog Peripheral Identification 1	591
0xFE8	WDTPeriphID2	RO	0x0000.0018	Watchdog Peripheral Identification 2	592
0xFEC	WDTPeriphID3	RO	0x0000.0001	Watchdog Peripheral Identification 3	593
0xFF0	WDTPCellID0	RO	0x0000.000D	Watchdog PrimeCell Identification 0	594
0xFF4	WDTPCellID1	RO	0x0000.00F0	Watchdog PrimeCell Identification 1	595
0xFF8	WDTPCellID2	RO	0x0000.0006	Watchdog PrimeCell Identification 2	596
0xFFC	WDTPCellID3	RO	0x0000.00B1	Watchdog PrimeCell Identification 3	597

11.5 Register Descriptions

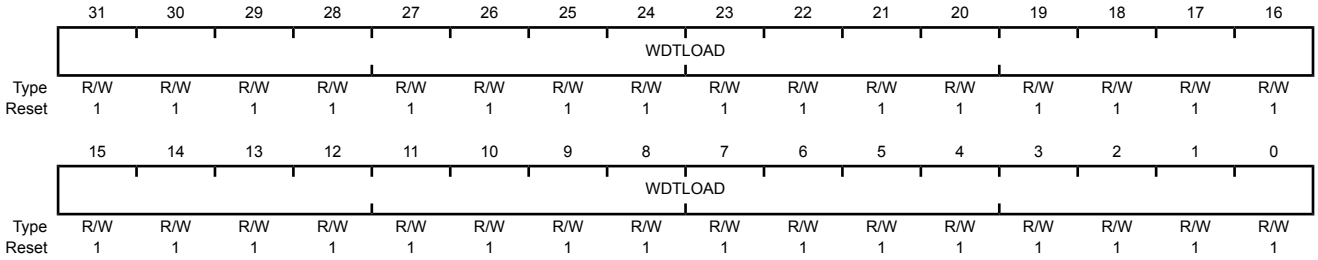
The remainder of this section lists and describes the WDT registers, in numerical order by address offset.

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

This register is the 32-bit interval value used by the 32-bit counter. When this register is written, the value is immediately loaded and the counter restarts counting down from the new value. If the **WDTLOAD** register is loaded with 0x0000.0000, an interrupt is immediately generated.

Watchdog Load (WDTLOAD)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x000
Type R/W, reset 0xFFFF.FFFF



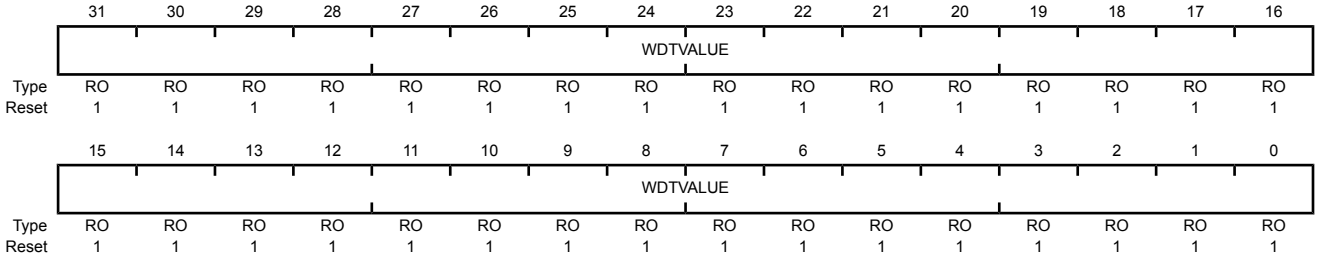
Bit/Field	Name	Type	Reset	Description
31:0	WDTLOAD	R/W	0xFFFF.FFFF	Watchdog Load Value

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

This register contains the current count value of the timer.

Watchdog Value (WDTVALUE)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x004
Type RO, reset 0xFFFF.FFFF



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

Register 3: Watchdog Control (WDTCTL), offset 0x008

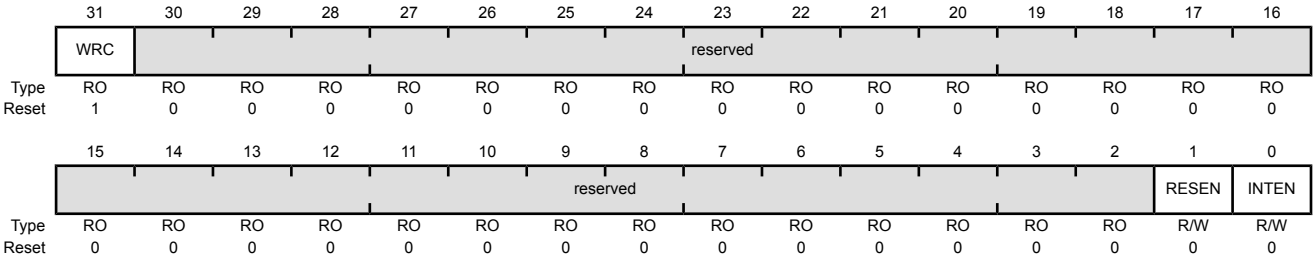
This register is the watchdog control register. The watchdog timer can be configured to generate a reset signal (on second time-out) or an interrupt on time-out.

When the watchdog interrupt has been enabled by setting the INTEN bit, all subsequent writes to the INTEN bit are ignored. The only mechanism that can re-enable writes to this bit is a hardware reset.

Important: Because the Watchdog Timer 1 module has an independent clocking domain, its registers must be written with a timing gap between accesses. Software must guarantee that this delay is inserted between back-to-back writes to WDT1 registers or between a write followed by a read to the registers. The timing for back-to-back reads from the WDT1 module has no restrictions. The WRC bit in the **Watchdog Control (WDTCTL)** register for WDT1 indicates that the required timing gap has elapsed. This bit is cleared on a write operation and set once the write completes, indicating to software that another write or read may be started safely. Software should poll **WDTCTL** for WRC=1 prior to accessing another register. Note that WDT0 does not have this restriction as it runs off the system clock and therefore does not have a WRC bit.

Watchdog Control (WDTCTL)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x008
Type R/W, reset 0x0000.0000 (WDT0) and 0x8000.0000 (WDT1)



Bit/Field	Name	Type	Reset	Description
31	WRC	RO	1	Write Complete The WRC values are defined as follows: Value Description 0 A write access to one of the WDT1 registers is in progress. 1 A write access is not in progress, and WDT1 registers can be read or written. **Note:** This bit is reserved for WDT0 and has a reset value of 0.
30:2	reserved	RO	0x000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

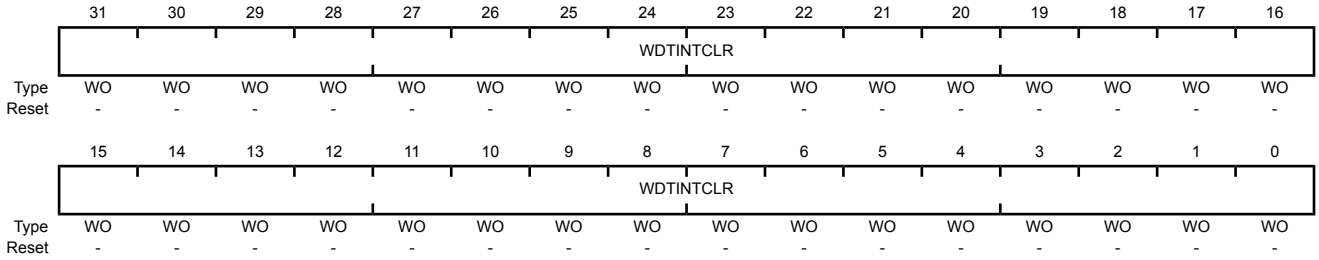
Bit/Field	Name	Type	Reset	Description
1	RESEN	R/W	0	Watchdog Reset Enable The RESEN values are defined as follows: Value Description 0 Disabled. 1 Enable the Watchdog module reset output.
0	INTEN	R/W	0	Watchdog Interrupt Enable The INTEN values are defined as follows: Value Description 0 Interrupt event disabled (once this bit is set, it can only be cleared by a hardware reset). 1 Interrupt event enabled. Once enabled, all writes are ignored.

Register 4: Watchdog Interrupt Clear (WDTICR), offset 0x00C

This register is the interrupt clear register. A write of any value to this register clears the Watchdog interrupt and reloads the 32-bit counter from the **WDTLOAD** register. Value for a read or reset is indeterminate.

Watchdog Interrupt Clear (WDTICR)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x00C
Type WO, reset -



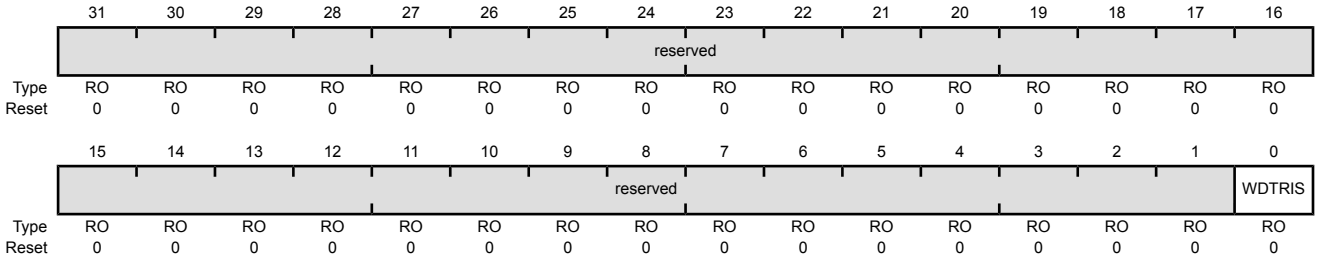
Bit/Field	Name	Type	Reset	Description
31:0	WDTINTCLR	WO	-	Watchdog Interrupt Clear

Register 5: Watchdog Raw Interrupt Status (WDTRIS), offset 0x010

This register is the raw interrupt status register. Watchdog interrupt events can be monitored via this register if the controller interrupt is masked.

Watchdog Raw Interrupt Status (WDTRIS)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x010
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	WDTRIS	RO	0	Watchdog Raw Interrupt Status

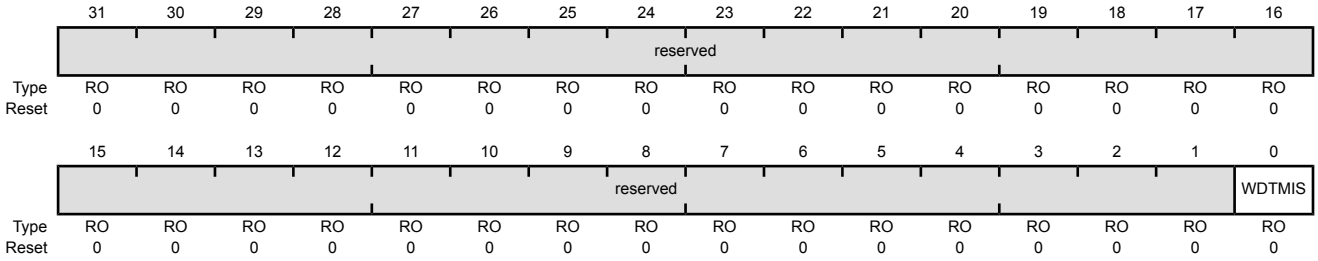
Value	Description
1	A watchdog time-out event has occurred.
0	The watchdog has not timed out.

Register 6: Watchdog Masked Interrupt Status (WDTMIS), offset 0x014

This register is the masked interrupt status register. The value of this register is the logical AND of the raw interrupt bit and the Watchdog interrupt enable bit.

Watchdog Masked Interrupt Status (WDTMIS)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x014
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	WDTMIS	RO	0	Watchdog Masked Interrupt Status

Value	Description
1	A watchdog time-out event has been signalled to the interrupt controller.
0	The watchdog has not timed out or the watchdog timer interrupt is masked.

Register 7: Watchdog Test (WDTTEST), offset 0x418

This register provides user-enabled stalling when the microcontroller asserts the CPU halt flag during debug.

Watchdog Test (WDTTEST)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x418
Type R/W, reset 0x0000.0000

Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	STALL	R/W	0	Watchdog Stall Enable Value Description 1 If the microcontroller is stopped with a debugger, the watchdog timer stops counting. Once the microcontroller is restarted, the watchdog timer resumes counting. 0 The watchdog timer continues counting if the microcontroller is stopped with a debugger.
7:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 8: Watchdog Lock (WDTLOCK), offset 0xC00

Writing 0x1ACC.E551 to the **WDTLOCK** register enables write access to all other registers. Writing any other value to the **WDTLOCK** register re-enables the locked state for register writes to all the other registers. Reading the **WDTLOCK** register returns the lock status rather than the 32-bit value written. Therefore, when write accesses are disabled, reading the **WDTLOCK** register returns 0x0000.0001 (when locked; otherwise, the returned value is 0x0000.0000 (unlocked)).

Watchdog Lock (WDTLOCK)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xC00
Type R/W, reset 0x0000.0000

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

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

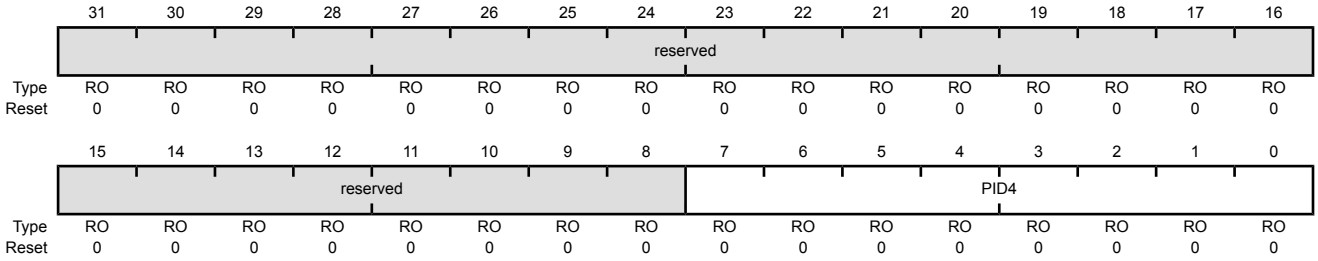
Bit/Field	Name	Type	Reset	Description
31:0	WDTLOCK	R/W	0x0000.0000	Watchdog Lock A write of the value 0x1ACC.E551 unlocks the watchdog registers for write access. A write of any other value reapplies the lock, preventing any register updates. A read of this register returns the following values: Value Description 0x0000.0001 Locked 0x0000.0000 Unlocked

Register 9: Watchdog Peripheral Identification 4 (WDTPeriphID4), offset 0xFD0

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

Watchdog Peripheral Identification 4 (WDTPeriphID4)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFD0
Type RO, reset 0x0000.0000



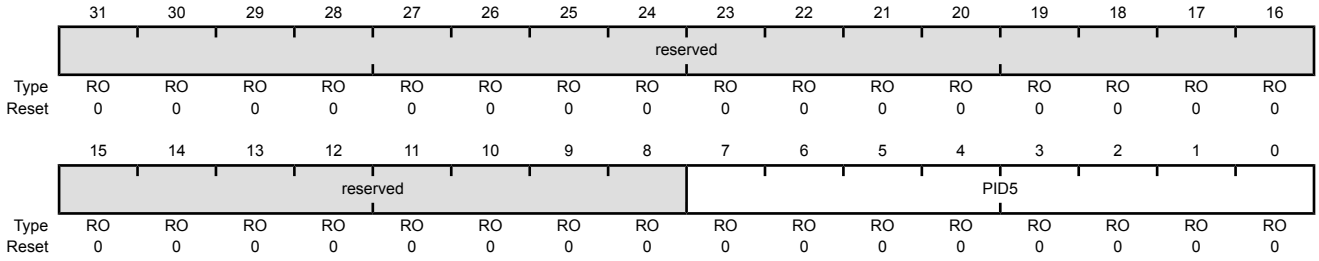
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	WDT Peripheral ID Register [7:0]

Register 10: Watchdog Peripheral Identification 5 (WDTPeriphID5), offset 0xFD4

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

Watchdog Peripheral Identification 5 (WDTPeriphID5)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFD4
Type RO, reset 0x0000.0000



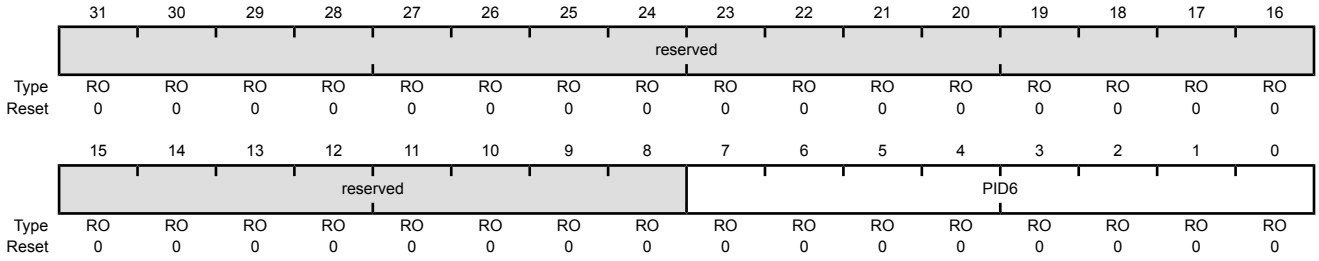
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	WDT Peripheral ID Register [15:8]

Register 11: Watchdog Peripheral Identification 6 (WDTPeriphID6), offset 0xFD8

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

Watchdog Peripheral Identification 6 (WDTPeriphID6)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFD8
Type RO, reset 0x0000.0000



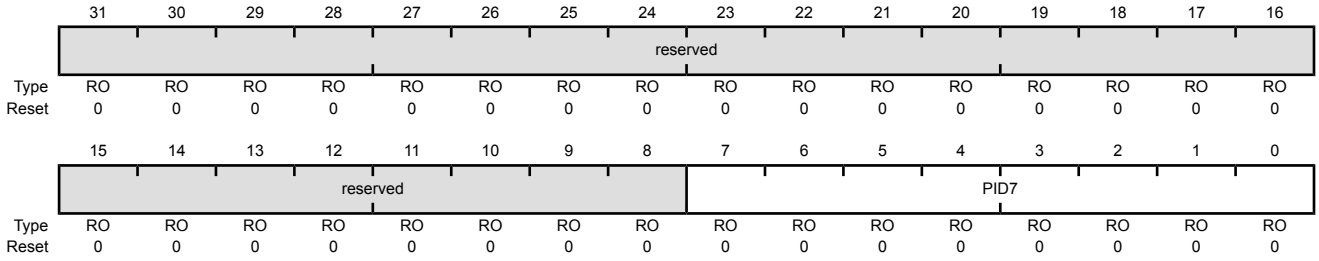
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	WDT Peripheral ID Register [23:16]

Register 12: Watchdog Peripheral Identification 7 (WDTPeriphID7), offset 0xFDC

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

Watchdog Peripheral Identification 7 (WDTPeriphID7)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFDC
Type RO, reset 0x0000.0000



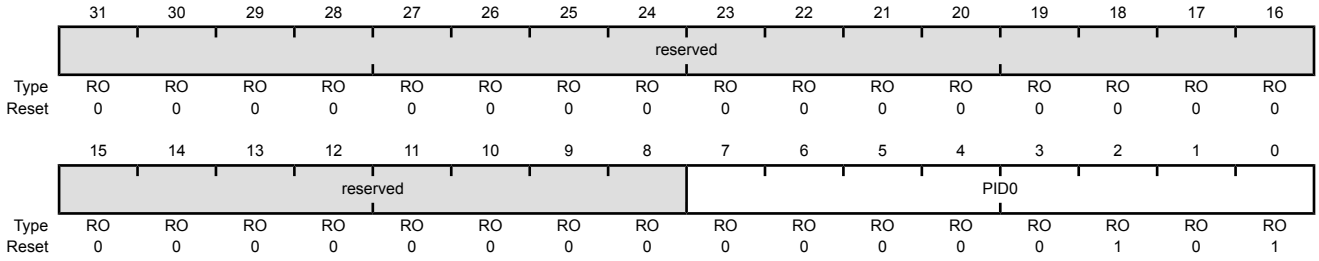
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	WDT Peripheral ID Register [31:24]

Register 13: Watchdog Peripheral Identification 0 (WDTPeriphID0), offset 0xFE0

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

Watchdog Peripheral Identification 0 (WDTPeriphID0)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFE0
Type RO, reset 0x0000.0005



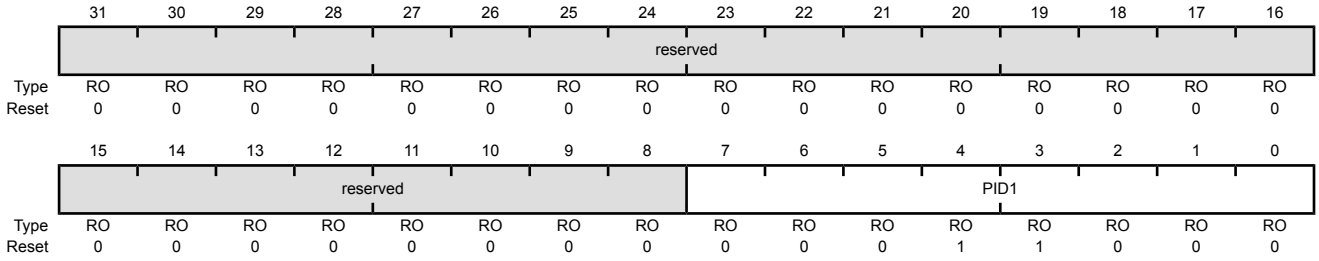
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x05	Watchdog Peripheral ID Register [7:0]

Register 14: Watchdog Peripheral Identification 1 (WDTPeriphID1), offset 0xFE4

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

Watchdog Peripheral Identification 1 (WDTPeriphID1)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFE4
Type RO, reset 0x0000.0018



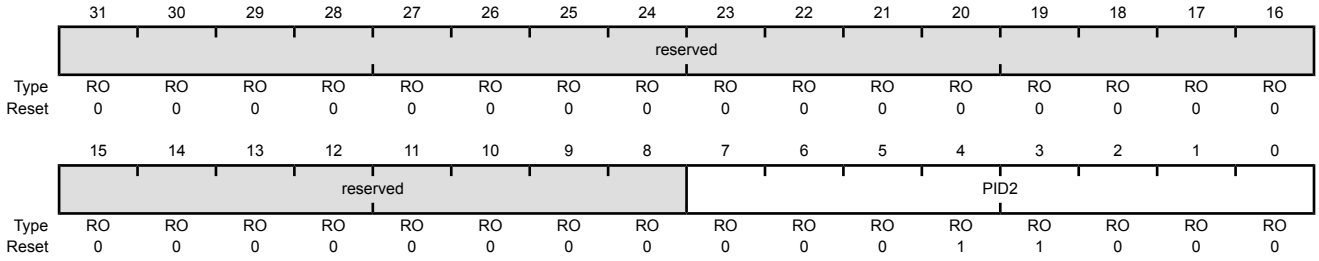
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x18	Watchdog Peripheral ID Register [15:8]

Register 15: Watchdog Peripheral Identification 2 (WDTPeriphID2), offset 0xFE8

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

Watchdog Peripheral Identification 2 (WDTPeriphID2)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFE8
Type RO, reset 0x0000.0018



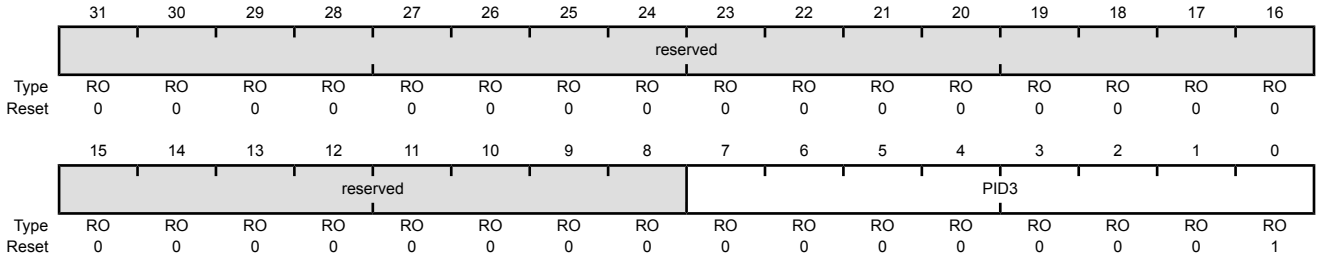
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	Watchdog Peripheral ID Register [23:16]

Register 16: Watchdog Peripheral Identification 3 (WDTPeriphID3), offset 0xFEC

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

Watchdog Peripheral Identification 3 (WDTPeriphID3)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFEC
Type RO, reset 0x0000.0001



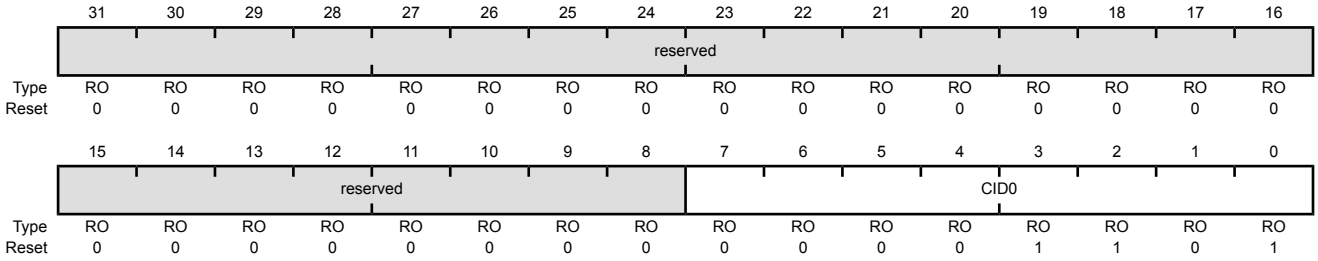
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	Watchdog Peripheral ID Register [31:24]

Register 17: Watchdog PrimeCell Identification 0 (WDTPCellID0), offset 0xFF0

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

Watchdog PrimeCell Identification 0 (WDTPCellID0)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFF0
Type RO, reset 0x0000.000D



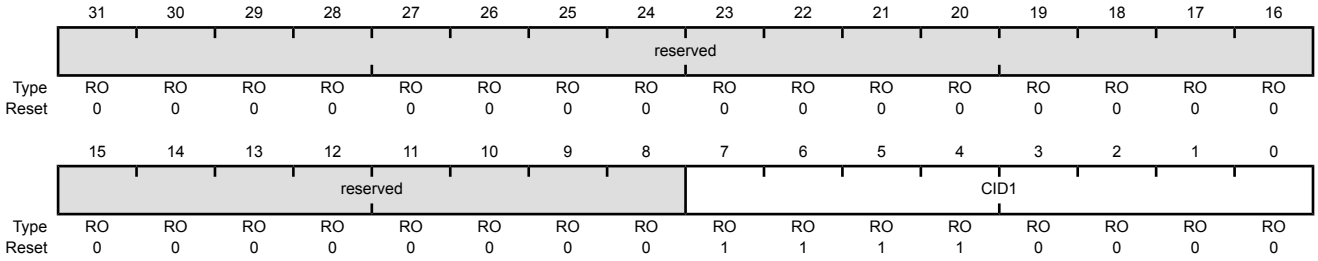
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	Watchdog PrimeCell ID Register [7:0]

Register 18: Watchdog PrimeCell Identification 1 (WDTPCellID1), offset 0xFF4

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

Watchdog PrimeCell Identification 1 (WDTPCellID1)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFF4
Type RO, reset 0x0000.00F0



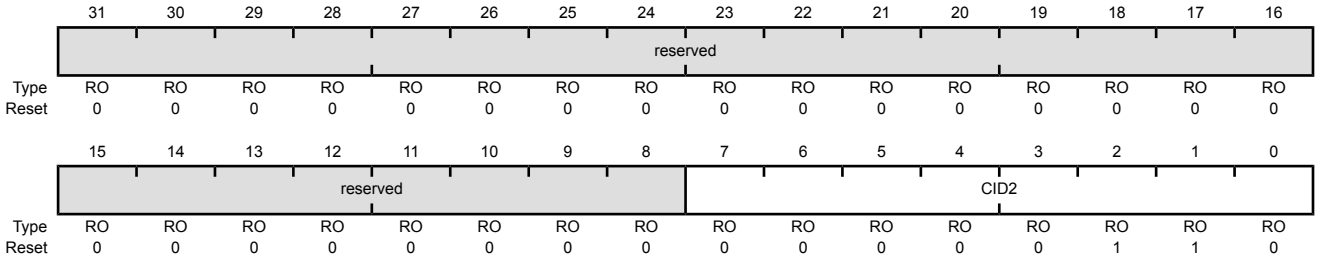
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	Watchdog PrimeCell ID Register [15:8]

Register 19: Watchdog PrimeCell Identification 2 (WDTPCellID2), offset 0xFF8

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

Watchdog PrimeCell Identification 2 (WDTPCellID2)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFF8
Type RO, reset 0x0000.0006



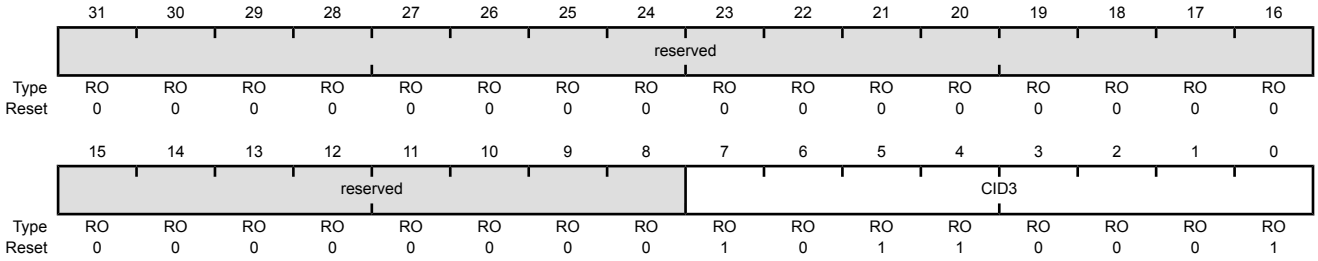
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x06	Watchdog PrimeCell ID Register [23:16]

Register 20: Watchdog PrimeCell Identification 3 (WDTPCellID3), offset 0xFFC

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

Watchdog PrimeCell Identification 3 (WDTPCellID3)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFFC
Type RO, reset 0x0000.00B1



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	Watchdog PrimeCell ID Register [31:24]

12 Analog-to-Digital Converter (ADC)

An analog-to-digital converter (ADC) is a peripheral that converts a continuous analog voltage to a discrete digital number. Two identical converter modules are included, which share 16 input channels.

The Stellaris® ADC module features 12-bit conversion resolution and supports 16 input channels, plus an internal temperature sensor. Each ADC module contains four programmable sequencers allowing the sampling of multiple analog input sources without controller intervention. Each sample sequencer provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequencer priority. A digital comparator function is included which allows the conversion value to be diverted to a digital comparator module. Each ADC module provides eight digital comparators. Each digital comparator evaluates the ADC conversion value against its two user-defined values to determine the operational range of the signal. The trigger source for ADC0 and ADC1 may be independent or the two ADC modules may operate from the same trigger source and operate on the same or different inputs. A phase shifter can delay the start of sampling by a specified phase angle. When using both ADC modules, it is possible to configure the converters to start the conversions coincidentally or within a relative phase from each other, see “Sample Phase Control” on page 604.

The Stellaris LM3S9U81 microcontroller provides two ADC modules with each having the following features:

- 16 shared analog input channels
- 12-bit precision ADC with an accurate 10-bit data compatibility mode
- Single-ended and differential-input configurations
- On-chip internal temperature sensor
- Maximum sample rate of one million samples/second
- Optional phase shift in sample time programmable from 22.5º to 337.5º
- Four programmable sample conversion sequencers from one to eight entries long, with corresponding conversion result FIFOs
- Flexible trigger control
  - Controller (software)
  - Timers
  - Analog Comparators
  - GPIO
- Hardware averaging of up to 64 samples
- Digital comparison unit providing eight digital comparators
- Converter uses an internal 3-V reference or an external reference
- Power and ground for the analog circuitry is separate from the digital power and ground

- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Dedicated channel for each sample sequencer
  - ADC module uses burst requests for DMA

12.1 Block Diagram

The Stellaris microcontroller contains two identical Analog-to-Digital Converter modules. These two modules, ADC0 and ADC1, share the same 16 analog input channels. Each ADC module operates independently and can therefore execute different sample sequences, sample any of the analog input channels at any time, and generate different interrupts and triggers. Figure 12-1 on page 599 shows how the two modules are connected to analog inputs and the system bus.

Figure 12-1. Implementation of Two ADC Blocks

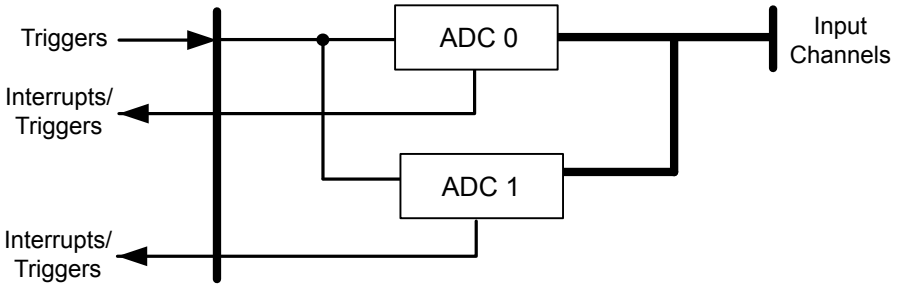
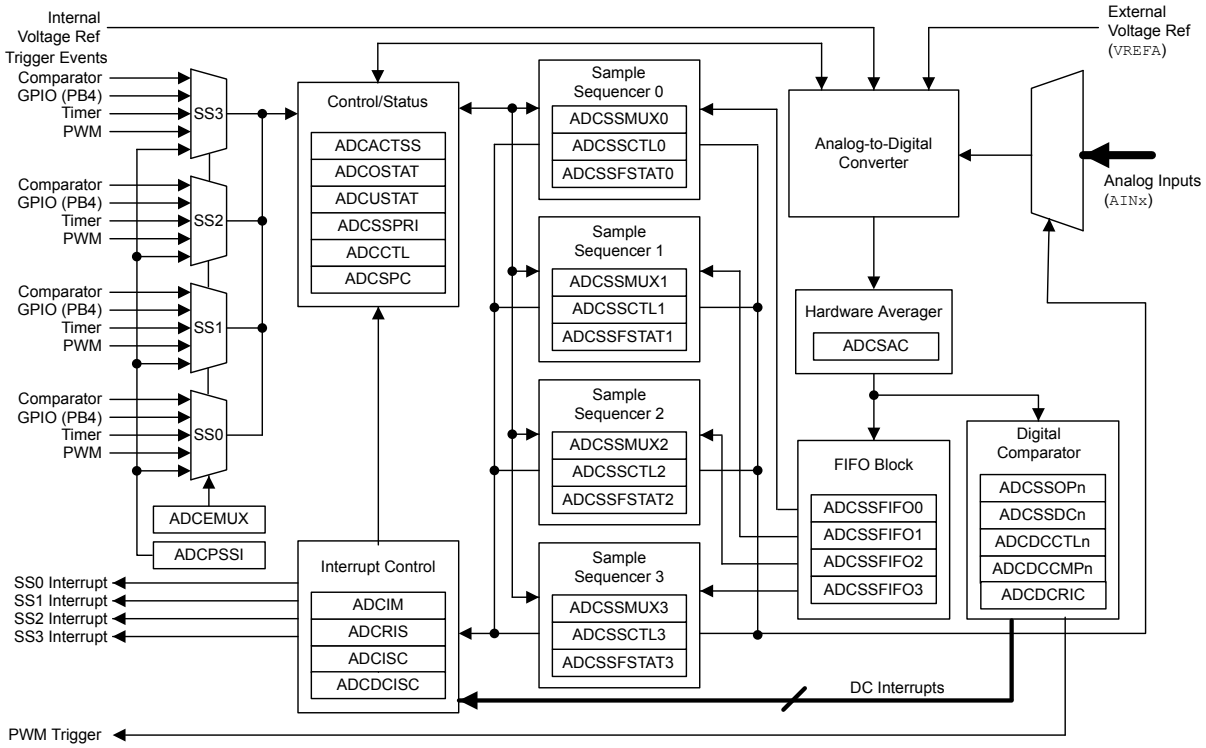


Figure 12-2 on page 600 provides details on the internal configuration of the ADC controls and data registers.

Figure 12-2. ADC Module Block Diagram



12.2 Signal Description

The following table lists the external signals of the ADC module and describes the function of each. The ADC signals are analog functions for some GPIO signals. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placement for the ADC signals. The AINx and VREFA analog signals are not 5-V tolerant and go through an isolation circuit before reaching their circuitry. These signals are configured by clearing the corresponding DEN bit in the **GPIO Digital Enable (GPIODEN)** register and setting the corresponding AMSEL bit in the **GPIO Analog Mode Select (GPIOAMSEL)** register. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 395.

Table 12-1. ADC Signals (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
AIN0	1	PE7	I	Analog	Analog-to-digital converter input 0.
AIN1	2	PE6	I	Analog	Analog-to-digital converter input 1.
AIN2	5	PE5	I	Analog	Analog-to-digital converter input 2.
AIN3	6	PE4	I	Analog	Analog-to-digital converter input 3.
AIN4	100	PD7	I	Analog	Analog-to-digital converter input 4.
AIN5	99	PD6	I	Analog	Analog-to-digital converter input 5.
AIN6	98	PD5	I	Analog	Analog-to-digital converter input 6.
AIN7	97	PD4	I	Analog	Analog-to-digital converter input 7.
AIN8	96	PE3	I	Analog	Analog-to-digital converter input 8.
AIN9	95	PE2	I	Analog	Analog-to-digital converter input 9.

Table 12-1. ADC Signals (100LQFP) *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
AIN10	92	PB4	I	Analog	Analog-to-digital converter input 10.
AIN11	91	PB5	I	Analog	Analog-to-digital converter input 11.
AIN12	13	PD3	I	Analog	Analog-to-digital converter input 12.
AIN13	12	PD2	I	Analog	Analog-to-digital converter input 13.
AIN14	11	PD1	I	Analog	Analog-to-digital converter input 14.
AIN15	10	PD0	I	Analog	Analog-to-digital converter input 15.
VREFA	90	PB6	I	Analog	This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AINn signal is converted to 4095. The VREFA input is limited to the range specified in Table 24-23 on page 1208 .

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

Table 12-2. ADC Signals (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
AIN0	B1	PE7	I	Analog	Analog-to-digital converter input 0.
AIN1	A1	PE6	I	Analog	Analog-to-digital converter input 1.
AIN2	B3	PE5	I	Analog	Analog-to-digital converter input 2.
AIN3	B2	PE4	I	Analog	Analog-to-digital converter input 3.
AIN4	A2	PD7	I	Analog	Analog-to-digital converter input 4.
AIN5	A3	PD6	I	Analog	Analog-to-digital converter input 5.
AIN6	C6	PD5	I	Analog	Analog-to-digital converter input 6.
AIN7	B5	PD4	I	Analog	Analog-to-digital converter input 7.
AIN8	B4	PE3	I	Analog	Analog-to-digital converter input 8.
AIN9	A4	PE2	I	Analog	Analog-to-digital converter input 9.
AIN10	A6	PB4	I	Analog	Analog-to-digital converter input 10.
AIN11	B7	PB5	I	Analog	Analog-to-digital converter input 11.
AIN12	H1	PD3	I	Analog	Analog-to-digital converter input 12.
AIN13	H2	PD2	I	Analog	Analog-to-digital converter input 13.
AIN14	G2	PD1	I	Analog	Analog-to-digital converter input 14.
AIN15	G1	PD0	I	Analog	Analog-to-digital converter input 15.
VREFA	A7	PB6	I	Analog	This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AINn signal is converted to 4095. The VREFA input is limited to the range specified in Table 24-23 on page 1208 .

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

12.3 Functional Description

The Stellaris ADC collects sample data by using a programmable sequence-based approach instead of the traditional single or double-sampling approaches found on many ADC modules. Each *sample sequence* is a fully programmed series of consecutive (back-to-back) samples, allowing the ADC to collect data from multiple input sources without having to be re-configured or serviced by the processor. The programming of each sample in the sample sequence includes parameters such as the input source and mode (differential versus single-ended input), interrupt generation on sample completion, and the indicator for the last sample in the sequence. In addition, the µDMA can be used to more efficiently move data from the sample sequencers without CPU intervention.

12.3.1 Sample Sequencers

The sampling control and data capture is handled by the sample sequencers. All of the sequencers are identical in implementation except for the number of samples that can be captured and the depth of the FIFO. Table 12-3 on page 602 shows the maximum number of samples that each sequencer can capture and its corresponding FIFO depth. Each sample that is captured is stored in the FIFO. In this implementation, each FIFO entry is a 32-bit word, with the lower 12 bits containing the conversion result.

Table 12-3. Samples and FIFO Depth of Sequencers

Sequencer	Number of Samples	Depth of FIFO
SS3	1	1
SS2	4	4
SS1	4	4
SS0	8	8

For a given sample sequence, each sample is defined by bit fields in the **ADC Sample Sequence Input Multiplexer Select (ADCSSMUXn)** and **ADC Sample Sequence Control (ADCSSCTLn)** registers, where "n" corresponds to the sequence number. The **ADCSSMUXn** fields select the input pin, while the **ADCSSCTLn** fields contain the sample control bits corresponding to parameters such as temperature sensor selection, interrupt enable, end of sequence, and differential input mode. Sample sequencers are enabled by setting the respective ASENn bit in the **ADC Active Sample Sequencer (ADCACTSS)** register and should be configured before being enabled. Sampling is then initiated by setting the SSn bit in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register. In addition, sample sequences may be initiated on multiple ADC modules simultaneously using the GSYNC and SYNCWAIT bits in the **ADCPSSI** register during the configuration of each ADC module. For more information on using these bits, refer to page 641.

When configuring a sample sequence, multiple uses of the same input pin within the same sequence are allowed. In the **ADCSSCTLn** register, the IEn bits can be set for any combination of samples, allowing interrupts to be generated after every sample in the sequence if necessary. Also, the END bit can be set at any point within a sample sequence. For example, if Sequencer 0 is used, the END bit can be set in the nibble associated with the fifth sample, allowing Sequencer 0 to complete execution of the sample sequence after the fifth sample.

After a sample sequence completes execution, the result data can be retrieved from the **ADC Sample Sequence Result FIFO (ADCSSFIFOn)** registers. The FIFOs are simple circular buffers that read a single address to "pop" result data. For software debug purposes, the positions of the FIFO head and tail pointers are visible in the **ADC Sample Sequence FIFO Status (ADCSSFSTATn)** registers along with FULL and EMPTY status flags. If a write is attempted when the FIFO is full, the write does not occur and an overflow condition is indicated. Overflow and underflow conditions are monitored using the **ADCOSTAT** and **ADCUSTAT** registers.

12.3.2 Module Control

Outside of the sample sequencers, the remainder of the control logic is responsible for tasks such as:

- Interrupt generation
- DMA operation
- Sequence prioritization
- Trigger configuration
- Comparator configuration
- External voltage reference
- Sample phase control

Most of the ADC control logic runs at the ADC clock rate of 16 MHz. The internal ADC divider is configured for 16-MHz operation automatically by hardware when the system XTAL is selected with the PLL.

12.3.2.1 Interrupts

The register configurations of the sample sequencers and digital comparators dictate which events generate raw interrupts, but do not have control over whether the interrupt is actually sent to the interrupt controller. The ADC module's interrupt signals are controlled by the state of the MASK bits in the **ADC Interrupt Mask (ADCIM)** register. Interrupt status can be viewed at two locations: the **ADC Raw Interrupt Status (ADCRIS)** register, which shows the raw status of the various interrupt signals; and the **ADC Interrupt Status and Clear (ADCISC)** register, which shows active interrupts that are enabled by the **ADCIM** register. Sequencer interrupts are cleared by writing a 1 to the corresponding IN bit in **ADCISC**. Digital comparator interrupts are cleared by writing a 1 to the **ADC Digital Comparator Interrupt Status and Clear (ADCDCISC)** register.

12.3.2.2 DMA Operation

DMA may be used to increase efficiency by allowing each sample sequencer to operate independently and transfer data without processor intervention or reconfiguration. The ADC module provides a request signal from each sample sequencer to the associated dedicated channel of the µDMA controller. The ADC does not support single transfer requests. A burst transfer request is asserted when the interrupt bit for the sample sequence is set (IE bit in the **ADCSSCTLn** register is set).

The arbitration size of the µDMA transfer must be a power of 2, and the associated IE bits in the **ADDSSCTLn** register must be set. For example, if the µDMA channel of SS0 has an arbitration size of four, the IE3 bit (4th sample) and the IE7 bit (8th sample) must be set. Thus the µDMA request occurs every time 4 samples have been acquired. No other special steps are needed to enable the ADC module for µDMA operation.

Refer to the “Micro Direct Memory Access (µDMA)” on page 334 for more details about programming the µDMA controller.

12.3.2.3 Prioritization

When sampling events (triggers) happen concurrently, they are prioritized for processing by the values in the **ADC Sample Sequencer Priority (ADCSSPRI)** register. Valid priority values are in the range of 0-3, with 0 being the highest priority and 3 being the lowest. Multiple active sample

sequencer units with the same priority do not provide consistent results, so software must ensure that all active sample sequencer units have a unique priority value.

12.3.2.4 Sampling Events

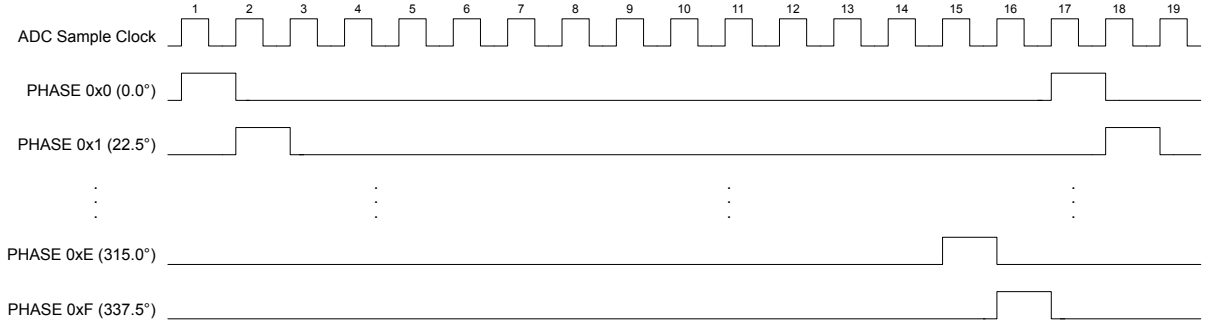
Sample triggering for each sample sequencer is defined in the **ADC Event Multiplexer Select (ADCEMUX)** register. Trigger sources include processor (default), analog comparators, an external signal on GPIO PB4, a GP Timer, and continuous sampling. The processor triggers sampling by setting the SSx bits in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register.

Care must be taken when using the continuous sampling trigger. If a sequencer's priority is too high, it is possible to starve other lower priority sequencers. Generally, a sample sequencer using continuous sampling should be set to the lowest priority. Continuous sampling can be used with a digital comparator to cause an interrupt when a particular voltage is seen on an input.

12.3.2.5 Sample Phase Control

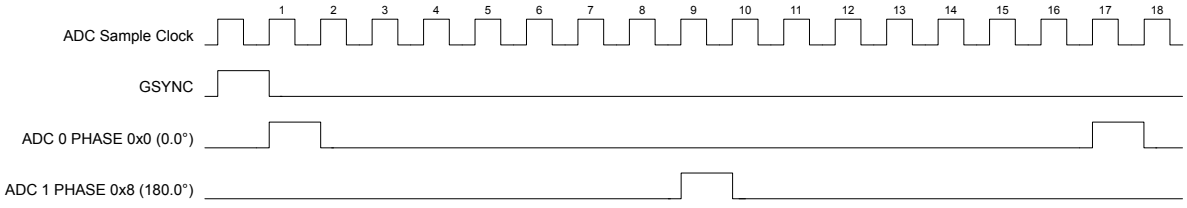
The trigger source for ADC0 and ADC1 may be independent or the two ADC modules may operate from the same trigger source and operate on the same or different inputs. If the converters are running at the same sample rate, they may be configured to start the conversions coincidentally or with one of 15 different discrete phases relative to each other. The sample time can be delayed from the standard sampling time in 22.5° increments up to 337.5º using the **ADC Sample Phase Control (ADCSPC)** register. Figure 12-3 on page 604 shows an example of various phase relationships at a 1 Msps rate.

Figure 12-3. ADC Sample Phases



This feature can be used to double the sampling rate of an input. Both ADC module 0 and ADC module 1 can be programmed to sample the same input. ADC module 0 could sample at the standard position (the PHASE field in the **ADCSPC** register is 0x0). ADC module 1 can be configured to sample at 180 (PHASE = 0x8). The two modules can be be synchronized using the GSYNC and SYNCWAIT bits in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register. Software could then combine the results from the two modules to create a sample rate of two million samples/second at 16 MHz as shown in Figure 12-4 on page 604.

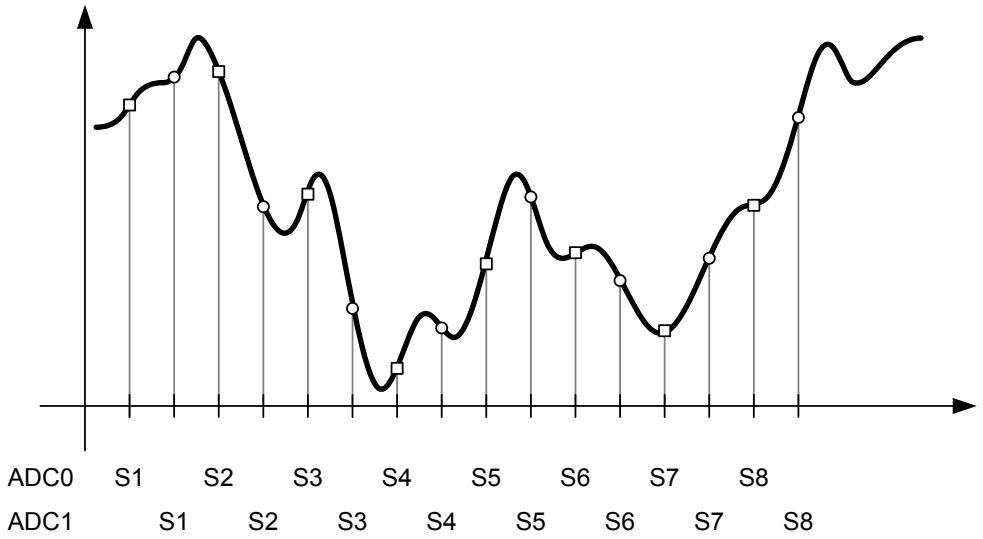
Figure 12-4. Doubling the ADC Sample Rate



Using the **ADCSPC** register, ADC0 and ADC1 may provide a number of interesting applications:

- Coincident sampling of different signals. The sample sequence steps run coincidently in both converters.
  - ADC Module 0, **ADCSPC** = 0x0, sampling AIN0
  - ADC Module 1, **ADCSPC** = 0x0, sampling AIN1
- Skewed sampling of the same signal. The sample sequence steps are 1/2 of an ADC clock (500 µs for a 1Ms/s ADC) out of phase with each other. This configuration doubles the conversion bandwidth of a single input when software combines the results as shown in Figure 12-5 on page 605.
  - ADC Module 0, **ADCSPC** = 0x0, sampling AIN0
  - ADC Module 1, **ADCSPC** = 0x8, sampling AIN0

Figure 12-5. Skewed Sampling



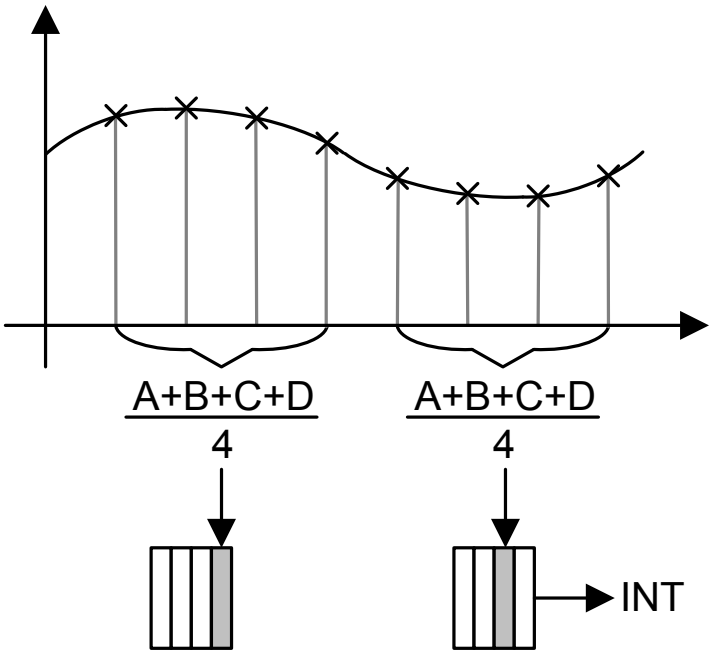
12.3.3 Hardware Sample Averaging Circuit

Higher precision results can be generated using the hardware averaging circuit, however, the improved results are at the cost of throughput. Up to 64 samples can be accumulated and averaged to form a single data entry in the sequencer FIFO. Throughput is decreased proportionally to the number of samples in the averaging calculation. For example, if the averaging circuit is configured to average 16 samples, the throughput is decreased by a factor of 16.

By default the averaging circuit is off, and all data from the converter passes through to the sequencer FIFO. The averaging hardware is controlled by the **ADC Sample Averaging Control (ADCSAC)** register (see page 643). A single averaging circuit has been implemented, thus all input channels receive the same amount of averaging whether they are single-ended or differential.

Figure 12-6 shows an example in which the **ADCSAC** register is set to 0x2 for 4x hardware oversampling and the IE1 bit is set for the sample sequence, resulting in an interrupt after the second averaged value is stored in the FIFO.

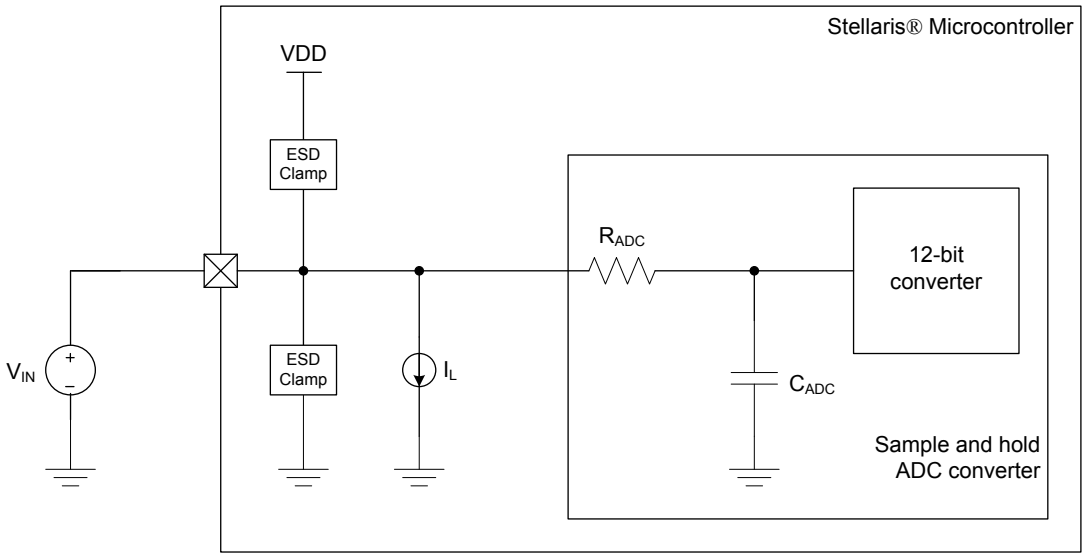
Figure 12-6. Sample Averaging Example



12.3.4 Analog-to-Digital Converter

The Analog-to-Digital Converter (ADC) module uses a Successive Approximation Register (SAR) architecture to deliver a 12-bit, low-power, high-precision conversion value. The ADC defaults to a 10-bit conversion result, providing backwards compatibility with previous generations of Stellaris microcontrollers. To enable 12-bit resolution, set the RES bit in the **ADC Control (ADCCTL)** register. The successive-approximation algorithm uses a current mode D/A converter to achieve lower settling time, resulting in higher conversion speeds for the A/D converter. In addition, built-in sample-and-hold circuitry with offset-calibration circuitry improves conversion accuracy. The ADC must be run from the PLL or a 16-MHz clock source. Figure 12-7 shows the ADC input equivalency diagram; for parameter values, see “Analog-to-Digital Converter (ADC)” on page 1207.

Figure 12-7. ADC Input Equivalency Diagram

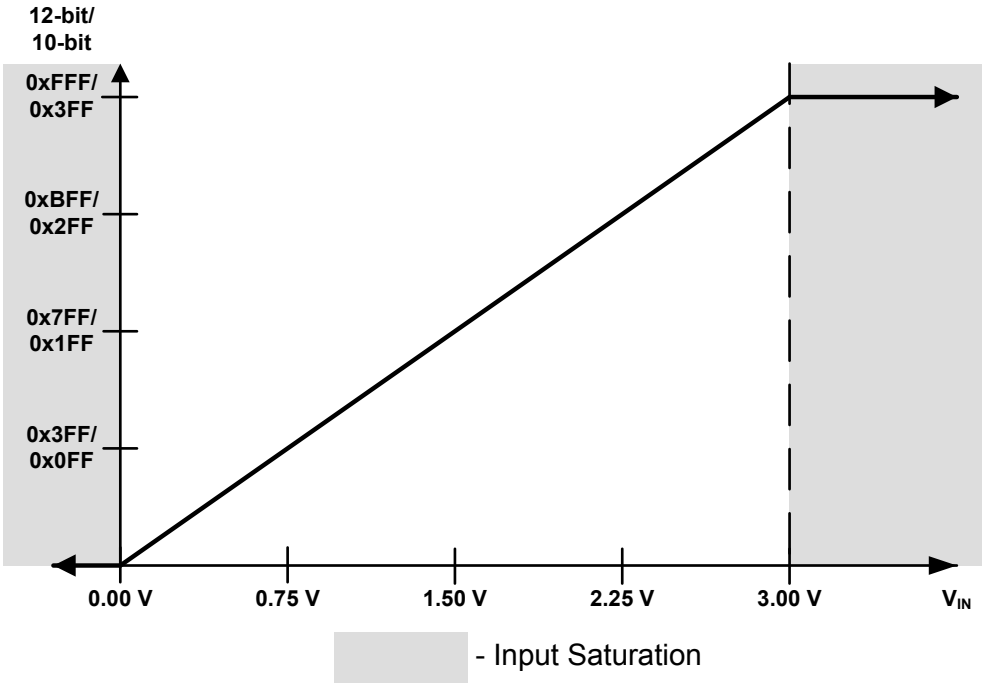


The ADC operates from both the 3.3-V analog and 1.2-V digital power supplies. The ADC clock can be configured to reduce power consumption when ADC conversions are not required (see "System Control" on page 199). The analog inputs are connected to the ADC through specially balanced input paths to minimize the distortion and cross-talk on the inputs. Detailed information on the ADC power supplies and analog inputs can be found in "Analog-to-Digital Converter (ADC)" on page 1207.

12.3.4.1 Internal Voltage Reference

The band-gap circuitry generates an internal 3.0 V reference that can be used by the ADC to produce a conversion value from the selected analog input. The range of this conversion value is from 0x000 to 0xFFF in 12-bit mode, or 0x3FF in 10-bit mode. In single-ended-input mode, the 0x000 value corresponds to an analog input voltage of 0.0 V; the 0xFFF in 12-bit mode, or 0x3FF in 10-bit mode value corresponds to an analog input voltage of 3.0 V. This configuration results in a resolution of approximately 0.7 mV in 12-bit mode and 2.9 mV per ADC code in 10-bit mode. While the analog input pads can handle voltages beyond this range, the ADC conversions saturate in under-voltage and over-voltage cases. Figure 12-8 on page 608 shows the ADC conversion function of the analog inputs.

Figure 12-8. Internal Voltage Conversion Result



12.3.4.2 External Voltage Reference

The ADC can use an external voltage reference to produce the conversion value from the selected analog input by configuring the VREF field in the **ADC Control (ADCCTL)** register. The VREF field specifies whether to use the internal, an external reference in the 3.0 V range, or an external reference in the 1.0 V range. While the range of the conversion value remains the same (0x000 to 0xFFF or 0x3FF), the analog voltage associated with the 0xFFF or 0x3FF value corresponds to the value of the voltage when using the 3.0-V setting and three times the voltage when using the 1.0-V setting, resulting in a smaller voltage resolution per ADC code. Ground is always used as the reference level for the minimum conversion value. Analog input voltages above the external voltage reference saturate to 0xFFF or 0x3FF while those below 0.0 V continue to saturate at 0x000. The V_{REFA} specification defines the useful range for the external voltage reference, see Table 24-23 on page 1208. Care must be taken to supply a reference voltage of acceptable quality.

Figure 12-9 on page 609 shows the ADC conversion function of the analog inputs when using anthe 3.0-V setting on the external voltage reference. Figure 12-10 on page 609 shows the ADC conversion function when using the 1.0-V setting on the external voltage reference.

The external voltage reference can be more accurate than the internal reference by using a high-precision source or trimming the source.

Figure 12-9. External Voltage Conversion Result with 3.0-V Setting

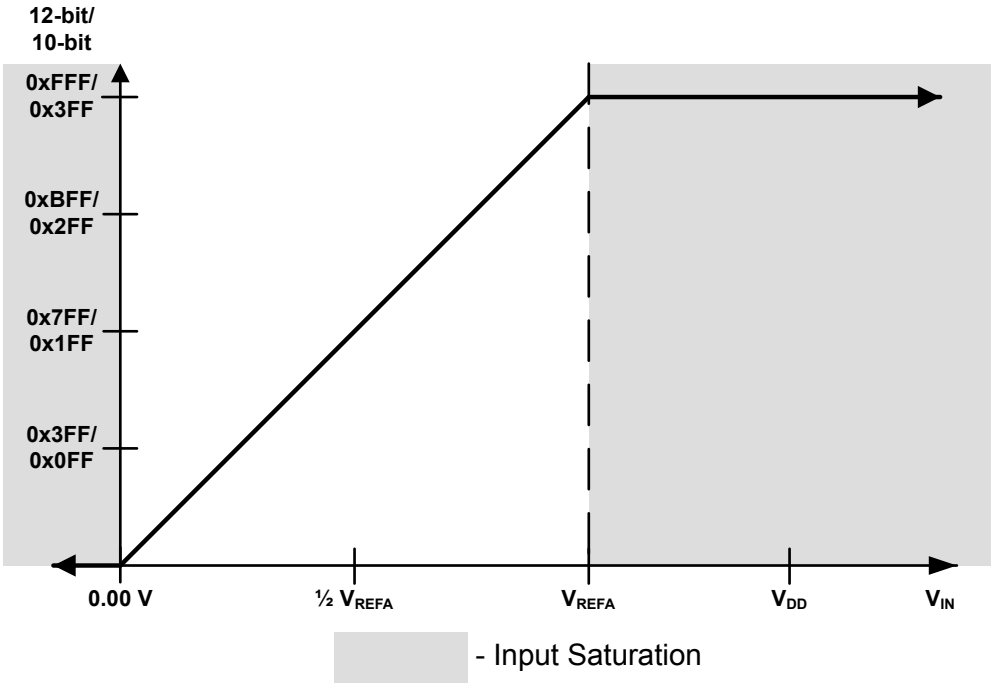
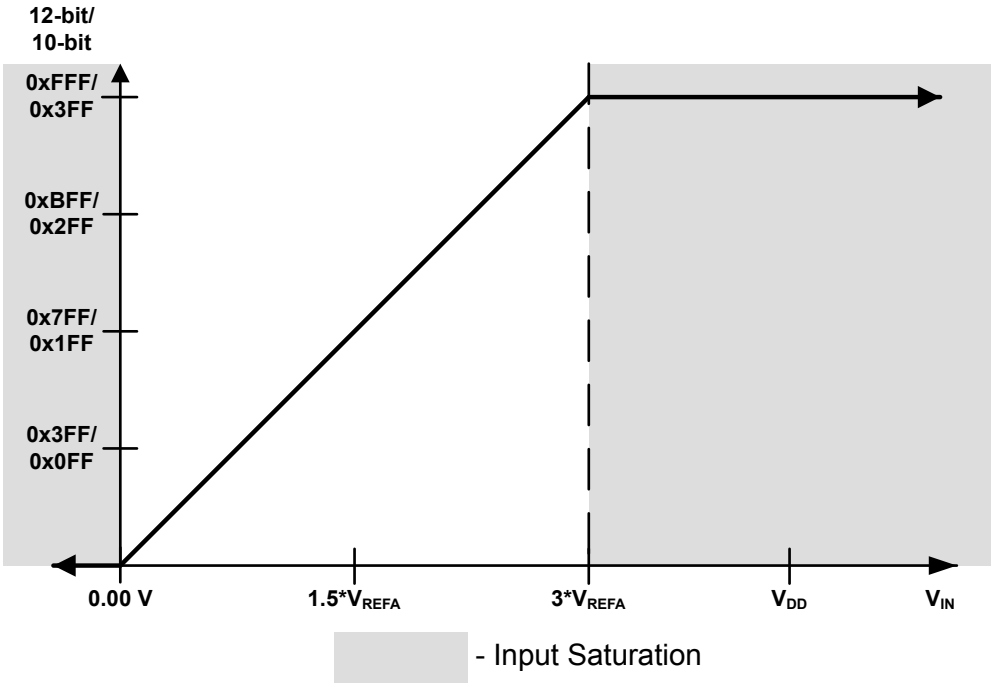


Figure 12-10. External Voltage Conversion Result with 1.0-V Setting



12.3.5 Differential Sampling

In addition to traditional single-ended sampling, the ADC module supports differential sampling of two analog input channels. To enable differential sampling, software must set the Dn bit in the **ADCSSCTL0n** register in a step's configuration nibble.

When a sequence step is configured for differential sampling, the input pair to sample must be configured in the **ADCSSMUXn** register. Differential pair 0 samples analog inputs 0 and 1; differential pair 1 samples analog inputs 2 and 3; and so on (see Table 12-4 on page 610). The ADC does not support other differential pairings such as analog input 0 with analog input 3.

Table 12-4. Differential Sampling Pairs

Differential Pair	Analog Inputs
0	0 and 1
1	2 and 3
2	4 and 5
3	6 and 7
4	8 and 9
5	10 and 11
6	12 and 13
7	14 and 15

The voltage sampled in differential mode is the difference between the odd and even channels:

ΔV (differential voltage) = V_{IN_EVEN} (even channel) – V_{IN_ODD} (odd channel), therefore:

- If $\Delta V = 0$, then the conversion result = 0x1FF for 10-bit and 0x7FF for 12-bit
- If $\Delta V > 0$, then the conversion result > 0x1FF (range is 0x1FF–0x3FF) for 10-bit and > 0x7FF (range is 0x7FF - 0xFFF) for 12-bit
- If $\Delta V < 0$, then the conversion result < 0x1FF (range is 0–0x1FF) for 10-bit and < 0x7FF (range is 0 - 0x7FF) for 12-bit

The differential pairs assign polarities to the analog inputs: the even-numbered input is always positive, and the odd-numbered input is always negative. In order for a valid conversion result to appear, the negative input must be in the range of ± 1.5 V of the positive input. If an analog input is greater than 3 V or less than 0 V (the valid range for analog inputs), the input voltage is clipped, meaning it appears as either 3 V or 0 V , respectively, to the ADC.

Figure 12-11 on page 611 shows an example of the negative input centered at 1.5 V. In this configuration, the differential range spans from -1.5 V to 1.5 V. Figure 12-12 on page 611 shows an example where the negative input is centered at 0.75 V, meaning inputs on the positive input saturate past a differential voltage of -0.75 V because the input voltage is less than 0 V. Figure 12-13 on page 612 shows an example of the negative input centered at 2.25 V, where inputs on the positive channel saturate past a differential voltage of 0.75 V since the input voltage would be greater than 3 V.

Figure 12-11. Differential Sampling Range, V_{IN_ODD} = 1.5 V

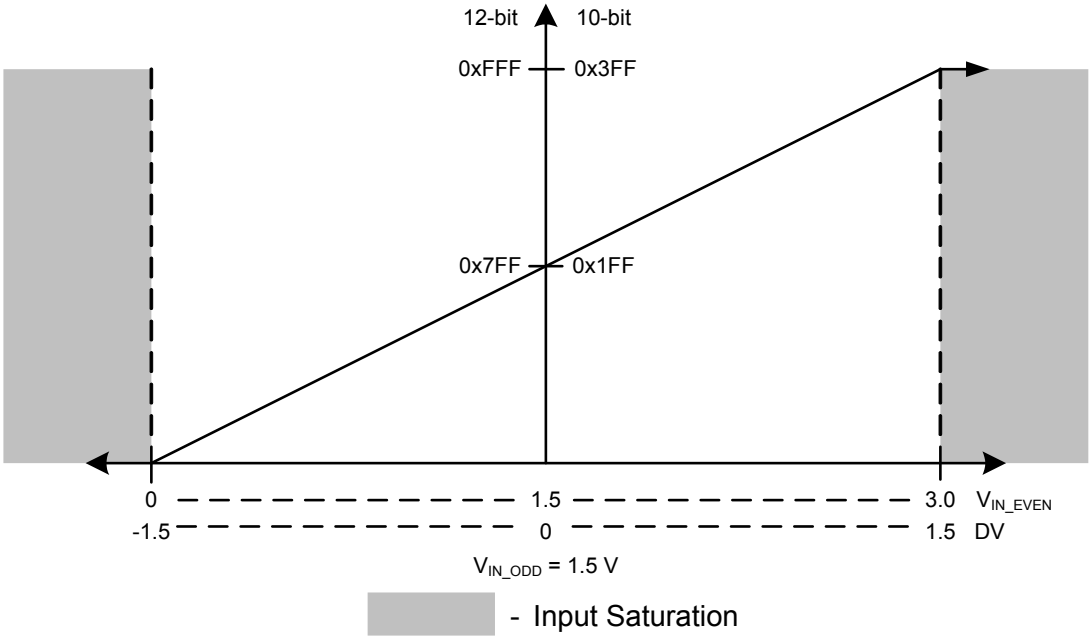


Figure 12-12. Differential Sampling Range, V_{IN_ODD} = 0.75 V

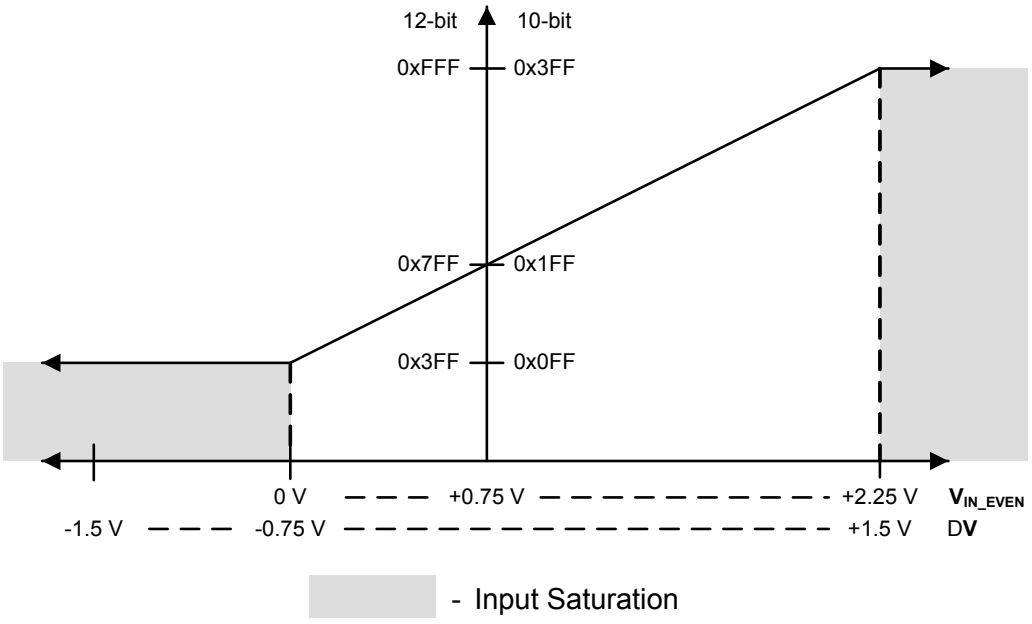
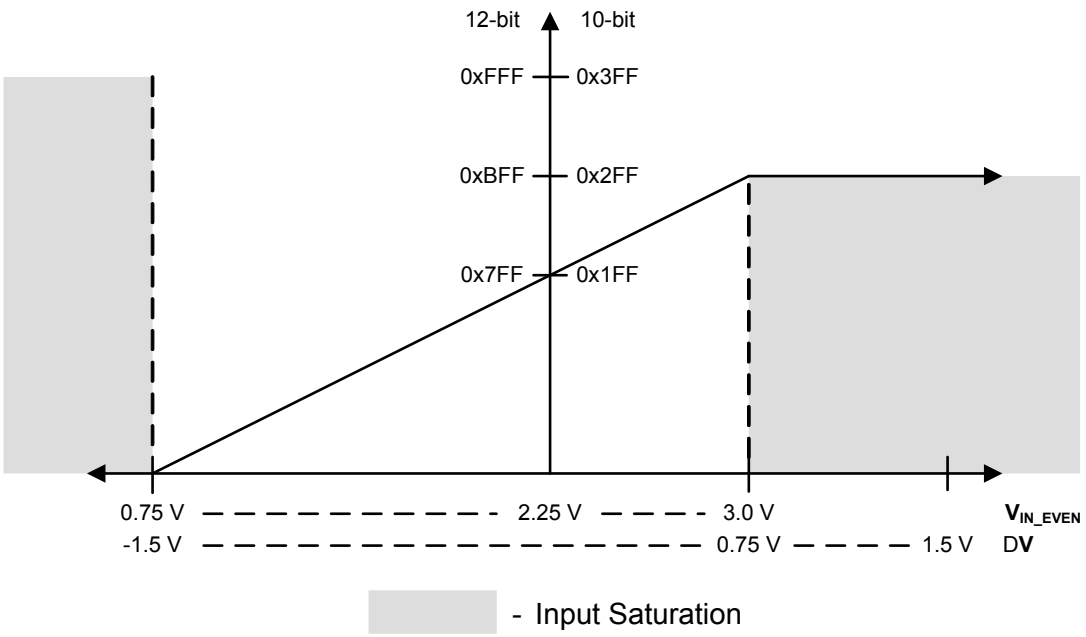


Figure 12-13. Differential Sampling Range, V_{IN_ODD} = 2.25 V



12.3.6 Internal Temperature Sensor

The temperature sensor's primary purpose is to notify the system that the internal temperature is too high or low for reliable operation.

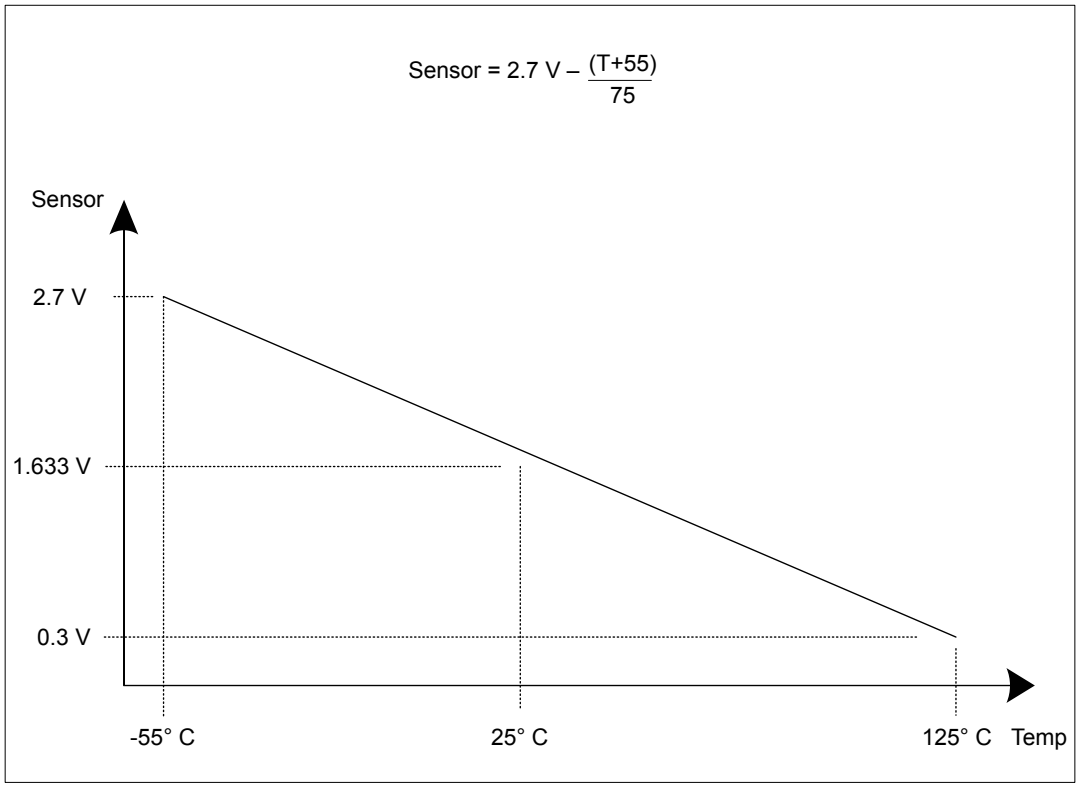
The temperature sensor does not have a separate enable, because it also contains the bandgap reference and must always be enabled. The reference is supplied to other analog modules; not just the ADC.

The internal temperature sensor provides an analog temperature reading as well as a reference voltage. This reference voltage, *SENSO*, is given by the following equation:

SENSO = 2.7 - ((T + 55) / 75)

This relation is shown in Figure 12-14 on page 613.

Figure 12-14. Internal Temperature Sensor Characteristic



The temperature sensor reading can be sampled in a sample sequence by setting the TSn bit in the **ADCSSCTLn** register. The temperature reading from the temperature sensor can also be given as a function of the ADC value. The following formula calculates temperature (in °C) based on the ADC reading:

```
Temperature = 147.5 - ((225 × ADC) / 4095)
```

12.3.7 Digital Comparator Unit

An ADC is commonly used to sample an external signal and to monitor its value to ensure that it remains in a given range. To automate this monitoring procedure and reduce the amount of processor overhead that is required, each module provides eight digital comparators. Conversions from the ADC that are sent to the digital comparators are compared against the user programmable limits in the **ADC Digital Comparator Range (ADCDCCMPn)** registers. If the observed signal moves out of the acceptable range, a processor interrupt can be generated. The digital comparators four operational modes (Once, Always, Hysteresis Once, Hysteresis Always) can be applied to three separate regions (low band, mid band, high band) as defined by the user.

12.3.7.1 Output Functions

ADC conversions can either be stored in the ADC Sample Sequence FIFOs or compared using the digital comparator resources as defined by the SnDC0P bits in the **ADC Sample Sequence n Operation (ADCSSOPn)** register. These selected ADC conversions are used by their respective digital comparator to monitor the external signal. Each comparator has two possible output functions: processor interrupts and triggers.

Each function has its own state machine to track the monitored signal. Even though the interrupt and trigger functions can be enabled individually or both at the same time, the same conversion data is used by each function to determine if the right conditions have been met to assert the associated output.

Interrupts

The digital comparator interrupt function is enabled by setting the CIE bit in the **ADC Digital Comparator Control (ADCDCCTLn)** register. This bit enables the interrupt function state machine to start monitoring the incoming ADC conversions. When the appropriate set of conditions is met, and the DCONSSx bit is set in the **ADCIM** register, an interrupt is sent to the interrupt controller.

12.3.7.2 Operational Modes

Four operational modes are provided to support a broad range of applications and multiple possible signaling requirements: Always, Once, Hysteresis Always, and Hysteresis Once. The operational mode is selected using the CIM field in the **ADCDCCTLn** register.

Always Mode

In the Always operational mode, the associated interrupt or trigger is asserted whenever the ADC conversion value meets its comparison criteria. The result is a string of assertions on the interrupt or trigger while the conversions are within the appropriate range.

Once Mode

In the Once operational mode, the associated interrupt or trigger is asserted whenever the ADC conversion value meets its comparison criteria, and the previous ADC conversion value did not. The result is a single assertion of the interrupt or trigger when the conversions are within the appropriate range.

Hysteresis-Always Mode

The Hysteresis-Always operational mode can only be used in conjunction with the low-band or high-band regions because the mid-band region must be crossed and the opposite region entered to clear the hysteresis condition. In the Hysteresis-Always mode, the associated interrupt or trigger is asserted in the following cases: 1) the ADC conversion value meets its comparison criteria or 2) a previous ADC conversion value has met the comparison criteria, and the hysteresis condition has not been cleared by entering the opposite region. The result is a string of assertions on the interrupt or trigger that continue until the opposite region is entered.

Hysteresis-Once Mode

The Hysteresis-Once operational mode can only be used in conjunction with the low-band or high-band regions because the mid-band region must be crossed and the opposite region entered to clear the hysteresis condition. In the Hysteresis-Once mode, the associated interrupt or trigger is asserted only when the ADC conversion value meets its comparison criteria, the hysteresis condition is clear, and the previous ADC conversion did not meet the comparison criteria. The result is a single assertion on the interrupt or trigger.

12.3.7.3 Function Ranges

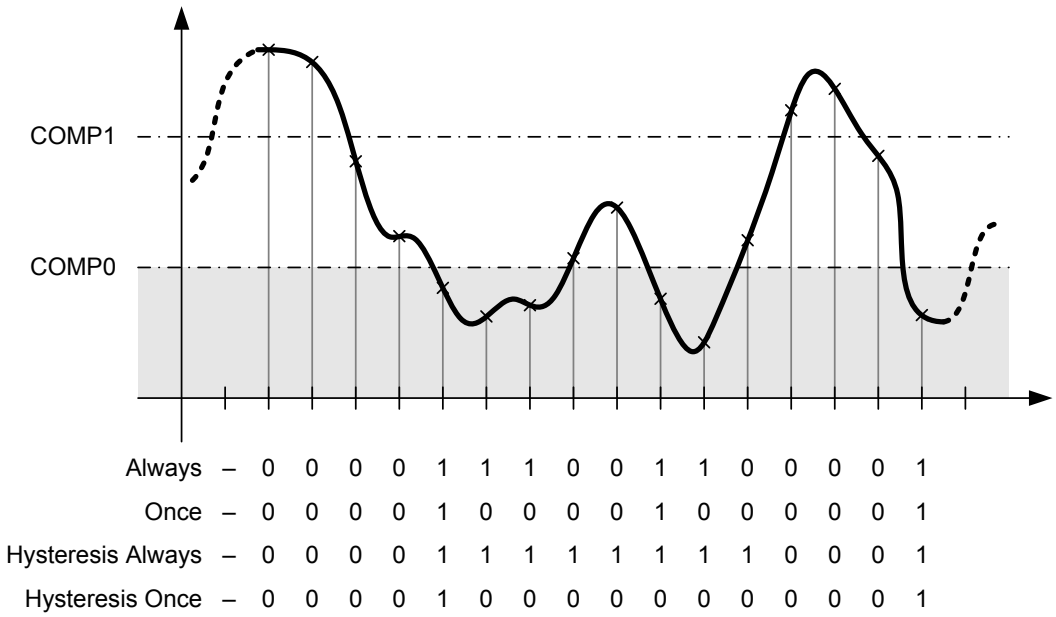
The two comparison values, COMP0 and COMP1, in the **ADC Digital Comparator Range (ADCDCCMPn)** register effectively break the conversion area into three distinct regions. These regions are referred to as the low-band (less than or equal to COMP0), mid-band (greater than COMP0 but less than or equal to COMP1), and high-band (greater than COMP1) regions. COMP0 and COMP1 may be programmed to the same value, effectively creating two regions, but COMP1 must always

be greater than or equal to the value of COMP0. A COMP1 value that is less than COMP0 generates unpredictable results.

Low-Band Operation

To operate in the low-band region, either the CIC field field in the **ADCDCCTLn** register must be programmed to 0x0. This setting causes interrupts or triggers to be generated in the low-band region as defined by the programmed operational mode. An example of the state of the interrupt/trigger signal in the low-band region for each of the operational modes is shown in Figure 12-15 on page 615. Note that a "0" in a column following the operational mode name (Always, Once, Hysteresis Always, and Hysteresis Once) indicates that the interrupt or trigger signal is de-asserted and a "1" indicates that the signal is asserted.

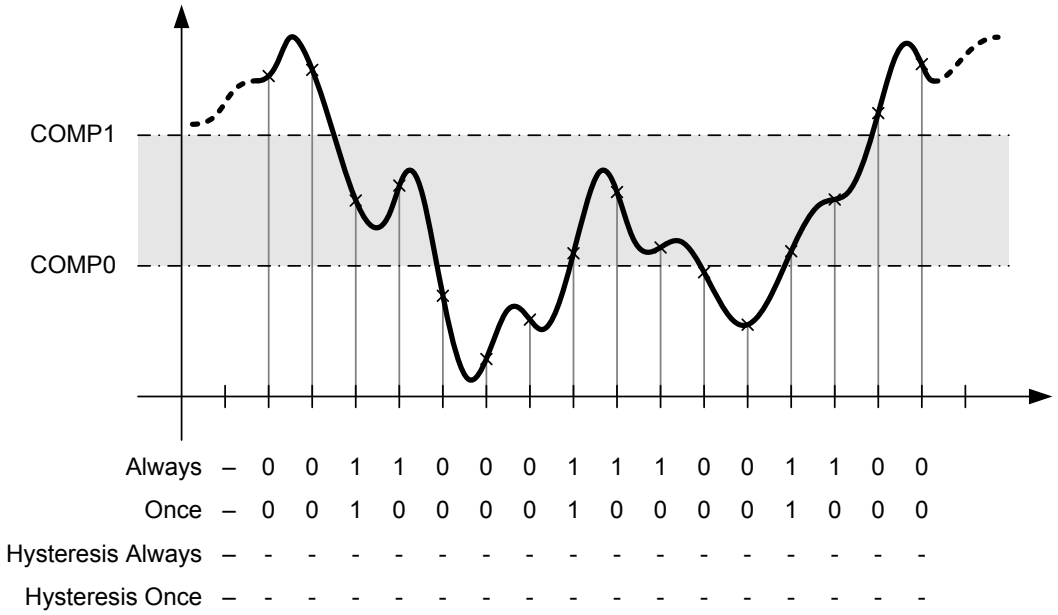
Figure 12-15. Low-Band Operation (CIC=0x0)



Mid-Band Operation

To operate in the mid-band region, either the CIC field field in the **ADCDCCTLn** register must be programmed to 0x1. This setting causes interrupts or triggers to be generated in the mid-band region according the operation mode. Only the Always and Once operational modes are available in the mid-band region. An example of the state of the interrupt/trigger signal in the mid-band region for each of the allowed operational modes is shown in Figure 12-16 on page 616. Note that a "0" in a column following the operational mode name (Always or Once) indicates that the interrupt or trigger signal is de-asserted and a "1" indicates that the signal is asserted.

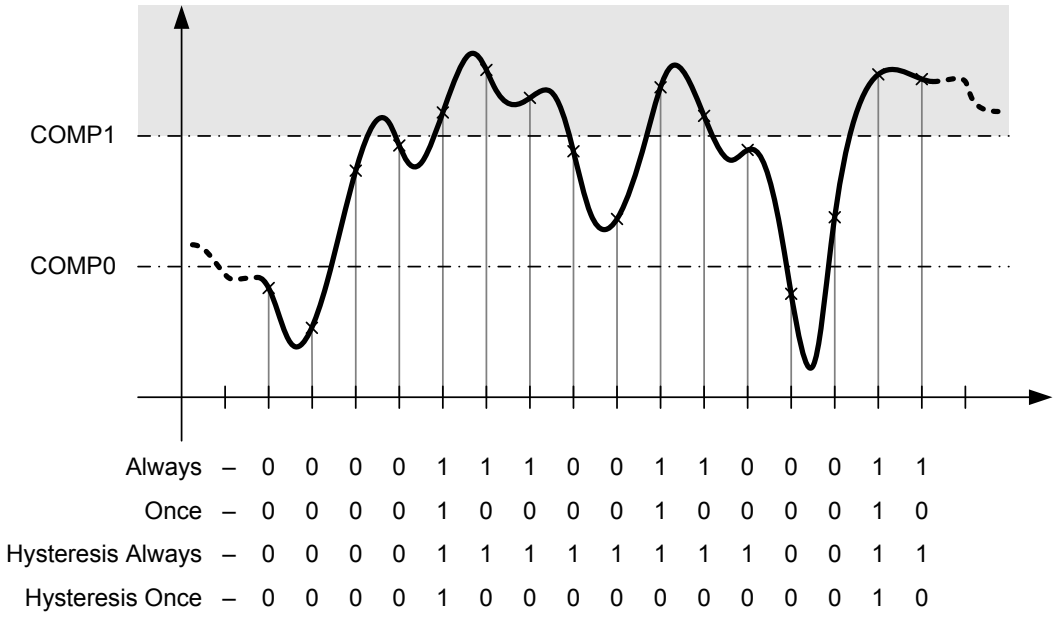
Figure 12-16. Mid-Band Operation (CIC=0x1)



High-Band Operation

To operate in the high-band region, either the `CIC` field field in the **ADCDCCTLn** register must be programmed to 0x3. This setting causes interrupts or triggers to be generated in the high-band region according the operation mode. An example of the state of the interrupt/trigger signal in the high-band region for each of the allowed operational modes is shown in Figure 12-17 on page 617. Note that a "0" in a column following the operational mode name (Always, Once, Hysteresis Always, and Hysteresis Once) indicates that the interrupt or trigger signal is de-asserted and a "1" indicates that the signal is asserted.

Figure 12-17. High-Band Operation (CIC=0x3)



12.4 Initialization and Configuration

In order for the ADC module to be used, the PLL must be enabled and programmed to a supported crystal frequency in the **RCC** register (see page 215). Using unsupported frequencies can cause faulty operation in the ADC module.

12.4.1 Module Initialization

Initialization of the ADC module is a simple process with very few steps: enabling the clock to the ADC, disabling the analog isolation circuit associated with all inputs that are to be used, and reconfiguring the sample sequencer priorities (if needed).

The initialization sequence for the ADC is as follows:

1. Enable the ADC clock by using the **RCGC0** register (see page 255).
2. Enable the clock to the appropriate GPIO modules via the **RCGC2** register (see page 272). To find out which GPIO ports to enable, refer to “Signal Description” on page 600.
3. Set the GPIO AFSEL bits for the ADC input pins (see page 419). To determine which GPIOs to configure, see Table 22-4 on page 1144.
4. Configure the AINx and VREFA signals to be analog inputs by clearing the corresponding DEN bit in the **GPIO Digital Enable (GPIODEN)** register (see page 430).
5. Disable the analog isolation circuit for all ADC input pins that are to be used by writing a 1 to the appropriate bits of the **GPIOAMSEL** register (see page 435) in the associated GPIO block.

6. If required by the application, reconfigure the sample sequencer priorities in the **ADCSSPRI** register. The default configuration has Sample Sequencer 0 with the highest priority and Sample Sequencer 3 as the lowest priority.

12.4.2 Sample Sequencer Configuration

Configuration of the sample sequencers is slightly more complex than the module initialization because each sample sequencer is completely programmable.

The configuration for each sample sequencer should be as follows:

1. Ensure that the sample sequencer is disabled by clearing the corresponding ASENn bit in the **ADCACTSS** register. Programming of the sample sequencers is allowed without having them enabled. Disabling the sequencer during programming prevents erroneous execution if a trigger event were to occur during the configuration process.
2. Configure the trigger event for the sample sequencer in the **ADCEMUX** register.
3. For each sample in the sample sequence, configure the corresponding input source in the **ADCSSMUXn** register.
4. For each sample in the sample sequence, configure the sample control bits in the corresponding nibble in the **ADCSSCTLn** register. When programming the last nibble, ensure that the END bit is set. Failure to set the END bit causes unpredictable behavior.
5. If interrupts are to be used, set the corresponding MASK bit in the **ADCIM** register.
6. Enable the sample sequencer logic by setting the corresponding ASENn bit in the **ADCACTSS** register.

12.5 Register Map

Table 12-5 on page 618 lists the ADC registers. The offset listed is a hexadecimal increment to the register's address, relative to that ADC module's base address of:

- ADC0: 0x4003.8000
- ADC1: 0x4003.9000

Note that the ADC module clock must be enabled before the registers can be programmed (see page 255). There must be a delay of 3 system clocks after the ADC module clock is enabled before any ADC module registers are accessed.

Table 12-5. ADC Register Map

Offset	Name	Type	Reset	Description	See page
0x000	ADCACTSS	R/W	0x0000.0000	ADC Active Sample Sequencer	621
0x004	ADCRIS	RO	0x0000.0000	ADC Raw Interrupt Status	622
0x008	ADCIM	R/W	0x0000.0000	ADC Interrupt Mask	624
0x00C	ADCISC	R/W1C	0x0000.0000	ADC Interrupt Status and Clear	626
0x010	ADCOSTAT	R/W1C	0x0000.0000	ADC Overflow Status	629
0x014	ADCEMUX	R/W	0x0000.0000	ADC Event Multiplexer Select	631

Table 12-5. ADC Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x018	ADCUSTAT	R/W1C	0x0000.0000	ADC Underflow Status	636
0x020	ADCSSPRI	R/W	0x0000.3210	ADC Sample Sequencer Priority	637
0x024	ADCSPC	R/W	0x0000.0000	ADC Sample Phase Control	639
0x028	ADCPSSI	R/W	-	ADC Processor Sample Sequence Initiate	641
0x030	ADCSAC	R/W	0x0000.0000	ADC Sample Averaging Control	643
0x034	ADCDCISC	R/W1C	0x0000.0000	ADC Digital Comparator Interrupt Status and Clear	644
0x038	ADCCTL	R/W	0x0000.0000	ADC Control	646
0x040	ADCSSMUX0	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 0	647
0x044	ADCSSCTL0	R/W	0x0000.0000	ADC Sample Sequence Control 0	649
0x048	ADCSSFIFO0	RO	-	ADC Sample Sequence Result FIFO 0	652
0x04C	ADCSSFSTAT0	RO	0x0000.0100	ADC Sample Sequence FIFO 0 Status	653
0x050	ADCSSOP0	R/W	0x0000.0000	ADC Sample Sequence 0 Operation	655
0x054	ADCSSDC0	R/W	0x0000.0000	ADC Sample Sequence 0 Digital Comparator Select	657
0x060	ADCSSMUX1	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 1	659
0x064	ADCSSCTL1	R/W	0x0000.0000	ADC Sample Sequence Control 1	660
0x068	ADCSSFIFO1	RO	-	ADC Sample Sequence Result FIFO 1	652
0x06C	ADCSSFSTAT1	RO	0x0000.0100	ADC Sample Sequence FIFO 1 Status	653
0x070	ADCSSOP1	R/W	0x0000.0000	ADC Sample Sequence 1 Operation	662
0x074	ADCSSDC1	R/W	0x0000.0000	ADC Sample Sequence 1 Digital Comparator Select	663
0x080	ADCSSMUX2	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 2	659
0x084	ADCSSCTL2	R/W	0x0000.0000	ADC Sample Sequence Control 2	660
0x088	ADCSSFIFO2	RO	-	ADC Sample Sequence Result FIFO 2	652
0x08C	ADCSSFSTAT2	RO	0x0000.0100	ADC Sample Sequence FIFO 2 Status	653
0x090	ADCSSOP2	R/W	0x0000.0000	ADC Sample Sequence 2 Operation	662
0x094	ADCSSDC2	R/W	0x0000.0000	ADC Sample Sequence 2 Digital Comparator Select	663
0x0A0	ADCSSMUX3	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 3	665
0x0A4	ADCSSCTL3	R/W	0x0000.0002	ADC Sample Sequence Control 3	666
0x0A8	ADCSSFIFO3	RO	-	ADC Sample Sequence Result FIFO 3	652
0x0AC	ADCSSFSTAT3	RO	0x0000.0100	ADC Sample Sequence FIFO 3 Status	653
0x0B0	ADCSSOP3	R/W	0x0000.0000	ADC Sample Sequence 3 Operation	667
0x0B4	ADCSSDC3	R/W	0x0000.0000	ADC Sample Sequence 3 Digital Comparator Select	668
0xD00	ADCDCRIC	R/W	0x0000.0000	ADC Digital Comparator Reset Initial Conditions	669

Table 12-5. ADC Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0xE00	ADCDCCTL0	R/W	0x0000.0000	ADC Digital Comparator Control 0	674
0xE04	ADCDCCTL1	R/W	0x0000.0000	ADC Digital Comparator Control 1	674
0xE08	ADCDCCTL2	R/W	0x0000.0000	ADC Digital Comparator Control 2	674
0xE0C	ADCDCCTL3	R/W	0x0000.0000	ADC Digital Comparator Control 3	674
0xE10	ADCDCCTL4	R/W	0x0000.0000	ADC Digital Comparator Control 4	674
0xE14	ADCDCCTL5	R/W	0x0000.0000	ADC Digital Comparator Control 5	674
0xE18	ADCDCCTL6	R/W	0x0000.0000	ADC Digital Comparator Control 6	674
0xE1C	ADCDCCTL7	R/W	0x0000.0000	ADC Digital Comparator Control 7	674
0xE40	ADCDCCMP0	R/W	0x0000.0000	ADC Digital Comparator Range 0	676
0xE44	ADCDCCMP1	R/W	0x0000.0000	ADC Digital Comparator Range 1	676
0xE48	ADCDCCMP2	R/W	0x0000.0000	ADC Digital Comparator Range 2	676
0xE4C	ADCDCCMP3	R/W	0x0000.0000	ADC Digital Comparator Range 3	676
0xE50	ADCDCCMP4	R/W	0x0000.0000	ADC Digital Comparator Range 4	676
0xE54	ADCDCCMP5	R/W	0x0000.0000	ADC Digital Comparator Range 5	676
0xE58	ADCDCCMP6	R/W	0x0000.0000	ADC Digital Comparator Range 6	676
0xE5C	ADCDCCMP7	R/W	0x0000.0000	ADC Digital Comparator Range 7	676

12.6 Register Descriptions

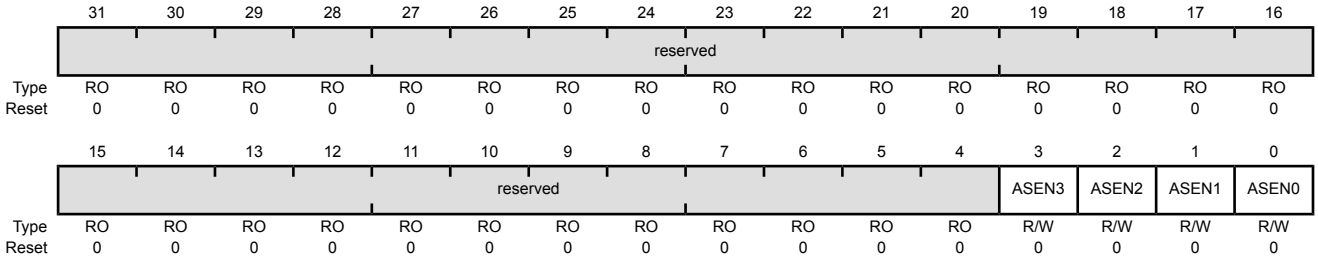
The remainder of this section lists and describes the ADC registers, in numerical order by address offset.

Register 1: ADC Active Sample Sequencer (ADCACTSS), offset 0x000

This register controls the activation of the sample sequencers. Each sample sequencer can be enabled or disabled independently.

ADC Active Sample Sequencer (ADCACTSS)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x000
Type R/W, reset 0x0000.0000



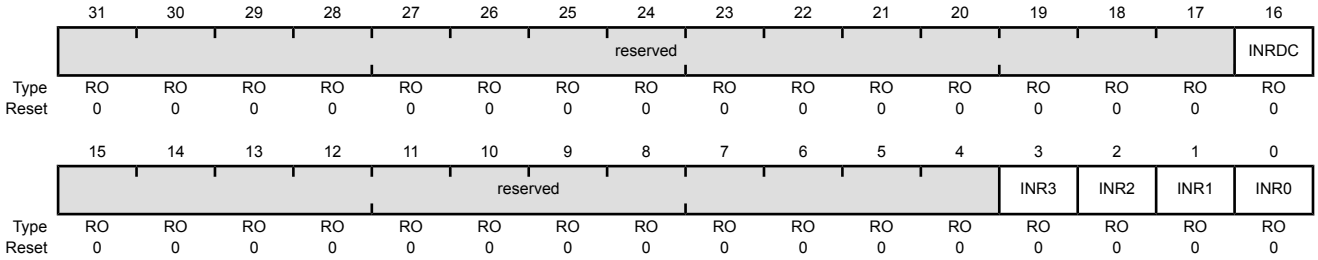
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	ASEN3	R/W	0	ADC SS3 Enable Value Description 1 Sample Sequencer 3 is enabled. 0 Sample Sequencer 3 is disabled.
2	ASEN2	R/W	0	ADC SS2 Enable Value Description 1 Sample Sequencer 2 is enabled. 0 Sample Sequencer 2 is disabled.
1	ASEN1	R/W	0	ADC SS1 Enable Value Description 1 Sample Sequencer 1 is enabled. 0 Sample Sequencer 1 is disabled.
0	ASEN0	R/W	0	ADC SS0 Enable Value Description 1 Sample Sequencer 0 is enabled. 0 Sample Sequencer 0 is disabled.

Register 2: ADC Raw Interrupt Status (ADCRIS), offset 0x004

This register shows the status of the raw interrupt signal of each sample sequencer. These bits may be polled by software to look for interrupt conditions without sending the interrupts to the interrupt controller.

ADC Raw Interrupt Status (ADCRIS)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x004
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	INRDC	RO	0	Digital Comparator Raw Interrupt Status Value Description 1 At least one bit in the **ADCDCISC** register is set, meaning that a digital comparator interrupt has occurred. 0 All bits in the **ADCDCISC** register are clear.
15:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	INR3	RO	0	SS3 Raw Interrupt Status Value Description 1 A sample has completed conversion and the respective **ADCSSCTL3** `IEn` bit is set, enabling a raw interrupt. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the `IN3` bit in the **ADCISC** register.
2	INR2	RO	0	SS2 Raw Interrupt Status Value Description 1 A sample has completed conversion and the respective **ADCSSCTL2** `IEn` bit is set, enabling a raw interrupt. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the `IN2` bit in the **ADCISC** register.

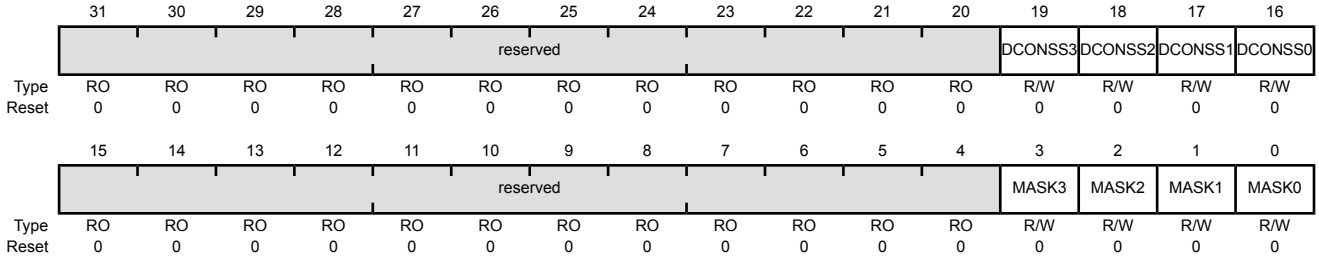
Bit/Field	Name	Type	Reset	Description
1	INR1	RO	0	SS1 Raw Interrupt Status Value Description 1 A sample has completed conversion and the respective **ADCSSCTL1** IEn bit is set, enabling a raw interrupt. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the IN1 bit in the **ADCISC** register.
0	INR0	RO	0	SS0 Raw Interrupt Status Value Description 1 A sample has completed conversion and the respective **ADCSSCTL0** IEn bit is set, enabling a raw interrupt. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the IN0 bit in the **ADCISC** register.

Register 3: ADC Interrupt Mask (ADCIM), offset 0x008

This register controls whether the sample sequencer and digital comparator raw interrupt signals are sent to the interrupt controller. Each raw interrupt signal can be masked independently. Only a single DCONSSn bit should be set at any given time. Setting more than one of these bits results in the INRDC bit from the **ADCRIS** register being masked, and no interrupt is generated on any of the sample sequencer interrupt lines.

ADC Interrupt Mask (ADCIM)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x008
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:20	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	DCONSS3	R/W	0	Digital Comparator Interrupt on SS3 Value Description 1 The raw interrupt signal from the digital comparators (INRDC bit in the **ADCRIS** register) is sent to the interrupt controller on the SS3 interrupt line. 0 The status of the digital comparators does not affect the SS3 interrupt status.
18	DCONSS2	R/W	0	Digital Comparator Interrupt on SS2 Value Description 1 The raw interrupt signal from the digital comparators (INRDC bit in the **ADCRIS** register) is sent to the interrupt controller on the SS2 interrupt line. 0 The status of the digital comparators does not affect the SS2 interrupt status.
17	DCONSS1	R/W	0	Digital Comparator Interrupt on SS1 Value Description 1 The raw interrupt signal from the digital comparators (INRDC bit in the **ADCRIS** register) is sent to the interrupt controller on the SS1 interrupt line. 0 The status of the digital comparators does not affect the SS1 interrupt status.

Bit/Field	Name	Type	Reset	Description
16	DCONSS0	R/W	0	Digital Comparator Interrupt on SS0 Value Description 1 The raw interrupt signal from the digital comparators (INRDC bit in the **ADCRIS** register) is sent to the interrupt controller on the SS0 interrupt line. 0 The status of the digital comparators does not affect the SS0 interrupt status.
15:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	MASK3	R/W	0	SS3 Interrupt Mask Value Description 1 The raw interrupt signal from Sample Sequencer 3 (**ADCRIS** register INR3 bit) is sent to the interrupt controller. 0 The status of Sample Sequencer 3 does not affect the SS3 interrupt status.
2	MASK2	R/W	0	SS2 Interrupt Mask Value Description 1 The raw interrupt signal from Sample Sequencer 2 (**ADCRIS** register INR2 bit) is sent to the interrupt controller. 0 The status of Sample Sequencer 2 does not affect the SS2 interrupt status.
1	MASK1	R/W	0	SS1 Interrupt Mask Value Description 1 The raw interrupt signal from Sample Sequencer 1 (**ADCRIS** register INR1 bit) is sent to the interrupt controller. 0 The status of Sample Sequencer 1 does not affect the SS1 interrupt status.
0	MASK0	R/W	0	SS0 Interrupt Mask Value Description 1 The raw interrupt signal from Sample Sequencer 0 (**ADCRIS** register INR0 bit) is sent to the interrupt controller. 0 The status of Sample Sequencer 0 does not affect the SS0 interrupt status.

Register 4: ADC Interrupt Status and Clear (ADCISC), offset 0x00C

This register provides the mechanism for clearing sample sequencer interrupt conditions and shows the status of interrupts generated by the sample sequencers and the digital comparators which have been sent to the interrupt controller. When read, each bit field is the logical AND of the respective INR and MASK bits. Sample sequencer interrupts are cleared by writing a 1 to the corresponding bit position. Digital comparator interrupts are cleared by writing a 1 to the appropriate bits in the **ADCDCISC** register. If software is polling the **ADCRIS** instead of generating interrupts, the sample sequence INRn bits are still cleared via the **ADCISC** register, even if the INn bit is not set.

ADC Interrupt Status and Clear (ADCISC)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x00C
Type R/W1C, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved												DCINSS3	DCINSS2	DCINSS1	DCINSS0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved												IN3	IN2	IN1	IN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:20	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	DCINSS3	RO	0	Digital Comparator Interrupt Status on SS3 Value Description 1 Both the INRDC bit in the **ADCRIS** register and the DCONSS3 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the **ADCRIS** register.
18	DCINSS2	RO	0	Digital Comparator Interrupt Status on SS2 Value Description 1 Both the INRDC bit in the **ADCRIS** register and the DCONSS2 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the **ADCRIS** register.

Bit/Field	Name	Type	Reset	Description
17	DCINSS1	RO	0	Digital Comparator Interrupt Status on SS1 Value Description 1 Both the INRDC bit in the **ADCRIS** register and the DCONSS1 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the **ADCRIS** register.
16	DCINSS0	RO	0	Digital Comparator Interrupt Status on SS0 Value Description 1 Both the INRDC bit in the **ADCRIS** register and the DCONSS0 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the **ADCRIS** register.
15:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	IN3	R/W1C	0	SS3 Interrupt Status and Clear Value Description 1 Both the INR3 bit in the **ADCRIS** register and the MASK3 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the INR3 bit in the **ADCRIS** register.
2	IN2	R/W1C	0	SS2 Interrupt Status and Clear Value Description 1 Both the INR2 bit in the **ADCRIS** register and the MASK2 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the INR2 bit in the **ADCRIS** register.

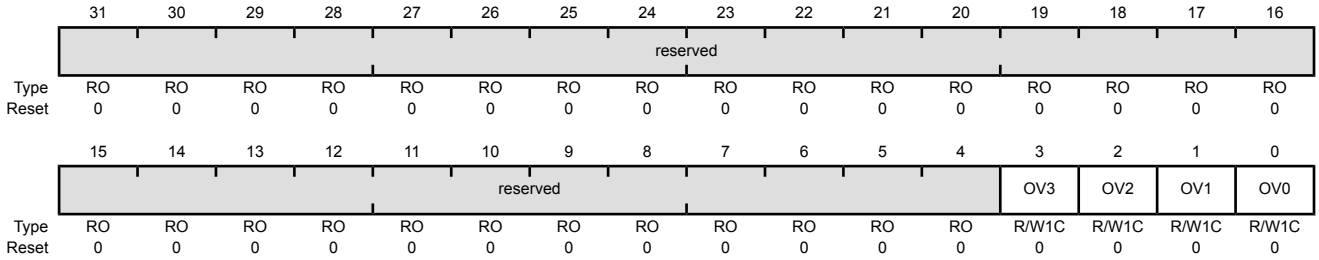
Bit/Field	Name	Type	Reset	Description
1	IN1	R/W1C	0	SS1 Interrupt Status and Clear Value Description 1 Both the INR1 bit in the **ADCRIS** register and the MASK1 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the INR1 bit in the **ADCRIS** register.
0	IN0	R/W1C	0	SS0 Interrupt Status and Clear Value Description 1 Both the INR0 bit in the **ADCRIS** register and the MASK0 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the INR0 bit in the **ADCRIS** register.

Register 5: ADC Overflow Status (ADCOSTAT), offset 0x010

This register indicates overflow conditions in the sample sequencer FIFOs. Once the overflow condition has been handled by software, the condition can be cleared by writing a 1 to the corresponding bit position.

ADC Overflow Status (ADCOSTAT)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x010
Type R/W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	OV3	R/W1C	0	SS3 FIFO Overflow Value Description 1 The FIFO for Sample Sequencer 3 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. 0 The FIFO has not overflowed. This bit is cleared by writing a 1.
2	OV2	R/W1C	0	SS2 FIFO Overflow Value Description 1 The FIFO for Sample Sequencer 2 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. 0 The FIFO has not overflowed. This bit is cleared by writing a 1.
1	OV1	R/W1C	0	SS1 FIFO Overflow Value Description 1 The FIFO for Sample Sequencer 1 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. 0 The FIFO has not overflowed. This bit is cleared by writing a 1.

Bit/Field	Name	Type	Reset	Description
0	OV0	R/W1C	0	SS0 FIFO Overflow

Value	Description
1	The FIFO for Sample Sequencer 0 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.
0	The FIFO has not overflowed.

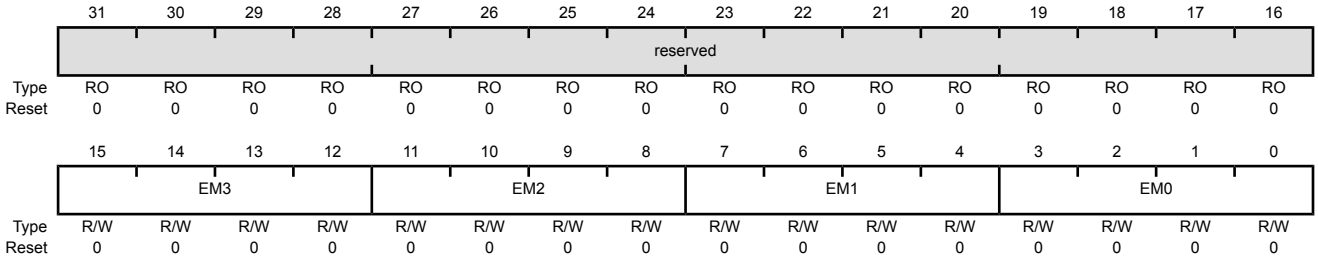
This bit is cleared by writing a 1.

Register 6: ADC Event Multiplexer Select (ADCEMUX), offset 0x014

The **ADCEMUX** selects the event (trigger) that initiates sampling for each sample sequencer. Each sample sequencer can be configured with a unique trigger source.

ADC Event Multiplexer Select (ADCEMUX)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x014
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
15:12	EM3	R/W	0x0	SS3 Trigger Select This field selects the trigger source for Sample Sequencer 3. The valid configurations for this field are:

Value	Event
0x0	Processor (default) The trigger is initiated by setting the SSn bit in the **ADCPSSI** register.
0x1	Analog Comparator 0 This trigger is configured by the **Analog Comparator Control 0 (ACCTL0)** register (page 1120).
0x2	Analog Comparator 1 This trigger is configured by the **Analog Comparator Control 1 (ACCTL1)** register (page 1120).
0x3	Analog Comparator 2 This trigger is configured by the **Analog Comparator Control 2 (ACCTL2)** register (page 1120).
0x4	External (GPIO PB4) This trigger is connected to the GPIO interrupt for PB4 (see "ADC Trigger Source" on page 403). **Note:** PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input.
0x5	Timer In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (page 548).
0x6	reserved
0x7	reserved
0x8	reserved
0x9	reserved
0xA-0xE	reserved
0xF	Always (continuously sample)

Bit/Field	Name	Type	Reset	Description
11:8	EM2	R/W	0x0	SS2 Trigger Select This field selects the trigger source for Sample Sequencer 2. The valid configurations for this field are:

Value	Event
0x0	Processor (default) The trigger is initiated by setting the SSn bit in the **ADCPSSI** register.
0x1	Analog Comparator 0 This trigger is configured by the **Analog Comparator Control 0 (ACCTL0)** register (page 1120).
0x2	Analog Comparator 1 This trigger is configured by the **Analog Comparator Control 1 (ACCTL1)** register (page 1120).
0x3	Analog Comparator 2 This trigger is configured by the **Analog Comparator Control 2 (ACCTL2)** register (page 1120).
0x4	External (GPIO PB4) This trigger is connected to the GPIO interrupt for PB4 (see "ADC Trigger Source" on page 403). **Note:** PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input.
0x5	Timer In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (page 548).
0x6	reserved
0x7	reserved
0x8	reserved
0x9	reserved
0xA-0xE	reserved
0xF	Always (continuously sample)

Bit/Field	Name	Type	Reset	Description
7:4	EM1	R/W	0x0	SS1 Trigger Select This field selects the trigger source for Sample Sequencer 1. The valid configurations for this field are:

Value	Event
0x0	Processor (default) The trigger is initiated by setting the SSn bit in the **ADCPSSI** register.
0x1	Analog Comparator 0 This trigger is configured by the **Analog Comparator Control 0 (ACCTL0)** register (page 1120).
0x2	Analog Comparator 1 This trigger is configured by the **Analog Comparator Control 1 (ACCTL1)** register (page 1120).
0x3	Analog Comparator 2 This trigger is configured by the **Analog Comparator Control 2 (ACCTL2)** register (page 1120).
0x4	External (GPIO PB4) This trigger is connected to the GPIO interrupt for PB4 (see "ADC Trigger Source" on page 403). **Note:** PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input.
0x5	Timer In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (page 548).
0x6	reserved
0x7	reserved
0x8	reserved
0x9	reserved
0xA-0xE	reserved
0xF	Always (continuously sample)

Bit/Field	Name	Type	Reset	Description
3:0	EM0	R/W	0x0	SS0 Trigger Select This field selects the trigger source for Sample Sequencer 0 The valid configurations for this field are:

Value	Event
0x0	Processor (default) The trigger is initiated by setting the SSn bit in the **ADCPSSI** register.
0x1	Analog Comparator 0 This trigger is configured by the **Analog Comparator Control 0 (ACCTL0)** register (page 1120).
0x2	Analog Comparator 1 This trigger is configured by the **Analog Comparator Control 1 (ACCTL1)** register (page 1120).
0x3	Analog Comparator 2 This trigger is configured by the **Analog Comparator Control 2 (ACCTL2)** register (page 1120).
0x4	External (GPIO PB4) This trigger is connected to the GPIO interrupt for PB4 (see "ADC Trigger Source" on page 403). **Note:** PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input.
0x5	Timer In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (page 548).
0x6	reserved
0x7	reserved
0x8	reserved
0x9	reserved
0xA-0xE	reserved
0xF	Always (continuously sample)

Register 7: ADC Underflow Status (ADCUSTAT), offset 0x018

This register indicates underflow conditions in the sample sequencer FIFOs. The corresponding underflow condition is cleared by writing a 1 to the relevant bit position.

ADC Underflow Status (ADCUSTAT)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x018
Type R/W1C, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												UV3	UV2	UV1	UV0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	UV3	R/W1C	0	SS3 FIFO Underflow The valid configurations for this field are shown below. This bit is cleared by writing a 1. Value Description 1 The FIFO for the Sample Sequencer has hit an underflow condition, meaning that the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. 0 The FIFO has not underflowed.
2	UV2	R/W1C	0	SS2 FIFO Underflow The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1.
1	UV1	R/W1C	0	SS1 FIFO Underflow The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1.
0	UV0	R/W1C	0	SS0 FIFO Underflow The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1.

Register 8: ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020

This register sets the priority for each of the sample sequencers. Out of reset, Sequencer 0 has the highest priority, and Sequencer 3 has the lowest priority. When reconfiguring sequence priorities, each sequence must have a unique priority for the ADC to operate properly.

ADC Sample Sequencer Priority (ADCSSPRI)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x020
Type R/W, reset 0x0000.3210

Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13:12	SS3	R/W	0x3	SS3 Priority This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 3. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.
11:10	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:8	SS2	R/W	0x2	SS2 Priority This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 2. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.
7:6	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:4	SS1	R/W	0x1	SS1 Priority This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 1. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.
3:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
1:0	SS0	R/W	0x0	SS0 Priority This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 0. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.

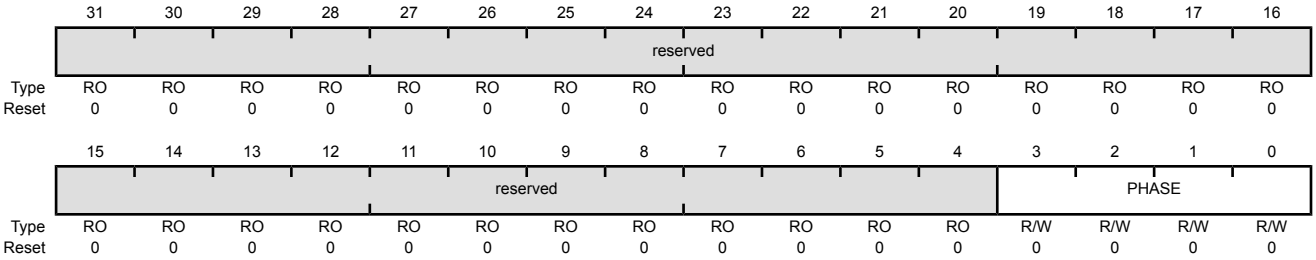
Register 9: ADC Sample Phase Control (ADCSPC), offset 0x024

This register allows the ADC module to sample at one of 16 different discrete phases from 0.0° through 337.5°. For example, the sample rate could be effectively doubled by sampling a signal using one ADC module configured with the standard sample time and the second ADC module configured with a 180.0° phase lag.

Note: Care should be taken when the PHASE field is non-zero, as the resulting delay in sampling the AINx input may result in undesirable system consequences. The time from ADC trigger to sample is increased and could make the response time longer than anticipated. The added latency could have ramifications in the system design. Designers should carefully consider the impact of this delay.

ADC Sample Phase Control (ADCSPC)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x024
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
3:0	PHASE	R/W	0x0	Phase Difference This field selects the sample phase difference from the standard sample time.

Value	Description
0x0	ADC sample lags by 0.0°
0x1	ADC sample lags by 22.5°
0x2	ADC sample lags by 45.0°
0x3	ADC sample lags by 67.5°
0x4	ADC sample lags by 90.0°
0x5	ADC sample lags by 112.5°
0x6	ADC sample lags by 135.0°
0x7	ADC sample lags by 157.5°
0x8	ADC sample lags by 180.0°
0x9	ADC sample lags by 202.5°
0xA	ADC sample lags by 225.0°
0xB	ADC sample lags by 247.5°
0xC	ADC sample lags by 270.0°
0xD	ADC sample lags by 292.5°
0xE	ADC sample lags by 315.0°
0xF	ADC sample lags by 337.5°

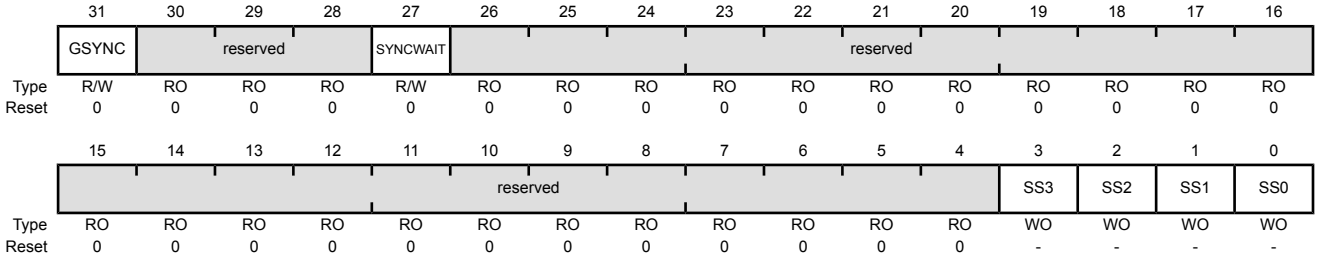
Register 10: ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028

This register provides a mechanism for application software to initiate sampling in the sample sequencers. Sample sequences can be initiated individually or in any combination. When multiple sequences are triggered simultaneously, the priority encodings in **ADCSSPRI** dictate execution order.

This register also provides a means to configure and then initiate concurrent sampling on all ADC modules. To do this, the first ADC module should be configured. The **ADCPSSI** register for that module should then be written. The appropriate SS bits should be set along with the SYNCWAIT bit. Additional ADC modules should then be configured following the same procedure. Once the final ADC module is configured, its **ADCPSSI** register should be written with the appropriate SS bits set along with the GSYNC bit. All of the ADC modules then begin concurrent sampling according to their configuration.

ADC Processor Sample Sequence Initiate (ADCPSSI)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x028
Type R/W, reset -



Bit/Field	Name	Type	Reset	Description
31	GSYNC	R/W	0	Global Synchronize Value Description 1 This bit initiates sampling in multiple ADC modules at the same time. Any ADC module that has been initialized by setting an SSn bit and the SYNCWAIT bit starts sampling once this bit is written. 0 This bit is cleared once sampling has been initiated.
30:28	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
27	SYNCWAIT	R/W	0	Synchronize Wait Value Description 1 This bit allows the sample sequences to be initiated, but delays sampling until the GSYNC bit is set. 0 Sampling begins when a sample sequence has been initiated.
26:4	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

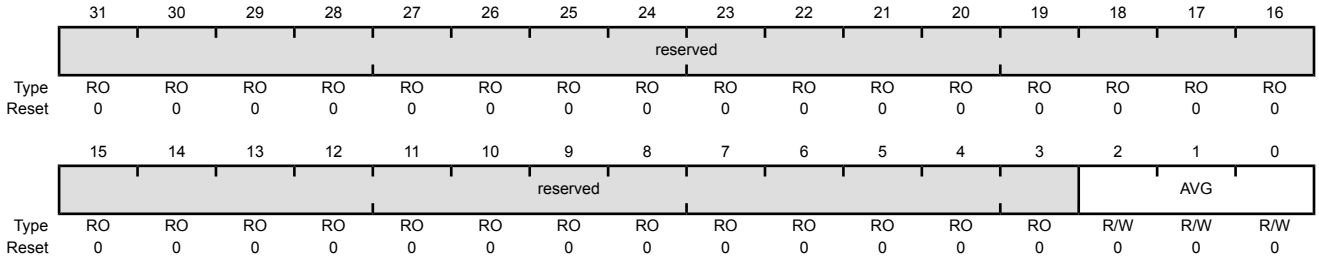
Bit/Field	Name	Type	Reset	Description
3	SS3	WO	-	SS3 Initiate Value Description 1 Begin sampling on Sample Sequencer 3, if the sequencer is enabled in the **ADCACTSS** register. 0 No effect. Only a write by software is valid; a read of this register returns no meaningful data.
2	SS2	WO	-	SS2 Initiate Value Description 1 Begin sampling on Sample Sequencer 2, if the sequencer is enabled in the **ADCACTSS** register. 0 No effect. Only a write by software is valid; a read of this register returns no meaningful data.
1	SS1	WO	-	SS1 Initiate Value Description 1 Begin sampling on Sample Sequencer 1, if the sequencer is enabled in the **ADCACTSS** register. 0 No effect. Only a write by software is valid; a read of this register returns no meaningful data.
0	SS0	WO	-	SS0 Initiate Value Description 1 Begin sampling on Sample Sequencer 0, if the sequencer is enabled in the **ADCACTSS** register. 0 No effect. Only a write by software is valid; a read of this register returns no meaningful data.

Register 11: ADC Sample Averaging Control (ADCSAC), offset 0x030

This register controls the amount of hardware averaging applied to conversion results. The final conversion result stored in the FIFO is averaged from 2^{AVG} consecutive ADC samples at the specified ADC speed. If AVG is 0, the sample is passed directly through without any averaging. If AVG=6, then 64 consecutive ADC samples are averaged to generate one result in the sequencer FIFO. An AVG=7 provides unpredictable results.

ADC Sample Averaging Control (ADCSAC)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x030
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	AVG	R/W	0x0	Hardware Averaging Control Specifies the amount of hardware averaging that will be applied to ADC samples. The AVG field can be any value between 0 and 6. Entering a value of 7 creates unpredictable results.

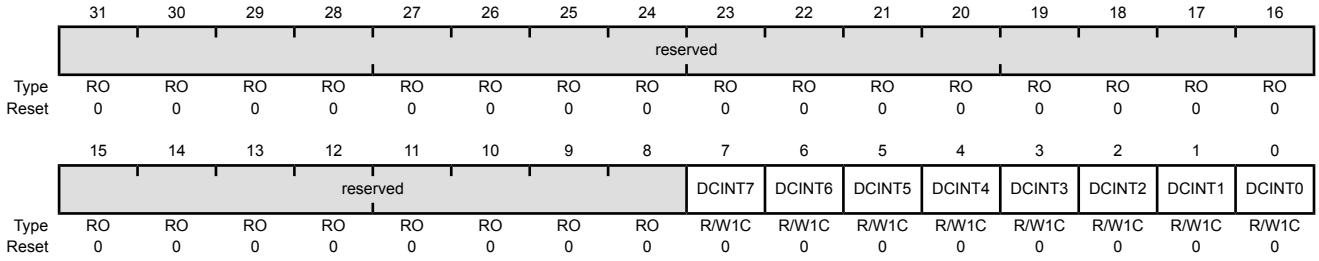
Value	Description
0x0	No hardware oversampling
0x1	2x hardware oversampling
0x2	4x hardware oversampling
0x3	8x hardware oversampling
0x4	16x hardware oversampling
0x5	32x hardware oversampling
0x6	64x hardware oversampling
0x7	reserved

Register 12: ADC Digital Comparator Interrupt Status and Clear (ADCDCISC), offset 0x034

This register provides status and acknowledgement of digital comparator interrupts. One bit is provided for each comparator.

ADC Digital Comparator Interrupt Status and Clear (ADCDCISC)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x034
Type R/W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	DCINT7	R/W1C	0	Digital Comparator 7 Interrupt Status and Clear Value Description 1 Digital Comparator 7 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.
6	DCINT6	R/W1C	0	Digital Comparator 6 Interrupt Status and Clear Value Description 1 Digital Comparator 6 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.
5	DCINT5	R/W1C	0	Digital Comparator 5 Interrupt Status and Clear Value Description 1 Digital Comparator 5 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.

Bit/Field	Name	Type	Reset	Description
4	DCINT4	R/W1C	0	Digital Comparator 4 Interrupt Status and Clear Value Description 1 Digital Comparator 4 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.
3	DCINT3	R/W1C	0	Digital Comparator 3 Interrupt Status and Clear Value Description 1 Digital Comparator 3 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.
2	DCINT2	R/W1C	0	Digital Comparator 2 Interrupt Status and Clear Value Description 1 Digital Comparator 2 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.
1	DCINT1	R/W1C	0	Digital Comparator 1 Interrupt Status and Clear Value Description 1 Digital Comparator 1 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.
0	DCINT0	R/W1C	0	Digital Comparator 0 Interrupt Status and Clear Value Description 1 Digital Comparator 0 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.

Register 13: ADC Control (ADCCTL), offset 0x038

This register configures various ADC module attributes, including the ADC resolution and the voltage reference. The resolution of the ADC defaults to 10-bit for backwards compatibility with other members of the Stellaris family, but can be configured to 12-bit resolution. The voltage reference for the conversion can be the internal 3.0-V reference, an external voltage reference in the range of 2.4 V to 3.06 V, or an external voltage reference in the range of 0.8 V to 1.02 V.

ADC Control (ADCCTL)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x038
Type R/W, reset 0x0000.0000

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

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

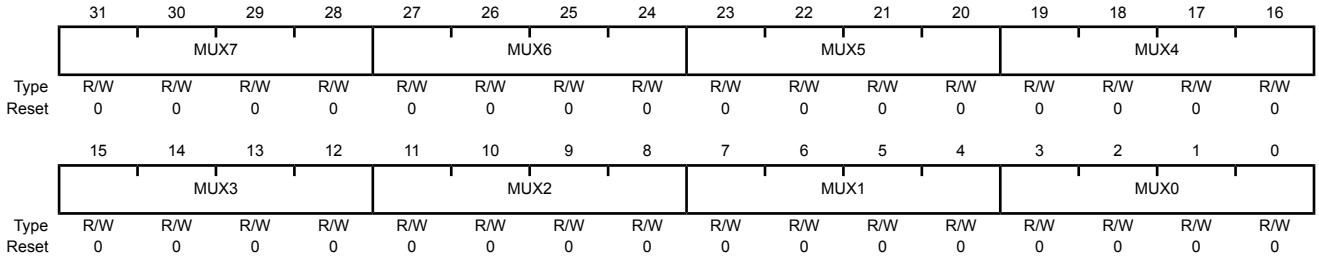
Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	RES	R/W	0	Sample Resolution Value Description 1 The ADC returns 12-bit data to the FIFOs. 0 The ADC returns 10-bit data to the FIFOs.
3:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1:0	VREF	R/W	0x0	Voltage Reference Select Value Description 0x0 Internal Reference The internal reference as the voltage reference. The conversion range is from 0 V to 3.0 V. 0x1 3.0 V External Reference A 3.0 V external VREFA input is the voltage reference. The ADC conversion range is 0.0 V to the voltage of the VREFA input. 0x2 Reserved 0x3 1.0 V External Reference A 1.0 V external VREFA input is the voltage reference. The ADC conversion range is 0.0 V to three times the voltage of the VREFA input.

Register 14: ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0), offset 0x040

This register defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 0. This register is 32 bits wide and contains information for eight possible samples.

ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x040
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:28	MUX7	R/W	0x0	8th Sample Input Select The MUX7 field is used during the eighth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion. The value set here indicates the corresponding pin, for example, a value of 0x1 indicates the input is AIN1.
27:24	MUX6	R/W	0x0	7th Sample Input Select The MUX6 field is used during the seventh sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
23:20	MUX5	R/W	0x0	6th Sample Input Select The MUX5 field is used during the sixth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
19:16	MUX4	R/W	0x0	5th Sample Input Select The MUX4 field is used during the fifth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
15:12	MUX3	R/W	0x0	4th Sample Input Select The MUX3 field is used during the fourth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
11:8	MUX2	R/W	0x0	3rd Sample Input Select The MUX2 field is used during the third sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.

Bit/Field	Name	Type	Reset	Description
7:4	MUX1	R/W	0x0	2nd Sample Input Select The MUX1 field is used during the second sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
3:0	MUX0	R/W	0x0	1st Sample Input Select The MUX0 field is used during the first sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.

Register 15: ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044

This register contains the configuration information for each sample for a sequence executed with a sample sequencer. When configuring a sample sequence, the END bit must be set for the final sample, whether it be after the first sample, eighth sample, or any sample in between. This register is 32 bits wide and contains information for eight possible samples.

ADC Sample Sequence Control 0 (ADCSSCTL0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x044
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TS7	IE7	END7	D7	TS6	IE6	END6	D6	TS5	IE5	END5	D5	TS4	IE4	END4	D4
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	TS7	R/W	0	8th Sample Temp Sensor Select Value Description 1 The temperature sensor is read during the eighth sample of the sample sequence. 0 The input pin specified by the **ADCSSMUXn** register is read during the eighth sample of the sample sequence.
30	IE7	R/W	0	8th Sample Interrupt Enable Value Description 1 The raw interrupt signal (INR0 bit) is asserted at the end of the eighth sample's conversion. If the MASK0 bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller. 0 The raw interrupt is not asserted to the interrupt controller. It is legal to have multiple samples within a sequence generate interrupts.
29	END7	R/W	0	8th Sample is End of Sequence Value Description 1 The eighth sample is the last sample of the sequence. 0 Another sample in the sequence is the final sample. It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero.

Bit/Field	Name	Type	Reset	Description
28	D7	R/W	0	8th Sample Diff Input Select Value Description 1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". 0 The analog inputs are not differentially sampled. Because the temperature sensor does not have a differential option, this bit must not be set when the TS7 bit is set.
27	TS6	R/W	0	7th Sample Temp Sensor Select Same definition as TS7 but used during the seventh sample.
26	IE6	R/W	0	7th Sample Interrupt Enable Same definition as IE7 but used during the seventh sample.
25	END6	R/W	0	7th Sample is End of Sequence Same definition as END7 but used during the seventh sample.
24	D6	R/W	0	7th Sample Diff Input Select Same definition as D7 but used during the seventh sample.
23	TS5	R/W	0	6th Sample Temp Sensor Select Same definition as TS7 but used during the sixth sample.
22	IE5	R/W	0	6th Sample Interrupt Enable Same definition as IE7 but used during the sixth sample.
21	END5	R/W	0	6th Sample is End of Sequence Same definition as END7 but used during the sixth sample.
20	D5	R/W	0	6th Sample Diff Input Select Same definition as D7 but used during the sixth sample.
19	TS4	R/W	0	5th Sample Temp Sensor Select Same definition as TS7 but used during the fifth sample.
18	IE4	R/W	0	5th Sample Interrupt Enable Same definition as IE7 but used during the fifth sample.
17	END4	R/W	0	5th Sample is End of Sequence Same definition as END7 but used during the fifth sample.
16	D4	R/W	0	5th Sample Diff Input Select Same definition as D7 but used during the fifth sample.
15	TS3	R/W	0	4th Sample Temp Sensor Select Same definition as TS7 but used during the fourth sample.
14	IE3	R/W	0	4th Sample Interrupt Enable Same definition as IE7 but used during the fourth sample.

Bit/Field	Name	Type	Reset	Description
13	END3	R/W	0	4th Sample is End of Sequence Same definition as END7 but used during the fourth sample.
12	D3	R/W	0	4th Sample Diff Input Select Same definition as D7 but used during the fourth sample.
11	TS2	R/W	0	3rd Sample Temp Sensor Select Same definition as TS7 but used during the third sample.
10	IE2	R/W	0	3rd Sample Interrupt Enable Same definition as IE7 but used during the third sample.
9	END2	R/W	0	3rd Sample is End of Sequence Same definition as END7 but used during the third sample.
8	D2	R/W	0	3rd Sample Diff Input Select Same definition as D7 but used during the third sample.
7	TS1	R/W	0	2nd Sample Temp Sensor Select Same definition as TS7 but used during the second sample.
6	IE1	R/W	0	2nd Sample Interrupt Enable Same definition as IE7 but used during the second sample.
5	END1	R/W	0	2nd Sample is End of Sequence Same definition as END7 but used during the second sample.
4	D1	R/W	0	2nd Sample Diff Input Select Same definition as D7 but used during the second sample.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as TS7 but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as IE7 but used during the first sample.
1	END0	R/W	0	1st Sample is End of Sequence Same definition as END7 but used during the first sample.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as D7 but used during the first sample.

Register 16: ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048

Register 17: ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068

Register 18: ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088

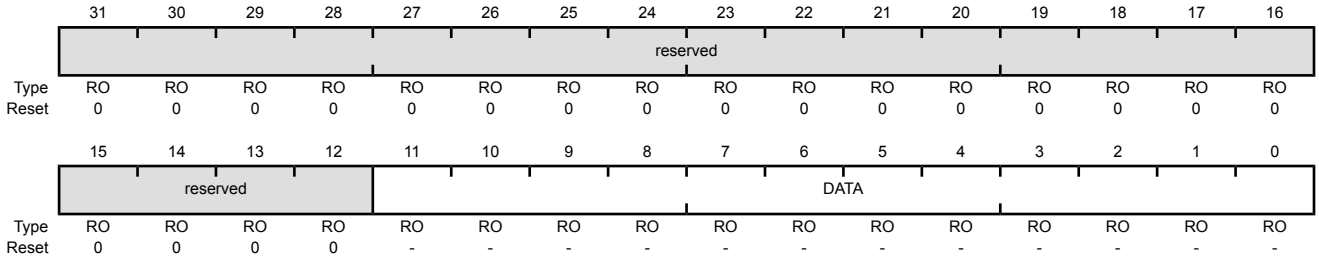
Register 19: ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8

Important: This register is read-sensitive. See the register description for details.

This register contains the conversion results for samples collected with the sample sequencer (the **ADCSSFIFO0** register is used for Sample Sequencer 0, **ADCSSFIFO1** for Sequencer 1, **ADCSSFIFO2** for Sequencer 2, and **ADCSSFIFO3** for Sequencer 3). Reads of this register return conversion result data in the order sample 0, sample 1, and so on, until the FIFO is empty. If the FIFO is not properly handled by software, overflow and underflow conditions are registered in the **ADCOSTAT** and **ADCUSTAT** registers.

ADC Sample Sequence Result FIFO n (ADCSSFIFOn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x048
Type RO, reset -



Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11:0	DATA	RO	-	Conversion Result Data

Register 20: ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C

Register 21: ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C

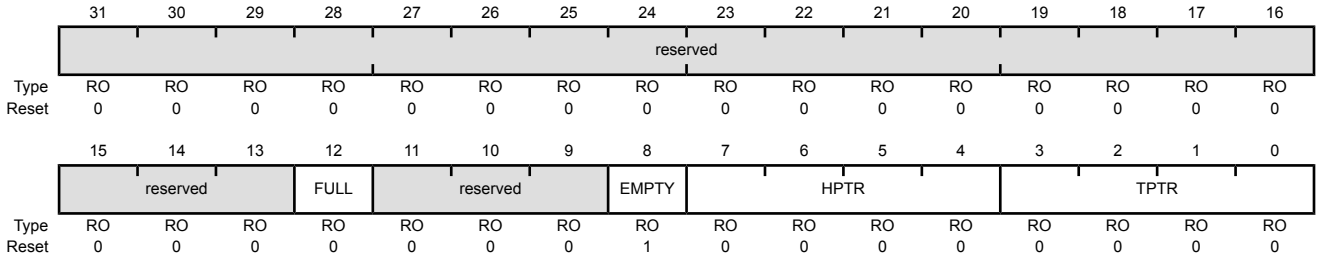
Register 22: ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C

Register 23: ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC

This register provides a window into the sample sequencer, providing full/empty status information as well as the positions of the head and tail pointers. The reset value of 0x100 indicates an empty FIFO with the head and tail pointers both pointing to index 0. The **ADCSSFSTAT0** register provides status on FIFO0, which has 8 entries; **ADCSSFSTAT1** on FIFO1, which has 4 entries; **ADCSSFSTAT2** on FIFO2, which has 4 entries; and **ADCSSFSTAT3** on FIFO3 which has a single entry.

ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x04C
Type RO, reset 0x0000.0100



Bit/Field	Name	Type	Reset	Description
31:13	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	FULL	RO	0	FIFO Full Value Description 1 The FIFO is currently full. 0 The FIFO is not currently full.
11:9	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	EMPTY	RO	1	FIFO Empty Value Description 1 The FIFO is currently empty. 0 The FIFO is not currently empty.

Bit/Field	Name	Type	Reset	Description
7:4	HPTR	RO	0x0	FIFO Head Pointer This field contains the current "head" pointer index for the FIFO, that is, the next entry to be written. Valid values are 0x0-0x7 for FIFO0; 0x0-0x3 for FIFO1 and FIFO2; and 0x0 for FIFO3.
3:0	TPTR	RO	0x0	FIFO Tail Pointer This field contains the current "tail" pointer index for the FIFO, that is, the next entry to be read. Valid values are 0x0-0x7 for FIFO0; 0x0-0x3 for FIFO1 and FIFO2; and 0x0 for FIFO3.

Register 24: ADC Sample Sequence 0 Operation (ADCSSOP0), offset 0x050

This register determines whether the sample from the given conversion on Sample Sequence 0 is saved in the Sample Sequence FIFO0 or sent to the digital comparator unit.

ADC Sample Sequence 0 Operation (ADCSSOP0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x050
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved			S7DCOP	reserved			S6DCOP	reserved			S5DCOP	reserved			S4DCOP
Type	RO	RO	RO	R/W	RO	RO	RO	R/W	RO	RO	RO	R/W	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			S3DCOP	reserved			S2DCOP	reserved			S1DCOP	reserved			S0DCOP
Type	RO	RO	RO	R/W	RO	RO	RO	R/W	RO	RO	RO	R/W	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	S7DCOP	R/W	0	Sample 7 Digital Comparator Operation Value Description 1 The eighth sample is sent to the digital comparator unit specified by the S7DCSEL bit in the **ADCSSDC0** register, and the value is not written to the FIFO. 0 The eighth sample is saved in Sample Sequence FIFO0.
27:25	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24	S6DCOP	R/W	0	Sample 6 Digital Comparator Operation Same definition as S7DCOP but used during the seventh sample.
23:21	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
20	S5DCOP	R/W	0	Sample 5 Digital Comparator Operation Same definition as S7DCOP but used during the sixth sample.
19:17	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	S4DCOP	R/W	0	Sample 4 Digital Comparator Operation Same definition as S7DCOP but used during the fifth sample.
15:13	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

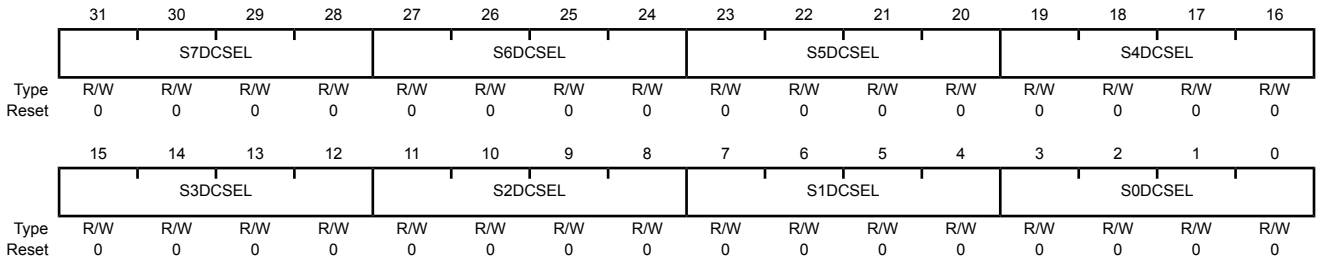
Bit/Field	Name	Type	Reset	Description
12	S3DCOP	R/W	0	Sample 3 Digital Comparator Operation Same definition as S7DCOP but used during the fourth sample.
11:9	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	S2DCOP	R/W	0	Sample 2 Digital Comparator Operation Same definition as S7DCOP but used during the third sample.
7:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	S1DCOP	R/W	0	Sample 1 Digital Comparator Operation Same definition as S7DCOP but used during the second sample.
3:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	S0DCOP	R/W	0	Sample 0 Digital Comparator Operation Same definition as S7DCOP but used during the first sample.

Register 25: ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0), offset 0x054

This register determines which digital comparator receives the sample from the given conversion on Sample Sequence 0, if the corresponding SnDC0P bit in the **ADCSSOP0** register is set.

ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x054
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:28	S7DCSEL	R/W	0x0	Sample 7 Digital Comparator Select When the S7DC0P bit in the **ADCSSOP0** register is set, this field indicates which digital comparator unit (and its associated set of control registers) receives the eighth sample from Sample Sequencer 0. **Note:** Values not listed are reserved. Value Description 0x0 Digital Comparator Unit 0 (**ADCDCCMP0** and **ADCDCCTL0**) 0x1 Digital Comparator Unit 1 (**ADCDCCMP1** and **ADCDCCTL1**) 0x2 Digital Comparator Unit 2 (**ADCDCCMP2** and **ADCDCCTL2**) 0x3 Digital Comparator Unit 3 (**ADCDCCMP3** and **ADCDCCTL3**) 0x4 Digital Comparator Unit 4 (**ADCDCCMP4** and **ADCDCCTL4**) 0x5 Digital Comparator Unit 5 (**ADCDCCMP5** and **ADCDCCTL5**) 0x6 Digital Comparator Unit 6 (**ADCDCCMP6** and **ADCDCCTL6**) 0x7 Digital Comparator Unit 7 (**ADCDCCMP7** and **ADCDCCTL7**)
27:24	S6DCSEL	R/W	0x0	Sample 6 Digital Comparator Select This field has the same encodings as S7DCSEL but is used during the seventh sample.
23:20	S5DCSEL	R/W	0x0	Sample 5 Digital Comparator Select This field has the same encodings as S7DCSEL but is used during the sixth sample.
19:16	S4DCSEL	R/W	0x0	Sample 4 Digital Comparator Select This field has the same encodings as S7DCSEL but is used during the fifth sample.
15:12	S3DCSEL	R/W	0x0	Sample 3 Digital Comparator Select This field has the same encodings as S7DCSEL but is used during the fourth sample.

Bit/Field	Name	Type	Reset	Description
11:8	S2DCSEL	R/W	0x0	Sample 2 Digital Comparator Select This field has the same encodings as S7DCSEL but is used during the third sample.
7:4	S1DCSEL	R/W	0x0	Sample 1 Digital Comparator Select This field has the same encodings as S7DCSEL but is used during the second sample.
3:0	S0DCSEL	R/W	0x0	Sample 0 Digital Comparator Select This field has the same encodings as S7DCSEL but is used during the first sample.

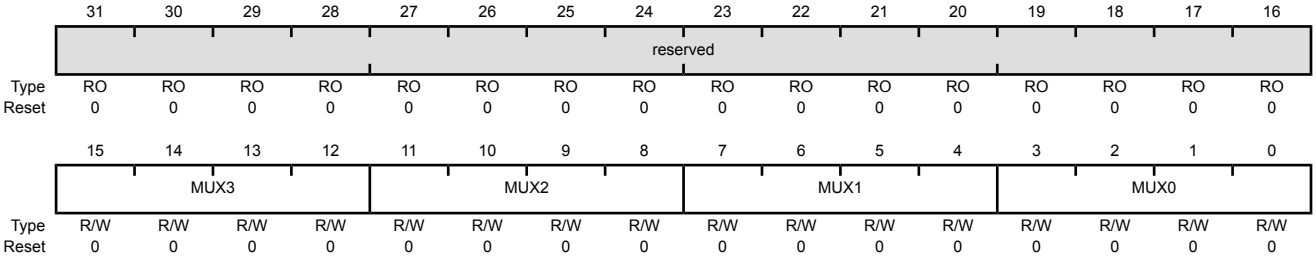
Register 26: ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060

Register 27: ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080

This register defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 1 or 2. These registers are 16 bits wide and contain information for four possible samples. See the **ADCSSMUX0** register on page 647 for detailed bit descriptions. The **ADCSSMUX1** register affects Sample Sequencer 1 and the **ADCSSMUX2** register affects Sample Sequencer 2.

ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x060
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:12	MUX3	R/W	0x0	4th Sample Input Select
11:8	MUX2	R/W	0x0	3rd Sample Input Select
7:4	MUX1	R/W	0x0	2nd Sample Input Select
3:0	MUX0	R/W	0x0	1st Sample Input Select

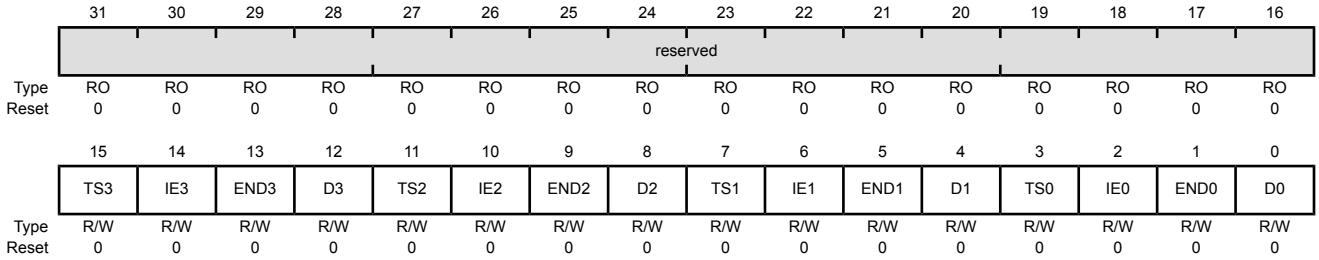
Register 28: ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064

Register 29: ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084

These registers contain the configuration information for each sample for a sequence executed with Sample Sequencer 1 or 2. When configuring a sample sequence, the END bit must be set for the final sample, whether it be after the first sample, fourth sample, or any sample in between. These registers are 16-bits wide and contain information for four possible samples. See the **ADCSSCTL0** register on page 649 for detailed bit descriptions. The **ADCSSCTL1** register configures Sample Sequencer 1 and the **ADCSSCTL2** register configures Sample Sequencer 2.

ADC Sample Sequence Control 1 (ADCSSCTL1)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x064
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	TS3	R/W	0	4th Sample Temp Sensor Select Same definition as TS7 but used during the fourth sample.
14	IE3	R/W	0	4th Sample Interrupt Enable Same definition as IE7 but used during the fourth sample.
13	END3	R/W	0	4th Sample is End of Sequence Same definition as END7 but used during the fourth sample.
12	D3	R/W	0	4th Sample Diff Input Select Same definition as D7 but used during the fourth sample.
11	TS2	R/W	0	3rd Sample Temp Sensor Select Same definition as TS7 but used during the third sample.
10	IE2	R/W	0	3rd Sample Interrupt Enable Same definition as IE7 but used during the third sample.
9	END2	R/W	0	3rd Sample is End of Sequence Same definition as END7 but used during the third sample.
8	D2	R/W	0	3rd Sample Diff Input Select Same definition as D7 but used during the third sample.

Bit/Field	Name	Type	Reset	Description
7	TS1	R/W	0	2nd Sample Temp Sensor Select Same definition as TS7 but used during the second sample.
6	IE1	R/W	0	2nd Sample Interrupt Enable Same definition as IE7 but used during the second sample.
5	END1	R/W	0	2nd Sample is End of Sequence Same definition as END7 but used during the second sample.
4	D1	R/W	0	2nd Sample Diff Input Select Same definition as D7 but used during the second sample.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as TS7 but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as IE7 but used during the first sample.
1	END0	R/W	0	1st Sample is End of Sequence Same definition as END7 but used during the first sample.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as D7 but used during the first sample.

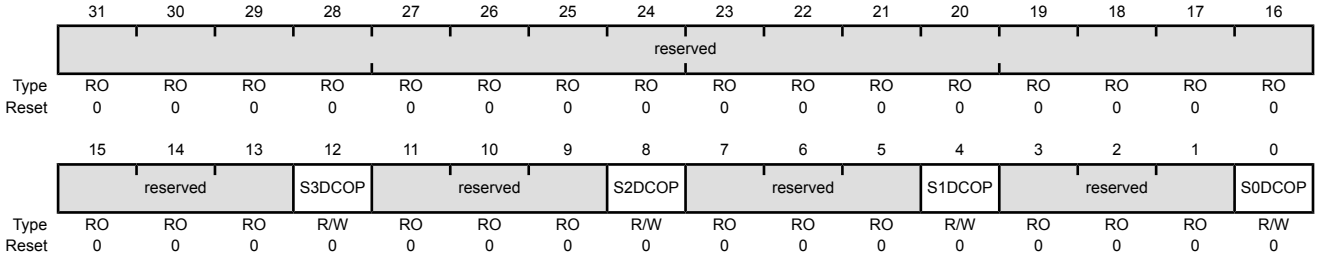
Register 30: ADC Sample Sequence 1 Operation (ADCSSOP1), offset 0x070

Register 31: ADC Sample Sequence 2 Operation (ADCSSOP2), offset 0x090

This register determines whether the sample from the given conversion on Sample Sequence n is saved in the Sample Sequence n FIFO or sent to the digital comparator unit. The **ADCSSOP1** register controls Sample Sequencer 1 and the **ADCSSOP2** register controls Sample Sequencer 2.

ADC Sample Sequence 1 Operation (ADCSSOP1)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x070
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:13	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	S3DCOP	R/W	0	Sample 3 Digital Comparator Operation Value Description 1 The fourth sample is sent to the digital comparator unit specified by the S3DCSEL bit in the **ADCSSDC0n** register, and the value is not written to the FIFO. 0 The fourth sample is saved in Sample Sequence FIFOn.
11:9	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	S2DCOP	R/W	0	Sample 2 Digital Comparator Operation Same definition as S3DCOP but used during the third sample.
7:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	S1DCOP	R/W	0	Sample 1 Digital Comparator Operation Same definition as S3DCOP but used during the second sample.
3:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	S0DCOP	R/W	0	Sample 0 Digital Comparator Operation Same definition as S3DCOP but used during the first sample.

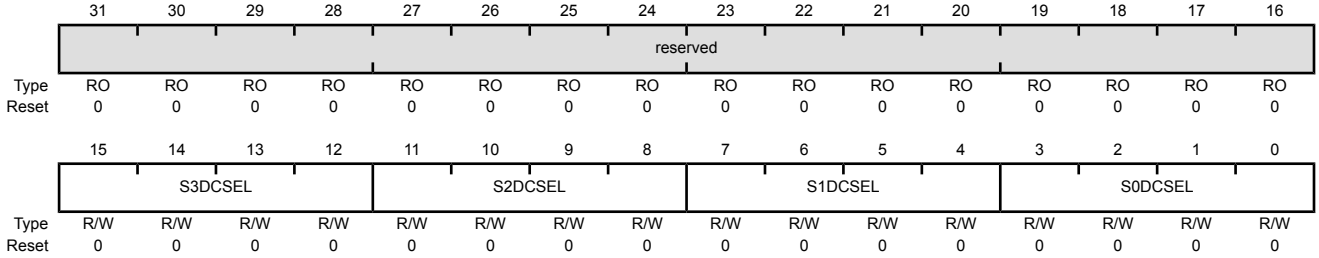
Register 32: ADC Sample Sequence 1 Digital Comparator Select (ADCSSDC1), offset 0x074

Register 33: ADC Sample Sequence 2 Digital Comparator Select (ADCSSDC2), offset 0x094

These registers determine which digital comparator receives the sample from the given conversion on Sample Sequence n if the corresponding SnDC0P bit in the **ADCSSOPn** register is set. The **ADCSSDC1** register controls the selection for Sample Sequencer 1 and the **ADCSSDC2** register controls the selection for Sample Sequencer 2.

ADC Sample Sequence 1 Digital Comparator Select (ADCSSDC1)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x074
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:12	S3DCSEL	R/W	0x0	Sample 3 Digital Comparator Select When the S3DC0P bit in the **ADCSSOPn** register is set, this field indicates which digital comparator unit (and its associated set of control registers) receives the eighth sample from Sample Sequencer n. **Note:** Values not listed are reserved. Value Description 0x0 Digital Comparator Unit 0 (**ADCDCCMP0** and **ADCCCTL0**) 0x1 Digital Comparator Unit 1 (**ADCDCCMP1** and **ADCCCTL1**) 0x2 Digital Comparator Unit 2 (**ADCDCCMP2** and **ADCCCTL2**) 0x3 Digital Comparator Unit 3 (**ADCDCCMP3** and **ADCCCTL3**) 0x4 Digital Comparator Unit 4 (**ADCDCCMP4** and **ADCCCTL4**) 0x5 Digital Comparator Unit 5 (**ADCDCCMP5** and **ADCCCTL5**) 0x6 Digital Comparator Unit 6 (**ADCDCCMP6** and **ADCCCTL6**) 0x7 Digital Comparator Unit 7 (**ADCDCCMP7** and **ADCCCTL7**)
11:8	S2DCSEL	R/W	0x0	Sample 2 Digital Comparator Select This field has the same encodings as S3DCSEL but is used during the third sample.

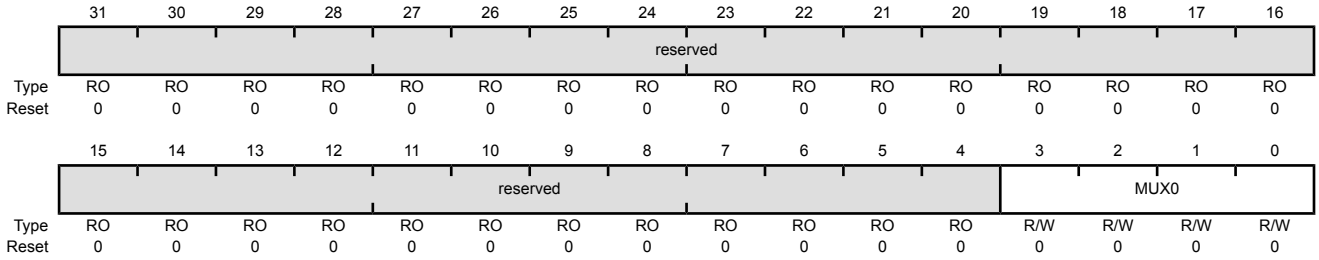
Bit/Field	Name	Type	Reset	Description
7:4	S1DCSEL	R/W	0x0	Sample 1 Digital Comparator Select This field has the same encodings as S3DCSEL but is used during the second sample.
3:0	S0DCSEL	R/W	0x0	Sample 0 Digital Comparator Select This field has the same encodings as S3DCSEL but is used during the first sample.

Register 34: ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0

This register defines the analog input configuration for the sample executed with Sample Sequencer 3. This register is 4 bits wide and contains information for one possible sample. See the **ADCSSMUX0** register on page 647 for detailed bit descriptions.

ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x0A0
Type R/W, reset 0x0000.0000



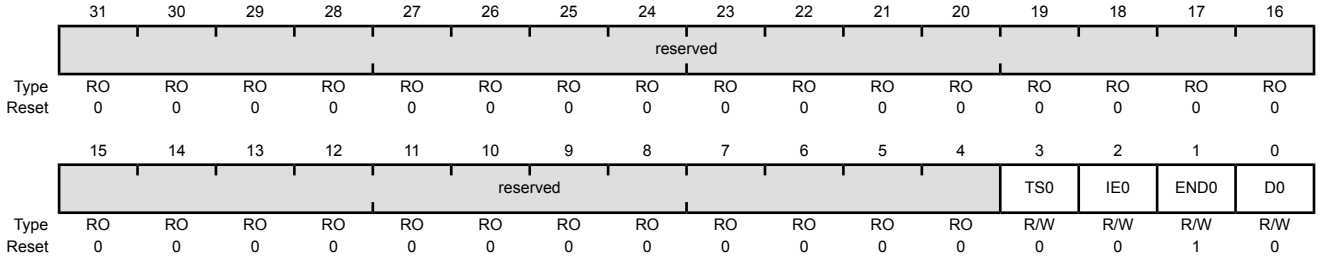
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	MUX0	R/W	0	1st Sample Input Select

Register 35: ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4

This register contains the configuration information for a sample executed with Sample Sequencer 3. The END0 bit is always set as this sequencer can execute only one sample. This register is 4 bits wide and contains information for one possible sample. See the **ADCSSCTL0** register on page 649 for detailed bit descriptions.

ADC Sample Sequence Control 3 (ADCSSCTL3)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x0A4
Type R/W, reset 0x0000.0002



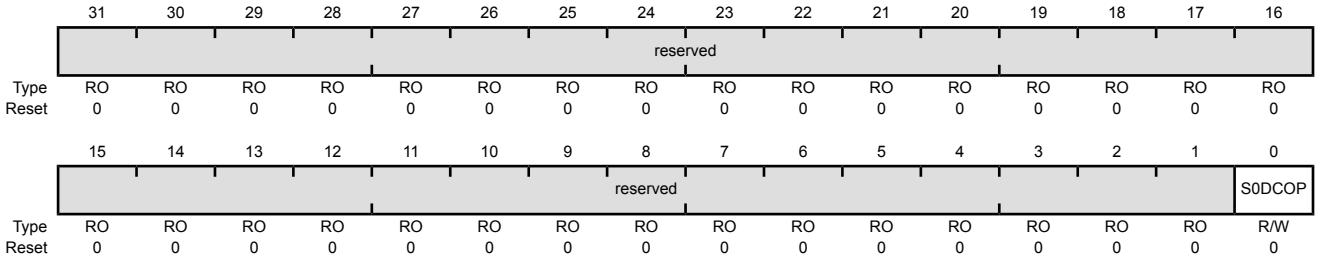
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as TS7 but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as IE7 but used during the first sample.
1	END0	R/W	1	1st Sample is End of Sequence Same definition as END7 but used during the first sample. Because this sequencer has only one entry, this bit must be set.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as D7 but used during the first sample.

Register 36: ADC Sample Sequence 3 Operation (ADCSSOP3), offset 0x0B0

This register determines whether the sample from the given conversion on Sample Sequence 3 is saved in the Sample Sequence 3 FIFO or sent to the digital comparator unit.

ADC Sample Sequence 3 Operation (ADCSSOP3)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x0B0
Type R/W, reset 0x0000.0000



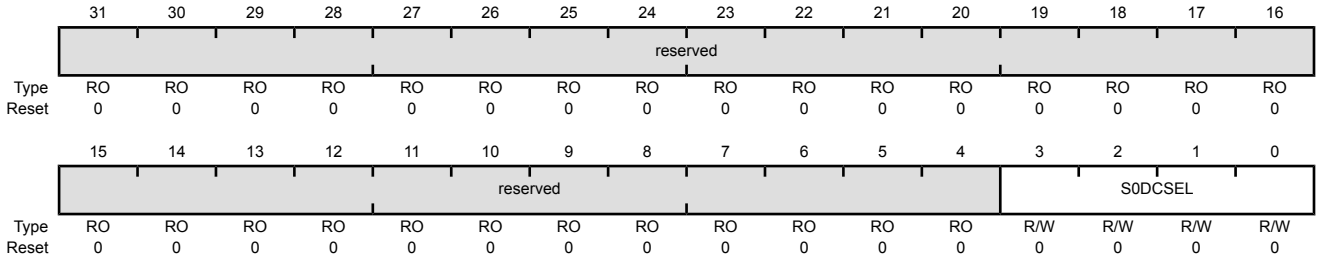
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	S0DCOP	R/W	0	Sample 0 Digital Comparator Operation Value Description 1 The sample is sent to the digital comparator unit specified by the S0DCSEL bit in the **ADCSSDC03** register, and the value is not written to the FIFO. 0 The sample is saved in Sample Sequence FIFO3.

Register 37: ADC Sample Sequence 3 Digital Comparator Select (ADCSSDC3), offset 0x0B4

This register determines which digital comparator receives the sample from the given conversion on Sample Sequence 3 if the corresponding SnDCOP bit in the **ADCSSOP3** register is set.

ADC Sample Sequence 3 Digital Comparator Select (ADCSSDC3)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x0B4
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	S0DCSEL	R/W	0x0	Sample 0 Digital Comparator Select When the S0DCOP bit in the **ADCSSOP3** register is set, this field indicates which digital comparator unit (and its associated set of control registers) receives the sample from Sample Sequencer 3. **Note:** Values not listed are reserved. Value Description 0x0 Digital Comparator Unit 0 (**ADCDCCMP0** and **ADCCCTL0**) 0x1 Digital Comparator Unit 1 (**ADCDCCMP1** and **ADCCCTL1**) 0x2 Digital Comparator Unit 2 (**ADCDCCMP2** and **ADCCCTL2**) 0x3 Digital Comparator Unit 3 (**ADCDCCMP3** and **ADCCCTL3**) 0x4 Digital Comparator Unit 4 (**ADCDCCMP4** and **ADCCCTL4**) 0x5 Digital Comparator Unit 5 (**ADCDCCMP5** and **ADCCCTL5**) 0x6 Digital Comparator Unit 6 (**ADCDCCMP6** and **ADCCCTL6**) 0x7 Digital Comparator Unit 7 (**ADCDCCMP7** and **ADCCCTL7**)

Register 38: ADC Digital Comparator Reset Initial Conditions (ADCDCRIC), offset 0xD00

This register provides the ability to reset any of the digital comparator interrupt or trigger functions back to their initial conditions. Resetting these functions ensures that the data that is being used by the interrupt and trigger functions in the digital comparator unit is not stale.

ADC Digital Comparator Reset Initial Conditions (ADCDCRIC)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0xD00
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved								DCTRIG7	DCTRIG6	DCTRIG5	DCTRIG4	DCTRIG3	DCTRIG2	DCTRIG1	DCTRIG0
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DCINT7	DCINT6	DCINT5	DCINT4	DCINT3	DCINT2	DCINT1	DCINT0
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23	DCTRIG7	R/W	0	Digital Comparator Trigger 7 Value Description 1 Resets the Digital Comparator 7 trigger unit to its initial conditions. 0 No effect. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. After setting this bit, software should wait until the bit clears before continuing.
22	DCTRIG6	R/W	0	Digital Comparator Trigger 6 Value Description 1 Resets the Digital Comparator 6 trigger unit to its initial conditions. 0 No effect. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.

Bit/Field	Name	Type	Reset	Description
21	DCTRIG5	R/W	0	Digital Comparator Trigger 5 Value Description 1 Resets the Digital Comparator 5 trigger unit to its initial conditions. 0 No effect. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
20	DCTRIG4	R/W	0	Digital Comparator Trigger 4 Value Description 1 Resets the Digital Comparator 4 trigger unit to its initial conditions. 0 No effect. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
19	DCTRIG3	R/W	0	Digital Comparator Trigger 3 Value Description 1 Resets the Digital Comparator 3 trigger unit to its initial conditions. 0 No effect. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
18	DCTRIG2	R/W	0	Digital Comparator Trigger 2 Value Description 1 Resets the Digital Comparator 2 trigger unit to its initial conditions. 0 No effect. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.

Bit/Field	Name	Type	Reset	Description
17	DCTRIG1	R/W	0	Digital Comparator Trigger 1 Value Description 1 Resets the Digital Comparator 1 trigger unit to its initial conditions. 0 No effect. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
16	DCTRIG0	R/W	0	Digital Comparator Trigger 0 Value Description 1 Resets the Digital Comparator 0 trigger unit to its initial conditions. 0 No effect. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
15:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	DCINT7	R/W	0	Digital Comparator Interrupt 7 Value Description 1 Resets the Digital Comparator 7 interrupt unit to its initial conditions. 0 No effect. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
6	DCINT6	R/W	0	Digital Comparator Interrupt 6 Value Description 1 Resets the Digital Comparator 6 interrupt unit to its initial conditions. 0 No effect. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.

Bit/Field	Name	Type	Reset	Description
5	DCINT5	R/W	0	Digital Comparator Interrupt 5 Value Description 1 Resets the Digital Comparator 5 interrupt unit to its initial conditions. 0 No effect. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
4	DCINT4	R/W	0	Digital Comparator Interrupt 4 Value Description 1 Resets the Digital Comparator 4 interrupt unit to its initial conditions. 0 No effect. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
3	DCINT3	R/W	0	Digital Comparator Interrupt 3 Value Description 1 Resets the Digital Comparator 3 interrupt unit to its initial conditions. 0 No effect. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
2	DCINT2	R/W	0	Digital Comparator Interrupt 2 Value Description 1 Resets the Digital Comparator 2 interrupt unit to its initial conditions. 0 No effect. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.

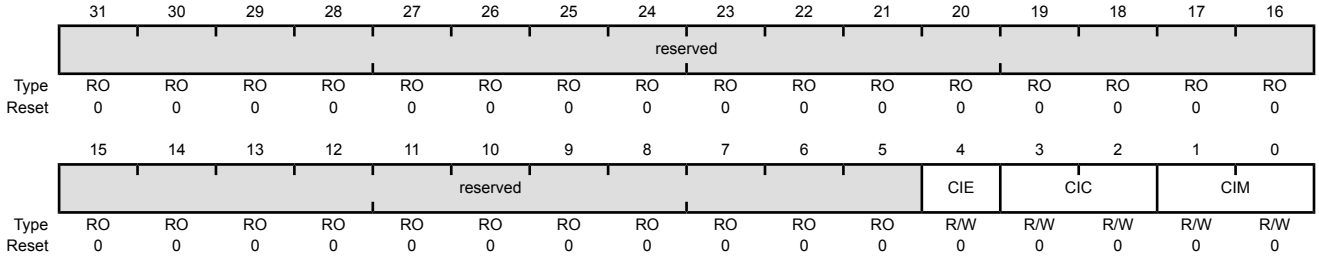
Bit/Field	Name	Type	Reset	Description
1	DCINT1	R/W	0	Digital Comparator Interrupt 1 Value Description 1 Resets the Digital Comparator 1 interrupt unit to its initial conditions. 0 No effect. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
0	DCINT0	R/W	0	Digital Comparator Interrupt 0 Value Description 1 Resets the Digital Comparator 0 interrupt unit to its initial conditions. 0 No effect. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.

Register 39: ADC Digital Comparator Control 0 (ADCDCCTL0), offset 0xE00
Register 40: ADC Digital Comparator Control 1 (ADCDCCTL1), offset 0xE04
Register 41: ADC Digital Comparator Control 2 (ADCDCCTL2), offset 0xE08
Register 42: ADC Digital Comparator Control 3 (ADCDCCTL3), offset 0xE0C
Register 43: ADC Digital Comparator Control 4 (ADCDCCTL4), offset 0xE10
Register 44: ADC Digital Comparator Control 5 (ADCDCCTL5), offset 0xE14
Register 45: ADC Digital Comparator Control 6 (ADCDCCTL6), offset 0xE18
Register 46: ADC Digital Comparator Control 7 (ADCDCCTL7), offset 0xE1C

This register provides the comparison encodings that generate an interrupt.

ADC Digital Comparator Control 0 (ADCDCCTL0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0xE00
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	CIE	R/W	0	Comparison Interrupt Enable

Value	Description
1	Enables the comparison interrupt. The ADC conversion data is used to determine if an interrupt should be generated according to the programming of the `CIC` and `CIM` fields.
0	Disables the comparison interrupt. ADC conversion data has no effect on interrupt generation.

Bit/Field	Name	Type	Reset	Description
3:2	CIC	R/W	0x0	Comparison Interrupt Condition This field specifies the operational region in which an interrupt is generated when the ADC conversion data is compared against the values of COMP0 and COMP1. The COMP0 and COMP1 fields are defined in the **ADCDCCMPx** registers. Value Description 0x0 Low Band ADC Data < COMP0 ≤ COMP1 0x1 Mid Band COMP0 ≤ ADC Data < COMP1 0x2 reserved 0x3 High Band COMP0 < COMP1 ≤ ADC Data
1:0	CIM	R/W	0x0	Comparison Interrupt Mode This field specifies the mode by which the interrupt comparison is made. Value Description 0x0 Always This mode generates an interrupt every time the ADC conversion data falls within the selected operational region. 0x1 Once This mode generates an interrupt the first time that the ADC conversion data enters the selected operational region. 0x2 Hysteresis Always This mode generates an interrupt when the ADC conversion data falls within the selected operational region and continues to generate the interrupt until the hysteresis condition is cleared by entering the opposite operational region. 0x3 Hysteresis Once This mode generates an interrupt the first time that the ADC conversion data falls within the selected operational region. No additional interrupts are generated until the hysteresis condition is cleared by entering the opposite operational region.

Register 47: ADC Digital Comparator Range 0 (ADCDCCMP0), offset 0xE40
Register 48: ADC Digital Comparator Range 1 (ADCDCCMP1), offset 0xE44
Register 49: ADC Digital Comparator Range 2 (ADCDCCMP2), offset 0xE48
Register 50: ADC Digital Comparator Range 3 (ADCDCCMP3), offset 0xE4C
Register 51: ADC Digital Comparator Range 4 (ADCDCCMP4), offset 0xE50
Register 52: ADC Digital Comparator Range 5 (ADCDCCMP5), offset 0xE54
Register 53: ADC Digital Comparator Range 6 (ADCDCCMP6), offset 0xE58
Register 54: ADC Digital Comparator Range 7 (ADCDCCMP7), offset 0xE5C

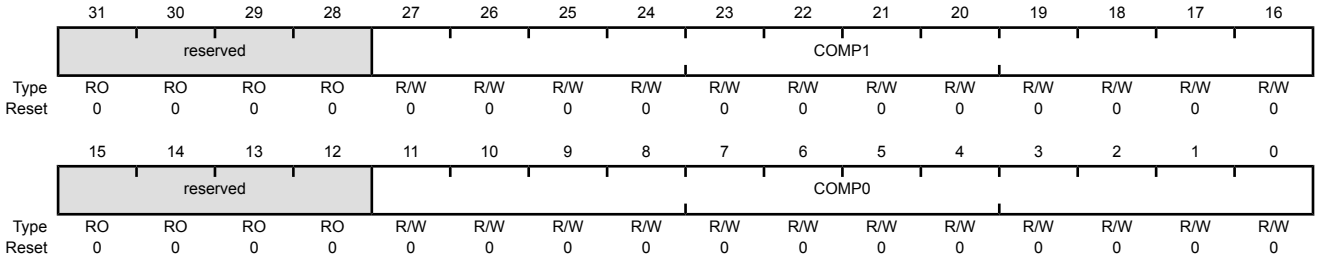
This register defines the comparison values that are used to determine if the ADC conversion data falls in the appropriate operating region.

Note: The value in the COMP1 field must be greater than or equal to the value in the COMP0 field or unexpected results can occur.

If the RES bit in the **ADCCTL** register is clear, selecting 10-bit resolution, use only bits [25:16] in the COMP1 field and bits [9:0] in the COMP0 field; otherwise unexpected results can occur.

ADC Digital Comparator Range 0 (ADCDCCMP0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0xE40
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:28	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
27:16	COMP1	R/W	0x000	Compare 1 The value in this field is compared against the ADC conversion data. The result of the comparison is used to determine if the data lies within the high-band region. Note that the value of COMP1 must be greater than or equal to the value of COMP0.
15:12	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
11:0	COMP0	R/W	0x000	Compare 0 The value in this field is compared against the ADC conversion data. The result of the comparison is used to determine if the data lies within the low-band region.

13 Universal Asynchronous Receivers/Transmitters (UARTs)

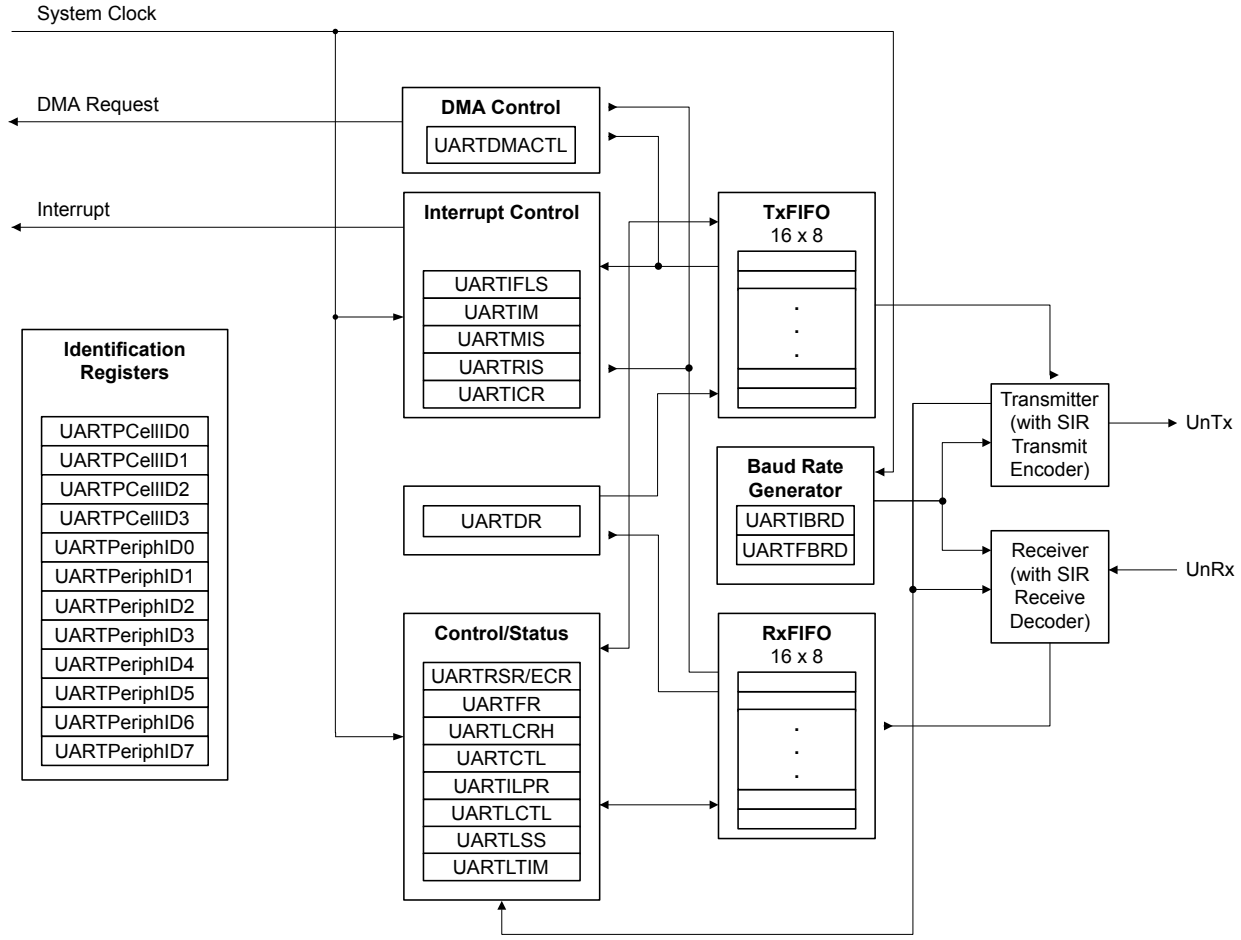
The Stellaris® LM3S9U81 controller includes three Universal Asynchronous Receiver/Transmitter (UART) with the following features:

- Programmable baud-rate generator allowing speeds up to 5 Mbps for regular speed (divide by 16) and 10 Mbps for high speed (divide by 8)
- Separate 16x8 transmit (TX) and receive (RX) FIFOs to reduce CPU interrupt service loading
- Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
- FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
- Standard asynchronous communication bits for start, stop, and parity
- Line-break generation and detection
- Fully programmable serial interface characteristics
  - 5, 6, 7, or 8 data bits
  - Even, odd, stick, or no-parity bit generation/detection
  - 1 or 2 stop bit generation
- IrDA serial-IR (SIR) encoder/decoder providing
  - Programmable use of IrDA Serial Infrared (SIR) or UART input/output
  - Support of IrDA SIR encoder/decoder functions for data rates up to 115.2 Kbps half-duplex
  - Support of normal 3/16 and low-power (1.41-2.23 µs) bit durations
  - Programmable internal clock generator enabling division of reference clock by 1 to 256 for low-power mode bit duration
- Support for communication with ISO 7816 smart cards
- Full modem handshake support (on UART1)
- LIN protocol support
- Standard FIFO-level and End-of-Transmission interrupts
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Separate channels for transmit and receive
  - Receive single request asserted when data is in the FIFO; burst request asserted at programmed FIFO level

– Transmit single request asserted when there is space in the FIFO; burst request asserted at programmed FIFO level

13.1 Block Diagram

Figure 13-1. UART Module Block Diagram



13.2 Signal Description

The following table lists the external signals of the UART module and describes the function of each. The UART signals are alternate functions for some GPIO signals and default to be GPIO signals at reset, with the exception of the U0Rx and U0Tx pins which default to the UART function. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for these UART signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 419) should be set to choose the UART function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 437) to assign the UART signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 395.

Table 13-1. UART Signals (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
U0Rx	26	PA0 (1)	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
U0Tx	27	PA1 (1)	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
U1CTS	2 10 34 50	PE6 (9) PD0 (9) PA6 (9) PJ3 (9)	I	TTL	UART module 1 Clear To Send modem flow control input signal.
U1DCD	1 11 35 52	PE7 (9) PD1 (9) PA7 (9) PJ4 (9)	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
U1DSR	47 53	PF0 (9) PJ5 (9)	I	TTL	UART module 1 Data Set Ready modem output control line.
U1DTR	55 100	PJ7 (9) PD7 (9)	O	TTL	UART module 1 Data Terminal Ready modem status input signal.
U1RI	97	PD4 (9)	I	TTL	UART module 1 Ring Indicator modem status input signal.
U1RTS	54 61	PJ6 (9) PF1 (9)	O	TTL	UART module 1 Request to Send modem flow control output line.
U1Rx	10 12 23 26 66 92	PD0 (5) PD2 (1) PC6 (5) PA0 (9) PB0 (5) PB4 (7)	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
U1Tx	11 13 22 27 67 91	PD1 (5) PD3 (1) PC7 (5) PA1 (9) PB1 (5) PB5 (7)	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
U2Rx	10 19 92 98	PD0 (4) PG0 (1) PB4 (4) PD5 (9)	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
U2Tx	6 11 18 99	PE4 (5) PD1 (4) PG1 (1) PD6 (9)	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.

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

Table 13-2. UART Signals (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
U0Rx	L3	PA0 (1)	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
U0Tx	M3	PA1 (1)	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.

Table 13-2. UART Signals (108BGA) *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
U1CTS	A1 G1 L6 M10	PE6 (9) PD0 (9) PA6 (9) PJ3 (9)	I	TTL	UART module 1 Clear To Send modem flow control input signal.
U1DCD	B1 G2 M6 K11	PE7 (9) PD1 (9) PA7 (9) PJ4 (9)	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
U1DSR	M9 K12	PF0 (9) PJ5 (9)	I	TTL	UART module 1 Data Set Ready modem output control line.
U1DTR	L12 A2	PJ7 (9) PD7 (9)	O	TTL	UART module 1 Data Terminal Ready modem status input signal.
U1RI	B5	PD4 (9)	I	TTL	UART module 1 Ring Indicator modem status input signal.
U1RTS	L10 H12	PJ6 (9) PF1 (9)	O	TTL	UART module 1 Request to Send modem flow control output line.
U1Rx	G1 H2 M2 L3 E12 A6	PD0 (5) PD2 (1) PC6 (5) PA0 (9) PB0 (5) PB4 (7)	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
U1Tx	G2 H1 L2 M3 D12 B7	PD1 (5) PD3 (1) PC7 (5) PA1 (9) PB1 (5) PB5 (7)	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
U2Rx	G1 K1 A6 C6	PD0 (4) PG0 (1) PB4 (4) PD5 (9)	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
U2Tx	B2 G2 K2 A3	PE4 (5) PD1 (4) PG1 (1) PD6 (9)	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.

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

13.3 Functional Description

Each Stellaris UART performs the functions of parallel-to-serial and serial-to-parallel conversions. It is similar in functionality to a 16C550 UART, but is not register compatible.

The UART is configured for transmit and/or receive via the TXE and RXE bits of the **UART Control (UARTCTL)** register (see page 706). Transmit and receive are both enabled out of reset. Before any control registers are programmed, the UART must be disabled by clearing the UARTEN bit in **UARTCTL**. If the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

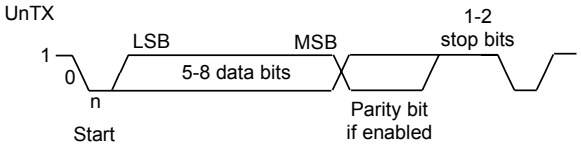
The UART module also includes a serial IR (SIR) encoder/decoder block that can be connected to an infrared transceiver to implement an IrDA SIR physical layer. The SIR function is programmed using the **UARTCTL** register.

13.3.1 Transmit/Receive Logic

The transmit logic performs parallel-to-serial conversion on the data read from the transmit FIFO. The control logic outputs the serial bit stream beginning with a start bit and followed by the data bits (LSB first), parity bit, and the stop bits according to the programmed configuration in the control registers. See Figure 13-2 on page 682 for details.

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

Figure 13-2. UART Character Frame



13.3.2 Baud-Rate Generation

The baud-rate divisor is a 22-bit number consisting of a 16-bit integer and a 6-bit fractional part. The number formed by these two values is used by the baud-rate generator to determine the bit period. Having a fractional baud-rate divisor 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 702) and the 6-bit fractional part is loaded with the **UART Fractional Baud-Rate Divisor (UARTFBRD)** register (see page 703). The baud-rate divisor (BRD) has the following relationship to the system clock (where *BRDI* is the integer part of the BRD and *BRDF* is the fractional part, separated by a decimal place.)

```
BRD = BRDI + BRDF = UARTSysClk / (ClkDiv * Baud Rate)
```

where UARTSysClk is the system clock connected to the UART, and ClkDiv is either 16 (if HSE in **UARTCTL** is clear) or 8 (if HSE is set).

The 6-bit fractional number (that is to be loaded into the DIVFRAC bit field in the **UARTFBRD** register) can be calculated by taking the fractional part of the baud-rate divisor, multiplying it by 64, and adding 0.5 to account for rounding errors:

```
UARTFBRD[DIVFRAC] = integer(BRDF * 64 + 0.5)
```

The UART generates an internal baud-rate reference clock at 8x or 16x the baud-rate (referred to as Baud8 and Baud16, depending on the setting of the HSE bit (bit 5) in **UARTCTL**). This reference clock is divided by 8 or 16 to generate the transmit clock, and is used for error detection during receive operations. Note that the state of the HSE bit has no effect on clock generation in ISO 7816 smart card mode (when the SMART bit in the **UARTCTL** register is set).

Along with the **UART Line Control, High Byte (UARTLCRH)** register (see page 704), the **UARTIBRD** and **UARTFBRD** registers form an internal 30-bit register. This internal register is only updated when a write operation to **UARTLCRH** is performed, so any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register for the changes to take effect.

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

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

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

13.3.3 Data Transmission

Data received or transmitted is stored in two 16-byte FIFOs, though the receive FIFO has an extra four bits per character for status information. For transmission, data is written into the transmit FIFO. If the UART is enabled, it causes a data frame to start transmitting with the parameters indicated in the **UARTLCRH** register. Data continues to be transmitted until there is no data left in the transmit FIFO. The BUSY bit in the **UART Flag (UARTFR)** register (see page 698) is asserted as soon as data is written to the transmit FIFO (that is, if the FIFO is non-empty) and remains asserted while data is being transmitted. The BUSY bit is negated only when the transmit FIFO is empty, and the last character has been transmitted from the shift register, including the stop bits. The UART can indicate that it is busy even though the UART may no longer be enabled.

When the receiver is idle (the UnRx signal is continuously 1), and the data input goes Low (a start bit has been received), the receive counter begins running and data is sampled on the eighth cycle of Baud16 or fourth cycle of Baud8 depending on the setting of the HSE bit (bit 5) in **UARTCTL** (described in "Transmit/Receive Logic" on page 682).

The start bit is valid and recognized if the UnRx signal is still low on the eighth cycle of Baud16 (HSE clear) or the fourth cycle of Baud 8 (HSE set), otherwise it is ignored. After a valid start bit is detected, successive data bits are sampled on every 16th cycle of Baud16 or 8th cycle of Baud8 (that is, one bit period later) according to the programmed length of the data characters and value of the HSE bit in **UARTCTL**. The parity bit is then checked if parity mode is enabled. Data length and parity are defined in the **UARTLCRH** register.

Lastly, a valid stop bit is confirmed if the UnRx signal is High, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO along with any error bits associated with that word.

13.3.4 Serial IR (SIR)

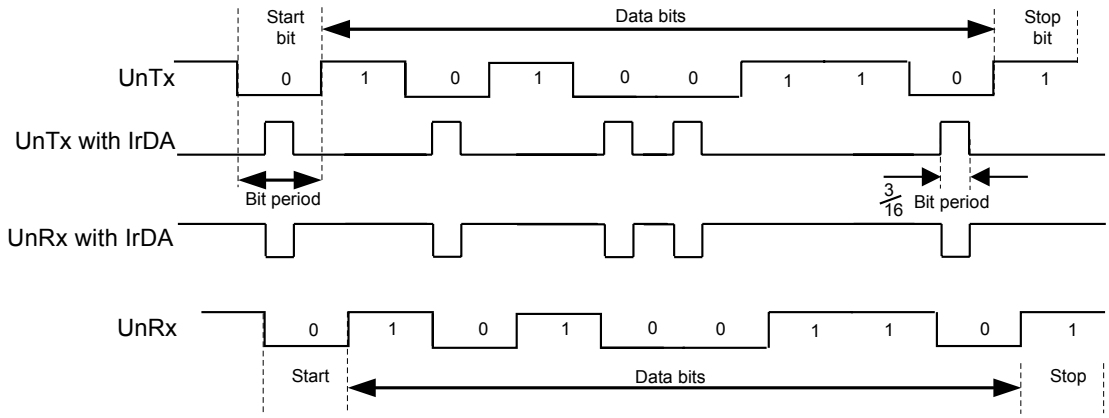
The UART peripheral includes an IrDA serial-IR (SIR) encoder/decoder block. The IrDA SIR block provides functionality that converts between an asynchronous UART data stream and a half-duplex serial SIR interface. No analog processing is performed on-chip. The role of the SIR block is to provide a digital encoded output and decoded input to the UART. When enabled, the SIR block uses the UnTx and UnRx pins for the SIR protocol. These signals should be connected to an infrared transceiver to implement an IrDA SIR physical layer link. The SIR block can receive and transmit, but it is only half-duplex so it cannot do both at the same time. Transmission must be stopped before data can be received. The IrDA SIR physical layer specifies a minimum 10-ms delay between transmission and reception.The SIR block has two modes of operation:

- In normal IrDA mode, a zero logic level is transmitted as a high pulse of 3/16th duration of the selected baud rate bit period on the output pin, while logic one levels are transmitted as a static LOW signal. These levels control the driver of an infrared transmitter, sending a pulse of light for each zero. On the reception side, the incoming light pulses energize the photo transistor base of the receiver, pulling its output LOW and driving the UART input pin LOW.
- In low-power IrDA mode, the width of the transmitted infrared pulse is set to three times the period of the internally generated IrLPBaud16 signal (1.63 µs, assuming a nominal 1.8432 MHz

frequency) by changing the appropriate bit in the **UARTCR** register. See page 701 for more information on IrDA low-power pulse-duration configuration.

Figure 13-3 on page 684 shows the UART transmit and receive signals, with and without IrDA modulation.

Figure 13-3. IrDA Data Modulation



In both normal and low-power IrDA modes:

- During transmission, the UART data bit is used as the base for encoding
- During reception, the decoded bits are transferred to the UART receive logic

The IrDA SIR physical layer specifies a half-duplex communication link, with a minimum 10-ms delay between transmission and reception. This delay must be generated by software because it is not automatically supported by the UART. The delay is required because the infrared receiver electronics might become biased or even saturated from the optical power coupled from the adjacent transmitter LED. This delay is known as latency or receiver setup time.

13.3.5 ISO 7816 Support

The UART offers basic support to allow communication with an ISO 7816 smartcard. When bit 3 (SMART) of the **UARTCTL** register is set, the UnTx signal is used as a bit clock, and the UnRx signal is used as the half-duplex communication line connected to the smartcard. A GPIO signal can be used to generate the reset signal to the smartcard. The remaining smartcard signals should be provided by the system design. The maximum clock rate in this mode is system clock / 16.

When using ISO 7816 mode, the **UARTLCRH** register must be set to transmit 8-bit words (WLEN bits 6:5 configured to 0x3) with EVEN parity (PEN set and EPS set). In this mode, the UART automatically uses 2 stop bits, and the STP2 bit of the **UARTLCRH** register is ignored.

If a parity error is detected during transmission, UnRx is pulled Low during the second stop bit. In this case, the UART aborts the transmission, flushes the transmit FIFO and discards any data it contains, and raises a parity error interrupt, allowing software to detect the problem and initiate retransmission of the affected data. Note that the UART does not support automatic retransmission in this case.

13.3.6 Modem Handshake Support

This section describes how to configure and use the modem flow control and status signals for UART1 when connected as a DTE (data terminal equipment) or as a DCE (data communications equipment). In general, a modem is a DCE and a computing device that connects to a modem is the DTE.

13.3.6.1 Signaling

The status signals provided by UART1 differ based on whether the UART is used as a DTE or DCE. When used as a DTE, the modem flow control and status signals are defined as:

- U1CTS is Clear To Send
- U1DSR is Data Set Ready
- U1DCD is Data Carrier Detect
- U1RI is Ring Indicator
- U1RTS is Request To Send
- U1DTR is Data Terminal Ready

When used as a DCE, the the modem flow control and status signals are defined as:

- U1CTS is Request To Send
- U1DSR is Data Terminal Ready
- U1RTS is Clear To Send
- U1DTR is Data Set Ready

Note that the support for DCE functions Data Carrier Detect and Ring Indicator are not provided. If these signals are required, their function can be emulated by using a general-purpose I/O signal and providing software support.

13.3.6.2 Flow Control

Flow control can be accomplished by either hardware or software. The following sections describe the different methods.

Hardware Flow Control (RTS/CTS)

Hardware flow control between two devices is accomplished by connecting the U1RTS output to the Clear-To-Send input on the receiving device, and connecting the Request-To-Send output on the receiving device to the U1CTS input.

The U1CTS input controls the transmitter. The transmitter may only transmit data when the U1CTS input is asserted. The U1RTS output signal indicates the state of the receive FIFO. U1CTS remains asserted until the preprogrammed watermark level is reached, indicating that the Receive FIFO has no space to store additional characters.

The **UARTCTL** register bits 15 (CTSEN) and 14 (RTSEN) specify the flow control mode as shown in Table 13-3 on page 686.

Table 13-3. Flow Control Mode

CTSEN	RTSEN	Description
1	1	RTS and CTS flow control enabled
1	0	Only CTS flow control enabled
0	1	Only RTS flow control enabled
0	0	Both RTS and CTS flow control disabled

Note that when RTSEN is 1, software cannot modify the $\overline{\text{U1RTS}}$ output value through the **UARTCTL** register Request to Send (RTS) bit, and the status of the RTS bit should be ignored.

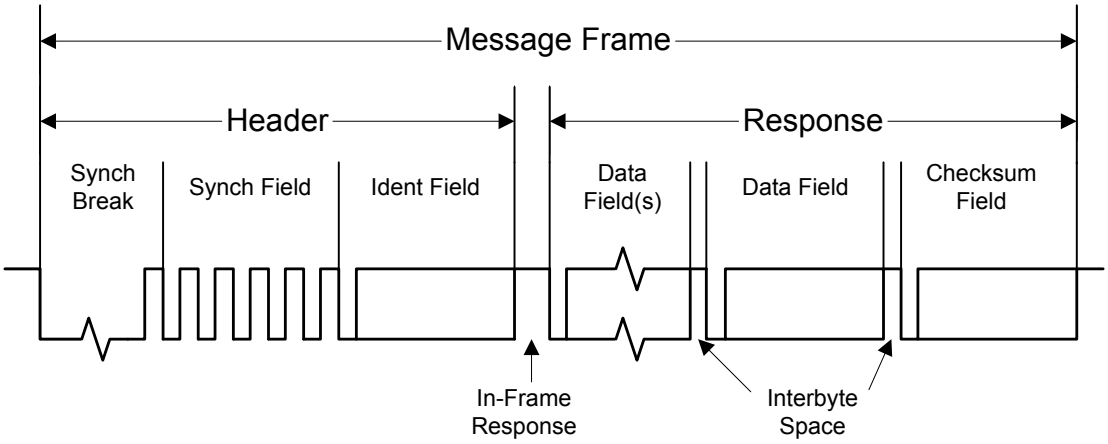
Software Flow Control (Modem Status Interrupts)

Software flow control between two devices is accomplished by using interrupts to indicate the status of the UART. Interrupts may be generated for the $\overline{\text{U1DSR}}$, $\overline{\text{U1DCD}}$, $\overline{\text{U1CTS}}$, and $\overline{\text{U1RI}}$ signals using bits 3:0 of the **UARTIM** register, respectively. The raw and masked interrupt status may be checked using the **UARTRIS** and **UARTMIS** register. These interrupts may be cleared using the **UARTICR** register.

13.3.7 LIN Support

The UART module offers hardware support for the LIN protocol as either a master or a slave. The LIN mode is enabled by setting the LIN bit in the **UARTCTL** register. A LIN message is identified by the use of a Sync Break at the beginning of the message. The Sync Break is a transmission of a series of 0s. The Sync Break is followed by the Sync data field (0x55). Figure 13-4 on page 686 illustrates the structure of a LIN message.

Figure 13-4. LIN Message



The UART should be configured as followed to operate in LIN mode:

1. Configure the UART for 1 start bit, 8 data bits, no parity, and 1 stop bit. Enable the Transmit FIFO.
2. Set the LIN bit in the **UARTCTL** register.

When preparing to send a LIN message, the TXFIFO should contain the Sync data (0x55) at FIFO location 0 and the Identifier data at location 1, followed by the data to be transmitted, and with the checksum in the final FIFO entry.

13.3.7.1 LIN Master

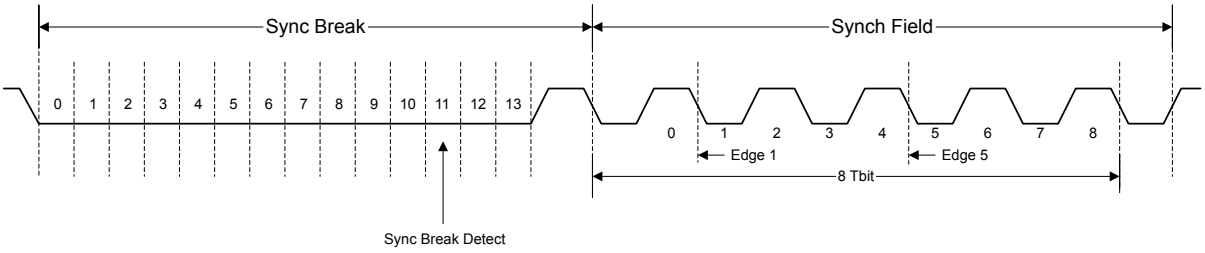
The UART is enabled to be the LIN master by setting the MASTER bit in the **UARTLCTL** register. The length of the Sync Break is programmable using the BLEN field in the **UARTLCTL** register and can be 13-16 bits (baud clock cycles).

13.3.7.2 LIN Slave

The LIN UART slave is required to adjust its baud rate to that of the LIN master. In slave mode, the LIN UART recognizes the Sync Break, which must be at least 13 bits in duration. A timer is provided to capture timing data on the 1st and 5th falling edges of the Sync field so that the baud rate can be adjusted to match the master.

After detecting a Sync Break, the UART waits for the synchronization field. The first falling edge generates an interrupt using the LME1RIS bit in the **UARTRIS** register, and the timer value is captured and stored in the **UARTLSS** register (T1). On the fifth falling edge, a second interrupt is generated using the LME5RIS bit in the **UARTRIS** register, and the timer value is captured again (T2). The actual baud rate can be calculated using (T2-T1)/8, and the local baud rate should be adjusted as needed. Figure 13-5 on page 687 illustrates the synchronization field.

Figure 13-5. LIN Synchronization Field



13.3.8 FIFO Operation

The UART has two 16x8 FIFOs; one for transmit and one for receive. Both FIFOs are accessed via the **UART Data (UARTDR)** register (see page 693). 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 704).

FIFO status can be monitored via the **UART Flag (UARTFR)** register (see page 698) 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. If the FIFOs are disabled, the empty and full flags are set according to the status of the 1-byte-deep holding registers.

The trigger points at which the FIFOs generate interrupts is controlled via the **UART Interrupt FIFO Level Select (UARTIFLS)** register (see page 710). Both FIFOs can be individually configured to trigger interrupts at different levels. Available configurations include ⅛, ¼, ½, ¾, and ⅞. For example,

if the ¼ option is selected for the receive FIFO, the UART generates a receive interrupt after 4 data bytes are received. Out of reset, both FIFOs are configured to trigger an interrupt at the ½ mark.

13.3.9 Interrupts

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

- Overrun Error
- Break Error
- Parity Error
- Framing Error
- Receive Timeout
- Transmit (when condition defined in the TXIFLSEL bit in the **UARTIFLS** register is met, or if the EOT bit in **UARTCTL** is set, when the last bit of all transmitted data leaves the serializer)
- Receive (when condition defined in the RXIFLSEL bit in the **UARTIFLS** register is met)

All of the interrupt events are ORed together before being sent to the interrupt controller, so the UART can only generate a single interrupt request to the controller at any given time. Software can service multiple interrupt events in a single interrupt service routine by reading the **UART Masked Interrupt Status (UARTMIS)** register (see page 720).

The interrupt events that can trigger a controller-level interrupt are defined in the **UART Interrupt Mask (UARTIM)** register (see page 712) by setting the corresponding IM bits. If interrupts are not used, the raw interrupt status is always visible via the **UART Raw Interrupt Status (UARTRIS)** register (see page 716).

Interrupts are always cleared (for both the **UARTMIS** and **UARTRIS** registers) by writing a 1 to the corresponding bit in the **UART Interrupt Clear (UARTICR)** register (see page 724).

The receive timeout interrupt is asserted when the receive FIFO is not empty, and no further data is received over a 32-bit period. The receive timeout interrupt is cleared either when the FIFO becomes empty through reading all the data (or by reading the holding register), or when a 1 is written to the corresponding bit in the **UARTICR** register.

The receive interrupt changes state when one of the following events occurs:

- If the FIFOs are enabled and the receive FIFO reaches the programmed trigger level, the RXRIS bit is set. The receive interrupt is cleared by reading data from the receive FIFO until it becomes less than the trigger level, or by clearing the interrupt by writing a 1 to the RXIC bit.
- If the FIFOs are disabled (have a depth of one location) and data is received thereby filling the location, the RXRIS bit is set. The receive interrupt is cleared by performing a single read of the receive FIFO, or by clearing the interrupt by writing a 1 to the RXIC bit.

The transmit interrupt changes state when one of the following events occurs:

- If the FIFOs are enabled and the transmit FIFO reaches the programmed trigger level, the TXRIS bit is set. The transmit interrupt is cleared by writing data to the transmit FIFO until it becomes greater than the trigger level, or by clearing the interrupt by writing a 1 to the TXIC bit.

- If the FIFOs are disabled (have a depth of one location) and there is no data present in the transmitters single location, the TXRIS bit is set. It is cleared by performing a single write to the transmit FIFO, or by clearing the interrupt by writing a 1 to the TXIC bit.

13.3.10 Loopback Operation

The UART can be placed into an internal loopback mode for diagnostic or debug work by setting the LBE bit in the **UARTCTL** register (see page 706). In loopback mode, data transmitted on the UnTx output is received on the UnRx input. Note that the LBE bit should be set before the UART is enabled.

13.3.11 DMA Operation

The UART provides an interface to the µDMA controller with separate channels for transmit and receive. The DMA operation of the UART is enabled through the **UART DMA Control (UARTDMACTL)** register. When DMA operation is enabled, the UART asserts a DMA request on the receive or transmit channel when the associated FIFO can transfer data. For the receive channel, a single transfer request is asserted whenever any data is in the receive FIFO. A burst transfer request is asserted whenever the amount of data in the receive FIFO is at or above the FIFO trigger level configured in the **UARTIFLS** register. For the transmit channel, a single transfer request is asserted whenever there is at least one empty location in the transmit FIFO. The burst request is asserted whenever the transmit FIFO contains fewer characters than the FIFO trigger level. The single and burst DMA transfer requests are handled automatically by the µDMA controller depending on how the DMA channel is configured.

To enable DMA operation for the receive channel, set the RXDMAE bit of the **DMA Control (UARTDMACTL)** register. To enable DMA operation for the transmit channel, set the TXDMAE bit of the **UARTDMACTL** register. The UART can also be configured to stop using DMA for the receive channel if a receive error occurs. If the DMAERR bit of the **UARTDMACR** register is set and a receive error occurs, the DMA receive requests are automatically disabled. This error condition can be cleared by clearing the appropriate UART error interrupt.

If DMA is enabled, then the µDMA controller triggers an interrupt when a transfer is complete. The interrupt occurs on the UART interrupt vector. Therefore, if interrupts are used for UART operation and DMA is enabled, the UART interrupt handler must be designed to handle the µDMA completion interrupt.

See “Micro Direct Memory Access (µDMA)” on page 334 for more details about programming the µDMA controller.

13.4 Initialization and Configuration

To enable and initialize the UART, the following steps are necessary:

1. The peripheral clock must be enabled by setting the UART0, UART1, or UART2 bits in the **RCGC1** register (see page 263).
2. The clock to the appropriate GPIO module must be enabled via the **RCGC2** register in the System Control module (see page 272).
3. Set the GPIO AFSEL bits for the appropriate pins (see page 419). To determine which GPIOs to configure, see Table 22-4 on page 1144.
4. Configure the GPIO current level and/or slew rate as specified for the mode selected (see page 421 and page 429).

5. Configure the PMCn fields in the **GPIOPCTL** register to assign the UART signals to the appropriate pins (see page 437 and Table 22-5 on page 1151).

To use the UART, the peripheral clock must be enabled by setting the appropriate bit in the **RCGC1** register (page 263). In addition, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register (page 272) in the System Control module. To find out which GPIO port to enable, refer to Table 22-5 on page 1151.

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

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

The first thing to consider when programming the UART is the baud-rate divisor (BRD), because the **UARTIBRD** and **UARTFBRD** registers must be written before the **UARTLCRH** register. Using the equation described in “Baud-Rate Generation” on page 682, 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 702) should be set to 10 decimal or 0xA. The value to be loaded into the **UARTFBRD** register (see page 703) is calculated by the equation:

```
UARTFBRD[DIVFRAC] = integer(0.8507 * 64 + 0.5) = 54
```

With the BRD values in hand, the UART configuration is written to the module in the following order:

1. Disable the UART by clearing the UARTEN bit in the **UARTCTL** register.
2. Write the integer portion of the BRD to the **UARTIBRD** register.
3. Write the fractional portion of the BRD to the **UARTFBRD** register.
4. Write the desired serial parameters to the **UARTLCRH** register (in this case, a value of 0x0000.0060).
5. Optionally, configure the µDMA channel (see “Micro Direct Memory Access (µDMA)” on page 334) and enable the DMA option(s) in the **UARTDMACTL** register.
6. Enable the UART by setting the UARTEN bit in the **UARTCTL** register.

13.5 Register Map

Table 13-4 on page 691 lists the UART registers. The offset listed is a hexadecimal increment to the register’s address, relative to that UART’s base address:

- UART0: 0x4000.C000

- UART1: 0x4000.D000
- UART2: 0x4000.E000

Note that the UART module clock must be enabled before the registers can be programmed (see page 263). There must be a delay of 3 system clocks after the UART module clock is enabled before any UART module registers are accessed.

Note: The UART must be disabled (see the UARTEN bit in the **UARTCTL** register on page 706) before any of the control registers are reprogrammed. When the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

Table 13-4. UART Register Map

Offset	Name	Type	Reset	Description	See page
0x000	UARTDR	R/W	0x0000.0000	UART Data	693
0x004	UARTRSR/UARTECR	R/W	0x0000.0000	UART Receive Status/Error Clear	695
0x018	UARTFR	RO	0x0000.0090	UART Flag	698
0x020	UARTILPR	R/W	0x0000.0000	UART IrDA Low-Power Register	701
0x024	UARTIBRD	R/W	0x0000.0000	UART Integer Baud-Rate Divisor	702
0x028	UARTFBRD	R/W	0x0000.0000	UART Fractional Baud-Rate Divisor	703
0x02C	UARTLCRH	R/W	0x0000.0000	UART Line Control	704
0x030	UARTCTL	R/W	0x0000.0300	UART Control	706
0x034	UARTIFLS	R/W	0x0000.0012	UART Interrupt FIFO Level Select	710
0x038	UARTIM	R/W	0x0000.0000	UART Interrupt Mask	712
0x03C	UARTRIS	RO	0x0000.000F	UART Raw Interrupt Status	716
0x040	UARTMIS	RO	0x0000.0000	UART Masked Interrupt Status	720
0x044	UARTICR	W1C	0x0000.0000	UART Interrupt Clear	724
0x048	UARTDMACTL	R/W	0x0000.0000	UART DMA Control	726
0x090	UARTLCTL	R/W	0x0000.0000	UART LIN Control	727
0x094	UARTLSS	RO	0x0000.0000	UART LIN Snap Shot	728
0x098	UARTLTIM	RO	0x0000.0000	UART LIN Timer	729
0xFD0	UARTPeriphID4	RO	0x0000.0000	UART Peripheral Identification 4	730
0xFD4	UARTPeriphID5	RO	0x0000.0000	UART Peripheral Identification 5	731
0xFD8	UARTPeriphID6	RO	0x0000.0000	UART Peripheral Identification 6	732
0xFDC	UARTPeriphID7	RO	0x0000.0000	UART Peripheral Identification 7	733
0xFE0	UARTPeriphID0	RO	0x0000.0060	UART Peripheral Identification 0	734
0xFE4	UARTPeriphID1	RO	0x0000.0000	UART Peripheral Identification 1	735
0xFE8	UARTPeriphID2	RO	0x0000.0018	UART Peripheral Identification 2	736
0xFEC	UARTPeriphID3	RO	0x0000.0001	UART Peripheral Identification 3	737

Table 13-4. UART Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0xFF0	UARTPCellID0	RO	0x0000.000D	UART PrimeCell Identification 0	738
0xFF4	UARTPCellID1	RO	0x0000.00F0	UART PrimeCell Identification 1	739
0xFF8	UARTPCellID2	RO	0x0000.0005	UART PrimeCell Identification 2	740
0xFFC	UARTPCellID3	RO	0x0000.00B1	UART PrimeCell Identification 3	741

13.6 Register Descriptions

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

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

Important: This register is read-sensitive. See the register description for details.

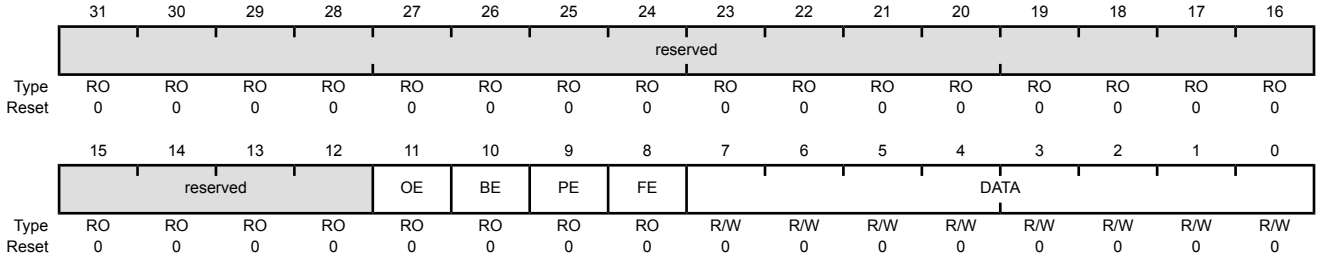
This register is the data register (the interface to the FIFOs).

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

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

UART Data (UARTDR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x000
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	OE	RO	0	UART Overrun Error Value Description 1 New data was received when the FIFO was full, resulting in data loss. 0 No data has been lost due to a FIFO overrun.
10	BE	RO	0	UART Break Error Value Description 1 A break condition has been detected, indicating that the receive data input was held Low for longer than a full-word transmission time (defined as start, data, parity, and stop bits). 0 No break condition has occurred In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the received data input goes to a 1 (marking state), and the next valid start bit is received.

Bit/Field	Name	Type	Reset	Description
9	PE	RO	0	UART Parity Error Value Description 1 The parity of the received data character does not match the parity defined by bits 2 and 7 of the **UARTLCRH** register. 0 No parity error has occurred In FIFO mode, this error is associated with the character at the top of the FIFO.
8	FE	RO	0	UART Framing Error Value Description 1 The received character does not have a valid stop bit (a valid stop bit is 1). 0 No framing error has occurred
7:0	DATA	R/W	0x00	Data Transmitted or Received Data that is to be transmitted via the UART is written to this field. When read, this field contains the data that was received by the UART.

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

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

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

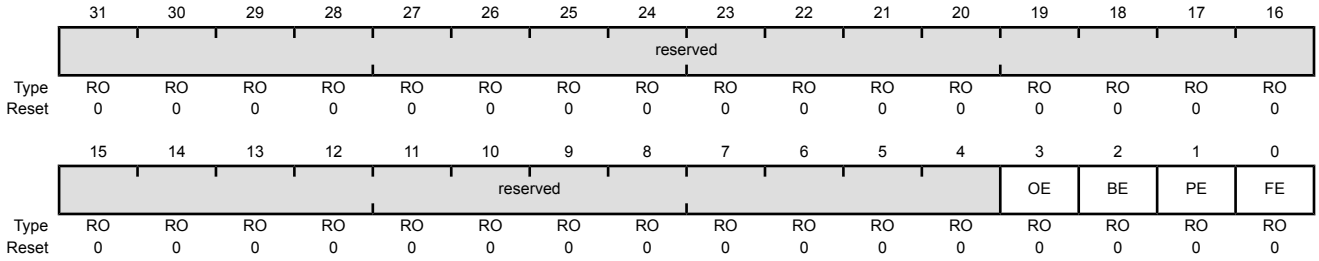
The **UARTRSR** register cannot be written.

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

Read-Only Status Register

UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x004
Type RO, reset 0x0000.0000



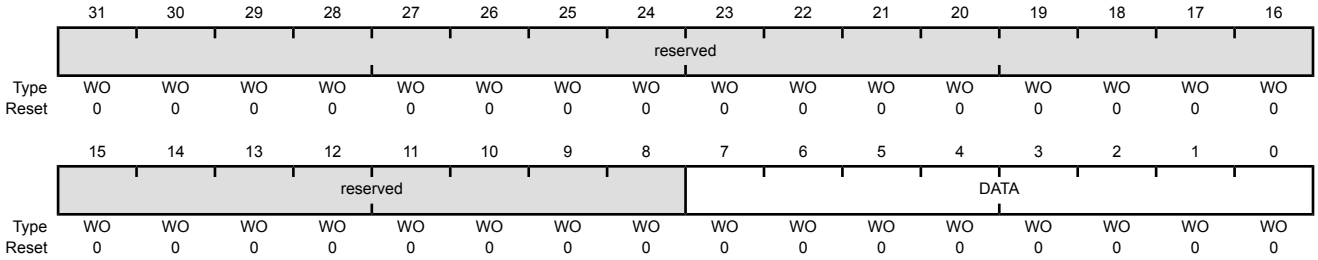
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	OE	RO	0	UART Overrun Error Value Description 1 New data was received when the FIFO was full, resulting in data loss. 0 No data has been lost due to a FIFO overrun. This bit is cleared by a write to **UARTECR**. The FIFO contents remain valid because no further data is written when the FIFO is full, only the contents of the shift register are overwritten. The CPU must read the data in order to empty the FIFO.

Bit/Field	Name	Type	Reset	Description
2	BE	RO	0	UART Break Error Value Description 1 A break condition has been detected, indicating that the receive data input was held Low for longer than a full-word transmission time (defined as start, data, parity, and stop bits). 0 No break condition has occurred This bit is cleared to 0 by a write to **UARTECR**. In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state) and the next valid start bit is received.
1	PE	RO	0	UART Parity Error Value Description 1 The parity of the received data character does not match the parity defined by bits 2 and 7 of the **UARTLCRH** register. 0 No parity error has occurred This bit is cleared to 0 by a write to **UARTECR**.
0	FE	RO	0	UART Framing Error Value Description 1 The received character does not have a valid stop bit (a valid stop bit is 1). 0 No framing error has occurred This bit is cleared to 0 by a write to **UARTECR**. In FIFO mode, this error is associated with the character at the top of the FIFO.

Write-Only Error Clear Register

UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x004
Type WO, reset 0x0000.0000



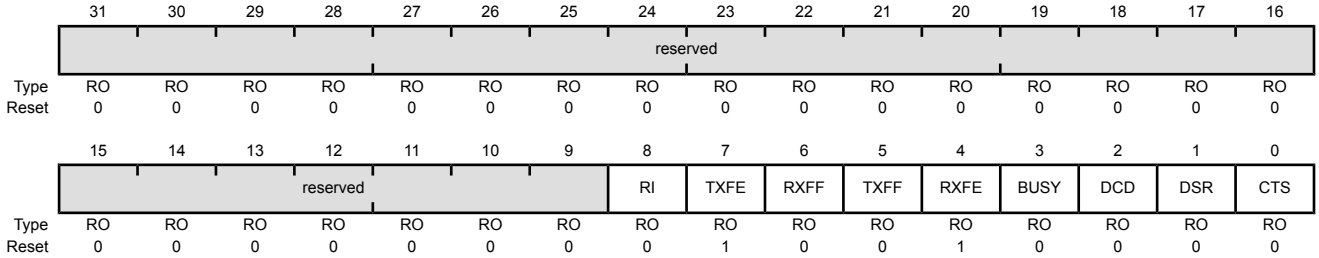
Bit/Field	Name	Type	Reset	Description
31:8	reserved	WO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	WO	0x00	Error Clear A write to this register of any data clears the framing, parity, break, and overrun flags.

Register 3: UART Flag (UARTFR), offset 0x018

The **UARTFR** register is the flag register. After reset, the TXFF, RXFF, and BUSY bits are 0, and TXFE and RXFE bits are 1. The RI, DCD, DSR and CTS bits indicate the modem flow control and status. Note that the modem bits are only implemented on UART1 and are reserved on UART0 and UART2.

UART Flag (UARTFR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x018
Type RO, reset 0x0000.0090



Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	RI	RO	0	Ring Indicator Value Description 1 The U1RI signal is asserted. 0 The U1RI signal is not asserted. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
7	TXFE	RO	1	UART Transmit FIFO Empty The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register. Value Description 1 If the FIFO is disabled (FEN is 0), the transmit holding register is empty. If the FIFO is enabled (FEN is 1), the transmit FIFO is empty. 0 The transmitter has data to transmit.

Bit/Field	Name	Type	Reset	Description
6	RXFF	RO	0	UART Receive FIFO Full The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register. Value Description 1 If the FIFO is disabled (FEN is 0), the receive holding register is full. If the FIFO is enabled (FEN is 1), the receive FIFO is full. 0 The receiver can receive data.
5	TXFF	RO	0	UART Transmit FIFO Full The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register. Value Description 1 If the FIFO is disabled (FEN is 0), the transmit holding register is full. If the FIFO is enabled (FEN is 1), the transmit FIFO is full. 0 The transmitter is not full.
4	RXFE	RO	1	UART Receive FIFO Empty The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register. Value Description 1 If the FIFO is disabled (FEN is 0), the receive holding register is empty. If the FIFO is enabled (FEN is 1), the receive FIFO is empty. 0 The receiver is not empty.
3	BUSY	RO	0	UART Busy Value Description 1 The UART is busy transmitting data. This bit remains set until the complete byte, including all stop bits, has been sent from the shift register. 0 The UART is not busy. This bit is set as soon as the transmit FIFO becomes non-empty (regardless of whether UART is enabled).
2	DCD	RO	0	Data Carrier Detect Value Description 1 The U1DCD signal is asserted. 0 The U1DCD signal is not asserted. This bit is implemented only on UART1 and is reserved for UART0 and UART2.

Bit/Field	Name	Type	Reset	Description
1	DSR	RO	0	Data Set Ready Value Description 1 The U1DSR signal is asserted. 0 The U1DSR signal is not asserted. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
0	CTS	RO	0	Clear To Send Value Description 1 The U1CTS signal is asserted. 0 The U1CTS signal is not asserted. This bit is implemented only on UART1 and is reserved for UART0 and UART2.

Register 4: UART IrDA Low-Power Register (UARTILPR), offset 0x020

The **UARTILPR** register stores the 8-bit low-power counter divisor value used to derive the low-power SIR pulse width clock by dividing down the system clock (SysClk). All the bits are cleared when reset.

The internal IrLPBaud16 clock is generated by dividing down SysClk according to the low-power divisor value written to **UARTILPR**. The duration of SIR pulses generated when low-power mode is enabled is three times the period of the IrLPBaud16 clock. The low-power divisor value is calculated as follows:

$$ILPDVSR = SysClk / F_{IrLPBaud16}$$

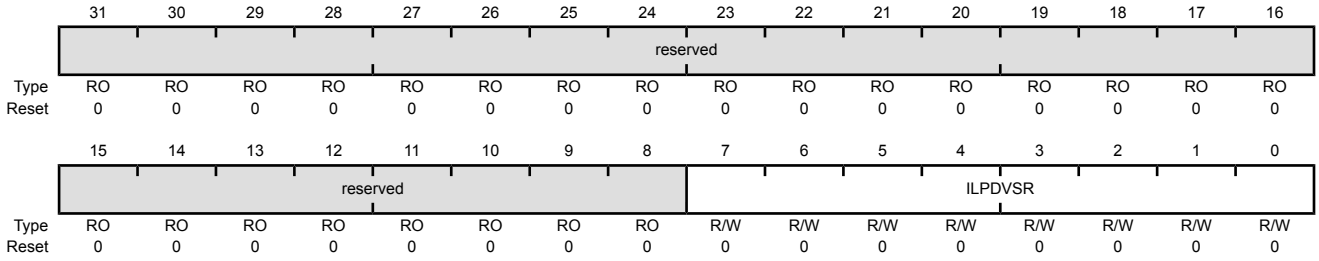
where $F_{IrLPBaud16}$ is nominally 1.8432 MHz.

The divisor must be programmed such that 1.42 MHz < $F_{IrLPBaud16}$ < 2.12 MHz, resulting in a low-power pulse duration of 1.41–2.11 µs (three times the period of IrLPBaud16). The minimum frequency of IrLPBaud16 ensures that pulses less than one period of IrLPBaud16 are rejected, but pulses greater than 1.4 µs are accepted as valid pulses.

Note: Zero is an illegal value. Programming a zero value results in no IrLPBaud16 pulses being generated.

UART IrDA Low-Power Register (UARTILPR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x020
Type R/W, reset 0x0000.0000



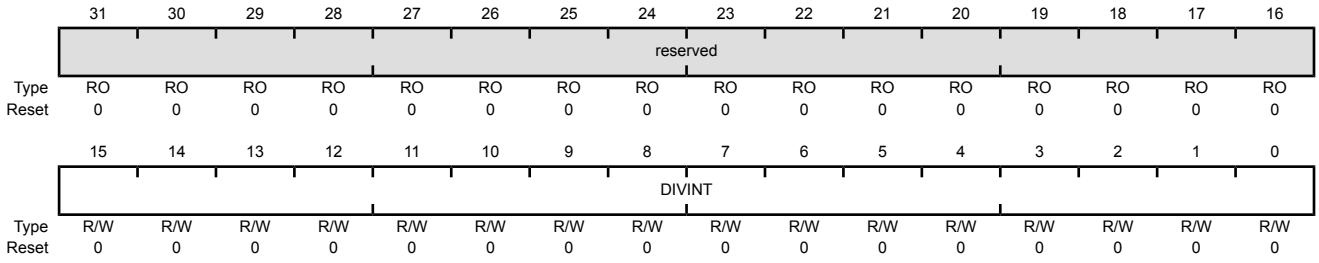
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ILPDVSR	R/W	0x00	IrDA Low-Power Divisor This field contains the 8-bit low-power divisor value.

Register 5: UART Integer Baud-Rate Divisor (UARTIBRD), offset 0x024

The **UARTIBRD** register is the integer part of the baud-rate divisor value. All the bits are cleared on reset. The minimum possible divide ratio is 1 (when **UARTIBRD**=0), in which case the **UARTFBRD** register is ignored. When changing the **UARTIBRD** register, the new value does not take effect until transmission/reception of the current character is complete. Any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register. See “Baud-Rate Generation” on page 682 for configuration details.

UART Integer Baud-Rate Divisor (UARTIBRD)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x024
Type R/W, reset 0x0000.0000



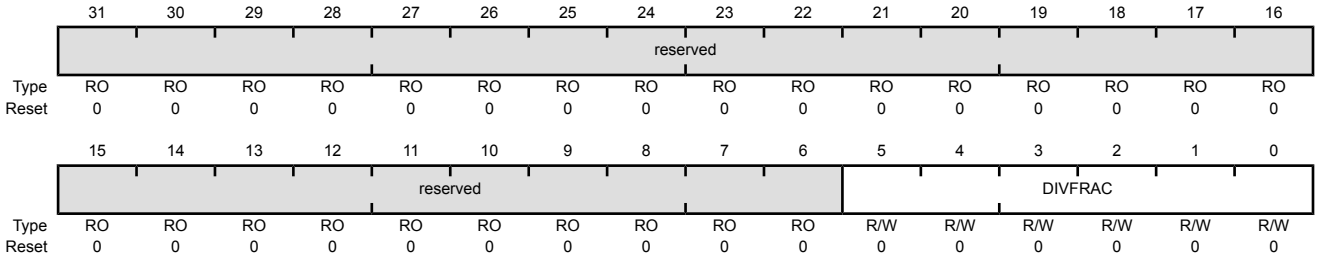
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	DIVINT	R/W	0x0000	Integer Baud-Rate Divisor

Register 6: UART Fractional Baud-Rate Divisor (UARTFBRD), offset 0x028

The **UARTFBRD** register is the fractional part of the baud-rate divisor value. All the bits are cleared on reset. When changing the **UARTFBRD** register, the new value does not take effect until transmission/reception of the current character is complete. Any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register. See “Baud-Rate Generation” on page 682 for configuration details.

UART Fractional Baud-Rate Divisor (UARTFBRD)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x028
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	DIVFRAC	R/W	0x0	Fractional Baud-Rate Divisor

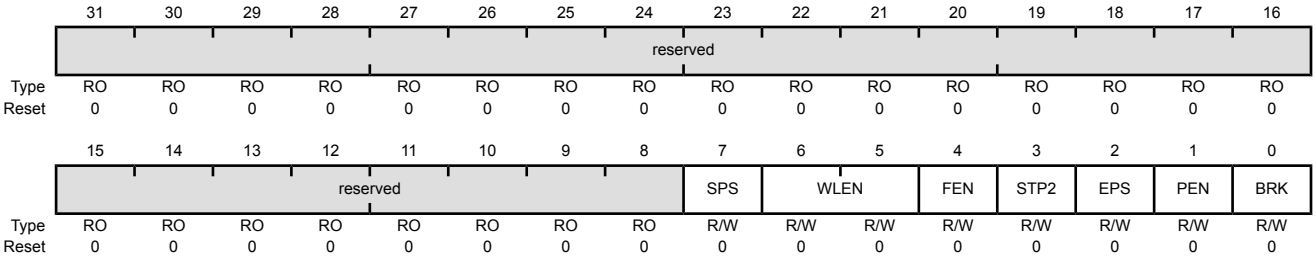
Register 7: UART Line Control (UARTLCRH), offset 0x02C

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

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

UART Line Control (UARTLCRH)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x02C
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	SPS	R/W	0	UART Stick Parity Select When bits 1, 2, and 7 of **UARTLCRH** are set, the parity bit is transmitted and checked as a 0. When bits 1 and 7 are set and 2 is cleared, the parity bit is transmitted and checked as a 1. When this bit is cleared, stick parity is disabled.
6:5	WLEN	R/W	0x0	UART Word Length The bits indicate the number of data bits transmitted or received in a frame as follows: Value Description 0x0 5 bits (default) 0x1 6 bits 0x2 7 bits 0x3 8 bits
4	FEN	R/W	0	UART Enable FIFOs Value Description 1 The transmit and receive FIFO buffers are enabled (FIFO mode). 0 The FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers.

Bit/Field	Name	Type	Reset	Description
3	STP2	R/W	0	UART Two Stop Bits Select Value Description 1 Two stop bits are transmitted at the end of a frame. The receive logic does not check for two stop bits being received. When in 7816 smartcard mode (the SMART bit is set in the **UARTCTL** register), the number of stop bits is forced to 2. 0 One stop bit is transmitted at the end of a frame.
2	EPS	R/W	0	UART Even Parity Select Value Description 1 Even parity generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and parity bits. 0 Odd parity is performed, which checks for an odd number of 1s. This bit has no effect when parity is disabled by the PEN bit.
1	PEN	R/W	0	UART Parity Enable Value Description 1 Parity checking and generation is enabled. 0 Parity is disabled and no parity bit is added to the data frame.
0	BRK	R/W	0	UART Send Break Value Description 1 A Low level is continually output on the UnTx signal, after completing transmission of the current character. For the proper execution of the break command, software must set this bit for at least two frames (character periods). 0 Normal use.

Register 8: UART Control (UARTCTL), offset 0x030

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

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

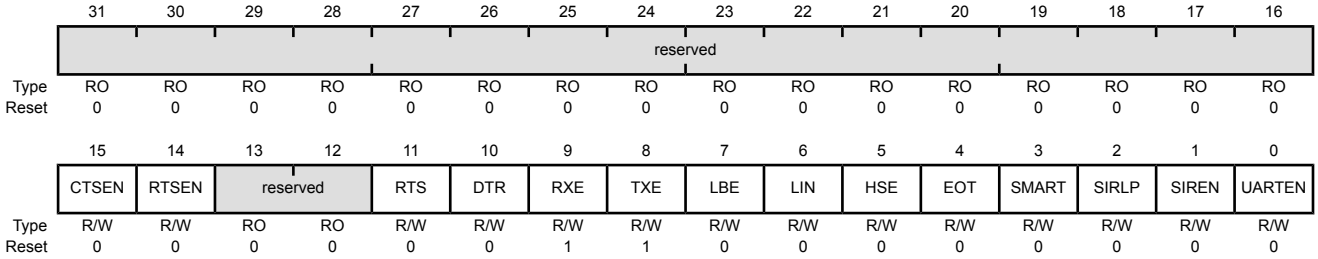
Note that bits [15:14,11:10] are only implemented on UART1. These bits are reserved on UART0 and UART2.

Note: The **UARTCTL** register should not be changed while the UART is enabled or else the results are unpredictable. The following sequence is recommended for making changes to the **UARTCTL** register.

1. Disable the UART.
2. Wait for the end of transmission or reception of the current character.
3. Flush the transmit FIFO by clearing bit 4 (FEN) in the line control register (**UARTLCRH**).
4. Reprogram the control register.
5. Enable the UART.

UART Control (UARTCTL)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x030
Type R/W, reset 0x0000.0300



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
15	CTSEN	R/W	0	Enable Clear To Send Value Description 1 CTS hardware flow control is enabled. Data is only transmitted when the U1CTS signal is asserted. 0 CTS hardware flow control is disabled. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
14	RTSEN	R/W	0	Enable Request to Send Value Description 1 RTS hardware flow control is enabled. Data is only requested (by asserting U1RTS) when the receive FIFO has available entries. 0 RTS hardware flow control is disabled. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
13:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	RTS	R/W	0	Request to Send When RTSEN is clear, the status of this bit is reflected on the U1RTS signal. If RTSEN is set, this bit is ignored on a write and should be ignored on read. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
10	DTR	R/W	0	Data Terminal Ready This bit sets the state of the U1DTR output. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
9	RXE	R/W	1	UART Receive Enable Value Description 1 The receive section of the UART is enabled. 0 The receive section of the UART is disabled. If the UART is disabled in the middle of a receive, it completes the current character before stopping. **Note:** To enable reception, the UARTEN bit must also be set.

Bit/Field	Name	Type	Reset	Description
8	TXE	R/W	1	UART Transmit Enable Value Description 1 The transmit section of the UART is enabled. 0 The transmit section of the UART is disabled. If the UART is disabled in the middle of a transmission, it completes the current character before stopping. **Note:** To enable transmission, the UARTEN bit must also be set.
7	LBE	R/W	0	UART Loop Back Enable Value Description 1 The UnTx path is fed through the UnRx path. 0 Normal operation.
6	LIN	R/W	0	LIN Mode Enable Value Description 1 The UART operates in LIN mode. 0 Normal operation.
5	HSE	R/W	0	High-Speed Enable Value Description 0 The UART is clocked using the system clock divided by 16. 1 The UART is clocked using the system clock divided by 8. **Note:** System clock used is also dependent on the baud-rate divisor configuration (see page 702) and page 703). The state of this bit has no effect on clock generation in ISO 7816 smart card mode (the SMART bit is set).
4	EOT	R/W	0	End of Transmission This bit determines the behavior of the TXRIS bit in the **UARTRIS** register. Value Description 1 The TXRIS bit is set only after all transmitted data, including stop bits, have cleared the serializer. 0 The TXRIS bit is set when the transmit FIFO condition specified in **UARTIFLS** is met.

Bit/Field	Name	Type	Reset	Description
3	SMART	R/W	0	ISO 7816 Smart Card Support Value Description 1 The UART operates in Smart Card mode. 0 Normal operation. The application must ensure that it sets 8-bit word length (WLEN set to 0x3) and even parity (PEN set to 1, EPS set to 1, SPS set to 0) in **UARTLCRH** when using ISO 7816 mode. In this mode, the value of the STP2 bit in **UARTLCRH** is ignored and the number of stop bits is forced to 2. Note that the UART does not support automatic retransmission on parity errors. If a parity error is detected on transmission, all further transmit operations are aborted and software must handle retransmission of the affected byte or message.
2	SIRLP	R/W	0	UART SIR Low-Power Mode This bit selects the IrDA encoding mode. Value Description 1 The UART operates in SIR Low-Power mode. Low-level bits are transmitted with a pulse width which is 3 times the period of the IrLPBaud16 input signal, regardless of the selected bit rate. 0 Low-level bits are transmitted as an active High pulse with a width of 3/16th of the bit period. Setting this bit uses less power, but might reduce transmission distances. See page 701 for more information.
1	SIREN	R/W	0	UART SIR Enable Value Description 1 The IrDA SIR block is enabled, and the UART will transmit and receive data using SIR protocol. 0 Normal operation.
0	UARTEN	R/W	0	UART Enable Value Description 1 The UART is enabled. 0 The UART is disabled. If the UART is disabled in the middle of transmission or reception, it completes the current character before stopping.

Register 9: UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034

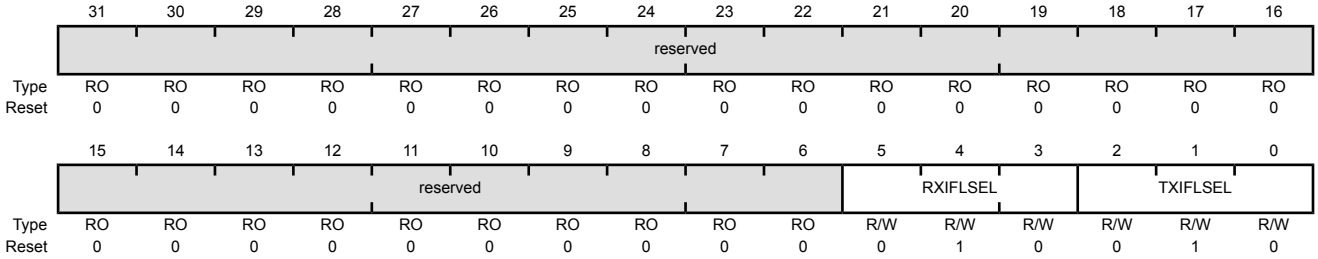
The **UARTIFLS** register is the interrupt FIFO level select register. You can use this register to define the FIFO level at which the TXRIS and RXRIS bits in the **UARTRIS** register are triggered.

The interrupts are generated based on a transition through a level rather than being based on the level. That is, the interrupts are generated when the fill level progresses through the trigger level. For example, if the receive trigger level is set to the half-way mark, the interrupt is triggered as the module is receiving the 9th character.

Out of reset, the TXIFLSEL and RXIFLSEL bits are configured so that the FIFOs trigger an interrupt at the half-way mark.

UART Interrupt FIFO Level Select (UARTIFLS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x034
Type R/W, reset 0x0000.0012



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:3	RXIFLSEL	R/W	0x2	UART Receive Interrupt FIFO Level Select The trigger points for the receive interrupt are as follows: Value Description 0x0 RX FIFO ≥ ⅛ full 0x1 RX FIFO ≥ ¼ full 0x2 RX FIFO ≥ ½ full (default) 0x3 RX FIFO ≥ ¾ full 0x4 RX FIFO ≥ ⅞ full 0x5-0x7 Reserved

Bit/Field	Name	Type	Reset	Description
2:0	TXIFLSEL	R/W	0x2	UART Transmit Interrupt FIFO Level Select The trigger points for the transmit interrupt are as follows:

Value	Description
0x0	TX FIFO ≤ ⅞ empty
0x1	TX FIFO ≤ ¾ empty
0x2	TX FIFO ≤ ½ empty (default)
0x3	TX FIFO ≤ ¼ empty
0x4	TX FIFO ≤ ⅛ empty
0x5-0x7	Reserved

Note: If the EOT bit in **UARTCTL** is set (see page 706), the transmit interrupt is generated once the FIFO is completely empty and all data including stop bits have left the transmit serializer. In this case, the setting of TXIFLSEL is ignored.

Register 10: UART Interrupt Mask (UARTIM), offset 0x038

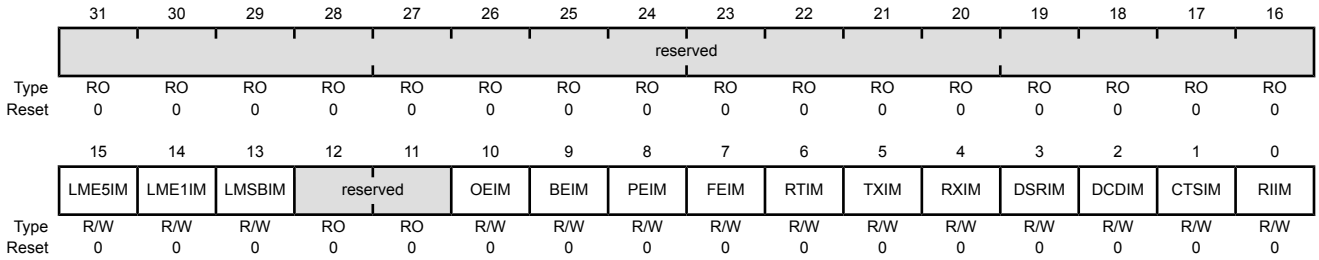
The **UARTIM** register is the interrupt mask set/clear register.

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

Note that bits [3:0] are only implemented on UART1. These bits are reserved on UART0 and UART2.

UART Interrupt Mask (UARTIM)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x038
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	LME5IM	R/W	0	LIN Mode Edge 5 Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the LME5RIS bit in the **UARTRIS** register is set. 0 The LME5RIS interrupt is suppressed and not sent to the interrupt controller.
14	LME1IM	R/W	0	LIN Mode Edge 1 Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the LME1RIS bit in the **UARTRIS** register is set. 0 The LME1RIS interrupt is suppressed and not sent to the interrupt controller.
13	LMSBIM	R/W	0	LIN Mode Sync Break Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the LMSBRIS bit in the **UARTRIS** register is set. 0 The LMSBRIS interrupt is suppressed and not sent to the interrupt controller.

Bit/Field	Name	Type	Reset	Description
12:11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEIM	R/W	0	UART Overrun Error Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the OERIS bit in the **UARTRIS** register is set. 0 The OERIS interrupt is suppressed and not sent to the interrupt controller.
9	BEIM	R/W	0	UART Break Error Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the BERIS bit in the **UARTRIS** register is set. 0 The BERIS interrupt is suppressed and not sent to the interrupt controller.
8	PEIM	R/W	0	UART Parity Error Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the PERIS bit in the **UARTRIS** register is set. 0 The PERIS interrupt is suppressed and not sent to the interrupt controller.
7	FEIM	R/W	0	UART Framing Error Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the FERIS bit in the **UARTRIS** register is set. 0 The FERIS interrupt is suppressed and not sent to the interrupt controller.
6	RTIM	R/W	0	UART Receive Time-Out Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the RTRIS bit in the **UARTRIS** register is set. 0 The RTRIS interrupt is suppressed and not sent to the interrupt controller.

Bit/Field	Name	Type	Reset	Description
5	TXIM	R/W	0	UART Transmit Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the TXRIS bit in the **UARTRIS** register is set. 0 The TXRIS interrupt is suppressed and not sent to the interrupt controller.
4	RXIM	R/W	0	UART Receive Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the RXRIS bit in the **UARTRIS** register is set. 0 The RXRIS interrupt is suppressed and not sent to the interrupt controller.
3	DSRIM	R/W	0	UART Data Set Ready Modem Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the DSRRIS bit in the **UARTRIS** register is set. 0 The DSRRIS interrupt is suppressed and not sent to the interrupt controller. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
2	DCDIM	R/W	0	UART Data Carrier Detect Modem Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the DCDRIS bit in the **UARTRIS** register is set. 0 The DCDRIS interrupt is suppressed and not sent to the interrupt controller. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
1	CTSIM	R/W	0	UART Clear to Send Modem Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the CTSRIS bit in the **UARTRIS** register is set. 0 The CTSRIS interrupt is suppressed and not sent to the interrupt controller. This bit is implemented only on UART1 and is reserved for UART0 and UART2.

Bit/Field	Name	Type	Reset	Description
0	RIIM	R/W	0	UART Ring Indicator Modem Interrupt Mask

Value	Description
1	An interrupt is sent to the interrupt controller when the RIRIS bit in the **UARTRIS** register is set.
0	The RIRIS interrupt is suppressed and not sent to the interrupt controller.

This bit is implemented only on UART1 and is reserved for UART0 and UART2.

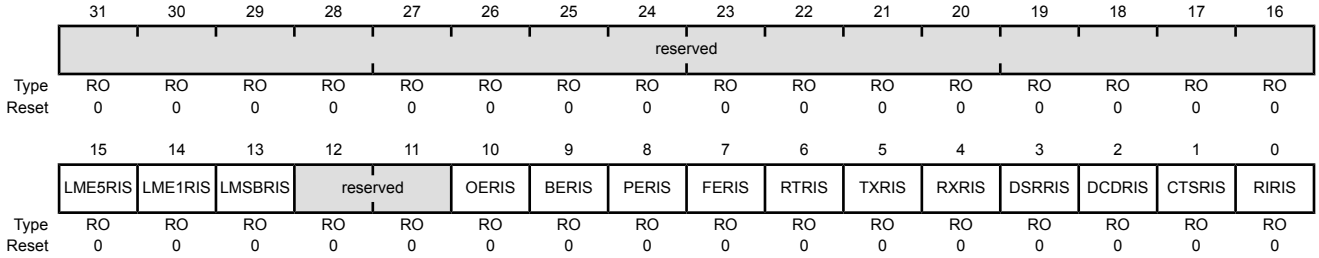
Register 11: UART Raw Interrupt Status (UARTRIS), offset 0x03C

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

Note that bits [3:0] are only implemented on UART1. These bits are reserved on UART0 and UART2.

UART Raw Interrupt Status (UARTRIS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x03C
Type RO, reset 0x0000.000F



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	LME5RIS	RO	0	LIN Mode Edge 5 Raw Interrupt Status Value Description 1 The timer value at the 5th falling edge of the LIN Sync Field has been captured. 0 No interrupt This bit is cleared by writing a 1 to the `LME5IC` bit in the **UARTICR** register.
14	LME1RIS	RO	0	LIN Mode Edge 1 Raw Interrupt Status Value Description 1 The timer value at the 1st falling edge of the LIN Sync Field has been captured. 0 No interrupt This bit is cleared by writing a 1 to the `LME1IC` bit in the **UARTICR** register.
13	LMSBRIS	RO	0	LIN Mode Sync Break Raw Interrupt Status Value Description 1 A LIN Sync Break has been detected. 0 No interrupt This bit is cleared by writing a 1 to the `LMSBIC` bit in the **UARTICR** register.

Bit/Field	Name	Type	Reset	Description
12:11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OERIS	RO	0	UART Overrun Error Raw Interrupt Status Value Description 1 An overrun error has occurred. 0 No interrupt This bit is cleared by writing a 1 to the OEIC bit in the **UARTICR** register.
9	BERIS	RO	0	UART Break Error Raw Interrupt Status Value Description 1 A break error has occurred. 0 No interrupt This bit is cleared by writing a 1 to the BEIC bit in the **UARTICR** register.
8	PERIS	RO	0	UART Parity Error Raw Interrupt Status Value Description 1 A parity error has occurred. 0 No interrupt This bit is cleared by writing a 1 to the PEIC bit in the **UARTICR** register.
7	FERIS	RO	0	UART Framing Error Raw Interrupt Status Value Description 1 A framing error has occurred. 0 No interrupt This bit is cleared by writing a 1 to the FEIC bit in the **UARTICR** register.
6	RTRIS	RO	0	UART Receive Time-Out Raw Interrupt Status Value Description 1 A receive time out has occurred. 0 No interrupt This bit is cleared by writing a 1 to the RTIC bit in the **UARTICR** register.

Bit/Field	Name	Type	Reset	Description
5	TXRIS	RO	0	UART Transmit Raw Interrupt Status Value Description 1 If the EOT bit in the **UARTCTL** register is clear, the transmit FIFO level has passed through the condition defined in the **UARTIFLS** register. If the EOT bit is set, the last bit of all transmitted data and flags has left the serializer. 0 No interrupt This bit is cleared by writing a 1 to the TXIC bit in the **UARTICR** register or by writing data to the transmit FIFO until it becomes greater than the trigger level, if the FIFO is enabled, or by writing a single byte if the FIFO is disabled.
4	RXRIS	RO	0	UART Receive Raw Interrupt Status Value Description 1 The receive FIFO level has passed through the condition defined in the **UARTIFLS** register. 0 No interrupt This bit is cleared by writing a 1 to the RXIC bit in the **UARTICR** register or by reading data from the receive FIFO until it becomes less than the trigger level, if the FIFO is enabled, or by reading a single byte if the FIFO is disabled.
3	DSRRIS	RO	0	UART Data Set Ready Modem Raw Interrupt Status Value Description 1 Data Set Ready used for software flow control. 0 No interrupt This bit is cleared by writing a 1 to the DSRIC bit in the **UARTICR** register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
2	DCDRIS	RO	0	UART Data Carrier Detect Modem Raw Interrupt Status Value Description 1 Data Carrier Detect used for software flow control. 0 No interrupt This bit is cleared by writing a 1 to the DCDIC bit in the **UARTICR** register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.

Bit/Field	Name	Type	Reset	Description
1	CTSRIS	RO	0	UART Clear to Send Modem Raw Interrupt Status Value Description 1 Clear to Send used for software flow control. 0 No interrupt This bit is cleared by writing a 1 to the CTSIC bit in the **UARTICR** register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
0	RIRIS	RO	0	UART Ring Indicator Modem Raw Interrupt Status Value Description 1 Ring Indicator used for software flow control. 0 No interrupt This bit is cleared by writing a 1 to the RIIC bit in the **UARTICR** register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.

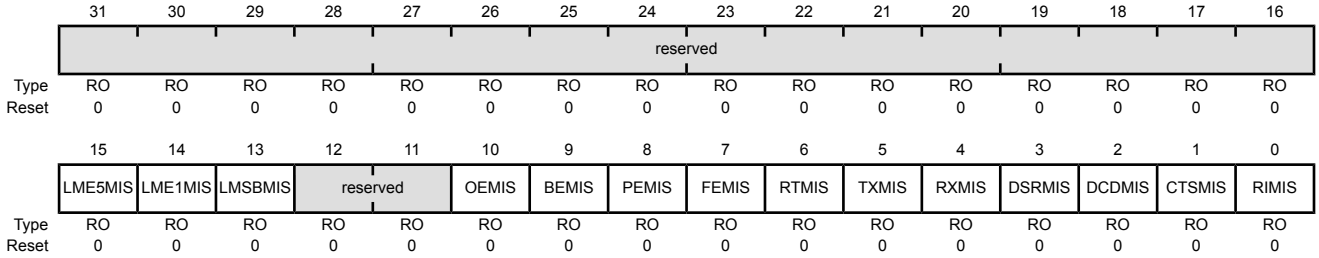
Register 12: UART Masked Interrupt Status (UARTMIS), offset 0x040

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

Note that bits [3:0] are only implemented on UART1. These bits are reserved on UART0 and UART2.

UART Masked Interrupt Status (UARTMIS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x040
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	LME5MIS	RO	0	LIN Mode Edge 5 Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to the 5th falling edge of the LIN Sync Field. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `LME5IC` bit in the **UARTICR** register.
14	LME1MIS	RO	0	LIN Mode Edge 1 Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to the 1st falling edge of the LIN Sync Field. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `LME1IC` bit in the **UARTICR** register.
13	LMSBMIS	RO	0	LIN Mode Sync Break Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to the receipt of a LIN Sync Break. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `LMSBIC` bit in the **UARTICR** register.

Bit/Field	Name	Type	Reset	Description
12:11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEMIS	RO	0	UART Overrun Error Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to an overrun error. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the OEIC bit in the **UARTICR** register.
9	BEMIS	RO	0	UART Break Error Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to a break error. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the BEIC bit in the **UARTICR** register.
8	PEMIS	RO	0	UART Parity Error Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to a parity error. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the PEIC bit in the **UARTICR** register.
7	FEMIS	RO	0	UART Framing Error Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to a framing error. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the FEIC bit in the **UARTICR** register.
6	RTMIS	RO	0	UART Receive Time-Out Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to a receive time out. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the RTIC bit in the **UARTICR** register.

Bit/Field	Name	Type	Reset	Description
5	TXMIS	RO	0	UART Transmit Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to passing through the specified transmit FIFO level (if the EOT bit is clear) or due to the transmission of the last data bit (if the EOT bit is set). 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the TXIC bit in the **UARTICR** register or by writing data to the transmit FIFO until it becomes greater than the trigger level, if the FIFO is enabled, or by writing a single byte if the FIFO is disabled.
4	RXMIS	RO	0	UART Receive Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to passing through the specified receive FIFO level. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the RXIC bit in the **UARTICR** register or by reading data from the receive FIFO until it becomes less than the trigger level, if the FIFO is enabled, or by reading a single byte if the FIFO is disabled.
3	DSRMIS	RO	0	UART Data Set Ready Modem Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to Data Set Ready. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the DSRIC bit in the **UARTICR** register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
2	DCDMIS	RO	0	UART Data Carrier Detect Modem Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to Data Carrier Detect. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the DCDIC bit in the **UARTICR** register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.

Bit/Field	Name	Type	Reset	Description
1	CTSMIS	RO	0	UART Clear to Send Modem Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to Clear to Send. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the CTSIC bit in the **UARTICR** register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
0	RIMIS	RO	0	UART Ring Indicator Modem Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to Ring Indicator. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the RIIC bit in the **UARTICR** register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.

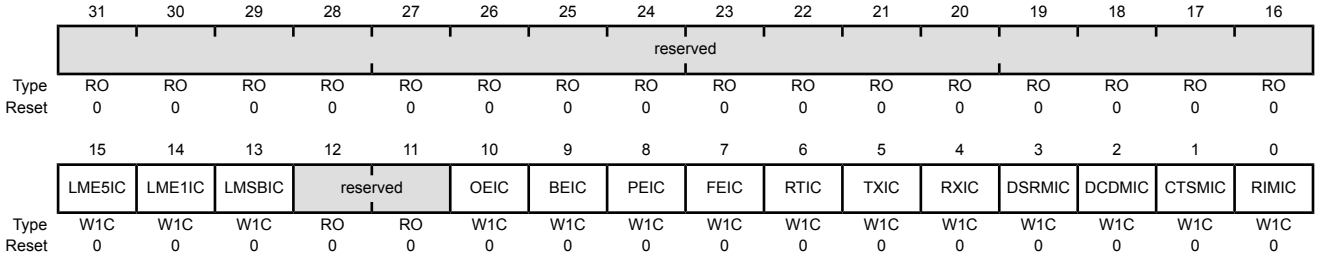
Register 13: UART Interrupt Clear (UARTICR), offset 0x044

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

Note that bits [3:0] are only implemented on UART1. These bits are reserved on UART0 and UART2.

UART Interrupt Clear (UARTICR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x044
Type W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	LME5IC	W1C	0	LIN Mode Edge 5 Interrupt Clear Writing a 1 to this bit clears the LME5RIS bit in the **UARTRIS** register and the LME5MIS bit in the **UARTMIS** register.
14	LME1IC	W1C	0	LIN Mode Edge 1 Interrupt Clear Writing a 1 to this bit clears the LME1RIS bit in the **UARTRIS** register and the LME1MIS bit in the **UARTMIS** register.
13	LMSBIC	W1C	0	LIN Mode Sync Break Interrupt Clear Writing a 1 to this bit clears the LMSBRIS bit in the **UARTRIS** register and the LMSBMIS bit in the **UARTMIS** register.
12:11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEIC	W1C	0	Overrun Error Interrupt Clear Writing a 1 to this bit clears the OERIS bit in the **UARTRIS** register and the OEMIS bit in the **UARTMIS** register.
9	BEIC	W1C	0	Break Error Interrupt Clear Writing a 1 to this bit clears the BERIS bit in the **UARTRIS** register and the BEMIS bit in the **UARTMIS** register.
8	PEIC	W1C	0	Parity Error Interrupt Clear Writing a 1 to this bit clears the PERIS bit in the **UARTRIS** register and the PEMIS bit in the **UARTMIS** register.

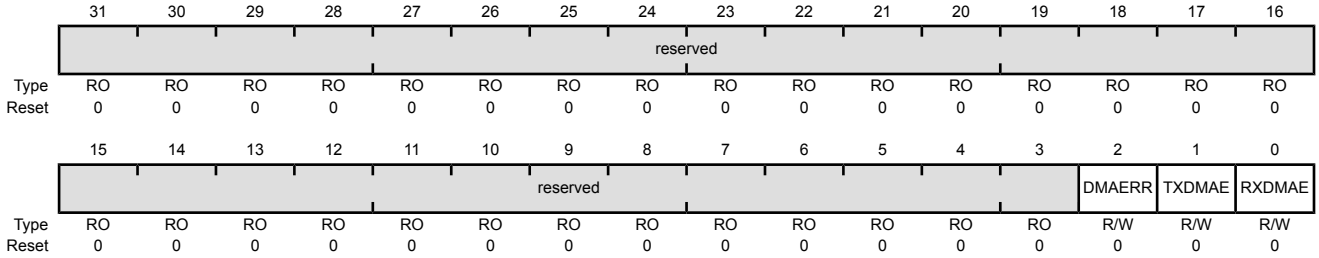
Bit/Field	Name	Type	Reset	Description
7	FEIC	W1C	0	Framing Error Interrupt Clear Writing a 1 to this bit clears the FERIS bit in the **UARTRIS** register and the FEMIS bit in the **UARTMIS** register.
6	RTIC	W1C	0	Receive Time-Out Interrupt Clear Writing a 1 to this bit clears the RTRIS bit in the **UARTRIS** register and the RTMIS bit in the **UARTMIS** register.
5	TXIC	W1C	0	Transmit Interrupt Clear Writing a 1 to this bit clears the TXRIS bit in the **UARTRIS** register and the TXMIS bit in the **UARTMIS** register.
4	RXIC	W1C	0	Receive Interrupt Clear Writing a 1 to this bit clears the RXRIS bit in the **UARTRIS** register and the RXMIS bit in the **UARTMIS** register.
3	DSRMIC	W1C	0	UART Data Set Ready Modem Interrupt Clear Writing a 1 to this bit clears the DSRRIS bit in the **UARTRIS** register and the DSRMIS bit in the **UARTMIS** register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
2	DCDMIC	W1C	0	UART Data Carrier Detect Modem Interrupt Clear Writing a 1 to this bit clears the DCDRIS bit in the **UARTRIS** register and the DCDMIS bit in the **UARTMIS** register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
1	CTSMIC	W1C	0	UART Clear to Send Modem Interrupt Clear Writing a 1 to this bit clears the CTSRIS bit in the **UARTRIS** register and the CTSMIS bit in the **UARTMIS** register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
0	RIMIC	W1C	0	UART Ring Indicator Modem Interrupt Clear Writing a 1 to this bit clears the RIRIS bit in the **UARTRIS** register and the RIMIS bit in the **UARTMIS** register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.

Register 14: UART DMA Control (UARTDMACTL), offset 0x048

The **UARTDMACTL** register is the DMA control register.

UART DMA Control (UARTDMACTL)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x048
Type R/W, reset 0x0000.0000



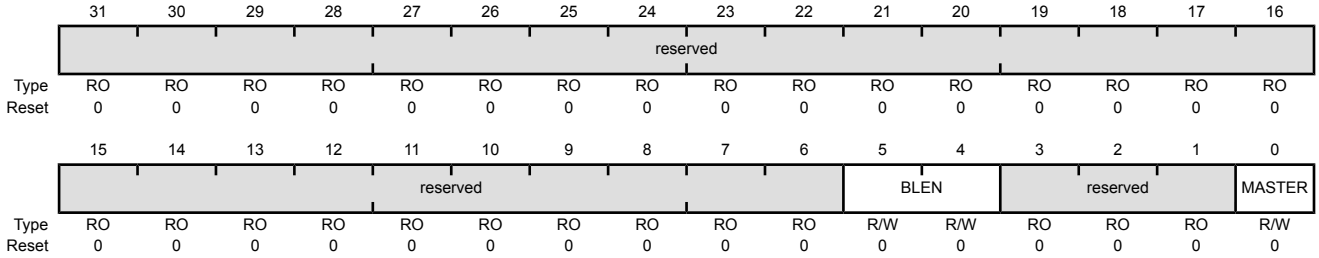
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	DMAERR	R/W	0	DMA on Error Value Description 1 µDMA receive requests are automatically disabled when a receive error occurs. 0 µDMA receive requests are unaffected when a receive error occurs.
1	TXDMAE	R/W	0	Transmit DMA Enable Value Description 1 µDMA for the transmit FIFO is enabled. 0 µDMA for the transmit FIFO is disabled.
0	RXDMAE	R/W	0	Receive DMA Enable Value Description 1 µDMA for the receive FIFO is enabled. 0 µDMA for the receive FIFO is disabled.

Register 15: UART LIN Control (UARTLCTL), offset 0x090

The **UARTLCTL** register is the configures the operation of the UART when in LIN mode.

UART LIN Control (UARTLCTL)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x090
Type R/W, reset 0x0000.0000



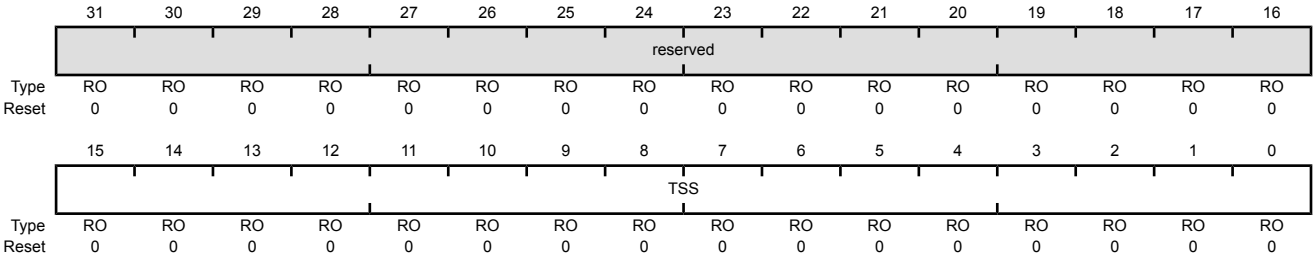
Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:4	BLEN	R/W	0x0	Sync Break Length Value Description 0x3 Sync break length is 16T bits 0x2 Sync break length is 15T bits 0x1 Sync break length is 14T bits 0x0 Sync break length is 13T bits (default)
3:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	MASTER	R/W	0	LIN Master Enable Value Description 1 The UART operates as a LIN master. 0 The UART operates as a LIN slave.

Register 16: UART LIN Snap Shot (UARTLSS), offset 0x094

The **UARTLSS** register captures the free-running timer value when either the Sync Edge 1 or the Sync Edge 5 is detected in LIN mode.

UART LIN Snap Shot (UARTLSS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x094
Type RO, reset 0x0000.0000



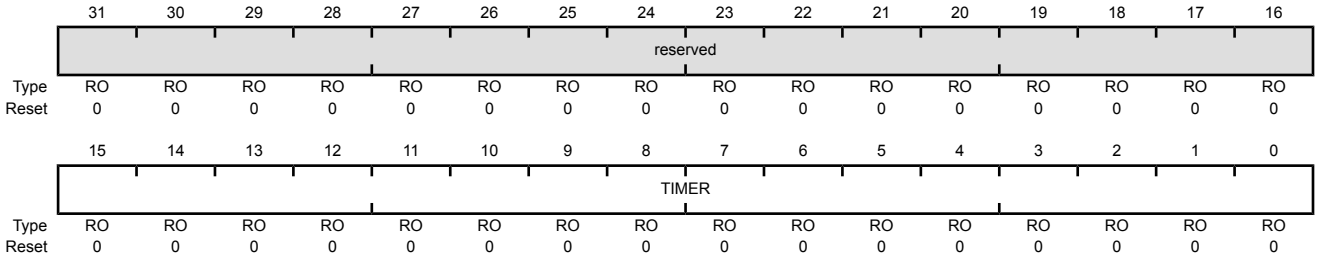
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TSS	RO	0x0000	Timer Snap Shot This field contains the value of the free-running timer when either the Sync Edge 5 or the Sync Edge 1 was detected.

Register 17: UART LIN Timer (UARTLTIM), offset 0x098

The **UARTLTIM** register contains the current timer value for the free-running timer that is used to calculate the baud rate when in LIN slave mode. The value in this register is used along with the value in the **UART LIN Snap Shot (UARTLSS)** register to adjust the baud rate to match that of the master.

UART LIN Timer (UARTLTIM)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x098
Type RO, reset 0x0000.0000



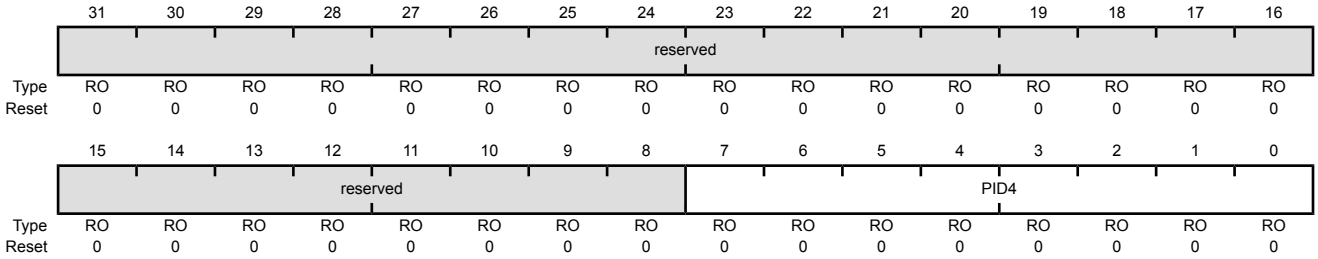
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TIMER	RO	0x0000	Timer Value This field contains the value of the free-running timer.

Register 18: UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0

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

UART Peripheral Identification 4 (UARTPeriphID4)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFD0
Type RO, reset 0x0000.0000



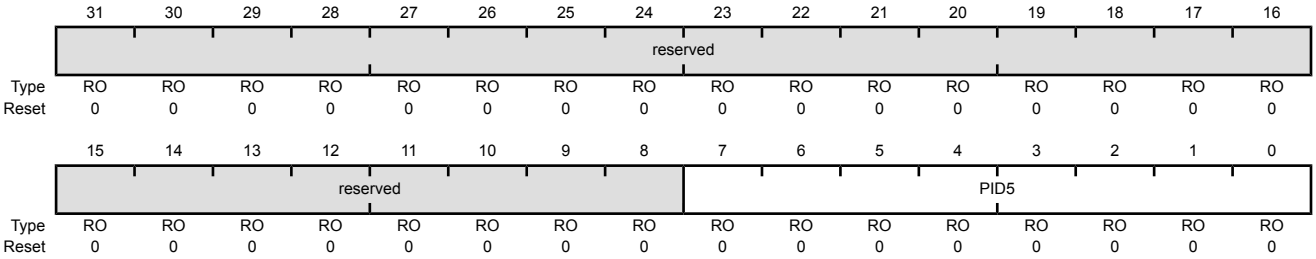
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	UART Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

Register 19: UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4

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

UART Peripheral Identification 5 (UARTPeriphID5)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFD4
Type RO, reset 0x0000.0000



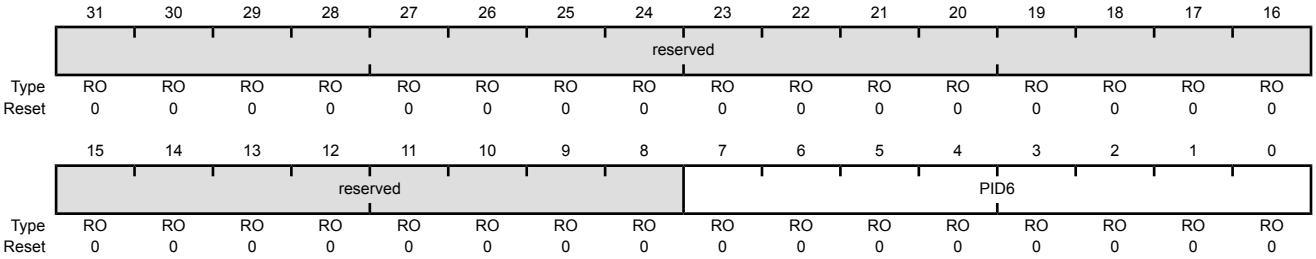
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	UART Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Register 20: UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8

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

UART Peripheral Identification 6 (UARTPeriphID6)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFD8
Type RO, reset 0x0000.0000



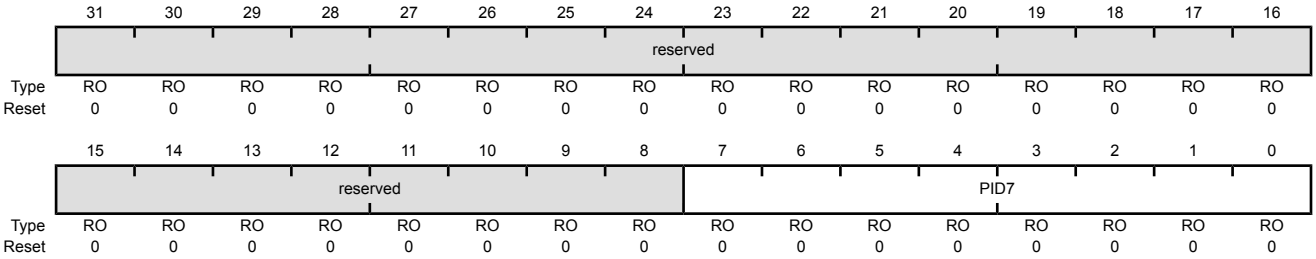
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	UART Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 21: UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC

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

UART Peripheral Identification 7 (UARTPeriphID7)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFDC
Type RO, reset 0x0000.0000



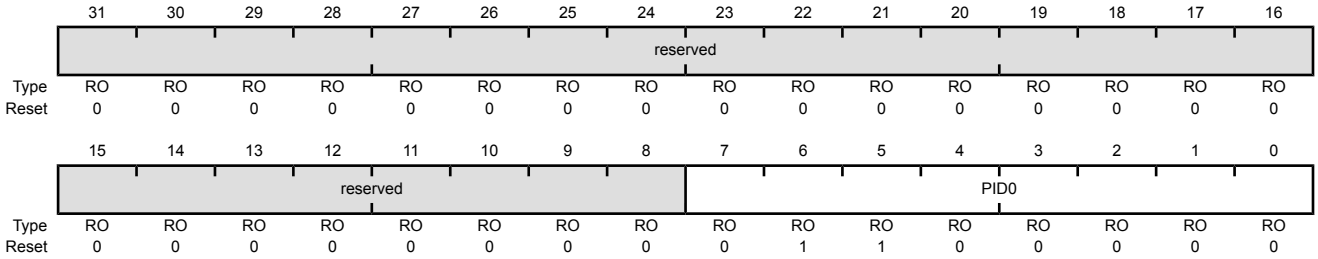
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	UART Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

Register 22: UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0

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

UART Peripheral Identification 0 (UARTPeriphID0)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFE0
Type RO, reset 0x0000.0060



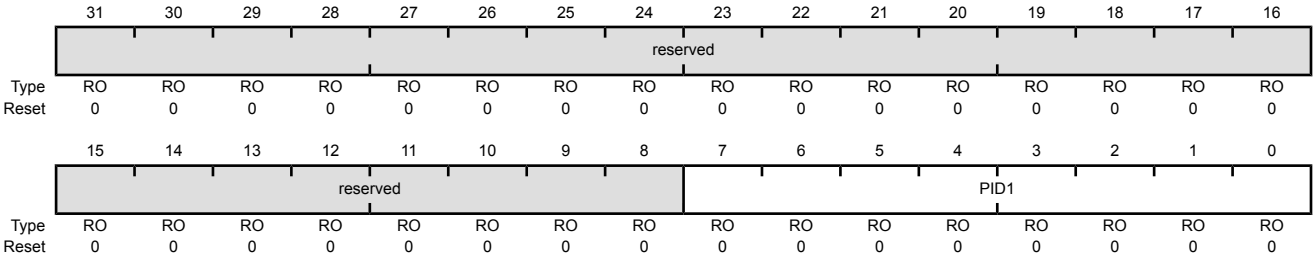
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x60	UART Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

Register 23: UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4

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

UART Peripheral Identification 1 (UARTPeriphID1)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFE4
Type RO, reset 0x0000.0000



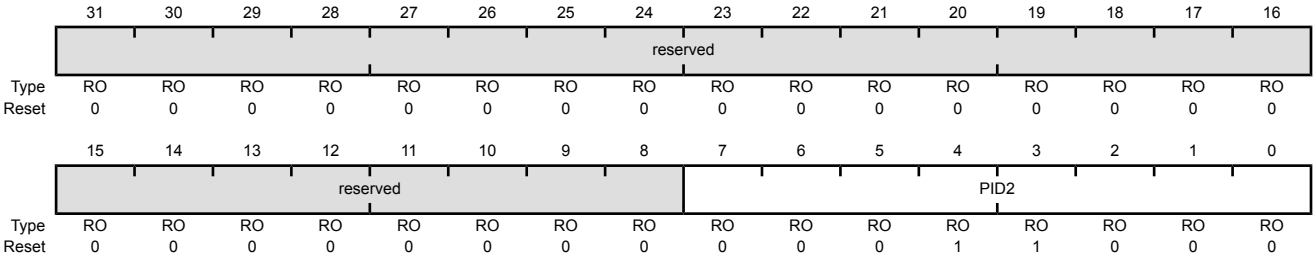
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	UART Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Register 24: UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8

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

UART Peripheral Identification 2 (UARTPeriphID2)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFE8
Type RO, reset 0x0000.0018



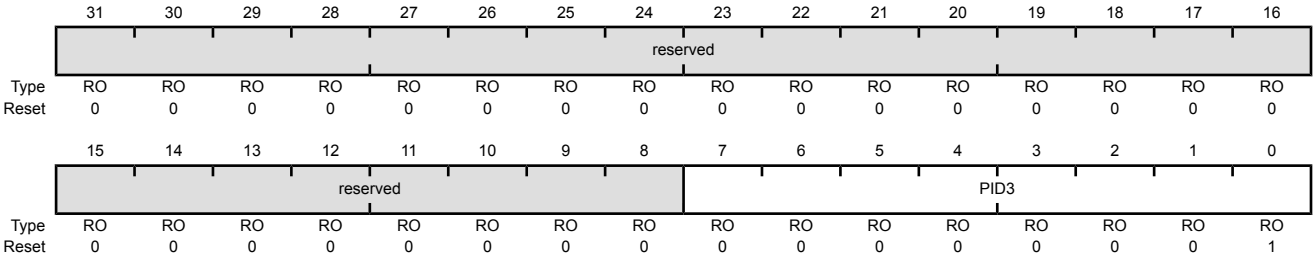
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	UART Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 25: UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC

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

UART Peripheral Identification 3 (UARTPeriphID3)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFEC
Type RO, reset 0x0000.0001



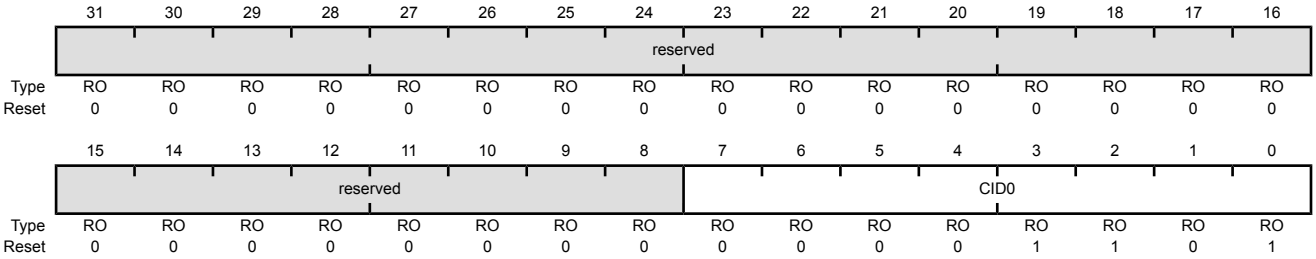
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	UART Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

Register 26: UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0

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

UART PrimeCell Identification 0 (UARTPCellID0)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFF0
Type RO, reset 0x0000.000D



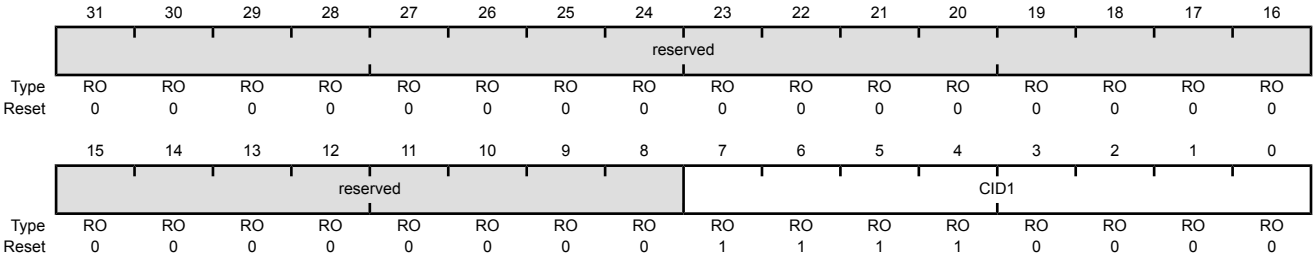
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	UART PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

Register 27: UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4

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

UART PrimeCell Identification 1 (UARTPCellID1)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFF4
Type RO, reset 0x0000.00F0



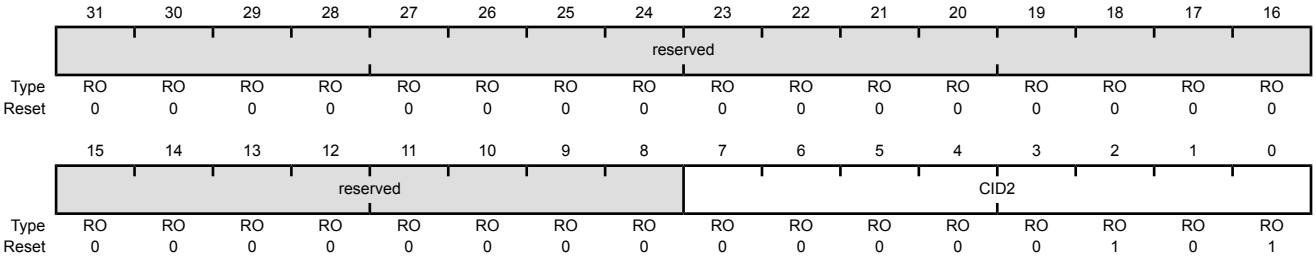
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	UART PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system.

Register 28: UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8

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

UART PrimeCell Identification 2 (UARTPCellID2)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFF8
Type RO, reset 0x0000.0005



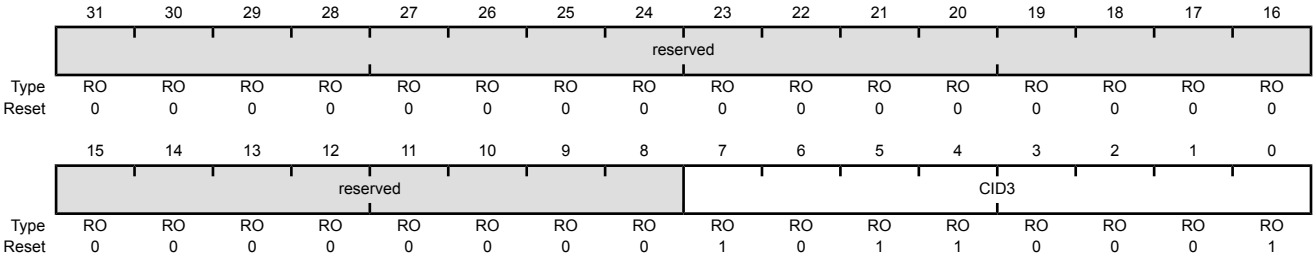
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	UART PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system.

Register 29: UART PrimeCell Identification 3 (UARTPCellID3), offset 0xFFC

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

UART PrimeCell Identification 3 (UARTPCellID3)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFFC
Type RO, reset 0x0000.00B1



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	UART PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system.

14 Synchronous Serial Interface (SSI)

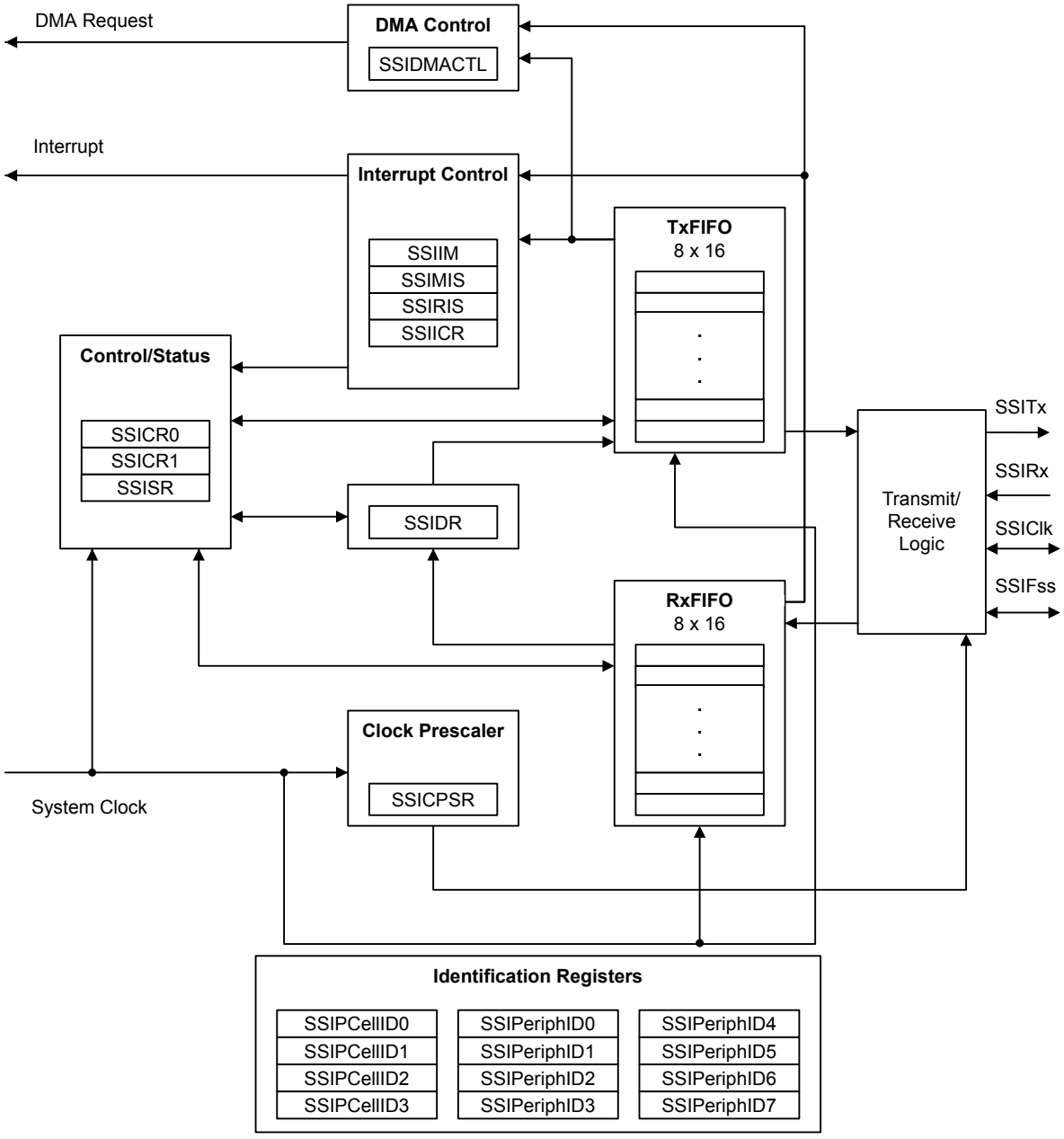
The Stellaris® microcontroller includes two Synchronous Serial Interface (SSI) modules. Each SSI is a master or slave interface for synchronous serial communication with peripheral devices that have either Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces.

The Stellaris LM3S9U81 controller includes two SSI modules with the following features:

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

14.1 Block Diagram

Figure 14-1. SSI Module Block Diagram



14.2 Signal Description

The following table lists the external signals of the SSI module and describes the function of each. The SSI signals are alternate functions for some GPIO signals and default to be GPIO signals at reset., with the exception of the SSI0Clk, SSI0Fss, SSI0Rx, and SSI0Tx pins which default to the SSI function. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the SSI signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 419) should be set to choose the SSI function. The number in

parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 437) to assign the SSI signal to the specified GPIO port pin. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 395.

Table 14-1. SSI Signals (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
SSI0Clk	28	PA2 (1)	I/O	TTL	SSI module 0 clock.
SSI0Fss	29	PA3 (1)	I/O	TTL	SSI module 0 frame.
SSI0Rx	30	PA4 (1)	I	TTL	SSI module 0 receive.
SSI0Tx	31	PA5 (1)	O	TTL	SSI module 0 transmit.
SSI1Clk	60 74 76	PF2 (9) PE0 (2) PH4 (11)	I/O	TTL	SSI module 1 clock.
SSI1Fss	59 63 75	PF3 (9) PH5 (11) PE1 (2)	I/O	TTL	SSI module 1 frame.
SSI1Rx	42 62 95	PF4 (9) PH6 (11) PE2 (2)	I	TTL	SSI module 1 receive.
SSI1Tx	15 41 96	PH7 (11) PF5 (9) PE3 (2)	O	TTL	SSI module 1 transmit.

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

Table 14-2. SSI Signals (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
SSI0Clk	M4	PA2 (1)	I/O	TTL	SSI module 0 clock.
SSI0Fss	L4	PA3 (1)	I/O	TTL	SSI module 0 frame.
SSI0Rx	L5	PA4 (1)	I	TTL	SSI module 0 receive.
SSI0Tx	M5	PA5 (1)	O	TTL	SSI module 0 transmit.
SSI1Clk	J11 B11 B10	PF2 (9) PE0 (2) PH4 (11)	I/O	TTL	SSI module 1 clock.
SSI1Fss	J12 F10 A12	PF3 (9) PH5 (11) PE1 (2)	I/O	TTL	SSI module 1 frame.
SSI1Rx	K4 G3 A4	PF4 (9) PH6 (11) PE2 (2)	I	TTL	SSI module 1 receive.
SSI1Tx	H3 K3 B4	PH7 (11) PF5 (9) PE3 (2)	O	TTL	SSI module 1 transmit.

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

14.3 Functional Description

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

and receive modes. The SSI also supports the µDMA interface. The transmit and receive FIFOs can be programmed as destination/source addresses in the µDMA module. µDMA operation is enabled by setting the appropriate bit(s) in the **SSIDMACTL** register (see page 771).

14.3.1 Bit Rate Generation

The SSI includes a programmable bit rate clock divider and prescaler to generate the serial output clock. Bit rates are supported to 2 MHz and higher, although maximum bit rate is determined by peripheral devices.

The serial bit rate is derived by dividing down the input clock (SysClk). The clock is first divided by an even prescale value CPSDVSR from 2 to 254, which is programmed in the **SSI Clock Prescale (SSICPSR)** register (see page 764). The clock is further divided by a value from 1 to 256, which is 1 + SCR, where SCR is the value programmed in the **SSI Control 0 (SSICR0)** register (see page 757).

The frequency of the output clock SSIClk is defined by:

```
SSIClk = SysClk / (CPSDVSR * (1 + SCR))
```

Note: For master mode, the system clock must be at least two times faster than the SSIClk, with the restriction that SSIClk cannot be faster than 25 MHz. For slave mode, the system clock must be at least 12 times faster than the SSIClk.

See “Synchronous Serial Interface (SSI)” on page 1208 to view SSI timing parameters.

14.3.2 FIFO Operation

14.3.2.1 Transmit FIFO

The common transmit FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. The CPU writes data to the FIFO by writing the **SSI Data (SSIDR)** register (see page 761), and data is stored in the FIFO until it is read out by the transmission logic.

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

In slave mode, the SSI transmits data each time the master initiates a transaction. If the transmit FIFO is empty and the master initiates, the slave transmits the 8th most recent value in the transmit FIFO. If less than 8 values have been written to the transmit FIFO since the SSI module clock was enabled using the SSI bit in the **RGCG1** register, then 0 is transmitted. Care should be taken to ensure that valid data is in the FIFO as needed. The SSI can be configured to generate an interrupt or a µDMA request when the FIFO is empty.

14.3.2.2 Receive FIFO

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

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

14.3.3 Interrupts

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

- Transmit FIFO service (when the transmit FIFO is half full or less)
- Receive FIFO service (when the receive FIFO is half full or more)

- Receive FIFO time-out
- Receive FIFO overrun
- End of transmission

All of the interrupt events are ORed together before being sent to the interrupt controller, so the SSI generates a single interrupt request to the controller regardless of the number of active interrupts. Each of the four individual maskable interrupts can be masked by clearing the appropriate bit in the **SSI Interrupt Mask (SSIIM)** register (see page 765). Setting the appropriate mask bit enables the interrupt.

The individual outputs, along with a combined interrupt output, allow use of either a global interrupt service routine or modular device drivers to handle interrupts. The transmit and receive dynamic dataflow interrupts have been separated from the status interrupts so that data can be read or written in response to the FIFO trigger levels. The status of the individual interrupt sources can be read from the **SSI Raw Interrupt Status (SSIRIS)** and **SSI Masked Interrupt Status (SSIMIS)** registers (see page 766 and page 768, respectively).

The receive FIFO has a time-out period that is 32 periods at the rate of SSIClk (whether or not SSIClk is currently active) and is started when the RX FIFO goes from EMPTY to not-EMPTY. If the RX FIFO is emptied before 32 clocks have passed, the time-out period is reset. As a result, the ISR should clear the Receive FIFO Time-out Interrupt just after reading out the RX FIFO by writing a 1 to the RTIC bit in the **SSI Interrupt Clear (SSIICR)** register. The interrupt should not be cleared so late that the ISR returns before the interrupt is actually cleared, or the ISR may be re-activated unnecessarily.

The End-of-Transmission (EOT) interrupt indicates that the data has been transmitted completely. This interrupt can be used to indicate when it is safe to turn off the SSI module clock or enter sleep mode. In addition, because transmitted data and received data complete at exactly the same time, the interrupt can also indicate that read data is ready immediately, without waiting for the receive FIFO time-out period to complete.

14.3.4 Frame Formats

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

- Texas Instruments synchronous serial
- Freescale SPI
- MICROWIRE

For all three formats, the serial clock (SSIClk) is held inactive while the SSI is idle, and SSIClk transitions at the programmed frequency only during active transmission or reception of data. The idle state of SSIClk is utilized to provide a receive timeout indication that occurs when the receive FIFO still contains data after a timeout period.

For Freescale SPI and MICROWIRE frame formats, the serial frame (SSIFss) pin is active Low, and is asserted (pulled down) during the entire transmission of the frame.

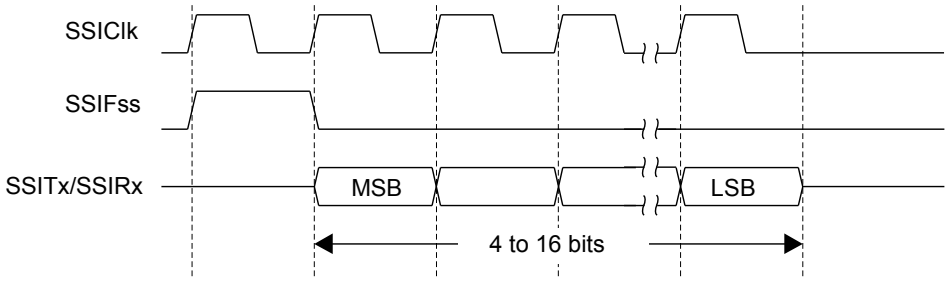
For Texas Instruments synchronous serial frame format, the SSIFss pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format, both the SSI and the off-chip slave device drive their output data on the rising edge of SSIClk and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

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

14.3.4.1 Texas Instruments Synchronous Serial Frame Format

Figure 14-2 on page 747 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

Figure 14-2. TI Synchronous Serial Frame Format (Single Transfer)

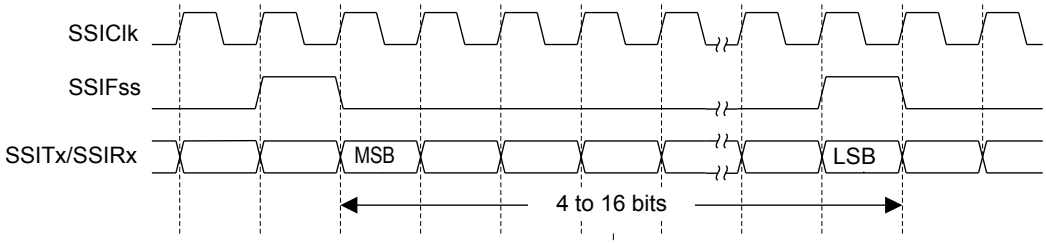


In this mode, SSIClk and SSIFss are forced Low, and the transmit data line SSITx is tristated whenever the SSI is idle. Once the bottom entry of the transmit FIFO contains data, SSIFss is pulsed High for one SSIClk period. The value to be transmitted is also transferred from the transmit FIFO to the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

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

Figure 14-3 on page 747 shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

Figure 14-3. TI Synchronous Serial Frame Format (Continuous Transfer)



14.3.4.2 Freescale SPI Frame Format

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

SPO Clock Polarity Bit

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

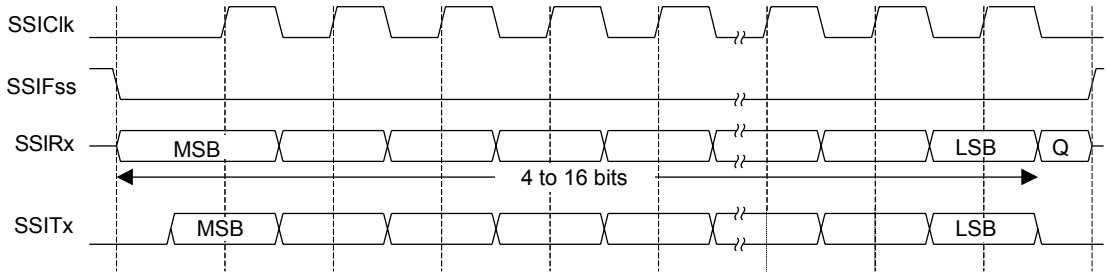
SPH Phase Control Bit

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

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

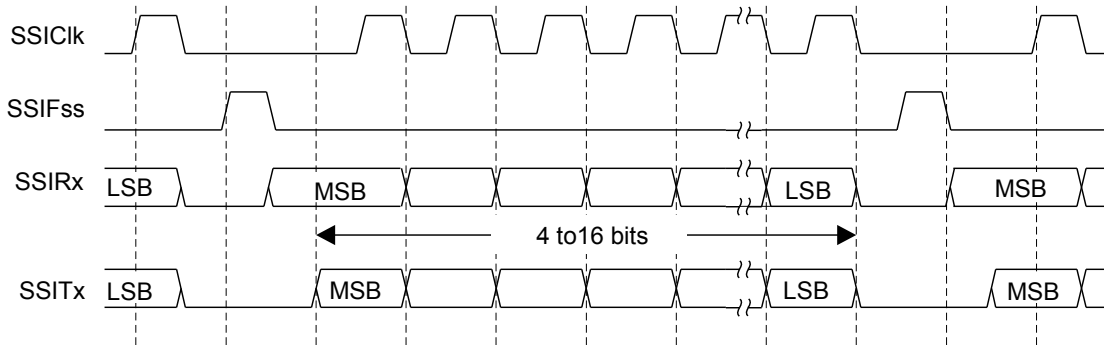
Single and continuous transmission signal sequences for Freescale SPI format with SPO=0 and SPH=0 are shown in Figure 14-4 on page 748 and Figure 14-5 on page 748.

Figure 14-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0



Note: Q is undefined.

Figure 14-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0



In this configuration, during idle periods:

- SSIClk is forced Low

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

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

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

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

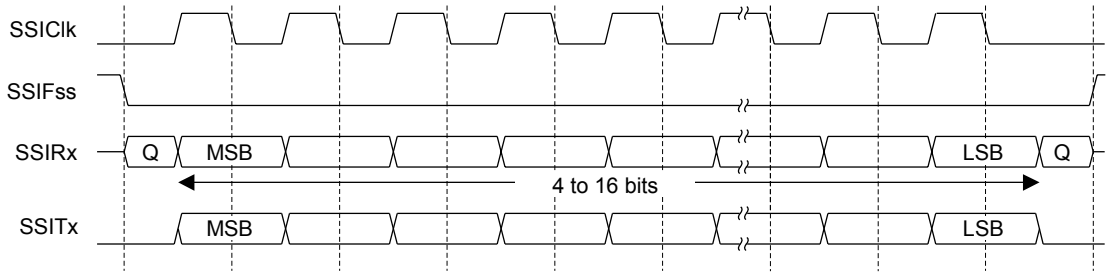
In the case of a single word transmission, after all bits of the data word have been transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

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

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

The transfer signal sequence for Freescale SPI format with SPO=0 and SPH=1 is shown in Figure 14-6 on page 749, which covers both single and continuous transfers.

Figure 14-6. Freescale SPI Frame Format with SPO=0 and SPH=1



Note: Q is undefined.

In this configuration, during idle periods:

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

- When the SSI is configured as a slave, it disables the SSIClk pad

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

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

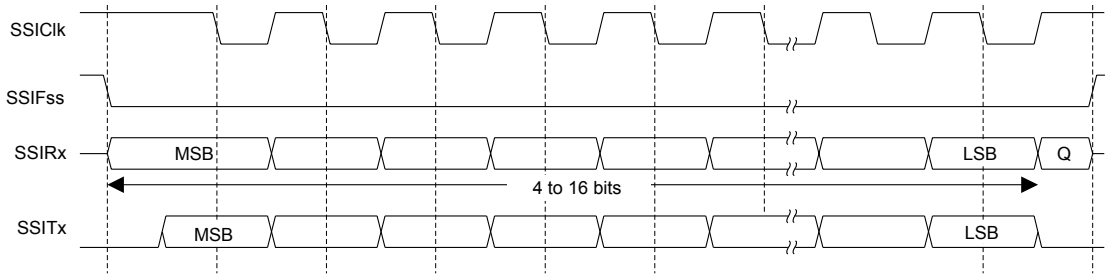
In the case of a single word transfer, after all bits have been transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

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

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

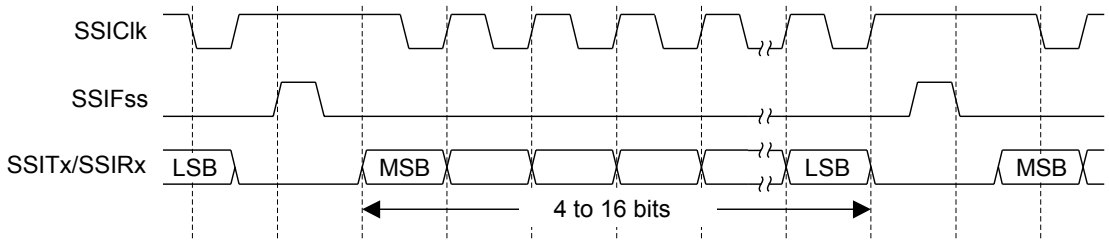
Single and continuous transmission signal sequences for Freescale SPI format with SPO=1 and SPH=0 are shown in Figure 14-7 on page 750 and Figure 14-8 on page 750.

Figure 14-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0



Note: Q is undefined.

Figure 14-8. Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0



In this configuration, during idle periods:

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

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

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

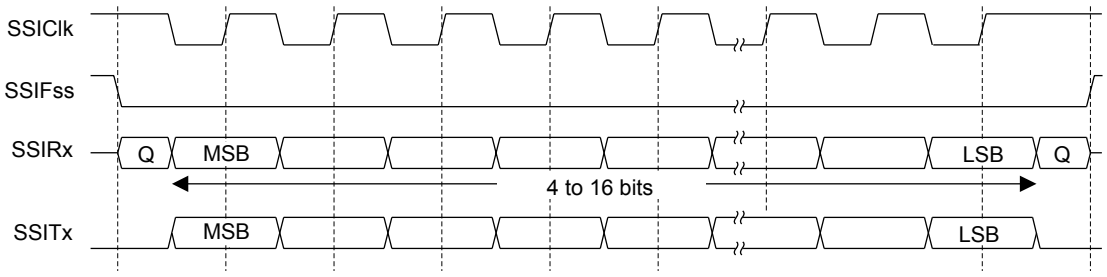
In the case of a single word transmission, after all bits of the data word are transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

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

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

The transfer signal sequence for Freescale SPI format with SPO=1 and SPH=1 is shown in Figure 14-9 on page 751, which covers both single and continuous transfers.

Figure 14-9. Freescale SPI Frame Format with SPO=1 and SPH=1



Note: Q is undefined.

In this configuration, during idle periods:

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

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

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

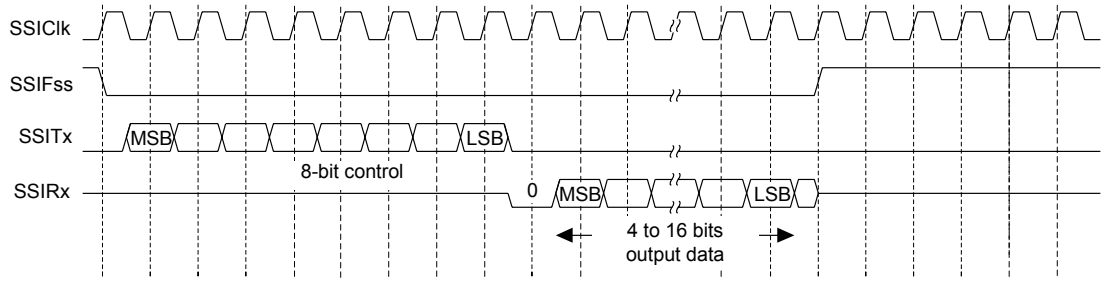
For continuous back-to-back transmissions, the SSIFss pin remains in its active Low state until the final bit of the last word has been captured and then returns to its idle state as described above.

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

14.3.4.7 MICROWIRE Frame Format

Figure 14-10 on page 752 shows the MICROWIRE frame format for a single frame. Figure 14-11 on page 753 shows the same format when back-to-back frames are transmitted.

Figure 14-10. MICROWIRE Frame Format (Single Frame)



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

In this configuration, during idle periods:

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

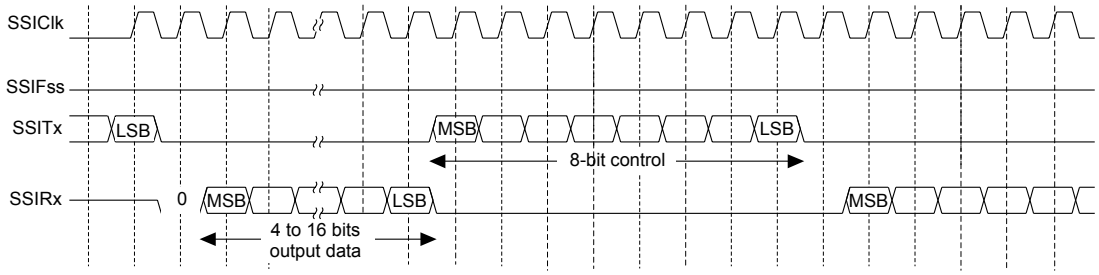
A transmission is triggered by writing a control byte to the transmit FIFO. The falling edge of SSIFss causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic and the MSB of the 8-bit control frame to be shifted out onto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

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

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

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

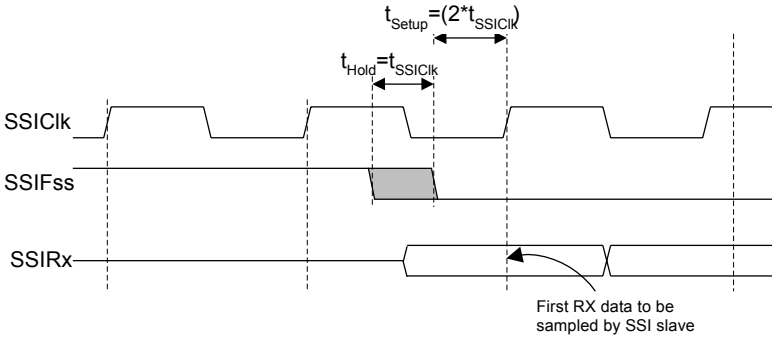
Figure 14-11. MICROWIRE Frame Format (Continuous Transfer)



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

Figure 14-12 on page 753 illustrates these setup and hold time requirements. With respect to the SSIClk rising edge on which the first bit of receive data is to be sampled by the SSI slave, SSIFss must have a setup of at least two times the period of SSIClk on which the SSI operates. With respect to the SSIClk rising edge previous to this edge, SSIFss must have a hold of at least one SSIClk period.

Figure 14-12. MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements



14.3.5 DMA Operation

The SSI peripheral provides an interface to the µDMA controller with separate channels for transmit and receive. The µDMA operation of the SSI is enabled through the **SSI DMA Control (SSIDMACTL)** register. When µDMA operation is enabled, the SSI asserts a µDMA request on the receive or transmit channel when the associated FIFO can transfer data. For the receive channel, a single transfer request is asserted whenever any data is in the receive FIFO. A burst transfer request is asserted whenever the amount of data in the receive FIFO is 4 or more items. For the transmit channel, a single transfer request is asserted whenever at least one empty location is in the transmit FIFO. The burst request is asserted whenever the transmit FIFO has 4 or more empty slots. The

single and burst µDMA transfer requests are handled automatically by the µDMA controller depending how the µDMA channel is configured. To enable µDMA operation for the receive channel, the RXDMAE bit of the **DMA Control (SSIDMACTL)** register should be set. To enable µDMA operation for the transmit channel, the TXDMAE bit of **SSIDMACTL** should be set. If µDMA is enabled, then the µDMA controller triggers an interrupt when a transfer is complete. The interrupt occurs on the SSI interrupt vector. Therefore, if interrupts are used for SSI operation and µDMA is enabled, the SSI interrupt handler must be designed to handle the µDMA completion interrupt.

See "Micro Direct Memory Access (µDMA)" on page 334 for more details about programming the µDMA controller.

14.4 Initialization and Configuration

To enable and initialize the SSI, the following steps are necessary:

1. Enable the SSI module by setting the SSI bit in the **RCGC1** register (see page 263).
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register (see page 272). To find out which GPIO port to enable, refer to Table 22-5 on page 1151.
3. Set the GPIO AFSEL bits for the appropriate pins (see page 419). To determine which GPIOs to configure, see Table 22-4 on page 1144.
4. Configure the PMCn fields in the **GPIOPCTL** register to assign the SSI signals to the appropriate pins. See page 437 and Table 22-5 on page 1151.

For each of the frame formats, the SSI is configured using the following steps:

1. Ensure that the SSE bit in the **SSICR1** register is clear before making any configuration changes.
2. Select whether the SSI is a master or slave:
   a. For master operations, set the **SSICR1** register to 0x0000.0000.
   b. For slave mode (output enabled), set the **SSICR1** register to 0x0000.0004.
   c. For slave mode (output disabled), set the **SSICR1** register to 0x0000.000C.
3. Configure the clock prescale divisor by writing the **SSICPSR** register.
4. Write the **SSICR0** register with the following configuration:
   - Serial clock rate (SCR)
   - Desired clock phase/polarity, if using Freescale SPI mode (SPH and SPO)
   - The protocol mode: Freescale SPI, TI SSF, MICROWIRE (FRF)
   - The data size (DSS)
5. Optionally, configure the µDMA channel (see "Micro Direct Memory Access (µDMA)" on page 334) and enable the DMA option(s) in the **SSIDMACTL** register.
6. Enable the SSI by setting the SSE bit in the **SSICR1** register.

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

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

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

```
SSIClk = SysClk / (CPSDVSR * (1 + SCR))
1x106  = 20x106 / (CPSDVSR * (1 + SCR))
```

In this case, if CPSDVSR=0x2, SCR must be 0x9.

The configuration sequence would be as follows:

1. Ensure that the SSE bit in the **SSICR1** register is clear.
2. Write the **SSICR1** register with a value of 0x0000.0000.
3. Write the **SSICPSR** register with a value of 0x0000.0002.
4. Write the **SSICR0** register with a value of 0x0000.09C7.
5. The SSI is then enabled by setting the SSE bit in the **SSICR1** register.

14.5 Register Map

Table 14-3 on page 755 lists the SSI registers. The offset listed is a hexadecimal increment to the register's address, relative to that SSI module's base address:

- SSI0: 0x4000.8000
- SSI1: 0x4000.9000

Note that the SSI module clock must be enabled before the registers can be programmed (see page 263). There must be a delay of 3 system clocks after the SSI module clock is enabled before any SSI module registers are accessed.

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

Table 14-3. SSI Register Map

Offset	Name	Type	Reset	Description	See page
0x000	SSICR0	R/W	0x0000.0000	SSI Control 0	757
0x004	SSICR1	R/W	0x0000.0000	SSI Control 1	759
0x008	SSIDR	R/W	0x0000.0000	SSI Data	761
0x00C	SSISR	RO	0x0000.0003	SSI Status	762
0x010	SSICPSR	R/W	0x0000.0000	SSI Clock Prescale	764
0x014	SSIIM	R/W	0x0000.0000	SSI Interrupt Mask	765

Table 14-3. SSI Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x018	SSIRIS	RO	0x0000.0008	SSI Raw Interrupt Status	766
0x01C	SSIMIS	RO	0x0000.0000	SSI Masked Interrupt Status	768
0x020	SSIICR	W1C	0x0000.0000	SSI Interrupt Clear	770
0x024	SSIDMACTL	R/W	0x0000.0000	SSI DMA Control	771
0xFD0	SSIPeriphID4	RO	0x0000.0000	SSI Peripheral Identification 4	772
0xFD4	SSIPeriphID5	RO	0x0000.0000	SSI Peripheral Identification 5	773
0xFD8	SSIPeriphID6	RO	0x0000.0000	SSI Peripheral Identification 6	774
0xFDC	SSIPeriphID7	RO	0x0000.0000	SSI Peripheral Identification 7	775
0xFE0	SSIPeriphID0	RO	0x0000.0022	SSI Peripheral Identification 0	776
0xFE4	SSIPeriphID1	RO	0x0000.0000	SSI Peripheral Identification 1	777
0xFE8	SSIPeriphID2	RO	0x0000.0018	SSI Peripheral Identification 2	778
0xFEC	SSIPeriphID3	RO	0x0000.0001	SSI Peripheral Identification 3	779
0xFF0	SSIPCellID0	RO	0x0000.000D	SSI PrimeCell Identification 0	780
0xFF4	SSIPCellID1	RO	0x0000.00F0	SSI PrimeCell Identification 1	781
0xFF8	SSIPCellID2	RO	0x0000.0005	SSI PrimeCell Identification 2	782
0xFFC	SSIPCellID3	RO	0x0000.00B1	SSI PrimeCell Identification 3	783

14.6 Register Descriptions

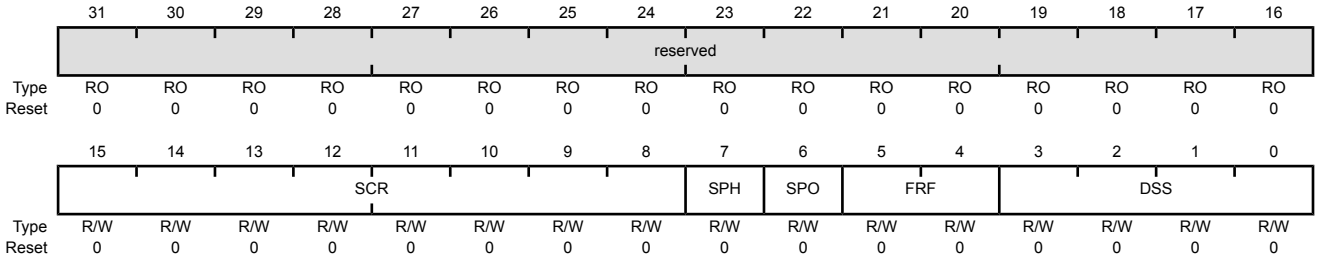
The remainder of this section lists and describes the SSI registers, in numerical order by address offset.

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

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

SSI Control 0 (SSICR0)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x000
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	SCR	R/W	0x00	SSI Serial Clock Rate This bit field is used to generate the transmit and receive bit rate of the SSI. The bit rate is: `BR=SysClk/(CPSDVSR * (1 + SCR))` where `CPSDVSR` is an even value from 2-254 programmed in the **SSICPSR** register, and `SCR` is a value from 0-255.
7	SPH	R/W	0	SSI Serial Clock Phase This bit is only applicable to the Freescale SPI Format. The `SPH` control bit selects the clock edge that captures data and allows it to change state. This bit has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. Value Description 0 Data is captured on the first clock edge transition. 1 Data is captured on the second clock edge transition.
6	SPO	R/W	0	SSI Serial Clock Polarity Value Description 0 A steady state Low value is placed on the `SSIClk` pin. 1 A steady state High value is placed on the `SSIClk` pin when data is not being transferred.

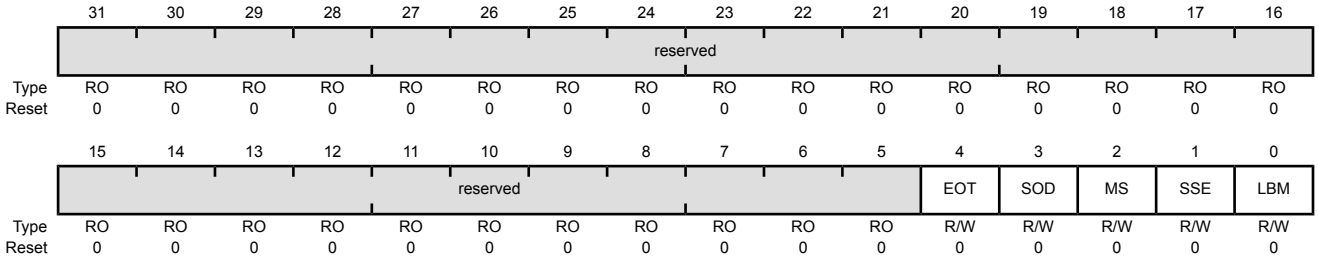
Bit/Field	Name	Type	Reset	Description
5:4	FRF	R/W	0x0	SSI Frame Format Select Value Frame Format 0x0 Freescale SPI Frame Format 0x1 Texas Instruments Synchronous Serial Frame Format 0x2 MICROWIRE Frame Format 0x3 Reserved
3:0	DSS	R/W	0x0	SSI Data Size Select Value Data Size 0x0-0x2 Reserved 0x3 4-bit data 0x4 5-bit data 0x5 6-bit data 0x6 7-bit data 0x7 8-bit data 0x8 9-bit data 0x9 10-bit data 0xA 11-bit data 0xB 12-bit data 0xC 13-bit data 0xD 14-bit data 0xE 15-bit data 0xF 16-bit data

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

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

SSI Control 1 (SSICR1)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x004
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	EOT	R/W	0	End of Transmission Value Description 0 The TXRIS interrupt indicates that the transmit FIFO is half full or less. 1 The End of Transmit interrupt mode for the TXRIS interrupt is enabled.
3	SOD	R/W	0	SSI Slave Mode Output Disable This bit is relevant only in the Slave mode (MS=1). In multiple-slave systems, it is possible for the SSI master to broadcast a message to all slaves in the system while ensuring that only one slave drives data onto the serial output line. In such systems, the TXD lines from multiple slaves could be tied together. To operate in such a system, the SOD bit can be configured so that the SSI slave does not drive the SSITx pin. Value Description 0 SSI can drive the SSITx output in Slave mode. 1 SSI must not drive the SSITx output in Slave mode.
2	MS	R/W	0	SSI Master/Slave Select This bit selects Master or Slave mode and can be modified only when the SSI is disabled (SSE=0). Value Description 0 The SSI is configured as a master. 1 The SSI is configured as a slave.

Bit/Field	Name	Type	Reset	Description
1	SSE	R/W	0	SSI Synchronous Serial Port Enable Value Description 0 SSI operation is disabled. 1 SSI operation is enabled. **Note:** This bit must be cleared before any control registers are reprogrammed.
0	LBM	R/W	0	SSI Loopback Mode Value Description 0 Normal serial port operation enabled. 1 Output of the transmit serial shift register is connected internally to the input of the receive serial shift register.

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

Important: This register is read-sensitive. See the register description for details.

The **SSIDR** register is 16-bits wide. When the **SSIDR** register is read, the entry in the receive FIFO that is pointed to by the current FIFO read pointer is accessed. When a data value is removed by the SSI receive logic from the incoming data frame, it is placed into the entry in the receive FIFO pointed to by the current FIFO write pointer.

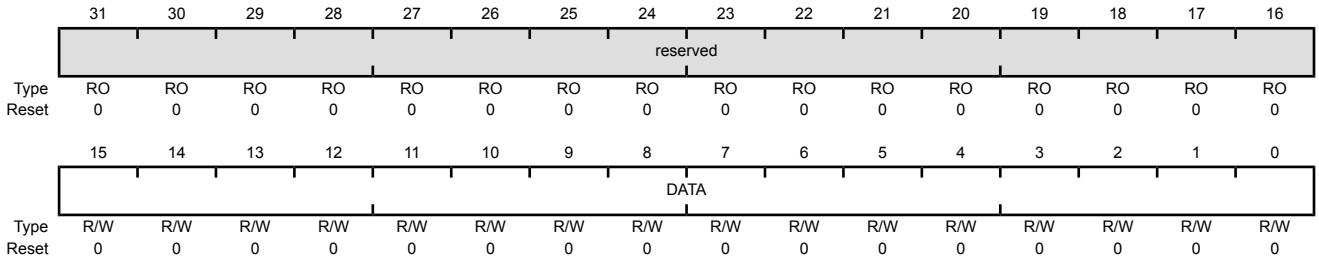
When the **SSIDR** register is written to, the entry in the transmit FIFO that is pointed to by the write pointer is written to. Data values are removed from the transmit FIFO one value at a time by the transmit logic. Each data value is loaded into the transmit serial shifter, then serially shifted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

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

When the SSI is programmed for MICROWIRE frame format, the default size for transmit data is eight bits (the most significant byte is ignored). The receive data size is controlled by the programmer. The transmit FIFO and the receive FIFO are not cleared even when the SSE bit in the **SSICR1** register is cleared, allowing the software to fill the transmit FIFO before enabling the SSI.

SSI Data (SSIDR)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x008
Type R/W, reset 0x0000.0000



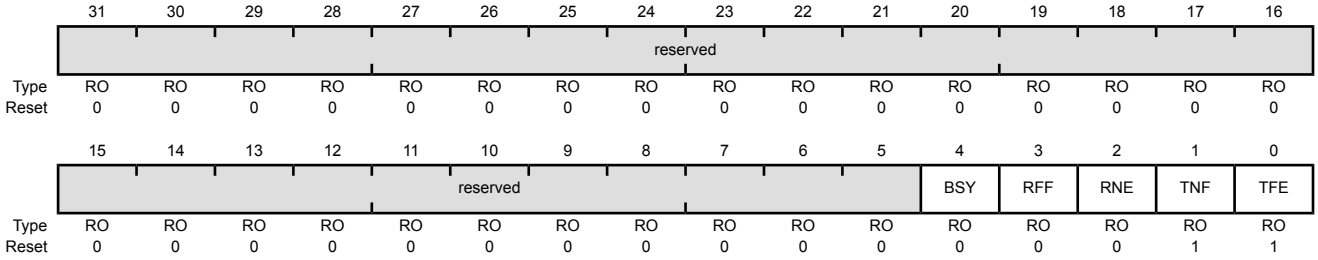
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	DATA	R/W	0x0000	SSI Receive/Transmit Data A read operation reads the receive FIFO. A write operation writes the transmit FIFO. Software must right-justify data when the SSI is programmed for a data size that is less than 16 bits. Unused bits at the top are ignored by the transmit logic. The receive logic automatically right-justifies the data.

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

The **SSISR** register contains bits that indicate the FIFO fill status and the SSI busy status.

SSI Status (SSISR)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x00C
Type RO, reset 0x0000.0003



Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	BSY	RO	0	SSI Busy Bit Value Description 0 The SSI is idle. 1 The SSI is currently transmitting and/or receiving a frame, or the transmit FIFO is not empty.
3	RFF	RO	0	SSI Receive FIFO Full Value Description 0 The receive FIFO is not full. 1 The receive FIFO is full.
2	RNE	RO	0	SSI Receive FIFO Not Empty Value Description 0 The receive FIFO is empty. 1 The receive FIFO is not empty.
1	TNF	RO	1	SSI Transmit FIFO Not Full Value Description 0 The transmit FIFO is full. 1 The transmit FIFO is not full.

Bit/Field	Name	Type	Reset	Description
0	TFE	RO	1	SSI Transmit FIFO Empty

Value	Description
0	The transmit FIFO is not empty.
1	The transmit FIFO is empty.

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

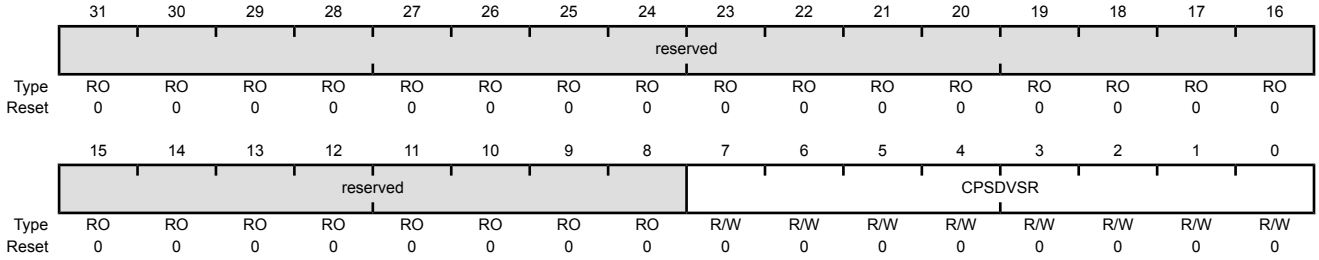
The **SSICPSR** register specifies the division factor which is used to derive the SSIClk from the system clock. The clock is further divided by a value from 1 to 256, which is 1 + SCR. SCR is programmed in the **SSICR0** register. The frequency of the SSIClk is defined by:

```
SSIClk = SysClk / (CPSDVSR * (1 + SCR))
```

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

SSI Clock Prescale (SSICPSR)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x010
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CPSDVSR	R/W	0x00	SSI Clock Prescale Divisor This value must be an even number from 2 to 254, depending on the frequency of SSIClk. The LSB always returns 0 on reads.

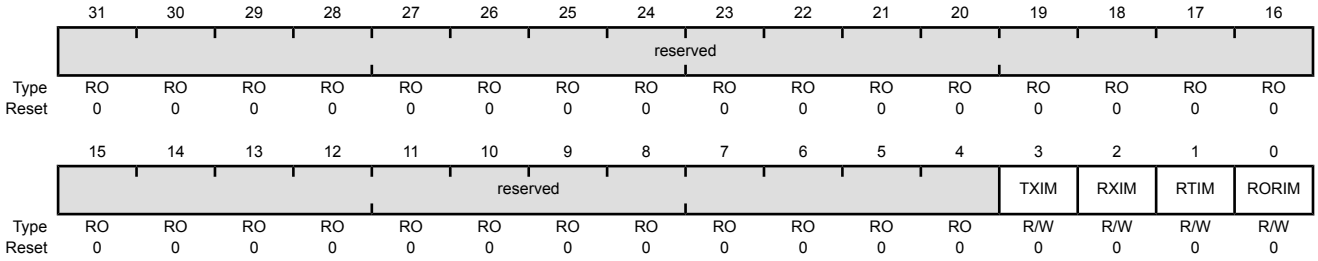
Register 6: SSI Interrupt Mask (SSIIM), offset 0x014

The **SSIIM** register is the interrupt mask set or clear register. It is a read/write register and all bits are cleared on reset.

On a read, this register gives the current value of the mask on the corresponding interrupt. Setting a bit sets the mask, preventing the interrupt from being signaled to the interrupt controller. Clearing a bit clears the corresponding mask, enabling the interrupt to be sent to the interrupt controller.

SSI Interrupt Mask (SSIIM)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x014
Type R/W, reset 0x0000.0000



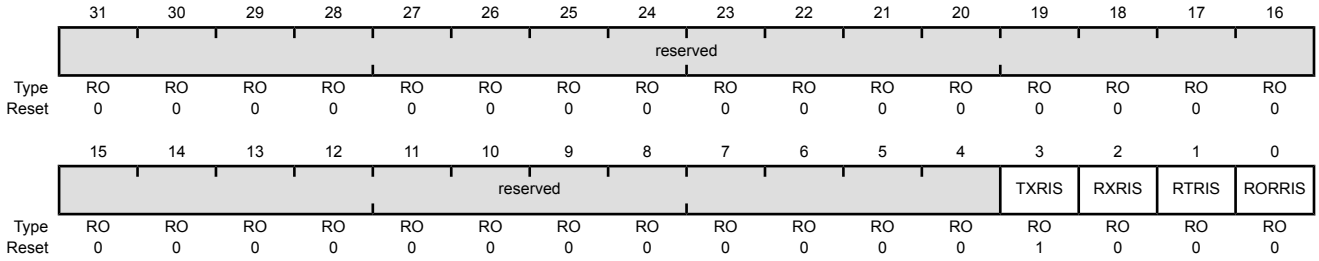
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXIM	R/W	0	SSI Transmit FIFO Interrupt Mask Value Description 0 The transmit FIFO interrupt is masked. 1 The transmit FIFO interrupt is not masked.
2	RXIM	R/W	0	SSI Receive FIFO Interrupt Mask Value Description 0 The receive FIFO interrupt is masked. 1 The receive FIFO interrupt is not masked.
1	RTIM	R/W	0	SSI Receive Time-Out Interrupt Mask Value Description 0 The receive FIFO time-out interrupt is masked. 1 The receive FIFO time-out interrupt is not masked.
0	RORIM	R/W	0	SSI Receive Overrun Interrupt Mask Value Description 0 The receive FIFO overrun interrupt is masked. 1 The receive FIFO overrun interrupt is not masked.

Register 7: SSI Raw Interrupt Status (SSIRIS), offset 0x018

The **SSIRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect.

SSI Raw Interrupt Status (SSIRIS)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x018
Type RO, reset 0x0000.0008



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXRIS	RO	1	SSI Transmit FIFO Raw Interrupt Status Value Description 0 No interrupt. 1 If the EOT bit in the **SSICR1** register is clear, the transmit FIFO is half empty or less. If the EOT bit is set, the transmit FIFO is empty, and the last bit has been transmitted out of the serializer. This bit is cleared when the transmit FIFO is more than half full (if the EOT bit is clear) or when it has any data in it (if the EOT bit is set).
2	RXRIS	RO	0	SSI Receive FIFO Raw Interrupt Status Value Description 0 No interrupt. 1 The receive FIFO is half full or more. This bit is cleared when the receive FIFO is less than half full.
1	RTRIS	RO	0	SSI Receive Time-Out Raw Interrupt Status Value Description 0 No interrupt. 1 The receive time-out has occurred. This bit is cleared when a 1 is written to the RTIC bit in the **SSI Interrupt Clear (SSIICR)** register.

Bit/Field	Name	Type	Reset	Description
0	RORRIS	RO	0	SSI Receive Overrun Raw Interrupt Status

Value	Description
0	No interrupt.
1	The receive FIFO has overflowed

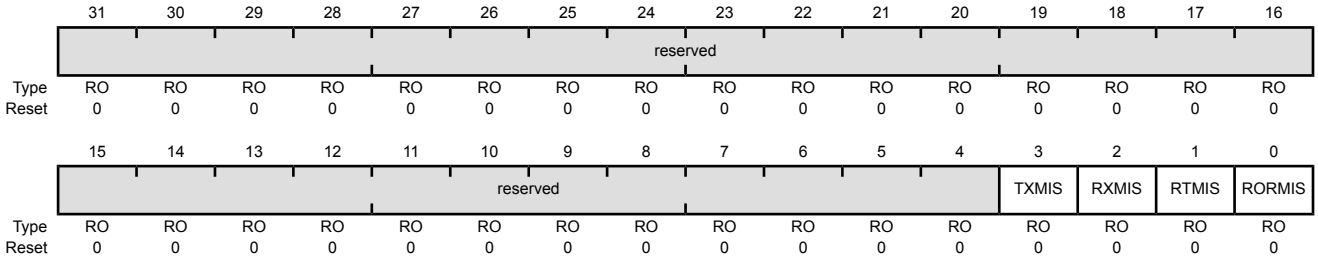
This bit is cleared when a 1 is written to the RORIC bit in the **SSI Interrupt Clear (SSIICR)** register.

Register 8: SSI Masked Interrupt Status (SSIMIS), offset 0x01C

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

SSI Masked Interrupt Status (SSIMIS)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x01C
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXMIS	RO	0	SSI Transmit FIFO Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to the transmit FIFO being half empty or less (if the EOT bit is clear) or due to the transmission of the last data bit (if the EOT bit is set). This bit is cleared when the transmit FIFO is more than half empty (if the EOT bit is clear) or when it has any data in it (if the EOT bit is set).
2	RXMIS	RO	0	SSI Receive FIFO Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to the receive FIFO being half full or more. This bit is cleared when the receive FIFO is less than half full.
1	RTMIS	RO	0	SSI Receive Time-Out Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to the receive time out. This bit is cleared when a 1 is written to the RTIC bit in the **SSI Interrupt Clear (SSIICR)** register.

Bit/Field	Name	Type	Reset	Description
0	RORMIS	RO	0	SSI Receive Overrun Masked Interrupt Status

Value	Description
0	An interrupt has not occurred or is masked.
1	An unmasked interrupt was signaled due to the receive FIFO overflowing.

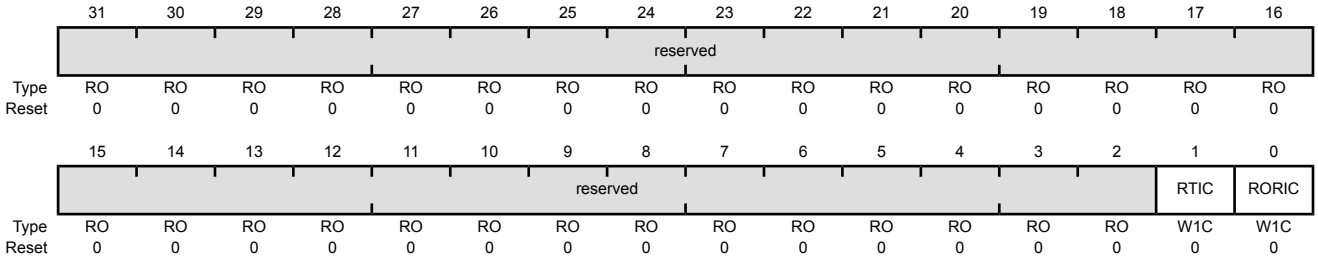
This bit is cleared when a 1 is written to the RORIC bit in the **SSI Interrupt Clear (SSIICR)** register.

Register 9: SSI Interrupt Clear (SSIICR), offset 0x020

The **SSIICR** register is the interrupt clear register. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

SSI Interrupt Clear (SSIICR)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x020
Type W1C, reset 0x0000.0000



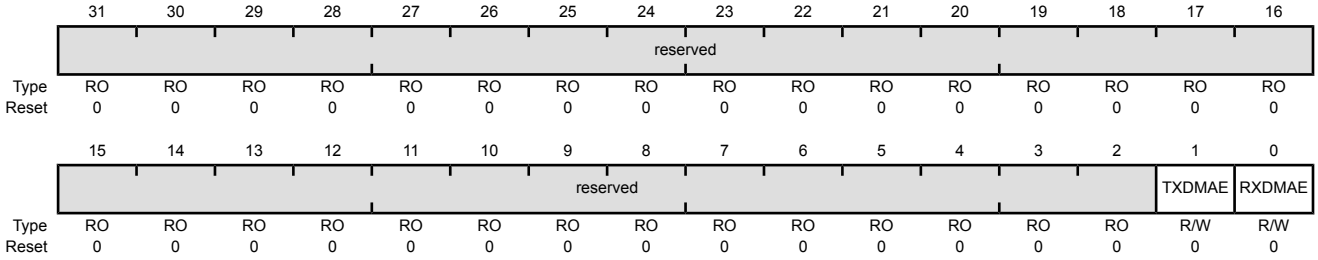
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	RTIC	W1C	0	SSI Receive Time-Out Interrupt Clear Writing a 1 to this bit clears the `RTRIS` bit in the **SSIRIS** register and the `RTMIS` bit in the **SSIMIS** register.
0	RORIC	W1C	0	SSI Receive Overrun Interrupt Clear Writing a 1 to this bit clears the `RORRIS` bit in the **SSIRIS** register and the `RORMIS` bit in the **SSIMIS** register.

Register 10: SSI DMA Control (SSIDMACTL), offset 0x024

The **SSIDMACTL** register is the µDMA control register.

SSI DMA Control (SSIDMACTL)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x024
Type R/W, reset 0x0000.0000



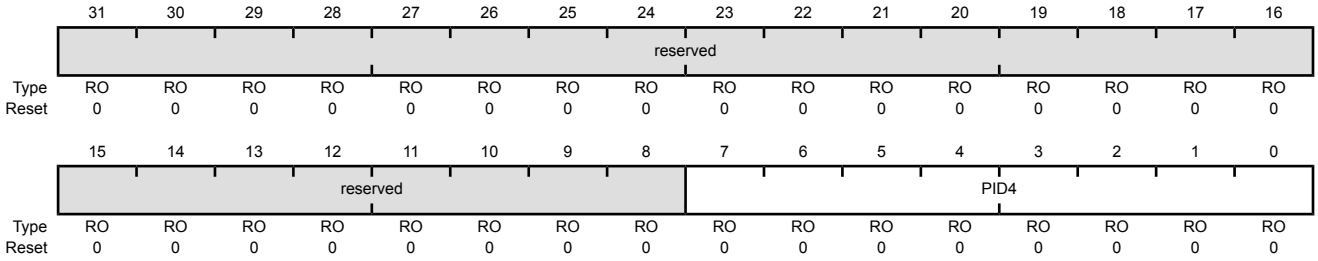
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	TXDMAE	R/W	0	Transmit DMA Enable Value Description 0 µDMA for the transmit FIFO is disabled. 1 µDMA for the transmit FIFO is enabled.
0	RXDMAE	R/W	0	Receive DMA Enable Value Description 0 µDMA for the receive FIFO is disabled. 1 µDMA for the receive FIFO is enabled.

Register 11: SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0

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

SSI Peripheral Identification 4 (SSIPeriphID4)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFD0
Type RO, reset 0x0000.0000



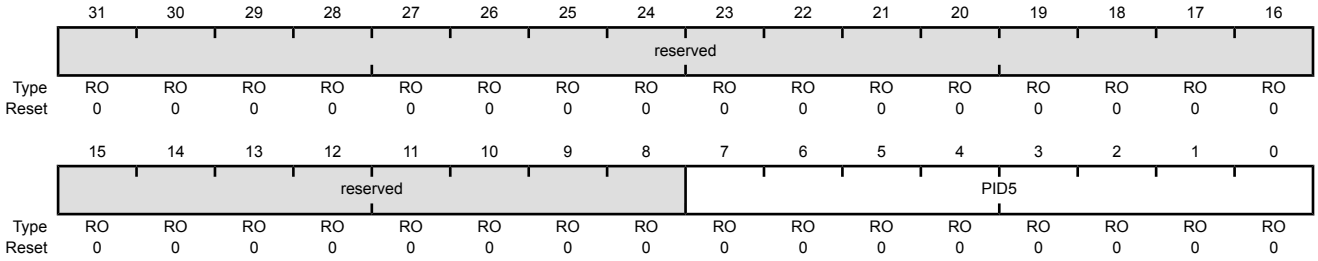
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	SSI Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

Register 12: SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4

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

SSI Peripheral Identification 5 (SSIPeriphID5)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFD4
Type RO, reset 0x0000.0000



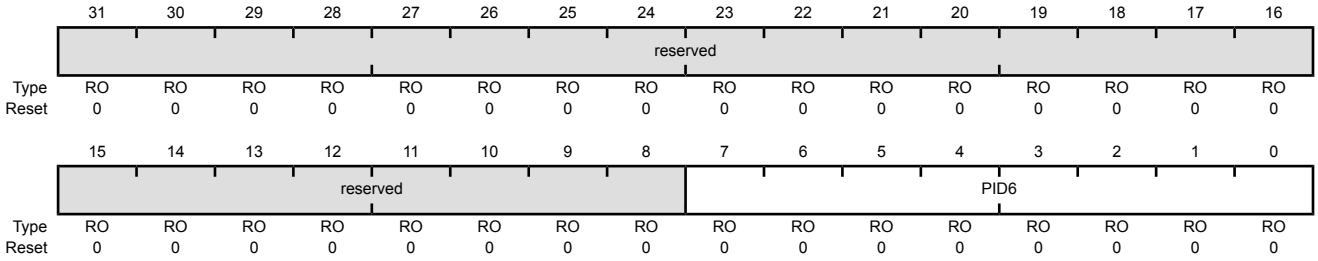
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	SSI Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Register 13: SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8

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

SSI Peripheral Identification 6 (SSIPeriphID6)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFD8
Type RO, reset 0x0000.0000



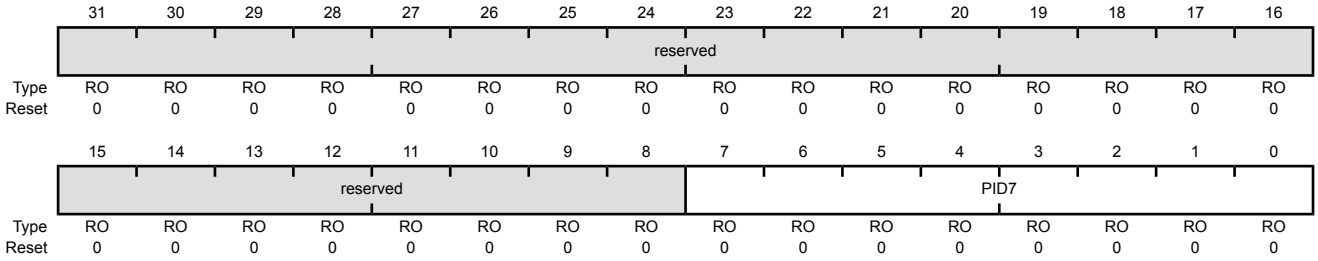
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	SSI Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 14: SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC

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

SSI Peripheral Identification 7 (SSIPeriphID7)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFDC
Type RO, reset 0x0000.0000



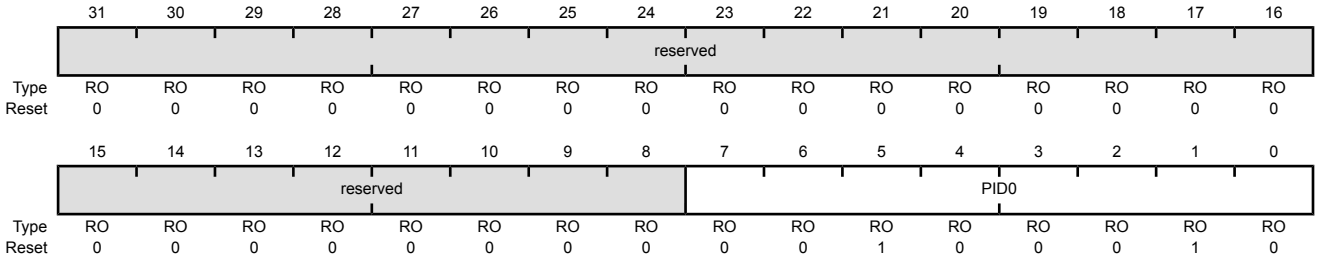
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	SSI Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

Register 15: SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0

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

SSI Peripheral Identification 0 (SSIPeriphID0)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFE0
Type RO, reset 0x0000.0022



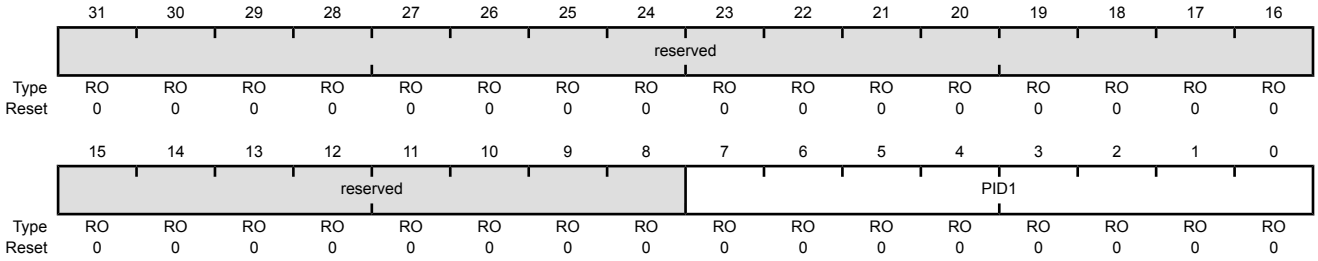
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x22	SSI Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

Register 16: SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4

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

SSI Peripheral Identification 1 (SSIPeriphID1)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFE4
Type RO, reset 0x0000.0000



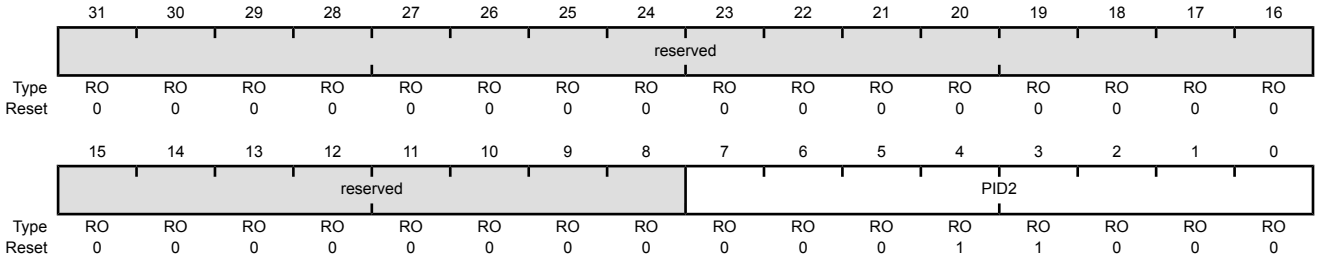
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	SSI Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Register 17: SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8

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

SSI Peripheral Identification 2 (SSIPeriphID2)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFE8
Type RO, reset 0x0000.0018



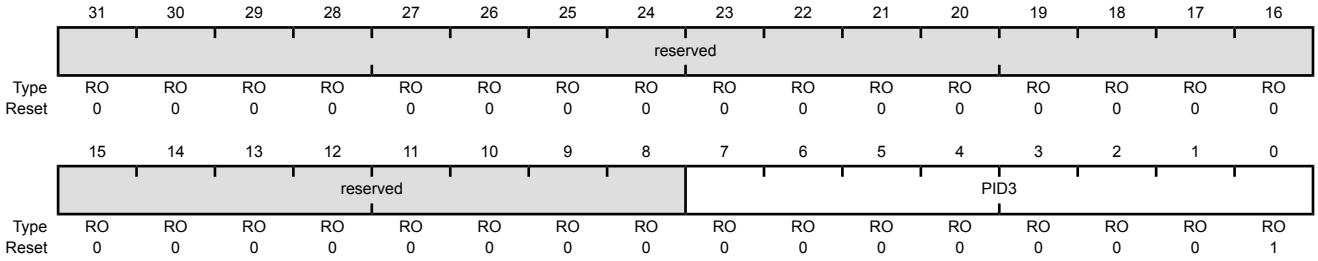
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	SSI Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 18: SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC

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

SSI Peripheral Identification 3 (SSIPeriphID3)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFEC
Type RO, reset 0x0000.0001



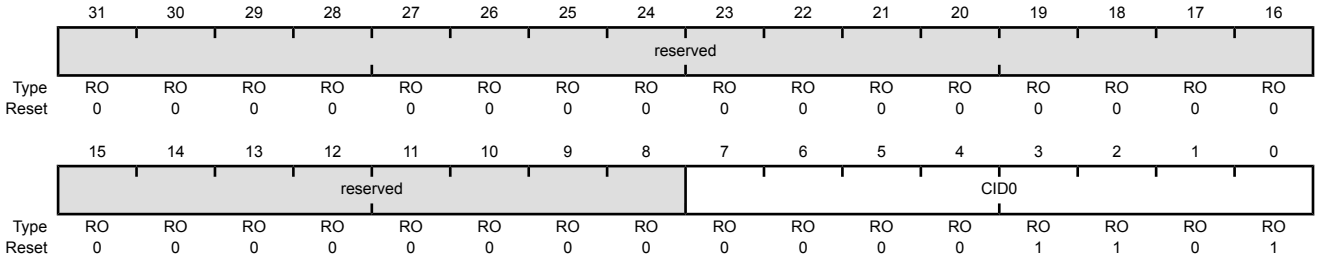
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	SSI Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

Register 19: SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0

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

SSI PrimeCell Identification 0 (SSIPCellID0)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFF0
Type RO, reset 0x0000.000D



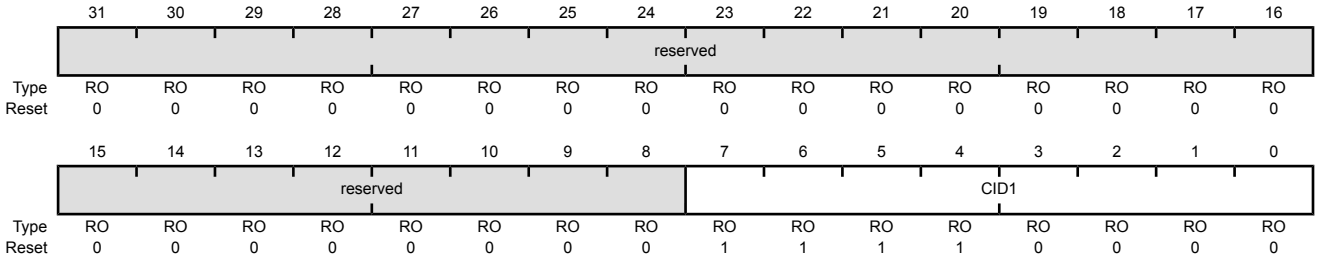
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	SSI PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

Register 20: SSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4

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

SSI PrimeCell Identification 1 (SSIPCellID1)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFF4
Type RO, reset 0x0000.00F0



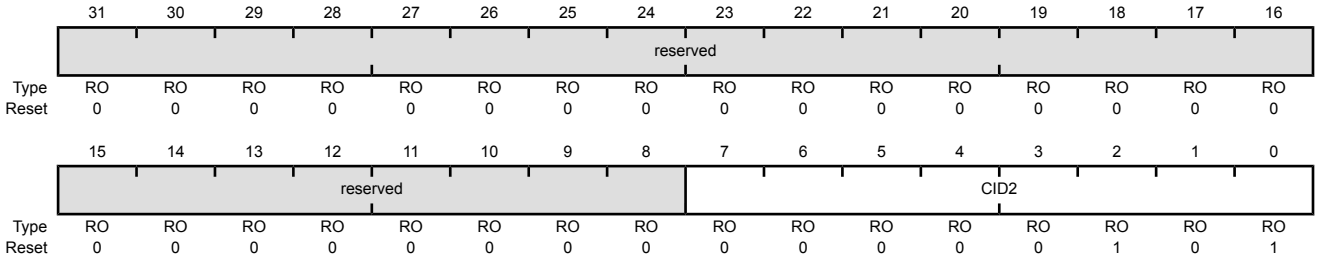
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	SSI PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system.

Register 21: SSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8

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

SSI PrimeCell Identification 2 (SSIPCellID2)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFF8
Type RO, reset 0x0000.0005



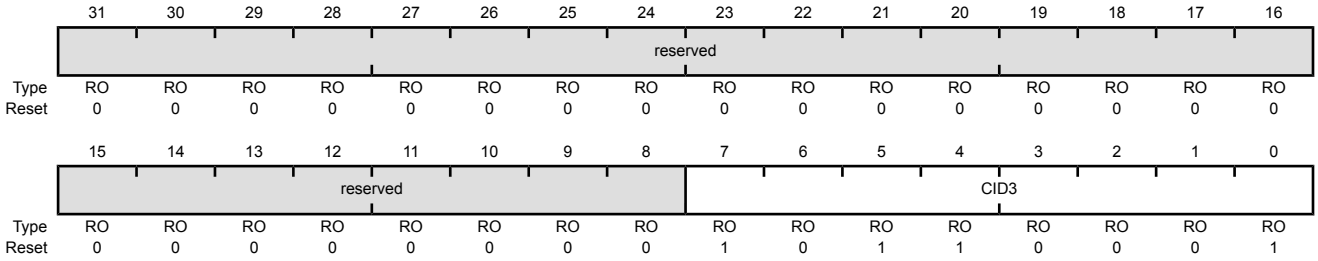
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	SSI PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system.

Register 22: SSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC

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

SSI PrimeCell Identification 3 (SSIPCellID3)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFFC
Type RO, reset 0x0000.00B1



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	SSI PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system.

15 Inter-Integrated Circuit (I^2C) Interface

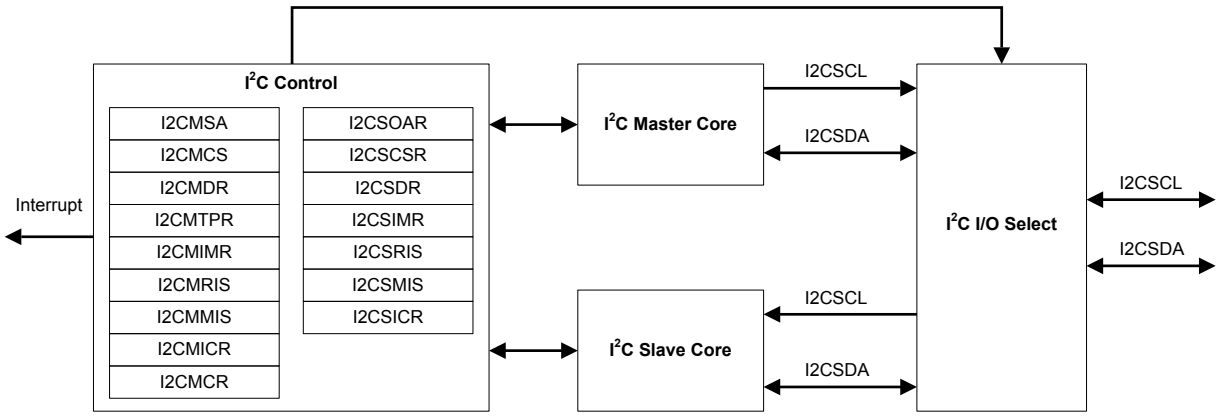
The Inter-Integrated Circuit (I^2C) bus provides bi-directional data transfer through a two-wire design (a serial data line SDA and a serial clock line SCL), and interfaces to external I^2C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and so on. The I^2C bus may also be used for system testing and diagnostic purposes in product development and manufacture. The LM3S9U81 microcontroller includes two I^2C modules, providing the ability to interact (both transmit and receive) with other I^2C devices on the bus.

The Stellaris® LM3S9U81 controller includes two I^2C modules with the following features:

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

15.1 Block Diagram

Figure 15-1. I²C Block Diagram



15.2 Signal Description

The following table lists the external signals of the I²C interface and describes the function of each. The I²C interface signals are alternate functions for some GPIO signals and default to be GPIO signals at reset., with the exception of the I2C0SCL and I2CSDA pins which default to the I²C function. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the I²C signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 419) should be set to choose the I²C function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 437) to assign the I²C signal to the specified GPIO port pin. Note that the I²C pins should be set to open drain using the **GPIO Open Drain Select (GPIOODR)** register. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 395.

Table 15-1. I2C Signals (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
I2C0SCL	72	PB2 (1)	I/O	OD	I²C module 0 clock.
I2C0SDA	65	PB3 (1)	I/O	OD	I²C module 0 data.
I2C1SCL	14 19 26 34	PJ0 (11) PG0 (3) PA0 (8) PA6 (1)	I/O	OD	I²C module 1 clock.
I2C1SDA	18 27 35 87	PG1 (3) PA1 (8) PA7 (1) PJ1 (11)	I/O	OD	I²C module 1 data.

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

Table 15-2. I2C Signals (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
I2C0SCL	A11	PB2 (1)	I/O	OD	I²C module 0 clock.

Table 15-2. I2C Signals (108BGA) *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
I2C0SDA	E11	PB3 (1)	I/O	OD	I2C module 0 data.
I2C1SCL	F3 K1 L3 L6	PJ0 (11) PG0 (3) PA0 (8) PA6 (1)	I/O	OD	I2C module 1 clock.
I2C1SDA	K2 M3 M6 B6	PG1 (3) PA1 (8) PA7 (1) PJ1 (11)	I/O	OD	I2C module 1 data.

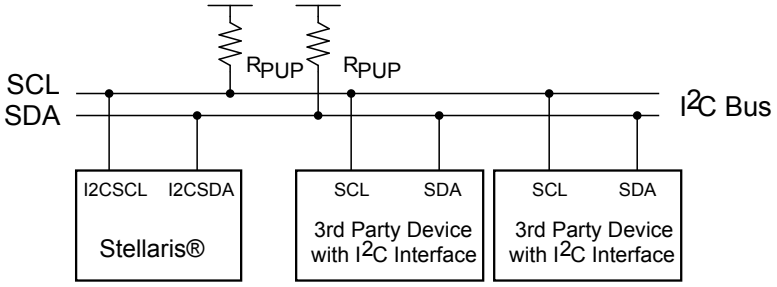
a. The TTL designation indicates the pin has TTL-compatible voltage levels.

15.3 Functional Description

Each I2C module is comprised of both master and slave functions. For proper operation, the SDA and SCL pins must be configured as open-drain signals. A typical I2C bus configuration is shown in Figure 15-2.

See "Inter-Integrated Circuit (I2C) Interface" on page 1210 for I2C timing diagrams.

Figure 15-2. I2C Bus Configuration



15.3.1 I2C Bus Functional Overview

The I2C bus uses only two signals: SDA and SCL, named I2CSDA and I2CSCL on Stellaris microcontrollers. SDA is the bi-directional serial data line and SCL is the bi-directional serial clock line. The bus is considered idle when both lines are High.

Every transaction on the I2C bus is nine bits long, consisting of eight data bits and a single acknowledge bit. The number of bytes per transfer (defined as the time between a valid START and STOP condition, described in "START and STOP Conditions" on page 786) is unrestricted, but each byte has to be followed by an acknowledge bit, and data must be transferred MSB first. When a receiver cannot receive another complete byte, it can hold the clock line SCL Low and force the transmitter into a wait state. The data transfer continues when the receiver releases the clock SCL.

15.3.1.1 START and STOP Conditions

The protocol of the I2C bus defines two states to begin and end a transaction: START and STOP. A High-to-Low transition on the SDA line while the SCL is High is defined as a START condition, and a Low-to-High transition on the SDA line while SCL is High is defined as a STOP condition. The bus is considered busy after a START condition and free after a STOP condition. See Figure 15-3.

Figure 15-3. START and STOP Conditions

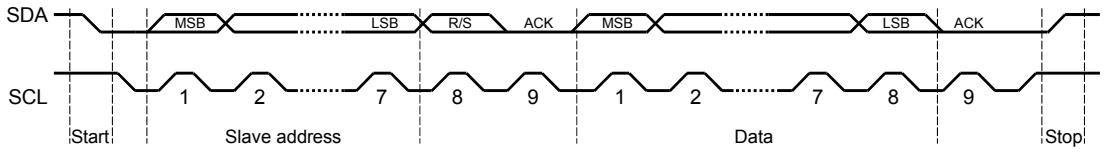
The STOP bit determines if the cycle stops at the end of the data cycle or continues on to a repeated START condition. To generate a single transmit cycle, the **I²C Master Slave Address (I2CMSA)** register is written with the desired address, the R/S bit is cleared, and the Control register is written with ACK=X (0 or 1), STOP=1, START=1, and RUN=1 to perform the operation and stop. When the operation is completed (or aborted due an error), the interrupt pin becomes active and the data may be read from the **I²C Master Data (I2CMDR)** register. When the I²C module operates in Master receiver mode, the ACK bit is normally set causing the I²C bus controller to transmit an acknowledge automatically after each byte. This bit must be cleared when the I²C bus controller requires no further data to be transmitted from the slave transmitter.

When operating in slave mode, two bits in the **I²C Slave Raw Interrupt Status (I2CSRIS)** register indicate detection of start and stop conditions on the bus; while two bits in the **I²C Slave Masked Interrupt Status (I2CSMIS)** register allow start and stop conditions to be promoted to controller interrupts (when interrupts are enabled).

15.3.1.2 Data Format with 7-Bit Address

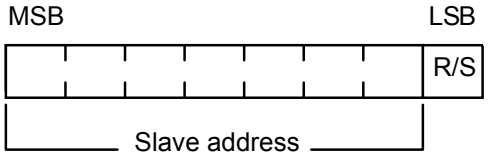
Data transfers follow the format shown in Figure 15-4. After the START condition, a slave address is transmitted. This address is 7-bits long followed by an eighth bit, which is a data direction bit (R/S bit in the **I2CMSA** register). If the R/S bit is clear, it indicates a transmit operation (send), and if it is set, it indicates a request for data (receive). A data transfer is always terminated by a STOP condition generated by the master, however, a master can initiate communications with another device on the bus by generating a repeated START condition and addressing another slave without first generating a STOP condition. Various combinations of receive/transmit formats are then possible within a single transfer.

Figure 15-4. Complete Data Transfer with a 7-Bit Address



The first seven bits of the first byte make up the slave address (see Figure 15-5). The eighth bit determines the direction of the message. A zero in the R/S position of the first byte means that the master transmits (sends) data to the selected slave, and a one in this position means that the master receives data from the slave.

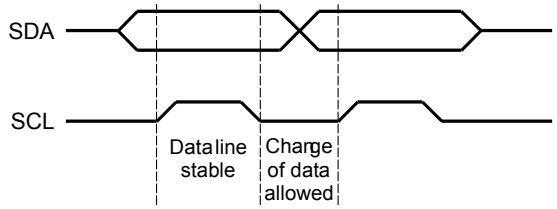
Figure 15-5. R/S Bit in First Byte



15.3.1.3 Data Validity

The data on the SDA line must be stable during the high period of the clock, and the data line can only change when SCL is Low (see Figure 15-6).

Figure 15-6. Data Validity During Bit Transfer on the I2C Bus



15.3.1.4 Acknowledge

All bus transactions have a required acknowledge clock cycle that is generated by the master. During the acknowledge cycle, the transmitter (which can be the master or slave) releases the SDA line. To acknowledge the transaction, the receiver must pull down SDA during the acknowledge clock cycle. The data transmitted out by the receiver during the acknowledge cycle must comply with the data validity requirements described in “Data Validity” on page 788.

When a slave receiver does not acknowledge the slave address, SDA must be left High by the slave so that the master can generate a STOP condition and abort the current transfer. If the master device is acting as a receiver during a transfer, it is responsible for acknowledging each transfer made by the slave. Because the master controls the number of bytes in the transfer, it signals the end of data to the slave transmitter by not generating an acknowledge on the last data byte. The slave transmitter must then release SDA to allow the master to generate the STOP or a repeated START condition.

15.3.1.5 Arbitration

A master may start a transfer only if the bus is idle. It's possible for two or more masters to generate a START condition within minimum hold time of the START condition. In these situations, an arbitration scheme takes place on the SDA line, while SCL is High. During arbitration, the first of the competing master devices to place a '1' (High) on SDA while another master transmits a '0' (Low) switches off its data output stage and retires until the bus is idle again.

Arbitration can take place over several bits. Its first stage is a comparison of address bits, and if both masters are trying to address the same device, arbitration continues on to the comparison of data bits.

15.3.2 Available Speed Modes

The I2C bus can run in either Standard mode (100 kbps) or Fast mode (400 kbps). The selected mode should match the speed of the other I2C devices on the bus.

15.3.2.1 Standard and Fast Modes

Standard and Fast modes are selected using a value in the **I²C Master Timer Period (I2CMTPR)** register that results in an SCL frequency of 100 kbps for Standard mode.

The I²C clock rate is determined by the parameters *CLK_PRD*, *TIMER_PRD*, *SCL_LP*, and *SCL_HP* where:

CLK_PRD is the system clock period

SCL_LP is the low phase of SCL (fixed at 6)

SCL_HP is the high phase of SCL (fixed at 4)

TIMER_PRD is the programmed value in the **I2CMTPR** register (see page 808).

The I²C clock period is calculated as follows:

SCL_PERIOD = 2 × (1 + *TIMER_PRD*) × (*SCL_LP* + *SCL_HP*) × *CLK_PRD*

For example:

CLK_PRD = 50 ns

TIMER_PRD = 2

SCL_LP=6

SCL_HP=4

yields a SCL frequency of:

1/*SCL_PERIOD* = 333 Khz

Table 15-3 gives examples of the timer periods that should be used to generate SCL frequencies based on various system clock frequencies.

Table 15-3. Examples of I²C Master Timer Period versus Speed Mode

System Clock	Timer Period	Standard Mode	Timer Period	Fast Mode
4 MHz	0x01	100 Kbps	-	-
6 MHz	0x02	100 Kbps	-	-
12.5 MHz	0x06	89 Kbps	0x01	312 Kbps
16.7 MHz	0x08	93 Kbps	0x02	278 Kbps
20 MHz	0x09	100 Kbps	0x02	333 Kbps
25 MHz	0x0C	96.2 Kbps	0x03	312 Kbps
33 MHz	0x10	97.1 Kbps	0x04	330 Kbps
40 MHz	0x13	100 Kbps	0x04	400 Kbps
50 MHz	0x18	100 Kbps	0x06	357 Kbps
80 MHz	0x27	100 Kbps	0x09	400 Kbps

15.3.3 Interrupts

The I²C can generate interrupts when the following conditions are observed:

- Master transaction completed
- Master arbitration lost

- Master transaction error
- Slave transaction received
- Slave transaction requested
- Stop condition on bus detected
- Start condition on bus detected

The I2C master and I2C slave modules have separate interrupt signals. While both modules can generate interrupts for multiple conditions, only a single interrupt signal is sent to the interrupt controller.

15.3.3.1 I2C Master Interrupts

The I2C master module generates an interrupt when a transaction completes (either transmit or receive), when arbitration is lost, or when an error occurs during a transaction. To enable the I2C master interrupt, software must set the IM bit in the **I2C Master Interrupt Mask (I2CMIMR)** register. When an interrupt condition is met, software must check the ERROR and ARBLST bits in the **I2C Master Control/Status (I2CMCS)** register to verify that an error didn't occur during the last transaction and to ensure that arbitration has not been lost. An error condition is asserted if the last transaction wasn't acknowledged by the slave. If an error is not detected and the master has not lost arbitration, the application can proceed with the transfer. The interrupt is cleared by writing a 1 to the IC bit in the **I2C Master Interrupt Clear (I2CMICR)** register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I2C Master Raw Interrupt Status (I2CMRIS)** register.

15.3.3.2 I2C Slave Interrupts

The slave module can generate an interrupt when data has been received or requested. This interrupt is enabled by setting the DATAIM bit in the **I2C Slave Interrupt Mask (I2CSIMR)** register. Software determines whether the module should write (transmit) or read (receive) data from the **I2C Slave Data (I2CSDR)** register, by checking the RREQ and TREQ bits of the **I2C Slave Control/Status (I2CSCSR)** register. If the slave module is in receive mode and the first byte of a transfer is received, the FBR bit is set along with the RREQ bit. The interrupt is cleared by setting the DATAIC bit in the **I2C Slave Interrupt Clear (I2CSICR)** register.

In addition, the slave module can generate an interrupt when a start and stop condition is detected. These interrupts are enabled by setting the STARTIM and STOPIM bits of the **I2C Slave Interrupt Mask (I2CSIMR)** register and cleared by writing a 1 to the STOPIC and STARTIC bits of the **I2C Slave Interrupt Clear (I2CSICR)** register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I2C Slave Raw Interrupt Status (I2CSRIS)** register.

15.3.4 Loopback Operation

The I2C modules can be placed into an internal loopback mode for diagnostic or debug work by setting the LPBK bit in the **I2C Master Configuration (I2CMCR)** register. In loopback mode, the SDA and SCL signals from the master and slave modules are tied together.

15.3.5 Command Sequence Flow Charts

This section details the steps required to perform the various I^2C transfer types in both master and slave mode.

15.3.5.1 I^2C Master Command Sequences

The figures that follow show the command sequences available for the I^2C master.

Figure 15-7. Master Single TRANSMIT

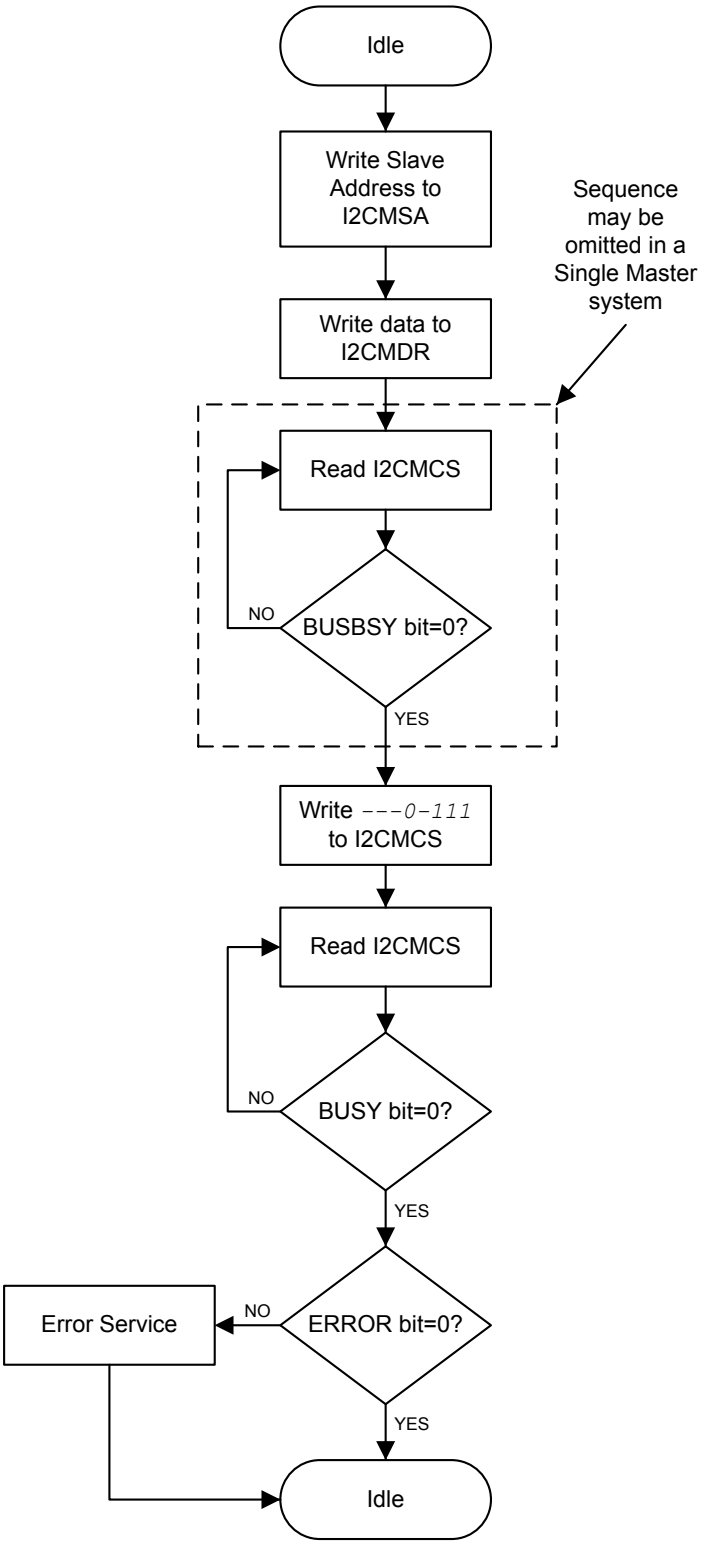


Figure 15-8. Master Single RECEIVE

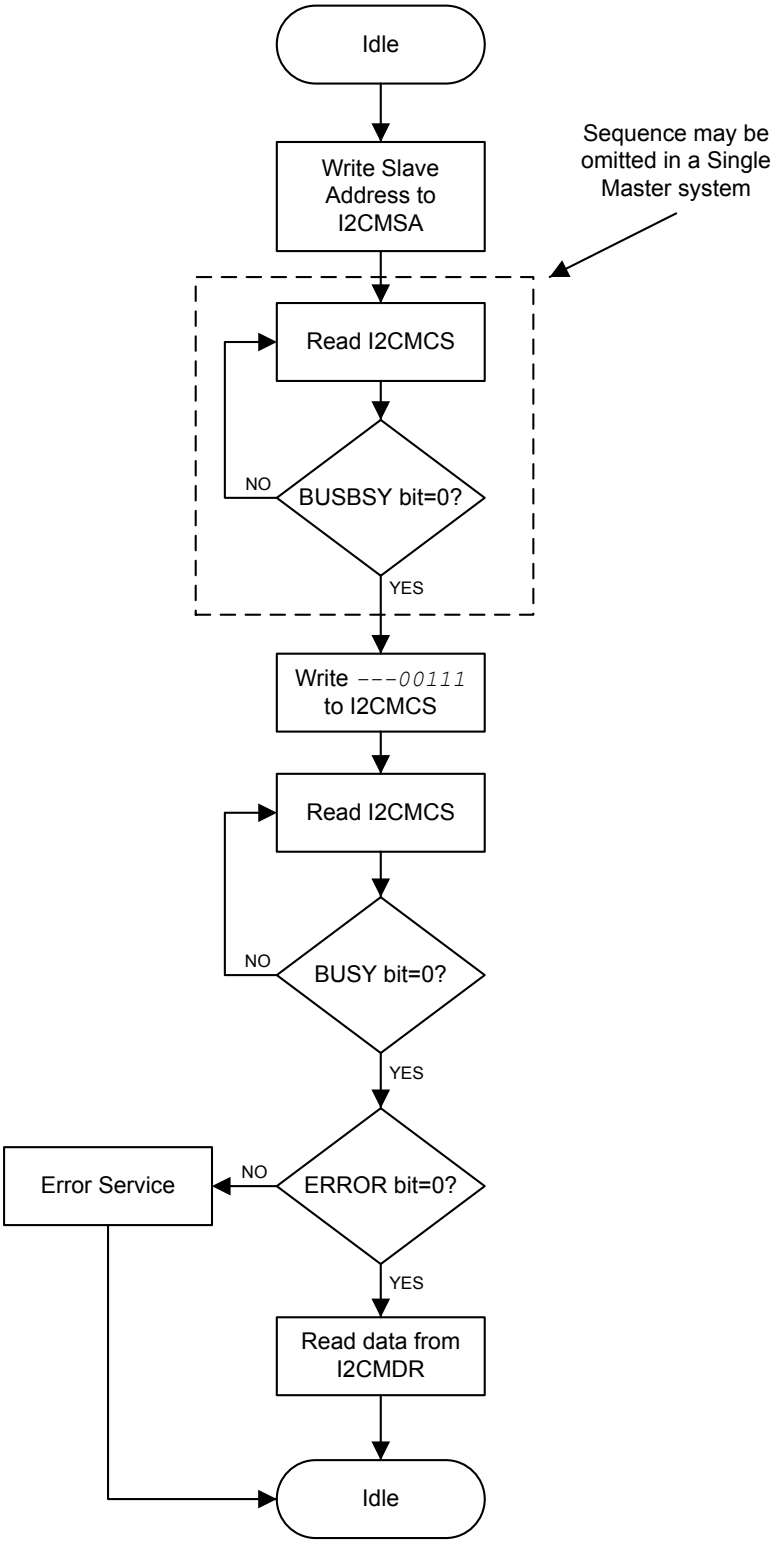


Figure 15-9. Master TRANSMIT with Repeated START

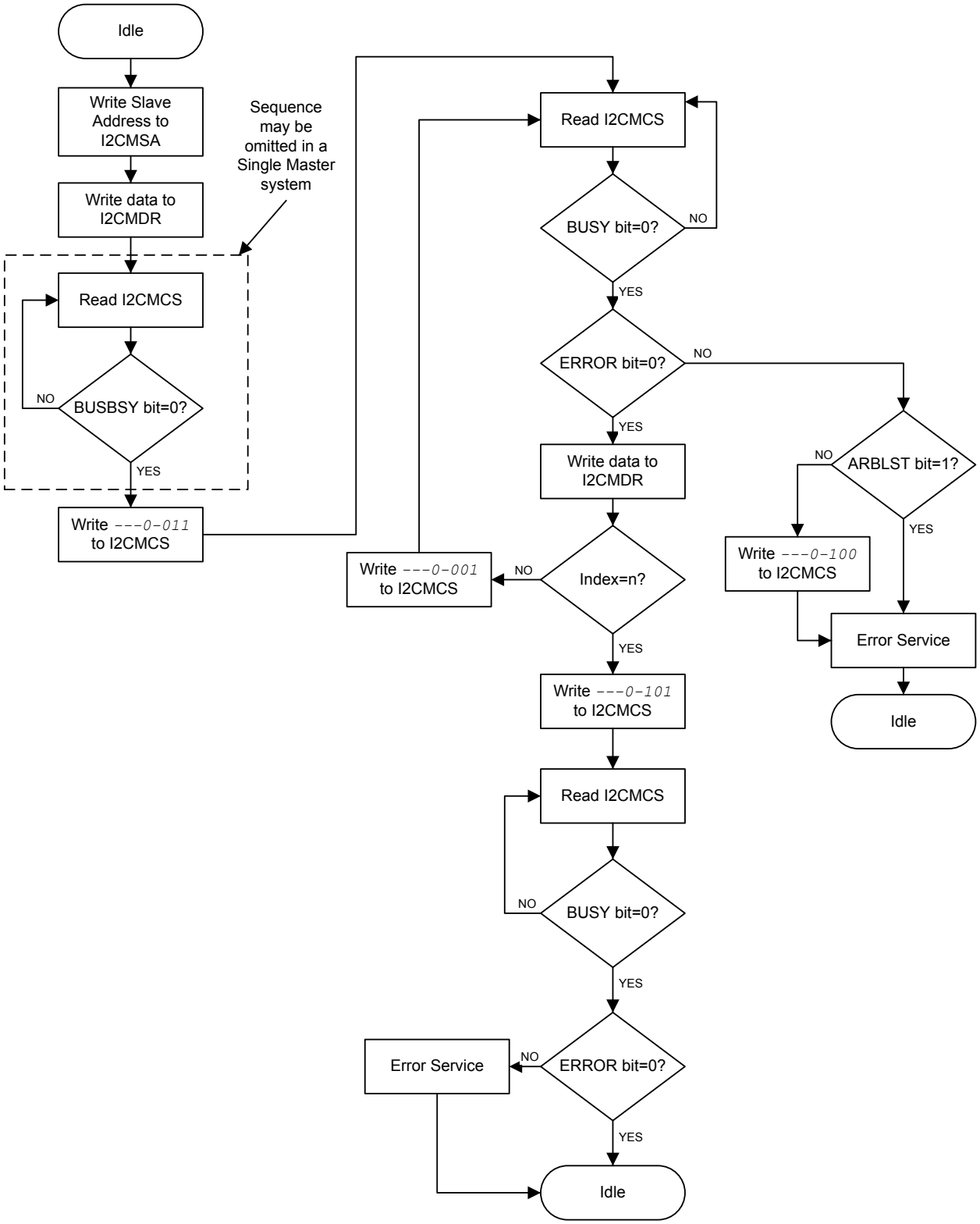


Figure 15-10. Master RECEIVE with Repeated START

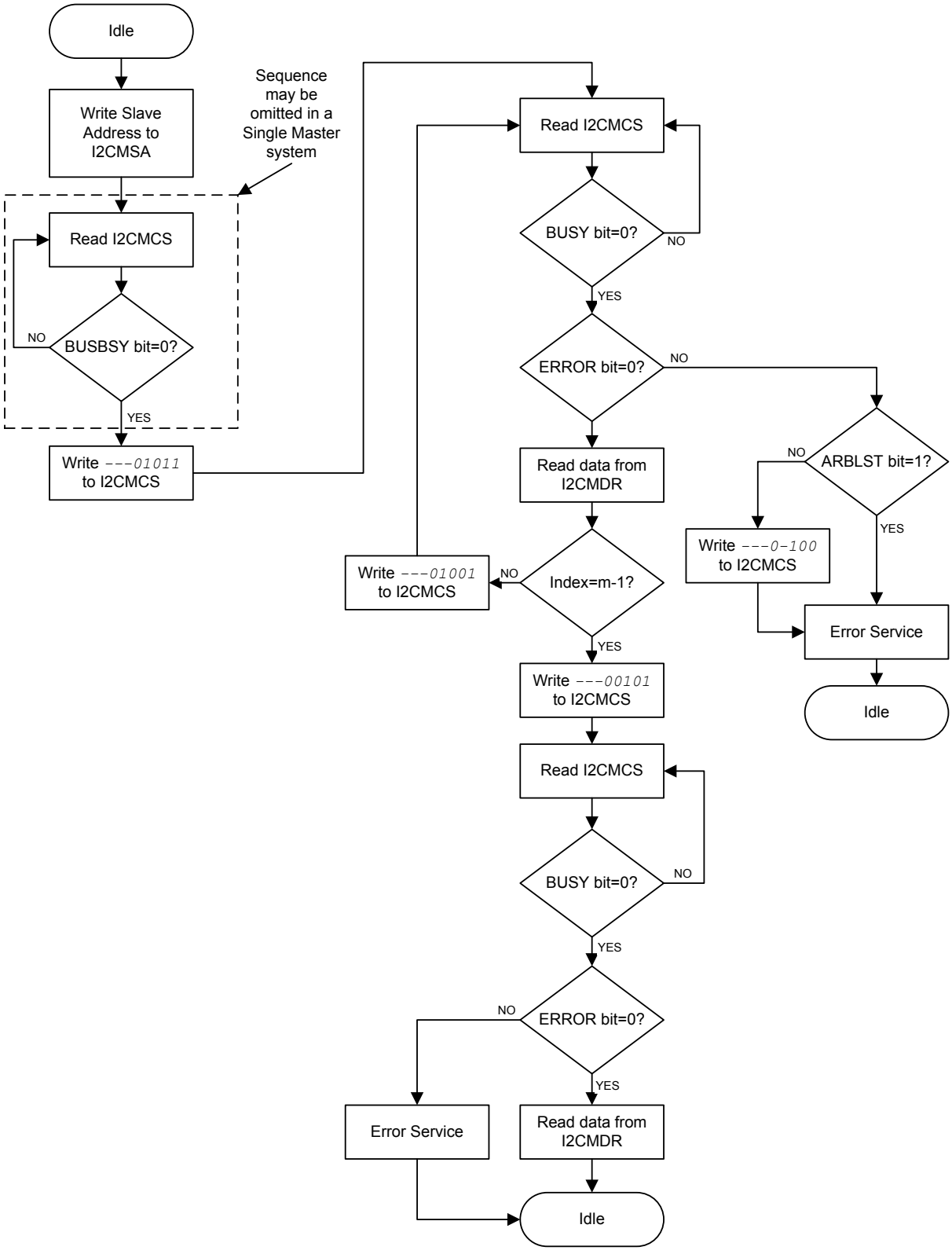


Figure 15-11. Master RECEIVE with Repeated START after TRANSMIT with Repeated START

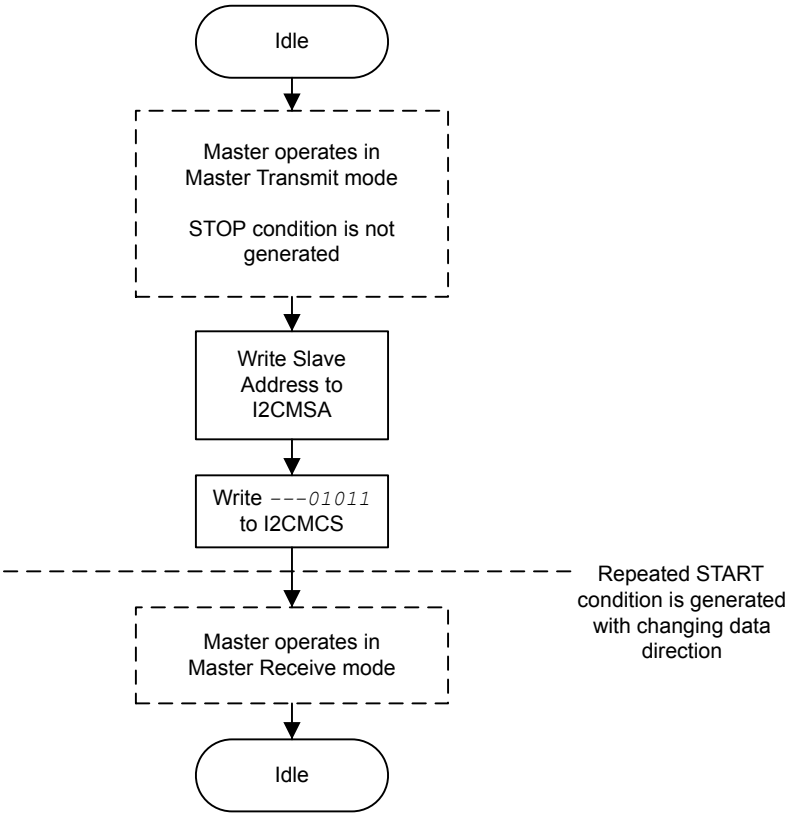
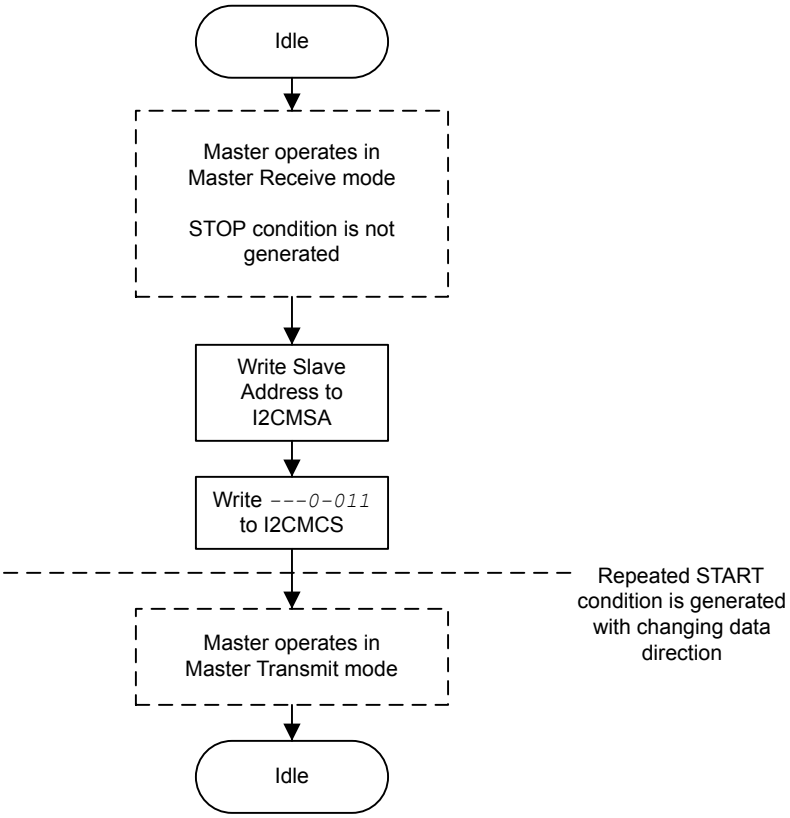


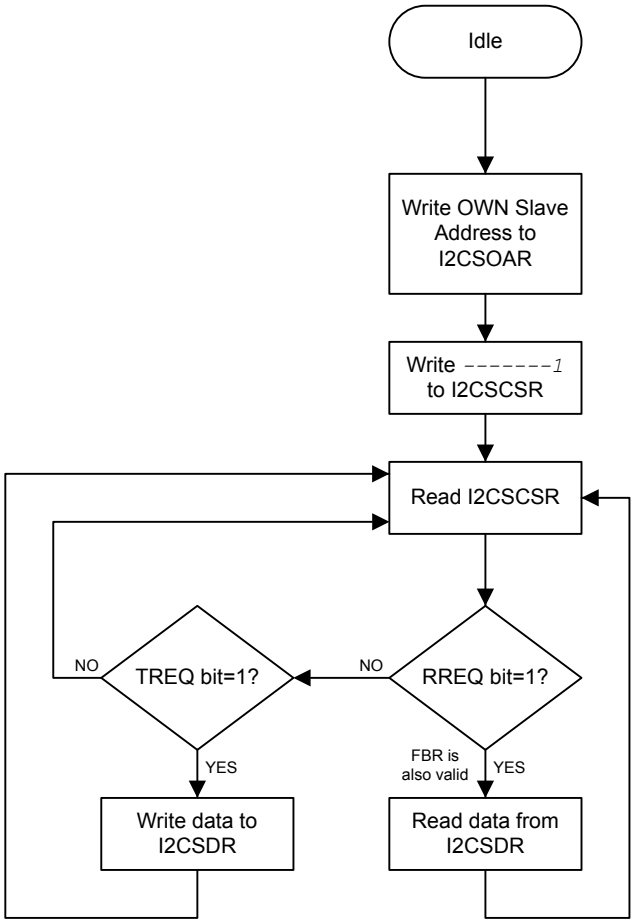
Figure 15-12. Master TRANSMIT with Repeated START after RECEIVE with Repeated START



15.3.5.2 I²C Slave Command Sequences

Figure 15-13 on page 798 presents the command sequence available for the I²C slave.

Figure 15-13. Slave Command Sequence



15.4 Initialization and Configuration

The following example shows how to configure the I2C module to transmit a single byte as a master. This assumes the system clock is 20 MHz.

1. Enable the I2C clock by writing a value of 0x0000.1000 to the **RCGC1** register in the System Control module (see page 263).
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register in the System Control module (see page 272). To find out which GPIO port to enable, refer to Table 22-5 on page 1151.
3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register (see page 419). To determine which GPIOs to configure, see Table 22-4 on page 1144.
4. Enable the I2C pins for open-drain operation. See page 424.
5. Configure the PMCn fields in the **GPIOPCTL** register to assign the I2C signals to the appropriate pins. See page 437 and Table 22-5 on page 1151.
6. Initialize the I2C Master by writing the **I2CMCR** register with a value of 0x0000.0010.

7. Set the desired SCL clock speed of 100 Kbps by writing the **I2CMTPR** register with the correct value. The value written to the **I2CMTPR** register represents the number of system clock periods in one SCL clock period. The TPR value is determined by the following equation:

   ```
   TPR = (System Clock/(2*(SCL_LP + SCL_HP)*SCL_CLK))-1;
   TPR = (20MHz/(2*(6+4)*100000))-1;
   TPR = 9
   ```

   Write the **I2CMTPR** register with the value of 0x0000.0009.

8. Specify the slave address of the master and that the next operation is a Transmit by writing the **I2CMSA** register with a value of 0x0000.0076. This sets the slave address to 0x3B.

9. Place data (byte) to be transmitted in the data register by writing the **I2CMDR** register with the desired data.

10. Initiate a single byte transmit of the data from Master to Slave by writing the **I2CMCS** register with a value of 0x0000.0007 (STOP, START, RUN).

11. Wait until the transmission completes by polling the **I2CMCS** register's BUSBSY bit until it has been cleared.

12. Check the ERROR bit in the **I2CMCS** register to confirm the transmit was acknowledged.

15.5 Register Map

Table 15-4 on page 799 lists the I²C registers. All addresses given are relative to the I²C base address:

- I²C 0: 0x4002.0000
- I²C 1: 0x4002.1000

Note that the I²C module clock must be enabled before the registers can be programmed (see page 263). There must be a delay of 3 system clocks after the I²C module clock is enabled before any I²C module registers are accessed.

The hw_i2c.h file in the StellarisWare® Driver Library uses a base address of 0x800 for the I²C slave registers. Be aware when using registers with offsets between 0x800 and 0x818 that StellarisWare uses an offset between 0x000 and 0x018 with the slave base address.

Table 15-4. Inter-Integrated Circuit (I²C) Interface Register Map

Offset	Name	Type	Reset	Description	See page
I²C Master					
0x000	I2CMSA	R/W	0x0000.0000	I2C Master Slave Address	801
0x004	I2CMCS	R/W	0x0000.0020	I2C Master Control/Status	802
0x008	I2CMDR	R/W	0x0000.0000	I2C Master Data	807
0x00C	I2CMTPR	R/W	0x0000.0001	I2C Master Timer Period	808
0x010	I2CMIMR	R/W	0x0000.0000	I2C Master Interrupt Mask	809
0x014	I2CMRIS	RO	0x0000.0000	I2C Master Raw Interrupt Status	810

Table 15-4. Inter-Integrated Circuit (I2C) Interface Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x018	I2CMMIS	RO	0x0000.0000	I2C Master Masked Interrupt Status	811
0x01C	I2CMICR	WO	0x0000.0000	I2C Master Interrupt Clear	812
0x020	I2CMCR	R/W	0x0000.0000	I2C Master Configuration	813
I2C Slave					
0x800	I2CSOAR	R/W	0x0000.0000	I2C Slave Own Address	814
0x804	I2CSCSR	RO	0x0000.0000	I2C Slave Control/Status	815
0x808	I2CSDR	R/W	0x0000.0000	I2C Slave Data	817
0x80C	I2CSIMR	R/W	0x0000.0000	I2C Slave Interrupt Mask	818
0x810	I2CSRIS	RO	0x0000.0000	I2C Slave Raw Interrupt Status	819
0x814	I2CSMIS	RO	0x0000.0000	I2C Slave Masked Interrupt Status	820
0x818	I2CSICR	WO	0x0000.0000	I2C Slave Interrupt Clear	821

15.6 Register Descriptions (I2C Master)

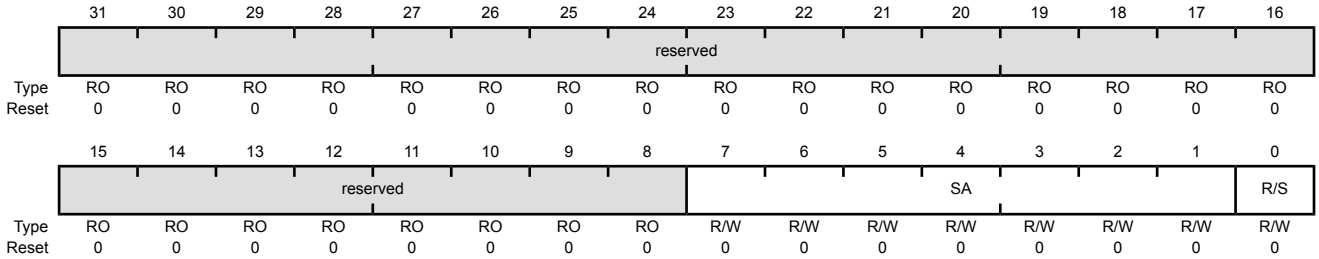
The remainder of this section lists and describes the I2C master registers, in numerical order by address offset.

Register 1: I²C Master Slave Address (I2CMSA), offset 0x000

This register consists of eight bits: seven address bits (A6-A0), and a Receive/Send bit, which determines if the next operation is a Receive (High), or Transmit (Low).

I2C Master Slave Address (I2CMSA)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x000
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:1	SA	R/W	0x00	I²C Slave Address This field specifies bits A6 through A0 of the slave address.
0	R/S	R/W	0	Receive/Send The R/S bit specifies if the next operation is a Receive (High) or Transmit (Low). Value Description 0 Transmit 1 Receive

Register 2: I2C Master Control/Status (I2CMCS), offset 0x004

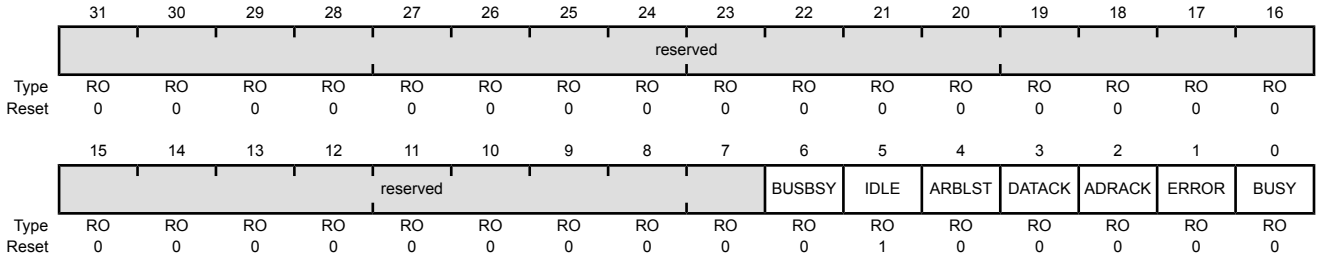
This register accesses status bits when read and control bits when written. When read, the status register indicates the state of the I2C bus controller. When written, the control register configures the I2C controller operation.

The START bit generates the START or REPEATED START condition. The STOP bit determines if the cycle stops at the end of the data cycle or continues on to a repeated START condition. To generate a single transmit cycle, the **I2C Master Slave Address (I2CMSA)** register is written with the desired address, the R/S bit is cleared, and this register is written with ACK=X (0 or 1), STOP=1, START=1, and RUN=1 to perform the operation and stop. When the operation is completed (or aborted due an error), an interrupt becomes active and the data may be read from the **I2CMDR** register. When the I2C module operates in Master receiver mode, the ACK bit is normally set, causing the I2C bus controller to transmit an acknowledge automatically after each byte. This bit must be cleared when the I2C bus controller requires no further data to be transmitted from the slave transmitter.

Read-Only Status Register

I2C Master Control/Status (I2CMCS)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x004
Type RO, reset 0x0000.0020



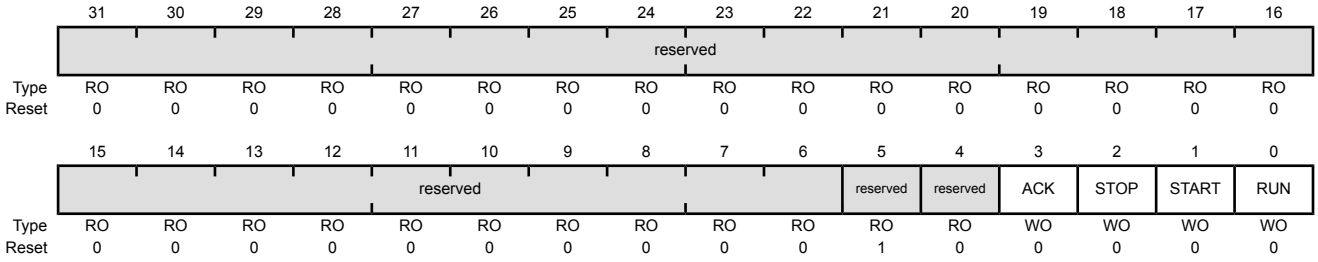
Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	BUSBSY	RO	0	Bus Busy Value Description 0 The I2C bus is idle. 1 The I2C bus is busy. The bit changes based on the START and STOP conditions.
5	IDLE	RO	1	I2C Idle Value Description 0 The I2C controller is not idle. 1 The I2C controller is idle.

Bit/Field	Name	Type	Reset	Description
4	ARBLST	RO	0	Arbitration Lost Value Description 0 The I^2C controller won arbitration. 1 The I^2C controller lost arbitration.
3	DATACK	RO	0	Acknowledge Data Value Description 0 The transmitted data was acknowledged 1 The transmitted data was not acknowledged.
2	ADRACK	RO	0	Acknowledge Address Value Description 0 The transmitted address was acknowledged 1 The transmitted address was not acknowledged.
1	ERROR	RO	0	Error Value Description 0 No error was detected on the last operation. 1 An error occurred on the last operation. The error can be from the slave address not being acknowledged or the transmit data not being acknowledged.
0	BUSY	RO	0	I^2C Busy Value Description 0 The controller is idle. 1 The controller is busy. When the BUSY bit is set, the other status bits are not valid.

Write-Only Control Register

I2C Master Control/Status (I2CMCS)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x004
Type WO, reset 0x0000.0020



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	reserved	RO	1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	ACK	WO	0	Data Acknowledge Enable Value Description 0 The received data byte is not acknowledged automatically by the master. 1 The received data byte is acknowledged automatically by the master. See field decoding in Table 15-5 on page 805.
2	STOP	WO	0	Generate STOP Value Description 0 The controller does not generate the STOP condition. 1 The controller generates the STOP condition. See field decoding in Table 15-5 on page 805.
1	START	WO	0	Generate START Value Description 0 The controller does not generate the START condition. 1 The controller generates the START or repeated START condition. See field decoding in .
0	RUN	WO	0	I^2C Master Enable Value Description 0 The master is disabled. 1 The master is enabled to transmit or receive data. See field decoding in Table 15-5 on page 805.

Table 15-5. Write Field Decoding for I2CMCS[3:0] Field

Current State	I2CMSA[0]	I2CMCS[3:0]				Description
	R/S	ACK	STOP	START	RUN	
Idle	0	X[a]	0	1	1	START condition followed by TRANSMIT (master goes to the Master Transmit state).
	0	X	1	1	1	START condition followed by a TRANSMIT and STOP condition (master remains in Idle state).
	1	0	0	1	1	START condition followed by RECEIVE operation with negative ACK (master goes to the Master Receive state).
	1	0	1	1	1	START condition followed by RECEIVE and STOP condition (master remains in Idle state).
	1	1	0	1	1	START condition followed by RECEIVE (master goes to the Master Receive state).
	1	1	1	1	1	Illegal
	All other combinations not listed are non-operations.					NOP
Master Transmit	X	X	0	0	1	TRANSMIT operation (master remains in Master Transmit state).
	X	X	1	0	0	STOP condition (master goes to Idle state).
	X	X	1	0	1	TRANSMIT followed by STOP condition (master goes to Idle state).
	0	X	0	1	1	Repeated START condition followed by a TRANSMIT (master remains in Master Transmit state).
	0	X	1	1	1	Repeated START condition followed by TRANSMIT and STOP condition (master goes to Idle state).
	1	0	0	1	1	Repeated START condition followed by a RECEIVE operation with a negative ACK (master goes to Master Receive state).
	1	0	1	1	1	Repeated START condition followed by a TRANSMIT and STOP condition (master goes to Idle state).
	1	1	0	1	1	Repeated START condition followed by RECEIVE (master goes to Master Receive state).
	1	1	1	1	1	Illegal.
	All other combinations not listed are non-operations.					NOP.

Table 15-5. Write Field Decoding for I2CMCS[3:0] Field *(continued)*

Current State	I2CMSA[0]	I2CMCS[3:0]				Description
	R/S	ACK	STOP	START	RUN	
Master Receive	X	0	0	0	1	RECEIVE operation with negative ACK (master remains in Master Receive state).
	X	X	1	0	0	STOP condition (master goes to Idle state).[b]
	X	0	1	0	1	RECEIVE followed by STOP condition (master goes to Idle state).
	X	1	0	0	1	RECEIVE operation (master remains in Master Receive state).
	X	1	1	0	1	Illegal.
	1	0	0	1	1	Repeated START condition followed by RECEIVE operation with a negative ACK (master remains in Master Receive state).
	1	0	1	1	1	Repeated START condition followed by RECEIVE and STOP condition (master goes to Idle state).
	1	1	0	1	1	Repeated START condition followed by RECEIVE (master remains in Master Receive state).
	0	X	0	1	1	Repeated START condition followed by TRANSMIT (master goes to Master Transmit state).
	0	X	1	1	1	Repeated START condition followed by TRANSMIT and STOP condition (master goes to Idle state).
	All other combinations not listed are non-operations.					NOP.

a. An X in a table cell indicates the bit can be 0 or 1.

b. In Master Receive mode, a STOP condition should be generated only after a Data Negative Acknowledge executed by the master or an Address Negative Acknowledge executed by the slave.

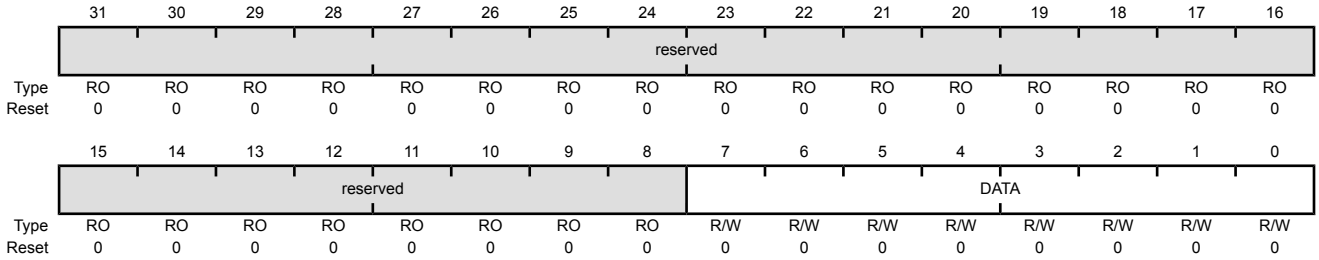
Register 3: I²C Master Data (I2CMDR), offset 0x008

Important: This register is read-sensitive. See the register description for details.

This register contains the data to be transmitted when in the Master Transmit state and the data received when in the Master Receive state.

I2C Master Data (I2CMDR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x008
Type R/W, reset 0x0000.0000



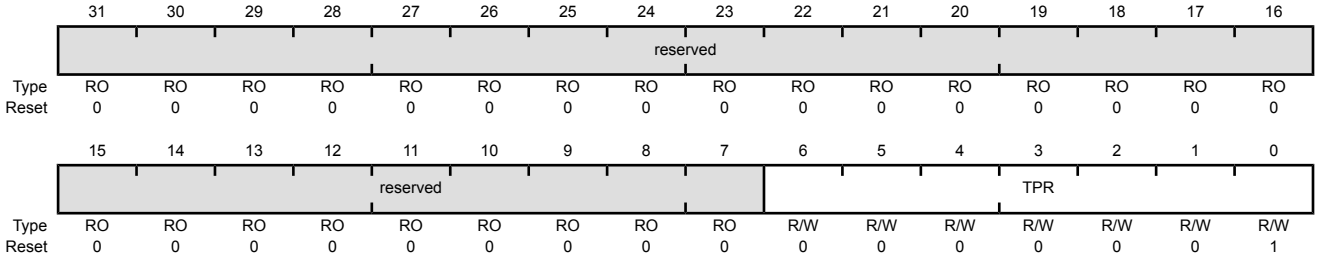
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x00	Data Transferred Data transferred during transaction.

Register 4: I2C Master Timer Period (I2CMTPR), offset 0x00C

This register specifies the period of the SCL clock.

I2C Master Timer Period (I2CMTPR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x00C
Type R/W, reset 0x0000.0001



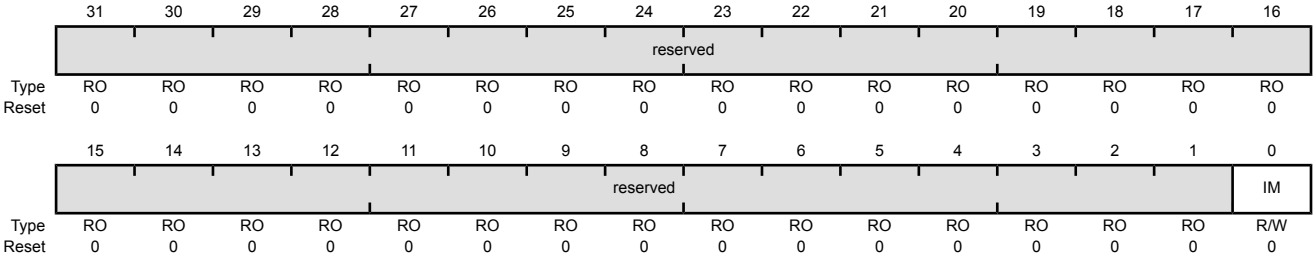
Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	TPR	R/W	0x1	SCL Clock Period This field specifies the period of the SCL clock. $SCL\_PRD = 2\times(1 + TPR)\times(SCL\_LP + SCL\_HP)\times CLK\_PRD$ where: *SCL_PRD* is the SCL line period (I2C clock). TPR is the Timer Period register value (range of 1 to 127). *SCL_LP* is the SCL Low period (fixed at 6). *SCL_HP* is the SCL High period (fixed at 4). *CLK_PRD* is the system clock period in ns.

Register 5: I²C Master Interrupt Mask (I2CMIMR), offset 0x010

This register controls whether a raw interrupt is promoted to a controller interrupt.

I2C Master Interrupt Mask (I2CMIMR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x010
Type R/W, reset 0x0000.0000



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

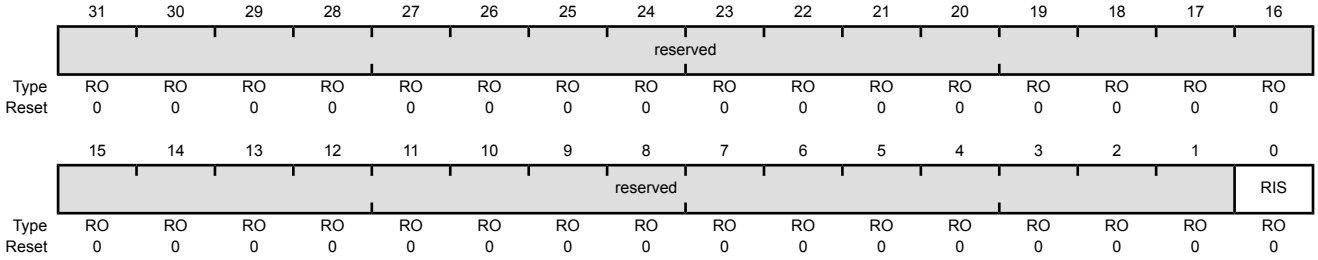
Value	Description
1	The master interrupt is sent to the interrupt controller when the RIS bit in the **I2CMRIS** register is set.
0	The RIS interrupt is suppressed and not sent to the interrupt controller.

Register 6: I2C Master Raw Interrupt Status (I2CMRIS), offset 0x014

This register specifies whether an interrupt is pending.

I2C Master Raw Interrupt Status (I2CMRIS)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x014
Type RO, reset 0x0000.0000



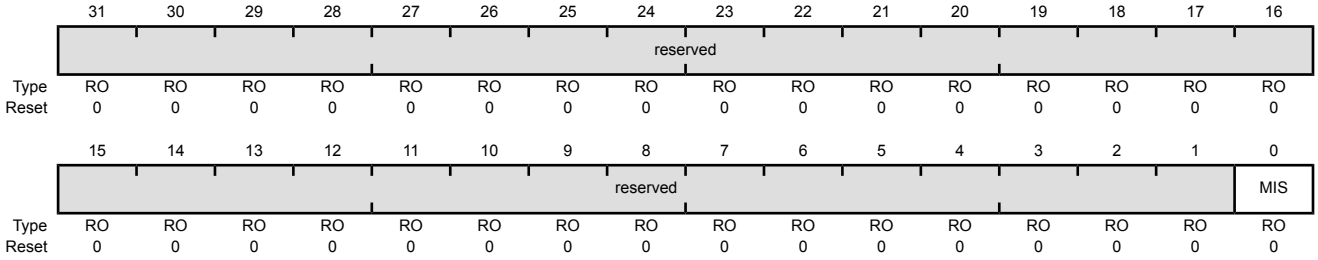
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	RIS	RO	0	Raw Interrupt Status Value Description 1 A master interrupt is pending. 0 No interrupt. This bit is cleared by writing a 1 to the IC bit in the **I2CMICR** register.

Register 7: I²C Master Masked Interrupt Status (I2CMMIS), offset 0x018

This register specifies whether an interrupt was signaled.

I2C Master Masked Interrupt Status (I2CMMIS)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x018
Type RO, reset 0x0000.0000



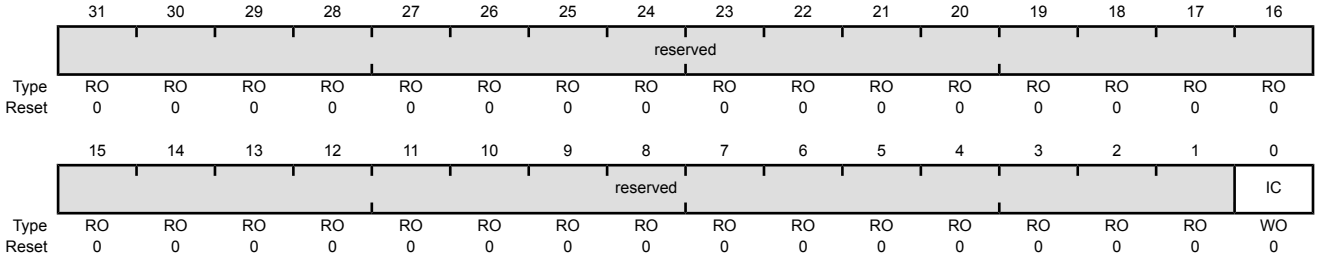
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	MIS	RO	0	Masked Interrupt Status Value Description 1 An unmasked master interrupt was signaled and is pending. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `IC` bit in the **I2CMICR** register.

Register 8: I²C Master Interrupt Clear (I2CMICR), offset 0x01C

This register clears the raw and masked interrupts.

I2C Master Interrupt Clear (I2CMICR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x01C
Type WO, reset 0x0000.0000

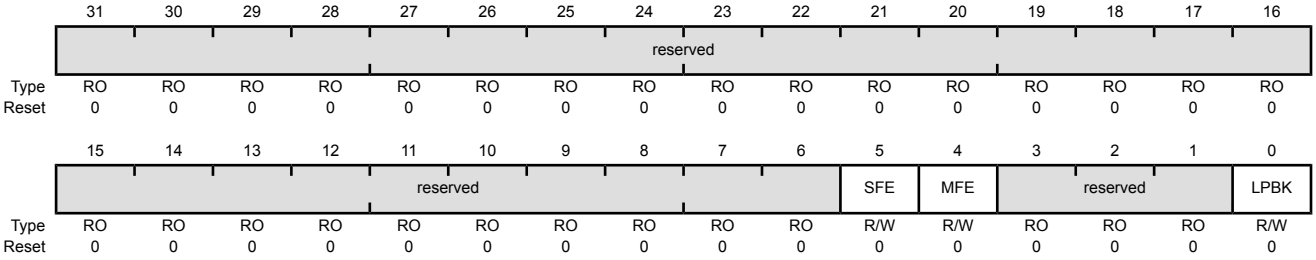


Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	IC	WO	0	Interrupt Clear Writing a 1 to this bit clears the `RIS` bit in the **I2CMRIS** register and the `MIS` bit in the **I2CMMIS** register. A read of this register returns no meaningful data.

Register 9: I²C Master Configuration (I2CMCR), offset 0x020

This register configures the mode (Master or Slave) and sets the interface for test mode loopback.

I2C Master Configuration (I2CMCR)
I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x020
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SFE	R/W	0	I²C Slave Function Enable Value Description 1 Slave mode is enabled. 0 Slave mode is disabled.
4	MFE	R/W	0	I²C Master Function Enable Value Description 1 Master mode is enabled. 0 Master mode is disabled.
3:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	LPBK	R/W	0	I²C Loopback Value Description 1 The controller in a test mode loopback configuration. 0 Normal operation.

15.7 Register Descriptions (I²C Slave)

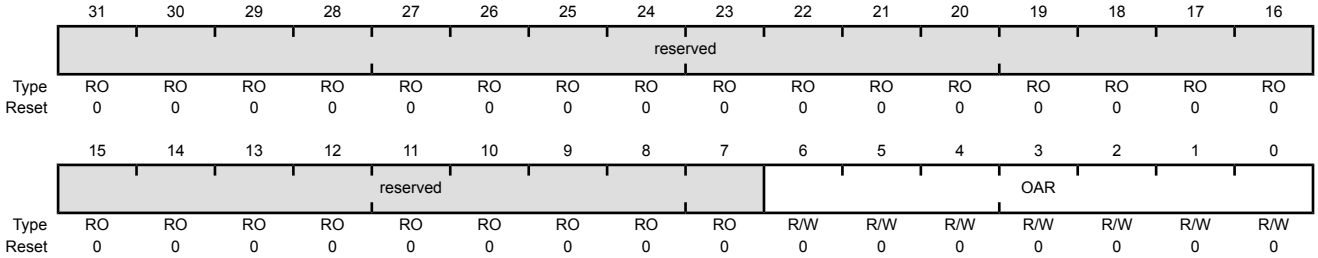
The remainder of this section lists and describes the I²C slave registers, in numerical order by address offset.

Register 10: I2C Slave Own Address (I2CSOAR), offset 0x800

This register consists of seven address bits that identify the Stellaris I2C device on the I2C bus.

I2C Slave Own Address (I2CSOAR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x800
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	OAR	R/W	0x00	I2C Slave Own Address This field specifies bits A6 through A0 of the slave address.

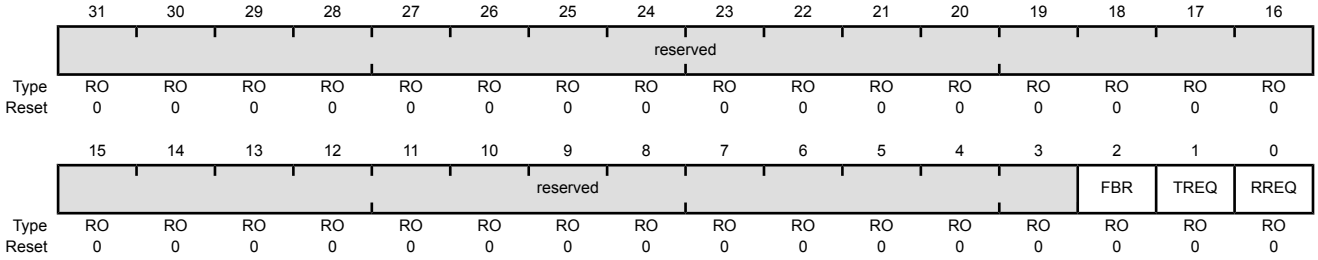
Register 11: I²C Slave Control/Status (I2CSCSR), offset 0x804

This register functions as a control register when written, and a status register when read.

Read-Only Status Register

I2C Slave Control/Status (I2CSCSR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x804
Type RO, reset 0x0000.0000

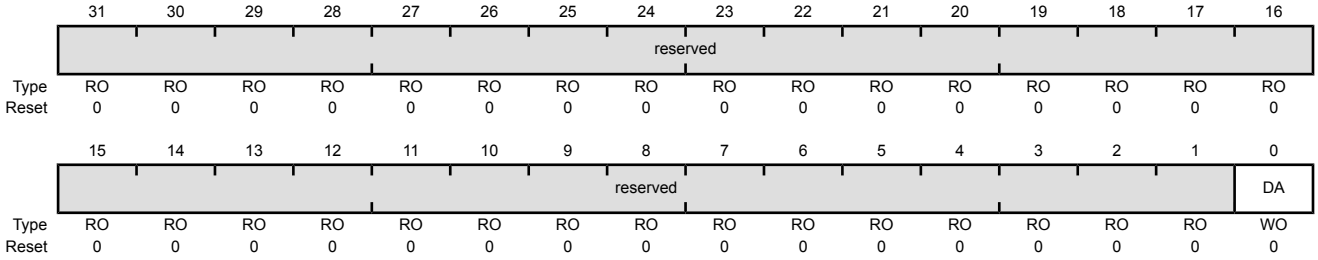


Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	FBR	RO	0	First Byte Received Value Description 1 The first byte following the slave's own address has been received. 0 The first byte has not been received. This bit is only valid when the RREQ bit is set and is automatically cleared when data has been read from the **I2CSDR** register. **Note:** This bit is not used for slave transmit operations.
1	TREQ	RO	0	Transmit Request Value Description 1 The I²C controller has been addressed as a slave transmitter and is using clock stretching to delay the master until data has been written to the **I2CSDR** register. 0 No outstanding transmit request.
0	RREQ	RO	0	Receive Request Value Description 1 The I²C controller has outstanding receive data from the I²C master and is using clock stretching to delay the master until the data has been read from the **I2CSDR** register. 0 No outstanding receive data.

Write-Only Control Register

I2C Slave Control/Status (I2CSCSR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x804
Type WO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	DA	WO	0	Device Active Value Description 0 Disables the I²C slave operation. 1 Enables the I²C slave operation. Once this bit has been set, it should not be set again unless it has been cleared by writing a 0 or by a reset, otherwise transfer failures may occur.

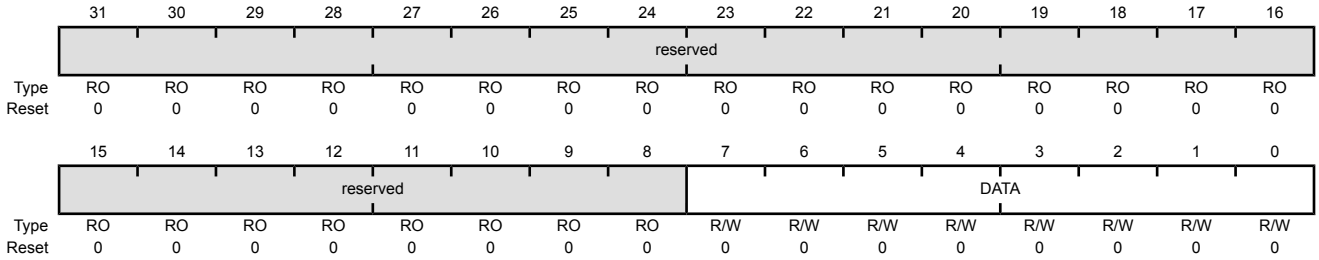
Register 12: I²C Slave Data (I2CSDR), offset 0x808

Important: This register is read-sensitive. See the register description for details.

This register contains the data to be transmitted when in the Slave Transmit state, and the data received when in the Slave Receive state.

I2C Slave Data (I2CSDR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x808
Type R/W, reset 0x0000.0000

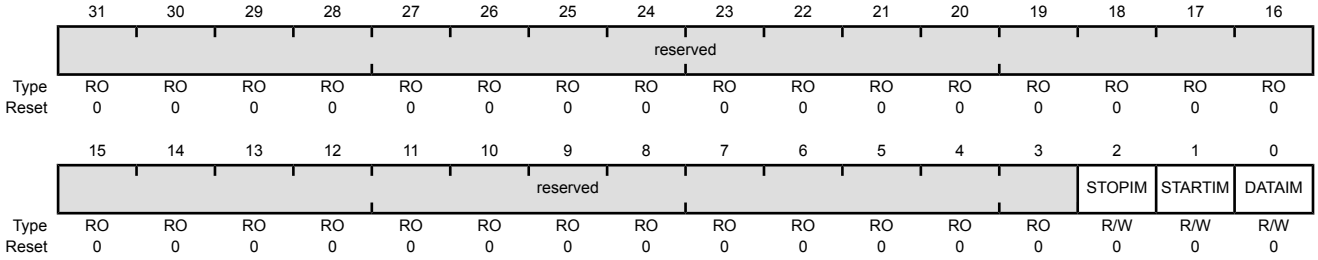


Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x00	Data for Transfer This field contains the data for transfer during a slave receive or transmit operation.

Register 13: I2C Slave Interrupt Mask (I2CSIMR), offset 0x80C

This register controls whether a raw interrupt is promoted to a controller interrupt.

I2C Slave Interrupt Mask (I2CSIMR)
I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x80C
Type R/W, reset 0x0000.0000



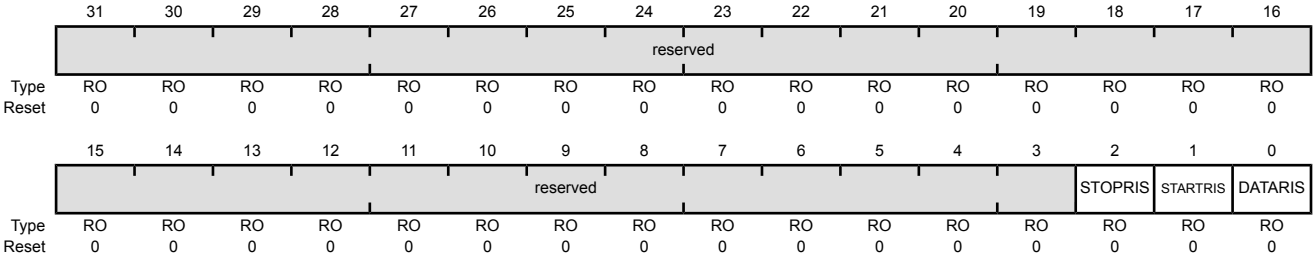
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPIM	R/W	0	Stop Condition Interrupt Mask Value Description 1 The STOP condition interrupt is sent to the interrupt controller when the STOPRIS bit in the **I2CSRIS** register is set. 0 The STOPRIS interrupt is suppressed and not sent to the interrupt controller.
1	STARTIM	R/W	0	Start Condition Interrupt Mask Value Description 1 The START condition interrupt is sent to the interrupt controller when the STARTRIS bit in the **I2CSRIS** register is set. 0 The STARTRIS interrupt is suppressed and not sent to the interrupt controller.
0	DATAIM	R/W	0	Data Interrupt Mask Value Description 1 The data received or data requested interrupt is sent to the interrupt controller when the DATARIS bit in the **I2CSRIS** register is set. 0 The DATARIS interrupt is suppressed and not sent to the interrupt controller.

Register 14: I²C Slave Raw Interrupt Status (I2CSRIS), offset 0x810

This register specifies whether an interrupt is pending.

I2C Slave Raw Interrupt Status (I2CSRIS)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x810
Type RO, reset 0x0000.0000



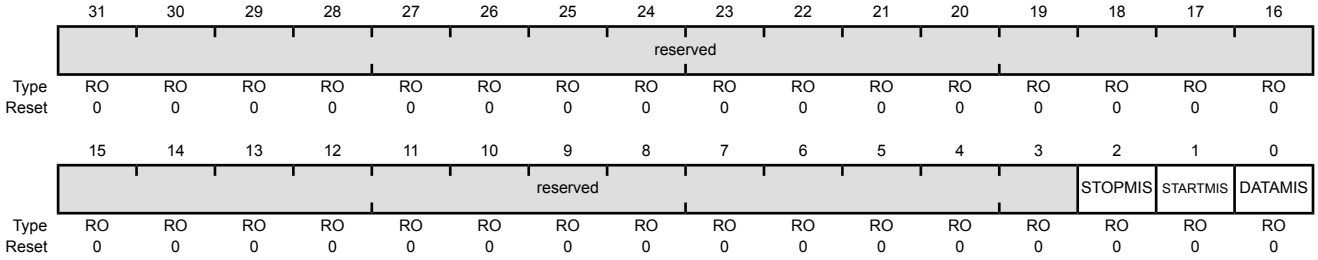
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPRIS	RO	0	Stop Condition Raw Interrupt Status Value Description 1 A STOP condition interrupt is pending. 0 No interrupt. This bit is cleared by writing a 1 to the `STOPIC` bit in the **I2CSICR** register.
1	STARTRIS	RO	0	Start Condition Raw Interrupt Status Value Description 1 A START condition interrupt is pending. 0 No interrupt. This bit is cleared by writing a 1 to the `STARTIC` bit in the **I2CSICR** register.
0	DATARIS	RO	0	Data Raw Interrupt Status Value Description 1 A data received or data requested interrupt is pending. 0 No interrupt. This bit is cleared by writing a 1 to the `DATAIC` bit in the **I2CSICR** register.

Register 15: I2C Slave Masked Interrupt Status (I2CSMIS), offset 0x814

This register specifies whether an interrupt was signaled.

I2C Slave Masked Interrupt Status (I2CSMIS)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x814
Type RO, reset 0x0000.0000



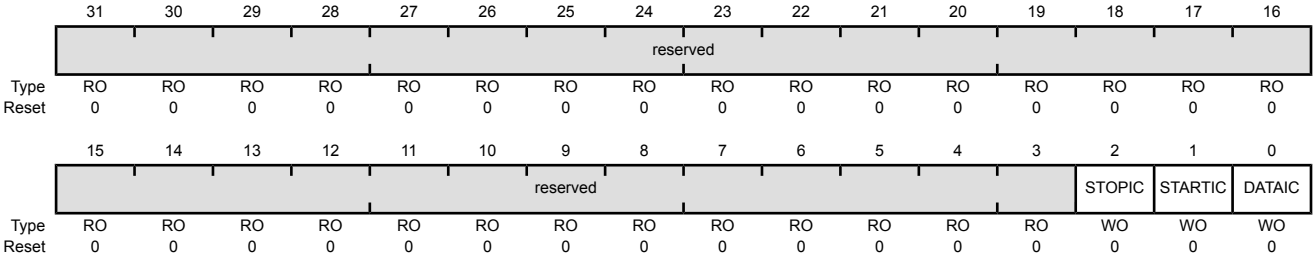
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPMIS	RO	0	Stop Condition Masked Interrupt Status Value Description 1 An unmasked STOP condition interrupt was signaled is pending. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the STOPIC bit in the **I2CSICR** register.
1	STARTMIS	RO	0	Start Condition Masked Interrupt Status Value Description 1 An unmasked START condition interrupt was signaled is pending. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the STARTIC bit in the **I2CSICR** register.
0	DATAMIS	RO	0	Data Masked Interrupt Status Value Description 1 An unmasked data received or data requested interrupt was signaled is pending. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the DATAIC bit in the **I2CSICR** register.

Register 16: I²C Slave Interrupt Clear (I2CSICR), offset 0x818

This register clears the raw interrupt. A read of this register returns no meaningful data.

I2C Slave Interrupt Clear (I2CSICR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x818
Type WO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPIC	WO	0	Stop Condition Interrupt Clear Writing a 1 to this bit clears the STOPRIS bit in the **I2CSRIS** register and the STOPMIS bit in the **I2CSMIS** register. A read of this register returns no meaningful data.
1	STARTIC	WO	0	Start Condition Interrupt Clear Writing a 1 to this bit clears the STOPRIS bit in the **I2CSRIS** register and the STOPMIS bit in the **I2CSMIS** register. A read of this register returns no meaningful data.
0	DATAIC	WO	0	Data Interrupt Clear Writing a 1 to this bit clears the STOPRIS bit in the **I2CSRIS** register and the STOPMIS bit in the **I2CSMIS** register. A read of this register returns no meaningful data.

16 Inter-Integrated Circuit Sound (I^2S) Interface

The I^2S module is a configurable serial audio core that contains a transmit module and a receive module. The module is configurable for the I^2S as well as Left-Justified and Right-Justified serial audio formats. Data can be in one of four modes: Stereo, Mono, Compact 16-bit Stereo and Compact 8-Bit Stereo.

The transmit and receive modules each have an 8-entry audio-sample FIFO. An audio sample can consist of a Left and Right Stereo sample, a Mono sample, or a Left and Right Compact Stereo sample. In Compact 16-Bit Stereo, each FIFO entry contains both the 16-bit left and 16-bit right samples, allowing efficient data transfers and requiring less memory space. In Compact 8-bit Stereo, each FIFO entry contains an 8-bit left and an 8-bit right sample, reducing memory requirements further.

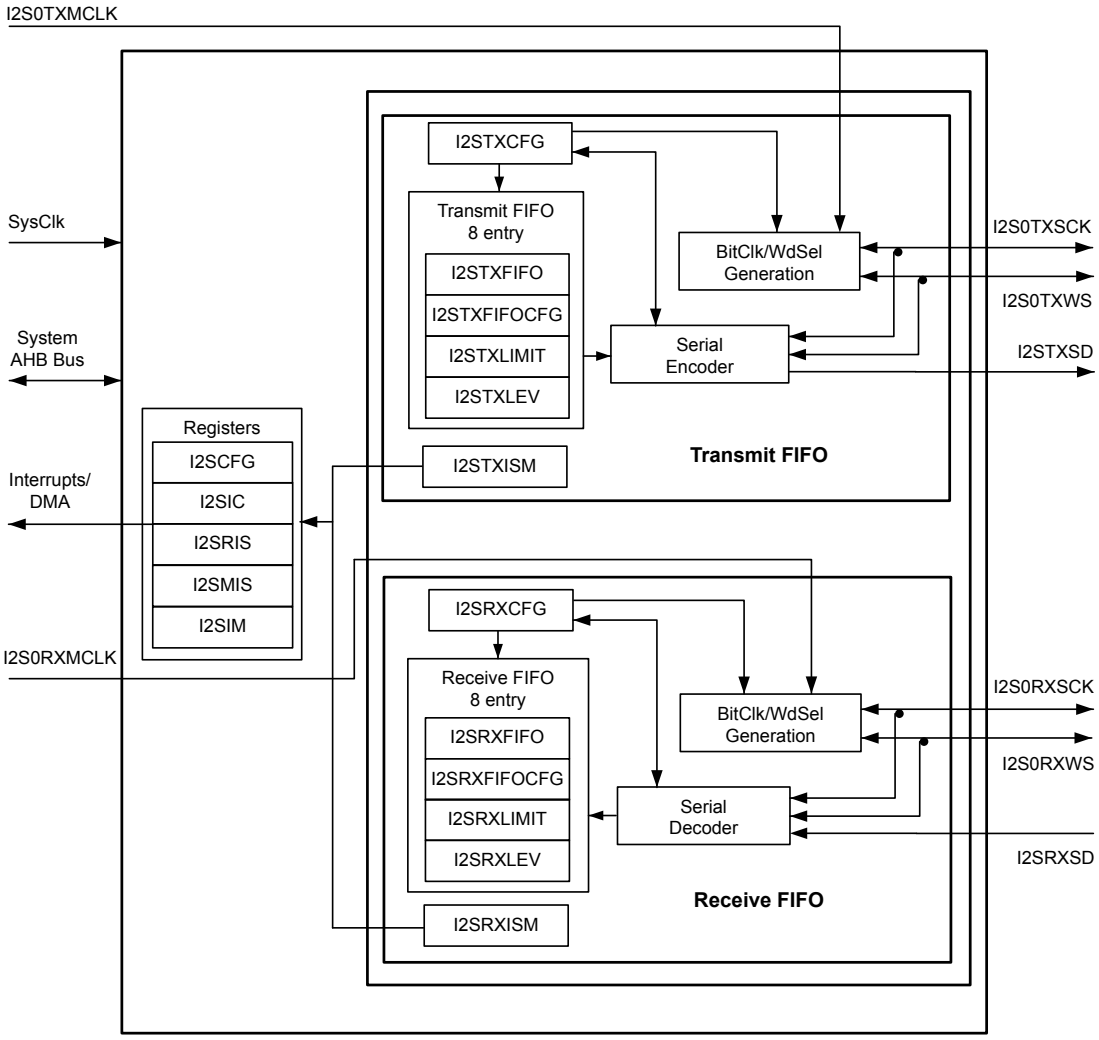
Both the transmitter and receiver are capable of being a master or a slave.

The Stellaris® I^2S module has the following features:

- Configurable audio format supporting I^2S, Left-justification, and Right-justification
- Configurable sample size from 8 to 32 bits
- Mono and Stereo support
- 8-, 16-, and 32-bit FIFO interface for packing memory
- Independent transmit and receive 8-entry FIFOs
- Configurable FIFO-level interrupt and µDMA requests
- Independent transmit and receive MCLK direction control
- Transmit and receive internal MCLK sources
- Independent transmit and receive control for serial clock and word select
- MCLK and SCLK can be independently set to master or slave
- Configurable transmit zero or last sample when FIFO empty
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Separate channels for transmit and receive
  - Burst requests
  - Channel requests asserted when FIFO contains required amount of data

16.1 Block Diagram

Figure 16-1. I²S Block Diagram



16.2 Signal Description

The following table lists the external signals of the I²S module and describes the function of each. The I²S module signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the I²S signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 419) should be set to choose the I²S function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 437) to assign the I²S signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 395.

Table 16-1. I2S Signals (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
I2S0RXMCLK	29 98	PA3 (9) PD5 (8)	I/O	TTL	I^2S module 0 receive master clock.
I2S0RXSCK	10	PD0 (8)	I/O	TTL	I^2S module 0 receive clock.
I2S0RXSD	28 97	PA2 (9) PD4 (8)	I/O	TTL	I^2S module 0 receive data.
I2S0RXWS	11	PD1 (8)	I/O	TTL	I^2S module 0 receive word select.
I2S0TXMCLK	61	PF1 (8)	I/O	TTL	I^2S module 0 transmit master clock.
I2S0TXSCK	30 90 99	PA4 (9) PB6 (9) PD6 (8)	I/O	TTL	I^2S module 0 transmit clock.
I2S0TXSD	5 47	PE5 (9) PF0 (8)	I/O	TTL	I^2S module 0 transmit data.
I2S0TXWS	6 31 100	PE4 (9) PA5 (9) PD7 (8)	I/O	TTL	I^2S module 0 transmit word select.

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

Table 16-2. I2S Signals (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
I2S0RXMCLK	L4 C6	PA3 (9) PD5 (8)	I/O	TTL	I^2S module 0 receive master clock.
I2S0RXSCK	G1	PD0 (8)	I/O	TTL	I^2S module 0 receive clock.
I2S0RXSD	M4 B5	PA2 (9) PD4 (8)	I/O	TTL	I^2S module 0 receive data.
I2S0RXWS	G2	PD1 (8)	I/O	TTL	I^2S module 0 receive word select.
I2S0TXMCLK	H12	PF1 (8)	I/O	TTL	I^2S module 0 transmit master clock.
I2S0TXSCK	L5 A7 A3	PA4 (9) PB6 (9) PD6 (8)	I/O	TTL	I^2S module 0 transmit clock.
I2S0TXSD	B3 M9	PE5 (9) PF0 (8)	I/O	TTL	I^2S module 0 transmit data.
I2S0TXWS	B2 M5 A2	PE4 (9) PA5 (9) PD7 (8)	I/O	TTL	I^2S module 0 transmit word select.

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

16.3 Functional Description

The Inter-Integrated Circuit Sound (I^2S) module contains separate transmit and receive engines. Each engine consists of the following:

- Serial encoder for the transmitter; serial decoder for the receiver
- 8-entry FIFO to store sample data
- Independent configuration of all programmable settings

The basic programming model of the I²S block is as follows:

- Configuration
  - Overall I²S module configuration in the **I²S Module Configuration (I2SCFG)** register. This register is used to select the MCLK source and enable the receiver and transmitter.
  - Transmit and receive configuration in the **I²S Transmit Module Configuration (I2STXCFG)** and **I²S Receive Module Configuration (I2SRXCFG)** registers. These registers set the basic parameters for the receiver and transmitter such as data configuration (justification, delay, read mode, sample size, and system data size); SCLK (polarity and source); and word select polarity.
  - Transmit and receive FIFO configuration in the **I²S Transmit FIFO Configuration (I2STXFIFOCFG)** and **I²S Receive FIFO Configuration (I2SRXFIFOCFG)** registers. These registers select the Compact Stereo mode size (16-bit or 8-bit), provide indication of whether the next sample is Left or Right, and select mono mode for the receiver.
- FIFO
  - Transmit and receive FIFO data in the **I²S Transmit FIFO Data (I2STXFIFO)** and **I²S Receive FIFO Data (I2SRXFIFO)** registers
  - Information on FIFO data levels in the **I²S Transmit FIFO Level (I2STXLEV)** and **I²S Receive FIFO Level (I2SRXLEV)** registers
  - Configuration for FIFO service requests based on FIFO levels in the **I²S Transmit FIFO Limit (I2STXLIMIT)** and **I²S Receive FIFO Limit (I2SRXLIM)** registers
- Interrupt Control
  - Interrupt masking configuration in the **I²S Interrupt Mask (I2SIM)** register
  - Raw and masked interrupt status in the **I²S Raw Interrupt Status (I2SRIS)** and **I²S Masked Interrupt Status (I2SMIS**) registers
  - Interrupt clearing through the **I²S Interrupt Clear (I2SIC)** register
  - Configuration for FIFO service requests interrupts and transmit/receive error interrupts in the **I²S Transmit Interrupt Status and Mask (I2STXISM)** and **I²S Receive Interrupt Status and Mask (I2SRXISM)** registers

Figure 16-2 on page 826 provides an example of an I²S data transfer. Figure 16-3 on page 826 provides an example of an Left-Justified data transfer. Figure 16-4 on page 826 provides an example of an Right-Justified data transfer.

Figure 16-2. I^2S Data Transfer

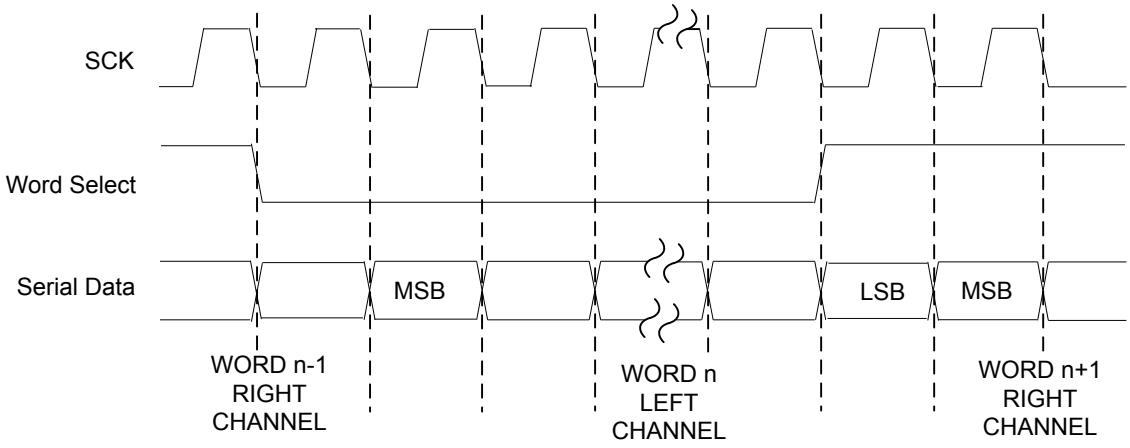


Figure 16-3. Left-Justified Data Transfer

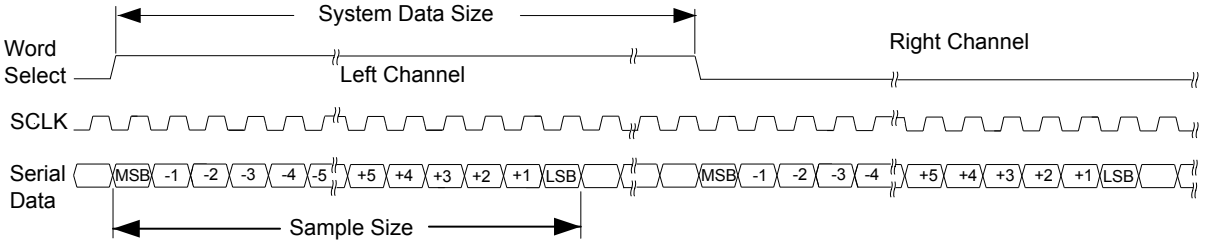
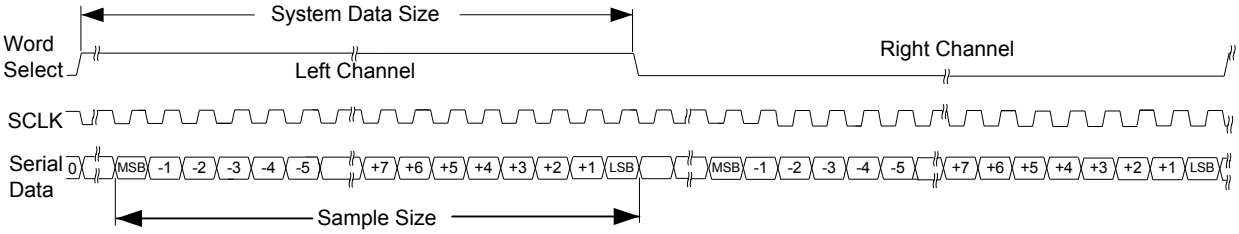


Figure 16-4. Right-Justified Data Transfer



16.3.1 Transmit

The transmitter consists of a serial encoder, an 8-entry FIFO, and control logic. The transmitter has independent MCLK (I2S0TXMCLK), SCLK (I2S0TXSCK), and Word-Select (I2S0TXWS) signals.

16.3.1.1 Serial Encoder

The serial encoder reads audio samples from the receive FIFO and converts them into an audio stream. By configuring the serial encoder, common audio formats I^2S, Left-Justified, and Right-Justified are supported. The MSB is transmitted first. The sample size and system data size are configurable with the SSZ and SDSZ bits in the **I^2S Transmit Module Configuration (I2STXCFG)** register. The sample size is the number of bits of data being transmitted, and the system data size is the number of I2S0TXSCK transitions between the word select transitions. The system data size must be large enough to accommodate the maximum sample size. In Mono mode, the sample data

is repeated in both the left and right channels. When the FIFO is empty, the user may select either transmission of zeros or of the last sample. The serial encoder is enabled using the TXEN bit in the **I²S Module Configuration (I2SCFG)** register.

16.3.1.2 FIFO Operation

The transmit FIFO stores eight Mono samples or eight Stereo sample-pairs of data and is accessed through the **I²S Transmit FIFO Data (I2STXFIFO)** register. The FIFO interface for the audio data is different based on the Write mode, defined by the **I²S Transmit FIFO Configuration (I2STXFIFOCFG)** Compact Stereo Sample Size bit (CSS) and the **I2STXCFG** Write Mode field (WM). All data samples are MSB-aligned. Table 16-3 on page 827 defines the interface for each Write mode. Stereo samples are written first left then right. The next sample (right or left) to be written is indicated by the LRS bit in the **I2STXFIFOCFG** register.

Table 16-3. I²S Transmit FIFO Interface

WM field in I2STXCFG	CSS bit in I2STXFIFOCFG	Write Mode	Sample Width	Samples per FIFO Write	Data Alignment
0x0	don't care	Stereo	8-32 bits	1	MSB
0x1	0	Compact Stereo - 16 bit	8-16 bits	2	MSB Right [31:16], Left [15:0]
0x1	1	Compact Stereo - 8 bit	8 bits	2	Right [15:8], Left[7:0]
0x2	don't care	Mono	8-32 bits	1	MSB

The number of samples in the transmit FIFO can be read using the **I²S Transmit FIFO Level (I2STXLEV)** register. The value ranges from 0 to 16. Stereo and compact stereo sample pairs are counted as two. The mono samples also increment the count by two, therefore, four mono samples will have a count of eight.

16.3.1.3 Clock Control

The transmitter MCLK and SCLK can be independently programmed to be the master or slave. The transmitter is programmed to be the master or slave of the SCLK using the MSL bit in the **I2STXCFG** register. When the transmitter is the master, the I2S0TXSCK frequency is the specified I2S0TXMCLK divided by four. The I2S0TXSCK may be inverted using the SCP bit in the **I2STXCFG** register.

The transmitter can also be the master or slave of the MCLK. When the transmitter is the master, the PLL must be active and a fractional clock divider must be programmed. See page 229 for the setup for the master I2S0TXMCLK source. An external transmit I2S0TXMCLK does not require the use of the PLL and is selected using the TXSLV bit in the **I2SCFG** register.

The following tables show combinations of the TXINT and TXFRAC bits in the **I²S MCLK Configuration (I2SMCLKCFG)** register that provide MCLK frequencies within acceptable error limits. In the table, Fs is the sampling frequency in kHz and possible crystal frequencies are shown in MHz across the top row of the table. The words "not supported" in the table mean that it is not possible to obtain the specified sampling frequencies with the specified crystal frequency within the error tolerance of 0.3%. The values in the table are based on the following values:

```
MCLK = Fs × 256

PLL = 400 MHz
```

The Integer value is taken from the result of the following calculation:

```
ROUND(PLL/MCLK)
```

The remaining fractional component is converted to binary, and the first four bits are the Fractional value.

Table 16-4. Crystal Frequency (Values from 3.5795 MHz to 5 MHz)

Sampling Frequency Fs (kHz)	Crystal Frequency (MHz)											
	3.5795		3.6864		4		4.096		4.9152		5	
	Integer	Fractional	Integer	Fractional	Integer	Fractional	Integer	Fractional	Integer	Fractional	Integer	Fractional
8	195	12	194	6	195	5	196	0	194	6	195	5
11.025	142	1	141	1	141	12	142	4	141	1	141	12
12	130	8	129	10	130	3	130	11	129	10	130	3
16	97	14	97	3	97	10	98	0	97	3	97	10
22.05	71	0	70	8	70	14	71	2	70	8	70	14
24	65	4	64	13	65	2	65	5	64	13	65	2
32	48	15	48	10	48	13	49	0	48	10	48	13
44.1	35	8	35	4	35	7	35	9	35	4	35	7
48	32	10	32	6	32	9	32	11	32	6	32	9
64	24	8	24	5	24	7	24	8	24	5	24	7
88.2	17	12	17	10	17	11	17	12	17	10	17	11
96	16	5	16	3	16	4	16	5	16	3	16	4
128	12	4	12	2	12	3	12	4	12	2	12	3
176.4	8	14	8	13	8	14	8	14	8	13	8	14
192	Not supported		Not supported		8	2	8	3	Not supported		8	2

Table 16-5. Crystal Frequency (Values from 5.12 MHz to 8.192 MHz)

Sampling Frequency Fs (kHz)	Crystal Frequency (MHz)											
	5.12		6		6.144		7.3728		8		8.192	
	Integer	Fractional	Integer	Fractional	Integer	Fractional	Integer	Fractional	Integer	Fractional	Integer	Fractional
8	195	0	195	5	195	0	194	6	195	5	194	11
11.025	141	8	141	12	141	8	141	1	141	12	141	4
12	130	0	130	3	130	0	129	10	130	3	129	12
16	97	8	97	10	97	8	97	3	97	10	97	5
22.05	70	12	70	14	70	12	70	8	70	14	70	10
24	65	0	65	2	65	0	64	13	65	2	64	14
32	48	12	48	13	48	12	48	10	48	13	48	11
44.1	35	6	35	7	35	6	35	4	35	7	35	5
48	32	8	32	9	32	8	32	6	32	9	32	7
64	24	6	24	7	24	6	24	5	24	7	24	5
88.2	17	11	17	11	17	11	17	10	17	11	17	11
96	16	4	16	4	16	4	16	3	16	4	16	4
128	12	3	12	3	12	3	12	2	12	3	12	3
176.4	Not supported		8	14	Not supported		8	13	8	14	8	13
192	8	2	8	2	8	2	Not supported		8	2	8	2

Table 16-6. Crystal Frequency (Values from 10 MHz to 14.3181 MHz)

Sampling Frequency Fs (kHz)	Crystal Frequency (MHz)									
	10		12		12.288		13.56		14.3181	
	Integer	Fractional	Integer	Fractional	Integer	Fractional	Integer	Fractional	Integer	Fractional
8	195	5	195	5	196	0	194	3	195	12
11.025	141	12	141	12	142	4	140	15	142	1
12	130	3	130	3	130	11	129	8	130	8
16	97	10	97	10	98	0	97	2	97	14
22.05	70	14	70	14	71	2	70f	7	71	0
24	65	2	65	2	65	5	64	12	65	4
32	48	13	48	13	49	0	48	9	48	15
44.1	35	7	35	7	35	9	35	4	35	8
48	32	9	32	9	32	11	32	6	32	10
64	24	7	24	7	24	8	24	4	24	8
88.2	17	11	17	11	17	12	17	10	17	12
96	16	4	16	4	16	5	16	3	16	5
128	12	3	12	3	12	4	12	2	12	4
176.4	8	14	8	14	8	14	8	13	8	14
192	8	2	8	2	8	3	Not supported		Not supported	

Table 16-7. Crystal Frequency (Values from 16 MHz to 16.384 MHz)

Sampling Frequency Fs (kHz)	Crystal Frequency (MHz)			
	16		16.384	
	Integer	Fractional	Integer	Fractional
8	195	5	192	0
11.025	141	12	139	5
12	130	3	128	0
16	97	10	96	0
22.05	70	14	69	11
24	65	2	64	0
32	48	13	48	0
44.1	35	7	34	13
48	32	9	32	0
64	24	7	24	0
88.2	17	11	17	7
96	16	4	16	0
128	12	3	12	0
176.4	8	14	8	11
192	8	2	8	0

16.3.1.4 Interrupt Control

A single interrupt is asserted to the CPU whenever any of the transmit or receive sources is asserted. The transmit module has two interrupt sources: the FIFO service request and write error. The interrupts may be masked using the TXSRIM and TXWEIM bits in the **I²S Interrupt Mask (I2SIM)**

register. The status of the interrupt source is indicated by the **I^2S Raw Interrupt Status (I2SRIS)** register. The status of enabled interrupts is indicated by the **I^2S Masked Interrupt Status (I2SMIS)** register. The FIFO level interrupt has a second level of masking using the FFM bit in the **I^2S Transmit Interrupt Status and Mask (I2STXISM)** register.

The FIFO service request interrupt is asserted when the FIFO level (indicated by the LEVEL field in the **I^2S Transmit FIFO Level (I2STXLEV)** register) is below the FIFO limit (programmed using the **I^2S Transmit FIFO Limit (I2STXLIMIT)** register) and both the TXSRIM and FFM bits are set. If software attempts to write to a full FIFO, a Transmit FIFO Write error occurs (indicated by the TXWERIS bit in the **I^2S Raw Interrupt Status (I2SRIS)** register). The TXWERIS bit in the **I2SRIS** register and the TXWEMIS bit in the **I2SMIS** register are cleared by setting the TXWEIC bit in the **I^2S Interrupt Clear (I2SIC)** register.

16.3.1.5 DMA Support

The µDMA can be used to more efficiently stream data to and from the I^2S bus. The I^2S tranmit and receive modules have separate µDMA channels. The FIFO Interrupt Mask bit (FFM) in the **I2STXISM** register must be set for the request signaling to propagate to the µDMA module. See “Micro Direct Memory Access (µDMA)” on page 334 for channel configuration.

The I^2S module uses the µDMA burst request signal, not the single request. Thus each time a µDMA request is made, the µDMA controller transfers the number of items specified as the burst size for the µDMA channel. Therefore, the µDMA channel burst size and the I^2S FIFO service request limit must be set to the same value (using the LIMIT field in the **I2STXLIMIT** register).

16.3.2 Receive

The receiver consists of a serial decoder, an 8-entry FIFO, and control logic. The receiver has independent MCLK (I2S0RXMCLK), SCLK (I2S0RXSCK), and Word-Select (I2S0RXWS) signals.

16.3.2.1 Serial Decoder

The serial decoder accepts incoming audio stream data and places the sample data in the receive FIFO. By configuring the serial decoder, common audio formats I^2S, Left-Justified, and Right-Justified are supported. The MSB is transmitted first. The sample size and system data size are configurable with the SSZ and SDSZ bits in the **I^2S Receive Module Configuration (I2SRXCFG)** register. The sample size is the number of bits of data being received, and the system data size is the number of I2S0RXSCK transitions between the word select transitions. The system data size must be large enough to accommodate the maximum sample size. Any bits received after the LSB are 0s. If the FIFO is full, the incoming sample (in Mono) or sample-pairs (Stereo) are dropped until the FIFO has space. The serial decoder is enabled using the RXEN bit in the **I2SCFG** register.

16.3.2.2 FIFO Operation

The receive FIFO stores eight Mono samples or eight Stereo sample-pairs of data and is accessed through the **I^2S Receive FIFO Data (I2SRXFIFO)** register. Table 16-8 on page 831 defines the interface for each Read mode. All data is stored MSB-aligned. The Stereo data is read left sample then right.

In Mono mode, the FIFO interface can be configured to read the right or left channel by setting the FIFO Mono Mode bit (FMM) in the **I^2S Receive FIFO Configuration (I2SRXFIFOCFG)** register. This enables reads from a single channel, where the channel selected can be either the right or left as determined by the LRP bit in the **I2SRXCFG** register.

Table 16-8. I²S Receive FIFO Interface

RM bit in I2RXCFG	CSS bit in I2SRXFIFOCFG	Read Mode	Sample Width	Samples per FIFO Read	Data Alignment
0	don't care	Stereo	8-32 bits	1	MSB
1	0	Compact Stereo - 16 bit	8-16 bits	2	MSB Right [31:15], Left [15:0]
1	1	Compact Stereo - 8 bit	8 bits	2	Right [15:8] Left[7:0]
0	don't care	Mono (FMM bit in the **I2SRXFIFOCFG** register must be set.)	8-32 bits	1	MSB

The number of samples in the receive FIFO can be read using the **I²S Receive FIFO Level (I2SRXLEV)** register. The value ranges from 0 to 16. Stereo and compact stereo sample pairs are counted as two. The mono samples also increment the count by two, therefore four Mono samples will have a count of eight.

16.3.2.3 Clock Control

The receiver MCLK and SCLK can be independently programmed to be the master or slave. The receiver is programmed to be the master or slave of the SCLK using the MSL bit in the **I2SRXCFG** register. When the receiver is the master, the I2S0RXSCK frequency is the specified I2S0RXMCLK divided by four. The I2S0RXSCK may be inverted using the SCP bit in the **I2SRXCFG** register.

The receiver can also be the master or slave of the MCLK. When the receiver is the master, the PLL must be active and a fractional clock divider must be programmed. See page 229 for the setup for the master I2S0RXMCLK source. An external transmit I2S0RXMCLK does not require the use of the PLL and is selected using the RXSLV bit in the **I2SCFG** register.

Refer to “Clock Control” on page 827 for combinations of the RXINT and RXFRAC bits in the **I²S MCLK Configuration (I2SMCLKCFG)** register that provide MCLK frequencies within acceptable error limits. In the table, Fs is the sampling frequency in kHz and possible crystal frequencies are shown in MHz across the top row of the table. The words "not supported" in the table mean that it is not possible to obtain the specified sampling frequencies with the specified crystal frequency within the error tolerance of 0.3%.

16.3.2.4 Interrupt Control

A single interrupt is asserted to the CPU whenever any of the transmit or receive sources is asserted. The receive module has two interrupt sources: the FIFO service request and read error. The interrupts may be masked using the RXSRIM and RXREIM bits in the **I2SIM** register. The status of the interrupt source is indicated by the **I2SRIS** register. The status of enabled interrupts is indicated by the **I2SMIS** register. The FIFO service request interrupt has a second level of masking using the FFM bit in the **I²S Receive Interrupt Status and Mask (I2SRXISM)** register. The sources may be masked using the **I2SIM** register.

The FIFO service request interrupt is asserted when the FIFO level (indicated by the LEVEL field in the **I²S Receive FIFO Level (I2SRXLEV)** register) is above the FIFO limit (programmed using the **I²S Receive FIFO Limit (I2SRXLIMIT)** register) and both the RXSRIM and FFM bits are set. An error occurs when reading an empty FIFO or if a stereo sample pair is not read left then right. To clear an interrupt, write a 1 to the appropriate bit in the **I2SIC** register. If software attempts to read an empty FIFO or if a stereo sample pair is not read left then right, a Receive FIFO Read error occurs (indicated by the RXRERIS bit in the **I2SRIS** register). The RXRERIS bit in the **I2SRIS** register and the RXREMIS bit in the **I2SMIS** register are cleared by setting the RXREIC bit in the **I2SIC** register.

16.3.2.5 DMA Support

The µDMA can be used to more efficiently stream data to and from the I^2S bus. The I^2S transmit and receive modules have separate µDMA channels. The FIFO Interrupt Mask bit (FFM) in the **I2SRXISM** register must be set for the request signaling to propagate to the µDMA module. See "Micro Direct Memory Access (µDMA)" on page 334 for channel configuration.

The I^2S module uses the µDMA burst request signal, not the single request. Thus each time a µDMA request is made, the µDMA controller transfers the number of items specified as the burst size for the µDMA channel. Therefore, the µDMA channel burst size and the I^2S FIFO service request limit must be set to the same value (using the LIMIT field in the **I2SRXLIMIT** register).

16.4 Initialization and Configuration

The default setup for the I^2S transmit and receive is to use external MCLK, external SCLK, Stereo, I^2S audio format, and 32-bit data samples. The following example shows how to configure a system using the internal MCLK, internal SCLK, Compact Stereo, and Left-Justified audio format with 16-bit data samples.

1. Enable the I^2S peripheral clock by writing a value of 0x1000.0000 to the **RCGC1** register in the System Control module (see page 263).
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register in the System Control module (see page 272). To find out which GPIO port to enable, refer to Table 22-5 on page 1151.
3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register (see page 419). To determine which GPIOs to configure, see Table 22-4 on page 1144.
4. Configure the PMCn fields in the **GPIOPCTL** register to assign the I^2S signals to the appropriate pins (see page 437 and Table 22-5 on page 1151).
5. Set up the MCLK sources for a 48-kHz sample rate. The input crystal is assumed to be 6 MHz for this example (internal source).
   - Enable the PLL by clearing the PWRDWN bit in the **RCC** register in the System Control module (see page 215).
   - Set the MCLK dividers and enable them by writing 0x0208.0208 to the **I2SMCLKCFG** register in the System Control module (see page 229).
   - Enable the MCLK internal sources by writing 0x8208.8208 to the **I2SMCLKCFG** register in the System Control module.

   To allow an external MCLK to be used, set bits 4 and 5 of the **I2SCFG** register. Starting up the PLL and enabling the MCLK sources is not required.
6. Set up the Serial Bit Clock SCLK source. By default, the SCLK is externally sourced.
   - Receiver: Masters the I2S0RXSCK by ORing 0x0040.0000 into the **I2SRXCFG** register.
   - Transmitter: Masters the I2S0TXSCK by ORing 0x0040.0000 into the **I2STXCFG** register.
7. Configure the Serial Encoder/Decoder (Left-Justified, Compact Stereo, 16-bit samples, 32-bit system data size).

- Set the audio format using the Justification (JST), Data Delay (DLY), SCLK polarity (SCP), and Left-Right Polarity (LRP) bits written to the **I2STXCFG** and **I2SRXCFG** registers. The settings are shown in the table below.

Table 16-9. Audio Formats Configuration

Audio Format	I2STXCFG/I2SRXCFG Register Bit			
	JST	DLY	SCP	LRP
I^2S	0	1	0	1
Left-Justified	0	0	0	0
Right-Justified	1	0	0	0

- Write 0x0140.3DF0 to both the **I2STXCFG** and **I2SRXCFG** registers to program the following configurations:
  - Set the sample size to 16 bits using the SSZ field of the **I2STXCFG** and **I2SRXCFG** registers.
  - Set the system data size to 32 bits using the SDSZ field of the **I2STXCFG** and **I2SRXCFG** registers.
  - Set the Write and Read modes using the WM and RM fields in the **I2STXCFG** and **I2SRXCFG** registers, respectively.

8. Set up the FIFO limits for triggering interrupts (also used for µDMA)
   - Set up the transmit FIFO to trigger when it has less than four sample pairs by writing a 0x0000.0008 to the **I2STXLIMIT** register.
   - Set up the receive FIFO to trigger when there are more than four sample pairs by writing a 0x0000.00008 to the **I2SRXLIMIT** register.
9. Enable interrupts.
   - Enable the transmit FIFO interrupt by setting the FFM bit in the **I2STXISM** register (write 0x0000.0001).
   - Set up the receive FIFO interrupts by setting the FFM bit in the **I2SRXISM** register (write 0x0000.0001).
   - Enable the TX FIFO service request, the TX Error, the RX FIFO service request, and the RX Error interrupts to be sent to the CPU by writing a 0x0000.0033 to the **I2SSIM** register.
10. Enable the Serial Encoder and Serial Decoders by writing a 0x0000.0003 to the **I2SCFG** register.

16.5 Register Map

Table 16-10 on page 834 lists the I^2S registers. The offset listed is a hexadecimal increment to the register's address, relative to the I^2S interface base address of 0x4005.4000. Note that the I^2S module clock must be enabled before the registers can be programmed (see page 263). There must be a delay of 3 system clocks after the I^2S module clock is enabled before any I^2S module registers are accessed.

Table 16-10. Inter-Integrated Circuit Sound (I^2S) Interface Register Map

Offset	Name	Type	Reset	Description	See page
0x000	I2STXFIFO	WO	0x0000.0000	I2S Transmit FIFO Data	835
0x004	I2STXFIFOCFG	R/W	0x0000.0000	I2S Transmit FIFO Configuration	836
0x008	I2STXCFG	R/W	0x1400.7DF0	I2S Transmit Module Configuration	837
0x00C	I2STXLIMIT	R/W	0x0000.0000	I2S Transmit FIFO Limit	839
0x010	I2STXISM	R/W	0x0000.0000	I2S Transmit Interrupt Status and Mask	840
0x018	I2STXLEV	RO	0x0000.0000	I2S Transmit FIFO Level	841
0x800	I2SRXFIFO	RO	0x0000.0000	I2S Receive FIFO Data	842
0x804	I2SRXFIFOCFG	R/W	0x0000.0000	I2S Receive FIFO Configuration	843
0x808	I2SRXCFG	R/W	0x1400.7DF0	I2S Receive Module Configuration	844
0x80C	I2SRXLIMIT	R/W	0x0000.7FFF	I2S Receive FIFO Limit	847
0x810	I2SRXISM	R/W	0x0000.0000	I2S Receive Interrupt Status and Mask	848
0x818	I2SRXLEV	RO	0x0000.0000	I2S Receive FIFO Level	849
0xC00	I2SCFG	R/W	0x0000.0000	I2S Module Configuration	850
0xC10	I2SIM	R/W	0x0000.0000	I2S Interrupt Mask	852
0xC14	I2SRIS	RO	0x0000.0000	I2S Raw Interrupt Status	854
0xC18	I2SMIS	RO	0x0000.0000	I2S Masked Interrupt Status	856
0xC1C	I2SIC	WO	0x0000.0000	I2S Interrupt Clear	858

16.6 Register Descriptions

The remainder of this section lists and describes the I^2S registers, in numerical order by address offset.

Register 1: I²S Transmit FIFO Data (I2STXFIFO), offset 0x000

This register is the 32-bit serial audio transmit data register. In Stereo mode, the data is written left, right, left, right, and so on. The LRS bit in the **I²S Transmit FIFO Configuration (I2STXFIFOCFG)** register can be read to verify the next position expected. In Compact 16-bit mode, bits [31:16] contain the right sample, and bits [15:0] contain the left sample. In Compact 8-bit mode, bits [15:8] contain the right sample, and bits [7:0] contain the left sample. In Mono mode, each 32-bit entry is a single sample.

Note that if the FIFO is full and a write is attempted, a transmit FIFO write error is generated.

I2S Transmit FIFO Data (I2STXFIFO)

Base 0x4005.4000
Offset 0x000
Type WO, reset 0x0000.0000

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

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

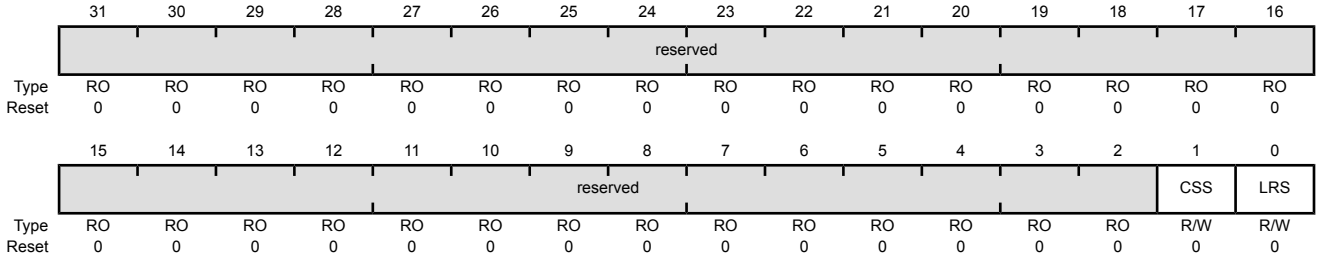
Bit/Field	Name	Type	Reset	Description
31:0	TXFIFO	WO	0x0000.0000	TX Data Serial audio sample data to be transmitted.

Register 2: I^2S Transmit FIFO Configuration (I2STXFIFOCFG), offset 0x004

This register configures the sample for dual-channel operation. In Stereo mode, the LRS bit toggles between left and right samples as the Transmit FIFO is written. The left sample is written first, followed by the right.

I2S Transmit FIFO Configuration (I2STXFIFOCFG)

Base 0x4005.4000
Offset 0x004
Type R/W, reset 0x0000.0000



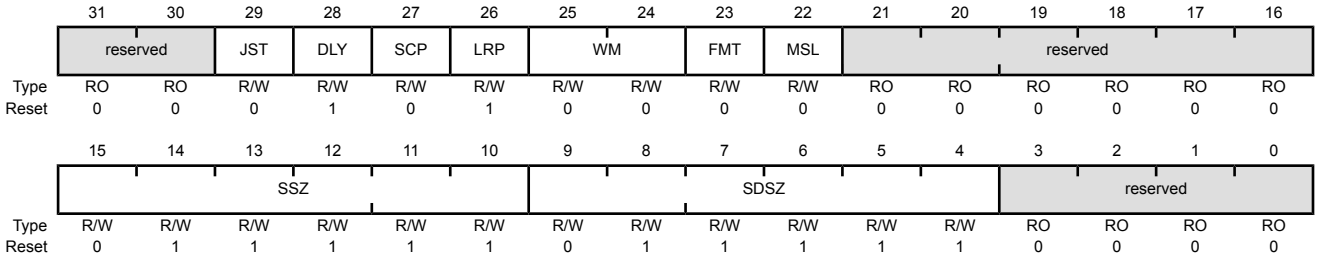
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	CSS	R/W	0	Compact Stereo Sample Size Value Description 0 The transmitter is in Compact 16-bit Stereo Mode with a 16-bit sample size. 1 The transmitter is in Compact 8-bit Stereo Mode with an 8-bit sample size.
0	LRS	R/W	0	Left-Right Sample Indicator Value Description 0 The left sample is the next position. 1 The right sample is the next position. In Mono mode and Compact stereo mode, this bit toggles as if it were in Stereo mode, but it has no meaning and should be ignored.

Register 3: I²S Transmit Module Configuration (I2STXCFG), offset 0x008

This register controls the configuration of the Transmit module.

I2S Transmit Module Configuration (I2STXCFG)

Base 0x4005.4000
Offset 0x008
Type R/W, reset 0x1400.7DF0



Bit/Field	Name	Type	Reset	Description
31:30	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
29	JST	R/W	0	Justification of Output Data Value Description 0 The data is Left-Justified. 1 The data is Right-Justified.
28	DLY	R/W	1	Data Delay Value Description 0 Data is latched on the next latching edge of I2S0TXSCK as defined by the SCP bit. This bit should be clear in Left-Justified or Right-Justified mode. 1 A one-I2S0TXSCK delay from the edge of I2S0TXWS is inserted before data is latched. This bit should be set in I²S mode.
27	SCP	R/W	0	SCLK Polarity Value Description 0 Data and the I2S0TXWS signal (when the MSL bit is set) are launched on the falling edge of I2S0TXSCK. 1 Data and the I2S0TXWS signal (when the MSL bit is set) are launched on the rising edge of I2S0TXSCK.
26	LRP	R/W	1	Left/Right Clock Polarity Value Description 0 I2S0TXWS is high during the transmission of the left channel data. 1 I2S0TXWS is high during the transmission of the right channel data.

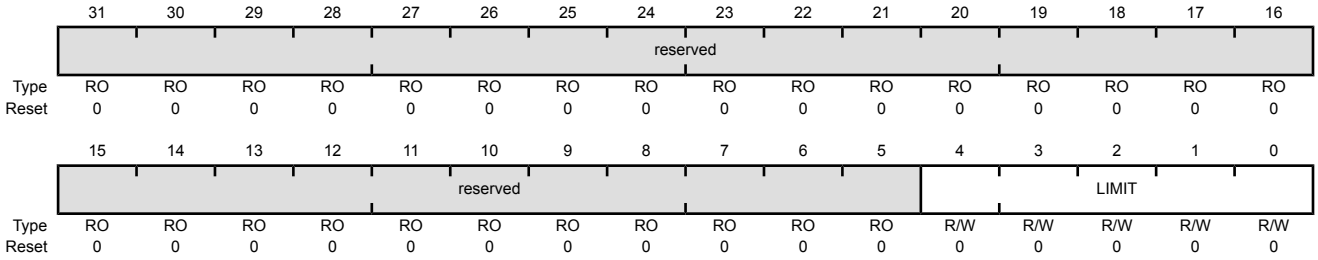
Bit/Field	Name	Type	Reset	Description
25:24	WM	R/W	0x0	Write Mode This bit field selects the mode in which the transmit data is stored in the FIFO and transmitted. Value Description 0x0 Stereo mode 0x1 Compact Stereo mode Left/Right sample packed. Refer to **I2STXFIFOCFG** for 8/16-bit sample size selection. 0x2 Mono mode 0x3 reserved
23	FMT	R/W	0	FIFO Empty Value Description 0 All zeroes are transmitted if the FIFO is empty. 1 The last sample is transmitted if the FIFO is empty.
22	MSL	R/W	0	SCLK Master/Slave Source of serial bit clock (`I2S0TXSCK`) and Word Select (`I2S0TXWS`). Value Description 0 The transmitter is a slave using the externally driven `I2S0TXSCK` and `I2S0TXWS` signals. 1 The transmitter is a master using the internally generated `I2S0TXSCK` and `I2S0TXWS` signals.
21:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:10	SSZ	R/W	0x1F	Sample Size This field contains the number of bits minus one in the sample. **Note:** This field is only used in Right-Justified mode. Unused bits are not masked.
9:4	SDSZ	R/W	0x1F	System Data Size This field contains the number of bits minus one during the high or low phase of the `I2S0TXWS` signal.
3:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 4: I²S Transmit FIFO Limit (I2STXLIMIT), offset 0x00C

This register sets the lower FIFO limit at which a FIFO service request is issued.

I2S Transmit FIFO Limit (I2STXLIMIT)

Base 0x4005.4000
Offset 0x00C
Type R/W, reset 0x0000.0000



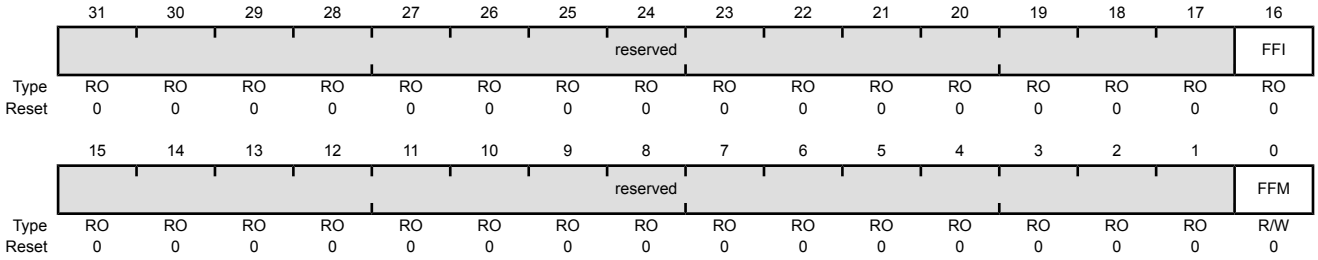
Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4:0	LIMIT	R/W	0x00	FIFO Limit This field sets the FIFO level at which a FIFO service request is issued, generating an interrupt or a µDMA transfer request. The transmit FIFO generates a service request when the number of items in the FIFO is less than the level specified by the `LIMIT` field. For example, if the `LIMIT` field is set to 8, then a service request is generated when there are less than 8 samples remaining in the transmit FIFO.

Register 5: I^2S Transmit Interrupt Status and Mask (I2STXISM), offset 0x010

This register indicates the transmit interrupt status and interrupt masking control.

I2S Transmit Interrupt Status and Mask (I2STXISM)

Base 0x4005.4000
Offset 0x010
Type R/W, reset 0x0000.0000



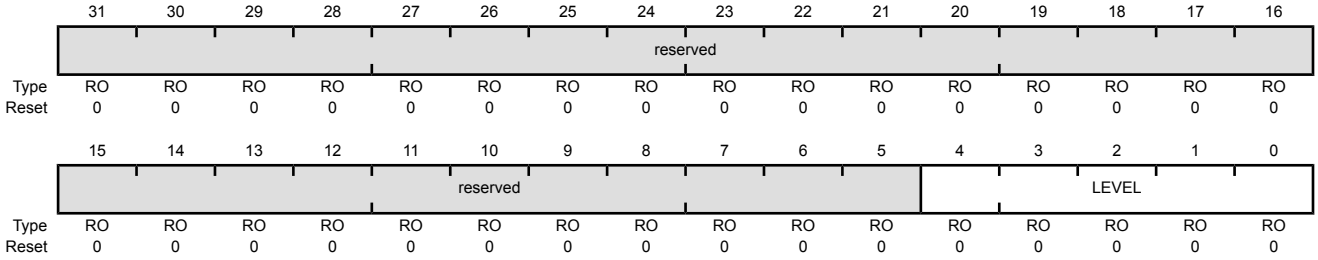
Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	FFI	RO	0	Transmit FIFO Service Request Interrupt Value Description 0 The FIFO level is equal to or above the FIFO limit. 1 The FIFO level is below the FIFO limit.
15:1	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	FFM	R/W	0	FIFO Interrupt Mask Value Description 0 The FIFO interrupt is masked and not sent to the CPU. 1 The FIFO interrupt is enabled to be sent to the interrupt controller.

Register 6: I²S Transmit FIFO Level (I2STXLEV), offset 0x018

The number of samples in the transmit FIFO can be read using the **I2STXLEV** register. The value ranges from 0 to 16. Stereo and Compact Stereo sample-pairs are counted as two. Mono samples also increment the count by two. For example, the LEVEL field is set to eight if there are four Mono samples.

I2S Transmit FIFO Level (I2STXLEV)

Base 0x4005.4000
Offset 0x018
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4:0	LEVEL	RO	0x00	Number of Audio Samples This field contains the number of samples in the FIFO.

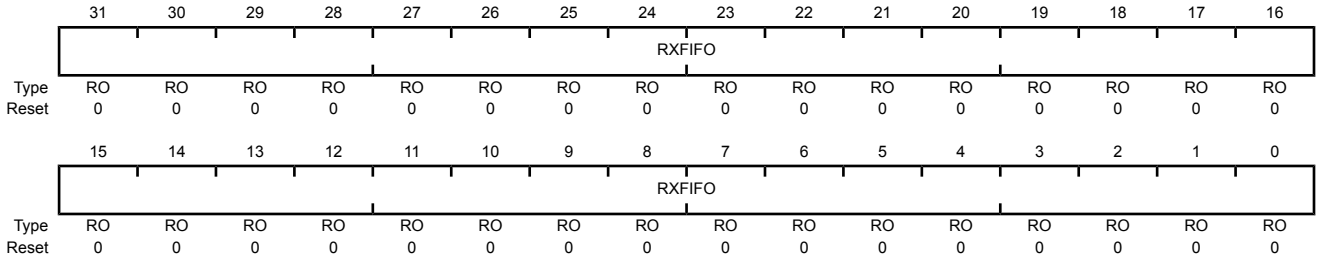
Register 7: I^2S Receive FIFO Data (I2SRXFIFO), offset 0x800

Important: This register is read-sensitive. See the register description for details.

This register is the 32-bit serial audio receive data register. In Stereo mode, the data is read left, right, left, right, and so on. The LRS bit in the **I^2S Receive FIFO Configuration (I2SRXFIFOCFG)** register can be read to verify the next position expected. In Compact 16-bit mode, bits [31:16] contain the right sample, and bits [15:0] contain the left sample. In Compact 8-bit mode, bits [15:8] contain the right sample, and bits [7:0] contain the left sample. In Mono mode, each 32-bit entry is a single sample. If the FIFO is empty, a read of this register returns a value of 0x0000.0000 and generates a receive FIFO read error.

I2S Receive FIFO Data (I2SRXFIFO)

Base 0x4005.4000
Offset 0x800
Type RO, reset 0x0000.0000



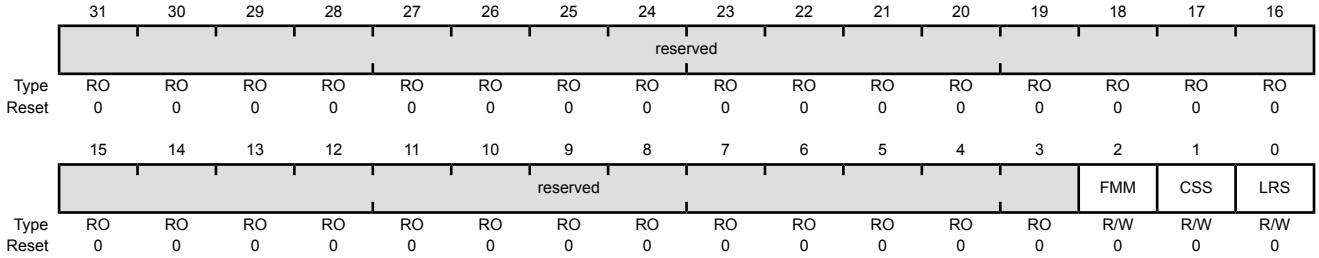
Bit/Field	Name	Type	Reset	Description
31:0	RXFIFO	RO	0x0000.0000	RX Data Serial audio sample data received. The read of an empty FIFO returns a value of 0x0.

Register 8: I²S Receive FIFO Configuration (I2SRXFIFOCFG), offset 0x804

This register configures the sample for dual-channel operation. In Stereo mode, the LRS bit toggles between Left and Right as the samples are read from the receive FIFO. In Mono mode, both the left and right samples are stored in the FIFO. The FMM bit can be used to read only the left or right sample as determined by the LRP bit. In Compact Stereo 8- or 16-bit mode, both the left and right samples are read in one access from the FIFO.

I2S Receive FIFO Configuration (I2SRXFIFOCFG)

Base 0x4005.4000
Offset 0x804
Type R/W, reset 0x0000.0000



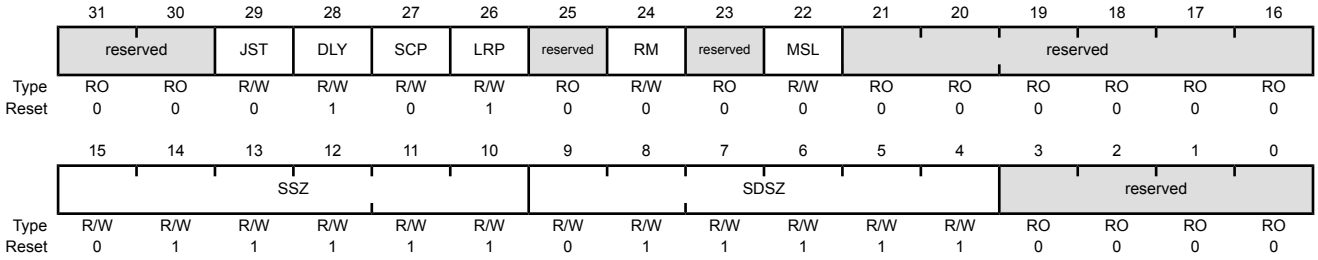
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	FMM	R/W	0	FIFO Mono Mode Value Description 0 The receiver is in Stereo Mode. 1 The receiver is in Mono mode. If the LRP bit in the **I2SRXCFG** register is clear, data is read while the I2S0RXWS signal is low (Right Channel); if the LRP bit is set, data is read while the I2S0RXWS signal is high (Left Channel).
1	CSS	R/W	0	Compact Stereo Sample Size Value Description 0 The receiver is in Compact 16-bit Stereo Mode with a 16-bit sample size. 1 The receiver is in Compact 8-bit Stereo Mode with a 8-bit sample size.
0	LRS	R/W	0	Left-Right Sample Indicator Value Description 0 The left sample is the next position to be read. 1 The right sample is the next position to be read. This bit is only meaningful in Compact Stereo Mode.

Register 9: I^2S Receive Module Configuration (I2SRXCFG), offset 0x808

This register controls the configuration of the receive module.

I2S Receive Module Configuration (I2SRXCFG)

Base 0x4005.4000
Offset 0x808
Type R/W, reset 0x1400.7DF0



Bit/Field	Name	Type	Reset	Description
31:30	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
29	JST	R/W	0	Justification of Input Data Value Description 0 The data is Left-Justified. 1 The data is Right-Justified.
28	DLY	R/W	1	Data Delay Value Description 0 Data is latched on the next latching edge of I2S0RXSCK as defined by the SCP bit. This bit should be clear in Left-Justified or Right-Justified mode. 1 A one-I2S0RXSCK delay from the edge of I2S0RXWS is inserted before data is latched. This bit should be set in I^2S mode.
27	SCP	R/W	0	SCLK Polarity Value Description 0 Data is latched on the rising edge and the I2S0RXWS signal (when the MSL bit is set) is launched on the falling edge of I2S0RXSCK. 1 Data is latched on the falling edge and the I2S0RXWS signal (when the MSL bit is set) is launched on the rising edge of I2S0RXSCK.

Bit/Field	Name	Type	Reset	Description
26	LRP	R/W	1	Left/Right Clock Polarity Value Description 0 In Stereo mode, I2S0RXWS is high during the transmission of the left channel data. In Mono mode, data is read while the I2S0RXWS signal is low (Right Channel). 1 In Stereo mode, I2S0RXWS is high during the transmission of the right channel data. In Mono mode, data is read while the I2S0RXWS signal is high (Left Channel).
25	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24	RM	R/W	0	Read Mode This bit selects the mode in which the receive data is received and stored in the FIFO. Value Description 0 Stereo/Mono mode **I2SRXFIFOCFG** FMM bit specifies Stereo or Mono FIFO read behavior. 1 Compact Stereo mode Left/Right sample packed. Refer to **I2SRXFIFOCFG** for 8/16-bit sample size selection.
23	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22	MSL	R/W	0	SCLK Master/Slave Value Description 0 The receiver is a slave and uses the externally driven I2S0RXSCK and I2S0RXWS signals. 1 The receiver is a master and uses the internally generated I2S0RXSCK and I2S0RXWS signals.
21:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:10	SSZ	R/W	0x1F	Sample Size This field contains the number of bits minus one in the sample.
9:4	SDSZ	R/W	0x1F	System Data Size This field contains the number of bits minus one during the high or low phase of the I2S0RXWS signal.

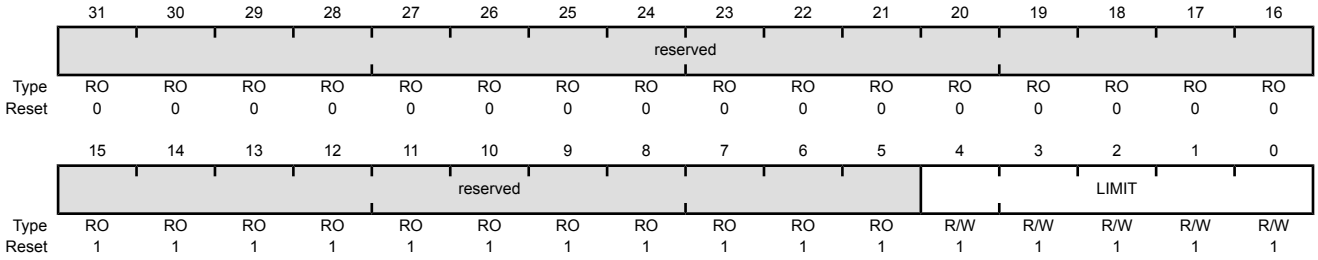
Bit/Field	Name	Type	Reset	Description
3:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 10: I²S Receive FIFO Limit (I2SRXLIMIT), offset 0x80C

This register sets the upper FIFO limit at which a FIFO service request is issued.

I2S Receive FIFO Limit (I2SRXLIMIT)

Base 0x4005.4000
Offset 0x80C
Type R/W, reset 0x0000.7FFF



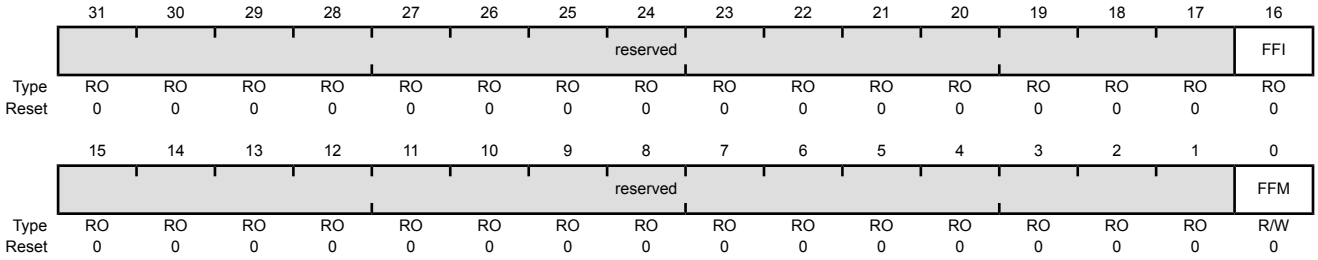
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:5	reserved	RO	0x7FF	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4:0	LIMIT	R/W	0x1F	FIFO Limit This field sets the FIFO level at which a FIFO service request is issued, generating an interrupt or a µDMA transfer request. The receive FIFO generates a service request when the number of items in the FIFO is greater than the level specified by the `LIMIT` field. For example, if the `LIMIT` field is set to 4, then a service request is generated when there are more than 4 samples remaining in the transmit FIFO.

Register 11: I^2S Receive Interrupt Status and Mask (I2SRXISM), offset 0x810

This register indicates the receive interrupt status and interrupt masking control.

I2S Receive Interrupt Status and Mask (I2SRXISM)

Base 0x4005.4000
Offset 0x810
Type R/W, reset 0x0000.0000



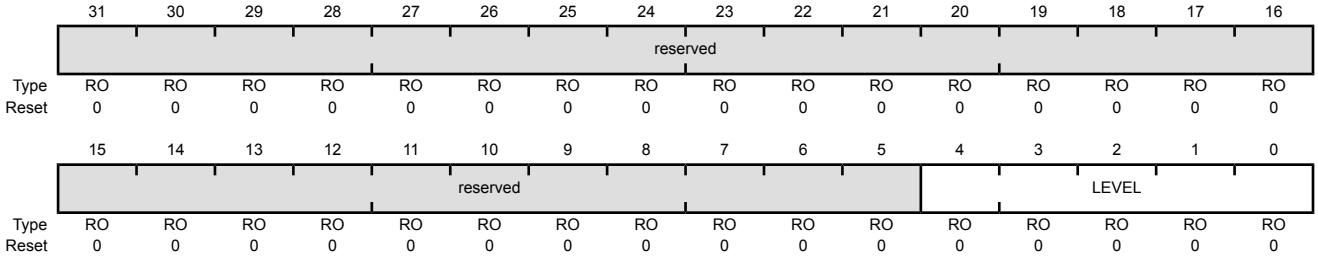
Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	FFI	RO	0	Receive FIFO Service Request Interrupt Value Description 0 The FIFO level is equal to or below the FIFO limit. 1 The FIFO level is above the FIFO limit.
15:1	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	FFM	R/W	0	FIFO Interrupt Mask Value Description 0 The FIFO interrupt is masked and not sent to the CPU. 1 The FIFO interrupt is enabled to be sent to the interrupt controller.

Register 12: I²S Receive FIFO Level (I2SRXLEV), offset 0x818

The number of samples in the receive FIFO can be read using the **I2SRXLEV** register. The value ranges from 0 to 16. Stereo and Compact Stereo sample pairs are counted as two. Mono samples also increment the count by two. For example, the LEVEL field is set to eight if there are four Mono samples.

I2S Receive FIFO Level (I2SRXLEV)

Base 0x4005.4000
Offset 0x818
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4:0	LEVEL	RO	0x00	Number of Audio Samples This field contains the number of samples in the FIFO.

Register 13: I^2S Module Configuration (I2SCFG), offset 0xC00

This register enables the transmit and receive serial engines and sets the source of the I2S0TXMCLK and I2S0RXMCLK signals.

I2S Module Configuration (I2SCFG)

Base 0x4005.4000
Offset 0xC00
Type R/W, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	RXSLV	R/W	0	Use External I2S0RXMCLK Value Description 0 The receiver uses the internally generated MCLK as the I2S0RXMCLK signal. See "Clock Control" on page 827 for information on how to program the I2S0RXMCLK. 1 The receiver uses the externally driven I2S0RXMCLK signal.
4	TXSLV	R/W	0	Use External I2S0TXMCLK Value Description 0 The transmitter uses the internally generated MCLK as the I2S0TXMCLK signal. See "Clock Control" on page 827 for information on how to program the I2S0TXMCLK. 1 The transmitter uses the externally driven I2S0TXMCLK signal.
3:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	RXEN	R/W	0	Serial Receive Engine Enable Value Description 0 Disables the serial receive engine. 1 Enables the serial receive engine.

Bit/Field	Name	Type	Reset	Description
0	TXEN	R/W	0	Serial Transmit Engine Enable

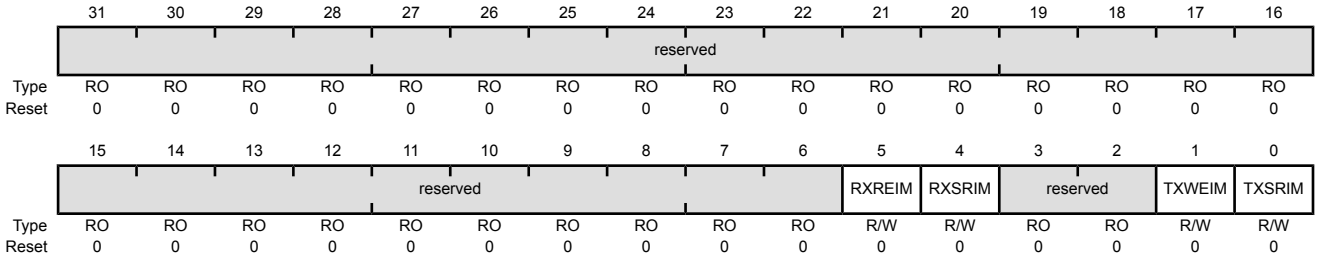
Value	Description
0	Disables the serial transmit engine.
1	Enables the serial transmit engine.

Register 14: I^2S Interrupt Mask (I2SIM), offset 0xC10

This register masks the interrupts to the CPU.

I2S Interrupt Mask (I2SIM)

Base 0x4005.4000
Offset 0xC10
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	RXREIM	R/W	0	Receive FIFO Read Error Value Description 0 The receive FIFO read error interrupt is masked and not sent to the CPU. 1 The receive FIFO read error is enabled to be sent to the interrupt controller.
4	RXSRIM	R/W	0	Receive FIFO Service Request Value Description 0 The receive FIFO service request interrupt is masked and not sent to the CPU. 1 The receive FIFO service request is enabled to be sent to the interrupt controller.
3:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	TXWEIM	R/W	0	Transmit FIFO Write Error Value Description 0 The transmit FIFO write error interrupt is masked and not sent to the CPU. 1 The transmit FIFO write error is enabled to be sent to the interrupt controller.

Bit/Field	Name	Type	Reset	Description
0	TXSRIM	R/W	0	Transmit FIFO Service Request

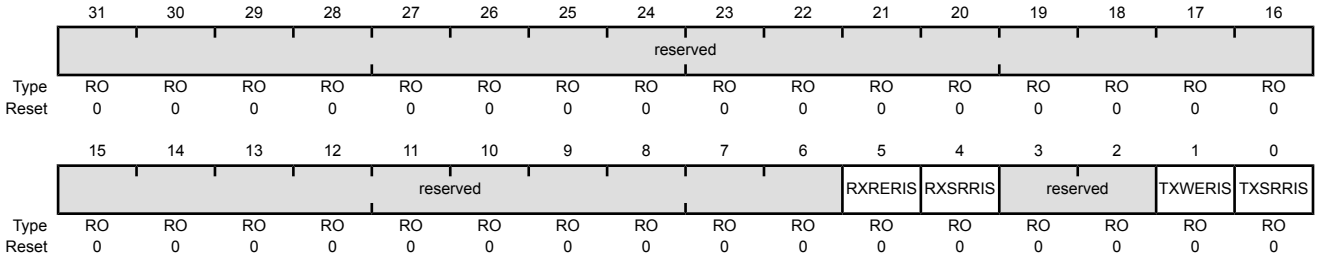
Value	Description
0	The transmit FIFO service request interrupt is masked and not sent to the CPU.
1	The transmit FIFO service request is enabled to be sent to the interrupt controller.

Register 15: I^2S Raw Interrupt Status (I2SRIS), offset 0xC14

This register reads the unmasked interrupt status.

I2S Raw Interrupt Status (I2SRIS)

Base 0x4005.4000
Offset 0xC14
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	RXRERIS	RO	0	Receive FIFO Read Error Value Description 1 A receive FIFO read error interrupt has occurred. 0 No interrupt This bit is cleared by setting the RXREIC bit in the **I2SIC** register.
4	RXSRRIS	RO	0	Receive FIFO Service Request Value Description 1 A receive FIFO service request interrupt has occurred. 0 No interrupt This bit is cleared when the level in the receive FIFO has risen to a value greater than the value programmed in the LIMIT field in the **I2SRXLIMIT** register.
3:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	TXWERIS	RO	0	Transmit FIFO Write Error Value Description 1 A transmit FIFO write error interrupt has occurred. 0 No interrupt This bit is cleared by setting the TXWEIC bit in the **I2SIC** register.

Bit/Field	Name	Type	Reset	Description
0	TXSRRIS	RO	0	Transmit FIFO Service Request

Value	Description
1	A transmit FIFO service request interrupt has occurred.
0	No interrupt

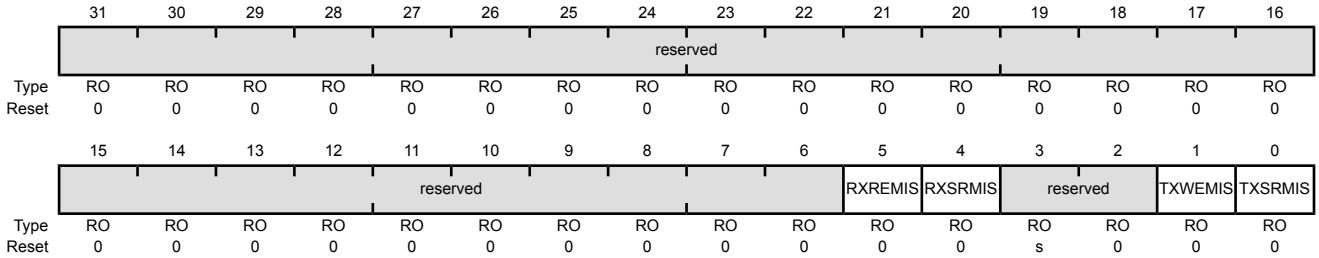
This bit is cleared when the level in the transmit FIFO has fallen to a value less than the value programmed in the `LIMIT` field in the **I2STXLIMIT** register.

Register 16: I^2S Masked Interrupt Status (I2SMIS), offset 0xC18

This register reads the masked interrupt status. The mask is defined in the **I2SIM** register.

I2S Masked Interrupt Status (I2SMIS)

Base 0x4005.4000
Offset 0xC18
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	RXREMIS	RO	0	Receive FIFO Read Error Value Description 1 An unmasked interrupt was signaled due to a receive FIFO read error. 0 An interrupt has not occurred or is masked. This bit is cleared by setting the RXREIC bit in the **I2SIC** register.
4	RXSRMIS	RO	0	Receive FIFO Service Request Value Description 1 An unmasked interrupt was signaled due to a receive FIFO service request. 0 An interrupt has not occurred or is masked. This bit is cleared when the level in the receive FIFO has risen to a value greater than the value programmed in the LIMIT field in the **I2SRXLIMIT** register.
3:2	reserved	RO	0s0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	TXWEMIS	RO	0	Transmit FIFO Write Error Value Description 1 An unmasked interrupt was signaled due to a transmit FIFO write error. 0 An interrupt has not occurred or is masked. This bit is cleared by setting the TXWEIC bit in the **I2SIC** register.

Bit/Field	Name	Type	Reset	Description
0	TXSRMIS	RO	0	Transmit FIFO Service Request

Value	Description
1	An unmasked interrupt was signaled due to a transmit FIFO service request.
0	An interrupt has not occurred or is masked.

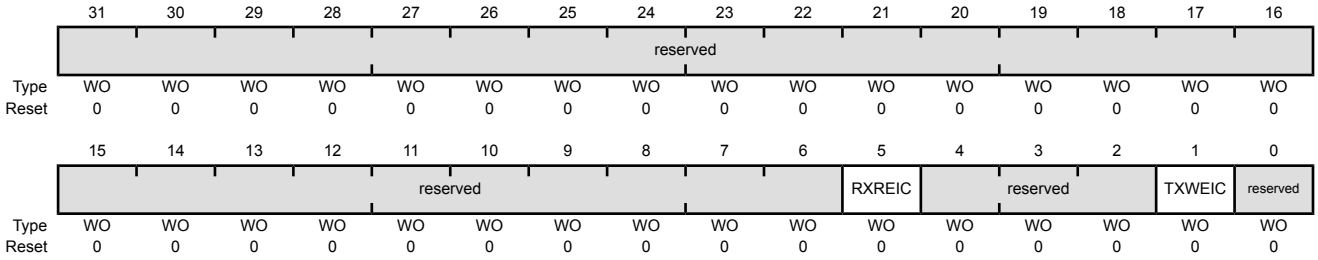
This bit is cleared when the level in the transmit FIFO has fallen to a value less than the value programmed in the `LIMIT` field in the **I2STXLIMIT** register.

Register 17: I^2S Interrupt Clear (I2SIC), offset 0xC1C

Writing a 1 to a bit in this register clears the corresponding interrupt.

I2S Interrupt Clear (I2SIC)

Base 0x4005.4000
Offset 0xC1C
Type WO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:6	reserved	WO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	RXREIC	WO	0	Receive FIFO Read Error Writing a 1 to this bit clears the RXRERIS bit in the **I2CRIS** register and the RXREMIS bit in the **I2CMIS** register.
4:2	reserved	WO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	TXWEIC	WO	0	Transmit FIFO Write Error Writing a 1 to this bit clears the TXWERIS bit in the **I2CRIS** register and the TXWEMIS bit in the **I2CMIS** register.
0	reserved	WO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

17 Controller Area Network (CAN) Module

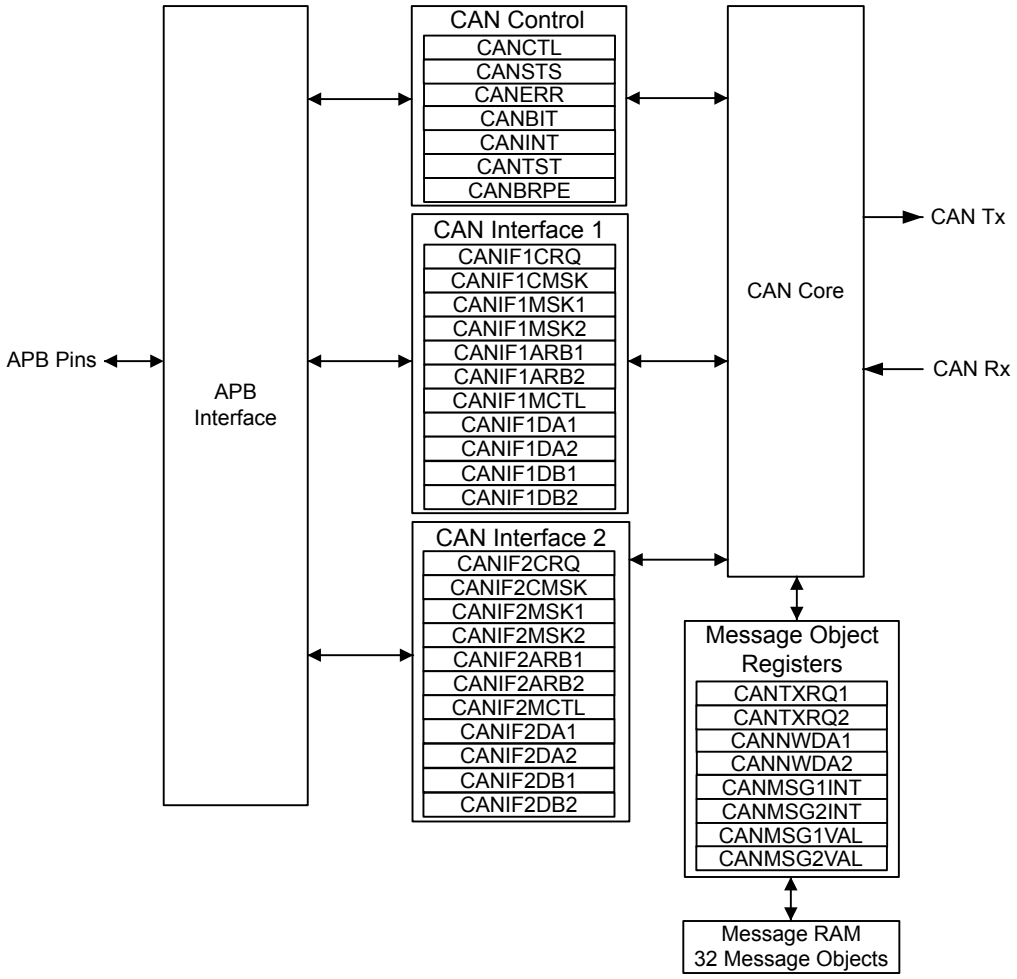
Controller Area Network (CAN) is a multicast, shared serial bus standard for connecting electronic control units (ECUs). CAN was specifically designed to be robust in electromagnetically-noisy environments and can utilize a differential balanced line like RS-485 or a more robust twisted-pair wire. Originally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

The Stellaris® LM3S9U81 microcontroller includes three CAN units with the following features:

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

17.1 Block Diagram

Figure 17-1. CAN Controller Block Diagram



17.2 Signal Description

The following table lists the external signals of the CAN controller and describes the function of each. The CAN controller signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the CAN signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 419) should be set to choose the CAN controller function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 437) to assign the CAN signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 395.

Table 17-1. Controller Area Network Signals (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
CAN0Rx	10 30 34 92	PD0 (2) PA4 (5) PA6 (6) PB4 (5)	I	TTL	CAN module 0 receive.
CAN0Tx	11 31 35 91	PD1 (2) PA5 (5) PA7 (6) PB5 (5)	O	TTL	CAN module 0 transmit.
CAN1Rx	47	PF0 (1)	I	TTL	CAN module 1 receive.
CAN1Tx	61	PF1 (1)	O	TTL	CAN module 1 transmit.
CAN2Rx	6	PE4 (2)	I	TTL	CAN module 2 receive.
CAN2Tx	5	PE5 (2)	O	TTL	CAN module 2 transmit.

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

Table 17-2. Controller Area Network Signals (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
CAN0Rx	G1 L5 L6 A6	PD0 (2) PA4 (5) PA6 (6) PB4 (5)	I	TTL	CAN module 0 receive.
CAN0Tx	G2 M5 M6 B7	PD1 (2) PA5 (5) PA7 (6) PB5 (5)	O	TTL	CAN module 0 transmit.
CAN1Rx	M9	PF0 (1)	I	TTL	CAN module 1 receive.
CAN1Tx	H12	PF1 (1)	O	TTL	CAN module 1 transmit.
CAN2Rx	B2	PE4 (2)	I	TTL	CAN module 2 receive.
CAN2Tx	B3	PE5 (2)	O	TTL	CAN module 2 transmit.

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

17.3 Functional Description

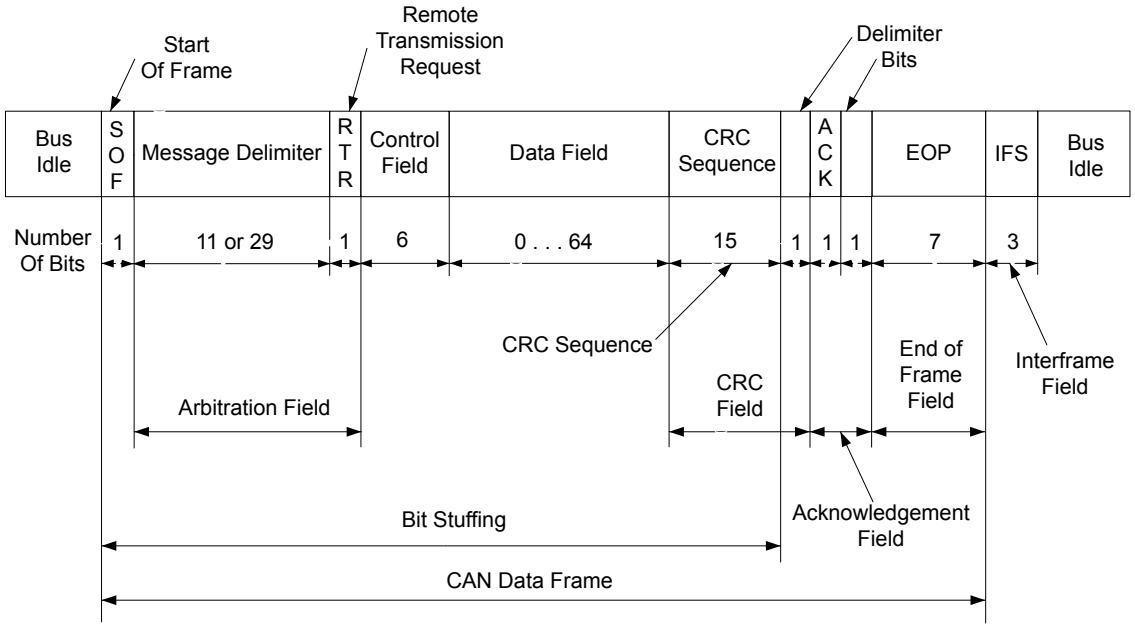
The Stellaris CAN controller conforms to the CAN protocol version 2.0 (parts A and B). Message transfers that include data, remote, error, and overload frames with an 11-bit identifier (standard) or a 29-bit identifier (extended) are supported. Transfer rates can be programmed up to 1 Mbps.

The CAN module consists of three major parts:

- CAN protocol controller and message handler
- Message memory
- CAN register interface

A data frame contains data for transmission, whereas a remote frame contains no data and is used to request the transmission of a specific message object. The CAN data/remote frame is constructed as shown in Figure 17-2.

Figure 17-2. CAN Data/Remote Frame



The protocol controller transfers and receives the serial data from the CAN bus and passes the data on to the message handler. The message handler then loads this information into the appropriate message object based on the current filtering and identifiers in the message object memory. The message handler is also responsible for generating interrupts based on events on the CAN bus.

The message object memory is a set of 32 identical memory blocks that hold the current configuration, status, and actual data for each message object. These memory blocks are accessed via either of the CAN message object register interfaces.

The message memory is not directly accessible in the Stellaris memory map, so the Stellaris CAN controller provides an interface to communicate with the message memory via two CAN interface register sets for communicating with the message objects. The message object memory cannot be directly accessed, so these two interfaces must be used to read or write to each message object. The two message object interfaces allow parallel access to the CAN controller message objects when multiple objects may have new information that must be processed. In general, one interface is used for transmit data and one for receive data.

17.3.1 Initialization

To use the CAN controller, the peripheral clock must be enabled using the **RCGC0** register (see page 255). In addition, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register (see page 272). To find out which GPIO port to enable, refer to Table 22-4 on page 1144. Set the GPIO AFSEL bits for the appropriate pins (see page 419). Configure the PMCn fields in the **GPIOPCTL** register to assign the CAN signals to the appropriate pins. See page 437 and Table 22-5 on page 1151.

Software initialization is started by setting the INIT bit in the **CAN Control (CANCTL)** register (with software or by a hardware reset) or by going bus-off, which occurs when the transmitter's error counter exceeds a count of 255. While INIT is set, all message transfers to and from the CAN bus are stopped and the CANnTX signal is held High. Entering the initialization state does not change the configuration of the CAN controller, the message objects, or the error counters. However, some configuration registers are only accessible while in the initialization state.

To initialize the CAN controller, set the **CAN Bit Timing (CANBIT)** register and configure each message object. If a message object is not needed, label it as not valid by clearing the MSGVAL bit in the **CAN IFn Arbitration 2 (CANIFnARB2)** register. Otherwise, the whole message object must be initialized, as the fields of the message object may not have valid information, causing unexpected results. Both the INIT and CCE bits in the **CANCTL** register must be set in order to access the **CANBIT** register and the **CAN Baud Rate Prescaler Extension (CANBRPE)** register to configure the bit timing. To leave the initialization state, the INIT bit must be cleared. Afterwards, the internal Bit Stream Processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (indicating a bus idle condition) before it takes part in bus activities and starts message transfers. Message object initialization does not require the CAN to be in the initialization state and can be done on the fly. However, message objects should all be configured to particular identifiers or set to not valid before message transfer starts. To change the configuration of a message object during normal operation, clear the MSGVAL bit in the **CANIFnARB2** register to indicate that the message object is not valid during the change. When the configuration is completed, set the MSGVAL bit again to indicate that the message object is once again valid.

17.3.2 Operation

Two sets of CAN Interface Registers (**CANIF1x** and **CANIF2x**) are used to access the message objects in the Message RAM. The CAN controller coordinates transfers to and from the Message RAM to and from the registers. The two sets are independent and identical and can be used to queue transactions. Generally, one interface is used to transmit data and one is used to receive data.

Once the CAN module is initialized and the INIT bit in the **CANCTL** register is cleared, the CAN module synchronizes itself to the CAN bus and starts the message transfer. As each message is received, it goes through the message handler's filtering process, and if it passes through the filter, is stored in the message object specified by the MNUM bit in the **CAN IFn Command Request (CANIFnCRQ)** register. The whole message (including all arbitration bits, data-length code, and eight data bytes) is stored in the message object. If the Identifier Mask (the MSK bits in the **CAN IFn Mask 1** and **CAN IFn Mask 2 (CANIFnMSKn)** registers) is used, the arbitration bits that are masked to "don't care" may be overwritten in the message object.

The CPU may read or write each message at any time via the CAN Interface Registers. The message handler guarantees data consistency in case of concurrent accesses.

The transmission of message objects is under the control of the software that is managing the CAN hardware. Message objects can be used for one-time data transfers or can be permanent message objects used to respond in a more periodic manner. Permanent message objects have all arbitration and control set up, and only the data bytes are updated. At the start of transmission, the appropriate TXRQST bit in the **CAN Transmission Request n (CANTXRQn)** register and the NEWDAT bit in the **CAN New Data n (CANNWDAn)** register are set. If several transmit messages are assigned to the same message object (when the number of message objects is not sufficient), the whole message object has to be configured before the transmission of this message is requested.

The transmission of any number of message objects may be requested at the same time; they are transmitted according to their internal priority, which is based on the message identifier (MNUM) for the message object, with 1 being the highest priority and 32 being the lowest priority. Messages may be updated or set to not valid any time, even when their requested transmission is still pending. The old data is discarded when a message is updated before its pending transmission has started. Depending on the configuration of the message object, the transmission of a message may be requested autonomously by the reception of a remote frame with a matching identifier.

Transmission can be automatically started by the reception of a matching remote frame. To enable this mode, set the RMTEN bit in the **CAN IFn Message Control (CANIFnMCTL)** register. A matching received remote frame causes the TXRQST bit to be set, and the message object automatically transfers its data or generates an interrupt indicating a remote frame was requested. A remote frame can be strictly a single message identifier, or it can be a range of values specified in the message object. The CAN mask registers, **CANIFnMSKn**, configure which groups of frames are identified as remote frame requests. The UMASK bit in the **CANIFnMCTL** register enables the MSK bits in the **CANIFnMSKn** register to filter which frames are identified as a remote frame request. The MXTD bit in the **CANIFnMSK2** register should be set if a remote frame request is expected to be triggered by 29-bit extended identifiers.

17.3.3 Transmitting Message Objects

If the internal transmit shift register of the CAN module is ready for loading, and if a data transfer is not occurring between the CAN Interface Registers and message RAM, the valid message object with the highest priority that has a pending transmission request is loaded into the transmit shift register by the message handler and the transmission is started. The message object's NEWDAT bit in the **CANNWDAn** register is cleared. After a successful transmission, and if no new data was written to the message object since the start of the transmission, the TXRQST bit in the **CANTXRQn** register is cleared. If the CAN controller is configured to interrupt on a successful transmission of a message object, (the TXIE bit in the **CAN IFn Message Control (CANIFnMCTL)** register is set), the INTPND bit in the **CANIFnMCTL** register is set after a successful transmission. If the CAN module has lost the arbitration or if an error occurred during the transmission, the message is re-transmitted as soon as the CAN bus is free again. If, meanwhile, the transmission of a message with higher priority has been requested, the messages are transmitted in the order of their priority.

17.3.4 Configuring a Transmit Message Object

The following steps illustrate how to configure a transmit message object.

1. In the **CAN IFn Command Mask (CANIFnCMASK)** register:
   - Set the WRNRD bit to specify a write to the **CANIFnCMASK** register; specify whether to transfer the IDMASK, DIR, and MXTD of the message object into the **CAN IFn** registers using the MASK bit
   - Specify whether to transfer the ID, DIR, XTD, and MSGVAL of the message object into the interface registers using the ARB bit
   - Specify whether to transfer the control bits into the interface registers using the CONTROL bit
   - Specify whether to clear the INTPND bit in the **CANIFnMCTL** register using the CLRINTPND bit
   - Specify whether to clear the NEWDAT bit in the **CANNWDAn** register using the NEWDAT bit
   - Specify which bits to transfer using the DATAA and DATAB bits
2. In the **CANIFnMSK1** register, use the MSK[15:0] bits to specify which of the bits in the 29-bit or 11-bit message identifier are used for acceptance filtering. Note that MSK[15:0] in this register are used for bits [15:0] of the 29-bit message identifier and are not used for an 11-bit identifier. A value of 0x00 enables all messages to pass through the acceptance filtering. Also

note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the **CANIFnMCTL** register.

3. In the **CANIFnMSK2** register, use the MSK[12:0] bits to specify which of the bits in the 29-bit or 11-bit message identifier are used for acceptance filtering. Note that MSK[12:0] are used for bits [28:16] of the 29-bit message identifier; whereas MSK[12:2] are used for bits [10:0] of the 11-bit message identifier. Use the MXTD and MDIR bits to specify whether to use XTD and DIR for acceptance filtering. A value of 0x00 enables all messages to pass through the acceptance filtering. Also note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the **CANIFnMCTL** register.

4. For a 29-bit identifier, configure ID[15:0] in the **CANIFnARB1** register for bits [15:0] of the message identifier and ID[12:0] in the **CANIFnARB2** register for bits [28:16] of the message identifier. Set the XTD bit to indicate an extended identifier; set the DIR bit to indicate transmit; and set the MSGVAL bit to indicate that the message object is valid.

5. For an 11-bit identifier, disregard the **CANIFnARB1** register and configure ID[12:2] in the **CANIFnARB2** register for bits [10:0] of the message identifier. Clear the XTD bit to indicate a standard identifier; set the DIR bit to indicate transmit; and set the MSGVAL bit to indicate that the message object is valid.

6. In the **CANIFnMCTL** register:

   - Optionally set the UMASK bit to enable the mask (MSK, MXTD, and MDIR specified in the **CANIFnMSK1** and **CANIFnMSK2** registers) for acceptance filtering
   - Optionally set the TXIE bit to enable the INTPND bit to be set after a successful transmission
   - Optionally set the RMTEN bit to enable the TXRQST bit to be set on the reception of a matching remote frame allowing automatic transmission
   - Set the EOB bit for a single message object
   - Configure the DLC[3:0] field to specify the size of the data frame. Take care during this configuration not to set the NEWDAT, MSGLST, INTPND or TXRQST bits.

7. Load the data to be transmitted into the **CAN IFn Data (CANIFnDA1, CANIFnDA2, CANIFnDB1, CANIFnDB2)** registers. Byte 0 of the CAN data frame is stored in DATA[7:0] in the **CANIFnDA1** register.

8. Program the number of the message object to be transmitted in the MNUM field in the **CAN IFn Command Request (CANIFnCRQ)** register.

9. When everything is properly configured, set the TXRQST bit in the **CANIFnMCTL** register. Once this bit is set, the message object is available to be transmitted, depending on priority and bus availability. Note that setting the RMTEN bit in the **CANIFnMCTL** register can also start message transmission if a matching remote frame has been received.

17.3.5 Updating a Transmit Message Object

The CPU may update the data bytes of a Transmit Message Object any time via the CAN Interface Registers and neither the MSGVAL bit in the **CANIFnARB2** register nor the TXRQST bits in the **CANIFnMCTL** register have to be cleared before the update.

Even if only some of the data bytes are to be updated, all four bytes of the corresponding **CANIFnDAn**/**CANIFnDBn** register have to be valid before the content of that register is transferred to the message object. Either the CPU must write all four bytes into the **CANIFnDAn**/**CANIFnDBn** register or the message object is transferred to the **CANIFnDAn**/**CANIFnDBn** register before the CPU writes the new data bytes.

In order to only update the data in a message object, the WRNRD, DATAA and DATAB bits in the **CANIFnMSKn** register are set, followed by writing the updated data into **CANIFnDA1**, **CANIFnDA2**, **CANIFnDB1**, and **CANIFnDB2** registers, and then the number of the message object is written to the MNUM field in the **CAN IFn Command Request (CANIFnCRQ)** register. To begin transmission of the new data as soon as possible, set the TXRQST bit in the **CANIFnMSKn** register.

To prevent the clearing of the TXRQST bit in the **CANIFnMCTL** register at the end of a transmission that may already be in progress while the data is updated, the NEWDAT and TXRQST bits have to be set at the same time in the **CANIFnMCTL** register. When these bits are set at the same time, NEWDAT is cleared as soon as the new transmission has started.

17.3.6 Accepting Received Message Objects

When the arbitration and control field (the ID and XTD bits in the **CANIFnARB2** and the RMTEN and DLC[3:0] bits of the **CANIFnMCTL** register) of an incoming message is completely shifted into the CAN controller, the message handling capability of the controller starts scanning the message RAM for a matching valid message object. To scan the message RAM for a matching message object, the controller uses the acceptance filtering programmed through the mask bits in the **CANIFnMSKn** register and enabled using the UMASK bit in the **CANIFnMCTL** register. Each valid message object, starting with object 1, is compared with the incoming message to locate a matching message object in the message RAM. If a match occurs, the scanning is stopped and the message handler proceeds depending on whether it is a data frame or remote frame that was received.

17.3.7 Receiving a Data Frame

The message handler stores the message from the CAN controller receive shift register into the matching message object in the message RAM. The data bytes, all arbitration bits, and the DLC bits are all stored into the corresponding message object. In this manner, the data bytes are connected with the identifier even if arbitration masks are used. The NEWDAT bit of the **CANIFnMCTL** register is set to indicate that new data has been received. The CPU should clear this bit when it reads the message object to indicate to the controller that the message has been received, and the buffer is free to receive more messages. If the CAN controller receives a message and the NEWDAT bit is already set, the MSGLST bit in the **CANIFnMCTL** register is set to indicate that the previous data was lost. If the system requires an interrupt on successful reception of a frame, the RXIE bit of the **CANIFnMCTL** register should be set. In this case, the INTPND bit of the same register is set, causing the **CANINT** register to point to the message object that just received a message. The TXRQST bit of this message object should be cleared to prevent the transmission of a remote frame.

17.3.8 Receiving a Remote Frame

A remote frame contains no data, but instead specifies which object should be transmitted. When a remote frame is received, three different configurations of the matching message object have to be considered:

Table 17-3. Message Object Configurations

Configuration in CANIFnMCTL	Description
■ DIR = 1 (direction = transmit); programmed in the **CANIFnARB2** register ■ RMTEN = 1 (set the TXRQST bit of the **CANIFnMCTL** register at reception of the frame to enable transmission) ■ UMASK = 1 or 0	At the reception of a matching remote frame, the TXRQST bit of this message object is set. The rest of the message object remains unchanged, and the controller automatically transfers the data in the message object as soon as possible.
■ DIR = 1 (direction = transmit); programmed in the **CANIFnARB2** register ■ RMTEN = 0 (do not change the TXRQST bit of the **CANIFnMCTL** register at reception of the frame) ■ UMASK = 0 (ignore mask in the **CANIFnMSKn** register)	At the reception of a matching remote frame, the TXRQST bit of this message object remains unchanged, and the remote frame is ignored. This remote frame is disabled, the data is not transferred and nothing indicates that the remote frame ever happened.
■ DIR = 1 (direction = transmit); programmed in the **CANIFnARB2** register ■ RMTEN = 0 (do not change the TXRQST bit of the **CANIFnMCTL** register at reception of the frame) ■ UMASK = 1 (use mask (MSK, MXTD, and MDIR in the **CANIFnMSKn** register) for acceptance filtering)	At the reception of a matching remote frame, the TXRQST bit of this message object is cleared. The arbitration and control field (ID + XTD + RMTEN + DLC) from the shift register is stored into the message object in the message RAM, and the NEWDAT bit of this message object is set. The data field of the message object remains unchanged; the remote frame is treated similar to a received data frame. This mode is useful for a remote data request from another CAN device for which the Stellaris controller does not have readily available data. The software must fill the data and answer the frame manually.

17.3.9 Receive/Transmit Priority

The receive/transmit priority for the message objects is controlled by the message number. Message object 1 has the highest priority, while message object 32 has the lowest priority. If more than one transmission request is pending, the message objects are transmitted in order based on the message object with the lowest message number. This prioritization is separate from that of the message identifier which is enforced by the CAN bus. As a result, if message object 1 and message object 2 both have valid messages to be transmitted, message object 1 is always transmitted first regardless of the message identifier in the message object itself.

17.3.10 Configuring a Receive Message Object

The following steps illustrate how to configure a receive message object.

1. Program the **CAN IFn Command Mask (CANIFnCMASK)** register as described in the "Configuring a Transmit Message Object" on page 864 section, except that the WRNRD bit is set to specify a write to the message RAM.

2. Program the **CANIFnMSK1**and **CANIFnMSK2** registers as described in the "Configuring a Transmit Message Object" on page 864 section to configure which bits are used for acceptance filtering. Note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the **CANIFnMCTL** register.

3. In the **CANIFnMSK2** register, use the MSK[12:0] bits to specify which of the bits in the 29-bit or 11-bit message identifier are used for acceptance filtering. Note that MSK[12:0] are used for bits [28:16] of the 29-bit message identifier; whereas MSK[12:2] are used for bits [10:0] of the 11-bit message identifier. Use the MXTD and MDIR bits to specify whether to use XTD and

DIR for acceptance filtering. A value of 0x00 enables all messages to pass through the acceptance filtering. Also note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the **CANIFnMCTL** register.

4. Program the **CANIFnARB1** and **CANIFnARB2** registers as described in the "Configuring a Transmit Message Object" on page 864 section to program XTD and ID bits for the message identifier to be received; set the MSGVAL bit to indicate a valid message; and clear the DIR bit to specify receive.

5. In the **CANIFnMCTL** register:

   - Optionally set the UMASK bit to enable the mask (MSK, MXTD, and MDIR specified in the **CANIFnMSK1** and **CANIFnMSK2** registers) for acceptance filtering
   - Optionally set the RXIE bit to enable the INTPND bit to be set after a successful reception
   - Clear the RMTEN bit to leave the TXRQST bit unchanged
   - Set the EOB bit for a single message object
   - Configure the DLC[3:0] field to specify the size of the data frame

   Take care during this configuration not to set the NEWDAT, MSGLST, INTPND or TXRQST bits.

6. Program the number of the message object to be received in the MNUM field in the **CAN IFn Command Request (CANIFnCRQ)** register. Reception of the message object begins as soon as a matching frame is available on the CAN bus.

When the message handler stores a data frame in the message object, it stores the received Data Length Code and eight data bytes in the **CANIFnDA1**, **CANIFnDA2**, **CANIFnDB1**, and **CANIFnDB2** register. Byte 0 of the CAN data frame is stored in DATA[7:0] in the **CANIFnDA1** register. If the Data Length Code is less than 8, the remaining bytes of the message object are overwritten by unspecified values.

The CAN mask registers can be used to allow groups of data frames to be received by a message object. The CAN mask registers, **CANIFnMSKn**, configure which groups of frames are received by a message object. The UMASK bit in the **CANIFnMCTL** register enables the MSK bits in the **CANIFnMSKn** register to filter which frames are received. The MXTD bit in the **CANIFnMSK2** register should be set if only 29-bit extended identifiers are expected by this message object.

17.3.11 Handling of Received Message Objects

The CPU may read a received message any time via the CAN Interface registers because the data consistency is guaranteed by the message handler state machine.

Typically, the CPU first writes 0x007F to the **CANIFnCMSK** register and then writes the number of the message object to the **CANIFnCRQ** register. That combination transfers the whole received message from the message RAM into the Message Buffer registers (**CANIFnMSKn**, **CANIFnARBn**, and **CANIFnMCTL**). Additionally, the NEWDAT and INTPND bits are cleared in the message RAM, acknowledging that the message has been read and clearing the pending interrupt generated by this message object.

If the message object uses masks for acceptance filtering, the **CANIFnARBn** registers show the full, unmasked ID for the received message.

The NEWDAT bit in the **CANIFnMCTL** register shows whether a new message has been received since the last time this message object was read. The MSGLST bit in the **CANIFnMCTL** register shows whether more than one message has been received since the last time this message object was read. MSGLST is not automatically cleared, and should be cleared by software after reading its status.

Using a remote frame, the CPU may request new data from another CAN node on the CAN bus. Setting the TXRQST bit of a receive object causes the transmission of a remote frame with the receive object's identifier. This remote frame triggers the other CAN node to start the transmission of the matching data frame. If the matching data frame is received before the remote frame could be transmitted, the TXRQST bit is automatically reset. This prevents the possible loss of data when the other device on the CAN bus has already transmitted the data slightly earlier than expected.

17.3.11.1 Configuration of a FIFO Buffer

With the exception of the EOB bit in the **CANIFnMCTL** register, the configuration of receive message objects belonging to a FIFO buffer is the same as the configuration of a single receive message object (see “Configuring a Receive Message Object” on page 867). To concatenate two or more message objects into a FIFO buffer, the identifiers and masks (if used) of these message objects have to be programmed to matching values. Due to the implicit priority of the message objects, the message object with the lowest message object number is the first message object in a FIFO buffer. The EOB bit of all message objects of a FIFO buffer except the last one must be cleared. The EOB bit of the last message object of a FIFO buffer is set, indicating it is the last entry in the buffer.

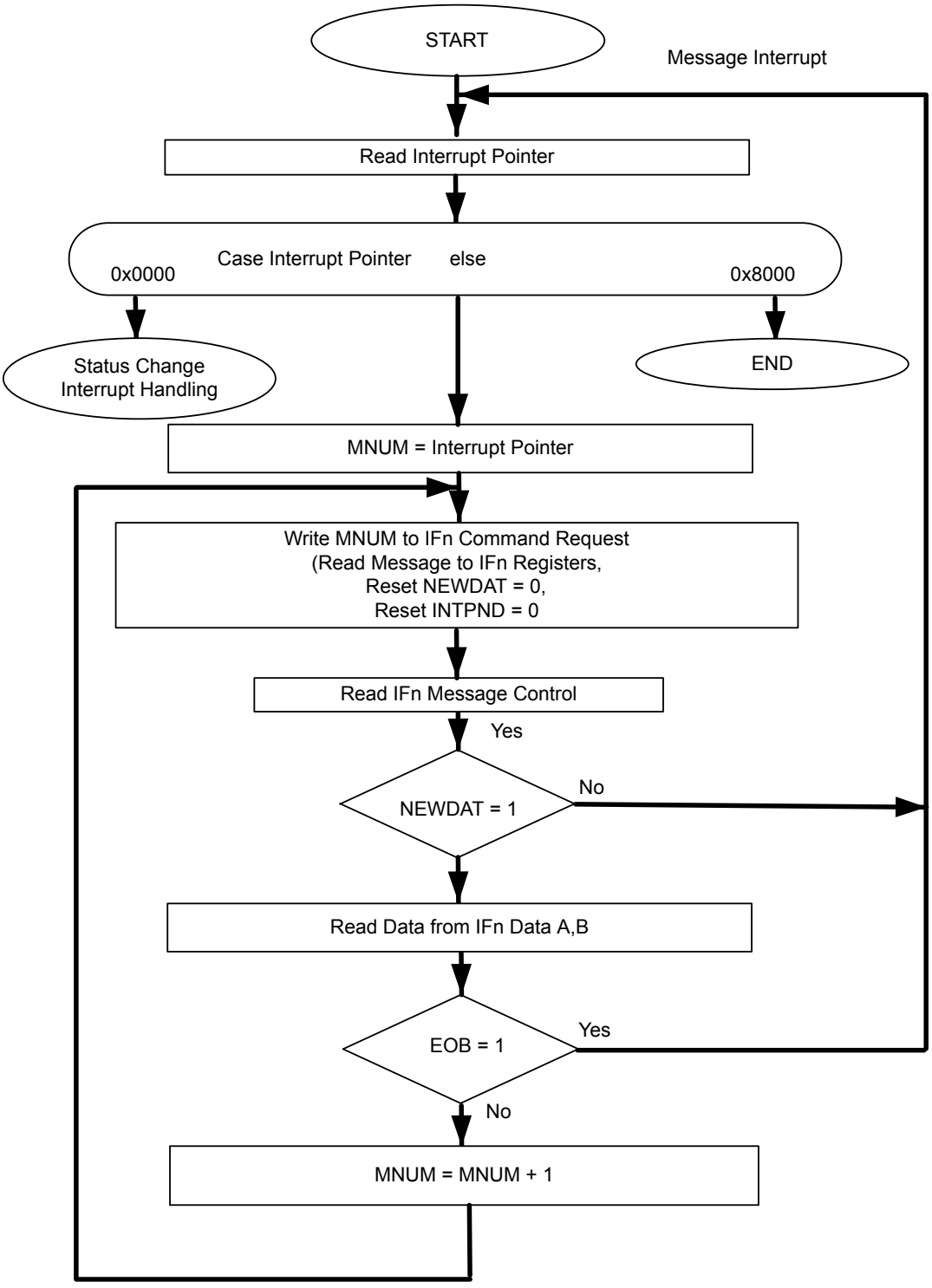
17.3.11.2 Reception of Messages with FIFO Buffers

Received messages with identifiers matching to a FIFO buffer are stored starting with the message object with the lowest message number. When a message is stored into a message object of a FIFO buffer, the NEWDAT of the **CANIFnMCTL** register bit of this message object is set. By setting NEWDAT while EOB is clear, the message object is locked and cannot be written to by the message handler until the CPU has cleared the NEWDAT bit. Messages are stored into a FIFO buffer until the last message object of this FIFO buffer is reached. Until all of the preceding message objects have been released by clearing the NEWDAT bit, all further messages for this FIFO buffer are written into the last message object of the FIFO buffer and therefore overwrite previous messages.

17.3.11.3 Reading from a FIFO Buffer

When the CPU transfers the contents of a message object from a FIFO buffer by writing its number to the **CANIFnCRQ** register, the TXRQST and CLRINTPND bits in the **CANIFnCMSK** register should be set such that the NEWDAT and INTPEND bits in the **CANIFnMCTL** register are cleared after the read. The values of these bits in the **CANIFnMCTL** register always reflect the status of the message object before the bits are cleared. To assure the correct function of a FIFO buffer, the CPU should read out the message objects starting with the message object with the lowest message number. When reading from the FIFO buffer, the user should be aware that a new received message is placed in the message object with the lowest message number for which the NEWDAT bit of the **CANIFnMCTL** register is clear. As a result, the order of the received messages in the FIFO is not guaranteed. Figure 17-3 on page 870 shows how a set of message objects which are concatenated to a FIFO Buffer can be handled by the CPU.

Figure 17-3. Message Objects in a FIFO Buffer



17.3.12 Handling of Interrupts

If several interrupts are pending, the **CAN Interrupt (CANINT)** register points to the pending interrupt with the highest priority, disregarding their chronological order. The status interrupt has the highest

priority. Among the message interrupts, the message object's interrupt with the lowest message number has the highest priority. A message interrupt is cleared by clearing the message object's INTPND bit in the **CANIFnMCTL** register or by reading the **CAN Status (CANSTS)** register. The status Interrupt is cleared by reading the **CANSTS** register.

The interrupt identifier INTID in the **CANINT** register indicates the cause of the interrupt. When no interrupt is pending, the register reads as 0x0000. If the value of the INTID field is different from 0, then an interrupt is pending. If the IE bit is set in the **CANCTL** register, the interrupt line to the interrupt controller is active. The interrupt line remains active until the INTID field is 0, meaning that all interrupt sources have been cleared (the cause of the interrupt is reset), or until IE is cleared, which disables interrupts from the CAN controller.

The INTID field of the **CANINT** register points to the pending message interrupt with the highest interrupt priority. The SIE bit in the **CANCTL** register controls whether a change of the RXOK, TXOK, and LEC bits in the **CANSTS** register can cause an interrupt. The EIE bit in the **CANCTL**register controls whether a change of the BOFF and EWARN bits in the **CANSTS** register can cause an interrupt. The IE bit in the **CANCTL** register controls whether any interrupt from the CAN controller actually generates an interrupt to the interrupt controller. The **CANINT** register is updated even when the IE bit in the **CANCTL** register is clear, but the interrupt is not indicated to the CPU.

A value of 0x8000 in the **CANINT** register indicates that an interrupt is pending because the CAN module has updated, but not necessarily changed, the **CANSTS** register, indicating that either an error or status interrupt has been generated. A write access to the **CANSTS** register can clear the RXOK, TXOK, and LEC bits in that same register; however, the only way to clear the source of a status interrupt is to read the **CANSTS** register.

The source of an interrupt can be determined in two ways during interrupt handling. The first is to read the INTID bit in the **CANINT** register to determine the highest priority interrupt that is pending, and the second is to read the **CAN Message Interrupt Pending (CANMSGnINT)** register to see all of the message objects that have pending interrupts.

An interrupt service routine reading the message that is the source of the interrupt may read the message and clear the message object's INTPND bit at the same time by setting the CLRINTPND bit in the **CANIFnCMSK** register. Once the INTPND bit has been cleared, the **CANINT** register contains the message number for the next message object with a pending interrupt.

17.3.13 Test Mode

A Test Mode is provided which allows various diagnostics to be performed. Test Mode is entered by setting the TEST bit in the **CANCTL** register. Once in Test Mode, the TX[1:0], LBACK, SILENT and BASIC bits in the **CAN Test (CANTST)** register can be used to put the CAN controller into the various diagnostic modes. The RX bit in the **CANTST** register allows monitoring of the CANnRX signal. All **CANTST** register functions are disabled when the TEST bit is cleared.

17.3.13.1 Silent Mode

Silent Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits (Acknowledge Bits, Error Frames). The CAN Controller is put in Silent Mode setting the SILENT bit in the **CANTST** register. In Silent Mode, the CAN controller is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and cannot start a transmission. If the CAN Controller is required to send a dominant bit (ACK bit, overload flag, or active error flag), the bit is rerouted internally so that the CAN Controller monitors this dominant bit, although the CAN bus remains in recessive state.

17.3.13.2 Loopback Mode

Loopback mode is useful for self-test functions. In Loopback Mode, the CAN Controller internally routes the CANnTX signal on to the CANnRX signal and treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into the message buffer. The CAN Controller is put in Loopback Mode by setting the LBACK bit in the **CANTST** register. To be independent from external stimulation, the CAN Controller ignores acknowledge errors (a recessive bit sampled in the acknowledge slot of a data/remote frame) in Loopback Mode. The actual value of the CANnRX signal is disregarded by the CAN Controller. The transmitted messages can be monitored on the CANnTX signal.

17.3.13.3 Loopback Combined with Silent Mode

Loopback Mode and Silent Mode can be combined to allow the CAN Controller to be tested without affecting a running CAN system connected to the CANnTX and CANnRX signals. In this mode, the CANnRX signal is disconnected from the CAN Controller and the CANnTX signal is held recessive. This mode is enabled by setting both the LBACK and SILENT bits in the **CANTST** register.

17.3.13.4 Basic Mode

Basic Mode allows the CAN Controller to be operated without the Message RAM. In Basic Mode, The CANIF1 registers are used as the transmit buffer. The transmission of the contents of the IF1 registers is requested by setting the BUSY bit of the **CANIF1CRQ** register. The CANIF1 registers are locked while the BUSY bit is set. The BUSY bit indicates that a transmission is pending. As soon the CAN bus is idle, the CANIF1 registers are loaded into the shift register of the CAN Controller and transmission is started. When the transmission has completed, the BUSY bit is cleared and the locked CANIF1 registers are released. A pending transmission can be aborted at any time by clearing the BUSY bit in the **CANIF1CRQ** register while the CANIF1 registers are locked. If the CPU has cleared the BUSY bit, a possible retransmission in case of lost arbitration or an error is disabled.

The CANIF2 Registers are used as a receive buffer. After the reception of a message, the contents of the shift register are stored in the CANIF2 registers, without any acceptance filtering. Additionally, the actual contents of the shift register can be monitored during the message transfer. Each time a read message object is initiated by setting the BUSY bit of the **CANIF2CRQ** register, the contents of the shift register are stored into the CANIF2 registers.

In Basic Mode, all message-object-related control and status bits and of the control bits of the **CANIFnCMSK** registers are not evaluated. The message number of the **CANIFnCRQ** registers is also not evaluated. In the **CANIF2MCTL** register, the NEWDAT and MSGLST bits retain their function, the DLC[3:0] field shows the received DLC, the other control bits are cleared.

Basic Mode is enabled by setting the BASIC bit in the **CANTST** register.

17.3.13.5 Transmit Control

Software can directly override control of the CANnTX signal in four different ways.

- CANnTX is controlled by the CAN Controller
- The sample point is driven on the CANnTX signal to monitor the bit timing
- CANnTX drives a low value
- CANnTX drives a high value

The last two functions, combined with the readable CAN receive pin CANnRX, can be used to check the physical layer of the CAN bus.

The Transmit Control function is enabled by programming the TX[1:0] field in the **CANTST** register. The three test functions for the CANnTX signal interfere with all CAN protocol functions. TX[1:0] must be cleared when CAN message transfer or Loopback Mode, Silent Mode, or Basic Mode are selected.

17.3.14 Bit Timing Configuration Error Considerations

Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly. In many cases, the CAN bit synchronization amends a faulty configuration of the CAN bit timing to such a degree that only occasionally an error frame is generated. In the case of arbitration, however, when two or more CAN nodes simultaneously try to transmit a frame, a misplaced sample point may cause one of the transmitters to become error passive. The analysis of such sporadic errors requires a detailed knowledge of the CAN bit synchronization inside a CAN node and of the CAN nodes' interaction on the CAN bus.

17.3.15 Bit Time and Bit Rate

The CAN system supports bit rates in the range of lower than 1 Kbps up to 1000 Kbps. Each member of the CAN network has its own clock generator. The timing parameter of the bit time can be configured individually for each CAN node, creating a common bit rate even though the CAN nodes' oscillator periods may be different.

Because of small variations in frequency caused by changes in temperature or voltage and by deteriorating components, these oscillators are not absolutely stable. As long as the variations remain inside a specific oscillator's tolerance range, the CAN nodes are able to compensate for the different bit rates by periodically resynchronizing to the bit stream.

According to the CAN specification, the bit time is divided into four segments (see Figure 17-4 on page 874): the Synchronization Segment, the Propagation Time Segment, the Phase Buffer Segment 1, and the Phase Buffer Segment 2. Each segment consists of a specific, programmable number of time quanta (see Table 17-4 on page 874). The length of the time quantum (t_q), which is the basic time unit of the bit time, is defined by the CAN controller's input clock (fsys) and the Baud Rate Prescaler (BRP):

t_q = BRP / fsys

The fsys input clock is the system clock frequency as configured by the **RCC** or **RCC2** registers (see page 215 or page 222).

The Synchronization Segment Sync is that part of the bit time where edges of the CAN bus level are expected to occur; the distance between an edge that occurs outside of Sync and the Sync is called the phase error of that edge.

The Propagation Time Segment Prop is intended to compensate for the physical delay times within the CAN network.

The Phase Buffer Segments Phase1 and Phase2 surround the Sample Point.

The (Re-)Synchronization Jump Width (SJW) defines how far a resynchronization may move the Sample Point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.

A given bit rate may be met by different bit-time configurations, but for the proper function of the CAN network, the physical delay times and the oscillator's tolerance range have to be considered.

Figure 17-4. CAN Bit Time

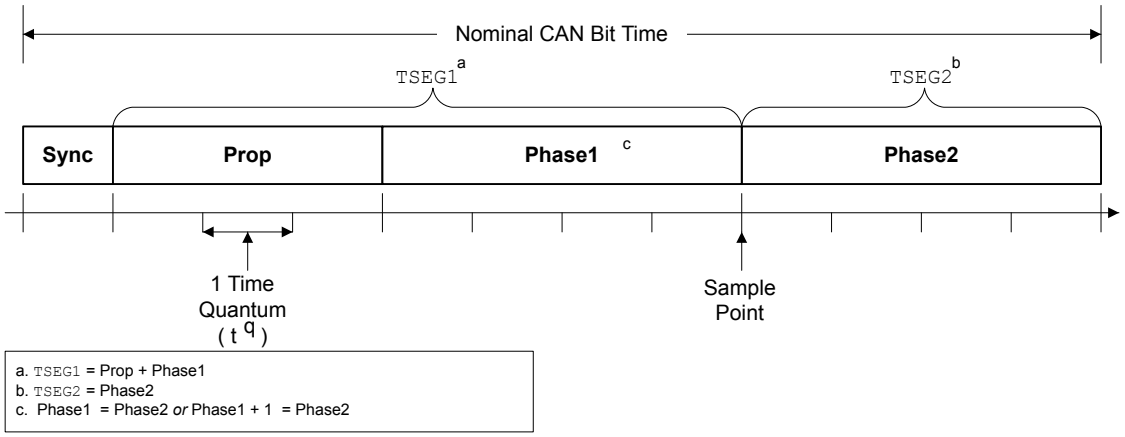


Table 17-4. CAN Protocol Ranges[a]

Parameter	Range	Remark
BRP	[1 .. 64]	Defines the length of the time quantum t_q. The **CANBRPE** register can be used to extend the range to 1024.
Sync	1 t_q	Fixed length, synchronization of bus input to system clock
Prop	[1 .. 8] t_q	Compensates for the physical delay times
Phase1	[1 .. 8] t_q	May be lengthened temporarily by synchronization
Phase2	[1 .. 8] t_q	May be shortened temporarily by synchronization
SJW	[1 .. 4] t_q	May not be longer than either Phase Buffer Segment

a. This table describes the minimum programmable ranges required by the CAN protocol.

The bit timing configuration is programmed in two register bytes in the **CANBIT** register. In the **CANBIT** register, the four components TSEG2, TSEG1, SJW, and BRP have to be programmed to a numerical value that is one less than its functional value; so instead of values in the range of [1..n], values in the range of [0..n-1] are programmed. That way, for example, SJW (functional range of [1..4]) is represented by only two bits in the SJW bit field. Table 17-5 shows the relationship between the **CANBIT** register values and the parameters.

Table 17-5. CANBIT Register Values

CANBIT Register Field	Setting
TSEG2	Phase2 - 1
TSEG1	Prop + Phase1 - 1
SJW	SJW - 1
BRP	BRP

Therefore, the length of the bit time is (programmed values):

$[TSEG1 + TSEG2 + 3] \times t_q$

or (functional values):

$[Sync + Prop + Phase1 + Phase2] \times t_q$

The data in the **CANBIT** register is the configuration input of the CAN protocol controller. The baud rate prescaler (configured by the BRP field) defines the length of the time quantum, the basic time

unit of the bit time; the bit timing logic (configured by TSEG1, TSEG2, and SJW) defines the number of time quanta in the bit time.

The processing of the bit time, the calculation of the position of the sample point, and occasional synchronizations are controlled by the CAN controller and are evaluated once per time quantum.

The CAN controller translates messages to and from frames. In addition, the controller generates and discards the enclosing fixed format bits, inserts and extracts stuff bits, calculates and checks the CRC code, performs the error management, and decides which type of synchronization is to be used. The bit value is received or transmitted at the sample point. The information processing time (IPT) is the time after the sample point needed to calculate the next bit to be transmitted on the CAN bus. The IPT includes any of the following: retrieving the next data bit, handling a CRC bit, determining if bit stuffing is required, generating an error flag or simply going idle.

The IPT is application-specific but may not be longer than 2 t_q; the CAN's IPT is 0 t_q. Its length is the lower limit of the programmed length of Phase2. In case of synchronization, Phase2 may be shortened to a value less than IPT, which does not affect bus timing.

17.3.16 Calculating the Bit Timing Parameters

Usually, the calculation of the bit timing configuration starts with a required bit rate or bit time. The resulting bit time (1/bit rate) must be an integer multiple of the system clock period.

The bit time may consist of 4 to 25 time quanta. Several combinations may lead to the required bit time, allowing iterations of the following steps.

The first part of the bit time to be defined is Prop. Its length depends on the delay times measured in the system. A maximum bus length as well as a maximum node delay has to be defined for expandable CAN bus systems. The resulting time for Prop is converted into time quanta (rounded up to the nearest integer multiple of t_q).

Sync is 1 t_q long (fixed), which leaves (bit time - Prop - 1) t_q for the two Phase Buffer Segments. If the number of remaining t_q is even, the Phase Buffer Segments have the same length, that is, Phase2 = Phase1, else Phase2 = Phase1 + 1.

The minimum nominal length of Phase2 has to be regarded as well. Phase2 may not be shorter than the CAN controller's Information Processing Time, which is, depending on the actual implementation, in the range of [0..2] t_q.

The length of the synchronization jump width is set to the least of 4, Phase1 or Phase2.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formula given below:

$$(1 - df) \times fnom \leq fosc \leq (1 + df) \times fnom$$

where:

- df = Maximum tolerance of oscillator frequency
- fosc = Actual oscillator frequency
- fnom = Nominal oscillator frequency

Maximum frequency tolerance must take into account the following formulas:

$$df \leq \frac{(Phase\_seg1, Phase\_seg2)\ \min}{2 \times (13 \times tbit - Phase\_Seg2)}$$

$$df\ \max = 2 \times df \times fnom$$

where:

- Phase1 and Phase2 are from Table 17-4 on page 874
- tbit = Bit Time
- dfmax = Maximum difference between two oscillators

If more than one configuration is possible, that configuration allowing the highest oscillator tolerance range should be chosen.

CAN nodes with different system clocks require different configurations to come to the same bit rate. The calculation of the propagation time in the CAN network, based on the nodes with the longest delay times, is done once for the whole network.

The CAN system's oscillator tolerance range is limited by the node with the lowest tolerance range.

The calculation may show that bus length or bit rate have to be decreased or that the oscillator frequencies' stability has to be increased in order to find a protocol-compliant configuration of the CAN bit timing.

17.3.16.1 Example for Bit Timing at High Baud Rate

In this example, the frequency of CAN clock is 25 MHz, and the bit rate is 1 Mbps.

```
bit time = 1 µs = n * tq = 5 * tq
tq = 200 ns
tq = (Baud rate Prescaler)/CAN Clock
Baud rate Prescaler = tq * CAN Clock
Baud rate Prescaler = 200E-9 * 25E6 = 5

tSync = 1 * tq = 200 ns                \\fixed at 1 time quanta

delay of bus driver 50 ns
delay of receiver circuit 30 ns
delay of bus line (40m) 220 ns
tProp 400 ns = 2 * tq                  \\400 is next integer multiple of tq

bit time = tSync + tTSeg1 + tTSeg2 = 5 * tq
bit time = tSync + tProp + tPhase 1 + tPhase2
tPhase 1 + tPhase2 = bit time - tSync - tProp
tPhase 1 + tPhase2 = (5 * tq) - (1 * tq) - (2 * tq)
tPhase 1 + tPhase2 = 2 * tq
tPhase1 = 1 * tq
tPhase2 = 1 * tq                       \\tPhase2 = tPhase1
```

```
tTSeg1 = tProp + tPhase1
tTSeg1 = (2 * tq) + (1 * tq)
tTSeg1 = 3 * tq

tTSeg2 = tPhase2
tTSeg2 = (Information Processing Time + 1) * tq
tTSeg2 = 1 * tq                          \\Assumes IPT=0

tSJW = 1 * tq                           \\Least of 4, Phase1 and Phase2
```

In the above example, the bit field values for the **CANBIT** register are:

TSEG2	= TSeg2 -1 = 1-1 = 0
TSEG1	= TSeg1 -1 = 3-1 = 2
SJW	= SJW -1 = 1-1 = 0
BRP	= Baud rate prescaler - 1 = 5-1 =4

The final value programmed into the **CANBIT** register = 0x0204.

17.3.16.2 Example for Bit Timing at Low Baud Rate

In this example, the frequency of the CAN clock is 50 MHz, and the bit rate is 100 Kbps.

```
bit time = 10 µs = n * tq = 10 * tq
tq = 1 µs
tq = (Baud rate Prescaler)/CAN Clock
Baud rate Prescaler = tq * CAN Clock
Baud rate Prescaler = 1E-6 * 50E6 = 50

tSync = 1 * tq = 1 µs                \\fixed at 1 time quanta

delay of bus driver 200 ns
delay of receiver circuit 80 ns
delay of bus line (40m) 220 ns
tProp 1 µs = 1 * tq                  \\1 µs is next integer multiple of tq

bit time = tSync + tTSeg1 + tTSeg2 = 10 * tq
bit time = tSync + tProp + tPhase 1 + tPhase2
tPhase 1 + tPhase2 = bit time - tSync - tProp
tPhase 1 + tPhase2 = (10 * tq) - (1 * tq)  - (1 * tq)
tPhase 1 + tPhase2 = 8 * tq
tPhase1 = 4 * tq
tPhase2 = 4 * tq                     \\tPhase1 = tPhase2
```

```
tTSeg1 = tProp + tPhase1
tTSeg1 = (1 * tq) + (4 * tq)
tTSeg1 = 5 * tq
tTSeg2 = tPhase2
tTSeg2 = (Information Processing Time + 4) × tq
tTSeg2 = 4 * tq                          \\Assumes IPT=0

tSJW = 4 * tq                            \\Least of 4, Phase1, and Phase2
```

TSEG2	= TSeg2 -1 = 4-1 = 3
TSEG1	= TSeg1 -1 = 5-1 = 4
SJW	= SJW -1 = 4-1 = 3
BRP	= Baud rate prescaler - 1 = 50-1 =49

The final value programmed into the **CANBIT** register = 0x34F1.

17.4 Register Map

Table 17-6 on page 878 lists the registers. All addresses given are relative to the CAN base address of:

- CAN0: 0x4004.0000
- CAN1: 0x4004.1000
- CAN2: 0x4004.2000

Note that the CAN controller clock must be enabled before the registers can be programmed (see page 255). There must be a delay of 3 system clocks after the CAN module clock is enabled before any CAN module registers are accessed.

Table 17-6. CAN Register Map

Offset	Name	Type	Reset	Description	See page
0x000	CANCTL	R/W	0x0000.0001	CAN Control	881
0x004	CANSTS	R/W	0x0000.0000	CAN Status	883
0x008	CANERR	RO	0x0000.0000	CAN Error Counter	886
0x00C	CANBIT	R/W	0x0000.2301	CAN Bit Timing	887
0x010	CANINT	RO	0x0000.0000	CAN Interrupt	888
0x014	CANTST	R/W	0x0000.0000	CAN Test	889
0x018	CANBRPE	R/W	0x0000.0000	CAN Baud Rate Prescaler Extension	891

Table 17-6. CAN Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x020	CANIF1CRQ	R/W	0x0000.0001	CAN IF1 Command Request	892
0x024	CANIF1CMSK	R/W	0x0000.0000	CAN IF1 Command Mask	893
0x028	CANIF1MSK1	R/W	0x0000.FFFF	CAN IF1 Mask 1	896
0x02C	CANIF1MSK2	R/W	0x0000.FFFF	CAN IF1 Mask 2	897
0x030	CANIF1ARB1	R/W	0x0000.0000	CAN IF1 Arbitration 1	899
0x034	CANIF1ARB2	R/W	0x0000.0000	CAN IF1 Arbitration 2	900
0x038	CANIF1MCTL	R/W	0x0000.0000	CAN IF1 Message Control	902
0x03C	CANIF1DA1	R/W	0x0000.0000	CAN IF1 Data A1	905
0x040	CANIF1DA2	R/W	0x0000.0000	CAN IF1 Data A2	905
0x044	CANIF1DB1	R/W	0x0000.0000	CAN IF1 Data B1	905
0x048	CANIF1DB2	R/W	0x0000.0000	CAN IF1 Data B2	905
0x080	CANIF2CRQ	R/W	0x0000.0001	CAN IF2 Command Request	892
0x084	CANIF2CMSK	R/W	0x0000.0000	CAN IF2 Command Mask	893
0x088	CANIF2MSK1	R/W	0x0000.FFFF	CAN IF2 Mask 1	896
0x08C	CANIF2MSK2	R/W	0x0000.FFFF	CAN IF2 Mask 2	897
0x090	CANIF2ARB1	R/W	0x0000.0000	CAN IF2 Arbitration 1	899
0x094	CANIF2ARB2	R/W	0x0000.0000	CAN IF2 Arbitration 2	900
0x098	CANIF2MCTL	R/W	0x0000.0000	CAN IF2 Message Control	902
0x09C	CANIF2DA1	R/W	0x0000.0000	CAN IF2 Data A1	905
0x0A0	CANIF2DA2	R/W	0x0000.0000	CAN IF2 Data A2	905
0x0A4	CANIF2DB1	R/W	0x0000.0000	CAN IF2 Data B1	905
0x0A8	CANIF2DB2	R/W	0x0000.0000	CAN IF2 Data B2	905
0x100	CANTXRQ1	RO	0x0000.0000	CAN Transmission Request 1	906
0x104	CANTXRQ2	RO	0x0000.0000	CAN Transmission Request 2	906
0x120	CANNWDA1	RO	0x0000.0000	CAN New Data 1	907
0x124	CANNWDA2	RO	0x0000.0000	CAN New Data 2	907
0x140	CANMSG1INT	RO	0x0000.0000	CAN Message 1 Interrupt Pending	908
0x144	CANMSG2INT	RO	0x0000.0000	CAN Message 2 Interrupt Pending	908
0x160	CANMSG1VAL	RO	0x0000.0000	CAN Message 1 Valid	909
0x164	CANMSG2VAL	RO	0x0000.0000	CAN Message 2 Valid	909

17.5 CAN Register Descriptions

The remainder of this section lists and describes the CAN registers, in numerical order by address offset. There are two sets of Interface Registers that are used to access the Message Objects in the Message RAM: **CANIF1x** and **CANIF2x**. The function of the two sets are identical and are used to queue transactions.

Register 1: CAN Control (CANCTL), offset 0x000

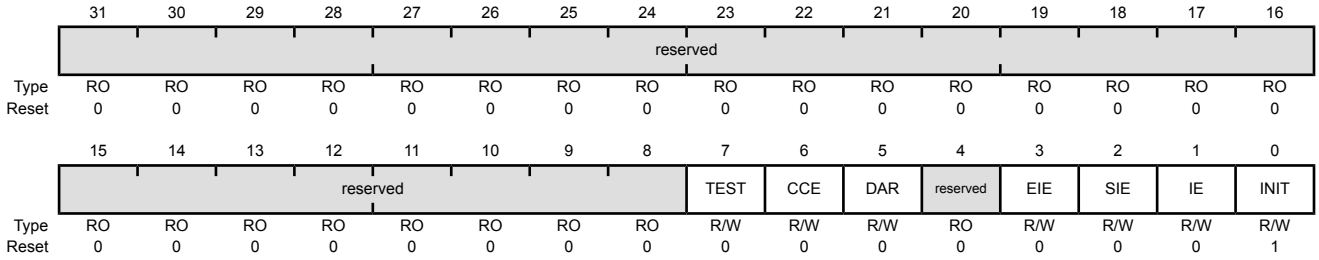
This control register initializes the module and enables test mode and interrupts.

The bus-off recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or clearing INIT. If the device goes bus-off, it sets INIT, stopping all bus activities. Once INIT has been cleared by the CPU, the device then waits for 129 occurrences of Bus Idle (129 * 11 consecutive High bits) before resuming normal operations. At the end of the bus-off recovery sequence, the Error Management Counters are reset.

During the waiting time after INIT is cleared, each time a sequence of 11 High bits has been monitored, a BITERROR0 code is written to the **CANSTS** register (the LEC field = 0x5), enabling the CPU to readily check whether the CAN bus is stuck Low or continuously disturbed, and to monitor the proceeding of the bus-off recovery sequence.

CAN Control (CANCTL)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
CAN2 base: 0x4004.2000
Offset 0x000
Type R/W, reset 0x0000.0001



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TEST	R/W	0	Test Mode Enable Value: 0 — The CAN controller is operating normally. Value: 1 — The CAN controller is in test mode.
6	CCE	R/W	0	Configuration Change Enable Value: 0 — Write accesses to the **CANBIT** register are not allowed. Value: 1 — Write accesses to the **CANBIT** register are allowed if the INIT bit is 1.
5	DAR	R/W	0	Disable Automatic-Retransmission Value: 0 — Auto-retransmission of disturbed messages is enabled. Value: 1 — Auto-retransmission is disabled.

Bit/Field	Name	Type	Reset	Description
4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EIE	R/W	0	Error Interrupt Enable Value: 0 — No error status interrupt is generated. Value: 1 — A change in the BOFF or EWARN bits in the **CANSTS** register generates an interrupt.
2	SIE	R/W	0	Status Interrupt Enable Value: 0 — No status interrupt is generated. Value: 1 — An interrupt is generated when a message has successfully been transmitted or received, or a CAN bus error has been detected. A change in the TXOK, RXOK or LEC bits in the **CANSTS** register generates an interrupt.
1	IE	R/W	0	CAN Interrupt Enable Value: 0 — Interrupts disabled. Value: 1 — Interrupts enabled.
0	INIT	R/W	1	Initialization Value: 0 — Normal operation. Value: 1 — Initialization started.

Register 2: CAN Status (CANSTS), offset 0x004

Important: This register is read-sensitive. See the register description for details.

The status register contains information for interrupt servicing such as Bus-Off, error count threshold, and error types.

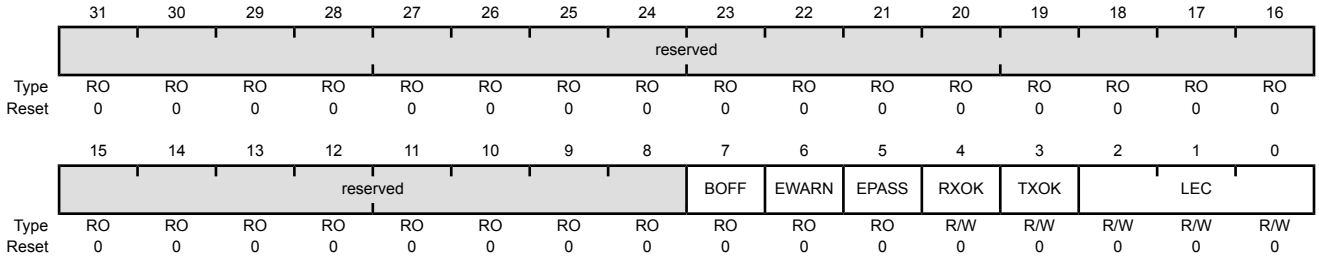
The LEC field holds the code that indicates the type of the last error to occur on the CAN bus. This field is cleared when a message has been transferred (reception or transmission) without error. The unused error code 0x7 may be written by the CPU to manually set this field to an invalid error so that it can be checked for a change later.

An error interrupt is generated by the BOFF and EWARN bits, and a status interrupt is generated by the RXOK, TXOK, and LEC bits, if the corresponding enable bits in the **CAN Control (CANCTL)** register are set. A change of the EPASS bit or a write to the RXOK, TXOK, or LEC bits does not generate an interrupt.

Reading the **CAN Status (CANSTS)** register clears the **CAN Interrupt (CANINT)** register, if it is pending.

CAN Status (CANSTS)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
CAN2 base: 0x4004.2000
Offset 0x004
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	BOFF	RO	0	Bus-Off Status Value: 0 — The CAN controller is not in bus-off state. Value: 1 — The CAN controller is in bus-off state.
6	EWARN	RO	0	Warning Status Value: 0 — Both error counters are below the error warning limit of 96. Value: 1 — At least one of the error counters has reached the error warning limit of 96.

Bit/Field	Name	Type	Reset	Description
5	EPASS	RO	0	Error Passive Value: Description 0: The CAN module is in the Error Active state, that is, the receive or transmit error count is less than or equal to 127. 1: The CAN module is in the Error Passive state, that is, the receive or transmit error count is greater than 127.
4	RXOK	R/W	0	Received a Message Successfully Value: Description 0: Since this bit was last cleared, no message has been successfully received. 1: Since this bit was last cleared, a message has been successfully received, independent of the result of the acceptance filtering. This bit must be cleared by writing a 0 to it.
3	TXOK	R/W	0	Transmitted a Message Successfully Value: Description 0: Since this bit was last cleared, no message has been successfully transmitted. 1: Since this bit was last cleared, a message has been successfully transmitted error-free and acknowledged by at least one other node. This bit must be cleared by writing a 0 to it.

Bit/Field	Name	Type	Reset	Description
2:0	LEC	R/W	0x0	Last Error Code This is the type of the last error to occur on the CAN bus.

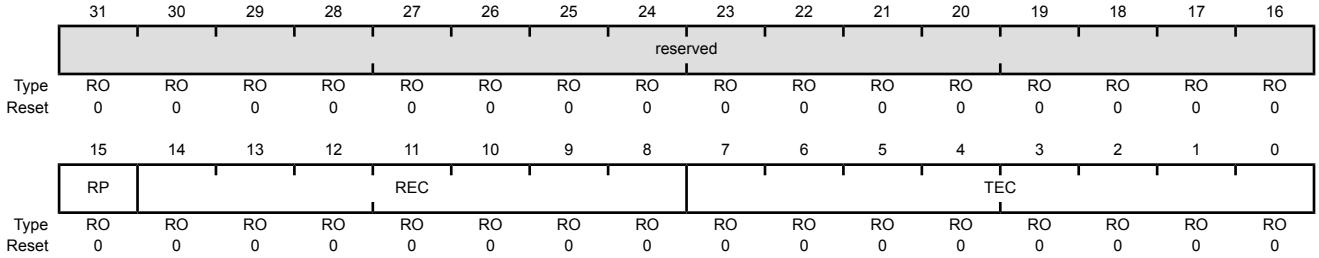
Value	Description
0x0	No Error
0x1	Stuff Error More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
0x2	Format Error A fixed format part of the received frame has the wrong format.
0x3	ACK Error The message transmitted was not acknowledged by another node.
0x4	Bit 1 Error When a message is transmitted, the CAN controller monitors the data lines to detect any conflicts. When the arbitration field is transmitted, data conflicts are a part of the arbitration protocol. When other frame fields are transmitted, data conflicts are considered errors. A Bit 1 Error indicates that the device wanted to send a High level (logical 1) but the monitored bus value was Low (logical 0).
0x5	Bit 0 Error A Bit 0 Error indicates that the device wanted to send a Low level (logical 0), but the monitored bus value was High (logical 1). During bus-off recovery, this status is set each time a sequence of 11 High bits has been monitored. By checking for this status, software can monitor the proceeding of the bus-off recovery sequence without any disturbances to the bus.
0x6	CRC Error The CRC checksum was incorrect in the received message, indicating that the calculated value received did not match the calculated CRC of the data.
0x7	No Event When the LEC bit shows this value, no CAN bus event was detected since this value was written to the LEC field.

Register 3: CAN Error Counter (CANERR), offset 0x008

This register contains the error counter values, which can be used to analyze the cause of an error.

CAN Error Counter (CANERR)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
CAN2 base: 0x4004.2000
Offset 0x008
Type RO, reset 0x0000.0000



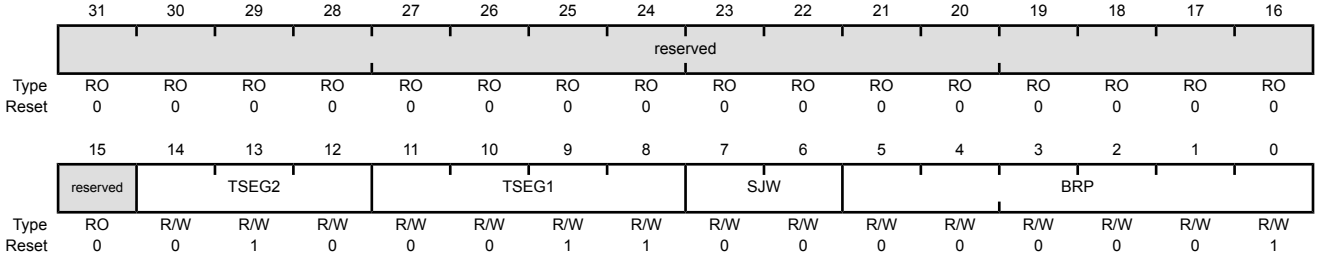
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	RP	RO	0	Received Error Passive Value: 0 – The Receive Error counter is below the Error Passive level (127 or less). Value: 1 – The Receive Error counter has reached the Error Passive level (128 or greater).
14:8	REC	RO	0x00	Receive Error Counter This field contains the state of the receiver error counter (0 to 127).
7:0	TEC	RO	0x00	Transmit Error Counter This field contains the state of the transmit error counter (0 to 255).

Register 4: CAN Bit Timing (CANBIT), offset 0x00C

This register is used to program the bit width and bit quantum. Values are programmed to the system clock frequency. This register is write-enabled by setting the CCE and INIT bits in the **CANCTL** register. See “Bit Time and Bit Rate” on page 873 for more information.

CAN Bit Timing (CANBIT)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
CAN2 base: 0x4004.2000
Offset 0x00C
Type R/W, reset 0x0000.2301



Bit/Field	Name	Type	Reset	Description
31:15	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14:12	TSEG2	R/W	0x2	Time Segment after Sample Point 0x00-0x07: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. So, for example, the reset value of 0x2 means that 3 (2+1) bit time quanta are defined for Phase2 (see Figure 17-4 on page 874). The bit time quanta is defined by the BRP field.
11:8	TSEG1	R/W	0x3	Time Segment Before Sample Point 0x00-0x0F: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. So, for example, the reset value of 0x3 means that 4 (3+1) bit time quanta are defined for Phase1 (see Figure 17-4 on page 874). The bit time quanta is defined by the BRP field.
7:6	SJW	R/W	0x0	(Re)Synchronization Jump Width 0x00-0x03: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. During the start of frame (SOF), if the CAN controller detects a phase error (misalignment), it can adjust the length of TSEG2 or TSEG1 by the value in SJW. So the reset value of 0 adjusts the length by 1 bit time quanta.
5:0	BRP	R/W	0x1	Baud Rate Prescaler The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quantum. 0x00-0x03F: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. BRP defines the number of CAN clock periods that make up 1 bit time quanta, so the reset value is 2 bit time quanta (1+1). The **CANBRPE** register can be used to further divide the bit time.

Register 5: CAN Interrupt (CANINT), offset 0x010

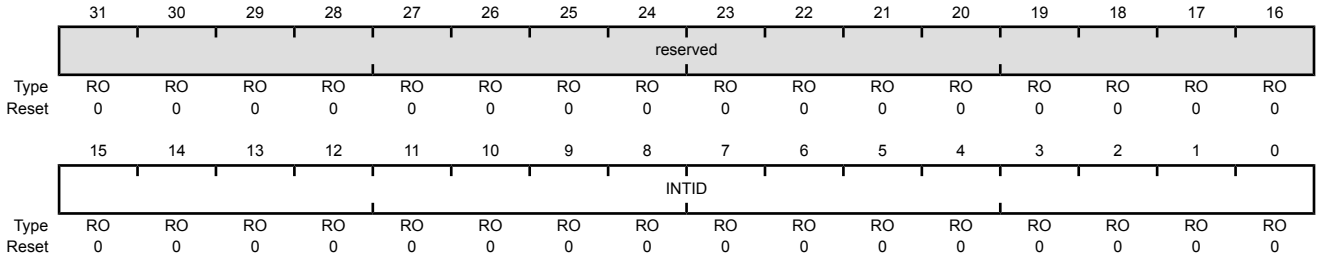
This register indicates the source of the interrupt.

If several interrupts are pending, the **CAN Interrupt (CANINT)** register points to the pending interrupt with the highest priority, disregarding the order in which the interrupts occurred. An interrupt remains pending until the CPU has cleared it. If the `INTID` field is not 0x0000 (the default) and the `IE` bit in the **CANCTL** register is set, the interrupt is active. The interrupt line remains active until the `INTID` field is cleared by reading the **CANSTS** register, or until the `IE` bit in the **CANCTL** register is cleared.

Note: Reading the **CAN Status (CANSTS)** register clears the **CAN Interrupt (CANINT)** register, if it is pending.

CAN Interrupt (CANINT)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
CAN2 base: 0x4004.2000
Offset 0x010
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	INTID	RO	0x0000	Interrupt Identifier The number in this field indicates the source of the interrupt.

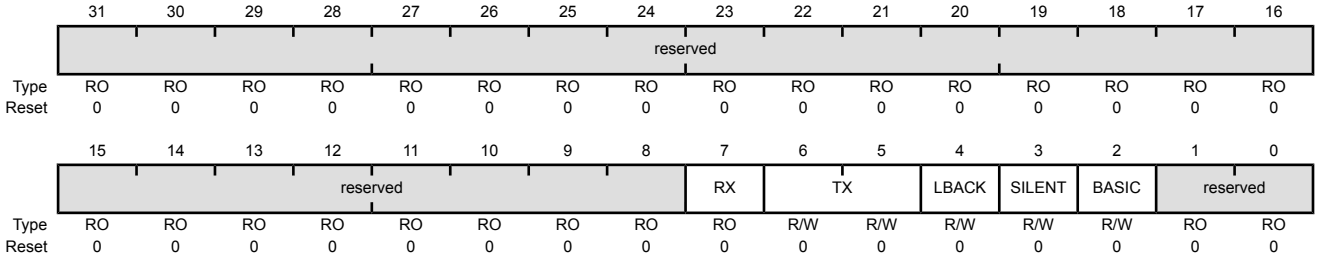
Value	Description
0x0000	No interrupt pending
0x0001-0x0020	Number of the message object that caused the interrupt
0x0021-0x7FFF	Reserved
0x8000	Status Interrupt
0x8001-0xFFFF	Reserved

Register 6: CAN Test (CANTST), offset 0x014

This register is used for self-test and external pin access. It is write-enabled by setting the TEST bit in the **CANCTL** register. Different test functions may be combined, however, CAN transfers are affected if the TX bits in this register are not zero.

CAN Test (CANTST)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
CAN2 base: 0x4004.2000
Offset 0x014
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	RX	RO	0	Receive Observation

Value	Description
0	The CANnRx pin is low.
1	The CANnRx pin is high.

Bit/Field	Name	Type	Reset	Description
6:5	TX	R/W	0x0	Transmit Control Overrides control of the CANnTx pin.

Value	Description
0x0	CAN Module Control CANnTx is controlled by the CAN module; default operation
0x1	Sample Point The sample point is driven on the CANnTx signal. This mode is useful to monitor bit timing.
0x2	Driven Low CANnTx drives a low value. This mode is useful for checking the physical layer of the CAN bus.
0x3	Driven High CANnTx drives a high value. This mode is useful for checking the physical layer of the CAN bus.

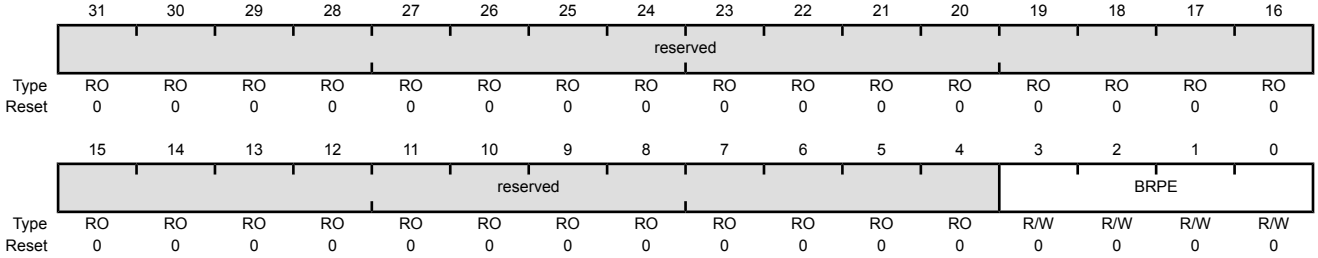
Bit/Field	Name	Type	Reset	Description
4	LBACK	R/W	0	Loopback Mode Value: 0 – Loopback mode is disabled. 1 – Loopback mode is enabled. In loopback mode, the data from the transmitter is routed into the receiver. Any data on the receive input is ignored.
3	SILENT	R/W	0	Silent Mode Value: 0 – Silent mode is disabled. 1 – Silent mode is enabled. In silent mode, the CAN controller does not transmit data but instead monitors the bus. This mode is also known as Bus Monitor mode.
2	BASIC	R/W	0	Basic Mode Value: 0 – Basic mode is disabled. 1 – Basic mode is enabled. In basic mode, software should use the **CANIF1** registers as the transmit buffer and use the **CANIF2** registers as the receive buffer.
1:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 7: CAN Baud Rate Prescaler Extension (CANBRPE), offset 0x018

This register is used to further divide the bit time set with the BRP bit in the **CANBIT** register. It is write-enabled by setting the CCE bit in the **CANCTL** register.

CAN Baud Rate Prescaler Extension (CANBRPE)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
CAN2 base: 0x4004.2000
Offset 0x018
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	BRPE	R/W	0x0	Baud Rate Prescaler Extension 0x00-0x0F: Extend the BRP bit in the **CANBIT** register to values up to 1023. The actual interpretation by the hardware is one more than the value programmed by BRPE (MSBs) and BRP (LSBs).

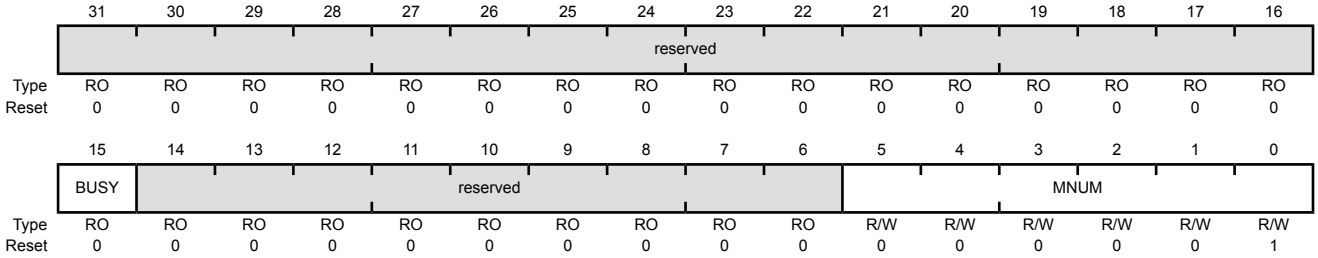
Register 8: CAN IF1 Command Request (CANIF1CRQ), offset 0x020

Register 9: CAN IF2 Command Request (CANIF2CRQ), offset 0x080

A message transfer is started as soon as there is a write of the message object number to the MNUM field when the TXRQST bit in the **CANIF1MCTL** register is set. With this write operation, the BUSY bit is automatically set to indicate that a transfer between the CAN Interface Registers and the internal message RAM is in progress. After a wait time of 3 to 6 CAN_CLK periods, the transfer between the interface register and the message RAM completes, which then clears the BUSY bit.

CAN IF1 Command Request (CANIF1CRQ)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
CAN2 base: 0x4004.2000
Offset 0x020
Type R/W, reset 0x0000.0001



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	BUSY	RO	0	Busy Flag Value: 0 — This bit is cleared when read/write action has finished. Value: 1 — This bit is set when a write occurs to the message number in this register.
14:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	MNUM	R/W	0x01	Message Number Selects one of the 32 message objects in the message RAM for data transfer. The message objects are numbered from 1 to 32. Value: 0x00 — Reserved. 0 is not a valid message number; it is interpreted as 0x20, or object 32. Value: 0x01-0x20 — Message Number. Indicates specified message object 1 to 32. Value: 0x21-0x3F — Reserved. Not a valid message number; values are shifted and it is interpreted as 0x01-0x1F.

Register 10: CAN IF1 Command Mask (CANIF1CMSK), offset 0x024

Register 11: CAN IF2 Command Mask (CANIF2CMSK), offset 0x084

Reading the Command Mask registers provides status for various functions. Writing to the Command Mask registers specifies the transfer direction and selects which buffer registers are the source or target of the data transfer.

Note that when a read from the message object buffer occurs when the WRNRD bit is clear and the CLRINTPND and/or NEWDAT bits are set, the interrupt pending and/or new data flags in the message object buffer are cleared.

CAN IF1 Command Mask (CANIF1CMSK)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
CAN2 base: 0x4004.2000
Offset 0x024
Type R/W, reset 0x0000.0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved								WRNRD	MASK	ARB	CONTROL	CLRINTPND	NEWDAT / TXRQST	DATAA	DATAB
RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	WRNRD	R/W	0	Write, Not Read

Value	Description
0	Transfer the data in the CAN message object specified by the the MNUM field in the **CANIFnCRQ** register into the CANIFn registers.
1	Transfer the data in the CANIFn registers to the CAN message object specified by the MNUM field in the **CAN Command Request (CANIFnCRQ)**.

Note: Interrupt pending and new data conditions in the message buffer can be cleared by reading from the buffer (WRNRD = 0) when the CLRINTPND and/or NEWDAT bits are set.

6	MASK	R/W	0	Access Mask Bits

Value	Description
0	Mask bits unchanged.
1	Transfer IDMASK + DIR + MXTD of the message object into the Interface registers.

Bit/Field	Name	Type	Reset	Description
5	ARB	R/W	0	Access Arbitration Bits Value: 0 – Arbitration bits unchanged. 1 – Transfer ID + DIR + XTD + MSGVAL of the message object into the Interface registers.
4	CONTROL	R/W	0	Access Control Bits Value: 0 – Control bits unchanged. 1 – Transfer control bits from the **CANIFnMCTL** register into the Interface registers.
3	CLRINTPND	R/W	0	Clear Interrupt Pending Bit The function of this bit depends on the configuration of the WRNRD bit. Value: 0 – If WRNRD is clear, the interrupt pending status is transferred from the message buffer into the **CANIFnMCTL** register. If WRNRD is set, the INTPND bit in the message object remains unchanged. 1 – If WRNRD is clear, the interrupt pending status is cleared in the message buffer. Note the value of this bit that is transferred to the **CANIFnMCTL** register always reflects the status of the bits before clearing. If WRNRD is set, the INTPND bit is cleared in the message object.
2	NEWDAT / TXRQST	R/W	0	NEWDAT / TXRQST Bit The function of this bit depends on the configuration of the WRNRD bit. Value: 0 – If WRNRD is clear, the value of the new data status is transferred from the message buffer into the **CANIFnMCTL** register. If WRNRD is set, a transmission is not requested. 1 – If WRNRD is clear, the new data status is cleared in the message buffer. Note the value of this bit that is transferred to the **CANIFnMCTL** register always reflects the status of the bits before clearing. If WRNRD is set, a transmission is requested. Note that when this bit is set, the TXRQST bit in the **CANIFnMCTL** register is ignored.

Bit/Field	Name	Type	Reset	Description
1	DATAA	R/W	0	Access Data Byte 0 to 3 The function of this bit depends on the configuration of the WRNRD bit. Value Description 0 Data bytes 0-3 are unchanged. 1 If WRNRD is clear, transfer data bytes 0-3 in **CANIFnDA1** and **CANIFnDA2** to the message object. If WRNRD is set, transfer data bytes 0-3 in message object to **CANIFnDA1** and **CANIFnDA2**.
0	DATAB	R/W	0	Access Data Byte 4 to 7 The function of this bit depends on the configuration of the WRNRD bit as follows: Value Description 0 Data bytes 4-7 are unchanged. 1 If WRNRD is clear, transfer data bytes 4-7 in **CANIFnDA1** and **CANIFnDA2** to the message object. If WRNRD is set, transfer data bytes 4-7 in message object to **CANIFnDA1** and **CANIFnDA2**.

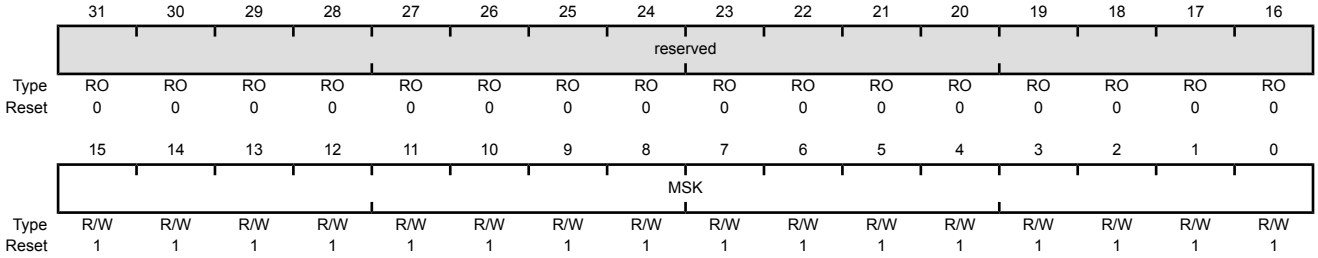
Register 12: CAN IF1 Mask 1 (CANIF1MSK1), offset 0x028

Register 13: CAN IF2 Mask 1 (CANIF2MSK1), offset 0x088

The mask information provided in this register accompanies the data (**CANIFnDAn**), arbitration information (**CANIFnARBn**), and control information (**CANIFnMCTL**) to the message object in the message RAM. The mask is used with the ID bit in the **CANIFnARBn** register for acceptance filtering. Additional mask information is contained in the **CANIFnMSK2** register.

CAN IF1 Mask 1 (CANIF1MSK1)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
CAN2 base: 0x4004.2000
Offset 0x028
Type R/W, reset 0x0000.FFFF



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	MSK	R/W	0xFFFF	Identifier Mask When using a 29-bit identifier, these bits are used for bits [15:0] of the ID. The MSK field in the **CANIFnMSK2** register are used for bits [28:16] of the ID. When using an 11-bit identifier, these bits are ignored.

Value	Description
0	The corresponding identifier field (ID) in the message object cannot inhibit the match in acceptance filtering.
1	The corresponding identifier field (ID) is used for acceptance filtering.

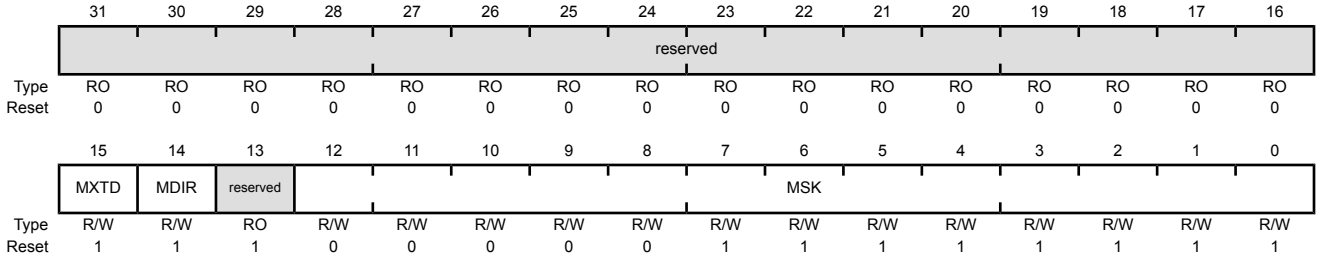
Register 14: CAN IF1 Mask 2 (CANIF1MSK2), offset 0x02C

Register 15: CAN IF2 Mask 2 (CANIF2MSK2), offset 0x08C

This register holds extended mask information that accompanies the **CANIFnMSK1** register.

CAN IF1 Mask 2 (CANIF1MSK2)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
CAN2 base: 0x4004.2000
Offset 0x02C
Type R/W, reset 0x0000.FFFF



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	MXTD	R/W	1	Mask Extended Identifier Value: Description 0: The extended identifier bit (XTD in the **CANIFnARB2** register) has no effect on the acceptance filtering. 1: The extended identifier bit XTD is used for acceptance filtering.
14	MDIR	R/W	1	Mask Message Direction Value: Description 0: The message direction bit (DIR in the **CANIFnARB2** register) has no effect for acceptance filtering. 1: The message direction bit DIR is used for acceptance filtering.
13	reserved	RO	1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
12:0	MSK	R/W	0xFF	Identifier Mask When using a 29-bit identifier, these bits are used for bits [28:16] of the ID. The MSK field in the **CANIFnMSK1** register are used for bits [15:0] of the ID. When using an 11-bit identifier, MSK[12:2] are used for bits [10:0] of the ID.

Value	Description
0	The corresponding identifier field (ID) in the message object cannot inhibit the match in acceptance filtering.
1	The corresponding identifier field (ID) is used for acceptance filtering.

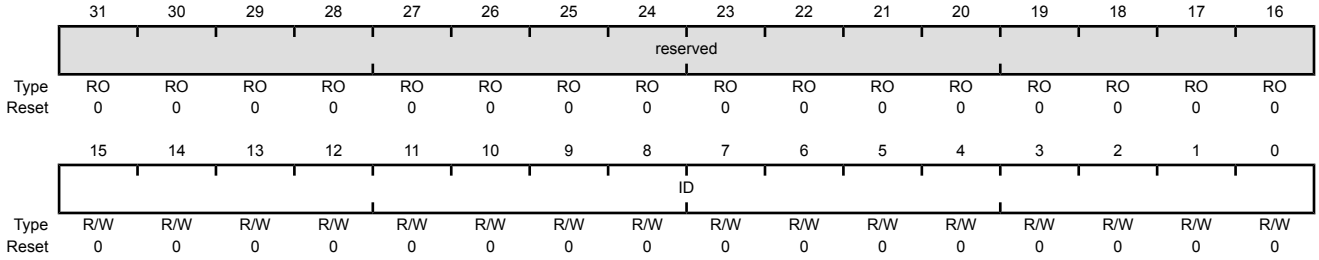
Register 16: CAN IF1 Arbitration 1 (CANIF1ARB1), offset 0x030

Register 17: CAN IF2 Arbitration 1 (CANIF2ARB1), offset 0x090

These registers hold the identifiers for acceptance filtering.

CAN IF1 Arbitration 1 (CANIF1ARB1)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
CAN2 base: 0x4004.2000
Offset 0x030
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	ID	R/W	0x0000	Message Identifier This bit field is used with the `ID` field in the **CANIFnARB2** register to create the message identifier. When using a 29-bit identifier, bits 15:0 of the **CANIFnARB1** register are [15:0] of the ID, while bits 12:0 of the **CANIFnARB2** register are [28:16] of the ID. When using an 11-bit identifier, these bits are not used.

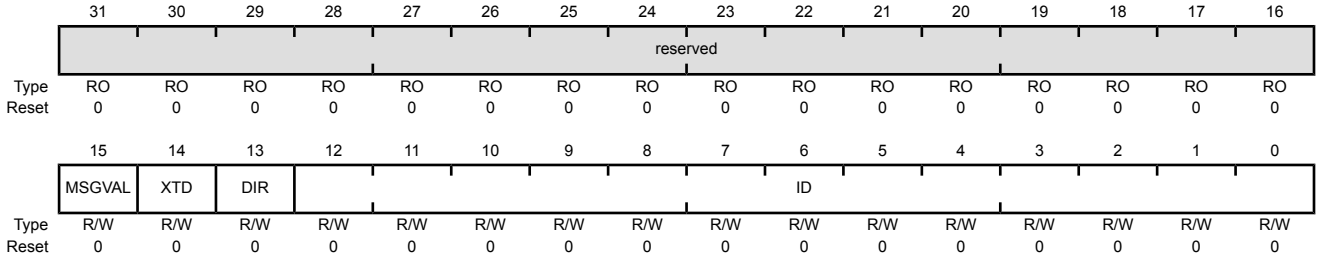
Register 18: CAN IF1 Arbitration 2 (CANIF1ARB2), offset 0x034

Register 19: CAN IF2 Arbitration 2 (CANIF2ARB2), offset 0x094

These registers hold information for acceptance filtering.

CAN IF1 Arbitration 2 (CANIF1ARB2)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
CAN2 base: 0x4004.2000
Offset 0x034
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	MSGVAL	R/W	0	Message Valid Value Description 0 The message object is ignored by the message handler. 1 The message object is configured and ready to be considered by the message handler within the CAN controller. All unused message objects should have this bit cleared during initialization and before clearing the INIT bit in the **CANCTL** register. The MSGVAL bit must also be cleared before any of the following bits are modified or if the message object is no longer required: the ID fields in the **CANIFnARBn** registers, the XTD and DIR bits in the **CANIFnARB2** register, or the DLC field in the **CANIFnMCTL** register.
14	XTD	R/W	0	Extended Identifier Value Description 0 An 11-bit Standard Identifier is used for this message object. 1 A 29-bit Extended Identifier is used for this message object.

Bit/Field	Name	Type	Reset	Description
13	DIR	R/W	0	Message Direction Value Description 0 Receive. When the TXRQST bit in the **CANIFnMCTL** register is set, a remote frame with the identifier of this message object is received. On reception of a data frame with matching identifier, that message is stored in this message object. 1 Transmit. When the TXRQST bit in the **CANIFnMCTL** register is set, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the TXRQST bit of this message object is set (if RMTEN=1).
12:0	ID	R/W	0x000	Message Identifier This bit field is used with the ID field in the **CANIFnARB2** register to create the message identifier. When using a 29-bit identifier, ID[15:0] of the **CANIFnARB1** register are [15:0] of the ID, while these bits, ID[12:0], are [28:16] of the ID. When using an 11-bit identifier, ID[12:2] are used for bits [10:0] of the ID. The ID field in the **CANIFnARB1** register is ignored.

Register 20: CAN IF1 Message Control (CANIF1MCTL), offset 0x038

Register 21: CAN IF2 Message Control (CANIF2MCTL), offset 0x098

This register holds the control information associated with the message object to be sent to the Message RAM.

CAN IF1 Message Control (CANIF1MCTL)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
CAN2 base: 0x4004.2000
Offset 0x038
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NEWDAT	MSGLST	INTPND	UMASK	TXIE	RXIE	RMTEN	TXRQST	EOB	reserved			DLC			
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	NEWDAT	R/W	0	New Data Value: 0 — No new data has been written into the data portion of this message object by the message handler since the last time this flag was cleared by the CPU. Value: 1 — The message handler or the CPU has written new data into the data portion of this message object.
14	MSGLST	R/W	0	Message Lost Value: 0 — No message was lost since the last time this bit was cleared by the CPU. Value: 1 — The message handler stored a new message into this object when NEWDAT was set; the CPU has lost a message. This bit is only valid for message objects when the DIR bit in the **CANIFnARB2** register is clear (receive).
13	INTPND	R/W	0	Interrupt Pending Value: 0 — This message object is not the source of an interrupt. Value: 1 — This message object is the source of an interrupt. The interrupt identifier in the **CANINT** register points to this message object if there is not another interrupt source with a higher priority.

Bit/Field	Name	Type	Reset	Description
12	UMASK	R/W	0	Use Acceptance Mask Value: 0 — Mask is ignored. 1 — Use mask (MSK, MXTD, and MDIR bits in the **CANIFnMSKn** registers) for acceptance filtering.
11	TXIE	R/W	0	Transmit Interrupt Enable Value: 0 — The INTPND bit in the **CANIFnMCTL** register is unchanged after a successful transmission of a frame. 1 — The INTPND bit in the **CANIFnMCTL** register is set after a successful transmission of a frame.
10	RXIE	R/W	0	Receive Interrupt Enable Value: 0 — The INTPND bit in the **CANIFnMCTL** register is unchanged after a successful reception of a frame. 1 — The INTPND bit in the **CANIFnMCTL** register is set after a successful reception of a frame.
9	RMTEN	R/W	0	Remote Enable Value: 0 — At the reception of a remote frame, the TXRQST bit in the **CANIFnMCTL** register is left unchanged. 1 — At the reception of a remote frame, the TXRQST bit in the **CANIFnMCTL** register is set.
8	TXRQST	R/W	0	Transmit Request Value: 0 — This message object is not waiting for transmission. 1 — The transmission of this message object is requested and is not yet done. **Note:** If the WRNRD and TXRQST bits in the **CANIFnCMSK** register are set, this bit is ignored.

Bit/Field	Name	Type	Reset	Description
7	EOB	R/W	0	End of Buffer

Value	Description
0	Message object belongs to a FIFO Buffer and is not the last message object of that FIFO Buffer.
1	Single message object or last message object of a FIFO Buffer.

This bit is used to concatenate two or more message objects (up to 32) to build a FIFO buffer. For a single message object (thus not belonging to a FIFO buffer), this bit must be set.

6:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	DLC	R/W	0x0	Data Length Code

Value	Description
0x0-0x8	Specifies the number of bytes in the data frame.
0x9-0xF	Defaults to a data frame with 8 bytes.

The DLC field in the **CANIFnMCTL** register of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it writes DLC to the value given by the received message.

Register 22: CAN IF1 Data A1 (CANIF1DA1), offset 0x03C

Register 23: CAN IF1 Data A2 (CANIF1DA2), offset 0x040

Register 24: CAN IF1 Data B1 (CANIF1DB1), offset 0x044

Register 25: CAN IF1 Data B2 (CANIF1DB2), offset 0x048

Register 26: CAN IF2 Data A1 (CANIF2DA1), offset 0x09C

Register 27: CAN IF2 Data A2 (CANIF2DA2), offset 0x0A0

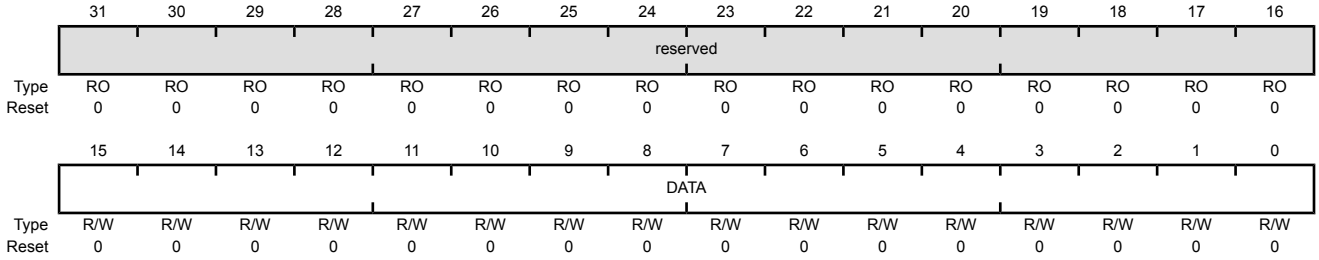
Register 28: CAN IF2 Data B1 (CANIF2DB1), offset 0x0A4

Register 29: CAN IF2 Data B2 (CANIF2DB2), offset 0x0A8

These registers contain the data to be sent or that has been received. In a CAN data frame, data byte 0 is the first byte to be transmitted or received and data byte 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte is transmitted first.

CAN IF1 Data A1 (CANIF1DA1)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
CAN2 base: 0x4004.2000
Offset 0x03C
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	DATA	R/W	0x0000	Data The **CANIFnDA1** registers contain data bytes 1 and 0; **CANIFnDA2** data bytes 3 and 2; **CANIFnDB1** data bytes 5 and 4; and **CANIFnDB2** data bytes 7 and 6.

Register 30: CAN Transmission Request 1 (CANTXRQ1), offset 0x100

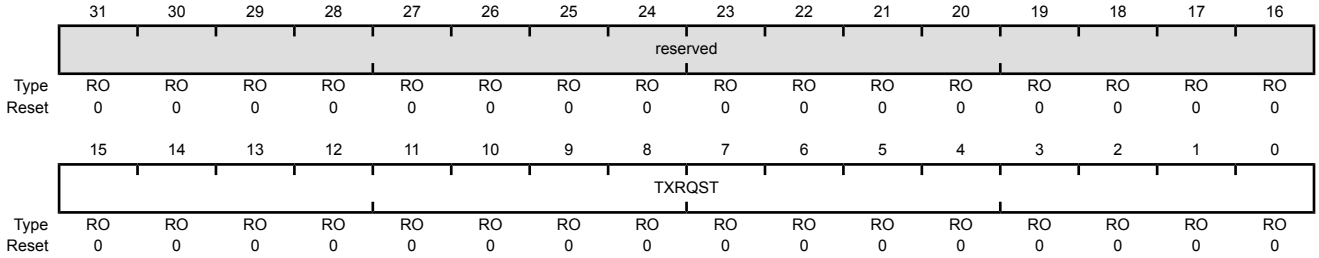
Register 31: CAN Transmission Request 2 (CANTXRQ2), offset 0x104

The **CANTXRQ1** and **CANTXRQ2** registers hold the TXRQST bits of the 32 message objects. By reading out these bits, the CPU can check which message object has a transmission request pending. The TXRQST bit of a specific message object can be changed by three sources: (1) the CPU via the **CANIFnMCTL** register, (2) the message handler state machine after the reception of a remote frame, or (3) the message handler state machine after a successful transmission.

The **CANTXRQ1** register contains the TXRQST bits of the first 16 message objects in the message RAM; the **CANTXRQ2** register contains the TXRQST bits of the second 16 message objects.

CAN Transmission Request 1 (CANTXRQ1)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
CAN2 base: 0x4004.2000
Offset 0x100
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TXRQST	RO	0x0000	Transmission Request Bits

Value	Description
0	The corresponding message object is not waiting for transmission.
1	The transmission of the corresponding message object is requested and is not yet done.

Register 32: CAN New Data 1 (CANNWDA1), offset 0x120

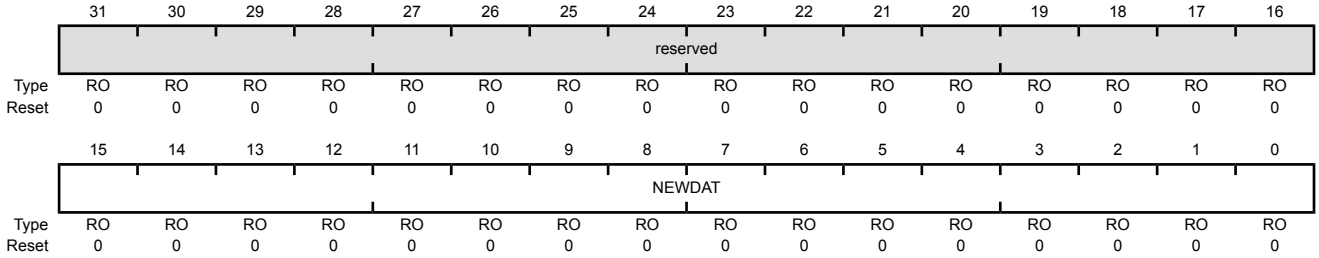
Register 33: CAN New Data 2 (CANNWDA2), offset 0x124

The **CANNWDA1** and **CANNWDA2** registers hold the NEWDAT bits of the 32 message objects. By reading these bits, the CPU can check which message object has its data portion updated. The NEWDAT bit of a specific message object can be changed by three sources: (1) the CPU via the **CANIFnMCTL** register, (2) the message handler state machine after the reception of a data frame, or (3) the message handler state machine after a successful transmission.

The **CANNWDA1** register contains the NEWDAT bits of the first 16 message objects in the message RAM; the **CANNWDA2** register contains the NEWDAT bits of the second 16 message objects.

CAN New Data 1 (CANNWDA1)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
CAN2 base: 0x4004.2000
Offset 0x120
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	NEWDAT	RO	0x0000	New Data Bits

Value	Description
0	No new data has been written into the data portion of the corresponding message object by the message handler since the last time this flag was cleared by the CPU.
1	The message handler or the CPU has written new data into the data portion of the corresponding message object.

Register 34: CAN Message 1 Interrupt Pending (CANMSG1INT), offset 0x140

Register 35: CAN Message 2 Interrupt Pending (CANMSG2INT), offset 0x144

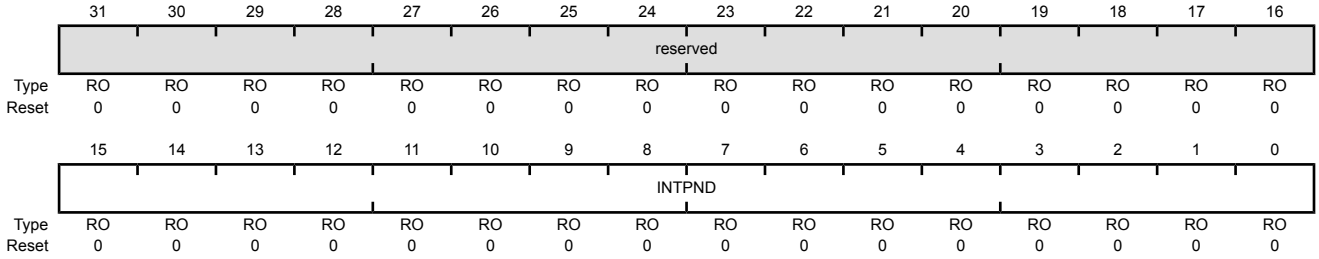
The **CANMSG1INT** and **CANMSG2INT** registers hold the INTPND bits of the 32 message objects. By reading these bits, the CPU can check which message object has an interrupt pending. The INTPND bit of a specific message object can be changed through two sources: (1) the CPU via the **CANIFnMCTL** register, or (2) the message handler state machine after the reception or transmission of a frame.

This field is also encoded in the **CANINT** register.

The **CANMSG1INT** register contains the INTPND bits of the first 16 message objects in the message RAM; the **CANMSG2INT** register contains the INTPND bits of the second 16 message objects.

CAN Message 1 Interrupt Pending (CANMSG1INT)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
CAN2 base: 0x4004.2000
Offset 0x140
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	INTPND	RO	0x0000	Interrupt Pending Bits

Value	Description
0	The corresponding message object is not the source of an interrupt.
1	The corresponding message object is the source of an interrupt.

Register 36: CAN Message 1 Valid (CANMSG1VAL), offset 0x160

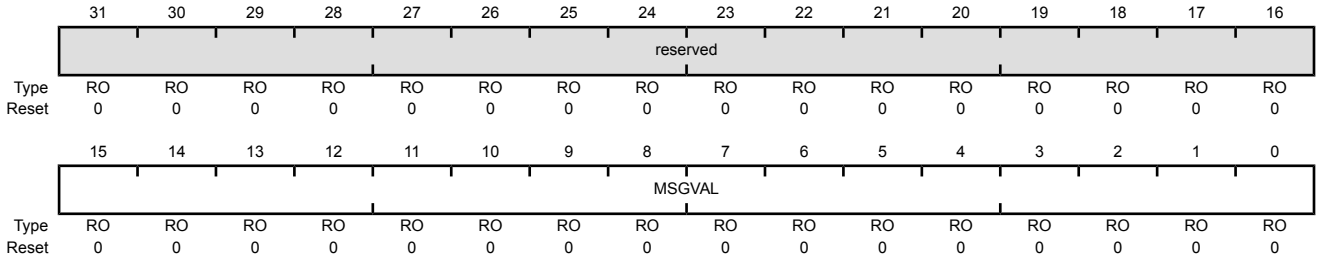
Register 37: CAN Message 2 Valid (CANMSG2VAL), offset 0x164

The **CANMSG1VAL** and **CANMSG2VAL** registers hold the MSGVAL bits of the 32 message objects. By reading these bits, the CPU can check which message object is valid. The message valid bit of a specific message object can be changed with the **CANIFnARB2** register.

The **CANMSG1VAL** register contains the MSGVAL bits of the first 16 message objects in the message RAM; the **CANMSG2VAL** register contains the MSGVAL bits of the second 16 message objects in the message RAM.

CAN Message 1 Valid (CANMSG1VAL)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
CAN2 base: 0x4004.2000
Offset 0x160
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	MSGVAL	RO	0x0000	Message Valid Bits

Value	Description
0	The corresponding message object is not configured and is ignored by the message handler.
1	The corresponding message object is configured and should be considered by the message handler.

18 Ethernet Controller

The Stellaris® Ethernet Controller consists of a fully integrated media access controller (MAC) and network physical (PHY) interface. The Ethernet Controller conforms to *IEEE 802.3* specifications and fully supports 10BASE-T and 100BASE-TX standards.

The Stellaris Ethernet Controller module has the following features:

- Conforms to the IEEE 802.3-2002 specification
  - 10BASE-T/100BASE-TX IEEE-802.3 compliant. Requires only a dual 1:1 isolation transformer interface to the line
  - 10BASE-T/100BASE-TX ENDEC, 100BASE-TX scrambler/descrambler
  - Full-featured auto-negotiation
- Multiple operational modes
  - Full- and half-duplex 100 Mbps
  - Full- and half-duplex 10 Mbps
  - Power-saving and power-down modes
- Highly configurable
  - Programmable MAC address
  - LED activity selection
  - Promiscuous mode support
  - CRC error-rejection control
  - User-configurable interrupts
- Physical media manipulation
  - MDI/MDI-X cross-over support through software assist
  - Register-programmable transmit amplitude
  - Automatic polarity correction and 10BASE-T signal reception
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Separate channels for transmit and receive
  - Receive channel request asserted on packet receipt
  - Transmit channel request asserted on empty transmit FIFO

18.1 Block Diagram

As shown in Figure 18-1 on page 911, the Ethernet Controller is functionally divided into two layers: the Media Access Controller (MAC) layer and the Network Physical (PHY) layer. These layers correspond to the OSI model layers 2 and 1, respectively. The CPU accesses the Ethernet Controller via the MAC layer. The MAC layer provides transmit and receive processing for Ethernet frames. The MAC layer also provides the interface to the PHY layer via an internal Media Independent Interface (MII). The PHY layer communicates with the Ethernet bus.

Figure 18-1. Ethernet Controller

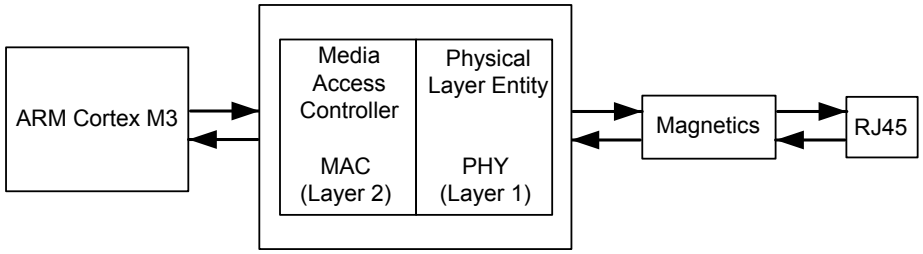
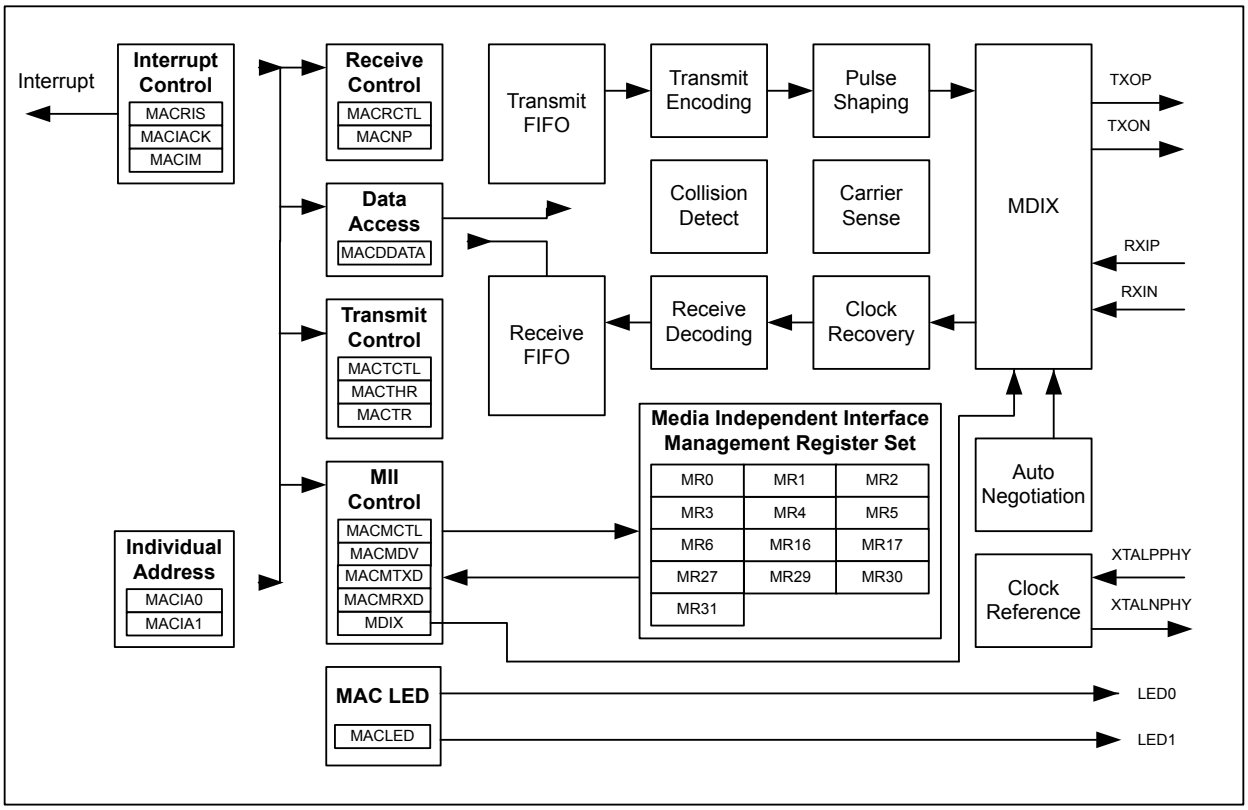


Figure 18-2 on page 911 shows more detail of the internal structure of the Ethernet Controller and how the register set relates to various functions.

Figure 18-2. Ethernet Controller Block Diagram



18.2 Signal Description

The following table lists the external signals of the Ethernet Controller and describes the function of each. The Ethernet LED signals are alternate functions for GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placement for the LED signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 419) should be set to choose the LED function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 437) to assign the LED0 and LED1 signals to the specified GPIO port pins. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 395. The remaining signals (with the word "fixed" in the Pin Mux/Pin Assignment column) have a fixed pin assignment and function.

Table 18-1. Ethernet Signals (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
ERBIAS	33	fixed	O	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
LED0	59	PF3 (1)	O	TTL	Ethernet LED 0.
LED1	60	PF2 (1)	O	TTL	Ethernet LED 1.
MDIO	58	fixed	I/O	OD	MDIO of the Ethernet PHY.
RXIN	37	fixed	I	Analog	RXIN of the Ethernet PHY.
RXIP	40	fixed	I	Analog	RXIP of the Ethernet PHY.
TXON	46	fixed	O	TTL	TXON of the Ethernet PHY.
TXOP	43	fixed	O	TTL	TXOP of the Ethernet PHY.
XTALNPHY	17	fixed	O	Analog	Ethernet PHY XTALN 25-MHz oscillator crystal output. Leave this pin unconnected when using a single-ended 25-MHz clock input connected to the XTALPPHY pin.
XTALPPHY	16	fixed	I	Analog	Ethernet PHY XTALP 25-MHz oscillator crystal input or external clock reference input.

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

Table 18-2. Ethernet Signals (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
ERBIAS	J3	fixed	O	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
LED0	J12	PF3 (1)	O	TTL	Ethernet LED 0.
LED1	J11	PF2 (1)	O	TTL	Ethernet LED 1.
MDIO	L9	fixed	I/O	OD	MDIO of the Ethernet PHY.
RXIN	L7	fixed	I	Analog	RXIN of the Ethernet PHY.
RXIP	M7	fixed	I	Analog	RXIP of the Ethernet PHY.
TXON	L8	fixed	O	TTL	TXON of the Ethernet PHY.
TXOP	M8	fixed	O	TTL	TXOP of the Ethernet PHY.
XTALNPHY	J1	fixed	O	Analog	Ethernet PHY XTALN 25-MHz oscillator crystal output. Leave this pin unconnected when using a single-ended 25-MHz clock input connected to the XTALPPHY pin.

Table 18-2. Ethernet Signals (108BGA) *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
XTALPPHY	J2	fixed	I	Analog	Ethernet PHY XTALP 25-MHz oscillator crystal input or external clock reference input.

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

18.3 Functional Description

Note: A 12.4-kΩ resistor should be connected between the ERBIAS and ground. The 12.4-kΩ resistor should have a 1% tolerance and should be located in close proximity to the ERBIAS pin. Power dissipation in the resistor is low, so a chip resistor of any geometry may be used.

The functional description of the Ethernet Controller is discussed in the following sections.

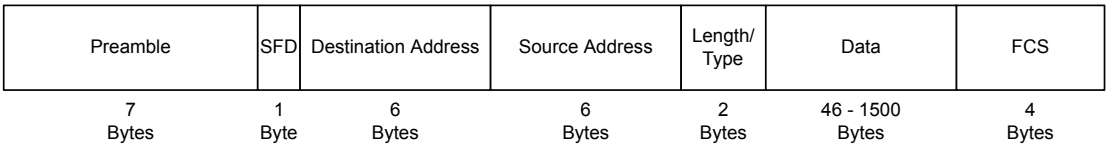
18.3.1 MAC Operation

The following sections describe the operation of the MAC layer, including an overview of the Ethernet frame format, the MAC layer FIFOs, Ethernet transmission and reception options, and LED indicators.

18.3.1.1 Ethernet Frame Format

Ethernet data is carried by Ethernet frames. The basic frame format is shown in Figure 18-3 on page 913.

Figure 18-3. Ethernet Frame



The seven fields of the frame are transmitted from left to right. The bits within the frame are transmitted from least to most significant bit.

- Preamble

  The Preamble field is used to synchronize with the received frame's timing. The preamble is 7 octets long.

- Start Frame Delimiter (SFD)

  The SFD field follows the preamble pattern and indicates the start of the frame. Its value is 1010.1011b.

- Destination Address (DA)

  This field specifies destination addresses for which the frame is intended. The LSB (bit 16 of DA oct 1 in the frame, see Table 18-3 on page 915) of the DA determines whether the address is an individual (0), or group/multicast (1) address.

- Source Address (SA)

  The source address field identifies the station from which the frame was initiated.

- Length/Type Field

The meaning of this field depends on its numeric value. This field can be interpreted as length or type code. The maximum length of the data field is 1500 octets. If the value of the Length/Type field is less than or equal to 1500 decimal, it indicates the number of MAC client data octets. If the value of this field is greater than or equal to 1536 decimal, then it encodes the type interpretation. The meaning of the Length/Type field when the value is between 1500 and 1536 decimal is unspecified by the IEEE 802.3 standard. However, the Ethernet Controller assumes type interpretation if the value of the Length/Type field is greater than 1500 decimal. The definition of the Type field is specified in the IEEE 802.3 standard. The first of the two octets in this field is most significant.

- Data

  The data field is a sequence of octets that is at least 46 in length, up to 1500 in length. Full data transparency is provided so any values can appear in this field. A minimum frame size of 46 octets is required to meet the IEEE standard. If the frame size is too small, the Ethernet Controller automatically appends extra bits (a pad), thus the pad can have a size of 0 to 46 octets. Data padding can be disabled by clearing the PADEN bit in the **Ethernet MAC Transmit Control (MACTCTL)** register.

  For the Ethernet Controller, data sent/received can be larger than 1500 bytes without causing a Frame Too Long error. Instead, a FIFO overrun error is reported using the FOV bit in the **Ethernet MAC Raw Interrupt Status (MACRIS)** register when the frame received is too large to fit into the Ethernet Controller's 2K RAM.

- Frame Check Sequence (FCS)

  The frame check sequence carries the cyclic redundancy check (CRC) value. The CRC is computed over the destination address, source address, length/type, and data (including pad) fields using the CRC-32 algorithm. The Ethernet Controller computes the FCS value one nibble at a time. For transmitted frames, this field is automatically inserted by the MAC layer, unless disabled by clearing the CRC bit in the **MACTCTL** register. For received frames, this field is automatically checked. If the FCS does not pass, the frame is not placed in the RX FIFO, unless the FCS check is disabled by clearing the BADCRC bit in the **MACRCTL** register.

18.3.1.2 MAC Layer FIFOs

The Ethernet Controller is capable of simultaneous transmission and reception. This feature is enabled by setting the DUPLEX bit in the **MACTCTL** register.

For Ethernet frame transmission, a 2-KB transmit FIFO is provided that can be used to store a single frame. While the *IEEE 802.3 specification* limits the size of an Ethernet frame's payload section to 1500 Bytes, the Ethernet Controller places no such limit. The full buffer can be used for a payload of up to 2032 bytes (as the first 16 bytes in the FIFO are reserved for destination address, source address and length/type information).

For Ethernet frame reception, a 2-KB receive FIFO is provided that can be used to store multiple frames, up to a maximum of 31 frames. If a frame is received, and there is insufficient space in the RX FIFO, an overflow error is indicated using the FOV bit in the **MACRIS** register.

For details regarding the TX and RX FIFO layout, refer to Table 18-3 on page 915. Please note the following difference between TX and RX FIFO layout. For the TX FIFO, the Data Length field in the first FIFO word refers to the Ethernet frame data payload, as shown in the 5th to nth FIFO positions. For the RX FIFO, the Frame Length field is the total length of the received Ethernet frame, including the Length/Type bytes and the FCS bits.

If FCS generation is disabled by clearing the CRC bit in the **MACTCTL** register, the last word in the TX FIFO must contain the FCS bytes for the frame that has been written to the FIFO.

Also note that if the length of the data payload section is not a multiple of 4, the FCS field is not aligned on a word boundary in the FIFO. However, for the RX FIFO, the beginning of the next frame is always on a word boundary.

Table 18-3. TX & RX FIFO Organization

FIFO Word Read/Write Sequence	Word Bit Fields	TX FIFO (Write)	RX FIFO (Read)
1st	7:0	Data Length Least Significant Byte	Frame Length Least Significant Byte
	15:8	Data Length Most Significant Byte	Frame Length Most Significant Byte
	23:16	DA oct 1	
	31:24	DA oct 2	
2nd	7:0	DA oct 3	
	15:8	DA oct 4	
	23:16	DA oct 5	
	31:24	DA oct 6	
3rd	7:0	SA oct 1	
	15:8	SA oct 2	
	23:16	SA oct 3	
	31:24	SA oct 4	
4th	7:0	SA oct 5	
	15:8	SA oct 6	
	23:16	Len/Type Most Significant Byte	
	31:24	Len/Type Least Significant Byte	
5th to nth	7:0	data oct n	
	15:8	data oct n+1	
	23:16	data oct n+2	
	31:24	data oct n+3	
last	7:0	FCS 1[a]	
	15:8	FCS 2[a]	
	23:16	FCS 3[a]	
	31:24	FCS 4[a]	

a. If the CRC bit in the **MACTCTL** register is clear, the FCS bytes must be written with the correct CRC. If the CRC bit is set, the Ethernet Controller automatically writes the FCS bytes.

18.3.1.3 Ethernet Transmission Options

At the MAC layer, the transmitter can be configured for both full-duplex and half-duplex operation by using the DUPLEX bit in the **MACTCTL** register. Note that in 10BASE-T half-duplex mode, the transmitted data is looped back on the receive path.

The Ethernet Controller automatically generates and inserts the Frame Check Sequence (FCS) at the end of the transmit frame when the CRC bit in the **MACTCTL** register is set. However, for test purposes, this feature can be disabled in order to generate a frame with an invalid CRC by clearing the CRC bit.

The *IEEE 802.3 specification* requires that the Ethernet frame payload section be a minimum of 46 bytes. The Ethernet Controller automatically pads the data section if the payload data section loaded

into the FIFO is less than the minimum 46 bytes when the PADEN bit in the **MACTCTL** register is set. This feature can be disabled by clearing the PADEN bit.

The transmitter must be enabled by setting the TXEN bit in the **MACTCTL** register.

18.3.1.4 Ethernet Reception Options

The Ethernet Controller RX FIFO should be cleared during software initialization. The receiver should first be disabled by clearing the RXEN bit in the **Ethernet MAC Receive Control (MACRCTL)** register, then the FIFO can be cleared by setting the RSTFIFO bit in the **MACRCTL** register.

The receiver automatically rejects frames that contain bad CRC values in the FCS field. In this case, a Receive Error interrupt is generated and the receive data is lost. To accept all frames, clear the BADCRC bit in the **MACRCTL** register.

In normal operating mode, the receiver accepts only those frames that have a destination address that matches the address programmed into the **Ethernet MAC Individual Address 0 (MACIA0)** and **Ethernet MAC Individual Address 1 (MACIA1)** registers. However, the Ethernet receiver can also be configured for Promiscuous and Multicast modes by setting the PRMS and AMUL bits in the **MACRCTL** register. It is important to note that when the receiver is enabled, all valid frames with a broadcast address of FF-FF-FF-FF-FF-FF in the Destination Address field are received and stored in the RX FIFO, even if the AMUL bit is not set.

18.3.1.5 LED Indicators

The Ethernet Controller supports two LED signals that can be used to indicate various states of operation. These signals are mapped to the LED0 and LED1 pins. By default, these pins are configured as GPIO signals (PF3 and PF2). For the Ethernet Controller to drive these signals, they must be reconfigured to their hardware function. See “General-Purpose Input/Outputs (GPIOs)” on page 395 for additional details. The function of these pins is programmable using the **Ethernet MAC LED Encoding (MACLED)** register. Refer to page 946 for additional details on how to program these LED functions.

18.3.2 Internal MII Operation

For the MII management interface to function properly, the MDIO signal must be connected through a 10 kΩ pull-up resistor to the +3.3 V supply. Failure to connect this pull-up resistor prevents management transactions on this internal MII to function. Note that it is possible for data transmission across the MII to still function since the PHY layer auto-negotiates the link parameters by default.

For the MII management interface to function properly, the internal clock must be divided down from the system clock to a frequency no greater than 2.5 MHz. The **Ethernet MAC Management Divider (MACMDV)** register contains the divider used for scaling down the system clock. See page 941 for more details about the use of this register.

18.3.3 PHY Operation

The Physical Layer (PHY) in the Ethernet Controller includes integrated ENDECs, scrambler/descrambler, dual-speed clock recovery, and full-featured auto-negotiation functions. The transmitter includes an on-chip pulse shaper and a low-power line driver. The receiver has an adaptive equalizer and a baseline restoration circuit required for accurate clock and data recovery. The transceiver interfaces to Category-5 unshielded twisted pair (Cat-5 UTP) cabling for 100BASE-TX applications, and Category-3 unshielded twisted pair (Cat-3 UTP) for 10BASE-T applications. The Ethernet Controller is connected to the line media via dual 1:1 isolation transformers. No external filter is required.

18.3.3.1 Clock Selection

The Ethernet Controller can be clocked from an on-chip crystal oscillator which can also be driven by an external oscillator. When using the on-chip crystal oscillator, a 25-MHz crystal should be connected between the XTALPPHY and XTALNPHY pins. Alternatively, an external 25-MHz clock input can be connected to the XTALPPHY pin. In this mode of operation, a crystal is not required and the XTALNPHY pin should be left unconnected. The Ethernet oscillator is powered down when the EPHY0 bit in the **Run Mode Clock Gating Control Register 2 (RCGC2)** register is clear. After setting the EPHY0 bit, software must wait 3.5 ms before accessing any of the MII Management registers. See "Ethernet Controller" on page 1212 for more information regarding the specifications of the Ethernet Controller.

18.3.3.2 Auto-Negotiation

The Ethernet Controller supports the auto-negotiation functions of Clause 28 of the *IEEE 802.3* standard for 10/100 Mbps operation over copper wiring. This function is controlled via register settings. The auto-negotiation function is turned on by default, and the ANEGEN bit in the **Ethernet PHY Management Register 0 - Control (MR0)** is set after reset. Software can disable the auto-negotiation function by clearing the ANEGEN bit. The contents of the **Ethernet PHY Management Register - Auto-Negotiation Advertisement (MR4)** are reflected to the Ethernet Controller's link partner during auto-negotiation via fast-link pulse coding.

Once auto-negotiation is complete, the SPEED bit in the **Ethernet PHY Management Register 31 – PHY Special Control/Status (MR31)** register reflects the actual speed. The AUTODONE bit in **MR31** is set to indicate that auto-negotiation is complete. Setting the RANEG bit in the **MR0** register also causes auto-negotiation to restart.

18.3.3.3 Polarity Correction

The Ethernet Controller is capable of automatic polarity reversal for 10BASE-T and auto-negotiation functions. The XPOL bit in the **Ethernet PHY Management Register 27 –Special Control/Status (MR27)** register is set to indicate the polarity has automatically been reversed.

18.3.3.4 MDI/MDI-X Configuration

The Ethernet Controller supports the MDI/MDI-X configuration as defined in *IEEE 802.3-2002 specification* through software assistance. The MDI/MDI-X configuration eliminates the need for cross-over cables when connecting to another device, such as a hub. Software can implement the MDI/MDI-X configuration using a function outlined by the pseudo code below. This code should be called periodically using one of the available timer resources on the Stellaris microcontroller such as the System Tick Timer or one of the General Purpose timers. The following code refers to the LINK bit in the **Ethernet PHY Management Register 1 - Status (MR1)**, the ENON bit in the **Ethernet PHY Management Register 17 - Mode Control/Status (MR17)**, and the EN bit of the **Ethernet PHY MDIX (MDIX)** register.

```
//
// Entry Point for MDI/MDI-X configuration.
//

//
// Increment the Link Active and Energy Detect Timers using the elapsed time
// since the last call to this function.  If using a periodic timer, the
// elapsed time should be a constant (the programmed period of the timer).
//
Increment Link Active Timer
```

```
Increment Energy Detect Timer

//
if(No Ethernet Link Active)
{
    //
    // If energy has been detected on the link, reset the Energy Detect Timer.
    // If it is a "new" energy detect, reset the link detect timer also.
    //
    if(Ethernet Energy Detected)
    {
        Reset Energy Detect Timer

        if(New Energy Detect)
        {
            Reset Link Detect Timer
        }
    }

    //
    // If the Energy or Link Detect timer has expired, toggle the MDI/MDI-X
    // mode. Typically, the Energy Detect Timer would be ~62ms, while the
    // Link Detect Timer would be ~2s
    //
    if((Energy Detect Timer Expired) or
       (Link Detect Timer Expired))
    {
        Reset Energy Detect Timer

        if(Random Event)
        {
            Reset Link Detect Timer
            Toggle MDI/MDI-X Mode
        }
    }
}

//
// Here, if an Ethernet Link has been detected, simply reset the timers
// for the next time around.
//
else
{
    Reset Link Detect Timer
    Reset Energy Detect Timer
}
```

18.3.3.5 Power Management

The PHY has two power-saving modes:

- Power-Down

- Energy Detect Power-Down

Power-down mode is activated by setting the PWRDN bit in the **MR0** register. When the PHY is in power-down mode, it consumes minimum power. When the PWRDN bit is cleared, the PHY powers up and is automatically reset.

The energy detect power-down mode is activated by setting the EDPD bit in the **MR17** register. In this mode of operation, when no energy is present on the line, the PHY is powered down, except for the managmenet interface, the SQUELCH circuit and the ENERGYON logic. The ENERGYON logic is used to detect the presence of valid energy from 100BASE-T, 10BASE-T, or auto-negotiation signals. While the PHY is powered down, nothing is transmitted. When link pulses or packets are received, the PHY powers-up. The PHY automatically resets itself into the state it had prior to power down and sets the EONIS bit in the **MR29** register. The first and possibly the second packet to activate the ENERGYON mode may be lost.

18.3.4 Interrupts

The Ethernet Controller can generate an interrupt for one or more of the following conditions:

- A frame has been received into an empty RX FIFO
- A frame transmission error has occurred
- A frame has been transmitted successfully
- A frame has been received with inadequate room in the RX FIFO (overrun)
- A frame has been received with one or more error conditions (for example, FCS failed)
- An MII management transaction between the MAC and PHY layers has completed
- One or more of the following PHY layer conditions occurs:
  - Auto-Negotiate Complete
  - Remote Fault
  - Link Partner Acknowledge
  - Parallel Detect Fault
  - Page Received

  Refer to **Ethernet PHY Management Register 29 - Interrupt Source Flags (MR29)** (see page 964) for additional details regarding PHY interrupts.

18.3.5 DMA Operation

The Ethernet peripheral provides request signals to the µDMA controller and has a dedicated channel for transmit and one for receive. The request is a single type for both channels. Burst requests are not supported. The RX channel request is asserted when a packet is received while the TX channel request is asserted when the transmit FIFO becomes empty.

No special configuration is needed to enable the Ethernet peripheral for use with the µDMA controller.

Because the size of a received packet is not known until the header is examined, it is best to set up the initial µDMA transfer to copy the first 4 words including the packet length plus the Ethernet

header from the RX FIFO when the RX request occurs. The µDMA causes an interrupt when this transfer is complete. Upon entering the interrupt handler, the packet length in the FIFO and the Ethernet header are in a buffer and can be examined. Once the packet length is known, then another µDMA transfer can be set up to transfer the remaining received packet payload from the FIFO into a buffer. This transfer should be initiated by software. Another interrupt occurs when this transfer is done.

Even though the TX channel generates a TX empty request, the recommended way to handle µDMA transfers for transmitting packets is to set up the transfer from the buffer containing the packet to the transmit FIFO, and then to initiate the transfer with a software request. An interrupt occurs when this transfer is complete. For both channels, the "auto-request" transfer mode should be used. See “Micro Direct Memory Access (µDMA)” on page 334 for more details about programming the µDMA controller.

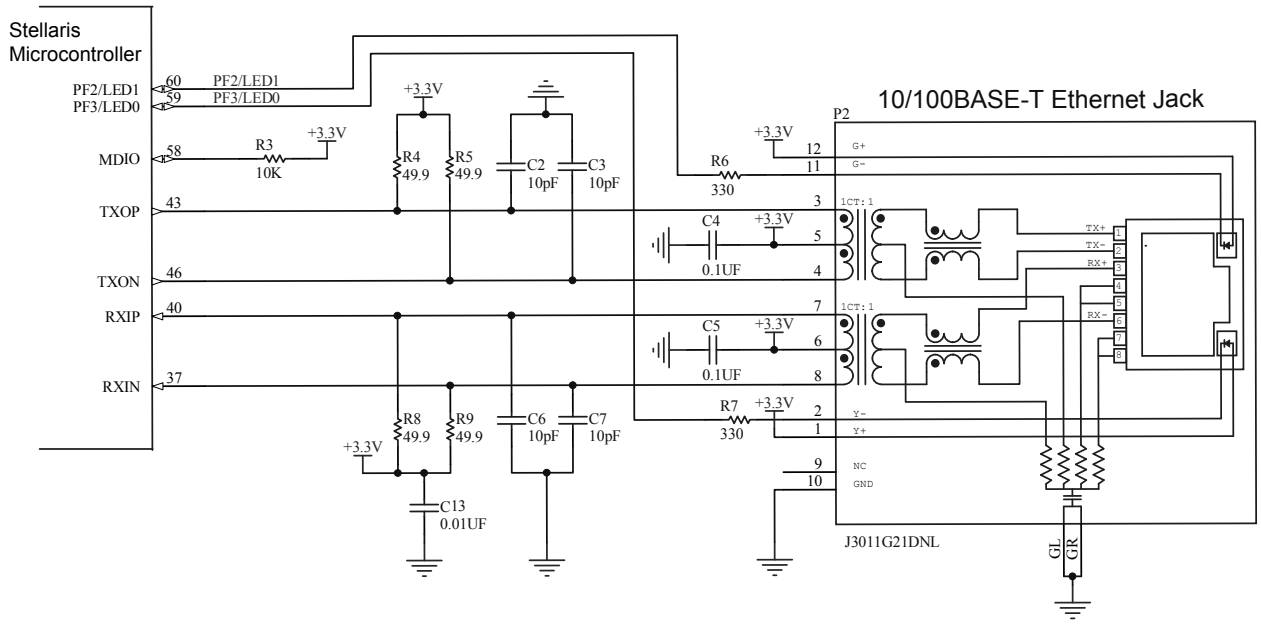
18.4 Initialization and Configuration

The following sections describe the hardware and software configuration required to set up the Ethernet Controller.

18.4.1 Hardware Configuration

Figure 18-4 on page 920 shows the proper method for interfacing the Ethernet Controller to a 10/100BASE-T Ethernet jack.

Figure 18-4. Interface to an Ethernet Jack



The following isolation transformers have been tested and are known to successfully interface to the Ethernet PHY layer.

- Isolation Transformers
  - TDK TLA-6T103
  - TDK TLA-6T118
  - Bel-Fuse S558-5999-46
  - Halo TG22-3506ND

  - Halo TG110-S050
  - PCA EPF8023G
  - Pulse PE-68515
  - Valor ST6118
  - YCL 20PMT04

- Isolation transformers with integrated RJ45 connector
  - TDK TLA-6T704
  - Delta RJS-1A08T089A

- Isolation transformers with integrated RJ45 connector, LEDs and termination resistors
  - Pulse J0011D21B/E
  - Pulse J3011G21DNL

18.4.2 Software Configuration

To use the Ethernet Controller, it must be enabled by setting the EPHY0 and EMAC0 bits in the **RCGC2** register (see page 272). In addition, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register in the System Control module. See page 272. To find out which GPIO port to enable, refer to Table 22-4 on page 1144. Configure the PMCn fields in the **GPIOPCTL** register to assign the Ethernet signals to the appropriate pins. See page 437 and Table 22-5 on page 1151.

The following steps can then be used to configure the Ethernet Controller for basic operation.

1. Program the **MACDIV** register to obtain a 2.5 MHz clock (or less) on the internal MII. Assuming a 20-MHz system clock, the **MACDIV** value should be 0x03 or greater.

2. Program the **MACIA0** and **MACIA1** register for address filtering.

3. Program the **MACTCTL** register for Auto CRC generation, padding, and full-duplex operation using a value of 0x16.

4. Program the **MACRCTL** register to flush the receive FIFO and reject frames with bad FCS using a value of 0x18.

5. Enable both the Transmitter and Receive by setting the LSB in both the **MACTCTL** and **MACRCTL** registers.

6. To transmit a frame, write the frame into the TX FIFO using the **Ethernet MAC Data (MACDATA)** register. Then set the NEWTX bit in the **Ethernet Mac Transmission Request (MACTR)** register to initiate the transmit process. When the NEWTX bit has been cleared, the TX FIFO is available for the next transmit frame.

7. To receive a frame, wait for the NPR field in the **Ethernet MAC Number of Packets (MACNP)** register to be non-zero. Then begin reading the frame from the RX FIFO by using the **MACDATA** register. To ensure that the entire packet is received, either use the DriverLib EthernetPacketGet() API or compare the number of bytes received to the Length field from the frame to determine when the packet has been completely read.

18.5 Register Map

Table 18-4 on page 922 lists the Ethernet MAC and MII Management registers. The MAC register addresses given are relative to the Ethernet base address of 0x4004.8000. The MII Management registers are accessed using the **MACMCTL** register. Note that the Ethernet controller clocks must

be enabled before the registers can be programmed (see page 272). There must be a delay of 3 system clocks after the Ethernet module clock is enabled before any Ethernet module registers are accessed. In addition, the Ethernet oscillator is powered down when the EPHY0 bit in the **Run Mode Clock Gating Control Register 2 (RCGC2)** register is clear. After setting the EPHY0 bit, software must wait 3.5 ms before accessing any of the MII Management registers.

The *IEEE 802.3* standard specifies a register set for controlling and gathering status from the PHY layer. The registers are collectively known as the MII Management registers and are detailed in Section 22.2.4 of the *IEEE 802.3 specification*. Table 18-4 on page 922 also lists these MII Management registers. All addresses given are absolute and are written directly to the REGADR field of the **Ethernet MAC Management Control (MACMCTL)** register. The format of registers 0 to 15 are defined by the IEEE specification and are common to all PHY layer implementations. The only variance allowed is for features that may or may not be supported by a specific PHY implementation. Registers 16 to 31 are vendor-specific registers, used to support features that are specific to a vendor's PHY implementation.

Table 18-4. Ethernet Register Map

Offset	Name	Type	Reset	Description	See page
Ethernet MAC (Ethernet Offset)					
0x000	MACRIS/MACIACK	R/W1C	0x0000.0000	Ethernet MAC Raw Interrupt Status/Acknowledge	924
0x004	MACIM	R/W	0x0000.007F	Ethernet MAC Interrupt Mask	927
0x008	MACRCTL	R/W	0x0000.0008	Ethernet MAC Receive Control	929
0x00C	MACTCTL	R/W	0x0000.0000	Ethernet MAC Transmit Control	931
0x010	MACDATA	R/W	0x0000.0000	Ethernet MAC Data	933
0x014	MACIA0	R/W	0x0000.0000	Ethernet MAC Individual Address 0	935
0x018	MACIA1	R/W	0x0000.0000	Ethernet MAC Individual Address 1	936
0x01C	MACTHR	R/W	0x0000.003F	Ethernet MAC Threshold	937
0x020	MACMCTL	R/W	0x0000.0000	Ethernet MAC Management Control	939
0x024	MACMDV	R/W	0x0000.0080	Ethernet MAC Management Divider	941
0x02C	MACMTXD	R/W	0x0000.0000	Ethernet MAC Management Transmit Data	942
0x030	MACMRXD	R/W	0x0000.0000	Ethernet MAC Management Receive Data	943
0x034	MACNP	RO	0x0000.0000	Ethernet MAC Number of Packets	944
0x038	MACTR	R/W	0x0000.0000	Ethernet MAC Transmission Request	945
0x040	MACLED	R/W	0x0000.0100	Ethernet MAC LED Encoding	946
0x044	MDIX	R/W	0x0000.0000	Ethernet PHY MDIX	948
MII Management (Accessed through the MACMCTL register)					
-	MR0	R/W	0x1000	Ethernet PHY Management Register 0 – Control	949
-	MR1	RO	0x7809	Ethernet PHY Management Register 1 – Status	951
-	MR2	RO	0x0161	Ethernet PHY Management Register 2 – PHY Identifier 1	953

Table 18-4. Ethernet Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
-	MR3	RO	0xB410	Ethernet PHY Management Register 3 – PHY Identifier 2	954
-	MR4	R/W	0x01E1	Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement	955
-	MR5	RO	0x0001	Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability	957
-	MR6	RO	0x0000	Ethernet PHY Management Register 6 – Auto-Negotiation Expansion	959
-	MR16	RO	0x0040	Ethernet PHY Management Register 16 – Vendor-Specific	960
-	MR17	R/W	0x0002	Ethernet PHY Management Register 17 – Mode Control/Status	961
-	MR27	RO	-	Ethernet PHY Management Register 27 – Special Control/Status	963
-	MR29	RO	0x0000	Ethernet PHY Management Register 29 – Interrupt Status	964
-	MR30	R/W	0x0000	Ethernet PHY Management Register 30 – Interrupt Mask	966
-	MR31	R/W	0x0040	Ethernet PHY Management Register 31 – PHY Special Control/Status	968

18.6 Ethernet MAC Register Descriptions

The remainder of this section lists and describes the Ethernet MAC registers, in numerical order by address offset. Also see “MII Management Register Descriptions” on page 948.

Register 1: Ethernet MAC Raw Interrupt Status/Acknowledge (MACRIS/MACIACK), offset 0x000

The **MACRIS/MACIACK** register is the interrupt status and acknowledge register. On a read, this register gives the current status value of the corresponding interrupt prior to masking. On a write, setting any bit clears the corresponding interrupt status bit.

Ethernet MAC Raw Interrupt Status/Acknowledge (MACRIS/MACIACK)

Base 0x4004.8000
Offset 0x000
Type R/W1C, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									PHYINT	MDINT	RXER	FOV	TXEMP	TXER	RXINT
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PHYINT	R/W1C	0	PHY Interrupt Value Description 1 An enabled interrupt in the PHY layer has occurred. **MR29** in the PHY must be read to determine the specific PHY event that triggered this interrupt. 0 No interrupt. This bit is cleared by writing a 1 to it.
5	MDINT	R/W1C	0	MII Transaction Complete Value Description 1 A transaction (read or write) on the MII interface has completed successfully. 0 No interrupt. This bit is cleared by writing a 1 to it.

Bit/Field	Name	Type	Reset	Description
4	RXER	R/W1C	0	Receive Error Value Description 1 An error was encountered on the receiver. The possible errors that can cause this interrupt bit to be set are: ■ A receive error occurs during the reception of a frame (100 Mbps only). ■ The frame is not an integer number of bytes (dribble bits) due to an alignment error. ■ The CRC of the frame does not pass the FCS check. ■ The length/type field is inconsistent with the frame data size when interpreted as a length field. 0 No interrupt. This bit is cleared by writing a 1 to it.
3	FOV	R/W1C	0	FIFO Overrun Value Description 1 An overrun was encountered on the receive FIFO. 0 No interrupt. This bit is cleared by writing a 1 to it.
2	TXEMP	R/W1C	0	Transmit FIFO Empty Value Description 1 The packet was transmitted and that the TX FIFO is empty. 0 No interrupt. This bit is cleared by writing a 1 to it.
1	TXER	R/W1C	0	Transmit Error Value Description 1 An error was encountered on the transmitter. The possible errors that can cause this interrupt bit to be set are: ■ The data length field stored in the TX FIFO exceeds 2032 decimal (buffer length - 16 bytes of header data). The frame is not sent when this error occurs. ■ The retransmission attempts during the backoff process have exceeded the maximum limit of 16 decimal. 0 No interrupt. Writing a 1 to this bit clears it and resets the TX FIFO write pointer.

Bit/Field	Name	Type	Reset	Description
0	RXINT	R/W1C	0	Packet Received

Value	Description
1	At least one packet has been received and is stored in the receiver FIFO.
0	No interrupt.

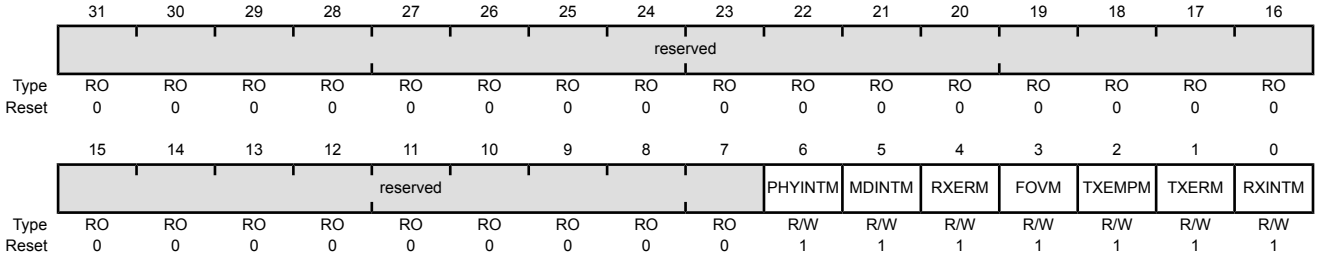
This bit is cleared by writing a 1 to it.

Register 2: Ethernet MAC Interrupt Mask (MACIM), offset 0x004

This register allows software to enable/disable Ethernet MAC interrupts. Clearing a bit disables the interrupt, while setting the bit enables it.

Ethernet MAC Interrupt Mask (MACIM)

Base 0x4004.8000
Offset 0x004
Type R/W, reset 0x0000.007F



Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PHYINTM	R/W	1	Mask PHY Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the PHYINT bit in the **MACRIS/MACIACK** register is set. 0 The PHYINT interrupt is suppressed and not sent to the interrupt controller.
5	MDINTM	R/W	1	Mask MII Transaction Complete Value Description 1 An interrupt is sent to the interrupt controller when the MDINT bit in the **MACRIS/MACIACK** register is set. 0 The MDINT interrupt is suppressed and not sent to the interrupt controller.
4	RXERM	R/W	1	Mask Receive Error Value Description 1 An interrupt is sent to the interrupt controller when the RXER bit in the **MACRIS/MACIACK** register is set. 0 The RXER interrupt is suppressed and not sent to the interrupt controller.

Bit/Field	Name	Type	Reset	Description
3	FOVM	R/W	1	Mask FIFO Overrun Value Description 1 An interrupt is sent to the interrupt controller when the FOV bit in the **MACRIS/MACIACK** register is set. 0 The FOV interrupt is suppressed and not sent to the interrupt controller.
2	TXEMPM	R/W	1	Mask Transmit FIFO Empty Value Description 1 An interrupt is sent to the interrupt controller when the TXEMP bit in the **MACRIS/MACIACK** register is set. 0 The TXEMP interrupt is suppressed and not sent to the interrupt controller.
1	TXERM	R/W	1	Mask Transmit Error Value Description 1 An interrupt is sent to the interrupt controller when the TXER bit in the **MACRIS/MACIACK** register is set. 0 The TXER interrupt is suppressed and not sent to the interrupt controller.
0	RXINTM	R/W	1	Mask Packet Received Value Description 1 An interrupt is sent to the interrupt controller when the RXINT bit in the **MACRIS/MACIACK** register is set. 0 The RXINT interrupt is suppressed and not sent to the interrupt controller.

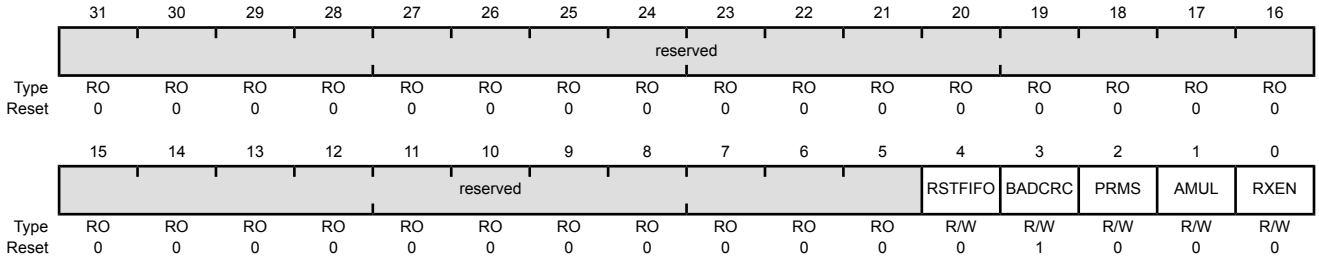
Register 3: Ethernet MAC Receive Control (MACRCTL), offset 0x008

This register configures the receiver and controls the types of frames that are received.

It is important to note that when the receiver is enabled, all valid frames with a broadcast address of FF-FF-FF-FF-FF-FF in the Destination Address field are received and stored in the RX FIFO, even if the AMUL bit is not set.

Ethernet MAC Receive Control (MACRCTL)

Base 0x4004.8000
Offset 0x008
Type R/W, reset 0x0000.0008



Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	RSTFIFO	R/W	0	Clear Receive FIFO Value Description 1 Clear the receive FIFO. The receive FIFO should be cleared when software initialization is performed. 0 No effect. This bit is automatically cleared when read. The receiver should be disabled (RXEN = 0), before a reset is initiated (RSTFIFO = 1). This sequence flushes and resets the RX FIFO.
3	BADCRC	R/W	1	Enable Reject Bad CRC Value Description 1 Enables the rejection of frames with an incorrectly calculated CRC. If a bad CRC is encountered, the RXER bit in the **MACRIS** register is set and the receiver FIFO is reset. 0 Disables the rejection of frames with an incorrectly calculated CRC.
2	PRMS	R/W	0	Enable Promiscuous Mode Value Description 1 Enables Promiscuous mode, which accepts all valid frames, regardless of the specified Destination Address. 0 Disables Promiscuous mode, accepting only frames with the programmed Destination Address.

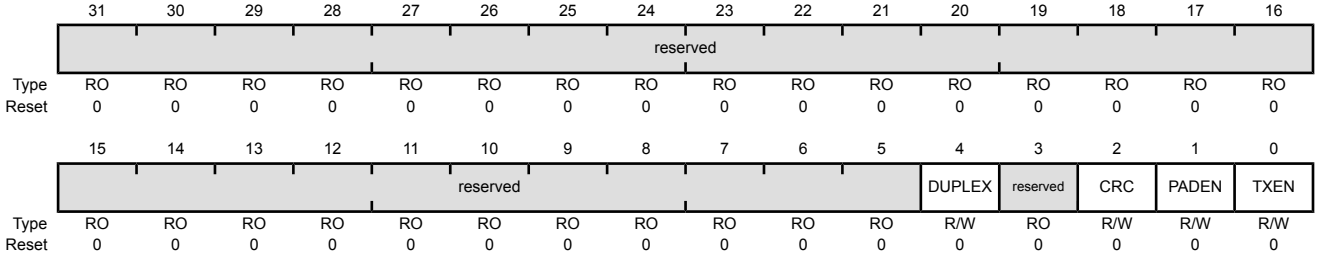
Bit/Field	Name	Type	Reset	Description
1	AMUL	R/W	0	Enable Multicast Frames Value Description 1 Enables the reception of multicast frames. 0 Disables the reception of multicast frames.
0	RXEN	R/W	0	Enable Receiver Value Description 1 Enables the Ethernet receiver. 0 Disables the receiver. All frames are ignored.

Register 4: Ethernet MAC Transmit Control (MACTCTL), offset 0x00C

This register configures the transmitter and controls the frames that are transmitted.

Ethernet MAC Transmit Control (MACTCTL)

Base 0x4004.8000
Offset 0x00C
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	DUPLEX	R/W	0	Enable Duplex Mode Value Description 1 Enables Duplex mode, allowing simultaneous transmission and reception. 0 Disables Duplex mode.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	CRC	R/W	0	Enable CRC Generation Value Description 1 Enables the automatic generation of the CRC and its placement at the end of the packet. 0 The frames placed in the TX FIFO are sent exactly as they are written into the FIFO. Note that this bit should generally be set.
1	PADEN	R/W	0	Enable Packet Padding Value Description 1 Enables the automatic padding of packets that do not meet the minimum frame size. 0 Disables automatic padding. Note that this bit should generally be set.

Bit/Field	Name	Type	Reset	Description
0	TXEN	R/W	0	Enable Transmitter

Value	Description
1	Enables the transmitter.
0	Disables the transmitter.

Register 5: Ethernet MAC Data (MACDATA), offset 0x010

Important: This register is read-sensitive. See the register description for details.

This register enables software to access the TX and RX FIFOs.

Reads from this register return the data stored in the RX FIFO from the location indicated by the read pointer. The read pointer is then auto incremented to the next RX FIFO location. Reading from the RX FIFO when a frame has not been received or is in the process of being received returns indeterminate data and does not increment the read pointer.

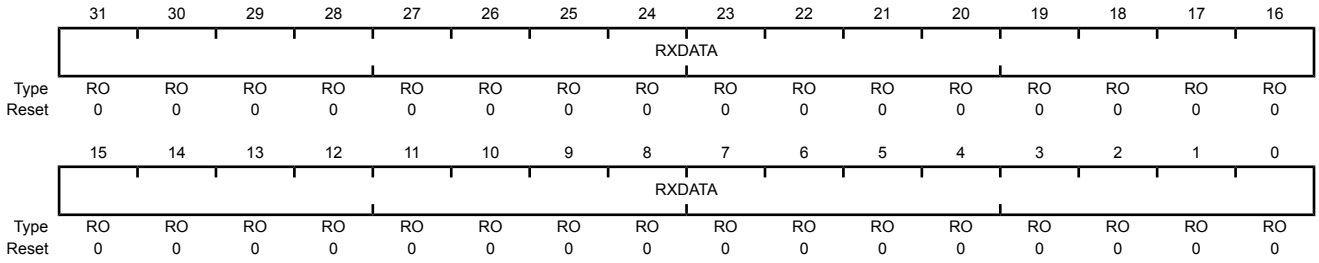
Writes to this register store the data in the TX FIFO at the location indicated by the write pointer. The write pointer is then auto incremented to the next TX FIFO location. Writing more data into the TX FIFO than indicated in the length field results in the data being lost. Writing less data into the TX FIFO than indicated in the length field results in indeterminate data being appended to the end of the frame to achieve the indicated length. Attempting to write the next frame into the TX FIFO before transmission of the first has completed results in the data being lost.

Bytes may not be randomly accessed in either the RX or TX FIFOs. Data must be read from the RX FIFO sequentially and stored in a buffer for further processing. Once a read has been performed, the data in the FIFO cannot be re-read. Data must be written to the TX FIFO sequentially. If an error is made in placing the frame into the TX FIFO, the write pointer can be reset to the start of the TX FIFO by writing the TXER bit of the **MACIACK** register and then the data re-written.

Reads

Ethernet MAC Data (MACDATA)

Base 0x4004.8000
Offset 0x010
Type RO, reset 0x0000.0000

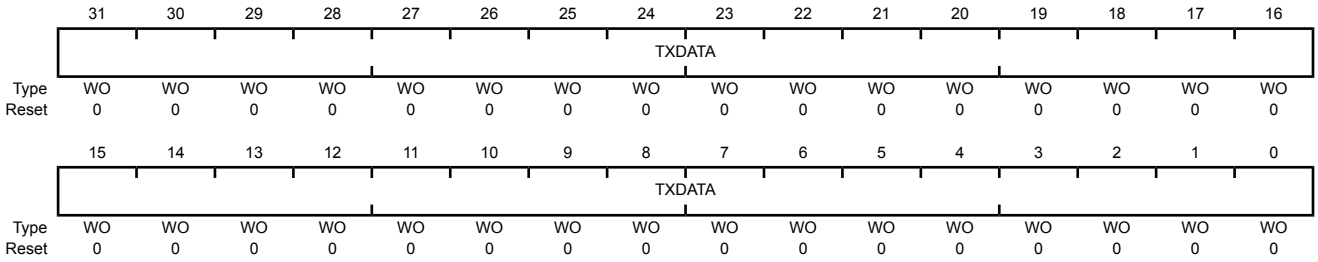


Bit/Field	Name	Type	Reset	Description
31:0	RXDATA	RO	0x0000.0000	Receive FIFO Data The RXDATA bits represent the next word of data stored in the RX FIFO.

Writes

Ethernet MAC Data (MACDATA)

Base 0x4004.8000
Offset 0x010
Type WO, reset 0x0000.0000



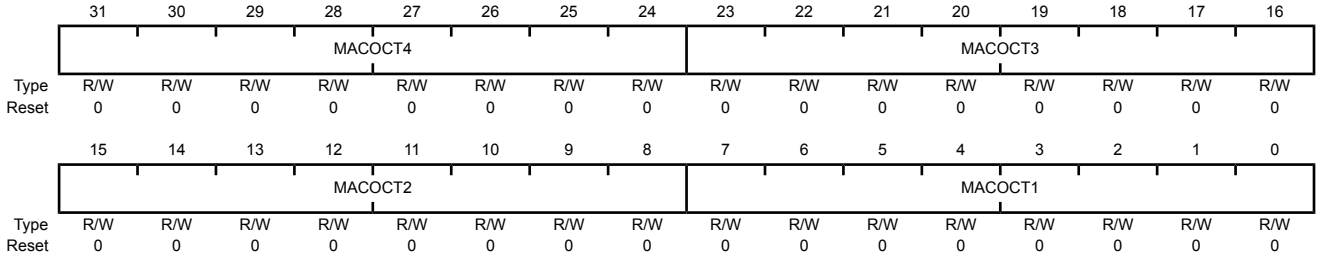
Bit/Field	Name	Type	Reset	Description
31:0	TXDATA	WO	0x0000.0000	Transmit FIFO Data The TXDATA bits represent the next word of data to place in the TX FIFO for transmission.

Register 6: Ethernet MAC Individual Address 0 (MACIA0), offset 0x014

This register enables software to program the first four bytes of the hardware MAC address of the Network Interface Card (NIC). The last two bytes are in **MACIA1**. The 6-byte Individual Address is compared against the incoming Destination Address fields to determine whether the frame should be received.

Ethernet MAC Individual Address 0 (MACIA0)

Base 0x4004.8000
Offset 0x014
Type R/W, reset 0x0000.0000



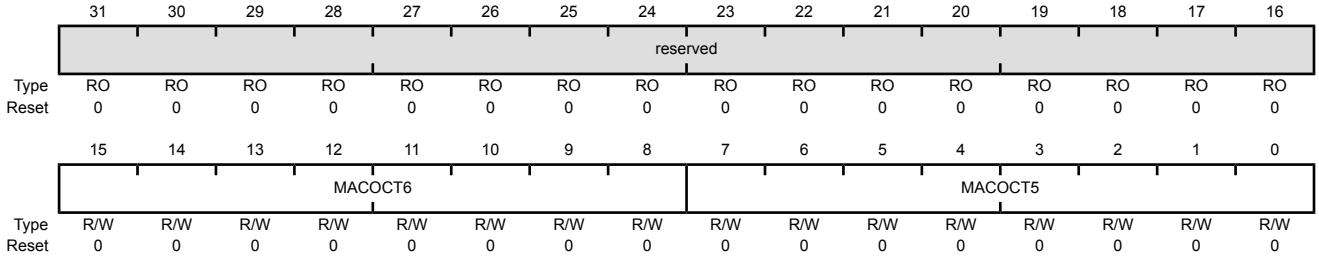
Bit/Field	Name	Type	Reset	Description
31:24	MACOCT4	R/W	0x00	MAC Address Octet 4 The MACOCT4 bits represent the fourth octet of the MAC address used to uniquely identify the Ethernet Controller.
23:16	MACOCT3	R/W	0x00	MAC Address Octet 3 The MACOCT3 bits represent the third octet of the MAC address used to uniquely identify the Ethernet Controller.
15:8	MACOCT2	R/W	0x00	MAC Address Octet 2 The MACOCT2 bits represent the second octet of the MAC address used to uniquely identify the Ethernet Controller.
7:0	MACOCT1	R/W	0x00	MAC Address Octet 1 The MACOCT1 bits represent the first octet of the MAC address used to uniquely identify the Ethernet Controller.

Register 7: Ethernet MAC Individual Address 1 (MACIA1), offset 0x018

This register enables software to program the last two bytes of the hardware MAC address of the Network Interface Card (NIC). The first four bytes are in **MACIA0**. The 6-byte IAR is compared against the incoming Destination Address fields to determine whether the frame should be received.

Ethernet MAC Individual Address 1 (MACIA1)

Base 0x4004.8000
Offset 0x018
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	MACOCT6	R/W	0x00	MAC Address Octet 6 The MACOCT6 bits represent the sixth octet of the MAC address used to uniquely identify each Ethernet Controller.
7:0	MACOCT5	R/W	0x00	MAC Address Octet 5 The MACOCT5 bits represent the fifth octet of the MAC address used to uniquely identify the Ethernet Controller.

Register 8: Ethernet MAC Threshold (MACTHR), offset 0x01C

In order to increase the transmission rate, it is possible to program the Ethernet Controller to begin transmission of the next frame prior to the completion of the transmission of the current frame.

Caution – Extreme care must be used when implementing this function. Software must be able to guarantee that the complete frame is able to be stored in the transmission FIFO prior to the completion of the transmission frame.

This register enables software to set the threshold level at which the transmission of the frame begins. If the THRESH bits are set to 0x3F, which is the reset value, the early transmission feature is disabled, and transmission does not start until the NEWTX bit is set in the **MACTR** register.

Writing the THRESH field to any value besides 0x3F enables the early transmission feature. Once the byte count of data in the TX FIFO reaches the value derived from the THRESH bits as shown below, transmission of the frame begins. When the THRESH field is clear, transmission of the frame begins after 4 bytes (a single write) are stored in the TX FIFO. Each increment of the THRESH bit field waits for an additional 32 bytes of data (eight writes) to be stored in the TX FIFO. Therefore, a value of 0x01 causes the transmitter to wait for 36 bytes of data to be written while a value of 0x02 makes the wait equal to 68 bytes of written data. In general, early transmission starts when:

Number of Bytes ≥ 4 ((*THRESH* x 8) + 1)

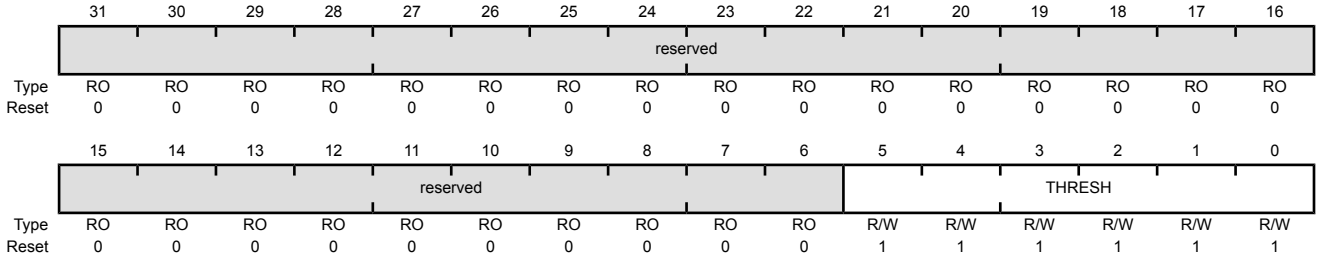
Reaching the threshold level has the same effect as setting the NEWTX bit in the **MACTR** register. Transmission of the frame begins, and then the number of bytes indicated by the Data Length field is transmitted. Because underrun checking is not performed, if any event, such as an interrupt, delays the filling of the FIFO, the tail pointer may reach and pass the write pointer in the TX FIFO. In this event, indeterminate values are transmitted rather than the end of the frame. Therefore, sufficient bus bandwidth for writing to the TX FIFO must be guaranteed by the software.

If a frame smaller than the threshold level must be sent, the NEWTX bit in the **MACTR** register must be set with an explicit write, which initiates the transmission of the frame even though the threshold limit has not been reached.

If the threshold level is set too small, it is possible for the transmitter to underrun. If this occurs, the transmit frame is aborted, and a transmit error occurs. Note that in this case, the TXER bit in the **MACRIS** is not set, meaning that the CPU receives no indication that a transmit error happened.

Ethernet MAC Threshold (MACTHR)

Base 0x4004.8000
Offset 0x01C
Type R/W, reset 0x0000.003F



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
5:0	THRESH	R/W	0x3F	Threshold Value The THRESH bits represent the early transmit threshold. Once the amount of data in the TX FIFO exceeds the value represented by the above equation, transmission of the packet begins.

Register 9: Ethernet MAC Management Control (MACMCTL), offset 0x020

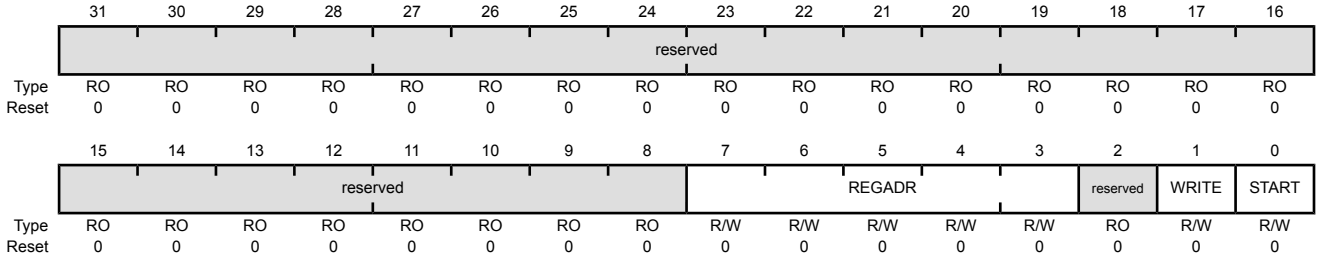
This register enables software to control the transfer of data to and from the MII Management registers in the Ethernet PHY layer. The address, name, type, reset configuration, and functional description of each of these registers can be found in Table 18-4 on page 922 and in "MII Management Register Descriptions" on page 948.

In order to initiate a *read* transaction from the MII Management registers, the WRITE bit must be cleared during the same cycle that the START bit is set.

In order to initiate a *write* transaction to the MII Management registers, the WRITE bit must be set during the same cycle that the START bit is set.

Ethernet MAC Management Control (MACMCTL)

Base 0x4004.8000
Offset 0x020
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:3	REGADR	R/W	0x0	MII Register Address The REGADR bit field represents the MII Management register address for the next MII management interface transaction. Refer to Table 18-4 on page 922 for the PHY register offsets. Note that any address that is not valid in the register map should not be written to, and any data read should be ignored.
2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	WRITE	R/W	0	MII Register Transaction Type Value Description 1 The next operation of the next MII management interface is a write transaction. 0 The next operation of the next MII management interface is a read transaction.

Bit/Field	Name	Type	Reset	Description
0	START	R/W	0	MII Register Transaction Enable

Value	Description
1	The MII register located at REGADR is read (WRITE=0) or written (WRITE=1).
0	No effect.

Register 10: Ethernet MAC Management Divider (MACMDV), offset 0x024

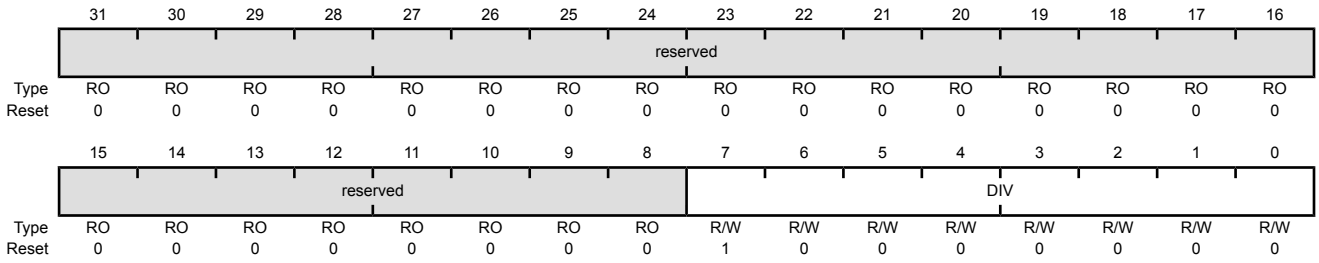
This register enables software to set the clock divider for the Management Data Clock (MDC). This clock is used to synchronize read and write transactions between the system and the MII Management registers. The frequency of the MDC clock can be calculated from the following formula:

$$F_{mdc} = \frac{F_{ipclk}}{2 \times (\mathrm{MACMDV} + 1)}$$

The clock divider must be written with a value that ensures that the MDC clock does not exceed a frequency of 2.5 MHz.

Ethernet MAC Management Divider (MACMDV)

Base 0x4004.8000
Offset 0x024
Type R/W, reset 0x0000.0080



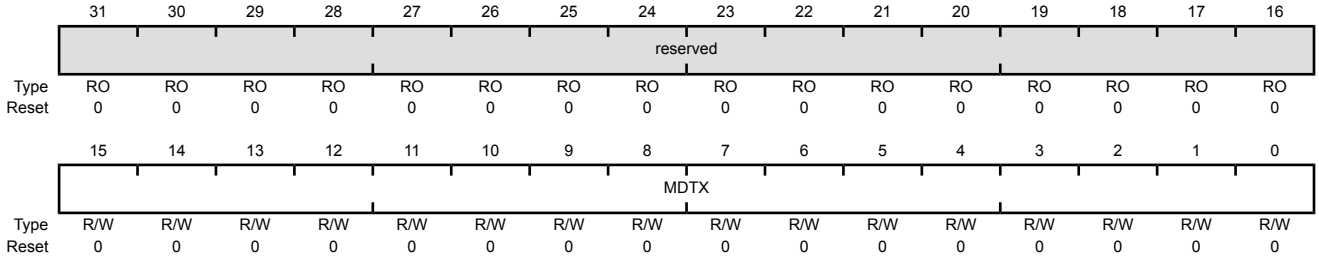
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DIV	R/W	0x80	Clock Divider The DIV bits are used to set the clock divider for the MDC clock used to transmit data between the MAC and PHY layers.

Register 11: Ethernet MAC Management Transmit Data (MACMTXD), offset 0x02C

This register holds the next value to be written to the MII Management registers.

Ethernet MAC Management Transmit Data (MACMTXD)

Base 0x4004.8000
Offset 0x02C
Type R/W, reset 0x0000.0000



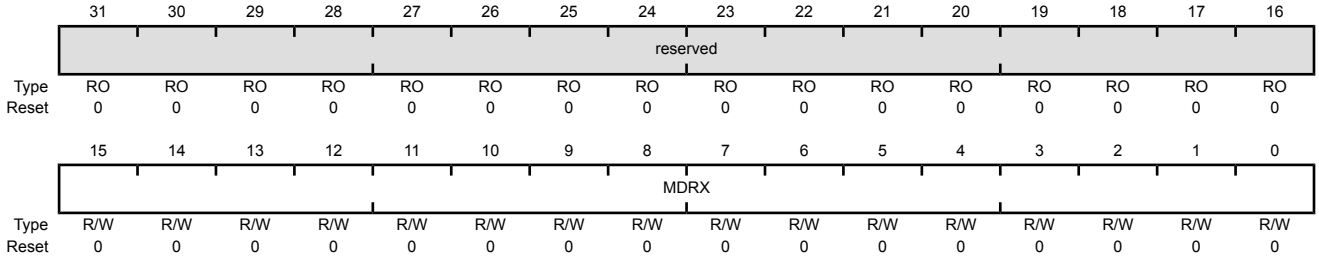
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	MDTX	R/W	0x0000	MII Register Transmit Data The MDTX bits represent the data to be written in the next MII management transaction.

Register 12: Ethernet MAC Management Receive Data (MACMRXD), offset 0x030

This register holds the last value read from the MII Management registers.

Ethernet MAC Management Receive Data (MACMRXD)

Base 0x4004.8000
Offset 0x030
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	MDRX	R/W	0x0000	MII Register Receive Data The MDRX bits represent the data that was read in the previous MII management transaction.

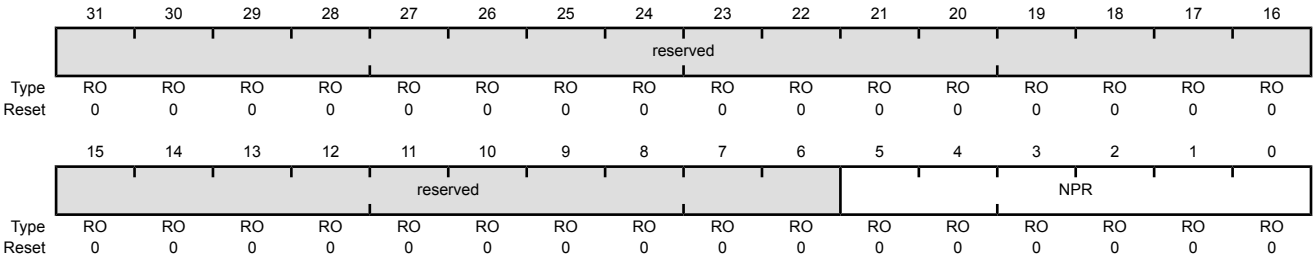
Register 13: Ethernet MAC Number of Packets (MACNP), offset 0x034

This register holds the number of frames that are currently in the RX FIFO. When NPR is 0, there are no frames in the RX FIFO, and the RXINT bit is clear. When NPR is any other value, at least one frame is in the RX FIFO, and the RXINT bit in the **MACRIS** register is set.

Note: The FCS bytes are not included in the NPR value. As a result, the NPR value could be zero before the FCS bytes are read from the FIFO. In addition, a new packet could be received before the NPR value reaches zero. To ensure that the entire packet is received, either use the DriverLib EthernetPacketGet() API or compare the number of bytes received to the Length field from the frame to determine when the packet has been completely read.

Ethernet MAC Number of Packets (MACNP)

Base 0x4004.8000
Offset 0x034
Type RO, reset 0x0000.0000



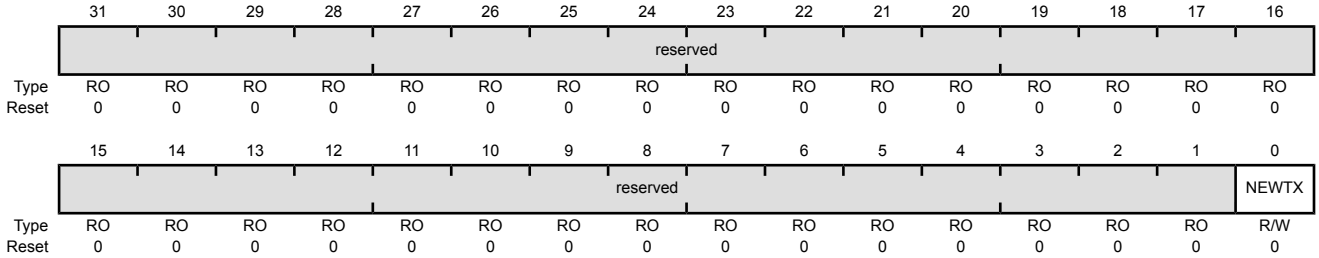
Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	NPR	RO	0x00	Number of Packets in Receive FIFO The NPR bits represent the number of packets stored in the RX FIFO. While the NPR field is greater than 0, the RXINT interrupt in the **MACRIS** register is set.

Register 14: Ethernet MAC Transmission Request (MACTR), offset 0x038

This register enables software to initiate the transmission of the frame currently located in the TX FIFO. Once the frame has been transmitted from the TX FIFO or a transmission error has been encountered, the NEWTX bit is automatically cleared.

Ethernet MAC Transmission Request (MACTR)

Base 0x4004.8000
Offset 0x038
Type R/W, reset 0x0000.0000



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

Value	Description
1	Initiates an Ethernet transmission once the packet has been placed in the TX FIFO.
0	The transmission has completed.

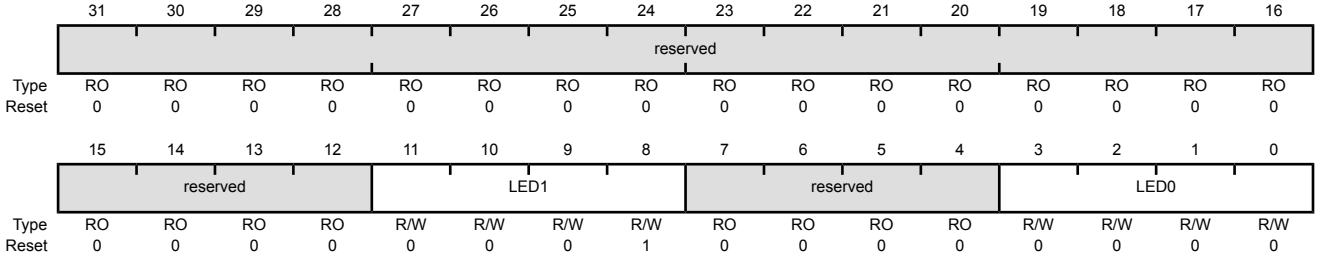
If early transmission is being used (see the **MACTHR** register), this bit does not need to be set.

Register 15: Ethernet MAC LED Encoding (MACLED), offset 0x040

This register enables software to select the source that causes the LED1 and LED0 signal to toggle.

Ethernet MAC LED Encoding (MACLED)

Base 0x4004.8000
Offset 0x040
Type R/W, reset 0x0000.0100



Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11:8	LED1	R/W	0x1	LED1 Source The LED1 field selects the source that toggles the LED1 signal. Value Description 0x0 Link OK 0x1 RX or TX Activity (Default LED1) Note that when RX or TX activity stops, the LED output is extended by 128 ms. 0x2-0x4 Reserved 0x5 100BASE-TX mode 0x6 10BASE-T mode 0x7 Full-Duplex 0x8 Link OK & Blink=RX or TX Activity 0x9-0xF Reserved
7:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
3:0	LED0	R/W	0x0	LED0 Source The LED0 field selects the source that toggles the LED0 signal. Value Description 0x0 Link OK (Default LED0) 0x1 RX or TX Activity Note that when RX or TX activity stops, the LED output is extended by 128 ms. 0x2-0x4 Reserved 0x5 100BASE-TX mode 0x6 10BASE-T mode 0x7 Full-Duplex 0x8 Link OK & Blink=RX or TX Activity 0x9-0xF Reserved

Register 16: Ethernet PHY MDIX (MDIX), offset 0x044

This register enables the transmit and receive lines to be reversed in order to implement the MDI/MDI-X functionality. Software can implement the MDI/MDI-X configuration by using any available timer resource such as SysTick (see “System Timer (SysTick)” on page 108 for more information) to implement this functionality. Once the Ethernet Controller has been configured and enabled, software should check to see if the LINK bit in the **MR1** register has been set within approximately 1 s; if not, set the EN bit of the **MDIX** register to switch the reverse the transmit and receive lines to the PHY layer. Software should check the LINK bit again after approximately another 1 s and if no link has been established, the EN bit should be cleared. Software must continue to change the termination back and forth by setting and clearing the EN bit every 1 s until a link is established.

Ethernet PHY MDIX (MDIX)

Base 0x4004.8000
Offset 0x044
Type R/W, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	EN	R/W	0	MDI/MDI-X Enable Value Description 1 The transmit and receive signals are switched such that data is received on the transmit signals TXOP and TXON; data is transmitted on the receive signals RXIP and RXIN 0 No effect.

18.7 MII Management Register Descriptions

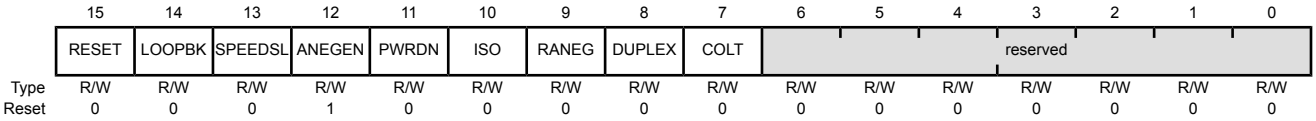
The *IEEE 802.3 standard* specifies a register set for controlling and gathering status from the PHY layer. The registers are collectively known as the MII Management registers. The **Ethernet MAC Management Control (MACMCTL)** register is used to access the MII Management registers, see page 939. All addresses given are absolute. Addresses not listed are reserved; these addresses should not be written to and any data read should be ignored. Also see “Ethernet MAC Register Descriptions” on page 923.

Register 17: Ethernet PHY Management Register 0 – Control (MR0), address 0x00

This register enables software to configure the operation of the PHY layer. The default settings of these registers are designed to initialize the Ethernet Controller to a normal operational mode without configuration.

Ethernet PHY Management Register 0 – Control (MR0)

Base 0x4004.8000
Address 0x00
Type R/W, reset 0x1000



Bit/Field	Name	Type	Reset	Description
15	RESET	R/W	0	Reset Registers Value Description 1 The PHY layer registers reset to their default state and the internal state machines are reinitialized. 0 No effect. Once the reset operation has completed, this bit is automatically cleared by hardware.
14	LOOPBK	R/W	0	Loopback Mode Value Description 1 Enables the Loopback mode of operation. The receiver ignores external inputs and receives the data that is transmitted by the transmitter. 0 No effect.
13	SPEEDSL	R/W	0	Speed Select Value Description 1 Enables the 100 Mbps mode of operation (100BASE-TX). 0 Enables the 10 Mbps mode of operation (10BASE-T).
12	ANEGEN	R/W	1	Auto-Negotiation Enable Value Description 1 Enables the auto-negotiation process. 0 No effect.

Bit/Field	Name	Type	Reset	Description
11	PWRDN	R/W	0	Power Down Value Description 1 The PHY layer is configured to be in a low-power consuming state. All data on the data inputs is ignored. 0 No effect.
10	ISO	R/W	0	Isolate Value Description 1 The transmit and receive data paths are isolated and all data being transmitted and received is ignored. 0 No effect.
9	RANEG	R/W	0	Restart Auto-Negotiation Value Description 1 Restarts the auto-negotiation process. 0 No effect. Once the restart has initiated, this bit is automatically cleared by hardware.
8	DUPLEX	R/W	0	Set Duplex Mode Value Description 1 Enables the Full-Duplex mode of operation. This bit can be set by software in a manual configuration process or by the auto-negotiation process. 0 Enables the Half-Duplex mode of operation. Note that in 10BASE-T half-duplex mode, the transmitted data is looped back on the receive path.
7	COLT	R/W	0	Collision Test Value Description 1 Enables the Collision Test mode of operation. 0 No effect. The COLT bit is set after the initiation of a transmission and is cleared once the transmission is halted.
6:0	reserved	R/W	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. These bits should always be written as zero.

Register 18: Ethernet PHY Management Register 1 – Status (MR1), address 0x01

This register enables software to determine the capabilities of the PHY layer and perform its initialization and operation appropriately.

Ethernet PHY Management Register 1 – Status (MR1)

Base 0x4004.8000
Address 0x01
Type RO, reset 0x7809

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	100X_F	100X_H	10T_F	10T_H	reserved					ANEGC	RFAULT	ANEGA	LINK	JAB	EXTD
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RC	RO	RO	RC	RO
Reset	0	1	1	1	1	0	0	0	0	0	0	0	1	0	0	1

Bit/Field	Name	Type	Reset	Description
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	100X_F	RO	1	100BASE-TX Full-Duplex Mode Value Description 1 The Ethernet Controller is capable of supporting 100BASE-TX Full-Duplex mode. 0 The Ethernet Controller is not capable of supporting 100BASE-TX Full-Duplex mode.
13	100X_H	RO	1	100BASE-TX Half-Duplex Mode Value Description 1 The Ethernet Controller is capable of supporting 100BASE-TX Half-Duplex mode. 0 The Ethernet Controller is not capable of supporting 100BASE-TX Half-Duplex mode.
12	10T_F	RO	1	10BASE-T Full-Duplex Mode Value Description 1 The Ethernet Controller is capable of supporting 10BASE-T Full-Duplex mode. 0 The Ethernet Controller is not capable of supporting 10BASE-T Full-Duplex mode.
11	10T_H	RO	1	10BASE-T Half-Duplex Mode Value Description 1 The Ethernet Controller is capable of supporting 10BASE-T Half-Duplex mode. 0 The Ethernet Controller is not capable of supporting 10BASE-T Half-Duplex mode.

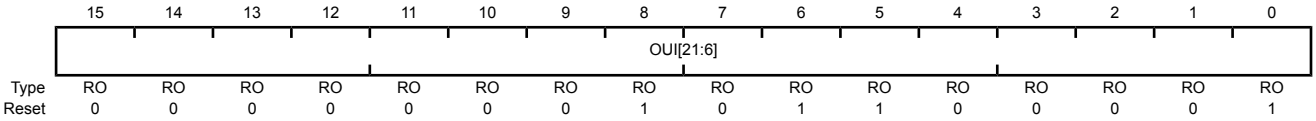
Bit/Field	Name	Type	Reset	Description
10:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	ANEGC	RO	0	Auto-Negotiation Complete Value Description 1 The auto-negotiation process has been completed and that the extended registers defined by the auto-negotiation protocol are valid. 0 The auto-negotiation process is not complete.
4	RFAULT	RC	0	Remote Fault Value Description 1 A remote fault condition has been detected. 0 A remote fault condition has not been detected. This bit remains set until it is read, even if the condition no longer exists.
3	ANEGA	RO	1	Auto-Negotiation Value Description 1 The Ethernet Controller has the ability to perform auto-negotiation. 0 The Ethernet Controller does not have the ability to perform auto-negotiation.
2	LINK	RO	0	Link Made Value Description 1 A valid link has been established by the Ethernet Controller. 0 A valid link has not been established by the Ethernet Controller.
1	JAB	RC	0	Jabber Condition Value Description 1 A jabber condition has been detected by the Ethernet Controller. 0 A jabber condition has not been detected by the Ethernet Controller. This bit remains set until it is read, even if the jabber condition no longer exists.
0	EXTD	RO	1	Extended Capabilities Value Description 1 The Ethernet Controller provides an extended set of capabilities that can be accessed through the extended register set. 0 The Ethernet Controller does not provide extended capabilities.

Register 19: Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2), address 0x02

This register, along with **MR3**, provides a 32-bit value indicating the manufacturer, model, and revision information.

Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2)

Base 0x4004.8000
Address 0x02
Type RO, reset 0x0161



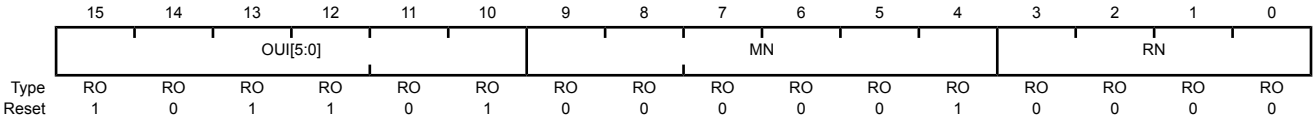
Bit/Field	Name	Type	Reset	Description
15:0	OUI[21:6]	RO	0x0161	Organizationally Unique Identifier[21:6] This field, along with the `OUI[5:0]` field in **MR3**, makes up the Organizationally Unique Identifier indicating the PHY manufacturer.

Register 20: Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3), address 0x03

This register, along with **MR2**, provides a 32-bit value indicating the manufacturer, model, and revision information.

Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3)

Base 0x4004.8000
Address 0x03
Type RO, reset 0xB410



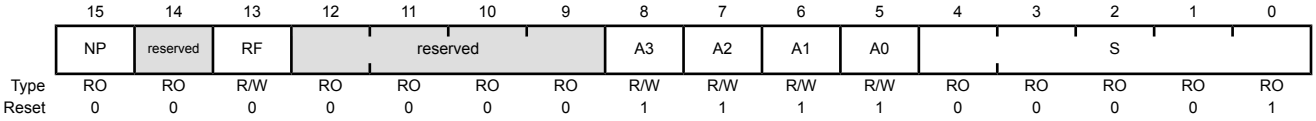
Bit/Field	Name	Type	Reset	Description
15:10	OUI[5:0]	RO	0x2D	Organizationally Unique Identifier[5:0] This field, along with the `OUI[21:6]` field in **MR2**, makes up the Organizationally Unique Identifier indicating the PHY manufacturer.
9:4	MN	RO	0x01	Model Number The `MN` field represents the Model Number of the PHY.
3:0	RN	RO	0x0	Revision Number The `RN` field represents the Revision Number of the PHY implementation.

Register 21: Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4), address 0x04

This register provides the advertised abilities of the Ethernet Controller used during auto-negotiation. Bits 8:5 represent the Technology Ability Field bits. This field can be overwritten by software to auto-negotiate to an alternate common technology. Writing to this register has no effect until auto-negotiation is re-initiated by setting the RANEG bit in the **MR0** register.

Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4)

Base 0x4004.8000
Address 0x04
Type R/W, reset 0x01E1



Bit/Field	Name	Type	Reset	Description
15	NP	RO	0	Next Page Value Description 1 The Ethernet Controller is capable of Next Page exchanges to provide more detailed information on the PHY layer's capabilities. 0 The Ethernet Controller is not capable of Next Page exchanges.
14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	RF	R/W	0	Remote Fault Value Description 1 Indicates to the link partner that a Remote Fault condition has been encountered. 0 No Remote Fault condition has been encountered.
12:9	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	A3	R/W	1	Technology Ability Field [3] Value Description 1 The Ethernet Controller supports the 100Base-TX full-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the **MR0** register. 0 The Ethernet Controller does not support the 100Base-TX full-duplex signaling protocol.

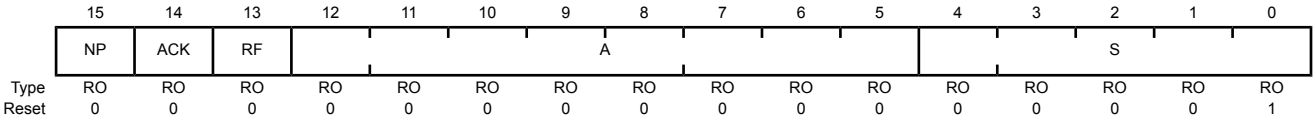
Bit/Field	Name	Type	Reset	Description
7	A2	R/W	1	Technology Ability Field [2] Value Description 1 The Ethernet Controller supports the 100Base-TX half-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the **MR0** register. 0 The Ethernet Controller does not support the 100Base-TX half-duplex signaling protocol.
6	A1	R/W	1	Technology Ability Field [1] Value Description 1 The Ethernet Controller supports the 10BASE-T full-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the **MR0** register. 0 The Ethernet Controller does not support the 10BASE-T full-duplex signaling protocol.
5	A0	R/W	1	Technology Ability Field [0] Value Description 1 The Ethernet Controller supports the 10BASE-T half-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the **MR0** register. 0 The Ethernet Controller does not support the 10BASE-T half-duplex signaling protocol.
4:0	S	RO	0x1	Selector Field This field encodes 32 possible messages for communicating between Ethernet Controllers. This field is hard-coded to 0x01, indicating that the Stellaris Ethernet Controller is *IEEE 802.3* compliant.

Register 22: Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5), address 0x05

This register provides the advertised abilities of the link partner's Ethernet Controller that are received and stored during auto-negotiation.

Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5)

Base 0x4004.8000
Address 0x05
Type RO, reset 0x0001



Bit/Field	Name	Type	Reset	Description
15	NP	RO	0	Next Page Value Description 1 The link partner's Ethernet Controller is capable of Next page exchanges to provide more detailed information on the Ethernet Controller's capabilities. 0 The link partner's Ethernet Controller is not capable of Next page exchanges
14	ACK	RO	0	Acknowledge Value Description 1 The Ethernet Controller has successfully received the link partner's advertised abilities during auto-negotiation. 0 The Ethernet Controller has not received the link partner's advertised abilities during auto-negotiation.
13	RF	RO	0	Remote Fault Value Description 1 The link partner is indicating that a Remote Fault condition has been encountered. 0 The link partner is not indicating that a Remote Fault condition has been encountered.
12:5	A	RO	0x00	Technology Ability Field This field encodes individual technologies that are supported by the Ethernet Controller. See the **MR4** register for definitions. Note that bits [12:9] describe functions that are not implemented on the Stellaris Ethernet Controller. Refer to the IEEE 802.3 standard for definitions.

Bit/Field	Name	Type	Reset	Description
4:0	S	RO	0x01	Selector Field This field encodes possible messages for communicating between Ethernet Controllers.

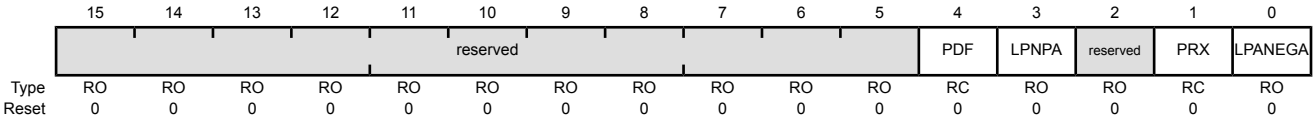
Value	Description
0x00	Reserved
0x01	IEEE Std 802.3
0x02	IEEE Std 802.9 ISLAN-16T
0x03	IEEE Std 802.5
0x04	IEEE Std 1394
0x05–0x1F	Reserved

Register 23: Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6), address 0x06

This register enables software to determine the auto-negotiation and next page capabilities of the Ethernet Controller and the link partner after auto-negotiation.

Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6)

Base 0x4004.8000
Address 0x06
Type RO, reset 0x0000



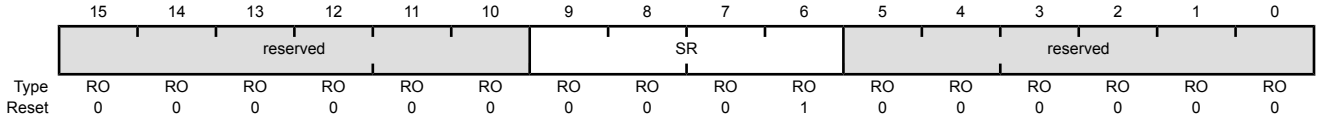
Bit/Field	Name	Type	Reset	Description
15:5	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	PDF	RC	0	Parallel Detection Fault Value Description 1 More than one technology was detected at link up. 0 Only one technology was detected at link up. This bit is automatically cleared when read.
3	LPNPA	RO	0	Link Partner is Next Page Able Value Description 1 The link partner is enabled to support next page. 0 The link partner is not enabled to support next page.
2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PRX	RC	0	New Page Received Value Description 1 A new page has been received from the link partner and stored. 0 A new page has not been received. This bit is automatically cleared when read.
0	LPANEGA	RO	0	Link Partner is Auto-Negotiation Able Value Description 1 The link partner is enabled to support auto-negotiation. 0 The link partner is not enabled to support auto-negotiation.

Register 24: Ethernet PHY Management Register 16 – Vendor-Specific (MR16), address 0x10

This register contains a silicon revision identifier.

Ethernet PHY Management Register 16 – Vendor-Specific (MR16)

Base 0x4004.8000
Address 0x10
Type RO, reset 0x0040



Bit/Field	Name	Type	Reset	Description
15:10	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:6	SR	RO	0x1	Silicon Revision Identifier This field contains the four-bit identifier for the silicon revision.
5:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 25: Ethernet PHY Management Register 17 – Mode Control/Status (MR17), address 0x11

This register provides the means for controlling and observing various PHY layer modes.

Ethernet PHY Management Register 17 – Mode Control/Status (MR17)

Base 0x4004.8000
Address 0x11
Type R/W, reset 0x0002

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	FASTRIP	EDPD	reserved	LSQE	reserved		FASTEST	reserved					FGLS	ENON	reserved
Type	R/W	R/W	R/W	R/W	R/W	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Bit/Field	Name	Type	Reset	Description
15	reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. **Important:** This bit must always be written with a 0 to ensure proper operation.
14	FASTRIP	R/W	0	10-BASE-T Fast Mode Enable Value Description 1 Enables PHYT_10 test mode. 0 No effect.
13	EDPD	R/W	0	Enable Energy Detect Power Down Value Description 1 Enables the Energy Detect Power Down mode. 0 No effect.
12	reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. **Important:** This bit must always be written with a 0 to ensure proper operation.
11	LSQE	R/W	0	Low Squelch Enable Value Description 1 Enables a lower threshold meaning more sensitivity to the signal levels. 0 No effect.
10:9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

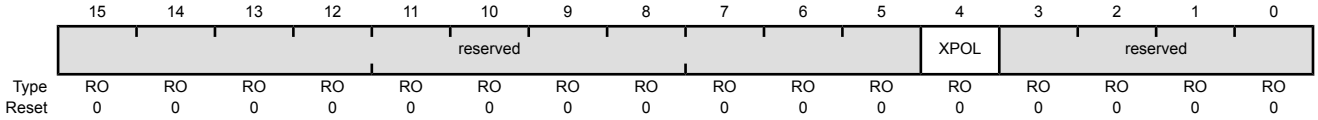
Bit/Field	Name	Type	Reset	Description
8	FASTEST	R/W	0	Auto-Negotiation Test Mode Value Description 1 Enables the Auto-Negotiation Test mode. 0 No effect.
7:3	reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. **Important:** This bit must always be written with a 0 to ensure proper operation.
2	FGLS	R/W	0	Force Good Link Status Value Description 1 Forces the 100BASE-T link to be active. 0 No effect. **Note:** This bit should only be set when testing.
1	ENON	RO	1	Energy On Value Description 1 Energy is detected on the line. 0 Valid energy has not been detected on the line within 256 ms. This bit is set by a hardware reset, but is unaffected by a software reset.
0	reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. **Important:** This bit must always be written with a 0 to ensure proper operation.

Register 26: Ethernet PHY Management Register 27 – Special Control/Status (MR27), address 0x1B

This register shows the status of the 10BASE-T polarity.

Ethernet PHY Management Register 27 – Special Control/Status (MR27)

Base 0x4004.8000
Address 0x1B
Type RO, reset -



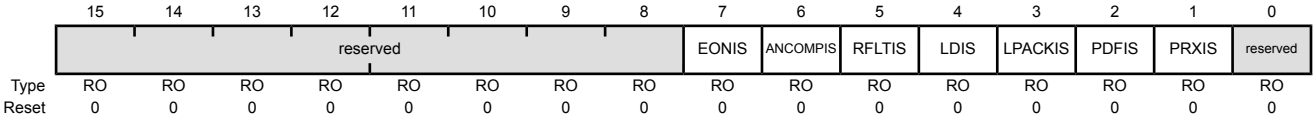
Bit/Field	Name	Type	Reset	Description
15:5	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	XPOL	RO	0	Polarity State of 10 BASE-T Value Description 1 The 10BASE-T is reversed polarity. 0 The 10BASE-T is normal polarity.
3:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 27: Ethernet PHY Management Register 29 – Interrupt Status (MR29), address 0x1D

This register contains information about the source of PHY layer interrupts. Reading this register clears any bits that are set. The PHYINT bit is set in the **MACRIS/MACIACK** register whenever any of the bits in this register are set.

Ethernet PHY Management Register 29 – Interrupt Status (MR29)

Base 0x4004.8000
Address 0x1D
Type RO, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	EONIS	RO	0	ENERGYON Interrupt Value Description 1 An interrupt has been generated due to the ENON bit being set in the **MR17** register. 0 No interrupt. This bit is cleared by reading the value.
6	ANCOMPIS	RO	0	Auto-Negotiation Complete Interrupt Value Description 1 An interrupt has been generated due to the completion of auto negotiation. 0 No interrupt. This bit is cleared by reading the value.
5	RFLTIS	RO	0	Remote Fault Interrupt Value Description 1 An interrupt has been generated due to the detection of a Remote Fault. 0 No interrupt. This bit is cleared by reading the value.
4	LDIS	RO	0	Link Down Interrupt Value Description 1 An interrupt has been generated because the LINK bit in **MR1** is clear. 0 No interrupt. This bit is cleared by reading the value.

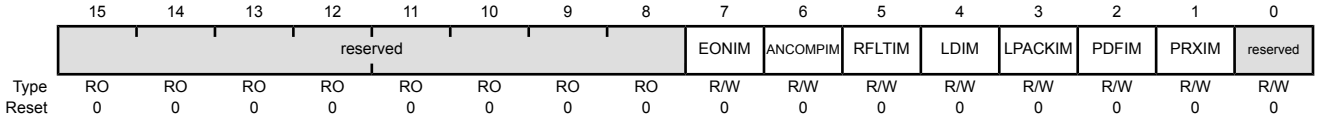
Bit/Field	Name	Type	Reset	Description
3	LPACKIS	RO	0	Auto-Negotiation LP Acknowledge Value Description 1 An interrupt has been generated due to the reception of an acknowledge message from the link partner during auto-negotiation. 0 No interrupt. This bit is cleared by reading the value.
2	PDFIS	RO	0	Parallel Detection Fault Value Description 1 An interrupt has been generated due to the detection of a parallel detection fault during auto negotiation. 0 No interrupt. This bit is cleared by reading the value.
1	PRXIS	RO	0	Auto Negotiation Page Received Value Description 1 An interrupt has been generated due to the reception of an auto negotiation page from the link partner. 0 No interrupt. This bit is cleared by reading the value.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 28: Ethernet PHY Management Register 30 – Interrupt Mask (MR30), address 0x1E

This register enables interrupts to be generated by the various sources of PHY layer interrupts.

Ethernet PHY Management Register 30 – Interrupt Mask (MR30)

Base 0x4004.8000
Address 0x1E
Type R/W, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	EONIM	R/W	0	ENERGYON Interrupt Enabled Value Description 1 An interrupt is sent to the interrupt controller when the EONIS bit in the **MR29** register is set. 0 The EONIS interrupt is suppressed and not sent to the interrupt controller.
6	ANCOMPIM	R/W	0	Auto-Negotiation Complete Interrupt Enabled Value Description 1 An interrupt is sent to the interrupt controller when the ANCOMPIS bit in the **MR29** register is set. 0 The ANCOMPIS interrupt is suppressed and not sent to the interrupt controller.
5	RFLTIM	R/W	0	Remote Fault Interrupt Enabled Value Description 1 An interrupt is sent to the interrupt controller when the RFLTIS bit in the **MR29** register is set. 0 The RFLTIS interrupt is suppressed and not sent to the interrupt controller.
4	LDIM	R/W	0	Link Down Interrupt Enabled Value Description 1 An interrupt is sent to the interrupt controller when the LDIS bit in the **MR29** register is set. 0 The LDIS interrupt is suppressed and not sent to the interrupt controller.

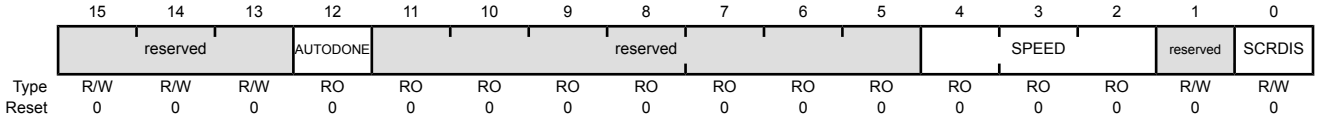
Bit/Field	Name	Type	Reset	Description
3	LPACKIM	R/W	0	Auto-Negotiation LP Acknowledge Enabled Value Description 1 An interrupt is sent to the interrupt controller when the LPACKIS bit in the **MR29** register is set. 0 The LPACKIS interrupt is suppressed and not sent to the interrupt controller.
2	PDFIM	R/W	0	Parallel Detection Fault Enabled Value Description 1 An interrupt is sent to the interrupt controller when the PDFIS bit in the **MR29** register is set. 0 The PDFIS interrupt is suppressed and not sent to the interrupt controller.
1	PRXIM	R/W	0	Auto Negotiation Page Received Enabled Value Description 1 An interrupt is sent to the interrupt controller when the PRXIS bit in the **MR29** register is set. 0 The PRXIS interrupt is suppressed and not sent to the interrupt controller.
0	reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 29: Ethernet PHY Management Register 31 – PHY Special Control/Status (MR31), address 0x1F

This register provides special control and status for the PHY layer.

Ethernet PHY Management Register 31 – PHY Special Control/Status (MR31)

Base 0x4004.8000
Address 0x1F
Type R/W, reset 0x0040



Bit/Field	Name	Type	Reset	Description
15:13	reserved	R/W	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. **Important:** This bit field must always be written with a 0 to ensure proper operation.
12	AUTODONE	RO	0	Auto Negotiation Done Value Description 1 Auto negotiation is complete. 0 Auto negotiation is not complete.
11:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4:2	SPEED	RO	0x0	HCD Speed Value Value Description 0x0 Reserved 0x1 10BASE-T half duplex 0x2 100BASE-T half duplex 0x3-0x4 Reserved 0x5 10BASE-T full duplex 0x6 100BASE-T full duplex 0x7 Reserved
1	reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	SCRDIS	R/W	0	Scramble Disable Value Description 1 Disables data scrambling. 0 Enables data scrambling.

19 Universal Serial Bus (USB) Controller

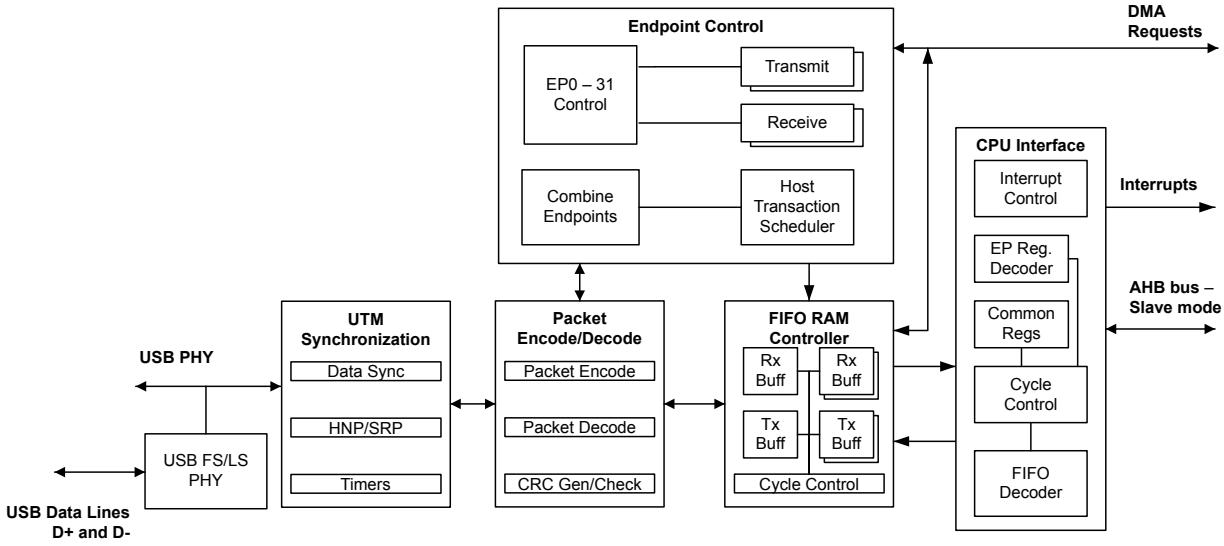
The Stellaris® USB controller operates as a full-speed or low-speed function controller during point-to-point communications with USB Host, Device, or OTG functions. The controller complies with the USB 2.0 standard, which includes SUSPEND and RESUME signaling. 32 endpoints including two hard-wired for control transfers (one endpoint for IN and one endpoint for OUT) plus 30 endpoints defined by firmware along with a dynamic sizable FIFO support multiple packet queueing. µDMA access to the FIFO allows minimal interference from system software. Software-controlled connect and disconnect allows flexibility during USB device start-up. The controller complies with OTG standard's session request protocol (SRP) and host negotiation protocol (HNP).

The Stellaris USB module has the following features:

- Complies with USB-IF certification standards
- USB 2.0 full-speed (12 Mbps) and low-speed (1.5 Mbps) operation with integrated PHY
- 4 transfer types: Control, Interrupt, Bulk, and Isochronous
- 32 endpoints
  - 1 dedicated control IN endpoint and 1 dedicated control OUT endpoint
  - 15 configurable IN endpoints and 15 configurable OUT endpoints
- 4 KB dedicated endpoint memory: one endpoint may be defined for double-buffered 1023-byte isochronous packet size
- VBUS droop and valid ID detection and interrupt
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Separate channels for transmit and receive for up to three IN endpoints and three OUT endpoints
  - Channel requests asserted when FIFO contains required amount of data

19.1 Block Diagram

Figure 19-1. USB Module Block Diagram



19.2 Signal Description

The following table lists the external signals of the USB controller and describes the function of each. Some USB controller signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for these USB signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 419) should be set to choose the USB function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 437) to assign the USB signal to the specified GPIO port pin. The USB0VBUS and USB0ID signals are configured by clearing the appropriate DEN bit in the **GPIO Digital Enable (GPIODEN)** register. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 395. The remaining signals (with the word "fixed" in the Pin Mux/Pin Assignment column) have a fixed pin assignment and function.

Note: When used in OTG mode, USB0VBUS and USB0ID do not require any configuration as they are dedicated pins for the USB controller and directly connect to the USB connector's VBUS and ID signals. If the USB controller is used as either a dedicated Host or Device, the DEVMODOTG and DEVMOD bits in the **USB General-Purpose Control and Status (USBGPCS)** register can be used to connect the USB0VBUS and USB0ID inputs to fixed levels internally, freeing the PB0 and PB1 pins for GPIO use. For proper self-powered Device operation, the VBUS value must still be monitored to assure that if the Host removes VBUS, the self-powered Device disables the D+/D- pull-up resistors. This function can be accomplished by connecting a standard GPIO to VBUS.

Table 19-1. USB Signals (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
USB0DM	70	fixed	I/O	Analog	Bidirectional differential data pin (D- per USB specification) for USB0.

Table 19-1. USB Signals (100LQFP) *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
USB0DP	71	fixed	I/O	Analog	Bidirectional differential data pin (D+ per USB specification) for USB0.
USB0EPEN	19 24 34 72 83	PG0 (7) PC5 (6) PA6 (8) PB2 (8) PH3 (4)	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
USB0ID	66	PB0	I	Analog	This signal senses the state of the USB ID signal. The USB PHY enables an integrated pull-up, and an external element (USB connector) indicates the initial state of the USB controller (pulled down is the A side of the cable and pulled up is the B side).
USB0PFLT	22 23 35 65 74 76 87	PC7 (6) PC6 (7) PA7 (8) PB3 (8) PE0 (9) PH4 (4) PJ1 (9)	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
USB0RBIAS	73	fixed	O	Analog	9.1-kΩ resistor (1% precision) used internally for USB analog circuitry.
USB0VBUS	67	PB1	I/O	Analog	This signal is used during the session request protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing.

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

Table 19-2. USB Signals (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
USB0DM	C11	fixed	I/O	Analog	Bidirectional differential data pin (D- per USB specification) for USB0.
USB0DP	C12	fixed	I/O	Analog	Bidirectional differential data pin (D+ per USB specification) for USB0.
USB0EPEN	K1 M1 L6 A11 D10	PG0 (7) PC5 (6) PA6 (8) PB2 (8) PH3 (4)	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
USB0ID	E12	PB0	I	Analog	This signal senses the state of the USB ID signal. The USB PHY enables an integrated pull-up, and an external element (USB connector) indicates the initial state of the USB controller (pulled down is the A side of the cable and pulled up is the B side).
USB0PFLT	L2 M2 M6 E11 B11 B10 B6	PC7 (6) PC6 (7) PA7 (8) PB3 (8) PE0 (9) PH4 (4) PJ1 (9)	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.

Table 19-2. USB Signals (108BGA) *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
USB0RBIAS	B12	fixed	O	Analog	9.1-kΩ resistor (1% precision) used internally for USB analog circuitry.
USB0VBUS	D12	PB1	I/O	Analog	This signal is used during the session request protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing.

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

19.3 Functional Description

Note: A 9.1-kΩ resistor should be connected between the USB0RBIAS and ground. The 9.1-kΩ resistor should have a 1% tolerance and should be located in close proximity to the USB0RBIAS pin. Power dissipation in the resistor is low, so a chip resistor of any geometry may be used.

The Stellaris USB controller provides full OTG negotiation by supporting both the session request protocol (SRP) and the host negotiation protocol (HNP). The session request protocol allows devices on the B side of a cable to request the A side device turn on VBUS. The host negotiation protocol is used after the initial session request protocol has powered the bus and provides a method to determine which end of the cable will act as the Host controller. When the device is connected to non-OTG peripherals or devices, the controller can detect which cable end was used and provides a register to indicate if the controller should act as the Host or the Device controller. This indication and the mode of operation are handled automatically by the USB controller. This auto-detection allows the system to use a single A/B connector instead of having both A and B connectors in the system and supports full OTG negotiations with other OTG devices.

In addition, the USB controller provides support for connecting to non-OTG peripherals or Host controllers. The USB controller can be configured to act as either a dedicated Host or Device, in which case, the USB0VBUS and USB0ID signals can be used as GPIOs. However, when the USB controller is acting as a self-powered Device, a GPIO input or analog comparator input must be connected to VBUS and configured to generate an interrupt when the VBUS level drops. This interrupt is used to disable the pullup resistor on the USB0DP signal.

Note: When the USB module is in operation, MOSC must be the clock source, either with or without using the PLL, and the system clock must be at least 30 MHz.

19.3.1 Operation as a Device

This section describes the Stellaris USB controller's actions when it is being used as a USB Device. Before the USB controller's operating mode is changed from Device to Host or Host to Device, software must reset the USB controller by setting the USB0 bit in the **Software Reset Control 2 (SRCR2)** register (see page 286). IN endpoints, OUT endpoints, entry into and exit from SUSPEND mode, and recognition of Start of Frame (SOF) are all described.

When in Device mode, IN transactions are controlled by an endpoint’s transmit interface and use the transmit endpoint registers for the given endpoint. OUT transactions are handled with an endpoint's receive interface and use the receive endpoint registers for the given endpoint.

When configuring the size of the FIFOs for endpoints, take into account the maximum packet size for an endpoint.

- **Bulk.** Bulk endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used (described further in the following section).
- **Interrupt.** Interrupt endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used.
- **Isochronous.** Isochronous endpoints are more flexible and can be up to 1023 bytes.
- **Control.** It is also possible to specify a separate control endpoint for a USB Device. However, in most cases the USB Device should use the dedicated control endpoint on the USB controller's endpoint 0.

19.3.1.1 Endpoints

When operating as a Device, the USB controller provides two dedicated control endpoints (IN and OUT) and 30 configurable endpoints (15 IN and 15 OUT) that can be used for communications with a Host controller. The endpoint number and direction associated with an endpoint is directly related to its register designation. For example, when the Host is transmitting to endpoint 1, all configuration and data is in the endpoint 1 transmit register interface.

Endpoint 0 is a dedicated control endpoint used for all control transactions to endpoint 0 during enumeration or when any other control requests are made to endpoint 0. Endpoint 0 uses the first 64 bytes of the USB controller's FIFO RAM as a shared memory for both IN and OUT transactions.

The remaining 30 endpoints can be configured as control, bulk, interrupt, or isochronous endpoints. They should be treated as 15 configurable IN and 15 configurable OUT endpoints. The endpoint pairs are not required to have the same type for their IN and OUT endpoint configuration. For example, the OUT portion of an endpoint pair could be a bulk endpoint, while the IN portion of that endpoint pair could be an interrupt endpoint. The address and size of the FIFOs attached to each endpoint can be modified to fit the application's needs.

19.3.1.2 IN Transactions as a Device

When operating as a USB Device, data for IN transactions is handled through the FIFOs attached to the transmit endpoints. The sizes of the FIFOs for the 15 configurable IN endpoints are determined by the **USB Transmit FIFO Start Address (USBTXFIFOADD)** register. The maximum size of a data packet that may be placed in a transmit endpoint's FIFO for transmission is programmable and is determined by the value written to the **USB Maximum Transmit Data Endpoint n (USBTXMAXPn)** register for that endpoint. The endpoint's FIFO can also be configured to use double-packet or single-packet buffering. When double-packet buffering is enabled, two data packets can be buffered in the FIFO, which also requires that the FIFO is at least two packets in size. When double-packet buffering is disabled, only one packet can be buffered, even if the packet size is less than half the FIFO size.

Note: The maximum packet size set for any endpoint must not exceed the FIFO size. The **USBTXMAXPn** register should not be written to while data is in the FIFO as unexpected results may occur.

Single-Packet Buffering

If the size of the transmit endpoint's FIFO is less than twice the maximum packet size for this endpoint (as set in the **USB Transmit Dynamic FIFO Sizing (USBTXFIFOSZ)** register), only one packet can be buffered in the FIFO and single-packet buffering is required. When each packet is completely loaded into the transmit FIFO, the TXRDY bit in the **USB Transmit Control and Status Endpoint n Low (USBTXCSRLn)** register must be set. If the AUTOSET bit in the **USB Transmit Control and Status Endpoint n High (USBTXCSRHn)** register is set, the TXRDY bit is automatically set when

a maximum-sized packet is loaded into the FIFO. For packet sizes less than the maximum, the TXRDY bit must be set manually. When the TXRDY bit is set, either manually or automatically, the packet is ready to be sent. When the packet has been successfully sent, both TXRDY and FIFONE are cleared, and the appropriate transmit endpoint interrupt signaled. At this point, the next packet can be loaded into the FIFO.

Double-Packet Buffering

If the size of the transmit endpoint's FIFO is at least twice the maximum packet size for this endpoint, two packets can be buffered in the FIFO and double-packet buffering is allowed. As each packet is loaded into the transmit FIFO, the TXRDY bit in the **USBTXCSRLn** register must be set. If the AUTOSET bit in the **USBTXCSRHn** register is set, the TXRDY bit is automatically set when a maximum-sized packet is loaded into the FIFO. For packet sizes less than the maximum, TXRDY must be set manually. When the TXRDY bit is set, either manually or automatically, the packet is ready to be sent. After the first packet is loaded, TXRDY is immediately cleared and an interrupt is generated. A second packet can now be loaded into the transmit FIFO and TXRDY set again (either manually or automatically if the packet is the maximum size). At this point, both packets are ready to be sent. After each packet has been successfully sent, TXRDY is automatically cleared and the appropriate transmit endpoint interrupt signaled to indicate that another packet can now be loaded into the transmit FIFO. The state of the FIFONE bit in the **USBTXCSRLn** register at this point indicates how many packets may be loaded. If the FIFONE bit is set, then another packet is in the FIFO and only one more packet can be loaded. If the FIFONE bit is clear, then no packets are in the FIFO and two more packets can be loaded.

Note: Double-packet buffering is disabled if an endpoint's corresponding EPn bit is set in the **USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFDIS)** register. This bit is set by default, so it must be cleared to enable double-packet buffering.

19.3.1.3 OUT Transactions as a Device

When in Device mode, OUT transactions are handled through the USB controller receive FIFOs. The sizes of the receive FIFOs for the 15 configurable OUT endpoints are determined by the **USB Receive FIFO Start Address (USBRXFIFOADD)** register. The maximum amount of data received by an endpoint in any packet is determined by the value written to the **USB Maximum Receive Data Endpoint n (USBRXMAXPn)** register for that endpoint. When double-packet buffering is enabled, two data packets can be buffered in the FIFO. When double-packet buffering is disabled, only one packet can be buffered even if the packet is less than half the FIFO size.

Note: In all cases, the maximum packet size must not exceed the FIFO size.

Single-Packet Buffering

If the size of the receive endpoint FIFO is less than twice the maximum packet size for an endpoint, only one data packet can be buffered in the FIFO and single-packet buffering is required. When a packet is received and placed in the receive FIFO, the RXRDY and FULL bits in the **USB Receive Control and Status Endpoint n Low (USBRXCSRLn)** register are set and the appropriate receive endpoint is signaled, indicating that a packet can now be unloaded from the FIFO. After the packet has been unloaded, the RXRDY bit must be cleared in order to allow further packets to be received. This action also generates the acknowledge signaling to the Host controller. If the AUTOCL bit in the **USB Receive Control and Status Endpoint n High (USBRXCSRHn)** register is set and a maximum-sized packet is unloaded from the FIFO, the RXRDY and FULL bits are cleared automatically. For packet sizes less than the maximum, RXRDY must be cleared manually.

Double-Packet Buffering

If the size of the receive endpoint FIFO is at least twice the maximum packet size for the endpoint, two data packets can be buffered and double-packet buffering can be used. When the first packet is received and loaded into the receive FIFO, the RXRDY bit in the **USBRXCSRLn** register is set and the appropriate receive endpoint interrupt is signaled to indicate that a packet can now be unloaded from the FIFO.

Note: The FULL bit in **USBRXCSRLn** is not set when the first packet is received. It is only set if a second packet is received and loaded into the receive FIFO.

After each packet has been unloaded, the RXRDY bit must be cleared to allow further packets to be received. If the AUTOCL bit in the **USBRXCSRHn** register is set and a maximum-sized packet is unloaded from the FIFO, the RXRDY bit is cleared automatically. For packet sizes less than the maximum, RXRDY must be cleared manually. If the FULL bit is set when RXRDY is cleared, the USB controller first clears the FULL bit, then sets RXRDY again to indicate that there is another packet waiting in the FIFO to be unloaded.

Note: Double-packet buffering is disabled if an endpoint's corresponding EPn bit is set in the **USB Receive Double Packet Buffer Disable (USBRXDPKTBUFDIS)** register. This bit is set by default, so it must be cleared to enable double-packet buffering.

19.3.1.4 Scheduling

The Device has no control over the scheduling of transactions as scheduling is determined by the Host controller. The Stellaris USB controller can set up a transaction at any time. The USB controller waits for the request from the Host controller and generates an interrupt when the transaction is complete or if it was terminated due to some error. If the Host controller makes a request and the Device controller is not ready, the USB controller sends a busy response (NAK) to all requests until it is ready.

19.3.1.5 Additional Actions

The USB controller responds automatically to certain conditions on the USB bus or actions by the Host controller such as when the USB controller automatically stalls a control transfer or unexpected zero length OUT data packets.

Stalled Control Transfer

The USB controller automatically issues a STALL handshake to a control transfer under the following conditions:

1. The Host sends more data during an OUT data phase of a control transfer than was specified in the Device request during the SETUP phase. This condition is detected by the USB controller when the Host sends an OUT token (instead of an IN token) after the last OUT packet has been unloaded and the DATAEND bit in the **USB Control and Status Endpoint 0 Low (USBCSRL0)** register has been set.

2. The Host requests more data during an IN data phase of a control transfer than was specified in the Device request during the SETUP phase. This condition is detected by the USB controller when the Host sends an IN token (instead of an OUT token) after the CPU has cleared TXRDY and set DATAEND in response to the ACK issued by the Host to what should have been the last packet.

3. The Host sends more than **USBRXMAXPn** bytes of data with an OUT data token.

4. The Host sends more than a zero length data packet for the OUT STATUS phase.

Zero Length OUT Data Packets

A zero-length OUT data packet is used to indicate the end of a control transfer. In normal operation, such packets should only be received after the entire length of the Device request has been transferred.

However, if the Host sends a zero-length OUT data packet before the entire length of Device request has been transferred, it is signaling the premature end of the transfer. In this case, the USB controller automatically flushes any IN token ready for the data phase from the FIFO and sets the DATAEND bit in the **USBCSRL0** register.

Setting the Device Address

When a Host is attempting to enumerate the USB Device, it requests that the Device change its address from zero to some other value. The address is changed by writing the value that the Host requested to the **USB Device Functional Address (USBFADDR)** register. However, care should be taken when writing to **USBFADDR** to avoid changing the address before the transaction is complete. This register should only be set after the SET_ADDRESS command is complete. Like all control transactions, the transaction is only complete after the Device has left the STATUS phase. In the case of a SET_ADDRESS command, the transaction is completed by responding to the IN request from the Host with a zero-byte packet. Once the Device has responded to the IN request, the **USBFADDR** register should be programmed to the new value as soon as possible to avoid missing any new commands sent to the new address.

Note: If the **USBFADDR** register is set to the new value as soon as the Device receives the OUT transaction with the SET_ADDRESS command in the packet, it changes the address during the control transfer. In this case, the Device does not receive the IN request that allows the USB transaction to exit the STATUS phase of the control transfer because it is sent to the old address. As a result, the Host does not get a response to the IN request, and the Host fails to enumerate the Device.

19.3.1.6 Device Mode SUSPEND

When no activity has occurred on the USB bus for 3 ms, the USB controller automatically enters SUSPEND mode. If the SUSPEND interrupt has been enabled in the **USB Interrupt Enable (USBIE)** register, an interrupt is generated at this time. When in SUSPEND mode, the PHY also goes into SUSPEND mode. When RESUME signaling is detected, the USB controller exits SUSPEND mode and takes the PHY out of SUSPEND. If the RESUME interrupt is enabled, an interrupt is generated. The USB controller can also be forced to exit SUSPEND mode by setting the RESUME bit in the **USB Power (USBPOWER)** register. When this bit is set, the USB controller exits SUSPEND mode and drives RESUME signaling onto the bus. The RESUME bit must be cleared after 10 ms (a maximum of 15 ms) to end RESUME signaling.

To meet USB power requirements, the controller can be put into Deep Sleep mode which keeps the controller in a static state.

19.3.1.7 Start-of-Frame

When the USB controller is operating in Device mode, it receives a Start-Of-Frame (SOF) packet from the Host once every millisecond. When the SOF packet is received, the 11-bit frame number contained in the packet is written into the **USB Frame Value (USBFRAME)** register, and an SOF interrupt is also signaled and can be handled by the application. Once the USB controller has started to receive SOF packets, it expects one every millisecond. If no SOF packet is received after 1.00358 ms, the packet is assumed to have been lost, and the **USBFRAME** register is not updated. The USB controller continues and resynchronizes these pulses to the received SOF packets when these packets are successfully received again.

19.3.1.8 USB RESET

When the USB controller is in Device mode and a RESET condition is detected on the USB bus, the USB controller automatically performs the following actions:

- Clears the **USBFADDR** register.
- Clears the **USB Endpoint Index (USBEPIDX)** register.
- Flushes all endpoint FIFOs.
- Clears all control/status registers.
- Enables all endpoint interrupts.
- Generates a RESET interrupt.

When the application software driving the USB controller receives a RESET interrupt, any open pipes are closed and the USB controller waits for bus enumeration to begin.

19.3.1.9 Connect/Disconnect

The USB controller connection to the USB bus is handled by software. The USB PHY can be switched between normal mode and non-driving mode by setting or clearing the SOFTCONN bit of the **USBPOWER** register. When the SOFTCONN bit is set, the PHY is placed in its normal mode, and the USB0DP/USB0DM lines of the USB bus are enabled. At the same time, the USB controller is placed into a state, in which it does not respond to any USB signaling except a USB RESET.

When the SOFTCONN bit is cleared, the PHY is put into non-driving mode, USB0DP and USB0DM are tristated, and the USB controller appears to other devices on the USB bus as if it has been disconnected. The non-driving mode is the default so the USB controller appears disconnected until the SOFTCONN bit has been set. The application software can then choose when to set the PHY into its normal mode. Systems with a lengthy initialization procedure may use this to ensure that initialization is complete, and the system is ready to perform enumeration before connecting to the USB bus. Once the SOFTCONN bit has been set, the USB controller can be disconnected by clearing this bit.

Note: The USB controller does not generate an interrupt when the Device is connected to the Host. However, an interrupt is generated when the Host terminates a session.

19.3.2 Operation as a Host

When the Stellaris USB controller is operating in Host mode, it can either be used for point-to-point communications with another USB device or, when attached to a hub, for communication with multiple devices. Before the USB controller's operating mode is changed from Host to Device or Device to Host, software must reset the USB controller by setting the USB0 bit in the **Software Reset Control 2 (SRCR2)** register (see page 286). Full-speed and low-speed USB devices are supported, both for point-to-point communication and for operation through a hub. The USB controller automatically carries out the necessary transaction translation needed to allow a low-speed or full-speed device to be used with a USB 2.0 hub. Control, bulk, isochronous, and interrupt transactions are supported. This section describes the USB controller's actions when it is being used as a USB Host. Configuration of IN endpoints, OUT endpoints, entry into and exit from SUSPEND mode, and RESET are all described.

When in Host mode, IN transactions are controlled by an endpoint’s receive interface. All IN transactions use the receive endpoint registers and all OUT endpoints use the transmit endpoint

registers for a given endpoint. As in Device mode, the FIFOs for endpoints should take into account the maximum packet size for an endpoint.

- **Bulk.** Bulk endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used (described further in the following section).
- **Interrupt.** Interrupt endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used.
- **Isochronous.** Isochronous endpoints are more flexible and can be up to 1023 bytes.
- **Control.** It is also possible to specify a separate control endpoint to communicate with a Device. However, in most cases the USB controller should use the dedicated control endpoint to communicate with a Device's endpoint 0.

19.3.2.1 Endpoints

The endpoint registers are used to control the USB endpoint interfaces which communicate with Device(s) that are connected. The endpoints consist of a dedicated control IN endpoint, a dedicated control OUT endpoint, 15 configurable OUT endpoints, and 15 configurable IN endpoints.

The dedicated control interface can only be used for control transactions to endpoint 0 of Devices. These control transactions are used during enumeration or other control functions that communicate using endpoint 0 of Devices. This control endpoint shares the first 64 bytes of the USB controller's FIFO RAM for IN and OUT transactions. The remaining IN and OUT interfaces can be configured to communicate with control, bulk, interrupt, or isochronous Device endpoints.

These USB interfaces can be used to simultaneously schedule as many as 15 independent OUT and 15 independent IN transactions to any endpoints on any Device. The IN and OUT controls are paired in three sets of registers. However, they can be configured to communicate with different types of endpoints and different endpoints on Devices. For example, the first pair of endpoint controls can be split so that the OUT portion is communicating with a Device's bulk OUT endpoint 1, while the IN portion is communicating with a Device's interrupt IN endpoint 2.

Before accessing any Device, whether for point-to-point communications or for communications via a hub, the relevant **USB Receive Functional Address Endpoint n (USBRXFUNCADDRn)** or **USB Transmit Functional Address Endpoint n (USBTXFUNCADDRn)** registers must be set for each receive or transmit endpoint to record the address of the Device being accessed.

The USB controller also supports connections to Devices through a USB hub by providing a register that specifies the hub address and port of each USB transfer. The FIFO address and size are customizable and can be specified for each USB IN and OUT transfer. Customization includes allowing one FIFO per transaction, sharing a FIFO across transactions, and allowing for double-buffered FIFOs.

19.3.2.2 IN Transactions as a Host

IN transactions are handled in a similar manner to the way in which OUT transactions are handled when the USB controller is in Device mode except that the transaction first must be initiated by setting the REQPKT bit in the **USBCSRL0** register, indicating to the transaction scheduler that there is an active transaction on this endpoint. The transaction scheduler then sends an IN token to the target Device. When the packet is received and placed in the receive FIFO, the RXRDY bit in the **USBCSRL0** register is set, and the appropriate receive endpoint interrupt is signaled to indicate that a packet can now be unloaded from the FIFO.

When the packet has been unloaded, RXRDY must be cleared. The AUTOCL bit in the **USBRXCSRHn** register can be used to have RXRDY automatically cleared when a maximum-sized packet has been

unloaded from the FIFO. The AUTORQ bit in **USBRXCSRHn** causes the REQPKT bit to be automatically set when the RXRDY bit is cleared. The AUTOCL and AUTORQ bits can be used with µDMA accesses to perform complete bulk transfers without main processor intervention. When the RXRDY bit is cleared, the controller sends an acknowledge to the Device. When there is a known number of packets to be transferred, the **USB Request Packet Count in Block Transfer Endpoint n (USBRQPKTCOUNTn)** register associated with the endpoint should be configured to the number of packets to be transferred. The USB controller decrements the value in the **USBRQPKTCOUNTn** register following each request. When the **USBRQPKTCOUNTn** value decrements to 0, the AUTORQ bit is cleared to prevent any further transactions being attempted. For cases where the size of the transfer is unknown, **USBRQPKTCOUNTn** should be cleared. AUTORQ then remains set until cleared by the reception of a short packet (that is, less than the MAXLOAD value in the **USBRXMAXPn** register) such as may occur at the end of a bulk transfer.

If the Device responds to a bulk or interrupt IN token with a NAK, the USB Host controller keeps retrying the transaction until any NAK Limit that has been set has been reached. If the target Device responds with a STALL, however, the USB Host controller does not retry the transaction but sets the STALLED bit in the **USBCSRL0** register. If the target Device does not respond to the IN token within the required time, or the packet contained a CRC or bit-stuff error, the USB Host controller retries the transaction. If after three attempts the target Device has still not responded, the USB Host controller clears the REQPKT bit and sets the ERROR bit in the **USBCSRL0** register.

19.3.2.3 OUT Transactions as a Host

OUT transactions are handled in a similar manner to the way in which IN transactions are handled when the USB controller is in Device mode. The TXRDY bit in the **USBTXCSRLn** register must be set as each packet is loaded into the transmit FIFO. Again, setting the AUTOSET bit in the **USBTXCSRHn** register automatically sets TXRDY when a maximum-sized packet has been loaded into the FIFO. Furthermore, AUTOSET can be used with the µDMA controller to perform complete bulk transfers without software intervention.

If the target Device responds to the OUT token with a NAK, the USB Host controller keeps retrying the transaction until the NAK Limit that has been set has been reached. However, if the target Device responds with a STALL, the USB controller does not retry the transaction but interrupts the main processor by setting the STALLED bit in the **USBTXCSRLn** register. If the target Device does not respond to the OUT token within the required time, or the packet contained a CRC or bit-stuff error, the USB Host controller retries the transaction. If after three attempts the target Device has still not responded, the USB controller flushes the FIFO and sets the ERROR bit in the **USBTXCSRLn** register.

19.3.2.4 Transaction Scheduling

Scheduling of transactions is handled automatically by the USB Host controller. The Host controller allows configuration of the endpoint communication scheduling based on the type of endpoint transaction. Interrupt transactions can be scheduled to occur in the range of every frame to every 255 frames in 1 frame increments. Bulk endpoints do not allow scheduling parameters, but do allow for a NAK timeout in the event an endpoint on a Device is not responding. Isochronous endpoints can be scheduled from every frame to every 2^{16} frames, in powers of 2.

The USB controller maintains a frame counter. If the target Device is a full-speed device, the USB controller automatically sends an SOF packet at the start of each frame and increments the frame counter. If the target Device is a low-speed device, a *K* state is transmitted on the bus to act as a *keep-alive* to stop the low-speed device from going into SUSPEND mode.

After the SOF packet has been transmitted, the USB Host controller cycles through all the configured endpoints looking for active transactions. An active transaction is defined as a receive endpoint for

which the REQPKT bit is set or a transmit endpoint for which the TXRDY bit and/or the FIFONE bit is set.

An isochronous or interrupt transaction is started if the transaction is found on the first scheduler cycle of a frame and if the interval counter for that endpoint has counted down to zero. As a result, only one interrupt or isochronous transaction occurs per endpoint every n frames, where n is the interval set via the **USB Host Transmit Interval Endpoint n (USBTXINTERVALn)** or **USB Host Receive Interval Endpoint n (USBRXINTERVALn)** register for that endpoint.

An active bulk transaction starts immediately, provided sufficient time is left in the frame to complete the transaction before the next SOF packet is due. If the transaction must be retried (for example, because a NAK was received or the target Device did not respond), then the transaction is not retried until the transaction scheduler has first checked all the other endpoints for active transactions. This process ensures that an endpoint that is sending a lot of NAKs does not block other transactions on the bus. The controller also allows the user to specify a limit to the length of time for NAKs to be received from a target Device before the endpoint times out.

19.3.2.5 USB Hubs

The following setup requirements apply to the USB Host controller only if it is used with a USB hub. When a full- or low-speed Device is connected to the USB controller via a USB 2.0 hub, details of the hub address and the hub port also must be recorded in the corresponding **USB Receive Hub Address Endpoint n (USBRXHUBADDRn)** and **USB Receive Hub Port Endpoint n (USBRXHUBPORTn)** or the **USB Transmit Hub Address Endpoint n (USBTXHUBADDRn)** and **USB Transmit Hub Port Endpoint n (USBTXHUBPORTn)** registers. In addition, the speed at which the Device operates (full or low) must be recorded in the **USB Type Endpoint 0 (USBTYPE0)** (endpoint 0), **USB Host Configure Transmit Type Endpoint n (USBTXTYPEn)**, or **USB Host Configure Receive Type Endpoint n (USBRXTYPEn)** registers for each endpoint that is accessed by the Device.

For hub communications, the settings in these registers record the current allocation of the endpoints to the attached USB Devices. To maximize the number of Devices supported, the USB Host controller allows this allocation to be changed dynamically by simply updating the address and speed information recorded in these registers. Any changes in the allocation of endpoints to Device functions must be made following the completion of any on-going transactions on the endpoints affected.

19.3.2.6 Babble

The USB Host controller does not start a transaction until the bus has been inactive for at least the minimum inter-packet delay. The controller also does not start a transaction unless it can be finished before the end of the frame. If the bus is still active at the end of a frame, then the USB Host controller assumes that the target Device to which it is connected has malfunctioned, and the USB controller suspends all transactions and generates a babble interrupt.

19.3.2.7 Host SUSPEND

If the SUSPEND bit in the **USBPOWER** register is set, the USB Host controller completes the current transaction then stops the transaction scheduler and frame counter. No further transactions are started and no SOF packets are generated.

To exit SUSPEND mode, set the RESUME bit and clear the SUSPEND bit. While the RESUME bit is set, the USB Host controller generates RESUME signaling on the bus. After 20 ms, the RESUME bit must be cleared, at which point the frame counter and transaction scheduler start. The Host supports the detection of a remote wake-up.

19.3.2.8 USB RESET

If the RESET bit in the **USBPOWER** register is set, the USB Host controller generates USB RESET signaling on the bus. The RESET bit must be set for at least 20 ms to ensure correct resetting of the target Device. After the CPU has cleared the bit, the USB Host controller starts its frame counter and transaction scheduler.

19.3.2.9 Connect/Disconnect

A session is started by setting the SESSION bit in the **USB Device Control (USBDEVCTL)** register, enabling the USB controller to wait for a Device to be connected. When a Device is detected, a connect interrupt is generated. The speed of the Device that has been connected can be determined by reading the **USBDEVCTL** register where the FSDEV bit is set for a full-speed Device, and the LSDEV bit is set for a low-speed Device. The USB controller must generate a RESET to the Device, and then the USB Host controller can begin Device enumeration. If the Device is disconnected while a session is in progress, a disconnect interrupt is generated.

19.3.3 OTG Mode

To conserve power, the USB On-The-Go (OTG) supplement allows VBUS to only be powered up when required and to be turned off when the bus is not in use. VBUS is always supplied by the A device on the bus. The USB OTG controller determines whether it is the A device or the B device by sampling the ID input from the PHY. This signal is pulled Low when an A-type plug is sensed (signifying that the USB OTG controller should act as the A device) but taken High when a B-type plug is sensed (signifying that the USB controller is a B device). Note that when switching between OTG A and OTG B, the USB controller retains all register contents.

19.3.3.1 Starting a Session

When the USB OTG controller is ready to start a session, the SESSION bit must be set in the **USBDEVCTL** register. The USB OTG controller then enables ID pin sensing. The ID input is either taken Low if an A-type connection is detected or High if a B-type connection is detected. The DEV bit in the **USBDEVCTL** register is also set to indicate whether the USB OTG controller has adopted the role of the A device or the B device. The USB OTG controller also provides an interrupt to indicate that ID pin sensing has completed and the mode value in the **USBDEVCTL** register is valid. This interrupt is enabled in the **USBIDVIM** register, and the status is checked in the **USBIDVISC** register. As soon as the USB controller has detected that it is on the A side of the cable, it must enable VBUS power within 100ms or the USB controller reverts to Device mode.

If the USB OTG controller is the A device, then the USB OTG controller enters Host mode (the A device is always the default Host), turns on VBUS, and waits for VBUS to go above the VBUS Valid threshold, as indicated by the VBUS bit in the **USBDEVCTL** register going to 0x3. The USB OTG controller then waits for a peripheral to be connected. When a peripheral is detected, a Connect interrupt is signaled and either the FSDEV or LSDEV bit in the **USBDEVCTL** register is set, depending whether a full-speed or a low-speed peripheral is detected. The USB controller then issues a RESET to the connected Device. The SESSION bit in the **USBDEVCTL** register can be cleared to end a session. The USB OTG controller also automatically ends the session if babble is detected or if VBUS drops below session valid.

Note: The USB OTG controller may not remain in Host mode when connected to high-current devices. Some devices draw enough current to momentarily drop VBUS below the VBUS-valid level causing the controller to drop out of Host mode. The only way to get back into Host mode is to allow VBUS to go below the Session End level. In this situation, the device is causing VBUS to drop repeatedly and pull VBUS back low the next time VBUS is enabled.

In addition, the USB OTG controller may not remain in Host mode when a device is told that it can start using it's active configuration. At this point the device starts drawing more current and can also drop VBUS below VBUS valid.

If the USB OTG controller is the B device, then the USB OTG controller requests a session using the session request protocol defined in the USB On-The-Go supplement, that is, it first discharges VBUS. Then when VBUS has gone below the Session End threshold (VBUS bit in the **USBDEVCTL** register goes to 0x0) and the line state has been a single-ended zero for > 2 ms, the USB OTG controller pulses the data line, then pulses VBUS. At the end of the session, the SESSION bit is cleared either by the USB OTG controller or by the application software. The USB OTG controller then causes the PHY to switch out the pull-up resistor on D+, signaling the A device to end the session.

19.3.3.2 Detecting Activity

When the other device of the OTG setup wishes to start a session, it either raises VBUS above the Session Valid threshold if it is the A device, or if it is the B device, it pulses the data line then pulses VBUS. Depending on which of these actions happens, the USB controller can determine whether it is the A device or the B device in the current setup and act accordingly. If VBUS is raised above the Session Valid threshold, then the USB controller is the B device. The USB controller sets the SESSION bit in the **USBDEVCTL** register. When RESET signaling is detected on the bus, a RESET interrupt is signaled, which is interpreted as the start of a session.

The USB controller is in Device mode as the B device is the default mode. At the end of the session, the A device turns off the power to VBUS. When VBUS drops below the Session Valid threshold, the USB controller detects this drop and clears the SESSION bit to indicate that the session has ended, causing a disconnect interrupt to be signaled. If data line and VBUS pulsing is detected, then the USB controller is the A device. The controller generates a SESSION REQUEST interrupt to indicate that the B device is requesting a session. The SESSION bit in the **USBDEVCTL** register must be set to start a session.

19.3.3.3 Host Negotiation

When the USB controller is the A device, ID is Low, and the controller automatically enters Host mode when a session starts. When the USB controller is the B device, ID is High, and the controller automatically enters Device mode when a session starts. However, software can request that the USB controller become the Host by setting the HOSTREQ bit in the **USBDEVCTL** register. This bit can be set either at the same time as requesting a Session Start by setting the SESSION bit in the **USBDEVCTL** register or at any time after a session has started. When the USB controller next enters SUSPEND mode and if the HOSTREQ bit remains set, the controller enters Host mode and begins host negotiation (as specified in the USB On-The-Go supplement) by causing the PHY to disconnect the pull-up resistor on the D+ line, causing the A device to switch to Device mode and connect its own pull-up resistor. When the USB controller detects this, a Connect interrupt is generated and the RESET bit in the **USBPOWER** register is set to begin resetting the A device. The USB controller begins this reset sequence automatically to ensure that RESET is started as required within 1 ms of the A device connecting its pull-up resistor. The main processor should wait at least 20 ms, then clear the RESET bit and enumerate the A device.

When the USB OTG controller B device has finished using the bus, the USB controller goes into SUSPEND mode by setting the SUSPEND bit in the **USBPOWER** register. The A device detects this and either terminates the session or reverts to Host mode. If the A device is USB OTG controller, it generates a Disconnect interrupt.

19.3.4 DMA Operation

The USB peripheral provides an interface connected to the µDMA controller with separate channels for 3 transmit endpoints and 3 receive endpoints. Software selects which endpoints to service with the µDMA channels using the **USB DMA Select (USBDMASEL)** register. The µDMA operation of the USB is enabled through the **USBTXCSRHn** and **USBRXCSRHn** registers, for the TX and RX channels respectively. When µDMA operation is enabled, the USB asserts a µDMA request on the enabled receive or transmit channel when the associated FIFO can transfer data. When either FIFO can transfer data, the burst request for that channel is asserted. The µDMA channel must be configured to operate in Basic mode, and the size of the µDMA transfer must be restricted to whole multiples of the size of the USB FIFO. Both read and write transfers of the USB FIFOs using µDMA must be configured in this manner. For example, if the USB endpoint is configured with a FIFO size of 64 bytes, the µDMA channel can be used to transfer 64 bytes to or from the endpoint FIFO. If the number of bytes to transfer is less than 64, then a programmed I/O method must be used to copy the data to or from the FIFO.

If the DMAMOD bit in the **USBTXCSRHn**/**USBRXCSRHn** register is clear, an interrupt is generated after every packet is transferred, but the µDMA continues transferring data. If the DMAMOD bit is set, an interrupt is generated only when the entire µDMA transfer is complete. The interrupt occurs on the USB interrupt vector. Therefore, if interrupts are used for USB operation and the µDMA is enabled, the USB interrupt handler must be designed to handle the µDMA completion interrupt.

Care must be taken when using the µDMA to unload the receive FIFO as data is read from the receive FIFO in 4 byte chunks regardless of value of the MAXLOAD field in the **USBRXCSRHn** register. The RXRDY bit is cleared as follows.

Table 19-3. Remainder (MAXLOAD/4)

Value	Description
0	MAXLOAD = 64 bytes
1	MAXLOAD = 61 bytes
2	MAXLOAD = 62 bytes
3	MAXLOAD = 63 bytes

Table 19-4. Actual Bytes Read

Value	Description
0	MAXLOAD
1	MAXLOAD+3
2	MAXLOAD+2
3	MAXLOAD+1

Table 19-5. Packet Sizes That Clear RXRDY

Value	Description
0	MAXLOAD, MAXLOAD-1, MAXLOAD-2, MAXLOAD-3
1	MAXLOAD
2	MAXLOAD, MAXLOAD-1
3	MAXLOAD, MAXLOAD-1, MAXLOAD-2

To enable DMA operation for the endpoint receive channel, the DMAEN bit of the **USBRXCSRHn** register should be set. To enable DMA operation for the endpoint transmit channel, the DMAEN bit of the **USBTXCSRHn** register must be set.

See “Micro Direct Memory Access (µDMA)” on page 334 for more details about programming the µDMA controller.

19.4 Initialization and Configuration

To use the USB Controller, the peripheral clock must be enabled via the **RCGC2** register (see page 272). In addition, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register in the System Control module (see page 272). To find out which GPIO port to enable, refer to Table 22-4 on page 1144. Configure the PMCn fields in the **GPIOPCTL** register to assign the USB signals to the appropriate pins (see page 437 and Table 22-5 on page 1151).

The initial configuration in all cases requires that the processor enable the USB controller and USB controller’s physical layer interface (PHY) before setting any registers. The next step is to enable the USB PLL so that the correct clocking is provided to the PHY. To ensure that voltage is not supplied to the bus incorrectly, the external power control signal, USB0EPEN, should be negated on start up by configuring the USB0EPEN and USB0PFLT pins to be controlled by the USB controller and not exhibit their default GPIO behavior.

Note: When used in OTG mode, USB0VBUS and USB0ID do not require any configuration as they are dedicated pins for the USB controller and directly connect to the USB connector's VBUS and ID signals. If the USB controller is used as either a dedicated Host or Device, the DEVMODOTG and DEVMOD bits in the **USB General-Purpose Control and Status (USBGPCS)** register can be used to connect the USB0VBUS and USB0ID inputs to fixed levels internally, freeing the PB0 and PB1 pins for GPIO use. For proper self-powered Device operation, the VBUS value must still be monitored to assure that if the Host removes VBUS, the self-powered Device disables the D+/D- pull-up resistors. This function can be accomplished by connecting a standard GPIO to VBUS.

19.4.1 Pin Configuration

When using the Device controller portion of the USB controller in a system that also provides Host functionality, the power to VBUS must be disabled to allow the external Host controller to supply power. Usually, the USB0EPEN signal is used to control the external regulator and should be negated to avoid having two devices driving the USB0VBUS power pin on the USB connector.

When the USB controller is acting as a Host, it is in control of two signals that are attached to an external voltage supply that provides power to VBUS. The Host controller uses the USB0EPEN signal to enable or disable power to the USB0VBUS pin on the USB connector. An input pin, USB0PFLT, provides feedback when there has been a power fault on VBUS. The USB0PFLT signal can be configured to either automatically negate the USB0EPEN signal to disable power, and/or it can generate an interrupt to the interrupt controller to allow software to handle the power fault condition. The polarity and actions related to both USB0EPEN and USB0PFLT are fully configurable in the USB controller. The controller also provides interrupts on Device insertion and removal to allow the Host controller code to respond to these external events.

19.4.2 Endpoint Configuration

To start communication in Host or Device mode, the endpoint registers must first be configured. In Host mode, this configuration establishes a connection between an endpoint register and an endpoint on a Device. In Device mode, an endpoint must be configured before enumerating to the Host controller.

In both cases, the endpoint 0 configuration is limited because it is a fixed-function, fixed-FIFO-size endpoint. In Device and Host modes, the endpoint requires little setup but does require a software-based state machine to progress through the setup, data, and status phases of a standard control transaction. In Device mode, the configuration of the remaining endpoints is done once before enumerating and then only changed if an alternate configuration is selected by the Host controller. In Host mode, the endpoints must be configured to operate as control, bulk, interrupt or isochronous mode. Once the type of endpoint is configured, a FIFO area must be assigned to each endpoint. In the case of bulk, control and interrupt endpoints, each has a maximum of 64 bytes per transaction. Isochronous endpoints can have packets with up to 1023 bytes per packet. In either mode, the maximum packet size for the given endpoint must be set prior to sending or receiving data.

Configuring each endpoint's FIFO involves reserving a portion of the overall USB FIFO RAM to each endpoint. The total FIFO RAM available is 4 Kbytes with the first 64 bytes reserved for endpoint 0. The endpoint's FIFO must be at least as large as the maximum packet size. The FIFO can also be configured as a double-buffered FIFO so that interrupts occur at the end of each packet and allow filling the other half of the FIFO.

If operating as a Device, the USB Device controller's soft connect must be enabled when the Device is ready to start communications, indicating to the Host controller that the Device is ready to start the enumeration process. If operating as a Host controller, the Device soft connect must be disabled and power must be provided to VBUS via the USB0EPEN signal.

19.5 Register Map

Table 19-6 on page 985 lists the registers. All addresses given are relative to the USB base address of 0x4005.0000. Note that the USB controller clock must be enabled before the registers can be programmed (see page 272). There must be a delay of 3 system clocks after the USB module clock is enabled before any USB module registers are accessed.

Table 19-6. Universal Serial Bus (USB) Controller Register Map

Offset	Name	Type	Reset	Description	See page
0x000	USBFADDR	R/W	0x00	USB Device Functional Address	997
0x001	USBPOWER	R/W	0x20	USB Power	998
0x002	USBTXIS	RO	0x0000	USB Transmit Interrupt Status	1001
0x004	USBRXIS	RO	0x0000	USB Receive Interrupt Status	1003
0x006	USBTXIE	R/W	0xFFFF	USB Transmit Interrupt Enable	1005
0x008	USBRXIE	R/W	0xFFFE	USB Receive Interrupt Enable	1007
0x00A	USBIS	RO	0x00	USB General Interrupt Status	1009
0x00B	USBIE	R/W	0x06	USB Interrupt Enable	1012
0x00C	USBFRAME	RO	0x0000	USB Frame Value	1015
0x00E	USBEPIDX	R/W	0x00	USB Endpoint Index	1016
0x00F	USBTEST	R/W	0x00	USB Test Mode	1017
0x020	USBFIFO0	R/W	0x0000.0000	USB FIFO Endpoint 0	1019
0x024	USBFIFO1	R/W	0x0000.0000	USB FIFO Endpoint 1	1019

Table 19-6. Universal Serial Bus (USB) Controller Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x028	USBFIFO2	R/W	0x0000.0000	USB FIFO Endpoint 2	1019
0x02C	USBFIFO3	R/W	0x0000.0000	USB FIFO Endpoint 3	1019
0x030	USBFIFO4	R/W	0x0000.0000	USB FIFO Endpoint 4	1019
0x034	USBFIFO5	R/W	0x0000.0000	USB FIFO Endpoint 5	1019
0x038	USBFIFO6	R/W	0x0000.0000	USB FIFO Endpoint 6	1019
0x03C	USBFIFO7	R/W	0x0000.0000	USB FIFO Endpoint 7	1019
0x040	USBFIFO8	R/W	0x0000.0000	USB FIFO Endpoint 8	1019
0x044	USBFIFO9	R/W	0x0000.0000	USB FIFO Endpoint 9	1019
0x048	USBFIFO10	R/W	0x0000.0000	USB FIFO Endpoint 10	1019
0x04C	USBFIFO11	R/W	0x0000.0000	USB FIFO Endpoint 11	1019
0x050	USBFIFO12	R/W	0x0000.0000	USB FIFO Endpoint 12	1019
0x054	USBFIFO13	R/W	0x0000.0000	USB FIFO Endpoint 13	1019
0x058	USBFIFO14	R/W	0x0000.0000	USB FIFO Endpoint 14	1019
0x05C	USBFIFO15	R/W	0x0000.0000	USB FIFO Endpoint 15	1019
0x060	USBDEVCTL	R/W	0x80	USB Device Control	1021
0x062	USBTXFIFOSZ	R/W	0x00	USB Transmit Dynamic FIFO Sizing	1023
0x063	USBRXFIFOSZ	R/W	0x00	USB Receive Dynamic FIFO Sizing	1023
0x064	USBTXFIFOADD	R/W	0x0000	USB Transmit FIFO Start Address	1024
0x066	USBRXFIFOADD	R/W	0x0000	USB Receive FIFO Start Address	1024
0x07A	USBCONTIM	R/W	0x5C	USB Connect Timing	1025
0x07B	USBVPLEN	R/W	0x3C	USB OTG VBUS Pulse Timing	1026
0x07D	USBFSEOF	R/W	0x77	USB Full-Speed Last Transaction to End of Frame Timing	1027
0x07E	USBLSEOF	R/W	0x72	USB Low-Speed Last Transaction to End of Frame Timing	1028
0x080	USBTXFUNCADDR0	R/W	0x00	USB Transmit Functional Address Endpoint 0	1029
0x082	USBTXHUBADDR0	R/W	0x00	USB Transmit Hub Address Endpoint 0	1031
0x083	USBTXHUBPORT0	R/W	0x00	USB Transmit Hub Port Endpoint 0	1033
0x088	USBTXFUNCADDR1	R/W	0x00	USB Transmit Functional Address Endpoint 1	1029
0x08A	USBTXHUBADDR1	R/W	0x00	USB Transmit Hub Address Endpoint 1	1031
0x08B	USBTXHUBPORT1	R/W	0x00	USB Transmit Hub Port Endpoint 1	1033
0x08C	USBRXFUNCADDR1	R/W	0x00	USB Receive Functional Address Endpoint 1	1035
0x08E	USBRXHUBADDR1	R/W	0x00	USB Receive Hub Address Endpoint 1	1037

Table 19-6. Universal Serial Bus (USB) Controller Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x08F	USBRXHUBPORT1	R/W	0x00	USB Receive Hub Port Endpoint 1	1039
0x090	USBTXFUNCADDR2	R/W	0x00	USB Transmit Functional Address Endpoint 2	1029
0x092	USBTXHUBADDR2	R/W	0x00	USB Transmit Hub Address Endpoint 2	1031
0x093	USBTXHUBPORT2	R/W	0x00	USB Transmit Hub Port Endpoint 2	1033
0x094	USBRXFUNCADDR2	R/W	0x00	USB Receive Functional Address Endpoint 2	1035
0x096	USBRXHUBADDR2	R/W	0x00	USB Receive Hub Address Endpoint 2	1037
0x097	USBRXHUBPORT2	R/W	0x00	USB Receive Hub Port Endpoint 2	1039
0x098	USBTXFUNCADDR3	R/W	0x00	USB Transmit Functional Address Endpoint 3	1029
0x09A	USBTXHUBADDR3	R/W	0x00	USB Transmit Hub Address Endpoint 3	1031
0x09B	USBTXHUBPORT3	R/W	0x00	USB Transmit Hub Port Endpoint 3	1033
0x09C	USBRXFUNCADDR3	R/W	0x00	USB Receive Functional Address Endpoint 3	1035
0x09E	USBRXHUBADDR3	R/W	0x00	USB Receive Hub Address Endpoint 3	1037
0x09F	USBRXHUBPORT3	R/W	0x00	USB Receive Hub Port Endpoint 3	1039
0x0A0	USBTXFUNCADDR4	R/W	0x00	USB Transmit Functional Address Endpoint 4	1029
0x0A2	USBTXHUBADDR4	R/W	0x00	USB Transmit Hub Address Endpoint 4	1031
0x0A3	USBTXHUBPORT4	R/W	0x00	USB Transmit Hub Port Endpoint 4	1033
0x0A4	USBRXFUNCADDR4	R/W	0x00	USB Receive Functional Address Endpoint 4	1035
0x0A6	USBRXHUBADDR4	R/W	0x00	USB Receive Hub Address Endpoint 4	1037
0x0A7	USBRXHUBPORT4	R/W	0x00	USB Receive Hub Port Endpoint 4	1039
0x0A8	USBTXFUNCADDR5	R/W	0x00	USB Transmit Functional Address Endpoint 5	1029
0x0AA	USBTXHUBADDR5	R/W	0x00	USB Transmit Hub Address Endpoint 5	1031
0x0AB	USBTXHUBPORT5	R/W	0x00	USB Transmit Hub Port Endpoint 5	1033
0x0AC	USBRXFUNCADDR5	R/W	0x00	USB Receive Functional Address Endpoint 5	1035
0x0AE	USBRXHUBADDR5	R/W	0x00	USB Receive Hub Address Endpoint 5	1037
0x0AF	USBRXHUBPORT5	R/W	0x00	USB Receive Hub Port Endpoint 5	1039
0x0B0	USBTXFUNCADDR6	R/W	0x00	USB Transmit Functional Address Endpoint 6	1029
0x0B2	USBTXHUBADDR6	R/W	0x00	USB Transmit Hub Address Endpoint 6	1031
0x0B3	USBTXHUBPORT6	R/W	0x00	USB Transmit Hub Port Endpoint 6	1033
0x0B4	USBRXFUNCADDR6	R/W	0x00	USB Receive Functional Address Endpoint 6	1035
0x0B6	USBRXHUBADDR6	R/W	0x00	USB Receive Hub Address Endpoint 6	1037
0x0B7	USBRXHUBPORT6	R/W	0x00	USB Receive Hub Port Endpoint 6	1039
0x0B8	USBTXFUNCADDR7	R/W	0x00	USB Transmit Functional Address Endpoint 7	1029

Table 19-6. Universal Serial Bus (USB) Controller Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x0BA	USBTXHUBADDR7	R/W	0x00	USB Transmit Hub Address Endpoint 7	1031
0x0BB	USBTXHUBPORT7	R/W	0x00	USB Transmit Hub Port Endpoint 7	1033
0x0BC	USBRXFUNCADDR7	R/W	0x00	USB Receive Functional Address Endpoint 7	1035
0x0BE	USBRXHUBADDR7	R/W	0x00	USB Receive Hub Address Endpoint 7	1037
0x0BF	USBRXHUBPORT7	R/W	0x00	USB Receive Hub Port Endpoint 7	1039
0x0C0	USBTXFUNCADDR8	R/W	0x00	USB Transmit Functional Address Endpoint 8	1029
0x0C2	USBTXHUBADDR8	R/W	0x00	USB Transmit Hub Address Endpoint 8	1031
0x0C3	USBTXHUBPORT8	R/W	0x00	USB Transmit Hub Port Endpoint 8	1033
0x0C4	USBRXFUNCADDR8	R/W	0x00	USB Receive Functional Address Endpoint 8	1035
0x0C6	USBRXHUBADDR8	R/W	0x00	USB Receive Hub Address Endpoint 8	1037
0x0C7	USBRXHUBPORT8	R/W	0x00	USB Receive Hub Port Endpoint 8	1039
0x0C8	USBTXFUNCADDR9	R/W	0x00	USB Transmit Functional Address Endpoint 9	1029
0x0CA	USBTXHUBADDR9	R/W	0x00	USB Transmit Hub Address Endpoint 9	1031
0x0CB	USBTXHUBPORT9	R/W	0x00	USB Transmit Hub Port Endpoint 9	1033
0x0CC	USBRXFUNCADDR9	R/W	0x00	USB Receive Functional Address Endpoint 9	1035
0x0CE	USBRXHUBADDR9	R/W	0x00	USB Receive Hub Address Endpoint 9	1037
0x0CF	USBRXHUBPORT9	R/W	0x00	USB Receive Hub Port Endpoint 9	1039
0x0D0	USBTXFUNCADDR10	R/W	0x00	USB Transmit Functional Address Endpoint 10	1029
0x0D2	USBTXHUBADDR10	R/W	0x00	USB Transmit Hub Address Endpoint 10	1031
0x0D3	USBTXHUBPORT10	R/W	0x00	USB Transmit Hub Port Endpoint 10	1033
0x0D4	USBRXFUNCADDR10	R/W	0x00	USB Receive Functional Address Endpoint 10	1035
0x0D6	USBRXHUBADDR10	R/W	0x00	USB Receive Hub Address Endpoint 10	1037
0x0D7	USBRXHUBPORT10	R/W	0x00	USB Receive Hub Port Endpoint 10	1039
0x0D8	USBTXFUNCADDR11	R/W	0x00	USB Transmit Functional Address Endpoint 11	1029
0x0DA	USBTXHUBADDR11	R/W	0x00	USB Transmit Hub Address Endpoint 11	1031
0x0DB	USBTXHUBPORT11	R/W	0x00	USB Transmit Hub Port Endpoint 11	1033
0x0DC	USBRXFUNCADDR11	R/W	0x00	USB Receive Functional Address Endpoint 11	1035
0x0DE	USBRXHUBADDR11	R/W	0x00	USB Receive Hub Address Endpoint 11	1037
0x0DF	USBRXHUBPORT11	R/W	0x00	USB Receive Hub Port Endpoint 11	1039
0x0E0	USBTXFUNCADDR12	R/W	0x00	USB Transmit Functional Address Endpoint 12	1029
0x0E2	USBTXHUBADDR12	R/W	0x00	USB Transmit Hub Address Endpoint 12	1031
0x0E3	USBTXHUBPORT12	R/W	0x00	USB Transmit Hub Port Endpoint 12	1033

Table 19-6. Universal Serial Bus (USB) Controller Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x0E4	USBRXFUNCADDR12	R/W	0x00	USB Receive Functional Address Endpoint 12	1035
0x0E6	USBRXHUBADDR12	R/W	0x00	USB Receive Hub Address Endpoint 12	1037
0x0E7	USBRXHUBPORT12	R/W	0x00	USB Receive Hub Port Endpoint 12	1039
0x0E8	USBTXFUNCADDR13	R/W	0x00	USB Transmit Functional Address Endpoint 13	1029
0x0EA	USBTXHUBADDR13	R/W	0x00	USB Transmit Hub Address Endpoint 13	1031
0x0EB	USBTXHUBPORT13	R/W	0x00	USB Transmit Hub Port Endpoint 13	1033
0x0EC	USBRXFUNCADDR13	R/W	0x00	USB Receive Functional Address Endpoint 13	1035
0x0EE	USBRXHUBADDR13	R/W	0x00	USB Receive Hub Address Endpoint 13	1037
0x0EF	USBRXHUBPORT13	R/W	0x00	USB Receive Hub Port Endpoint 13	1039
0x0F0	USBTXFUNCADDR14	R/W	0x00	USB Transmit Functional Address Endpoint 14	1029
0x0F2	USBTXHUBADDR14	R/W	0x00	USB Transmit Hub Address Endpoint 14	1031
0x0F3	USBTXHUBPORT14	R/W	0x00	USB Transmit Hub Port Endpoint 14	1033
0x0F4	USBRXFUNCADDR14	R/W	0x00	USB Receive Functional Address Endpoint 14	1035
0x0F6	USBRXHUBADDR14	R/W	0x00	USB Receive Hub Address Endpoint 14	1037
0x0F7	USBRXHUBPORT14	R/W	0x00	USB Receive Hub Port Endpoint 14	1039
0x0F8	USBTXFUNCADDR15	R/W	0x00	USB Transmit Functional Address Endpoint 15	1029
0x0FA	USBTXHUBADDR15	R/W	0x00	USB Transmit Hub Address Endpoint 15	1031
0x0FB	USBTXHUBPORT15	R/W	0x00	USB Transmit Hub Port Endpoint 15	1033
0x0FC	USBRXFUNCADDR15	R/W	0x00	USB Receive Functional Address Endpoint 15	1035
0x0FE	USBRXHUBADDR15	R/W	0x00	USB Receive Hub Address Endpoint 15	1037
0x0FF	USBRXHUBPORT15	R/W	0x00	USB Receive Hub Port Endpoint 15	1039
0x102	USBCSRL0	W1C	0x00	USB Control and Status Endpoint 0 Low	1043
0x103	USBCSRH0	W1C	0x00	USB Control and Status Endpoint 0 High	1047
0x108	USBCOUNT0	RO	0x00	USB Receive Byte Count Endpoint 0	1049
0x10A	USBTYPE0	R/W	0x00	USB Type Endpoint 0	1050
0x10B	USBNAKLMT	R/W	0x00	USB NAK Limit	1051
0x110	USBTXMAXP1	R/W	0x0000	USB Maximum Transmit Data Endpoint 1	1041
0x112	USBTXCSRL1	R/W	0x00	USB Transmit Control and Status Endpoint 1 Low	1052
0x113	USBTXCSRH1	R/W	0x00	USB Transmit Control and Status Endpoint 1 High	1057
0x114	USBRXMAXP1	R/W	0x0000	USB Maximum Receive Data Endpoint 1	1061
0x116	USBRXCSRL1	R/W	0x00	USB Receive Control and Status Endpoint 1 Low	1063
0x117	USBRXCSRH1	R/W	0x00	USB Receive Control and Status Endpoint 1 High	1068

Table 19-6. Universal Serial Bus (USB) Controller Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x118	USBRXCOUNT1	RO	0x0000	USB Receive Byte Count Endpoint 1	1073
0x11A	USBTXTYPE1	R/W	0x00	USB Host Transmit Configure Type Endpoint 1	1075
0x11B	USBTXINTERVAL1	R/W	0x00	USB Host Transmit Interval Endpoint 1	1077
0x11C	USBRXTYPE1	R/W	0x00	USB Host Configure Receive Type Endpoint 1	1079
0x11D	USBRXINTERVAL1	R/W	0x00	USB Host Receive Polling Interval Endpoint 1	1081
0x120	USBTXMAXP2	R/W	0x0000	USB Maximum Transmit Data Endpoint 2	1041
0x122	USBTXCSRL2	R/W	0x00	USB Transmit Control and Status Endpoint 2 Low	1052
0x123	USBTXCSRH2	R/W	0x00	USB Transmit Control and Status Endpoint 2 High	1057
0x124	USBRXMAXP2	R/W	0x0000	USB Maximum Receive Data Endpoint 2	1061
0x126	USBRXCSRL2	R/W	0x00	USB Receive Control and Status Endpoint 2 Low	1063
0x127	USBRXCSRH2	R/W	0x00	USB Receive Control and Status Endpoint 2 High	1068
0x128	USBRXCOUNT2	RO	0x0000	USB Receive Byte Count Endpoint 2	1073
0x12A	USBTXTYPE2	R/W	0x00	USB Host Transmit Configure Type Endpoint 2	1075
0x12B	USBTXINTERVAL2	R/W	0x00	USB Host Transmit Interval Endpoint 2	1077
0x12C	USBRXTYPE2	R/W	0x00	USB Host Configure Receive Type Endpoint 2	1079
0x12D	USBRXINTERVAL2	R/W	0x00	USB Host Receive Polling Interval Endpoint 2	1081
0x130	USBTXMAXP3	R/W	0x0000	USB Maximum Transmit Data Endpoint 3	1041
0x132	USBTXCSRL3	R/W	0x00	USB Transmit Control and Status Endpoint 3 Low	1052
0x133	USBTXCSRH3	R/W	0x00	USB Transmit Control and Status Endpoint 3 High	1057
0x134	USBRXMAXP3	R/W	0x0000	USB Maximum Receive Data Endpoint 3	1061
0x136	USBRXCSRL3	R/W	0x00	USB Receive Control and Status Endpoint 3 Low	1063
0x137	USBRXCSRH3	R/W	0x00	USB Receive Control and Status Endpoint 3 High	1068
0x138	USBRXCOUNT3	RO	0x0000	USB Receive Byte Count Endpoint 3	1073
0x13A	USBTXTYPE3	R/W	0x00	USB Host Transmit Configure Type Endpoint 3	1075
0x13B	USBTXINTERVAL3	R/W	0x00	USB Host Transmit Interval Endpoint 3	1077
0x13C	USBRXTYPE3	R/W	0x00	USB Host Configure Receive Type Endpoint 3	1079
0x13D	USBRXINTERVAL3	R/W	0x00	USB Host Receive Polling Interval Endpoint 3	1081
0x140	USBTXMAXP4	R/W	0x0000	USB Maximum Transmit Data Endpoint 4	1041
0x142	USBTXCSRL4	R/W	0x00	USB Transmit Control and Status Endpoint 4 Low	1052
0x143	USBTXCSRH4	R/W	0x00	USB Transmit Control and Status Endpoint 4 High	1057
0x144	USBRXMAXP4	R/W	0x0000	USB Maximum Receive Data Endpoint 4	1061
0x146	USBRXCSRL4	R/W	0x00	USB Receive Control and Status Endpoint 4 Low	1063

Table 19-6. Universal Serial Bus (USB) Controller Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x147	USBRXCSRH4	R/W	0x00	USB Receive Control and Status Endpoint 4 High	1068
0x148	USBRXCOUNT4	RO	0x0000	USB Receive Byte Count Endpoint 4	1073
0x14A	USBTXTYPE4	R/W	0x00	USB Host Transmit Configure Type Endpoint 4	1075
0x14B	USBTXINTERVAL4	R/W	0x00	USB Host Transmit Interval Endpoint 4	1077
0x14C	USBRXTYPE4	R/W	0x00	USB Host Configure Receive Type Endpoint 4	1079
0x14D	USBRXINTERVAL4	R/W	0x00	USB Host Receive Polling Interval Endpoint 4	1081
0x150	USBTXMAXP5	R/W	0x0000	USB Maximum Transmit Data Endpoint 5	1041
0x152	USBTXCSRL5	R/W	0x00	USB Transmit Control and Status Endpoint 5 Low	1052
0x153	USBTXCSRH5	R/W	0x00	USB Transmit Control and Status Endpoint 5 High	1057
0x154	USBRXMAXP5	R/W	0x0000	USB Maximum Receive Data Endpoint 5	1061
0x156	USBRXCSRL5	R/W	0x00	USB Receive Control and Status Endpoint 5 Low	1063
0x157	USBRXCSRH5	R/W	0x00	USB Receive Control and Status Endpoint 5 High	1068
0x158	USBRXCOUNT5	RO	0x0000	USB Receive Byte Count Endpoint 5	1073
0x15A	USBTXTYPE5	R/W	0x00	USB Host Transmit Configure Type Endpoint 5	1075
0x15B	USBTXINTERVAL5	R/W	0x00	USB Host Transmit Interval Endpoint 5	1077
0x15C	USBRXTYPE5	R/W	0x00	USB Host Configure Receive Type Endpoint 5	1079
0x15D	USBRXINTERVAL5	R/W	0x00	USB Host Receive Polling Interval Endpoint 5	1081
0x160	USBTXMAXP6	R/W	0x0000	USB Maximum Transmit Data Endpoint 6	1041
0x162	USBTXCSRL6	R/W	0x00	USB Transmit Control and Status Endpoint 6 Low	1052
0x163	USBTXCSRH6	R/W	0x00	USB Transmit Control and Status Endpoint 6 High	1057
0x164	USBRXMAXP6	R/W	0x0000	USB Maximum Receive Data Endpoint 6	1061
0x166	USBRXCSRL6	R/W	0x00	USB Receive Control and Status Endpoint 6 Low	1063
0x167	USBRXCSRH6	R/W	0x00	USB Receive Control and Status Endpoint 6 High	1068
0x168	USBRXCOUNT6	RO	0x0000	USB Receive Byte Count Endpoint 6	1073
0x16A	USBTXTYPE6	R/W	0x00	USB Host Transmit Configure Type Endpoint 6	1075
0x16B	USBTXINTERVAL6	R/W	0x00	USB Host Transmit Interval Endpoint 6	1077
0x16C	USBRXTYPE6	R/W	0x00	USB Host Configure Receive Type Endpoint 6	1079
0x16D	USBRXINTERVAL6	R/W	0x00	USB Host Receive Polling Interval Endpoint 6	1081
0x170	USBTXMAXP7	R/W	0x0000	USB Maximum Transmit Data Endpoint 7	1041
0x172	USBTXCSRL7	R/W	0x00	USB Transmit Control and Status Endpoint 7 Low	1052
0x173	USBTXCSRH7	R/W	0x00	USB Transmit Control and Status Endpoint 7 High	1057
0x174	USBRXMAXP7	R/W	0x0000	USB Maximum Receive Data Endpoint 7	1061

Table 19-6. Universal Serial Bus (USB) Controller Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x176	USBRXCSRL7	R/W	0x00	USB Receive Control and Status Endpoint 7 Low	1063
0x177	USBRXCSRH7	R/W	0x00	USB Receive Control and Status Endpoint 7 High	1068
0x178	USBRXCOUNT7	RO	0x0000	USB Receive Byte Count Endpoint 7	1073
0x17A	USBTXTYPE7	R/W	0x00	USB Host Transmit Configure Type Endpoint 7	1075
0x17B	USBTXINTERVAL7	R/W	0x00	USB Host Transmit Interval Endpoint 7	1077
0x17C	USBRXTYPE7	R/W	0x00	USB Host Configure Receive Type Endpoint 7	1079
0x17D	USBRXINTERVAL7	R/W	0x00	USB Host Receive Polling Interval Endpoint 7	1081
0x180	USBTXMAXP8	R/W	0x0000	USB Maximum Transmit Data Endpoint 8	1041
0x182	USBTXCSRL8	R/W	0x00	USB Transmit Control and Status Endpoint 8 Low	1052
0x183	USBTXCSRH8	R/W	0x00	USB Transmit Control and Status Endpoint 8 High	1057
0x184	USBRXMAXP8	R/W	0x0000	USB Maximum Receive Data Endpoint 8	1061
0x186	USBRXCSRL8	R/W	0x00	USB Receive Control and Status Endpoint 8 Low	1063
0x187	USBRXCSRH8	R/W	0x00	USB Receive Control and Status Endpoint 8 High	1068
0x188	USBRXCOUNT8	RO	0x0000	USB Receive Byte Count Endpoint 8	1073
0x18A	USBTXTYPE8	R/W	0x00	USB Host Transmit Configure Type Endpoint 8	1075
0x18B	USBTXINTERVAL8	R/W	0x00	USB Host Transmit Interval Endpoint 8	1077
0x18C	USBRXTYPE8	R/W	0x00	USB Host Configure Receive Type Endpoint 8	1079
0x18D	USBRXINTERVAL8	R/W	0x00	USB Host Receive Polling Interval Endpoint 8	1081
0x190	USBTXMAXP9	R/W	0x0000	USB Maximum Transmit Data Endpoint 9	1041
0x192	USBTXCSRL9	R/W	0x00	USB Transmit Control and Status Endpoint 9 Low	1052
0x193	USBTXCSRH9	R/W	0x00	USB Transmit Control and Status Endpoint 9 High	1057
0x194	USBRXMAXP9	R/W	0x0000	USB Maximum Receive Data Endpoint 9	1061
0x196	USBRXCSRL9	R/W	0x00	USB Receive Control and Status Endpoint 9 Low	1063
0x197	USBRXCSRH9	R/W	0x00	USB Receive Control and Status Endpoint 9 High	1068
0x198	USBRXCOUNT9	RO	0x0000	USB Receive Byte Count Endpoint 9	1073
0x19A	USBTXTYPE9	R/W	0x00	USB Host Transmit Configure Type Endpoint 9	1075
0x19B	USBTXINTERVAL9	R/W	0x00	USB Host Transmit Interval Endpoint 9	1077
0x19C	USBRXTYPE9	R/W	0x00	USB Host Configure Receive Type Endpoint 9	1079
0x19D	USBRXINTERVAL9	R/W	0x00	USB Host Receive Polling Interval Endpoint 9	1081
0x1A0	USBTXMAXP10	R/W	0x0000	USB Maximum Transmit Data Endpoint 10	1041
0x1A2	USBTXCSRL10	R/W	0x00	USB Transmit Control and Status Endpoint 10 Low	1052
0x1A3	USBTXCSRH10	R/W	0x00	USB Transmit Control and Status Endpoint 10 High	1057

Table 19-6. Universal Serial Bus (USB) Controller Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x1A4	USBRXMAXP10	R/W	0x0000	USB Maximum Receive Data Endpoint 10	1061
0x1A6	USBRXCSRL10	R/W	0x00	USB Receive Control and Status Endpoint 10 Low	1063
0x1A7	USBRXCSRH10	R/W	0x00	USB Receive Control and Status Endpoint 10 High	1068
0x1A8	USBRXCOUNT10	RO	0x0000	USB Receive Byte Count Endpoint 10	1073
0x1AA	USBTXTYPE10	R/W	0x00	USB Host Transmit Configure Type Endpoint 10	1075
0x1AB	USBTXINTERVAL10	R/W	0x00	USB Host Transmit Interval Endpoint 10	1077
0x1AC	USBRXTYPE10	R/W	0x00	USB Host Configure Receive Type Endpoint 10	1079
0x1AD	USBRXINTERVAL10	R/W	0x00	USB Host Receive Polling Interval Endpoint 10	1081
0x1B0	USBTXMAXP11	R/W	0x0000	USB Maximum Transmit Data Endpoint 11	1041
0x1B2	USBTXCSRL11	R/W	0x00	USB Transmit Control and Status Endpoint 11 Low	1052
0x1B3	USBTXCSRH11	R/W	0x00	USB Transmit Control and Status Endpoint 11 High	1057
0x1B4	USBRXMAXP11	R/W	0x0000	USB Maximum Receive Data Endpoint 11	1061
0x1B6	USBRXCSRL11	R/W	0x00	USB Receive Control and Status Endpoint 11 Low	1063
0x1B7	USBRXCSRH11	R/W	0x00	USB Receive Control and Status Endpoint 11 High	1068
0x1B8	USBRXCOUNT11	RO	0x0000	USB Receive Byte Count Endpoint 11	1073
0x1BA	USBTXTYPE11	R/W	0x00	USB Host Transmit Configure Type Endpoint 11	1075
0x1BB	USBTXINTERVAL11	R/W	0x00	USB Host Transmit Interval Endpoint 11	1077
0x1BC	USBRXTYPE11	R/W	0x00	USB Host Configure Receive Type Endpoint 11	1079
0x1BD	USBRXINTERVAL11	R/W	0x00	USB Host Receive Polling Interval Endpoint 11	1081
0x1C0	USBTXMAXP12	R/W	0x0000	USB Maximum Transmit Data Endpoint 12	1041
0x1C2	USBTXCSRL12	R/W	0x00	USB Transmit Control and Status Endpoint 12 Low	1052
0x1C3	USBTXCSRH12	R/W	0x00	USB Transmit Control and Status Endpoint 12 High	1057
0x1C4	USBRXMAXP12	R/W	0x0000	USB Maximum Receive Data Endpoint 12	1061
0x1C6	USBRXCSRL12	R/W	0x00	USB Receive Control and Status Endpoint 12 Low	1063
0x1C7	USBRXCSRH12	R/W	0x00	USB Receive Control and Status Endpoint 12 High	1068
0x1C8	USBRXCOUNT12	RO	0x0000	USB Receive Byte Count Endpoint 12	1073
0x1CA	USBTXTYPE12	R/W	0x00	USB Host Transmit Configure Type Endpoint 12	1075
0x1CB	USBTXINTERVAL12	R/W	0x00	USB Host Transmit Interval Endpoint 12	1077
0x1CC	USBRXTYPE12	R/W	0x00	USB Host Configure Receive Type Endpoint 12	1079
0x1CD	USBRXINTERVAL12	R/W	0x00	USB Host Receive Polling Interval Endpoint 12	1081
0x1D0	USBTXMAXP13	R/W	0x0000	USB Maximum Transmit Data Endpoint 13	1041
0x1D2	USBTXCSRL13	R/W	0x00	USB Transmit Control and Status Endpoint 13 Low	1052

Table 19-6. Universal Serial Bus (USB) Controller Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x1D3	USBTXCSRH13	R/W	0x00	USB Transmit Control and Status Endpoint 13 High	1057
0x1D4	USBRXMAXP13	R/W	0x0000	USB Maximum Receive Data Endpoint 13	1061
0x1D6	USBRXCSRL13	R/W	0x00	USB Receive Control and Status Endpoint 13 Low	1063
0x1D7	USBRXCSRH13	R/W	0x00	USB Receive Control and Status Endpoint 13 High	1068
0x1D8	USBRXCOUNT13	RO	0x0000	USB Receive Byte Count Endpoint 13	1073
0x1DA	USBTXTYPE13	R/W	0x00	USB Host Transmit Configure Type Endpoint 13	1075
0x1DB	USBTXINTERVAL13	R/W	0x00	USB Host Transmit Interval Endpoint 13	1077
0x1DC	USBRXTYPE13	R/W	0x00	USB Host Configure Receive Type Endpoint 13	1079
0x1DD	USBRXINTERVAL13	R/W	0x00	USB Host Receive Polling Interval Endpoint 13	1081
0x1E0	USBTXMAXP14	R/W	0x0000	USB Maximum Transmit Data Endpoint 14	1041
0x1E2	USBTXCSRL14	R/W	0x00	USB Transmit Control and Status Endpoint 14 Low	1052
0x1E3	USBTXCSRH14	R/W	0x00	USB Transmit Control and Status Endpoint 14 High	1057
0x1E4	USBRXMAXP14	R/W	0x0000	USB Maximum Receive Data Endpoint 14	1061
0x1E6	USBRXCSRL14	R/W	0x00	USB Receive Control and Status Endpoint 14 Low	1063
0x1E7	USBRXCSRH14	R/W	0x00	USB Receive Control and Status Endpoint 14 High	1068
0x1E8	USBRXCOUNT14	RO	0x0000	USB Receive Byte Count Endpoint 14	1073
0x1EA	USBTXTYPE14	R/W	0x00	USB Host Transmit Configure Type Endpoint 14	1075
0x1EB	USBTXINTERVAL14	R/W	0x00	USB Host Transmit Interval Endpoint 14	1077
0x1EC	USBRXTYPE14	R/W	0x00	USB Host Configure Receive Type Endpoint 14	1079
0x1ED	USBRXINTERVAL14	R/W	0x00	USB Host Receive Polling Interval Endpoint 14	1081
0x1F0	USBTXMAXP15	R/W	0x0000	USB Maximum Transmit Data Endpoint 15	1041
0x1F2	USBTXCSRL15	R/W	0x00	USB Transmit Control and Status Endpoint 15 Low	1052
0x1F3	USBTXCSRH15	R/W	0x00	USB Transmit Control and Status Endpoint 15 High	1057
0x1F4	USBRXMAXP15	R/W	0x0000	USB Maximum Receive Data Endpoint 15	1061
0x1F6	USBRXCSRL15	R/W	0x00	USB Receive Control and Status Endpoint 15 Low	1063
0x1F7	USBRXCSRH15	R/W	0x00	USB Receive Control and Status Endpoint 15 High	1068
0x1F8	USBRXCOUNT15	RO	0x0000	USB Receive Byte Count Endpoint 15	1073
0x1FA	USBTXTYPE15	R/W	0x00	USB Host Transmit Configure Type Endpoint 15	1075
0x1FB	USBTXINTERVAL15	R/W	0x00	USB Host Transmit Interval Endpoint 15	1077
0x1FC	USBRXTYPE15	R/W	0x00	USB Host Configure Receive Type Endpoint 15	1079
0x1FD	USBRXINTERVAL15	R/W	0x00	USB Host Receive Polling Interval Endpoint 15	1081

Table 19-6. Universal Serial Bus (USB) Controller Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x304	USBRQPKTCOUNT1	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 1	1083
0x308	USBRQPKTCOUNT2	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 2	1083
0x30C	USBRQPKTCOUNT3	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 3	1083
0x310	USBRQPKTCOUNT4	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 4	1083
0x314	USBRQPKTCOUNT5	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 5	1083
0x318	USBRQPKTCOUNT6	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 6	1083
0x31C	USBRQPKTCOUNT7	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 7	1083
0x320	USBRQPKTCOUNT8	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 8	1083
0x324	USBRQPKTCOUNT9	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 9	1083
0x328	USBRQPKTCOUNT10	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 10	1083
0x32C	USBRQPKTCOUNT11	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 11	1083
0x330	USBRQPKTCOUNT12	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 12	1083
0x334	USBRQPKTCOUNT13	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 13	1083
0x338	USBRQPKTCOUNT14	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 14	1083
0x33C	USBRQPKTCOUNT15	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 15	1083
0x340	USBRXDPKTBUFDIS	R/W	0x0000	USB Receive Double Packet Buffer Disable	1085
0x342	USBTXDPKTBUFDIS	R/W	0x0000	USB Transmit Double Packet Buffer Disable	1087
0x400	USBEPC	R/W	0x0000.0000	USB External Power Control	1089
0x404	USBEPCRIS	RO	0x0000.0000	USB External Power Control Raw Interrupt Status	1092
0x408	USBEPCIM	R/W	0x0000.0000	USB External Power Control Interrupt Mask	1093
0x40C	USBEPCISC	R/W	0x0000.0000	USB External Power Control Interrupt Status and Clear	1094
0x410	USBDRRIS	RO	0x0000.0000	USB Device RESUME Raw Interrupt Status	1095
0x414	USBDRIM	R/W	0x0000.0000	USB Device RESUME Interrupt Mask	1096
0x418	USBDRISC	W1C	0x0000.0000	USB Device RESUME Interrupt Status and Clear	1097

Table 19-6. Universal Serial Bus (USB) Controller Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x41C	USBGPCS	R/W	0x0000.0001	USB General-Purpose Control and Status	1098
0x430	USBVDC	R/W	0x0000.0000	USB VBUS Droop Control	1099
0x434	USBVDCRIS	RO	0x0000.0000	USB VBUS Droop Control Raw Interrupt Status	1100
0x438	USBVDCIM	R/W	0x0000.0000	USB VBUS Droop Control Interrupt Mask	1101
0x43C	USBVDCISC	R/W	0x0000.0000	USB VBUS Droop Control Interrupt Status and Clear	1102
0x444	USBIDVRIS	RO	0x0000.0000	USB ID Valid Detect Raw Interrupt Status	1103
0x448	USBIDVIM	R/W	0x0000.0000	USB ID Valid Detect Interrupt Mask	1104
0x44C	USBIDVISC	R/W1C	0x0000.0000	USB ID Valid Detect Interrupt Status and Clear	1105
0x450	USBDMASEL	R/W	0x0033.2211	USB DMA Select	1106

19.6 Register Descriptions

The LM3S9U81 USB controller has On-The-Go (OTG) capabilities as specified in the USB0 bit field in the **DC6** register (see page 244).

OTG B / Device

This icon indicates that the register is used in OTG B or Device mode. Some registers are used for both Host and Device mode and may have different bit definitions depending on the mode.

OTG A / Host

This icon indicates that the register is used in OTG A or Host mode. Some registers are used for both Host and Device mode and may have different bit definitions depending on the mode. The USB controller is in OTG B or Device mode upon reset, so the reset values shown for these registers apply to the Device mode definition.

OTG

This icon indicates that the register is used for OTG-specific functions such as ID detection and negotiation. Once OTG negotiation is complete, then the USB controller registers are used according to their Host or Device mode meanings depending on whether the OTG negotiations made the USB controller OTG A (Host) or OTG B (Device).

Register 1: USB Device Functional Address (USBFADDR), offset 0x000

OTG B / Device

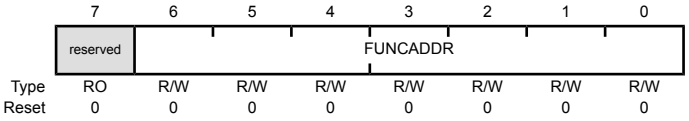
USBFADDR is an 8-bit register that contains the 7-bit address of the Device part of the transaction.

When the USB controller is being used in Device mode (the HOST bit in the **USBDEVCTL** register is clear), this register must be written with the address received through a SET_ADDRESS command, which is then used for decoding the function address in subsequent token packets.

Important: See the section called “Setting the Device Address” on page 976 for special considerations when writing this register.

USB Device Functional Address (USBFADDR)

Base 0x4005.0000
Offset 0x000
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	FUNCADDR	R/W	0x00	Function Address Function Address of Device as received through SET_ADDRESS.

Register 2: USB Power (USBPOWER), offset 0x001

OTG A / Host

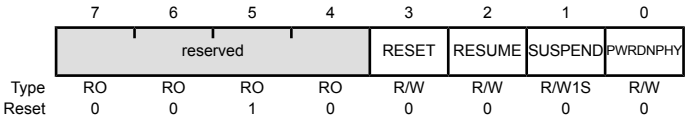
OTG B / Device

USBPOWER is an 8-bit register used for controlling SUSPEND and RESUME signaling and some basic operational aspects of the USB controller.

OTG A / Host Mode

USB Power (USBPOWER)

Base 0x4005.0000
Offset 0x001
Type R/W, reset 0x20



Bit/Field	Name	Type	Reset	Description
7:4	reserved	RO	0x2	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	RESET	R/W	0	RESET Signaling Value Description 1 Enables RESET signaling on the bus. 0 Ends RESET signaling on the bus.
2	RESUME	R/W	0	RESUME Signaling Value Description 1 Enables RESUME signaling when the Device is in SUSPEND mode. 0 Ends RESUME signaling on the bus. This bit must be cleared by software 20 ms after being set.
1	SUSPEND	R/W1S	0	SUSPEND Mode Value Description 1 Enables SUSPEND mode. 0 No effect.

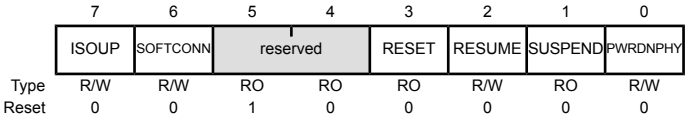
Bit/Field	Name	Type	Reset	Description
0	PWRDNPHY	R/W	0	Power Down PHY

Value	Description
1	Powers down the internal USB PHY.
0	No effect.

OTG B / Device Mode

USB Power (USBPOWER)

Base 0x4005.0000
Offset 0x001
Type R/W, reset 0x20



Bit/Field	Name	Type	Reset	Description
7	ISOUP	R/W	0	Isochronous Update

Value	Description
1	The USB controller waits for an SOF token from the time the TXRDY bit is set in the **USBTXCSRLn** register before sending the packet. If an IN token is received before an SOF token, then a zero-length data packet is sent.
0	No effect.

Note: This bit is only valid for isochronous transfers.

Bit/Field	Name	Type	Reset	Description
6	SOFTCONN	R/W	0	Soft Connect/Disconnect

Value	Description
1	The USB D+/D- lines are enabled.
0	The USB D+/D- lines are tri-stated.

Bit/Field	Name	Type	Reset	Description
5:4	reserved	RO	0x2	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	RESET	RO	0	RESET Signaling

Value	Description
1	RESET signaling is present on the bus.
0	RESET signaling is not present on the bus.

Bit/Field	Name	Type	Reset	Description
2	RESUME	R/W	0	RESUME Signaling Value Description 1 Enables RESUME signaling when the Device is in SUSPEND mode. 0 Ends RESUME signaling on the bus. This bit must be cleared by software 10 ms (a maximum of 15 ms) after being set.
1	SUSPEND	RO	0	SUSPEND Mode Value Description 1 The USB controller is in SUSPEND mode. 0 This bit is cleared when software reads the interrupt register or sets the RESUME bit above.
0	PWRDNPHY	R/W	0	Power Down PHY Value Description 1 Powers down the internal USB PHY. 0 No effect.

Register 3: USB Transmit Interrupt Status (USBTXIS), offset 0x002

Important: This register is read-sensitive. See the register description for details.

OTG A / Host

OTG B / Device

USBTXIS is a 16-bit read-only register that indicates which interrupts are currently active for endpoint 0 and the transmit endpoints 1–15. The meaning of the EPn bits in this register is based on the mode of the device. The EP1 through EP15 bits always indicate that the USB controller is sending data; however, in Host mode, the bits refer to OUT endpoints; while in Device mode, the bits refer to IN endpoints. The EP0 bit is special in Host and Device modes and indicates that either a control IN or control OUT endpoint has generated an interrupt.

Note: Bits relating to endpoints that have not been configured always return 0. Note also that all active interrupts are cleared when this register is read.

USB Transmit Interrupt Status (USBTXIS)

Base 0x4005.0000
Offset 0x002
Type RO, reset 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
15	EP15	RO	0	TX Endpoint 15 Interrupt Value Description 0 No interrupt. 1 The Endpoint 15 transmit interrupt is asserted.
14	EP14	RO	0	TX Endpoint 14 Interrupt Same description as EP15.
13	EP13	RO	0	TX Endpoint 13 Interrupt Same description as EP15.
12	EP12	RO	0	TX Endpoint 12 Interrupt Same description as EP15.
11	EP11	RO	0	TX Endpoint 11 Interrupt Same description as EP15.
10	EP10	RO	0	TX Endpoint 10 Interrupt Same description as EP15.
9	EP9	RO	0	TX Endpoint 9 Interrupt Same description as EP15.
8	EP8	RO	0	TX Endpoint 8 Interrupt Same description as EP15.
7	EP7	RO	0	TX Endpoint 7 Interrupt Same description as EP15.

Bit/Field	Name	Type	Reset	Description
6	EP6	RO	0	TX Endpoint 6 Interrupt Same description as EP15.
5	EP5	RO	0	TX Endpoint 5 Interrupt Same description as EP15.
4	EP4	RO	0	TX Endpoint 4 Interrupt Same description as EP15.
3	EP3	RO	0	TX Endpoint 3 Interrupt Same description as EP15.
2	EP2	RO	0	TX Endpoint 2 Interrupt Same description as EP15.
1	EP1	RO	0	TX Endpoint 1 Interrupt Same description as EP15.
0	EP0	RO	0	TX and RX Endpoint 0 Interrupt Value Description 0 No interrupt. 1 The Endpoint 0 transmit and receive interrupt is asserted.

Register 4: USB Receive Interrupt Status (USBRXIS), offset 0x004

Important: This register is read-sensitive. See the register description for details.

OTG A / Host

USBRXIS is a 16-bit read-only register that indicates which of the interrupts for receive endpoints 1–15 are currently active.

Note: Bits relating to endpoints that have not been configured always return 0. Note also that all active interrupts are cleared when this register is read.

OTG B / Device

USB Receive Interrupt Status (USBRXIS)
Base 0x4005.0000
Offset 0x004
Type RO, reset 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	reserved
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
15	EP15	RO	0	RX Endpoint 15 Interrupt Value Description 0 No interrupt. 1 The Endpoint 15 receive interrupt is asserted.
14	EP14	RO	0	RX Endpoint 14 Interrupt Same description as EP15.
13	EP13	RO	0	RX Endpoint 13 Interrupt Same description as EP15.
12	EP12	RO	0	RX Endpoint 12 Interrupt Same description as EP15.
11	EP11	RO	0	RX Endpoint 11 Interrupt Same description as EP15.
10	EP10	RO	0	RX Endpoint 10 Interrupt Same description as EP15.
9	EP9	RO	0	RX Endpoint 9 Interrupt Same description as EP15.
8	EP8	RO	0	RX Endpoint 8 Interrupt Same description as EP15.
7	EP7	RO	0	RX Endpoint 7 Interrupt Same description as EP15.
6	EP6	RO	0	RX Endpoint 6 Interrupt Same description as EP15.

Bit/Field	Name	Type	Reset	Description
5	EP5	RO	0	RX Endpoint 5 Interrupt Same description as EP15.
4	EP4	RO	0	RX Endpoint 4 Interrupt Same description as EP15.
3	EP3	RO	0	RX Endpoint 3 Interrupt Same description as EP15.
2	EP2	RO	0	RX Endpoint 2 Interrupt Same description as EP15.
1	EP1	RO	0	RX Endpoint 1 Interrupt Same description as EP15.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 5: USB Transmit Interrupt Enable (USBTXIE), offset 0x006

OTG A / Host

OTG B / Device

USBTXIE is a 16-bit register that provides interrupt enable bits for the interrupts in the **USBTXIS** register. When a bit is set, the USB interrupt is asserted to the interrupt controller when the corresponding interrupt bit in the **USBTXIS** register is set. When a bit is cleared, the interrupt in the **USBTXIS** register is still set but the USB interrupt to the interrupt controller is not asserted. On reset, all interrupts are enabled.

USB Transmit Interrupt Enable (USBTXIE)

Base 0x4005.0000
Offset 0x006
Type R/W, reset 0xFFFF

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
15	EP15	R/W	1	TX Endpoint 15 Interrupt Enable Value Description 1 An interrupt is sent to the interrupt controller when the EP15 bit in the **USBTXIS** register is set. 0 The EP15 transmit interrupt is suppressed and not sent to the interrupt controller.
14	EP14	R/W	1	TX Endpoint 14 Interrupt Enable Same description as EP15.
13	EP13	R/W	1	TX Endpoint 13 Interrupt Enable Same description as EP15.
12	EP12	R/W	1	TX Endpoint 12 Interrupt Enable Same description as EP15.
11	EP11	R/W	1	TX Endpoint 11 Interrupt Enable Same description as EP15.
10	EP10	R/W	1	TX Endpoint 10 Interrupt Enable Same description as EP15.
9	EP9	R/W	1	TX Endpoint 9 Interrupt Enable Same description as EP15.
8	EP8	R/W	1	TX Endpoint 8 Interrupt Enable Same description as EP15.
7	EP7	R/W	1	TX Endpoint 7 Interrupt Enable Same description as EP15.
6	EP6	R/W	1	TX Endpoint 6 Interrupt Enable Same description as EP15.

Bit/Field	Name	Type	Reset	Description
5	EP5	R/W	1	TX Endpoint 5 Interrupt Enable Same description as EP15.
4	EP4	R/W	1	TX Endpoint 4 Interrupt Enable Same description as EP15.
3	EP3	R/W	1	TX Endpoint 3 Interrupt Enable Same description as EP15.
2	EP2	R/W	1	TX Endpoint 2 Interrupt Enable Same description as EP15.
1	EP1	R/W	1	TX Endpoint 1 Interrupt Enable Same description as EP15.
0	EP0	R/W	1	TX and RX Endpoint 0 Interrupt Enable Value Description 1 An interrupt is sent to the interrupt controller when the EP0 bit in the **USBTXIS** register is set. 0 The EP0 transmit and receive interrupt is suppressed and not sent to the interrupt controller.

Register 6: USB Receive Interrupt Enable (USBRXIE), offset 0x008

OTG A / Host

OTG B / Device

USBRXIE is a 16-bit register that provides interrupt enable bits for the interrupts in the **USBRXIS** register. When a bit is set, the USB interrupt is asserted to the interrupt controller when the corresponding interrupt bit in the **USBRXIS** register is set. When a bit is cleared, the interrupt in the **USBRXIS** register is still set but the USB interrupt to the interrupt controller is not asserted. On reset, all interrupts are enabled.

USB Receive Interrupt Enable (USBRXIE)

Base 0x4005.0000
Offset 0x008
Type R/W, reset 0xFFFE

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	reserved
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RO
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Bit/Field	Name	Type	Reset	Description
15	EP15	R/W	1	RX Endpoint 15 Interrupt Enable Value Description 1 An interrupt is sent to the interrupt controller when the EP15 bit in the **USBRXIS** register is set. 0 The EP15 receive interrupt is suppressed and not sent to the interrupt controller.
14	EP14	R/W	1	RX Endpoint 14 Interrupt Enable Same description as EP15.
13	EP13	R/W	1	RX Endpoint 13 Interrupt Enable Same description as EP15.
12	EP12	R/W	1	RX Endpoint 12 Interrupt Enable Same description as EP15.
11	EP11	R/W	1	RX Endpoint 11 Interrupt Enable Same description as EP15.
10	EP10	R/W	1	RX Endpoint 10 Interrupt Enable Same description as EP15.
9	EP9	R/W	1	RX Endpoint 9 Interrupt Enable Same description as EP15.
8	EP8	R/W	1	RX Endpoint 8 Interrupt Enable Same description as EP15.
7	EP7	R/W	1	RX Endpoint 7 Interrupt Enable Same description as EP15.
6	EP6	R/W	1	RX Endpoint 6 Interrupt Enable Same description as EP15.

Bit/Field	Name	Type	Reset	Description
5	EP5	R/W	1	RX Endpoint 5 Interrupt Enable Same description as EP15.
4	EP4	R/W	1	RX Endpoint 4 Interrupt Enable Same description as EP15.
3	EP3	R/W	1	RX Endpoint 3 Interrupt Enable Same description as EP15.
2	EP2	R/W	1	RX Endpoint 2 Interrupt Enable Same description as EP15.
1	EP1	R/W	1	RX Endpoint 1 Interrupt Enable Same description as EP15.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 7: USB General Interrupt Status (USBIS), offset 0x00A

Important: This register is read-sensitive. See the register description for details.

OTG A / Host

OTG B / Device

USBIS is an 8-bit read-only register that indicates which USB interrupts are currently active. All active interrupts are cleared when this register is read.

OTG A / Host Mode

USB General Interrupt Status (USBIS)

Base 0x4005.0000
Offset 0x00A
Type RO, reset 0x00

	7	6	5	4	3	2	1	0
	VBUSERR	SESREQ	DISCON	CONN	SOF	BABBLE	RESUME	reserved
Type	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	VBUSERR	RO	0	VBUS Error Value Description 1 VBUS has dropped below the VBUS Valid threshold during a session. 0 No interrupt.
6	SESREQ	RO	0	SESSION REQUEST Value Description 1 SESSION REQUEST signaling has been detected. 0 No interrupt.
5	DISCON	RO	0	Session Disconnect Value Description 1 A Device disconnect has been detected. 0 No interrupt.
4	CONN	RO	0	Session Connect Value Description 1 A Device connection has been detected. 0 No interrupt.

Bit/Field	Name	Type	Reset	Description
3	SOF	RO	0	Start of Frame Value Description 1 A new frame has started. 0 No interrupt.
2	BABBLE	RO	0	Babble Detected Value Description 1 Babble has been detected. This interrupt is active only after the first SOF has been sent. 0 No interrupt.
1	RESUME	RO	0	RESUME Signaling Detected Value Description 1 RESUME signaling has been detected on the bus while the USB controller is in SUSPEND mode. 0 No interrupt. This interrupt can only be used if the USB controller's system clock is enabled. If the user disables the clock programming, the **USBDRRIS**, **USBDRIM**, and **USBDRISC** registers should be used.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

OTG B / Device Mode

USB General Interrupt Status (USBIS)

Base 0x4005.0000
Offset 0x00A
Type RO, reset 0x00

	7	6	5	4	3	2	1	0
	reserved		DISCON	reserved	SOF	RESET	RESUME	SUSPEND
Type	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7:6	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	DISCON	RO	0	Session Disconnect Value Description 1 The device has been disconnected from the host. 0 No interrupt.

Bit/Field	Name	Type	Reset	Description
4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	SOF	RO	0	Start of Frame Value Description 1 A new frame has started. 0 No interrupt.
2	RESET	RO	0	RESET Signaling Detected Value Description 1 RESET signaling has been detected on the bus. 0 No interrupt.
1	RESUME	RO	0	RESUME Signaling Detected Value Description 1 RESUME signaling has been detected on the bus while the USB controller is in SUSPEND mode. 0 No interrupt. This interrupt can only be used if the USB controller's system clock is enabled. If the user disables the clock programming, the **USBDRRIS**, **USBDRIM**, and **USBDRISC** registers should be used.
0	SUSPEND	RO	0	SUSPEND Signaling Detected Value Description 1 SUSPEND signaling has been detected on the bus. 0 No interrupt.

Register 8: USB Interrupt Enable (USBIE), offset 0x00B

OTG A / Host

OTG B / Device

USBIE is an 8-bit register that provides interrupt enable bits for each of the interrupts in **USBIS**. At reset interrupts 1 and 2 are enabled in Device mode.

OTG A / Host Mode

USB Interrupt Enable (USBIE)

Base 0x4005.0000
Offset 0x00B
Type R/W, reset 0x06

	7	6	5	4	3	2	1	0
	VBUSERR	SESREQ	DISCON	CONN	SOF	BABBLE	RESUME	reserved
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RO
Reset	0	0	0	0	0	1	1	0

Bit/Field	Name	Type	Reset	Description
7	VBUSERR	R/W	0	Enable VBUS Error Interrupt

Value	Description
1	An interrupt is sent to the interrupt controller when the VBUSERR bit in the **USBIS** register is set.
0	The VBUSERR interrupt is suppressed and not sent to the interrupt controller.

Bit/Field	Name	Type	Reset	Description
6	SESREQ	R/W	0	Enable Session Request

Value	Description
1	An interrupt is sent to the interrupt controller when the SESREEQ bit in the **USBIS** register is set.
0	The SESREQ interrupt is suppressed and not sent to the interrupt controller.

Bit/Field	Name	Type	Reset	Description
5	DISCON	R/W	0	Enable Disconnect Interrupt

Value	Description
1	An interrupt is sent to the interrupt controller when the DISCON bit in the **USBIS** register is set.
0	The DISCON interrupt is suppressed and not sent to the interrupt controller.

Bit/Field	Name	Type	Reset	Description
4	CONN	R/W	0	Enable Connect Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the CONN bit in the **USBIS** register is set. 0 The CONN interrupt is suppressed and not sent to the interrupt controller.
3	SOF	R/W	0	Enable Start-of-Frame Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the SOF bit in the **USBIS** register is set. 0 The SOF interrupt is suppressed and not sent to the interrupt controller.
2	BABBLE	R/W	1	Enable Babble Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the BABBLE bit in the **USBIS** register is set. 0 The BABBLE interrupt is suppressed and not sent to the interrupt controller.
1	RESUME	R/W	1	Enable RESUME Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the RESUME bit in the **USBIS** register is set. 0 The RESUME interrupt is suppressed and not sent to the interrupt controller.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

OTG B / Device Mode

USB Interrupt Enable (USBIE)

Base 0x4005.0000
Offset 0x00B
Type R/W, reset 0x06

	7	6	5	4	3	2	1	0
	reserved		DISCON	reserved	SOF	RESET	RESUME	SUSPEND
Type	RO	RO	R/W	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

Bit/Field	Name	Type	Reset	Description
7:6	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	DISCON	R/W	0	Enable Disconnect Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the DISCON bit in the **USBIS** register is set. 0 The DISCON interrupt is suppressed and not sent to the interrupt controller.
4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	SOF	R/W	0	Enable Start-of-Frame Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the SOF bit in the **USBIS** register is set. 0 The SOF interrupt is suppressed and not sent to the interrupt controller.
2	RESET	R/W	1	Enable RESET Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the RESET bit in the **USBIS** register is set. 0 The RESET interrupt is suppressed and not sent to the interrupt controller.
1	RESUME	R/W	1	Enable RESUME Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the RESUME bit in the **USBIS** register is set. 0 The RESUME interrupt is suppressed and not sent to the interrupt controller.
0	SUSPEND	R/W	0	Enable SUSPEND Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the SUSPEND bit in the **USBIS** register is set. 0 The SUSPEND interrupt is suppressed and not sent to the interrupt controller.

Register 9: USB Frame Value (USBFRAME), offset 0x00C

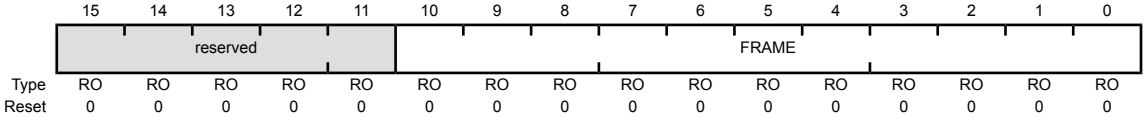
OTG A / Host

OTG B / Device

USBFRAME is a 16-bit read-only register that holds the last received frame number.

USB Frame Value (USBFRAME)

Base 0x4005.0000
Offset 0x00C
Type RO, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:11	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10:0	FRAME	RO	0x000	Frame Number

Register 10: USB Endpoint Index (USBEPIDX), offset 0x00E

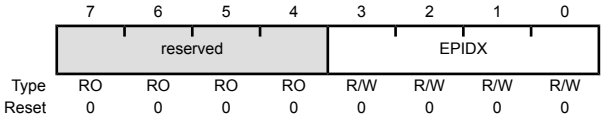
OTG A / Host

Each endpoint's buffer can be accessed by configuring a FIFO size and starting address. The **USBEPIDX** 8-bit register is used with the **USBTXFIFOSZ**, **USBRXFIFOSZ**, **USBTXFIFOADD**, and **USBRXFIFOADD** registers.

OTG B / Device

USB Endpoint Index (USBEPIDX)

Base 0x4005.0000
Offset 0x00E
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	EPIDX	R/W	0x0	Endpoint Index This bit field configures which endpoint is accessed when reading or writing to one of the USB controller's indexed registers. A value of 0x0 corresponds to Endpoint 0 and a value of 0xF corresponds to Endpoint 15.

Register 11: USB Test Mode (USBTEST), offset 0x00F

OTG A / Host

USBTEST is an 8-bit register that is primarily used to put the USB controller into one of the four test modes for operation described in the *USB 2.0 Specification*, in response to a SET FEATURE: USBTESTMODE command. This register is not used in normal operation.

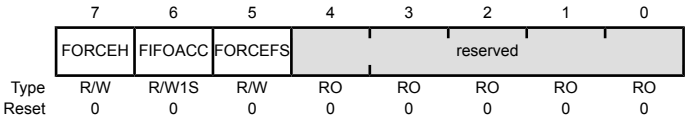
Note: Only one of these bits should be set at any time.

OTG B / Device

OTG A / Host Mode

USB Test Mode (USBTEST)

Base 0x4005.0000
Offset 0x00F
Type R/W, reset 0x00



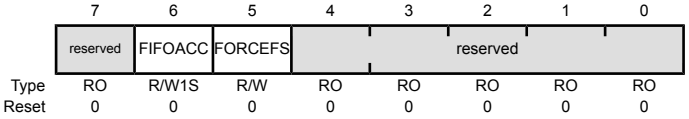
Bit/Field	Name	Type	Reset	Description
7	FORCEH	R/W	0	Force Host Mode Value Description 1 Forces the USB controller to enter Host mode when the SESSION bit is set, regardless of whether the USB controller is connected to any peripheral. The state of the USB0DP and USB0DM signals is ignored. The USB controller then remains in Host mode until the SESSION bit is cleared, even if a Device is disconnected. If the FORCEH bit remains set, the USB controller re-enters Host mode the next time the SESSION bit is set. 0 No effect. While in this mode, status of the bus connection may be read using the DEV bit of the **USBDEVCTL** register. The operating speed is determined from the FORCEFS bit.
6	FIFOACC	R/W1S	0	FIFO Access Value Description 1 Transfers the packet in the endpoint 0 transmit FIFO to the endpoint 0 receive FIFO. 0 No effect. This bit is cleared automatically.
5	FORCEFS	R/W	0	Force Full-Speed Mode Value Description 1 Forces the USB controller into Full-Speed mode upon receiving a USB RESET. 0 The USB controller operates at Low Speed.

Bit/Field	Name	Type	Reset	Description
4:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

OTG B / Device Mode

USB Test Mode (USBTEST)

Base 0x4005.0000
Offset 0x00F
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	FIFOACC	R/W1S	0	FIFO Access Value Description 1 Transfers the packet in the endpoint 0 transmit FIFO to the endpoint 0 receive FIFO. 0 No effect. This bit is cleared automatically.
5	FORCEFS	R/W	0	Force Full-Speed Mode Value Description 1 Forces the USB controller into Full-Speed mode upon receiving a USB RESET. 0 The USB controller operates at Low Speed.
4:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 12: USB FIFO Endpoint 0 (USBFIFO0), offset 0x020
Register 13: USB FIFO Endpoint 1 (USBFIFO1), offset 0x024
Register 14: USB FIFO Endpoint 2 (USBFIFO2), offset 0x028
Register 15: USB FIFO Endpoint 3 (USBFIFO3), offset 0x02C
Register 16: USB FIFO Endpoint 4 (USBFIFO4), offset 0x030
Register 17: USB FIFO Endpoint 5 (USBFIFO5), offset 0x034
Register 18: USB FIFO Endpoint 6 (USBFIFO6), offset 0x038
Register 19: USB FIFO Endpoint 7 (USBFIFO7), offset 0x03C
Register 20: USB FIFO Endpoint 8 (USBFIFO8), offset 0x040
Register 21: USB FIFO Endpoint 9 (USBFIFO9), offset 0x044
Register 22: USB FIFO Endpoint 10 (USBFIFO10), offset 0x048
Register 23: USB FIFO Endpoint 11 (USBFIFO11), offset 0x04C
Register 24: USB FIFO Endpoint 12 (USBFIFO12), offset 0x050
Register 25: USB FIFO Endpoint 13 (USBFIFO13), offset 0x054
Register 26: USB FIFO Endpoint 14 (USBFIFO14), offset 0x058
Register 27: USB FIFO Endpoint 15 (USBFIFO15), offset 0x05C

Important: This register is read-sensitive. See the register description for details.

OTG A / Host

These 32-bit registers provide an address for CPU access to the FIFOs for each endpoint. Writing to these addresses loads data into the Transmit FIFO for the corresponding endpoint. Reading from these addresses unloads data from the Receive FIFO for the corresponding endpoint.

OTG B / Device

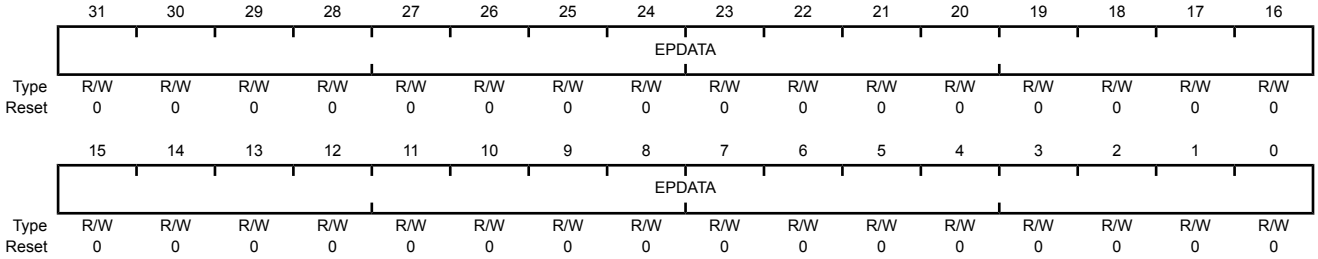
Transfers to and from FIFOs may be 8-bit, 16-bit or 32-bit as required, and any combination of accesses is allowed provided the data accessed is contiguous. All transfers associated with one packet must be of the same width so that the data is consistently byte-, halfword- or word-aligned. However, the last transfer may contain fewer bytes than the previous transfers in order to complete an odd-byte or odd-word transfer.

Depending on the size of the FIFO and the expected maximum packet size, the FIFOs support either single-packet or double-packet buffering (see the section called “Single-Packet Buffering” on page 974). Burst writing of multiple packets is not supported as flags must be set after each packet is written.

Following a STALL response or a transmit error on endpoint 1–15, the associated FIFO is completely flushed.

USB FIFO Endpoint 0 (USBFIFO0)

Base 0x4005.0000
Offset 0x020
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	EPDATA	R/W	0x0000.0000	Endpoint Data Writing to this register loads the data into the Transmit FIFO and reading unloads data from the Receive FIFO.

Register 28: USB Device Control (USBDEVCTL), offset 0x060

OTG A / Host

USBDEVCTL is an 8-bit register used for controlling and monitoring the USB VBUS line. If the PHY is suspended, no PHY clock is received and the VBUS is not sampled. In addition, in Host mode, **USBDEVCTL** provides the status information for the current operating mode (Host or Device) of the USB controller. If the USB controller is in Host mode, this register also indicates if a full- or low-speed Device has been connected.

USB Device Control (USBDEVCTL)
Base 0x4005.0000
Offset 0x060
Type R/W, reset 0x80

	7	6	5	4	3	2	1	0
	DEV	FSDEV	LSDEV	VBUS		HOST	HOSTREQ	SESSION
Type	RO	RO	RO	RO	RO	RO	R/W	R/W
Reset	1	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	DEV	RO	1	Device Mode Value Description 0 The USB controller is operating on the OTG A side of the cable. 1 The USB controller is operating on the OTG B side of the cable. **Note:** This value is only valid while a session is in progress.
6	FSDEV	RO	0	Full-Speed Device Detected Value Description 0 A full-speed Device has not been detected on the port. 1 A full-speed Device has been detected on the port.
5	LSDEV	RO	0	Low-Speed Device Detected Value Description 0 A low-speed Device has not been detected on the port. 1 A low-speed Device has been detected on the port.
4:3	VBUS	RO	0x0	VBUS Level Value Description 0x0 Below SessionEnd VBUS is detected as under 0.5 V. 0x1 Above SessionEnd, below AValid VBUS is detected as above 0.5 V and under 1.5 V. 0x2 Above AValid, below VBUSValid VBUS is detected as above 1.5 V and below 4.75 V. 0x3 Above VBUSValid VBUS is detected as above 4.75 V.

Bit/Field	Name	Type	Reset	Description
2	HOST	RO	0	Host Mode Value Description 0 The USB controller is acting as a Device. 1 The USB controller is acting as a Host. **Note:** This value is only valid while a session is in progress.
1	HOSTREQ	R/W	0	Host Request Value Description 0 No effect. 1 Initiates the Host Negotiation when SUSPEND mode is entered. This bit is cleared when Host Negotiation is completed.
0	SESSION	R/W	0	Session Start/End *When operating as an OTG A device:* Value Description 0 When cleared by software, this bit ends a session. 1 When set by software, this bit starts a session. *When operating as an OTG B device:* Value Description 0 The USB controller has ended a session. When the USB controller is in SUSPEND mode, this bit may be cleared by software to perform a software disconnect. 1 The USB controller has started a session. When set by software, the Session Request Protocol is initiated. **Note:** Clearing this bit when the USB controller is not suspended results in undefined behavior.

Register 29: USB Transmit Dynamic FIFO Sizing (USBTXFIFOSZ), offset 0x062

Register 30: USB Receive Dynamic FIFO Sizing (USBRXFIFOSZ), offset 0x063

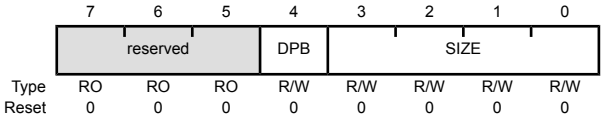
OTG A / Host

OTG B / Device

These 8-bit registers allow the selected TX/RX endpoint FIFOs to be dynamically sized. **USBEPIDX** is used to configure each transmit endpoint's FIFO size.

USB Transmit Dynamic FIFO Sizing (USBTXFIFOSZ)

Base 0x4005.0000
Offset 0x062
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	DPB	R/W	0	Double Packet Buffer Support Value Description 0 Only single-packet buffering is supported. 1 Double-packet buffering is supported.
3:0	SIZE	R/W	0x0	Max Packet Size Maximum packet size to be allowed. If DPB = 0, the FIFO also is this size; if DPB = 1, the FIFO is twice this size. Value Packet Size (Bytes) 0x0 8 0x1 16 0x2 32 0x3 64 0x4 128 0x5 256 0x6 512 0x7 1024 0x8 2048 0x9-0xF Reserved

Register 31: USB Transmit FIFO Start Address (USBTXFIFOADD), offset 0x064

Register 32: USB Receive FIFO Start Address (USBRXFIFOADD), offset 0x066

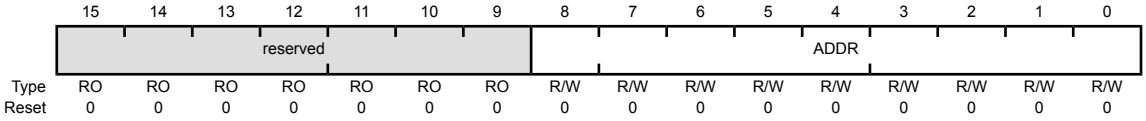
OTG A / Host

USBTXFIFOADD and **USBRXFIFOADD** are 16-bit registers that control the start address of the selected transmit and receive endpoint FIFOs.

OTG B / Device

USB Transmit FIFO Start Address (USBTXFIFOADD)

Base 0x4005.0000
Offset 0x064
Type R/W, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:9	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8:0	ADDR	R/W	0x00	Transmit/Receive Start Address Start address of the endpoint FIFO.

Value	Start Address
0x0	0
0x1	8
0x2	16
0x3	24
0x4	32
0x5	40
0x6	48
0x7	56
0x8	64
...	...
0x1FF	4095

Register 33: USB Connect Timing (USBCONTIM), offset 0x07A

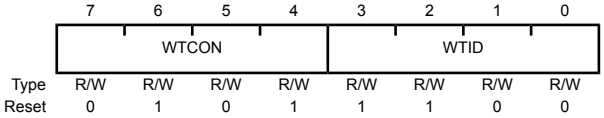
OTG A / Host

OTG B / Device

This 8-bit configuration register specifies connection and negotiation delays.

USB Connect Timing (USBCONTIM)

Base 0x4005.0000
Offset 0x07A
Type R/W, reset 0x5C



Bit/Field	Name	Type	Reset	Description
7:4	WTCON	R/W	0x5	Connect Wait This field configures the wait required to allow for the user's connect/disconnect filter, in units of 533.3 ns. The default corresponds to 2.667 µs.
3:0	WTID	R/W	0xC	Wait ID This field configures the delay required from the enable of the ID detection to when the ID value is valid, in units of 4.369 ms. The default corresponds to 52.43 ms.

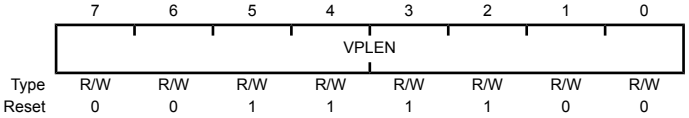
Register 34: USB OTG VBUS Pulse Timing (USBVPLEN), offset 0x07B

OTG

This 8-bit configuration register specifies the duration of the VBUS pulsing charge.

USB OTG VBUS Pulse Timing (USBVPLEN)

Base 0x4005.0000
Offset 0x07B
Type R/W, reset 0x3C



Bit/Field	Name	Type	Reset	Description
7:0	VPLEN	R/W	0x3C	VBUS Pulse Length This field configures the duration of the VBUS pulsing charge in units of 546.1 µs. The default corresponds to 32.77 ms.

Register 35: USB Full-Speed Last Transaction to End of Frame Timing (USBFSEOF), offset 0x07D

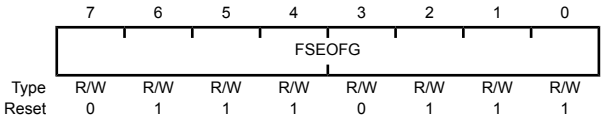
OTG A / Host

This 8-bit configuration register specifies the minimum time gap allowed between the start of the last transaction and the EOF for full-speed transactions.

OTG B / Device

USB Full-Speed Last Transaction to End of Frame Timing (USBFSEOF)

Base 0x4005.0000
Offset 0x07D
Type R/W, reset 0x77



Bit/Field	Name	Type	Reset	Description
7:0	FSEOFG	R/W	0x77	Full-Speed End-of-Frame Gap This field is used during full-speed transactions to configure the gap between the last transaction and the End-of-Frame (EOF), in units of 533.3 ns. The default corresponds to 63.46 µs.

Register 36: USB Low-Speed Last Transaction to End of Frame Timing (USBLSEOF), offset 0x07E

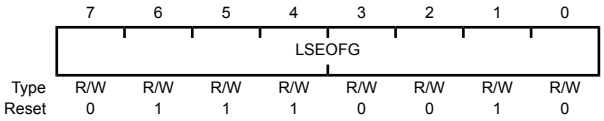
OTG A / Host

This 8-bit configuration register specifies the minimum time gap that is to be allowed between the start of the last transaction and the EOF for low-speed transactions.

OTG B / Device

USB Low-Speed Last Transaction to End of Frame Timing (USBLSEOF)

Base 0x4005.0000
Offset 0x07E
Type R/W, reset 0x72



Bit/Field	Name	Type	Reset	Description
7:0	LSEOFG	R/W	0x72	Low-Speed End-of-Frame Gap This field is used during low-speed transactions to set the gap between the last transaction and the End-of-Frame (EOF), in units of 1.067 µs. The default corresponds to 121.6 µs.

Register 37: USB Transmit Functional Address Endpoint 0 (USBTXFUNCADDR0), offset 0x080

Register 38: USB Transmit Functional Address Endpoint 1 (USBTXFUNCADDR1), offset 0x088

Register 39: USB Transmit Functional Address Endpoint 2 (USBTXFUNCADDR2), offset 0x090

Register 40: USB Transmit Functional Address Endpoint 3 (USBTXFUNCADDR3), offset 0x098

Register 41: USB Transmit Functional Address Endpoint 4 (USBTXFUNCADDR4), offset 0x0A0

Register 42: USB Transmit Functional Address Endpoint 5 (USBTXFUNCADDR5), offset 0x0A8

Register 43: USB Transmit Functional Address Endpoint 6 (USBTXFUNCADDR6), offset 0x0B0

Register 44: USB Transmit Functional Address Endpoint 7 (USBTXFUNCADDR7), offset 0x0B8

Register 45: USB Transmit Functional Address Endpoint 8 (USBTXFUNCADDR8), offset 0x0C0

Register 46: USB Transmit Functional Address Endpoint 9 (USBTXFUNCADDR9), offset 0x0C8

Register 47: USB Transmit Functional Address Endpoint 10 (USBTXFUNCADDR10), offset 0x0D0

Register 48: USB Transmit Functional Address Endpoint 11 (USBTXFUNCADDR11), offset 0x0D8

Register 49: USB Transmit Functional Address Endpoint 12 (USBTXFUNCADDR12), offset 0x0E0

Register 50: USB Transmit Functional Address Endpoint 13 (USBTXFUNCADDR13), offset 0x0E8

Register 51: USB Transmit Functional Address Endpoint 14 (USBTXFUNCADDR14), offset 0x0F0

Register 52: USB Transmit Functional Address Endpoint 15 (USBTXFUNCADDR15), offset 0x0F8

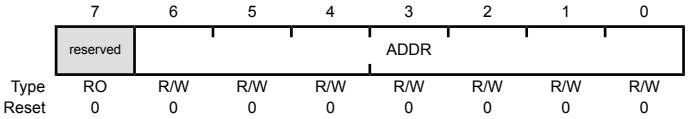
OTG A / Host

USBTXFUNCADDRn is an 8-bit read/write register that records the address of the target function to be accessed through the associated endpoint (EPn). **USBTXFUNCADDRn** must be defined for each transmit endpoint that is used.

Note: **USBTXFUNCADDR0** is used for both receive and transmit for endpoint 0.

USB Transmit Functional Address Endpoint 0 (USBTXFUNCADDR0)

Base 0x4005.0000
Offset 0x080
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	ADDR	R/W	0x00	Device Address Specifies the USB bus address for the target Device.

Register 53: USB Transmit Hub Address Endpoint 0 (USBTXHUBADDR0), offset 0x082

Register 54: USB Transmit Hub Address Endpoint 1 (USBTXHUBADDR1), offset 0x08A

Register 55: USB Transmit Hub Address Endpoint 2 (USBTXHUBADDR2), offset 0x092

Register 56: USB Transmit Hub Address Endpoint 3 (USBTXHUBADDR3), offset 0x09A

Register 57: USB Transmit Hub Address Endpoint 4 (USBTXHUBADDR4), offset 0x0A2

Register 58: USB Transmit Hub Address Endpoint 5 (USBTXHUBADDR5), offset 0x0AA

Register 59: USB Transmit Hub Address Endpoint 6 (USBTXHUBADDR6), offset 0x0B2

Register 60: USB Transmit Hub Address Endpoint 7 (USBTXHUBADDR7), offset 0x0BA

Register 61: USB Transmit Hub Address Endpoint 8 (USBTXHUBADDR8), offset 0x0C2

Register 62: USB Transmit Hub Address Endpoint 9 (USBTXHUBADDR9), offset 0x0CA

Register 63: USB Transmit Hub Address Endpoint 10 (USBTXHUBADDR10), offset 0x0D2

Register 64: USB Transmit Hub Address Endpoint 11 (USBTXHUBADDR11), offset 0x0DA

Register 65: USB Transmit Hub Address Endpoint 12 (USBTXHUBADDR12), offset 0x0E2

Register 66: USB Transmit Hub Address Endpoint 13 (USBTXHUBADDR13), offset 0x0EA

Register 67: USB Transmit Hub Address Endpoint 14 (USBTXHUBADDR14), offset 0x0F2

Register 68: USB Transmit Hub Address Endpoint 15 (USBTXHUBADDR15), offset 0x0FA

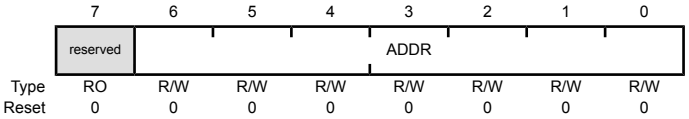
OTG A / Host

USBTXHUBADDRn is an 8-bit read/write register that, like **USBTXHUBPORTn**, only must be written when a USB Device is connected to transmit endpoint EPn via a USB 2.0 hub. This register records the address of the USB 2.0 hub through which the target associated with the endpoint is accessed.

Note: **USBTXHUBADDR0** is used for both receive and transmit for endpoint 0.

USB Transmit Hub Address Endpoint 0 (USBTXHUBADDR0)

Base 0x4005.0000
Offset 0x082
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	ADDR	R/W	0x00	Hub Address This field specifies the USB bus address for the USB 2.0 hub.

Register 69: USB Transmit Hub Port Endpoint 0 (USBTXHUBPORT0), offset 0x083

Register 70: USB Transmit Hub Port Endpoint 1 (USBTXHUBPORT1), offset 0x08B

Register 71: USB Transmit Hub Port Endpoint 2 (USBTXHUBPORT2), offset 0x093

Register 72: USB Transmit Hub Port Endpoint 3 (USBTXHUBPORT3), offset 0x09B

Register 73: USB Transmit Hub Port Endpoint 4 (USBTXHUBPORT4), offset 0x0A3

Register 74: USB Transmit Hub Port Endpoint 5 (USBTXHUBPORT5), offset 0x0AB

Register 75: USB Transmit Hub Port Endpoint 6 (USBTXHUBPORT6), offset 0x0B3

Register 76: USB Transmit Hub Port Endpoint 7 (USBTXHUBPORT7), offset 0x0BB

Register 77: USB Transmit Hub Port Endpoint 8 (USBTXHUBPORT8), offset 0x0C3

Register 78: USB Transmit Hub Port Endpoint 9 (USBTXHUBPORT9), offset 0x0CB

Register 79: USB Transmit Hub Port Endpoint 10 (USBTXHUBPORT10), offset 0x0D3

Register 80: USB Transmit Hub Port Endpoint 11 (USBTXHUBPORT11), offset 0x0DB

Register 81: USB Transmit Hub Port Endpoint 12 (USBTXHUBPORT12), offset 0x0E3

Register 82: USB Transmit Hub Port Endpoint 13 (USBTXHUBPORT13), offset 0x0EB

Register 83: USB Transmit Hub Port Endpoint 14 (USBTXHUBPORT14), offset 0x0F3

Register 84: USB Transmit Hub Port Endpoint 15 (USBTXHUBPORT15), offset 0x0FB

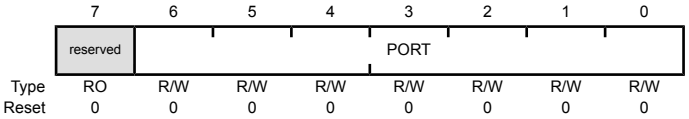
OTG A / Host

USBTXHUBPORTn is an 8-bit read/write register that, like **USBTXHUBADDRn**, only must be written when a full- or low-speed Device is connected to transmit endpoint EPn via a USB 2.0 hub. This register records the port of the USB 2.0 hub through which the target associated with the endpoint is accessed.

Note: **USBTXHUBPORT0** is used for both receive and transmit for endpoint 0.

USB Transmit Hub Port Endpoint 0 (USBTXHUBPORT0)

Base 0x4005.0000
Offset 0x083
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	PORT	R/W	0x00	Hub Port This field specifies the USB hub port number.

Register 85: USB Receive Functional Address Endpoint 1 (USBRXFUNCADDR1), offset 0x08C

Register 86: USB Receive Functional Address Endpoint 2 (USBRXFUNCADDR2), offset 0x094

Register 87: USB Receive Functional Address Endpoint 3 (USBRXFUNCADDR3), offset 0x09C

Register 88: USB Receive Functional Address Endpoint 4 (USBRXFUNCADDR4), offset 0x0A4

Register 89: USB Receive Functional Address Endpoint 5 (USBRXFUNCADDR5), offset 0x0AC

Register 90: USB Receive Functional Address Endpoint 6 (USBRXFUNCADDR6), offset 0x0B4

Register 91: USB Receive Functional Address Endpoint 7 (USBRXFUNCADDR7), offset 0x0BC

Register 92: USB Receive Functional Address Endpoint 8 (USBRXFUNCADDR8), offset 0x0C4

Register 93: USB Receive Functional Address Endpoint 9 (USBRXFUNCADDR9), offset 0x0CC

Register 94: USB Receive Functional Address Endpoint 10 (USBRXFUNCADDR10), offset 0x0D4

Register 95: USB Receive Functional Address Endpoint 11 (USBRXFUNCADDR11), offset 0x0DC

Register 96: USB Receive Functional Address Endpoint 12 (USBRXFUNCADDR12), offset 0x0E4

Register 97: USB Receive Functional Address Endpoint 13 (USBRXFUNCADDR13), offset 0x0EC

Register 98: USB Receive Functional Address Endpoint 14 (USBRXFUNCADDR14), offset 0x0F4

Register 99: USB Receive Functional Address Endpoint 15 (USBRXFUNCADDR15), offset 0x0FC

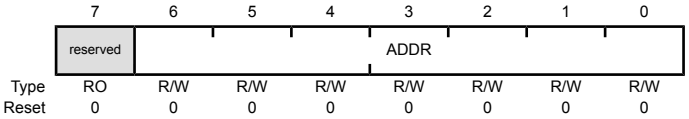
OTG A / Host

USBRXFUNCADDRn is an 8-bit read/write register that records the address of the target function accessed through the associated endpoint (EPn). **USBRXFUNCADDRn** must be defined for each receive endpoint that is used.

Note: **USBTXFUNCADDR0** is used for both receive and transmit for endpoint 0.

USB Receive Functional Address Endpoint 1 (USBRXFUNCADDR1)

Base 0x4005.0000
Offset 0x08C
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	ADDR	R/W	0x00	Device Address This field specifies the USB bus address for the target Device.

Register 100: USB Receive Hub Address Endpoint 1 (USBRXHUBADDR1), offset 0x08E

Register 101: USB Receive Hub Address Endpoint 2 (USBRXHUBADDR2), offset 0x096

Register 102: USB Receive Hub Address Endpoint 3 (USBRXHUBADDR3), offset 0x09E

Register 103: USB Receive Hub Address Endpoint 4 (USBRXHUBADDR4), offset 0x0A6

Register 104: USB Receive Hub Address Endpoint 5 (USBRXHUBADDR5), offset 0x0AE

Register 105: USB Receive Hub Address Endpoint 6 (USBRXHUBADDR6), offset 0x0B6

Register 106: USB Receive Hub Address Endpoint 7 (USBRXHUBADDR7), offset 0x0BE

Register 107: USB Receive Hub Address Endpoint 8 (USBRXHUBADDR8), offset 0x0C6

Register 108: USB Receive Hub Address Endpoint 9 (USBRXHUBADDR9), offset 0x0CE

Register 109: USB Receive Hub Address Endpoint 10 (USBRXHUBADDR10), offset 0x0D6

Register 110: USB Receive Hub Address Endpoint 11 (USBRXHUBADDR11), offset 0x0DE

Register 111: USB Receive Hub Address Endpoint 12 (USBRXHUBADDR12), offset 0x0E6

Register 112: USB Receive Hub Address Endpoint 13 (USBRXHUBADDR13), offset 0x0EE

Register 113: USB Receive Hub Address Endpoint 14 (USBRXHUBADDR14), offset 0x0F6

Register 114: USB Receive Hub Address Endpoint 15 (USBRXHUBADDR15), offset 0x0FE

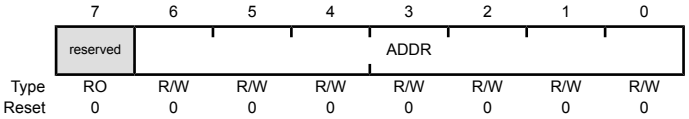
OTG A / Host

USBRXHUBADDRn is an 8-bit read/write register that, like **USBRXHUBPORTn**, only must be written when a full- or low-speed Device is connected to receive endpoint EPn via a USB 2.0 hub. This register records the address of the USB 2.0 hub through which the target associated with the endpoint is accessed.

Note: **USBTXHUBADDR0** is used for both receive and transmit for endpoint 0.

USB Receive Hub Address Endpoint 1 (USBRXHUBADDR1)

Base 0x4005.0000
Offset 0x08E
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	ADDR	R/W	0x00	Hub Address This field specifies the USB bus address for the USB 2.0 hub.

Register 115: USB Receive Hub Port Endpoint 1 (USBRXHUBPORT1), offset 0x08F

Register 116: USB Receive Hub Port Endpoint 2 (USBRXHUBPORT2), offset 0x097

Register 117: USB Receive Hub Port Endpoint 3 (USBRXHUBPORT3), offset 0x09F

Register 118: USB Receive Hub Port Endpoint 4 (USBRXHUBPORT4), offset 0x0A7

Register 119: USB Receive Hub Port Endpoint 5 (USBRXHUBPORT5), offset 0x0AF

Register 120: USB Receive Hub Port Endpoint 6 (USBRXHUBPORT6), offset 0x0B7

Register 121: USB Receive Hub Port Endpoint 7 (USBRXHUBPORT7), offset 0x0BF

Register 122: USB Receive Hub Port Endpoint 8 (USBRXHUBPORT8), offset 0x0C7

Register 123: USB Receive Hub Port Endpoint 9 (USBRXHUBPORT9), offset 0x0CF

Register 124: USB Receive Hub Port Endpoint 10 (USBRXHUBPORT10), offset 0x0D7

Register 125: USB Receive Hub Port Endpoint 11 (USBRXHUBPORT11), offset 0x0DF

Register 126: USB Receive Hub Port Endpoint 12 (USBRXHUBPORT12), offset 0x0E7

Register 127: USB Receive Hub Port Endpoint 13 (USBRXHUBPORT13), offset 0x0EF

Register 128: USB Receive Hub Port Endpoint 14 (USBRXHUBPORT14), offset 0x0F7

Register 129: USB Receive Hub Port Endpoint 15 (USBRXHUBPORT15), offset 0x0FF

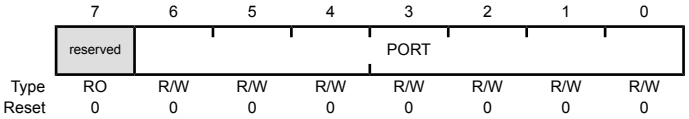
OTG A / Host

USBRXHUBPORTn is an 8-bit read/write register that, like **USBRXHUBADDRn**, only must be written when a full- or low-speed Device is connected to receive endpoint EPn via a USB 2.0 hub. This register records the port of the USB 2.0 hub through which the target associated with the endpoint is accessed.

Note: **USBTXHUBPORT0** is used for both receive and transmit for endpoint 0.

USB Receive Hub Port Endpoint 1 (USBRXHUBPORT1)

Base 0x4005.0000
Offset 0x08F
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	PORT	R/W	0x00	Hub Port This field specifies the USB hub port number.

Register 130: USB Maximum Transmit Data Endpoint 1 (USBTXMAXP1), offset 0x110

Register 131: USB Maximum Transmit Data Endpoint 2 (USBTXMAXP2), offset 0x120

Register 132: USB Maximum Transmit Data Endpoint 3 (USBTXMAXP3), offset 0x130

Register 133: USB Maximum Transmit Data Endpoint 4 (USBTXMAXP4), offset 0x140

Register 134: USB Maximum Transmit Data Endpoint 5 (USBTXMAXP5), offset 0x150

Register 135: USB Maximum Transmit Data Endpoint 6 (USBTXMAXP6), offset 0x160

Register 136: USB Maximum Transmit Data Endpoint 7 (USBTXMAXP7), offset 0x170

Register 137: USB Maximum Transmit Data Endpoint 8 (USBTXMAXP8), offset 0x180

Register 138: USB Maximum Transmit Data Endpoint 9 (USBTXMAXP9), offset 0x190

Register 139: USB Maximum Transmit Data Endpoint 10 (USBTXMAXP10), offset 0x1A0

Register 140: USB Maximum Transmit Data Endpoint 11 (USBTXMAXP11), offset 0x1B0

Register 141: USB Maximum Transmit Data Endpoint 12 (USBTXMAXP12), offset 0x1C0

Register 142: USB Maximum Transmit Data Endpoint 13 (USBTXMAXP13), offset 0x1D0

Register 143: USB Maximum Transmit Data Endpoint 14 (USBTXMAXP14), offset 0x1E0

Register 144: USB Maximum Transmit Data Endpoint 15 (USBTXMAXP15), offset 0x1F0

OTG A / Host

OTG B / Device

The **USBTXMAXPn** 16-bit register defines the maximum amount of data that can be transferred through the transmit endpoint in a single operation.

Bits 10:0 define (in bytes) the maximum payload transmitted in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the *USB Specification* on packet sizes for bulk, interrupt and isochronous transfers in full-speed operation.

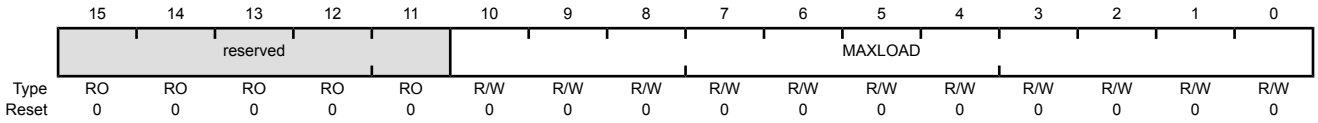
The total amount of data represented by the value written to this register must not exceed the FIFO size for the transmit endpoint, and must not exceed half the FIFO size if double-buffering is required.

If this register is changed after packets have been sent from the endpoint, the transmit endpoint FIFO must be completely flushed (using the FLUSH bit in **USBTXCSRLn**) after writing the new value to this register.

Note: **USBTXMAXPn** must be set to an even number of bytes for proper interrupt generation in µDMA Basic Mode.

USB Maximum Transmit Data Endpoint 1 (USBTXMAXP1)

Base 0x4005.0000
Offset 0x110
Type R/W, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:11	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10:0	MAXLOAD	R/W	0x000	Maximum Payload This field specifies the maximum payload in bytes per transaction.

Register 145: USB Control and Status Endpoint 0 Low (USBCSRL0), offset 0x102

OTG A / Host

OTG B / Device

USBCSRL0 is an 8-bit register that provides control and status bits for endpoint 0.

OTG A / Host Mode

USB Control and Status Endpoint 0 Low (USBCSRL0)

Base 0x4005.0000
Offset 0x102
Type W1C, reset 0x00

	7	6	5	4	3	2	1	0
	NAKTO	STATUS	REQPKT	ERROR	SETUP	STALLED	TXRDY	RXRDY
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	NAKTO	R/W	0	NAK Timeout Value Description 0 No timeout. 1 Indicates that endpoint 0 is halted following the receipt of NAK responses for longer than the time set by the **USBNAKLMT** register. Software must clear this bit to allow the endpoint to continue.
6	STATUS	R/W	0	STATUS Packet Value Description 0 No transaction. 1 Initiates a STATUS stage transaction. This bit must be set at the same time as the TXRDY or REQPKT bit is set. Setting this bit ensures that the DT bit is set in the **USBCSRH0** register so that a DATA1 packet is used for the STATUS stage transaction. This bit is automatically cleared when the STATUS stage is over.
5	REQPKT	R/W	0	Request Packet Value Description 0 No request. 1 Requests an IN transaction. This bit is cleared when the RXRDY bit is set.

Bit/Field	Name	Type	Reset	Description
4	ERROR	R/W	0	Error Value Description 0 No error. 1 Three attempts have been made to perform a transaction with no response from the peripheral. The EP0 bit in the **USBTXIS** register is also set in this situation. Software must clear this bit.
3	SETUP	R/W	0	Setup Packet Value Description 0 Sends an OUT token. 1 Sends a SETUP token instead of an OUT token for the transaction. This bit should be set at the same time as the TXRDY bit is set. Setting this bit always clears the DT bit in the **USBCSRH0** register to send a DATA0 packet.
2	STALLED	R/W	0	Endpoint Stalled Value Description 0 No handshake has been received. 1 A STALL handshake has been received. Software must clear this bit.
1	TXRDY	R/W	0	Transmit Packet Ready Value Description 0 No transmit packet is ready. 1 Software sets this bit after loading a data packet into the TX FIFO. The EP0 bit in the **USBTXIS** register is also set in this situation. If both the TXRDY and SETUP bits are set, a setup packet is sent. If just TXRDY is set, an OUT packet is sent. This bit is cleared automatically when the data packet has been transmitted.
0	RXRDY	R/W	0	Receive Packet Ready Value Description 0 No received packet has been received. 1 Indicates that a data packet has been received in the RX FIFO. The EP0 bit in the **USBTXIS** register is also set in this situation. Software must clear this bit after the packet has been read from the FIFO to acknowledge that the data has been read from the FIFO.

OTG B / Device Mode

USB Control and Status Endpoint 0 Low (USBCSRL0)

Base 0x4005.0000
Offset 0x102
Type W1C, reset 0x00

	7	6	5	4	3	2	1	0
	SETENDC	RXRDYC	STALL	SETEND	DATAEND	STALLED	TXRDY	RXRDY
Type	W1C	W1C	R/W	RO	R/W	R/W	R/W	RO
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	SETENDC	W1C	0	Setup End Clear Writing a 1 to this bit clears the SETEND bit.
6	RXRDYC	W1C	0	RXRDY Clear Writing a 1 to this bit clears the RXRDY bit.
5	STALL	R/W	0	Send Stall Value Description 0 No effect. 1 Terminates the current transaction and transmits the STALL handshake. This bit is cleared automatically after the STALL handshake is transmitted.
4	SETEND	RO	0	Setup End Value Description 0 A control transaction has not ended or ended after the DATAEND bit was set. 1 A control transaction has ended before the DATAEND bit has been set. The EP0 bit in the **USBTXIS** register is also set in this situation. This bit is cleared by writing a 1 to the SETENDC bit.
3	DATAEND	R/W	0	Data End Value Description 0 No effect. 1 Set this bit in the following situations: ■ When setting TXRDY for the last data packet ■ When clearing RXRDY after unloading the last data packet ■ When setting TXRDY for a zero-length data packet This bit is cleared automatically.

Bit/Field	Name	Type	Reset	Description
2	STALLED	R/W	0	Endpoint Stalled Value Description 0 A STALL handshake has not been transmitted. 1 A STALL handshake has been transmitted. Software must clear this bit.
1	TXRDY	R/W	0	Transmit Packet Ready Value Description 0 No transmit packet is ready. 1 Software sets this bit after loading an IN data packet into the TX FIFO. The EP0 bit in the **USBTXIS** register is also set in this situation. This bit is cleared automatically when the data packet has been transmitted.
0	RXRDY	RO	0	Receive Packet Ready Value Description 0 No data packet has been received. 1 A data packet has been received. The EP0 bit in the **USBTXIS** register is also set in this situation. This bit is cleared by writing a 1 to the RXRDYC bit.

Register 146: USB Control and Status Endpoint 0 High (USBCSRH0), offset 0x103

OTG A / Host

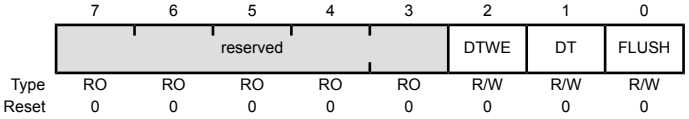
OTG B / Device

USBSR0H is an 8-bit register that provides control and status bits for endpoint 0.

OTG A / Host Mode

USB Control and Status Endpoint 0 High (USBCSRH0)

Base 0x4005.0000
Offset 0x103
Type W1C, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:3	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	DTWE	R/W	0	Data Toggle Write Enable Value Description 0 The DT bit cannot be written. 1 Enables the current state of the endpoint 0 data toggle to be written (see DT bit). This bit is automatically cleared once the new value is written.
1	DT	R/W	0	Data Toggle When read, this bit indicates the current state of the endpoint 0 data toggle. If DTWE is set, this bit may be written with the required setting of the data toggle. If DTWE is Low, this bit cannot be written. Care should be taken when writing to this bit as it should only be changed to RESET USB endpoint 0.

Bit/Field	Name	Type	Reset	Description
0	FLUSH	R/W	0	Flush FIFO

Value	Description
0	No effect.
1	Flushes the next packet to be transmitted/read from the endpoint 0 FIFO. The FIFO pointer is reset and the TXRDY/RXRDY bit is cleared.

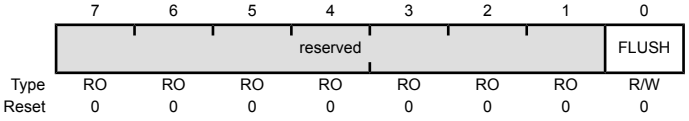
This bit is automatically cleared after the flush is performed.

Important: This bit should only be set when TXRDY/RXRDY is set. At other times, it may cause data to be corrupted.

OTG B / Device Mode

USB Control and Status Endpoint 0 High (USBCSRH0)

Base 0x4005.0000
Offset 0x103
Type W1C, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	FLUSH	R/W	0	Flush FIFO

Value	Description
0	No effect.
1	Flushes the next packet to be transmitted/read from the endpoint 0 FIFO. The FIFO pointer is reset and the TXRDY/RXRDY bit is cleared.

This bit is automatically cleared after the flush is performed.

Important: This bit should only be set when TXRDY/RXRDY is set. At other times, it may cause data to be corrupted.

Register 147: USB Receive Byte Count Endpoint 0 (USBCOUNT0), offset 0x108

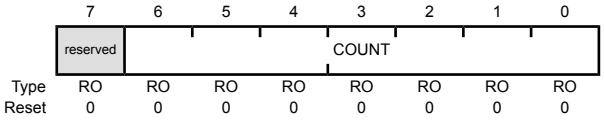
OTG A / Host

USBCOUNT0 is an 8-bit read-only register that indicates the number of received data bytes in the endpoint 0 FIFO. The value returned changes as the contents of the FIFO change and is only valid while the RXRDY bit is set.

OTG B / Device

USB Receive Byte Count Endpoint 0 (USBCOUNT0)

Base 0x4005.0000
Offset 0x108
Type RO, reset 0x00



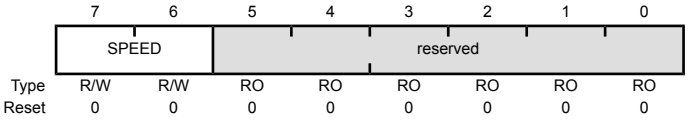
Bit/Field	Name	Type	Reset	Description
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	COUNT	RO	0x00	FIFO Count COUNT is a read-only value that indicates the number of received data bytes in the endpoint 0 FIFO.

Register 148: USB Type Endpoint 0 (USBTYPE0), offset 0x10A

OTG A / Host

This is an 8-bit register that must be written with the operating speed of the targeted Device being communicated with using endpoint 0.

USB Type Endpoint 0 (USBTYPE0)
Base 0x4005.0000
Offset 0x10A
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:6	SPEED	R/W	0x0	Operating Speed This field specifies the operating speed of the target Device. If selected, the target is assumed to have the same connection speed as the USB controller. Value / Description 0x0 - 0x1 / Reserved 0x2 / Full 0x3 / Low
5:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 149: USB NAK Limit (USBNAKLMT), offset 0x10B

OTG A / Host

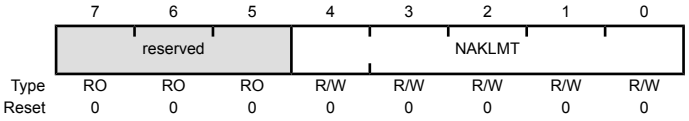
USBNAKLMT is an 8-bit register that sets the number of frames after which endpoint 0 should time out on receiving a stream of NAK responses. (Equivalent settings for other endpoints can be made through their **USBTXINTERVALn** and **USBRXINTERVALn** registers.)

The number of frames selected is $2^{(m-1)}$ (where *m* is the value set in the register, with valid values of 2–16). If the Host receives NAK responses from the target for more frames than the number represented by the limit set in this register, the endpoint is halted.

Note: A value of 0 or 1 disables the NAK timeout function.

USB NAK Limit (USBNAKLMT)

Base 0x4005.0000
Offset 0x10B
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4:0	NAKLMT	R/W	0x0	EP0 NAK Limit This field specifies the number of frames after receiving a stream of NAK responses.

Register 150: USB Transmit Control and Status Endpoint 1 Low (USBTXCSRL1), offset 0x112

Register 151: USB Transmit Control and Status Endpoint 2 Low (USBTXCSRL2), offset 0x122

Register 152: USB Transmit Control and Status Endpoint 3 Low (USBTXCSRL3), offset 0x132

Register 153: USB Transmit Control and Status Endpoint 4 Low (USBTXCSRL4), offset 0x142

Register 154: USB Transmit Control and Status Endpoint 5 Low (USBTXCSRL5), offset 0x152

Register 155: USB Transmit Control and Status Endpoint 6 Low (USBTXCSRL6), offset 0x162

Register 156: USB Transmit Control and Status Endpoint 7 Low (USBTXCSRL7), offset 0x172

Register 157: USB Transmit Control and Status Endpoint 8 Low (USBTXCSRL8), offset 0x182

Register 158: USB Transmit Control and Status Endpoint 9 Low (USBTXCSRL9), offset 0x192

Register 159: USB Transmit Control and Status Endpoint 10 Low (USBTXCSRL10), offset 0x1A2

Register 160: USB Transmit Control and Status Endpoint 11 Low (USBTXCSRL11), offset 0x1B2

Register 161: USB Transmit Control and Status Endpoint 12 Low (USBTXCSRL12), offset 0x1C2

Register 162: USB Transmit Control and Status Endpoint 13 Low (USBTXCSRL13), offset 0x1D2

Register 163: USB Transmit Control and Status Endpoint 14 Low (USBTXCSRL14), offset 0x1E2

Register 164: USB Transmit Control and Status Endpoint 15 Low (USBTXCSRL15), offset 0x1F2

OTG A / Host

OTG B / Device

USBTXCSRLn is an 8-bit register that provides control and status bits for transfers through the currently selected transmit endpoint.

OTG A / Host Mode

USB Transmit Control and Status Endpoint 1 Low (USBTXCSRL1)

Base 0x4005.0000
Offset 0x112
Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
	NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	NAKTO	R/W	0	NAK Timeout Value Description 0 No timeout. 1 *Bulk endpoints only:* Indicates that the transmit endpoint is halted following the receipt of NAK responses for longer than the time set by the NAKLMT field in the **USBTXINTERVALn** register. Software must clear this bit to allow the endpoint to continue.
6	CLRDT	R/W	0	Clear Data Toggle Writing a 1 to this bit clears the DT bit in the **USBTXCSRHn** register.
5	STALLED	R/W	0	Endpoint Stalled Value Description 0 A STALL handshake has not been received. 1 Indicates that a STALL handshake has been received. When this bit is set, any µDMA request that is in progress is stopped, the FIFO is completely flushed, and the TXRDY bit is cleared. Software must clear this bit.
4	SETUP	R/W	0	Setup Packet Value Description 0 No SETUP token is sent. 1 Sends a SETUP token instead of an OUT token for the transaction. This bit should be set at the same time as the TXRDY bit is set. **Note:** Setting this bit also clears the DT bit in the **USBTXCSRHn** register.

Bit/Field	Name	Type	Reset	Description
3	FLUSH	R/W	0	Flush FIFO

Value	Description
0	No effect.
1	Flushes the latest packet from the endpoint transmit FIFO. The FIFO pointer is reset and the TXRDY bit is cleared. The EPn bit in the **USBTXIS** register is also set in this situation.

This bit may be set simultaneously with the TXRDY bit to abort the packet that is currently being loaded into the FIFO. Note that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO.

Important: This bit should only be set when the TXRDY bit is set. At other times, it may cause data to be corrupted.

Bit/Field	Name	Type	Reset	Description
2	ERROR	R/W	0	Error

Value	Description
0	No error.
1	Three attempts have been made to send a packet and no handshake packet has been received. The TXRDY bit is cleared, the EPn bit in the **USBTXIS** register is set, and the FIFO is completely flushed in this situation.

Software must clear this bit.

Note: This is valid only when the endpoint is operating in Bulk or Interrupt mode.

Bit/Field	Name	Type	Reset	Description
1	FIFONE	R/W	0	FIFO Not Empty

Value	Description
0	The FIFO is empty.
1	At least one packet is in the transmit FIFO.

Bit/Field	Name	Type	Reset	Description
0	TXRDY	R/W	0	Transmit Packet Ready

Value	Description
0	No transmit packet is ready.
1	Software sets this bit after loading a data packet into the TX FIFO.

This bit is cleared automatically when a data packet has been transmitted. The EPn bit in the **USBTXIS** register is also set at this point. TXRDY is also automatically cleared prior to loading a second packet into a double-buffered FIFO.

OTG B / Device Mode

USB Transmit Control and Status Endpoint 1 Low (USBTXCSRL1)

Base 0x4005.0000
Offset 0x112
Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
	reserved	CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
Type	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	CLRDT	R/W	0	Clear Data Toggle Writing a 1 to this bit clears the DT bit in the **USBTXCSRHn** register.
5	STALLED	R/W	0	Endpoint Stalled Value Description 0 A STALL handshake has not been transmitted. 1 A STALL handshake has been transmitted. The FIFO is flushed and the TXRDY bit is cleared. Software must clear this bit.
4	STALL	R/W	0	Send STALL Value Description 0 No effect. 1 Issues a STALL handshake to an IN token. Software clears this bit to terminate the STALL condition. **Note:** This bit has no effect in isochronous transfers.
3	FLUSH	R/W	0	Flush FIFO Value Description 0 No effect. 1 Flushes the latest packet from the endpoint transmit FIFO. The FIFO pointer is reset and the TXRDY bit is cleared. The EPn bit in the **USBTXIS** register is also set in this situation. This bit may be set simultaneously with the TXRDY bit to abort the packet that is currently being loaded into the FIFO. Note that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO. **Important:** This bit should only be set when the TXRDY bit is set. At other times, it may cause data to be corrupted.

Bit/Field	Name	Type	Reset	Description
2	UNDRN	R/W	0	Underrun Value Description 0 No underrun. 1 An IN token has been received when TXRDY is not set. Software must clear this bit.
1	FIFONE	R/W	0	FIFO Not Empty Value Description 0 The FIFO is empty. 1 At least one packet is in the transmit FIFO.
0	TXRDY	R/W	0	Transmit Packet Ready Value Description 0 No transmit packet is ready. 1 Software sets this bit after loading a data packet into the TX FIFO. This bit is cleared automatically when a data packet has been transmitted. The EPn bit in the **USBTXIS** register is also set at this point. TXRDY is also automatically cleared prior to loading a second packet into a double-buffered FIFO.

Register 165: USB Transmit Control and Status Endpoint 1 High (USBTXCSRH1), offset 0x113

Register 166: USB Transmit Control and Status Endpoint 2 High (USBTXCSRH2), offset 0x123

Register 167: USB Transmit Control and Status Endpoint 3 High (USBTXCSRH3), offset 0x133

Register 168: USB Transmit Control and Status Endpoint 4 High (USBTXCSRH4), offset 0x143

Register 169: USB Transmit Control and Status Endpoint 5 High (USBTXCSRH5), offset 0x153

Register 170: USB Transmit Control and Status Endpoint 6 High (USBTXCSRH6), offset 0x163

Register 171: USB Transmit Control and Status Endpoint 7 High (USBTXCSRH7), offset 0x173

Register 172: USB Transmit Control and Status Endpoint 8 High (USBTXCSRH8), offset 0x183

Register 173: USB Transmit Control and Status Endpoint 9 High (USBTXCSRH9), offset 0x193

Register 174: USB Transmit Control and Status Endpoint 10 High (USBTXCSRH10), offset 0x1A3

Register 175: USB Transmit Control and Status Endpoint 11 High (USBTXCSRH11), offset 0x1B3

Register 176: USB Transmit Control and Status Endpoint 12 High (USBTXCSRH12), offset 0x1C3

Register 177: USB Transmit Control and Status Endpoint 13 High (USBTXCSRH13), offset 0x1D3

Register 178: USB Transmit Control and Status Endpoint 14 High (USBTXCSRH14), offset 0x1E3

Register 179: USB Transmit Control and Status Endpoint 15 High (USBTXCSRH15), offset 0x1F3

OTG A / Host

OTG B / Device

USBTXCSRHn is an 8-bit register that provides additional control for transfers through the currently selected transmit endpoint.

OTG A / Host Mode

USB Transmit Control and Status Endpoint 1 High (USBTXCSRH1)

Base 0x4005.0000
Offset 0x113
Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
	AUTOSET	reserved	MODE	DMAEN	FDT	DMAMOD	DTWE	DT
Type	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	AUTOSET	R/W	0	Auto Set Value Description 0 The TXRDY bit must be set manually. 1 Enables the TXRDY bit to be automatically set when data of the maximum packet size (value in **USBTXMAXPn**) is loaded into the transmit FIFO. If a packet of less than the maximum packet size is loaded, then the TXRDY bit must be set manually.
6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	MODE	R/W	0	Mode Value Description 0 Enables the endpoint direction as RX. 1 Enables the endpoint direction as TX. **Note:** This bit only has an effect when the same endpoint FIFO is used for both transmit and receive transactions.
4	DMAEN	R/W	0	DMA Request Enable Value Description 0 Disables the µDMA request for the transmit endpoint. 1 Enables the µDMA request for the transmit endpoint. **Note:** 3 TX and 3 /RX endpoints can be connected to the µDMA module. If this bit is set for a particular endpoint, the DMAATX, DMABTX, or DMACTX field in the **USB DMA Select (USBDMASEL)** register must be programmed correspondingly.
3	FDT	R/W	0	Force Data Toggle Value Description 0 No effect. 1 Forces the endpoint DT bit to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This bit can be used by interrupt transmit endpoints that are used to communicate rate feedback for isochronous endpoints.

Bit/Field	Name	Type	Reset	Description
2	DMAMOD	R/W	0	DMA Request Mode Value Description 0 An interrupt is generated after every µDMA packet transfer. 1 An interrupt is generated only after the entire µDMA transfer is complete. **Note:** This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared.
1	DTWE	R/W	0	Data Toggle Write Enable Value Description 0 The DT bit cannot be written. 1 Enables the current state of the transmit endpoint data to be written (see DT bit). This bit is automatically cleared once the new value is written.
0	DT	R/W	0	Data Toggle When read, this bit indicates the current state of the transmit endpoint data toggle. If DTWE is High, this bit may be written with the required setting of the data toggle. If DTWE is Low, any value written to this bit is ignored. Care should be taken when writing to this bit as it should only be changed to RESET the transmit endpoint.

OTG B / Device Mode

USB Transmit Control and Status Endpoint 1 High (USBTXCSRH1)

Base 0x4005.0000
Offset 0x113
Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
	AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD	reserved	
Type	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	AUTOSET	R/W	0	Auto Set Value Description 0 The TXRDY bit must be set manually. 1 Enables the TXRDY bit to be automatically set when data of the maximum packet size (value in **USBTXMAXPn**) is loaded into the transmit FIFO. If a packet of less than the maximum packet size is loaded, then the TXRDY bit must be set manually.

Bit/Field	Name	Type	Reset	Description
6	ISO	R/W	0	Isochronous Transfers Value Description 0 Enables the transmit endpoint for bulk or interrupt transfers. 1 Enables the transmit endpoint for isochronous transfers.
5	MODE	R/W	0	Mode Value Description 0 Enables the endpoint direction as RX. 1 Enables the endpoint direction as TX. **Note:** This bit only has an effect where the same endpoint FIFO is used for both transmit and receive transactions.
4	DMAEN	R/W	0	DMA Request Enable Value Description 0 Disables the µDMA request for the transmit endpoint. 1 Enables the µDMA request for the transmit endpoint. **Note:** 3 TX and 3 RX endpoints can be connected to the µDMA module. If this bit is set for a particular endpoint, the DMAATX, DMABTX, or DMACTX field in the **USB DMA Select (USBDMASEL)** register must be programmed correspondingly.
3	FDT	R/W	0	Force Data Toggle Value Description 0 No effect. 1 Forces the endpoint DT bit to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This bit can be used by interrupt transmit endpoints that are used to communicate rate feedback for isochronous endpoints.
2	DMAMOD	R/W	0	DMA Request Mode Value Description 0 An interrupt is generated after every µDMA packet transfer. 1 An interrupt is generated only after the entire µDMA transfer is complete. **Note:** This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared.
1:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 180: USB Maximum Receive Data Endpoint 1 (USBRXMAXP1), offset 0x114

Register 181: USB Maximum Receive Data Endpoint 2 (USBRXMAXP2), offset 0x124

Register 182: USB Maximum Receive Data Endpoint 3 (USBRXMAXP3), offset 0x134

Register 183: USB Maximum Receive Data Endpoint 4 (USBRXMAXP4), offset 0x144

Register 184: USB Maximum Receive Data Endpoint 5 (USBRXMAXP5), offset 0x154

Register 185: USB Maximum Receive Data Endpoint 6 (USBRXMAXP6), offset 0x164

Register 186: USB Maximum Receive Data Endpoint 7 (USBRXMAXP7), offset 0x174

Register 187: USB Maximum Receive Data Endpoint 8 (USBRXMAXP8), offset 0x184

Register 188: USB Maximum Receive Data Endpoint 9 (USBRXMAXP9), offset 0x194

Register 189: USB Maximum Receive Data Endpoint 10 (USBRXMAXP10), offset 0x1A4

Register 190: USB Maximum Receive Data Endpoint 11 (USBRXMAXP11), offset 0x1B4

Register 191: USB Maximum Receive Data Endpoint 12 (USBRXMAXP12), offset 0x1C4

Register 192: USB Maximum Receive Data Endpoint 13 (USBRXMAXP13), offset 0x1D4

Register 193: USB Maximum Receive Data Endpoint 14 (USBRXMAXP14), offset 0x1E4

Register 194: USB Maximum Receive Data Endpoint 15 (USBRXMAXP15), offset 0x1F4

OTG A / Host

OTG B / Device

The **USBRXMAXPn** is a 16-bit register which defines the maximum amount of data that can be transferred through the selected receive endpoint in a single operation.

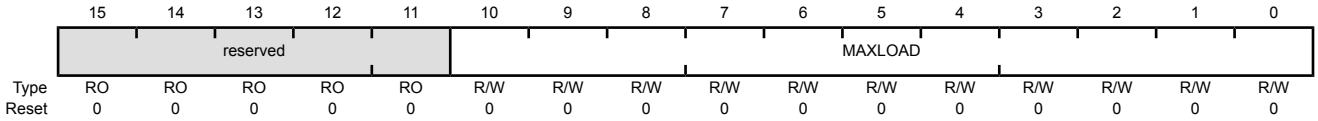
Bits 10:0 define (in bytes) the maximum payload transmitted in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the *USB Specification* on packet sizes for bulk, interrupt and isochronous transfers in full-speed operations.

The total amount of data represented by the value written to this register must not exceed the FIFO size for the receive endpoint, and must not exceed half the FIFO size if double-buffering is required.

Note: **USBRXMAXPn** must be set to an even number of bytes for proper interrupt generation in µDMA Basic mode.

USB Maximum Receive Data Endpoint 1 (USBRXMAXP1)

Base 0x4005.0000
Offset 0x114
Type R/W, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:11	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10:0	MAXLOAD	R/W	0x000	Maximum Payload The maximum payload in bytes per transaction.

Register 195: USB Receive Control and Status Endpoint 1 Low (USBRXCSRL1), offset 0x116

Register 196: USB Receive Control and Status Endpoint 2 Low (USBRXCSRL2), offset 0x126

Register 197: USB Receive Control and Status Endpoint 3 Low (USBRXCSRL3), offset 0x136

Register 198: USB Receive Control and Status Endpoint 4 Low (USBRXCSRL4), offset 0x146

Register 199: USB Receive Control and Status Endpoint 5 Low (USBRXCSRL5), offset 0x156

Register 200: USB Receive Control and Status Endpoint 6 Low (USBRXCSRL6), offset 0x166

Register 201: USB Receive Control and Status Endpoint 7 Low (USBRXCSRL7), offset 0x176

Register 202: USB Receive Control and Status Endpoint 8 Low (USBRXCSRL8), offset 0x186

Register 203: USB Receive Control and Status Endpoint 9 Low (USBRXCSRL9), offset 0x196

Register 204: USB Receive Control and Status Endpoint 10 Low (USBRXCSRL10), offset 0x1A6

Register 205: USB Receive Control and Status Endpoint 11 Low (USBRXCSRL11), offset 0x1B6

Register 206: USB Receive Control and Status Endpoint 12 Low (USBRXCSRL12), offset 0x1C6

Register 207: USB Receive Control and Status Endpoint 13 Low (USBRXCSRL13), offset 0x1D6

Register 208: USB Receive Control and Status Endpoint 14 Low (USBRXCSRL14), offset 0x1E6

Register 209: USB Receive Control and Status Endpoint 15 Low (USBRXCSRL15), offset 0x1F6

OTG A / Host

OTG B / Device

USBRXCSRLn is an 8-bit register that provides control and status bits for transfers through the currently selected receive endpoint.

OTG A / Host Mode

USB Receive Control and Status Endpoint 1 Low (USBRXCSRL1)

Base 0x4005.0000
Offset 0x116
Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
	CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
Type	W1C	R/W	R/W	R/W	R/W	R/W	RO	R/W
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	CLRDT	W1C	0	Clear Data Toggle Writing a 1 to this bit clears the DT bit in the **USBRXCSRHn** register.
6	STALLED	R/W	0	Endpoint Stalled Value Description 0 A STALL handshake has not been received. 1 A STALL handshake has been received. The EPn bit in the **USBRXIS** register is also set. Software must clear this bit.
5	REQPKT	R/W	0	Request Packet Value Description 0 No request. 1 Requests an IN transaction. This bit is cleared when RXRDY is set.
4	FLUSH	R/W	0	Flush FIFO Value Description 0 No effect. 1 Flushes the next packet to be read from the endpoint receive FIFO. The FIFO pointer is reset and the RXRDY bit is cleared. Note that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO. **Important:** This bit should only be set when the RXRDY bit is set. At other times, it may cause data to be corrupted.

Bit/Field	Name	Type	Reset	Description
3	DATAERR / NAKTO	R/W	0	Data Error / NAK Timeout Value Description 0 Normal operation. 1 *Isochronous endpoints only:* Indicates that RXRDY is set and the data packet has a CRC or bit-stuff error. This bit is cleared when RXRDY is cleared. *Bulk endpoints only:* Indicates that the receive endpoint is halted following the receipt of NAK responses for longer than the time set by the NAKLMT field in the **USBRXINTERVALn** register. Software must clear this bit to allow the endpoint to continue.
2	ERROR	R/W	0	Error Value Description 0 No error. 1 Three attempts have been made to receive a packet and no data packet has been received. The EPn bit in the **USBRXIS** register is set in this situation. Software must clear this bit. **Note:** This bit is only valid when the receive endpoint is operating in Bulk or Interrupt mode. In Isochronous mode, it always returns zero.
1	FULL	RO	0	FIFO Full Value Description 0 The receive FIFO is not full. 1 No more packets can be loaded into the receive FIFO.
0	RXRDY	R/W	0	Receive Packet Ready Value Description 0 No data packet has been received. 1 A data packet has been received. The EPn bit in the **USBRXIS** register is also set in this situation. If the AUTOCLR bit in the **USBRXCSRHn** register is set, then the this bit is automatically cleared when a packet of **USBRXMAXPn** bytes has been unloaded from the receive FIFO. If the AUTOCLR bit is clear, or if packets of less than the maximum packet size are unloaded, then software must clear this bit manually when the packet has been unloaded from the receive FIFO.

OTG B / Device Mode

USB Receive Control and Status Endpoint 1 Low (USBRXCSRL1)

Base 0x4005.0000
Offset 0x116
Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
	CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
Type	W1C	R/W	R/W	R/W	RO	R/W	RO	R/W
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	CLRDT	W1C	0	Clear Data Toggle Writing a 1 to this bit clears the DT bit in the **USBRXCSRHn** register.
6	STALLED	R/W	0	Endpoint Stalled Value Description 0 A STALL handshake has not been transmitted. 1 A STALL handshake has been transmitted. Software must clear this bit.
5	STALL	R/W	0	Send STALL Value Description 0 No effect. 1 Issues a STALL handshake. Software must clear this bit to terminate the STALL condition. **Note:** This bit has no effect where the endpoint is being used for isochronous transfers.
4	FLUSH	R/W	0	Flush FIFO Value Description 0 No effect. 1 Flushes the next packet from the endpoint receive FIFO. The FIFO pointer is reset and the RXRDY bit is cleared. The CPU writes a 1 to this bit to flush the next packet to be read from the endpoint receive FIFO. The FIFO pointer is reset and the RXRDY bit is cleared. Note that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO. **Important:** This bit should only be set when the RXRDY bit is set. At other times, it may cause data to be corrupted.

Bit/Field	Name	Type	Reset	Description
3	DATAERR	RO	0	Data Error Value Description 0 Normal operation. 1 Indicates that RXRDY is set and the data packet has a CRC or bit-stuff error. This bit is cleared when RXRDY is cleared. **Note:** This bit is only valid when the endpoint is operating in Isochronous mode. In Bulk mode, it always returns zero.
2	OVER	R/W	0	Overrun Value Description 0 No overrun error. 1 Indicates that an OUT packet cannot be loaded into the receive FIFO. Software must clear this bit. **Note:** This bit is only valid when the endpoint is operating in Isochronous mode. In Bulk mode, it always returns zero.
1	FULL	RO	0	FIFO Full Value Description 0 The receive FIFO is not full. 1 No more packets can be loaded into the receive FIFO.
0	RXRDY	R/W	0	Receive Packet Ready Value Description 0 No data packet has been received. 1 A data packet has been received. The EPn bit in the **USBRXIS** register is also set in this situation. If the AUTOCLR bit in the **USBRXCSRHn** register is set, then the this bit is automatically cleared when a packet of **USBRXMAXPn** bytes has been unloaded from the receive FIFO. If the AUTOCLR bit is clear, or if packets of less than the maximum packet size are unloaded, then software must clear this bit manually when the packet has been unloaded from the receive FIFO.

Register 210: USB Receive Control and Status Endpoint 1 High (USBRXCSRH1), offset 0x117

Register 211: USB Receive Control and Status Endpoint 2 High (USBRXCSRH2), offset 0x127

Register 212: USB Receive Control and Status Endpoint 3 High (USBRXCSRH3), offset 0x137

Register 213: USB Receive Control and Status Endpoint 4 High (USBRXCSRH4), offset 0x147

Register 214: USB Receive Control and Status Endpoint 5 High (USBRXCSRH5), offset 0x157

Register 215: USB Receive Control and Status Endpoint 6 High (USBRXCSRH6), offset 0x167

Register 216: USB Receive Control and Status Endpoint 7 High (USBRXCSRH7), offset 0x177

Register 217: USB Receive Control and Status Endpoint 8 High (USBRXCSRH8), offset 0x187

Register 218: USB Receive Control and Status Endpoint 9 High (USBRXCSRH9), offset 0x197

Register 219: USB Receive Control and Status Endpoint 10 High (USBRXCSRH10), offset 0x1A7

Register 220: USB Receive Control and Status Endpoint 11 High (USBRXCSRH11), offset 0x1B7

Register 221: USB Receive Control and Status Endpoint 12 High (USBRXCSRH12), offset 0x1C7

Register 222: USB Receive Control and Status Endpoint 13 High (USBRXCSRH13), offset 0x1D7

Register 223: USB Receive Control and Status Endpoint 14 High (USBRXCSRH14), offset 0x1E7

Register 224: USB Receive Control and Status Endpoint 15 High (USBRXCSRH15), offset 0x1F7

OTG A / Host

OTG B / Device

USBRXCSRHn is an 8-bit register that provides additional control and status bits for transfers through the currently selected receive endpoint.

OTG A / Host Mode

USB Receive Control and Status Endpoint 1 High (USBRXCSRH1)

Base 0x4005.0000
Offset 0x117
Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
	AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	reserved
Type	R/W	R/W	R/W	RO	R/W	RO	RO	RO
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	AUTOCL	R/W	0	Auto Clear

Value	Description
0	No effect.
1	Enables the RXRDY bit to be automatically cleared when a packet of **USBRXMAXPn** bytes has been unloaded from the receive FIFO. When packets of less than the maximum packet size are unloaded, RXRDY must be cleared manually. Care must be taken when using µDMA to unload the receive FIFO as data is read from the receive FIFO in 4 byte chunks regardless of the value of the MAXLOAD field in the **USBRXMAXPn** register, see "DMA Operation" on page 983.

Bit/Field	Name	Type	Reset	Description
6	AUTORQ	R/W	0	Auto Request

Value	Description
0	No effect.
1	Enables the REQPKT bit to be automatically set when the RXRDY bit is cleared.

Note: This bit is automatically cleared when a short packet is received.

Bit/Field	Name	Type	Reset	Description
5	DMAEN	R/W	0	DMA Request Enable

Value	Description
0	Disables the µDMA request for the receive endpoint.
1	Enables the µDMA request for the receive endpoint.

Note: 3 TX and 3 RX endpoints can be connected to the µDMA module. If this bit is set for a particular endpoint, the DMAARX, DMABRX, or DMACRX field in the **USB DMA Select (USBDMASEL)** register must be programmed correspondingly.

Bit/Field	Name	Type	Reset	Description
4	PIDERR	RO	0	PID Error

Value	Description
0	No error.
1	Indicates a PID error in the received packet of an isochronous transaction.

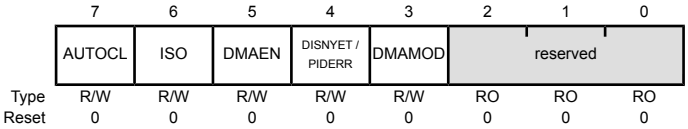
This bit is ignored in bulk or interrupt transactions.

Bit/Field	Name	Type	Reset	Description
3	DMAMOD	R/W	0	DMA Request Mode Value Description 0 An interrupt is generated after every µDMA packet transfer. 1 An interrupt is generated only after the entire µDMA transfer is complete. **Note:** This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared.
2	DTWE	RO	0	Data Toggle Write Enable Value Description 0 The DT bit cannot be written. 1 Enables the current state of the receive endpoint data to be written (see DT bit). This bit is automatically cleared once the new value is written.
1	DT	RO	0	Data Toggle When read, this bit indicates the current state of the receive data toggle. If DTWE is High, this bit may be written with the required setting of the data toggle. If DTWE is Low, any value written to this bit is ignored. Care should be taken when writing to this bit as it should only be changed to RESET the receive endpoint.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

OTG B / Device Mode

USB Receive Control and Status Endpoint 1 High (USBRXCSRH1)

Base 0x4005.0000
Offset 0x117
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	AUTOCL	R/W	0	Auto Clear Value Description 0 No effect. 1 Enables the RXRDY bit to be automatically cleared when a packet of **USBRXMAXPn** bytes has been unloaded from the receive FIFO. When packets of less than the maximum packet size are unloaded, RXRDY must be cleared manually. Care must be taken when using µDMA to unload the receive FIFO as data is read from the receive FIFO in 4 byte chunks regardless of the value of the MAXLOAD field in the **USBRXMAXPn** register, see "DMA Operation" on page 983.
6	ISO	R/W	0	Isochronous Transfers Value Description 0 Enables the receive endpoint for isochronous transfers. 1 Enables the receive endpoint for bulk/interrupt transfers.
5	DMAEN	R/W	0	DMA Request Enable Value Description 0 Disables the µDMA request for the receive endpoint. 1 Enables the µDMA request for the receive endpoint. **Note:** 3 TX and 3 RX endpoints can be connected to the µDMA module. If this bit is set for a particular endpoint, the DMAARX, DMABRX, or DMACRX field in the **USB DMA Select (USBDMASEL)** register must be programmed correspondingly.
4	DISNYET / PIDERR	R/W	0	Disable NYET / PID Error Value Description 0 No effect. 1 *For bulk or interrupt transactions:* Disables the sending of NYET handshakes. When this bit is set, all successfully received packets are acknowledged, including at the point at which the FIFO becomes full. *For isochronous transactions:* Indicates a PID error in the received packet.
3	DMAMOD	R/W	0	DMA Request Mode Value Description 0 An interrupt is generated after every µDMA packet transfer. 1 An interrupt is generated only after the entire µDMA transfer is complete. **Note:** This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared.

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

Register 225: USB Receive Byte Count Endpoint 1 (USBRXCOUNT1), offset 0x118

Register 226: USB Receive Byte Count Endpoint 2 (USBRXCOUNT2), offset 0x128

Register 227: USB Receive Byte Count Endpoint 3 (USBRXCOUNT3), offset 0x138

Register 228: USB Receive Byte Count Endpoint 4 (USBRXCOUNT4), offset 0x148

Register 229: USB Receive Byte Count Endpoint 5 (USBRXCOUNT5), offset 0x158

Register 230: USB Receive Byte Count Endpoint 6 (USBRXCOUNT6), offset 0x168

Register 231: USB Receive Byte Count Endpoint 7 (USBRXCOUNT7), offset 0x178

Register 232: USB Receive Byte Count Endpoint 8 (USBRXCOUNT8), offset 0x188

Register 233: USB Receive Byte Count Endpoint 9 (USBRXCOUNT9), offset 0x198

Register 234: USB Receive Byte Count Endpoint 10 (USBRXCOUNT10), offset 0x1A8

Register 235: USB Receive Byte Count Endpoint 11 (USBRXCOUNT11), offset 0x1B8

Register 236: USB Receive Byte Count Endpoint 12 (USBRXCOUNT12), offset 0x1C8

Register 237: USB Receive Byte Count Endpoint 13 (USBRXCOUNT13), offset 0x1D8

Register 238: USB Receive Byte Count Endpoint 14 (USBRXCOUNT14), offset 0x1E8

Register 239: USB Receive Byte Count Endpoint 15 (USBRXCOUNT15), offset 0x1F8

OTG A / Host

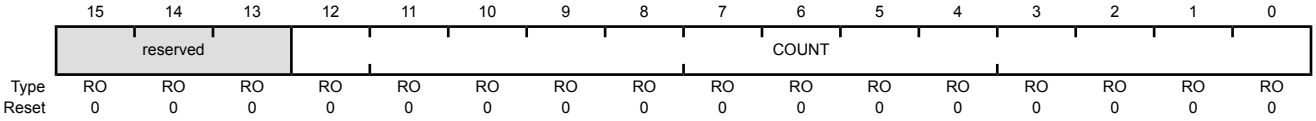
Note: The value returned changes as the FIFO is unloaded and is only valid while the RXRDY bit in the **USBRXCSRLn** register is set.

OTG B / Device

USBRXCOUNTn is a 16-bit read-only register that holds the number of data bytes in the packet currently in line to be read from the receive FIFO. If the packet is transmitted as multiple bulk packets, the number given is for the combined packet.

USB Receive Byte Count Endpoint 1 (USBRXCOUNT1)

Base 0x4005.0000
Offset 0x118
Type RO, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:13	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12:0	COUNT	RO	0x000	Receive Packet Count Indicates the number of bytes in the receive packet.

Register 240: USB Host Transmit Configure Type Endpoint 1 (USBTXTYPE1), offset 0x11A

Register 241: USB Host Transmit Configure Type Endpoint 2 (USBTXTYPE2), offset 0x12A

Register 242: USB Host Transmit Configure Type Endpoint 3 (USBTXTYPE3), offset 0x13A

Register 243: USB Host Transmit Configure Type Endpoint 4 (USBTXTYPE4), offset 0x14A

Register 244: USB Host Transmit Configure Type Endpoint 5 (USBTXTYPE5), offset 0x15A

Register 245: USB Host Transmit Configure Type Endpoint 6 (USBTXTYPE6), offset 0x16A

Register 246: USB Host Transmit Configure Type Endpoint 7 (USBTXTYPE7), offset 0x17A

Register 247: USB Host Transmit Configure Type Endpoint 8 (USBTXTYPE8), offset 0x18A

Register 248: USB Host Transmit Configure Type Endpoint 9 (USBTXTYPE9), offset 0x19A

Register 249: USB Host Transmit Configure Type Endpoint 10 (USBTXTYPE10), offset 0x1AA

Register 250: USB Host Transmit Configure Type Endpoint 11 (USBTXTYPE11), offset 0x1BA

Register 251: USB Host Transmit Configure Type Endpoint 12 (USBTXTYPE12), offset 0x1CA

Register 252: USB Host Transmit Configure Type Endpoint 13 (USBTXTYPE13), offset 0x1DA

Register 253: USB Host Transmit Configure Type Endpoint 14 (USBTXTYPE14), offset 0x1EA

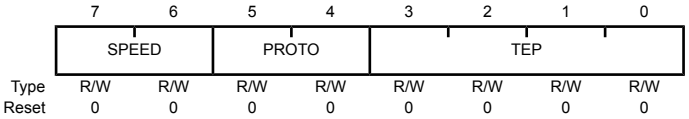
Register 254: USB Host Transmit Configure Type Endpoint 15 (USBTXTYPE15), offset 0x1FA

OTG A / Host

USBTXTYPEn is an 8-bit register that must be written with the endpoint number to be targeted by the endpoint, the transaction protocol to use for the currently selected transmit endpoint, and its operating speed.

USB Host Transmit Configure Type Endpoint 1 (USBTXTYPE1)

Base 0x4005.0000
Offset 0x11A
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:6	SPEED	R/W	0x0	Operating Speed This bit field specifies the operating speed of the target Device: Value Description 0x0 Default The target is assumed to be using the same connection speed as the USB controller. 0x1 Reserved 0x2 Full 0x3 Low
5:4	PROTO	R/W	0x0	Protocol Software must configure this bit field to select the required protocol for the transmit endpoint: Value Description 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt
3:0	TEP	R/W	0x0	Target Endpoint Number Software must configure this value to the endpoint number contained in the transmit endpoint descriptor returned to the USB controller during Device enumeration.

Register 255: USB Host Transmit Interval Endpoint 1 (USBTXINTERVAL1), offset 0x11B

Register 256: USB Host Transmit Interval Endpoint 2 (USBTXINTERVAL2), offset 0x12B

Register 257: USB Host Transmit Interval Endpoint 3 (USBTXINTERVAL3), offset 0x13B

Register 258: USB Host Transmit Interval Endpoint 4 (USBTXINTERVAL4), offset 0x14B

Register 259: USB Host Transmit Interval Endpoint 5 (USBTXINTERVAL5), offset 0x15B

Register 260: USB Host Transmit Interval Endpoint 6 (USBTXINTERVAL6), offset 0x16B

Register 261: USB Host Transmit Interval Endpoint 7 (USBTXINTERVAL7), offset 0x17B

Register 262: USB Host Transmit Interval Endpoint 8 (USBTXINTERVAL8), offset 0x18B

Register 263: USB Host Transmit Interval Endpoint 9 (USBTXINTERVAL9), offset 0x19B

Register 264: USB Host Transmit Interval Endpoint 10 (USBTXINTERVAL10), offset 0x1AB

Register 265: USB Host Transmit Interval Endpoint 11 (USBTXINTERVAL11), offset 0x1BB

Register 266: USB Host Transmit Interval Endpoint 12 (USBTXINTERVAL12), offset 0x1CB

Register 267: USB Host Transmit Interval Endpoint 13 (USBTXINTERVAL13), offset 0x1DB

Register 268: USB Host Transmit Interval Endpoint 14 (USBTXINTERVAL14), offset 0x1EB

Register 269: USB Host Transmit Interval Endpoint 15 (USBTXINTERVAL15), offset 0x1FB

OTG A / Host

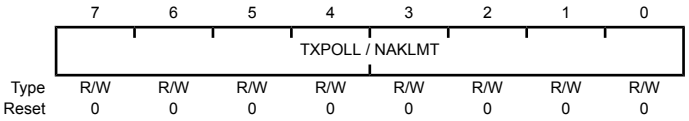
USBTXINTERVALn is an 8-bit register that, for interrupt and isochronous transfers, defines the polling interval for the currently selected transmit endpoint. For bulk endpoints, this register defines the number of frames after which the endpoint should time out on receiving a stream of NAK responses.

The **USBTXINTERVALn** register value defines a number of frames, as follows:

Transfer Type	Speed	Valid values (m)	Interpretation
Interrupt	Low-Speed or Full-Speed	0x01 – 0xFF	The polling interval is *m* frames.
Isochronous	Full-Speed	0x01 – 0x10	The polling interval is $2^{(m-1)}$ frames.
Bulk	Full-Speed	0x02 – 0x10	The NAK Limit is $2^{(m-1)}$ frames. A value of 0 or 1 disables the NAK timeout function.

USB Host Transmit Interval Endpoint 1 (USBTXINTERVAL1)

Base 0x4005.0000
Offset 0x11B
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:0	TXPOLL / NAKLMT	R/W	0x00	TX Polling / NAK Limit The polling interval for interrupt/isochronous transfers; the NAK limit for bulk transfers. See table above for valid entries; other values are reserved.

Register 270: USB Host Configure Receive Type Endpoint 1 (USBRXTYPE1), offset 0x11C

Register 271: USB Host Configure Receive Type Endpoint 2 (USBRXTYPE2), offset 0x12C

Register 272: USB Host Configure Receive Type Endpoint 3 (USBRXTYPE3), offset 0x13C

Register 273: USB Host Configure Receive Type Endpoint 4 (USBRXTYPE4), offset 0x14C

Register 274: USB Host Configure Receive Type Endpoint 5 (USBRXTYPE5), offset 0x15C

Register 275: USB Host Configure Receive Type Endpoint 6 (USBRXTYPE6), offset 0x16C

Register 276: USB Host Configure Receive Type Endpoint 7 (USBRXTYPE7), offset 0x17C

Register 277: USB Host Configure Receive Type Endpoint 8 (USBRXTYPE8), offset 0x18C

Register 278: USB Host Configure Receive Type Endpoint 9 (USBRXTYPE9), offset 0x19C

Register 279: USB Host Configure Receive Type Endpoint 10 (USBRXTYPE10), offset 0x1AC

Register 280: USB Host Configure Receive Type Endpoint 11 (USBRXTYPE11), offset 0x1BC

Register 281: USB Host Configure Receive Type Endpoint 12 (USBRXTYPE12), offset 0x1CC

Register 282: USB Host Configure Receive Type Endpoint 13 (USBRXTYPE13), offset 0x1DC

Register 283: USB Host Configure Receive Type Endpoint 14 (USBRXTYPE14), offset 0x1EC

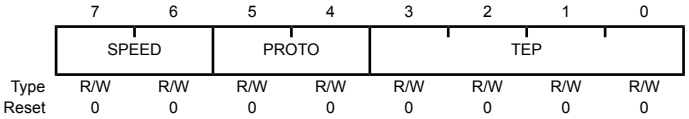
Register 284: USB Host Configure Receive Type Endpoint 15 (USBRXTYPE15), offset 0x1FC

OTG A / Host

USBRXTYPEn is an 8-bit register that must be written with the endpoint number to be targeted by the endpoint, the transaction protocol to use for the currently selected receive endpoint, and its operating speed.

USB Host Configure Receive Type Endpoint 1 (USBRXTYPE1)

Base 0x4005.0000
Offset 0x11C
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:6	SPEED	R/W	0x0	Operating Speed This bit field specifies the operating speed of the target Device: Value Description 0x0 Default The target is assumed to be using the same connection speed as the USB controller. 0x1 Reserved 0x2 Full 0x3 Low
5:4	PROTO	R/W	0x0	Protocol Software must configure this bit field to select the required protocol for the receive endpoint: Value Description 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt
3:0	TEP	R/W	0x0	Target Endpoint Number Software must set this value to the endpoint number contained in the receive endpoint descriptor returned to the USB controller during Device enumeration.

Register 285: USB Host Receive Polling Interval Endpoint 1 (USBRXINTERVAL1), offset 0x11D

Register 286: USB Host Receive Polling Interval Endpoint 2 (USBRXINTERVAL2), offset 0x12D

Register 287: USB Host Receive Polling Interval Endpoint 3 (USBRXINTERVAL3), offset 0x13D

Register 288: USB Host Receive Polling Interval Endpoint 4 (USBRXINTERVAL4), offset 0x14D

Register 289: USB Host Receive Polling Interval Endpoint 5 (USBRXINTERVAL5), offset 0x15D

Register 290: USB Host Receive Polling Interval Endpoint 6 (USBRXINTERVAL6), offset 0x16D

Register 291: USB Host Receive Polling Interval Endpoint 7 (USBRXINTERVAL7), offset 0x17D

Register 292: USB Host Receive Polling Interval Endpoint 8 (USBRXINTERVAL8), offset 0x18D

Register 293: USB Host Receive Polling Interval Endpoint 9 (USBRXINTERVAL9), offset 0x19D

Register 294: USB Host Receive Polling Interval Endpoint 10 (USBRXINTERVAL10), offset 0x1AD

Register 295: USB Host Receive Polling Interval Endpoint 11 (USBRXINTERVAL11), offset 0x1BD

Register 296: USB Host Receive Polling Interval Endpoint 12 (USBRXINTERVAL12), offset 0x1CD

Register 297: USB Host Receive Polling Interval Endpoint 13 (USBRXINTERVAL13), offset 0x1DD

Register 298: USB Host Receive Polling Interval Endpoint 14 (USBRXINTERVAL14), offset 0x1ED

Register 299: USB Host Receive Polling Interval Endpoint 15 (USBRXINTERVAL15), offset 0x1FD

OTG A / Host

USBRXINTERVALn is an 8-bit register that, for interrupt and isochronous transfers, defines the polling interval for the currently selected receive endpoint. For bulk endpoints, this register defines the number of frames after which the endpoint should time out on receiving a stream of NAK responses.

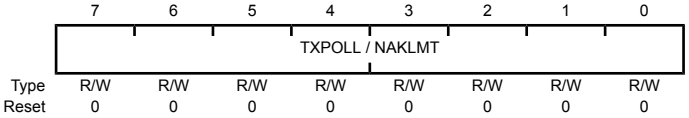
The **USBRXINTERVALn** register value defines a number of frames, as follows:

Transfer Type	Speed	Valid values (m)	Interpretation
Interrupt	Low-Speed or Full-Speed	0x01 – 0xFF	The polling interval is *m* frames.
Isochronous	Full-Speed	0x01 – 0x10	The polling interval is $2^{(m-1)}$ frames.

Transfer Type	Speed	Valid values (m)	Interpretation
Bulk	Full-Speed	0x02 – 0x10	The NAK Limit is $2^{(m-1)}$ frames. A value of 0 or 1 disables the NAK timeout function.

USB Host Receive Polling Interval Endpoint 1 (USBRXINTERVAL1)

Base 0x4005.0000
Offset 0x11D
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:0	TXPOLL / NAKLMT	R/W	0x00	RX Polling / NAK Limit The polling interval for interrupt/isochronous transfers; the NAK limit for bulk transfers. See table above for valid entries; other values are reserved.

Register 300: USB Request Packet Count in Block Transfer Endpoint 1 (USBRQPKTCOUNT1), offset 0x304

Register 301: USB Request Packet Count in Block Transfer Endpoint 2 (USBRQPKTCOUNT2), offset 0x308

Register 302: USB Request Packet Count in Block Transfer Endpoint 3 (USBRQPKTCOUNT3), offset 0x30C

Register 303: USB Request Packet Count in Block Transfer Endpoint 4 (USBRQPKTCOUNT4), offset 0x310

Register 304: USB Request Packet Count in Block Transfer Endpoint 5 (USBRQPKTCOUNT5), offset 0x314

Register 305: USB Request Packet Count in Block Transfer Endpoint 6 (USBRQPKTCOUNT6), offset 0x318

Register 306: USB Request Packet Count in Block Transfer Endpoint 7 (USBRQPKTCOUNT7), offset 0x31C

Register 307: USB Request Packet Count in Block Transfer Endpoint 8 (USBRQPKTCOUNT8), offset 0x320

Register 308: USB Request Packet Count in Block Transfer Endpoint 9 (USBRQPKTCOUNT9), offset 0x324

Register 309: USB Request Packet Count in Block Transfer Endpoint 10 (USBRQPKTCOUNT10), offset 0x328

Register 310: USB Request Packet Count in Block Transfer Endpoint 11 (USBRQPKTCOUNT11), offset 0x32C

Register 311: USB Request Packet Count in Block Transfer Endpoint 12 (USBRQPKTCOUNT12), offset 0x330

Register 312: USB Request Packet Count in Block Transfer Endpoint 13 (USBRQPKTCOUNT13), offset 0x334

Register 313: USB Request Packet Count in Block Transfer Endpoint 14 (USBRQPKTCOUNT14), offset 0x338

Register 314: USB Request Packet Count in Block Transfer Endpoint 15 (USBRQPKTCOUNT15), offset 0x33C

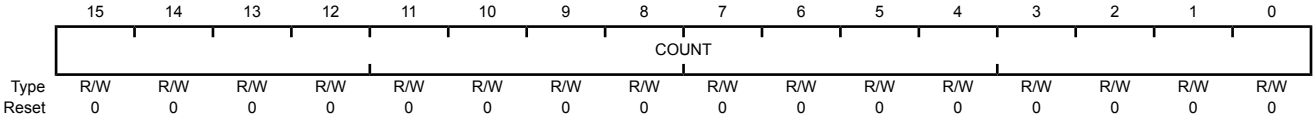
OTG A / Host

This 16-bit read/write register is used in Host mode to specify the number of packets that are to be transferred in a block transfer of one or more bulk packets to receive endpoint n. The USB controller uses the value recorded in this register to determine the number of requests to issue where the AUTORQ bit in the **USBRXCSRHn** register has been set. See “IN Transactions as a Host” on page 978.

Note: Multiple packets combined into a single bulk packet within the FIFO count as one packet.

USB Request Packet Count in Block Transfer Endpoint 1 (USBRQPKTCOUNT1)

Base 0x4005.0000
Offset 0x304
Type R/W, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:0	COUNT	R/W	0x0000	Block Transfer Packet Count Sets the number of packets of the size defined by the MAXLOAD bit field that are to be transferred in a block transfer. **Note:** This is only used in Host mode when AUTORQ is set. The bit has no effect in Device mode or when AUTORQ is not set.

Register 315: USB Receive Double Packet Buffer Disable (USBRXDPKTBUFDIS), offset 0x340

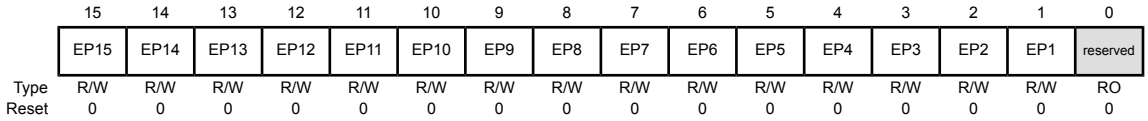
OTG A / Host

OTG B / Device

USBRXDPKTBUFDIS is a 16-bit register that indicates which of the receive endpoints have disabled the double-packet buffer functionality (see the section called “Double-Packet Buffering” on page 975).

USB Receive Double Packet Buffer Disable (USBRXDPKTBUFDIS)

Base 0x4005.0000
Offset 0x340
Type R/W, reset 0x0000



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	reserved
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
15	EP15	R/W	0	EP15 RX Double-Packet Buffer Disable Value Description 0 Disables double-packet buffering. 1 Enables double-packet buffering.
14	EP14	R/W	0	EP14 RX Double-Packet Buffer Disable Same description as EP15.
13	EP13	R/W	0	EP13 RX Double-Packet Buffer Disable Same description as EP15.
12	EP12	R/W	0	EP12 RX Double-Packet Buffer Disable Same description as EP15.
11	EP11	R/W	0	EP11 RX Double-Packet Buffer Disable Same description as EP15.
10	EP10	R/W	0	EP10 RX Double-Packet Buffer Disable Same description as EP15.
9	EP9	R/W	0	EP9 RX Double-Packet Buffer Disable Same description as EP15.
8	EP8	R/W	0	EP8 RX Double-Packet Buffer Disable Same description as EP15.
7	EP7	R/W	0	EP7 RX Double-Packet Buffer Disable Same description as EP15.
6	EP6	R/W	0	EP6 RX Double-Packet Buffer Disable Same description as EP15.
5	EP5	R/W	0	EP5 RX Double-Packet Buffer Disable Same description as EP15.
4	EP4	R/W	0	EP4 RX Double-Packet Buffer Disable Same description as EP15.

Bit/Field	Name	Type	Reset	Description
3	EP3	R/W	0	EP3 RX Double-Packet Buffer Disable Same description as EP15.
2	EP2	R/W	0	EP2 RX Double-Packet Buffer Disable Same description as EP15.
1	EP1	R/W	0	EP1 RX Double-Packet Buffer Disable Same description as EP15.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 316: USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFDIS), offset 0x342

OTG A / Host

OTG B / Device

USBTXDPKTBUFDIS is a 16-bit register that indicates which of the transmit endpoints have disabled the double-packet buffer functionality (see the section called “Double-Packet Buffering” on page 974).

USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFDIS)

Base 0x4005.0000
Offset 0x342
Type R/W, reset 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	reserved
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
15	EP15	R/W	0	EP15 TX Double-Packet Buffer Disable Value Description 0 Disables double-packet buffering. 1 Enables double-packet buffering.
14	EP14	R/W	0	EP14 TX Double-Packet Buffer Disable Same description as EP15.
13	EP13	R/W	0	EP13 TX Double-Packet Buffer Disable Same description as EP15.
12	EP12	R/W	0	EP12 TX Double-Packet Buffer Disable Same description as EP15.
11	EP11	R/W	0	EP11 TX Double-Packet Buffer Disable Same description as EP15.
10	EP10	R/W	0	EP10 TX Double-Packet Buffer Disable Same description as EP15.
9	EP9	R/W	0	EP9 TX Double-Packet Buffer Disable Same description as EP15.
8	EP8	R/W	0	EP8 TX Double-Packet Buffer Disable Same description as EP15.
7	EP7	R/W	0	EP7 TX Double-Packet Buffer Disable Same description as EP15.
6	EP6	R/W	0	EP6 TX Double-Packet Buffer Disable Same description as EP15.
5	EP5	R/W	0	EP5 TX Double-Packet Buffer Disable Same description as EP15.
4	EP4	R/W	0	EP4 TX Double-Packet Buffer Disable Same description as EP15.

Bit/Field	Name	Type	Reset	Description
3	EP3	R/W	0	EP3 TX Double-Packet Buffer Disable Same description as EP15.
2	EP2	R/W	0	EP2 TX Double-Packet Buffer Disable Same description as EP15.
1	EP1	R/W	0	EP1 TX Double-Packet Buffer Disable Same description as EP15.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 317: USB External Power Control (USBEPC), offset 0x400

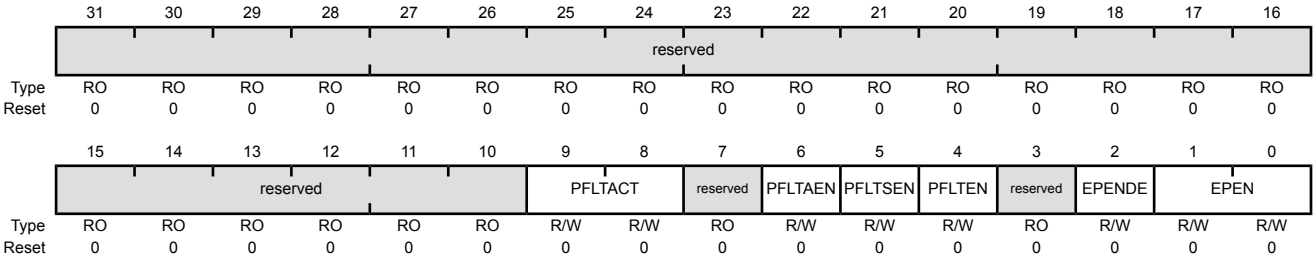
OTG A / Host

This 32-bit register specifies the function of the two-pin external power interface (USB0EPEN and USB0PFLT). The assertion of the power fault input may generate an automatic action, as controlled by the hardware configuration registers. The automatic action is necessary because the fault condition may require a response faster than one provided by firmware.

OTG B / Device

USB External Power Control (USBEPC)

Base 0x4005.0000
Offset 0x400
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:8	PFLTACT	R/W	0x0	Power Fault Action This bit field specifies how the USB0EPEN signal is changed when detecting a USB power fault. Value Description 0x0 Unchanged USB0EPEN is controlled by the combination of the EPEN and EPENDE bits. 0x1 Tristate USB0EPEN is undriven (tristate). 0x2 Low USB0EPEN is driven Low. 0x3 High USB0EPEN is driven High.
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
6	PFLTAEN	R/W	0	Power Fault Action Enable This bit specifies whether a USB power fault triggers any automatic corrective action regarding the driven state of the USB0EPEN signal. Value Description 0 Disabled USB0EPEN is controlled by the combination of the EPEN and EPENDE bits. 1 Enabled The USB0EPEN output is automatically changed to the state specified by the PFLTACT field.
5	PFLTSEN	R/W	0	Power Fault Sense This bit specifies the logical sense of the USB0PFLT input signal that indicates an error condition. The complementary state is the inactive state. Value Description 0 Low Fault If USB0PFLT is driven Low, the power fault is signaled internally (if enabled by the PFLTEN bit). 1 High Fault If USB0PFLT is driven High, the power fault is signaled internally (if enabled by the PFLTEN bit).
4	PFLTEN	R/W	0	Power Fault Input Enable This bit specifies whether the USB0PFLT input signal is used in internal logic. Value Description 0 Not Used The USB0PFLT signal is ignored. 1 Used The USB0PFLT signal is used internally.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
2	EPENDE	R/W	0	EPEN Drive Enable This bit specifies whether the USB0EPEN signal is driven or undriven (tristate). When driven, the signal value is specified by the EPEN field. When not driven, the EPEN field is ignored and the USB0EPEN signal is placed in a high-impedance state. Value Description 0 Not Driven The USB0EPEN signal is high impedance. 1 Driven The USB0EPEN signal is driven to the logical value specified by the value of the EPEN field. The USB0EPEN signal is undriven at reset because the sense of the external power supply enable is unknown. By adding the high-impedance state, system designers may bias the power supply enable to the disabled state using a large resistor (100 kΩ) and later configure and drive the output signal to enable the power supply.
1:0	EPEN	R/W	0x0	External Power Supply Enable Configuration This bit field specifies and controls the logical value driven on the USB0EPEN signal. Value Description 0x0 Power Enable Active Low The USB0EPEN signal is driven Low if the EPENDE bit is set. 0x1 Power Enable Active High The USB0EPEN signal is driven High if the EPENDE bit is set. 0x2 Power Enable High if VBUS Low The USB0EPEN signal is driven High when the A device is not recognized. 0x3 Power Enable High if VBUS High The USB0EPEN signal is driven High when the A device is recognized.

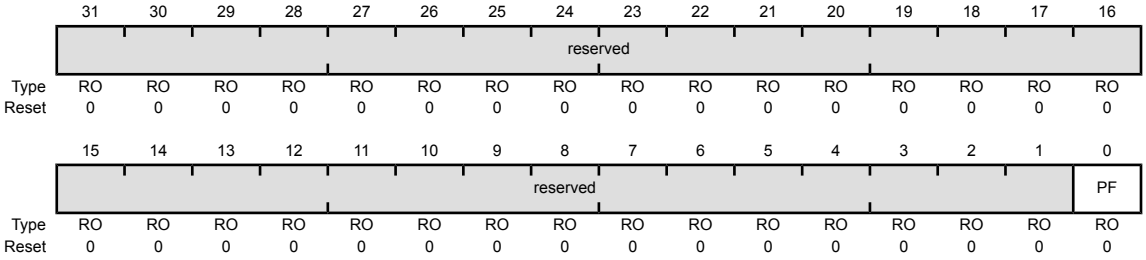
Register 318: USB External Power Control Raw Interrupt Status (USBEPCRIS), offset 0x404

OTG A / Host

OTG B / Device

This 32-bit register specifies the unmasked interrupt status of the two-pin external power interface.

USB External Power Control Raw Interrupt Status (USBEPCRIS)
Base 0x4005.0000
Offset 0x404
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	PF	RO	0	USB Power Fault Interrupt Status Value Description 1 A Power Fault status has been detected. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the PF bit in the **USBEPCISC** register.

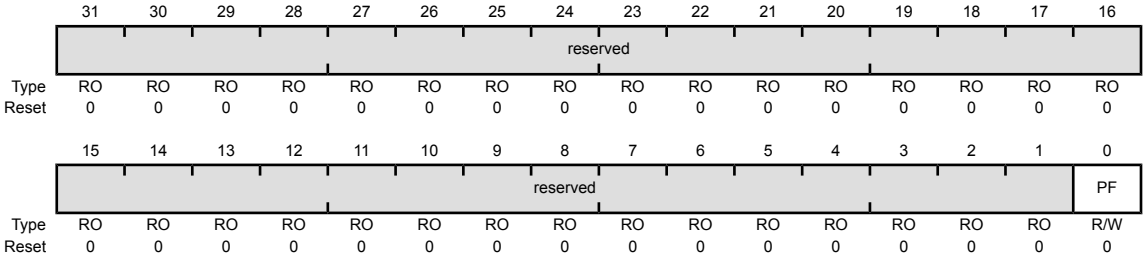
Register 319: USB External Power Control Interrupt Mask (USBEPCIM), offset 0x408

OTG A / Host

OTG B / Device

This 32-bit register specifies the interrupt mask of the two-pin external power interface.

USB External Power Control Interrupt Mask (USBEPCIM)
Base 0x4005.0000
Offset 0x408
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	PF	R/W	0	USB Power Fault Interrupt Mask

Value	Description
1	The raw interrupt signal from a detected power fault is sent to the interrupt controller.
0	A detected power fault does not affect the interrupt status.

Register 320: USB External Power Control Interrupt Status and Clear (USBEPCISC), offset 0x40C

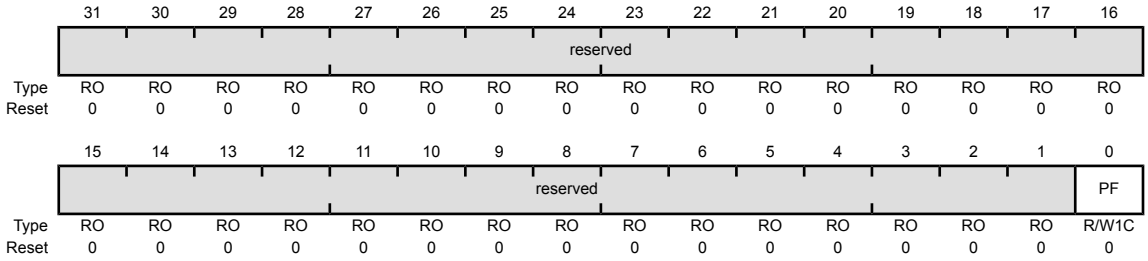
OTG A / Host

OTG B / Device

This 32-bit register specifies the masked interrupt status of the two-pin external power interface. It also provides a method to clear the interrupt state.

USB External Power Control Interrupt Status and Clear (USBEPCISC)

Base 0x4005.0000
Offset 0x40C
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	PF	R/W1C	0	USB Power Fault Interrupt Status and Clear Value Description 1 The PF bits in the **USBEPCRIS** and **USBEPCIM** registers are set, providing an interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the PF bit in the **USBEPCRIS** register.

Register 321: USB Device RESUME Raw Interrupt Status (USBDRRIS), offset 0x410

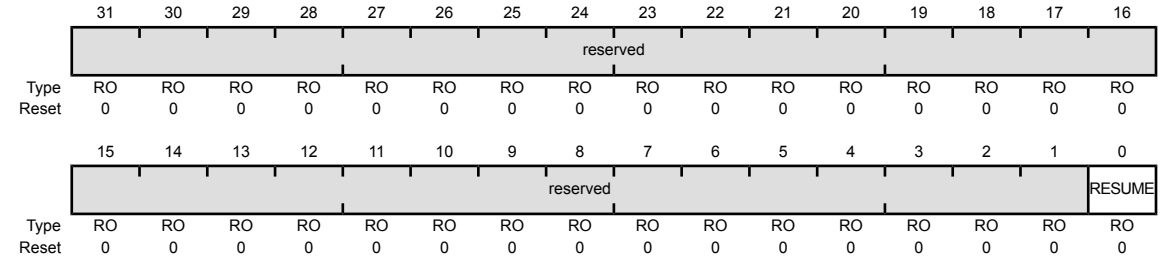
OTG A / Host

OTG B / Device

The **USBDRRIS** 32-bit register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect.

USB Device RESUME Raw Interrupt Status (USBDRRIS)

Base 0x4005.0000
Offset 0x410
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	RESUME	RO	0	RESUME Interrupt Status Value Description 1 A RESUME status has been detected. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the RESUME bit in the **USBDRISC** register.

Register 322: USB Device RESUME Interrupt Mask (USBDRIM), offset 0x414

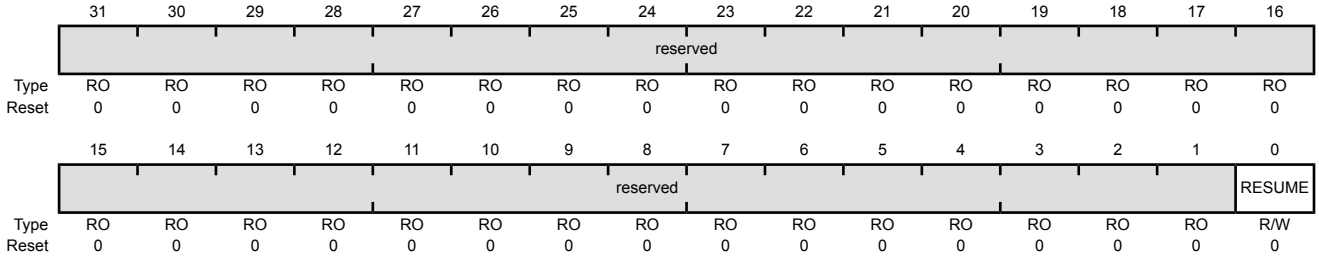
OTG A / Host

The **USBDRIM** 32-bit register is the masked interrupt status register. On a read, this register gives the current value of the mask on the corresponding interrupt. Setting a bit sets the mask, preventing the interrupt from being signaled to the interrupt controller. Clearing a bit clears the corresponding mask, enabling the interrupt to be sent to the interrupt controller.

OTG B / Device

USB Device RESUME Interrupt Mask (USBDRIM)

Base 0x4005.0000
Offset 0x414
Type R/W, reset 0x0000.0000



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

Value	Description
1	The raw interrupt signal from a detected RESUME is sent to the interrupt controller. This bit should only be set when a SUSPEND has been detected (the SUSPEND bit in the **USBIS** register is set).
0	A detected RESUME does not affect the interrupt status.

Register 323: USB Device RESUME Interrupt Status and Clear (USBDRISC), offset 0x418

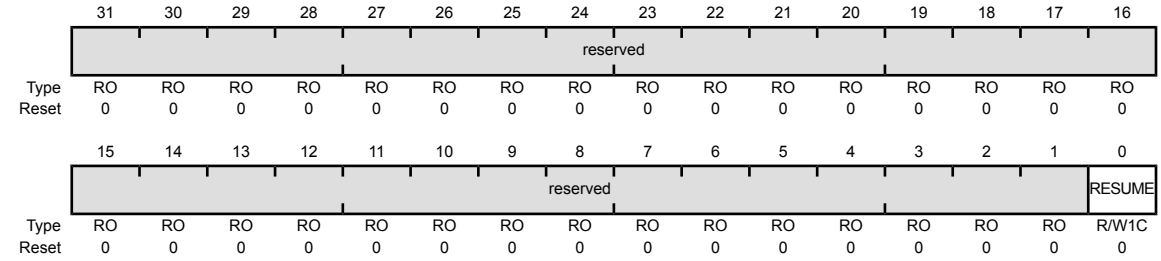
OTG A / Host

OTG B / Device

The **USBDRISC** 32-bit register is the interrupt clear register. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

USB Device RESUME Interrupt Status and Clear (USBDRISC)

Base 0x4005.0000
Offset 0x418
Type W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	RESUME	R/W1C	0	RESUME Interrupt Status and Clear Value Description 1 The RESUME bits in the **USBDRRIS** and **USBDRCIM** registers are set, providing an interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the RESUME bit in the **USBDRCRIS** register.

Register 324: USB General-Purpose Control and Status (USBGPCS), offset 0x41C

OTG A / Host

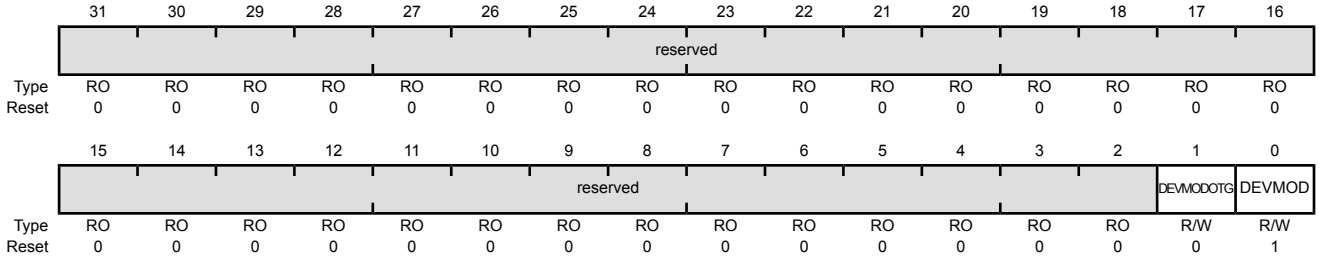
OTG B / Device

USBGPCS provides the state of the internal ID signal.

Note: When used in OTG mode, USB0VBUS and USB0ID do not require any configuration as they are dedicated pins for the USB controller and directly connect to the USB connector's VBUS and ID signals. If the USB controller is used as either a dedicated Host or Device, the DEVMODOTG and DEVMOD bits in the **USB General-Purpose Control and Status (USBGPCS)** register can be used to connect the USB0VBUS and USB0ID inputs to fixed levels internally, freeing the PB0 and PB1 pins for GPIO use. For proper self-powered Device operation, the VBUS value must still be monitored to assure that if the Host removes VBUS, the self-powered Device disables the D+/D- pull-up resistors. This function can be accomplished by connecting a standard GPIO to VBUS.

USB General-Purpose Control and Status (USBGPCS)

Base 0x4005.0000
Offset 0x41C
Type R/W, reset 0x0000.0001



Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	DEVMODOTG	R/W	0	Enable Device Mode This bit enables the DEVMOD bit to control the state of the internal ID signal in OTG mode. Value Description 0 The mode is specified by the state of the internal ID signal. 1 This bit enables the DEVMOD bit to control the internal ID signal.
0	DEVMOD	R/W	1	Device Mode This bit specifies the state of the internal ID signal in Host mode and in OTG mode when the DEVMODOTG bit is set. In Device mode this bit is ignored (assumed set). Value Description 0 Host mode 1 Device mode

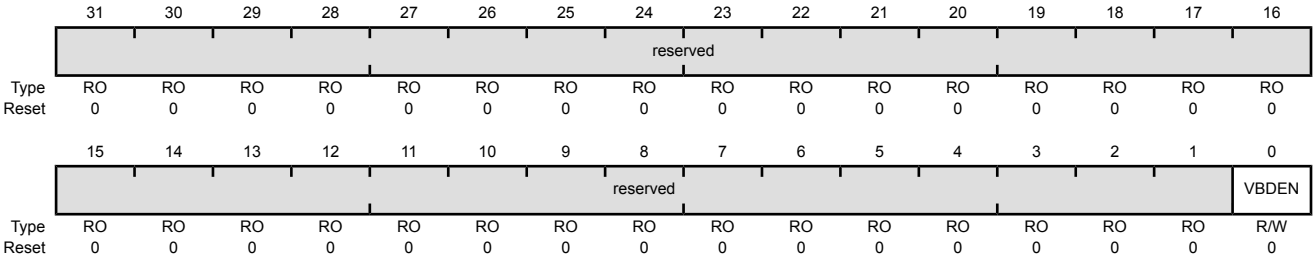
Register 325: USB VBUS Droop Control (USBVDC), offset 0x430

OTG A / Host

This 32-bit register enables a controlled masking of VBUS to compensate for any in-rush current by a Device that is connected to the Host controller. The in-rush current can cause VBUS to droop, causing the USB controller's behavior to be unexpected. The USB Host controller allows VBUS to fall lower than the VBUS Valid level (4.75 V) but not below AValid (2.0 V) for 65 microseconds without signaling a VBUSERR interrupt in the controller. Without this, any glitch on VBUS would force the USB Host controller to remove power from VBUS and then re-enumerate the Device.

USB VBUS Droop Control (USBVDC)

Base 0x4005.0000
Offset 0x430
Type R/W, reset 0x0000.0000



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

Value	Description
0	No effect.
1	Any changes from VBUSVALID are masked when VBUS goes below 4.75 V but not lower than 2.0 V for 65 microseconds. During this time, the VBUS state indicates VBUSVALID.

Register 326: USB VBUS Droop Control Raw Interrupt Status (USBVDCRIS), offset 0x434

OTG A / Host

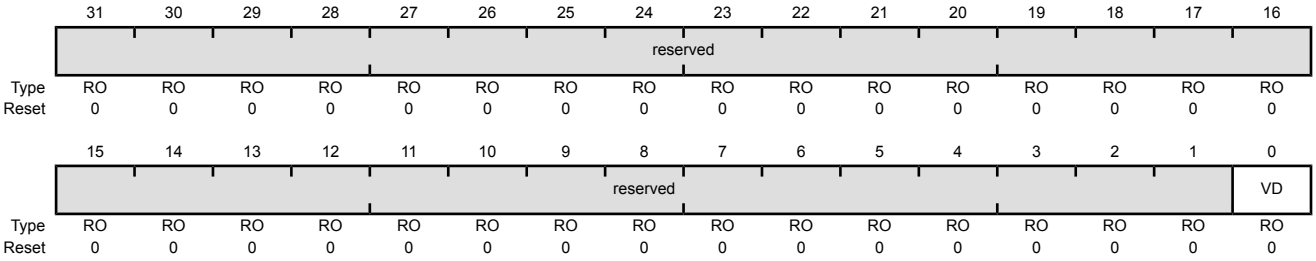
This 32-bit register specifies the unmasked interrupt status of the VBUS droop limit of 65 microseconds.

USB VBUS Droop Control Raw Interrupt Status (USBVDCRIS)

Base 0x4005.0000

Offset 0x434

Type RO, reset 0x0000.0000



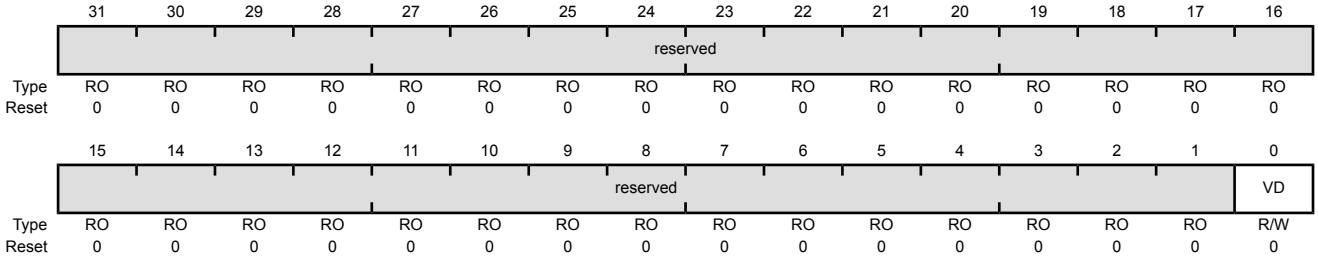
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	VD	RO	0	VBUS Droop Raw Interrupt Status Value Description 1 A VBUS droop lasting for 65 microseconds has been detected. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the VD bit in the **USBVDCISC** register.

Register 327: USB VBUS Droop Control Interrupt Mask (USBVDCIM), offset 0x438

OTG A / Host

This 32-bit register specifies the interrupt mask of the VBUS droop.

USB VBUS Droop Control Interrupt Mask (USBVDCIM)
Base 0x4005.0000
Offset 0x438
Type R/W, reset 0x0000.0000



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

Value	Description
1	The raw interrupt signal from a detected VBUS droop is sent to the interrupt controller.
0	A detected VBUS droop does not affect the interrupt status.

Register 328: USB VBUS Droop Control Interrupt Status and Clear (USBVDCISC), offset 0x43C

OTG A / Host

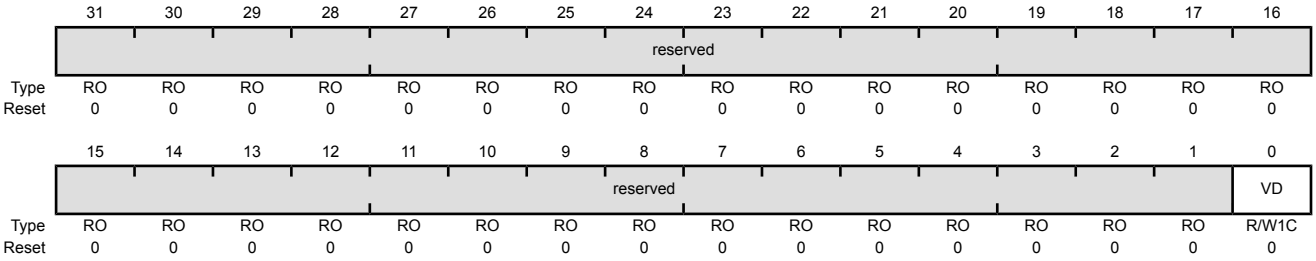
This 32-bit register specifies the masked interrupt status of the VBUS droop and provides a method to clear the interrupt state.

USB VBUS Droop Control Interrupt Status and Clear (USBVDCISC)

Base 0x4005.0000

Offset 0x43C

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	VD	R/W1C	0	VBUS Droop Interrupt Status and Clear Value Description 1 The VD bits in the **USBVDCRIS** and **USBVDCIM** registers are set, providing an interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the VD bit in the **USBVDCRIS** register.

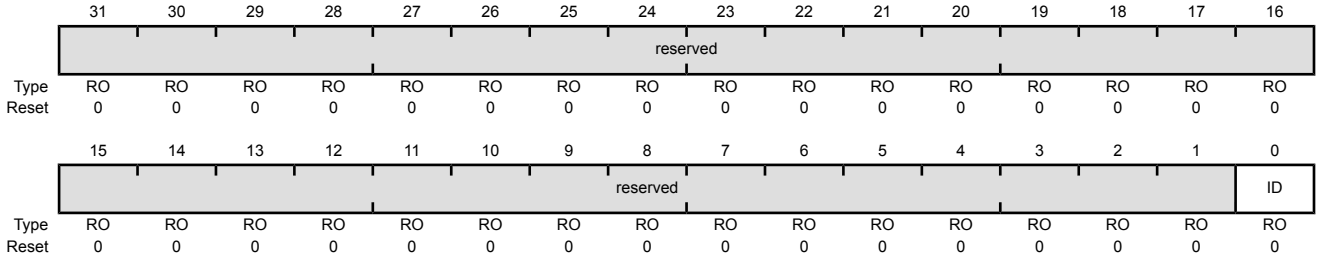
Register 329: USB ID Valid Detect Raw Interrupt Status (USBIDVRIS), offset 0x444

OTG

This 32-bit register specifies whether the unmasked interrupt status of the ID value is valid.

USB ID Valid Detect Raw Interrupt Status (USBIDVRIS)

Base 0x4005.0000
Offset 0x444
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	ID	RO	0	ID Valid Detect Raw Interrupt Status Value Description 1 A valid ID has been detected. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the `ID` bit in the **USBIDVISC** register.

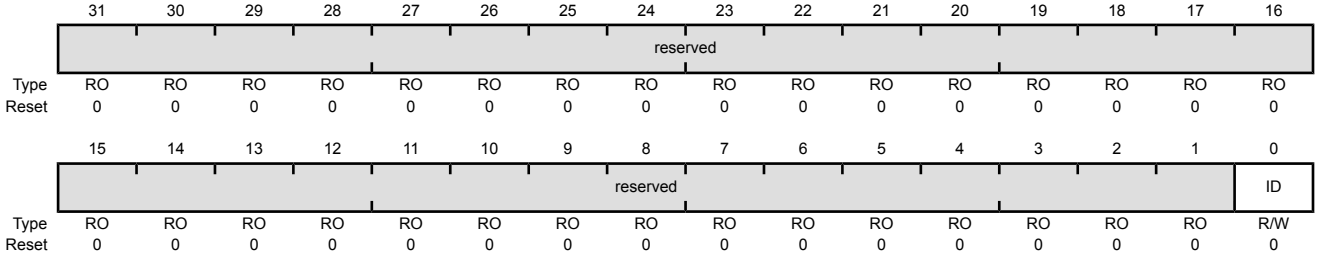
Register 330: USB ID Valid Detect Interrupt Mask (USBIDVIM), offset 0x448

OTG

This 32-bit register specifies the interrupt mask of the ID valid detection.

USB ID Valid Detect Interrupt Mask (USBIDVIM)

Base 0x4005.0000
Offset 0x448
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	ID	R/W	0	ID Valid Detect Interrupt Mask Value Description 1 The raw interrupt signal from a detected ID valid is sent to the interrupt controller. 0 A detected ID valid does not affect the interrupt status.

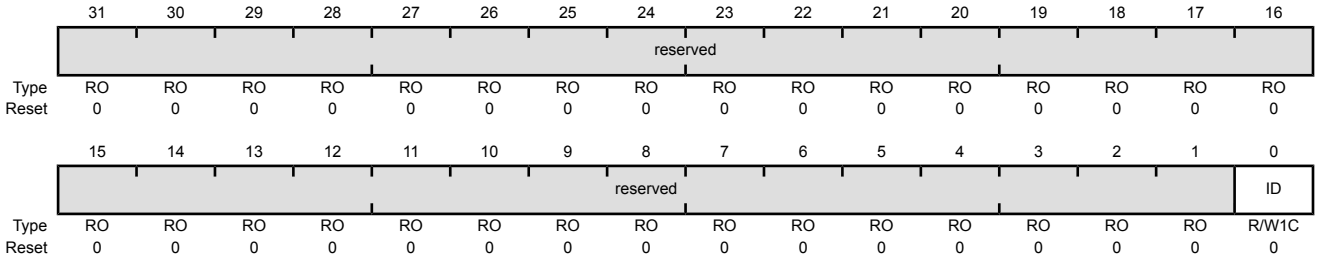
Register 331: USB ID Valid Detect Interrupt Status and Clear (USBIDVISC), offset 0x44C

OTG

This 32-bit register specifies the masked interrupt status of the ID valid detect. It also provides a method to clear the interrupt state.

USB ID Valid Detect Interrupt Status and Clear (USBIDVISC)

Base 0x4005.0000
Offset 0x44C
Type R/W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	ID	R/W1C	0	ID Valid Detect Interrupt Status and Clear Value Description 1 The `ID` bits in the **USBIDVRIS** and **USBIDVIM** registers are set, providing an interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the `ID` bit in the **USBIDVRIS** register.

Register 332: USB DMA Select (USBDMASEL), offset 0x450

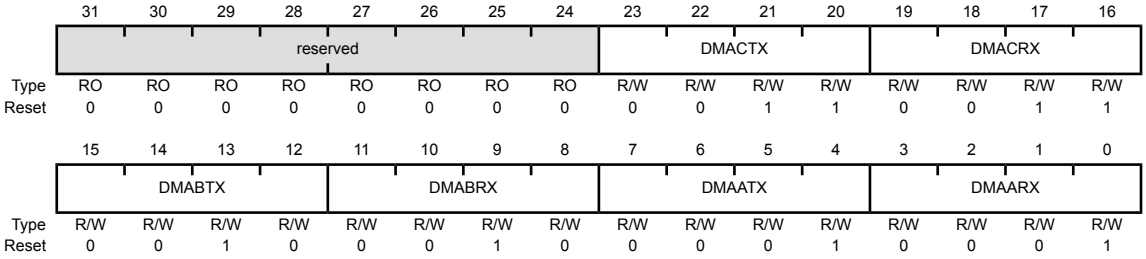
OTG A / Host

This 32-bit register specifies which endpoints are mapped to the 6 allocated µDMA channels, see Table 7-1 on page 336 for more information on channel assignments.

OTG B / Device

USB DMA Select (USBDMASEL)

Base 0x4005.0000
Offset 0x450
Type R/W, reset 0x0033.2211



Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:20	DMACTX	R/W	0x3	DMA C TX Select Specifies the TX mapping of the third USB endpoint on µDMA channel 5 (primary assignment). Value Description 0x0 reserved 0x1 Endpoint 1 TX 0x2 Endpoint 2 TX 0x3 Endpoint 3 TX 0x4 Endpoint 4 TX 0x5 Endpoint 5 TX 0x6 Endpoint 6 TX 0x7 Endpoint 7 TX 0x8 Endpoint 8 TX 0x9 Endpoint 9 TX 0xA Endpoint 10 TX 0xB Endpoint 11 TX 0xC Endpoint 12 TX 0xD Endpoint 13 TX 0xE Endpoint 14 TX 0xF Endpoint 15 TX

Bit/Field	Name	Type	Reset	Description
19:16	DMACRX	R/W	0x3	DMA C RX Select Specifies the RX and TX mapping of the third USB endpoint on µDMA channel 4 (primary assignment). Value Description 0x0 reserved 0x1 Endpoint 1 RX 0x2 Endpoint 2 RX 0x3 Endpoint 3 RX 0x4 Endpoint 4 RX 0x5 Endpoint 5 RX 0x6 Endpoint 6 RX 0x7 Endpoint 7 RX 0x8 Endpoint 8 RX 0x9 Endpoint 9 RX 0xA Endpoint 10 RX 0xB Endpoint 11 RX 0xC Endpoint 12 RX 0xD Endpoint 13 RX 0xE Endpoint 14 RX 0xF Endpoint 15 RX
15:12	DMABTX	R/W	0x2	DMA B TX Select Specifies the TX mapping of the second USB endpoint on µDMA channel 3 (primary assignment). Same bit definitions as the DMACTX field.
11:8	DMABRX	R/W	0x2	DMA B RX Select Specifies the RX mapping of the second USB endpoint on µDMA channel 2 (primary assignment). Same bit definitions as the DMACRX field.
7:4	DMAATX	R/W	0x1	DMA A TX Select Specifies the TX mapping of the first USB endpoint on µDMA channel 1 (primary assignment). Same bit definitions as the DMACTX field.
3:0	DMAARX	R/W	0x1	DMA A RX Select Specifies the RX mapping of the first USB endpoint on µDMA channel 0 (primary assignment). Same bit definitions as the DMACRX field.

20 Analog Comparators

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

Note: Not all comparators have the option to drive an output pin. See “Signal Description” on page 1109 for more information.

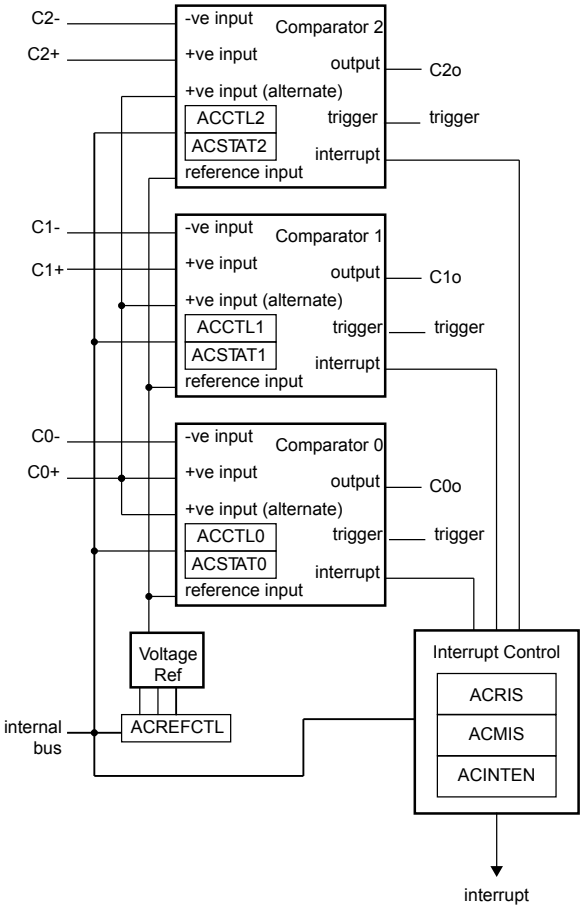
The comparator can provide its output to a device pin, acting as a replacement for an analog comparator on the board. In addition, the comparator can signal the application via interrupts or trigger the start of a sample sequence in the ADC. The interrupt generation and ADC triggering logic is separate and independent. This flexibility means, for example, that an interrupt can be generated on a rising edge and the ADC triggered on a falling edge.

The Stellaris® LM3S9U81 microcontroller provides three independent integrated analog comparators with the following functions:

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

20.1 Block Diagram

Figure 20-1. Analog Comparator Module Block Diagram



20.2 Signal Description

The following table lists the external signals of the Analog Comparators and describes the function of each. The Analog Comparator output signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the Analog Comparator signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 419) should be set to choose the Analog Comparator function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 437) to assign the Analog Comparator signal to the specified GPIO port pin. The positive and negative input signals are configured by clearing the DEN bit in the **GPIO Digital Enable (GPIODEN)** register. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 395.

Table 20-1. Analog Comparators Signals (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
C0+	90	PB6	I	Analog	Analog comparator 0 positive input.
C0-	92	PB4	I	Analog	Analog comparator 0 negative input.

Table 20-1. Analog Comparators Signals (100LQFP) *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
C0o	24 42 90 91 100	PC5 (3) PF4 (2) PB6 (3) PB5 (1) PD7 (2)	O	TTL	Analog comparator 0 output.
C1+	24	PC5	I	Analog	Analog comparator 1 positive input.
C1-	91	PB5	I	Analog	Analog comparator 1 negative input.
C1o	2 22 24 41 84	PE6 (2) PC7 (7) PC5 (2) PF5 (2) PH2 (2)	O	TTL	Analog comparator 1 output.
C2+	23	PC6	I	Analog	Analog comparator 2 positive input.
C2-	22	PC7	I	Analog	Analog comparator 2 negative input.
C2o	1 23	PE7 (2) PC6 (3)	O	TTL	Analog comparator 2 output.

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

Table 20-2. Analog Comparators Signals (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
C0+	A7	PB6	I	Analog	Analog comparator 0 positive input.
C0-	A6	PB4	I	Analog	Analog comparator 0 negative input.
C0o	M1 K4 A7 B7 A2	PC5 (3) PF4 (2) PB6 (3) PB5 (1) PD7 (2)	O	TTL	Analog comparator 0 output.
C1+	M1	PC5	I	Analog	Analog comparator 1 positive input.
C1-	B7	PB5	I	Analog	Analog comparator 1 negative input.
C1o	A1 L2 M1 K3 D11	PE6 (2) PC7 (7) PC5 (2) PF5 (2) PH2 (2)	O	TTL	Analog comparator 1 output.
C2+	M2	PC6	I	Analog	Analog comparator 2 positive input.
C2-	L2	PC7	I	Analog	Analog comparator 2 negative input.
C2o	B1 M2	PE7 (2) PC6 (3)	O	TTL	Analog comparator 2 output.

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

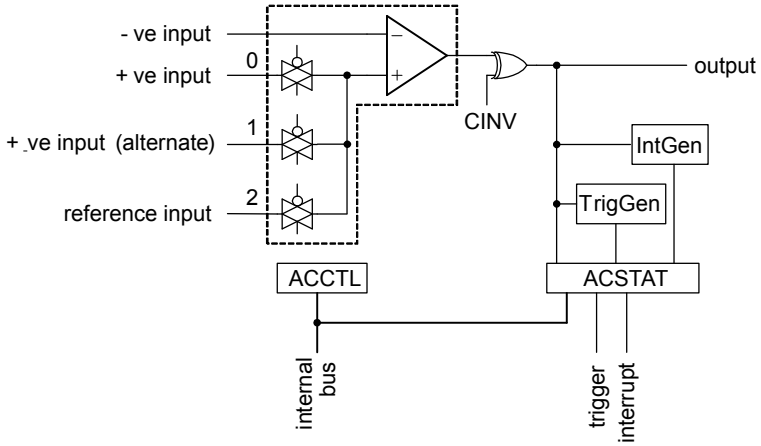
20.3 Functional Description

The comparator compares the VIN- and VIN+ inputs to produce an output, VOUT.

```
VIN- < VIN+, VOUT = 1
VIN- > VIN+, VOUT = 0
```

As shown in Figure 20-2 on page 1111, the input source for VIN- is an external input, Cn-. In addition to an external input, Cn+, input sources for VIN+ can be the C0+ or an internal reference, V_{IREF}.

Figure 20-2. Structure of Comparator Unit



A comparator is configured through two status/control registers, **Analog Comparator Control (ACCTL)** and **Analog Comparator Status (ACSTAT)**. The internal reference is configured through one control register, **Analog Comparator Reference Voltage Control (ACREFCTL)**. Interrupt status and control are configured through three registers, **Analog Comparator Masked Interrupt Status (ACMIS)**, **Analog Comparator Raw Interrupt Status (ACRIS)**, and **Analog Comparator Interrupt Enable (ACINTEN)**.

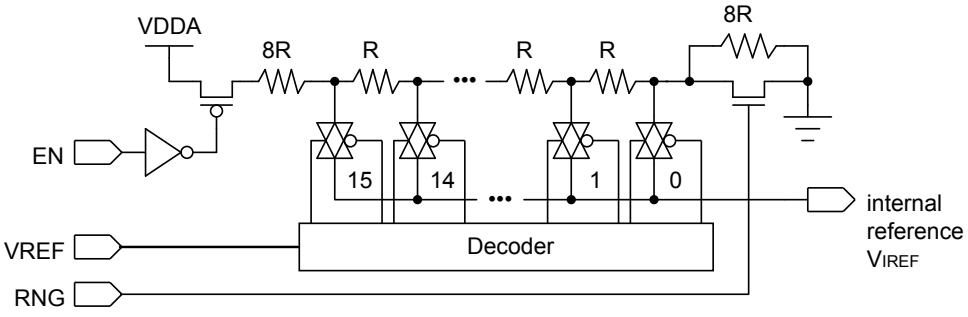
Typically, the comparator output is used internally to generate an interrupt as controlled by the ISEN bit in the **ACCTL** register. The output may also be used to drive an external pin, Co or generate an analog-to-digital converter (ADC) trigger.

Important: The ASRCP bits in the **ACCTL** register must be set before using the analog comparators.

20.3.1 Internal Reference Programming

The structure of the internal reference is shown in Figure 20-3 on page 1111. The internal reference is controlled by a single configuration register (**ACREFCTL**).

Figure 20-3. Comparator Internal Reference Structure



The internal reference can be programmed in one of two modes (low range or high range) depending on the RNG bit in the **ACREFCTL** register. When RNG is clear, the internal reference is in high-range mode, and when RNG is set the internal reference is in low-range mode.

In each range, the internal reference, V_{IREF}, has 16 pre-programmed thresholds or step values. The threshold to be used to compare the external input voltage against is selected using the VREF field in the **ACREFCTL** register.

In the high-range mode, the V_{IREF} threshold voltages start at the ideal high-range starting voltage of $V_{DDA}/3.875$ and increase in ideal constant voltage steps of $V_{DDA}/31$.

In the low-range mode, the V_{IREF} threshold voltages start at:0V and increase in ideal constant voltage steps of $V_{DDA}/23$. The ideal V_{IREF} step voltages for each mode and their dependence on the RNG and VREF fields are summarized in Table 20-3 on page 1112.

Table 20-3. 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. It is recommended that RNG=1 and VREF=0 to minimize noise on the reference ground.
EN=1	RNG=0	Total resistance in ladder is 31 R. $V_{IREF} = V_{DDA} \times \frac{R_{VREF}}{R_T}$ $V_{IREF} = V_{DDA} \times \frac{(VREF + 8)}{31}$ $V_{IREF} = 0.85 + 0.106 \times VREF$ The range of internal reference in this mode is 0.85-2.448 V.
	RNG=1	Total resistance in ladder is 23 R. $V_{IREF} = V_{DDA} \times \frac{R_{VREF}}{R_T}$ $V_{IREF} = V_{DDA} \times \frac{VREF}{23}$ $V_{IREF} = 0.143 \times VREF$ The range of internal reference for this mode is 0-2.152 V.

20.4 Initialization and Configuration

The following example shows how to configure an analog comparator to read back its output value from an internal register.

1. Enable the analog comparator clock by writing a value of 0x0010.0000 to the **RCGC1** register in the System Control module (see page 263).
2. Enable the clock to the appropriate GPIO modules via the **RCGC2** register (see page 272). To find out which GPIO ports to enable, refer to Table 22-5 on page 1151.
3. In the GPIO module, enable the GPIO port/pin associated with the input signals as GPIO inputs. To determine which GPIO to configure, see Table 22-4 on page 1144.
4. Configure the PMCn fields in the **GPIOPCTL** register to assign the analog comparator output signals to the appropriate pins (see page 437 and Table 22-5 on page 1151).
5. Configure the internal voltage reference to 1.65 V by writing the **ACREFCTL** register with the value 0x0000.030C.
6. Configure the comparator to use the internal voltage reference and to *not* invert the output by writing the **ACCTLn** register with the value of 0x0000.040C.
7. Delay for 10 µs.
8. Read the comparator output value by reading the **ACSTATn** register's OVAL value.

Change the level of the comparator negative input signal C- to see the OVAL value change.

20.5 Register Map

Table 20-4 on page 1113 lists the comparator registers. The offset listed is a hexadecimal increment to the register's address, relative to the Analog Comparator base address of 0x4003.C000. Note that the analog comparator clock must be enabled before the registers can be programmed (see page 263). There must be a delay of 3 system clocks after the analog comparator module clock is enabled before any analog comparator module registers are accessed.

Table 20-4. Analog Comparators Register Map

Offset	Name	Type	Reset	Description	See page
0x000	ACMIS	R/W1C	0x0000.0000	Analog Comparator Masked Interrupt Status	1115
0x004	ACRIS	RO	0x0000.0000	Analog Comparator Raw Interrupt Status	1116
0x008	ACINTEN	R/W	0x0000.0000	Analog Comparator Interrupt Enable	1117
0x010	ACREFCTL	R/W	0x0000.0000	Analog Comparator Reference Voltage Control	1118
0x020	ACSTAT0	RO	0x0000.0000	Analog Comparator Status 0	1119
0x024	ACCTL0	R/W	0x0000.0000	Analog Comparator Control 0	1120
0x040	ACSTAT1	RO	0x0000.0000	Analog Comparator Status 1	1119
0x044	ACCTL1	R/W	0x0000.0000	Analog Comparator Control 1	1120
0x060	ACSTAT2	RO	0x0000.0000	Analog Comparator Status 2	1119
0x064	ACCTL2	R/W	0x0000.0000	Analog Comparator Control 2	1120

20.6 Register Descriptions

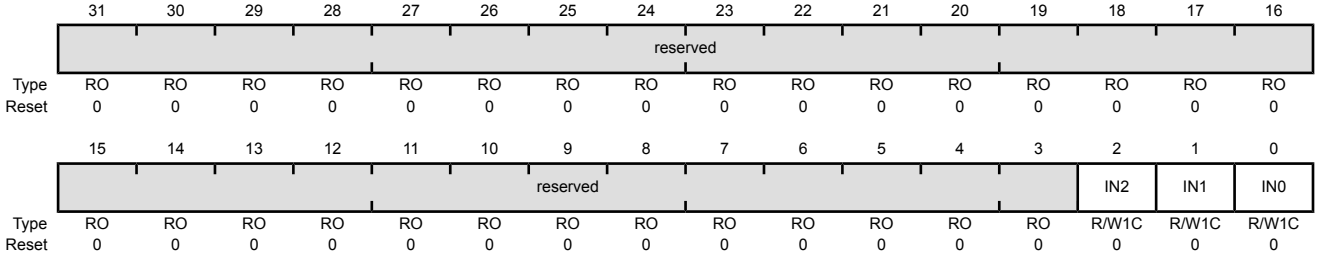
The remainder of this section lists and describes the Analog Comparator registers, in numerical order by address offset.

Register 1: Analog Comparator Masked Interrupt Status (ACMIS), offset 0x000

This register provides a summary of the interrupt status (masked) of the comparators.

Analog Comparator Masked Interrupt Status (ACMIS)

Base 0x4003.C000
Offset 0x000
Type R/W1C, reset 0x0000.0000



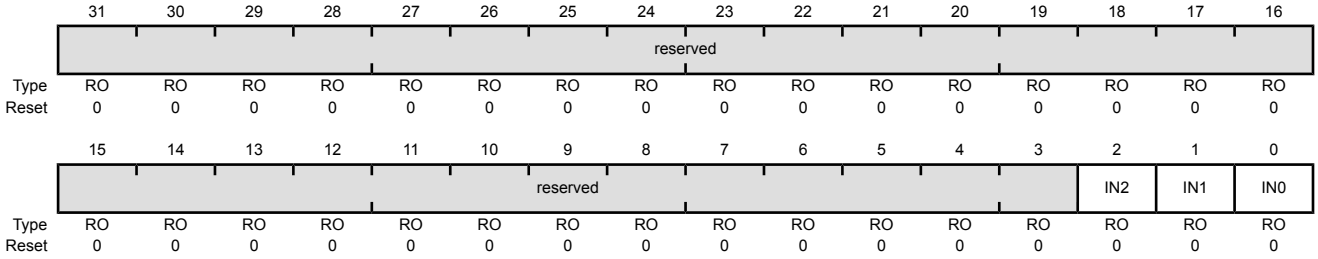
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	IN2	R/W1C	0	Comparator 2 Masked Interrupt Status Value Description 1 The `IN2` bits in the **ACRIS** register and the **ACINTEN** registers are set, providing an interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the `IN2` bit in the **ACRIS** register.
1	IN1	R/W1C	0	Comparator 1 Masked Interrupt Status Value Description 1 The `IN1` bits in the **ACRIS** register and the **ACINTEN** registers are set, providing an interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the `IN1` bit in the **ACRIS** register.
0	IN0	R/W1C	0	Comparator 0 Masked Interrupt Status Value Description 1 The `IN0` bits in the **ACRIS** register and the **ACINTEN** registers are set, providing an interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the `IN0` bit in the **ACRIS** register.

Register 2: Analog Comparator Raw Interrupt Status (ACRIS), offset 0x004

This register provides a summary of the interrupt status (raw) of the comparators. The bits in this register must be enabled to generate interrupts using the **ACINTEN** register.

Analog Comparator Raw Interrupt Status (ACRIS)

Base 0x4003.C000
Offset 0x004
Type RO, reset 0x0000.0000



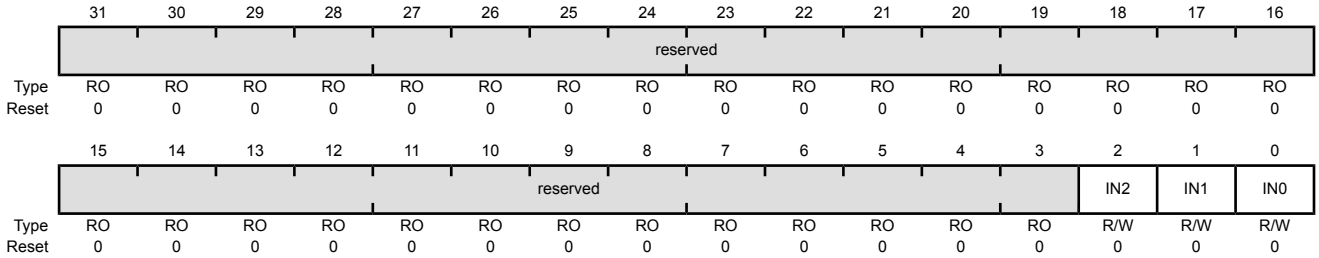
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	IN2	RO	0	Comparator 2 Interrupt Status Value Description 1 Comparator 2 has generated an interrupt for an event as configured by the ISEN bit in the **ACCTL2** register. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the IN2 bit in the **ACMIS** register.
1	IN1	RO	0	Comparator 1 Interrupt Status Value Description 1 Comparator 1 has generated an interruptfor an event as configured by the ISEN bit in the **ACCTL1** register. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the IN1 bit in the **ACMIS** register.
0	IN0	RO	0	Comparator 0 Interrupt Status Value Description 1 Comparator 0 has generated an interrupt for an event as configured by the ISEN bit in the **ACCTL0** register. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the IN0 bit in the **ACMIS** register.

Register 3: Analog Comparator Interrupt Enable (ACINTEN), offset 0x008

This register provides the interrupt enable for the comparators.

Analog Comparator Interrupt Enable (ACINTEN)

Base 0x4003.C000
Offset 0x008
Type R/W, reset 0x0000.0000



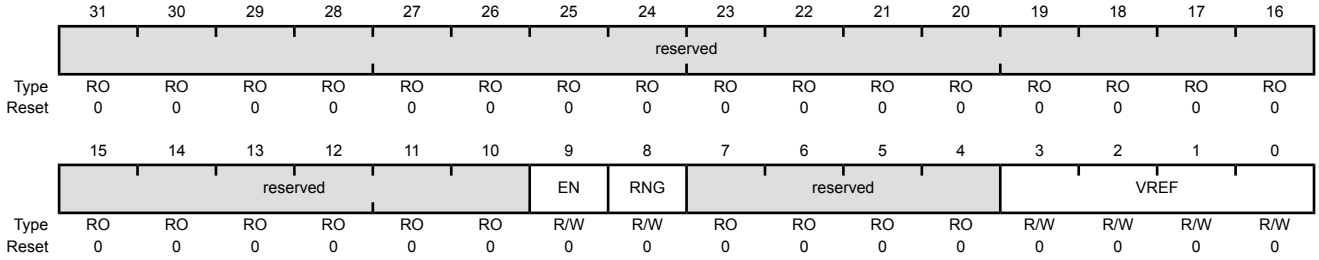
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	IN2	R/W	0	Comparator 2 Interrupt Enable Value Description 1 The raw interrupt signal comparator 2 is sent to the interrupt controller. 0 A comparator 2 interrupt does not affect the interrupt status.
1	IN1	R/W	0	Comparator 1 Interrupt Enable Value Description 1 The raw interrupt signal comparator 1 is sent to the interrupt controller. 0 A comparator 1 interrupt does not affect the interrupt status.
0	IN0	R/W	0	Comparator 0 Interrupt Enable Value Description 1 The raw interrupt signal comparator 0 is sent to the interrupt controller. 0 A comparator 0 interrupt does not affect the interrupt status.

Register 4: Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x010

This register specifies whether the resistor ladder is powered on as well as the range and tap.

Analog Comparator Reference Voltage Control (ACREFCTL)

Base 0x4003.C000
Offset 0x010
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	EN	R/W	0	Resistor Ladder Enable Value Description 0 The resistor ladder is unpowered. 1 Powers on the resistor ladder. The resistor ladder is connected to V_{DDA}. This bit is cleared at reset so that the internal reference consumes the least amount of power if it is not used.
8	RNG	R/W	0	Resistor Ladder Range Value Description 0 The resistor ladder has a total resistance of 31 R. 1 The resistor ladder has a total resistance of 23 R.
7:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	VREF	R/W	0x0	Resistor Ladder Voltage Ref The VREF bit field specifies the resistor ladder tap that is passed through an analog multiplexer. The voltage corresponding to the tap position is the internal reference voltage available for comparison. See Table 20-3 on page 1112 for some output reference voltage examples.

Register 5: Analog Comparator Status 0 (ACSTAT0), offset 0x020

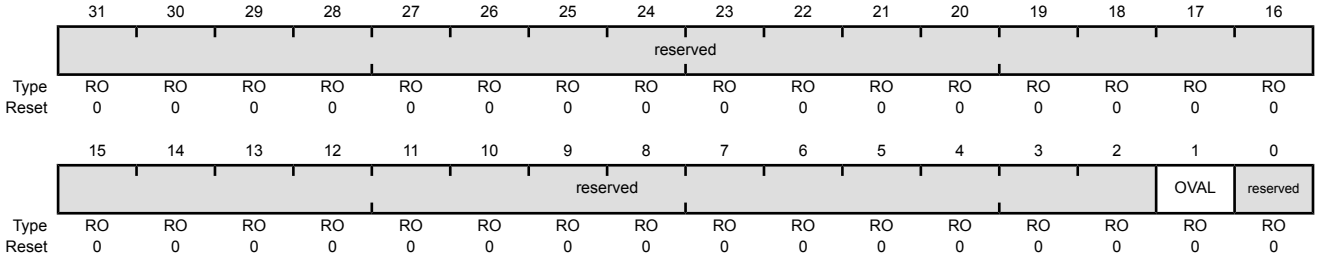
Register 6: Analog Comparator Status 1 (ACSTAT1), offset 0x040

Register 7: Analog Comparator Status 2 (ACSTAT2), offset 0x060

These registers specify the current output value of the comparator.

Analog Comparator Status 0 (ACSTAT0)

Base 0x4003.C000
Offset 0x020
Type RO, reset 0x0000.0000



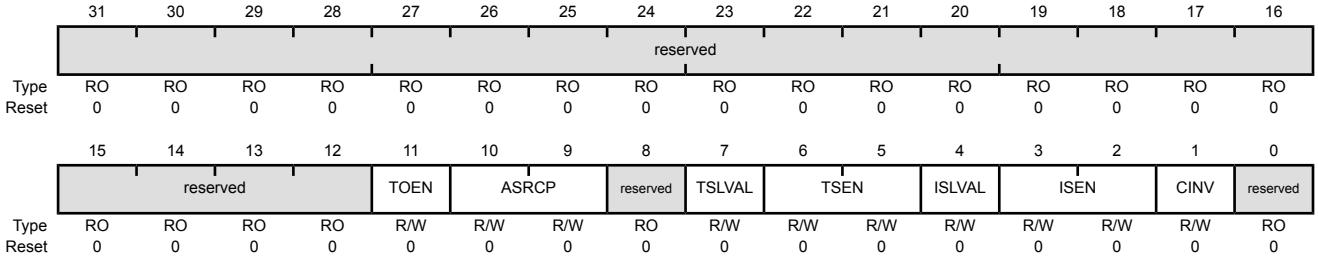
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	OVAL	RO	0	Comparator Output Value Value Description 0 VIN- > VIN+ 1 VIN- < VIN+ VIN - is the voltage on the Cn - pin. VIN+ is the voltage on the Cn+ pin, the C0+ pin, or the internal voltage reference (V_{IREF}) as defined by the ASRCP bit in the **ACCTL** register.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 8: Analog Comparator Control 0 (ACCTL0), offset 0x024
Register 9: Analog Comparator Control 1 (ACCTL1), offset 0x044
Register 10: Analog Comparator Control 2 (ACCTL2), offset 0x064

These registers configure the comparator's input and output.

Analog Comparator Control 0 (ACCTL0)

Base 0x4003.C000
Offset 0x024
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	TOEN	R/W	0	Trigger Output Enable Value Description 0 ADC events are suppressed and not sent to the ADC. 1 ADC events are sent to the ADC.
10:9	ASRCP	R/W	0x0	Analog Source Positive The ASRCP field specifies the source of input voltage to the VIN+ terminal of the comparator. The encodings for this field are as follows: Value Description 0x0 Pin value of Cn+ 0x1 Pin value of C0+ 0x2 Internal voltage reference (V_{IREF}) 0x3 Reserved
8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TSLVAL	R/W	0	Trigger Sense Level Value Value Description 0 An ADC event is generated if the comparator output is Low. 1 An ADC event is generated if the comparator output is High.

Bit/Field	Name	Type	Reset	Description
6:5	TSEN	R/W	0x0	Trigger Sense The TSEN field specifies the sense of the comparator output that generates an ADC event. The sense conditioning is as follows: Value Description 0x0 Level sense, see TSLVAL 0x1 Falling edge 0x2 Rising edge 0x3 Either edge
4	ISLVAL	R/W	0	Interrupt Sense Level Value Value Description 0 An interrupt is generated if the comparator output is Low. 1 An interrupt is generated if the comparator output is High.
3:2	ISEN	R/W	0x0	Interrupt Sense The ISEN field specifies the sense of the comparator output that generates an interrupt. The sense conditioning is as follows: Value Description 0x0 Level sense, see ISLVAL 0x1 Falling edge 0x2 Rising edge 0x3 Either edge
1	CINV	R/W	0	Comparator Output Invert Value Description 0 The output of the comparator is unchanged. 1 The output of the comparator is inverted prior to being processed by hardware.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

21 Pin Diagram

The LM3S9U81 microcontroller pin diagram is shown below.

Each GPIO signal is identified by its GPIO port unless it defaults to an alternate function on reset. In this case, the GPIO port name is followed by the default alternate function. To see a complete list of possible functions for each pin, see Table 22-5 on page 1151.

Figure 21-1. 100-Pin LQFP Package Pin Diagram

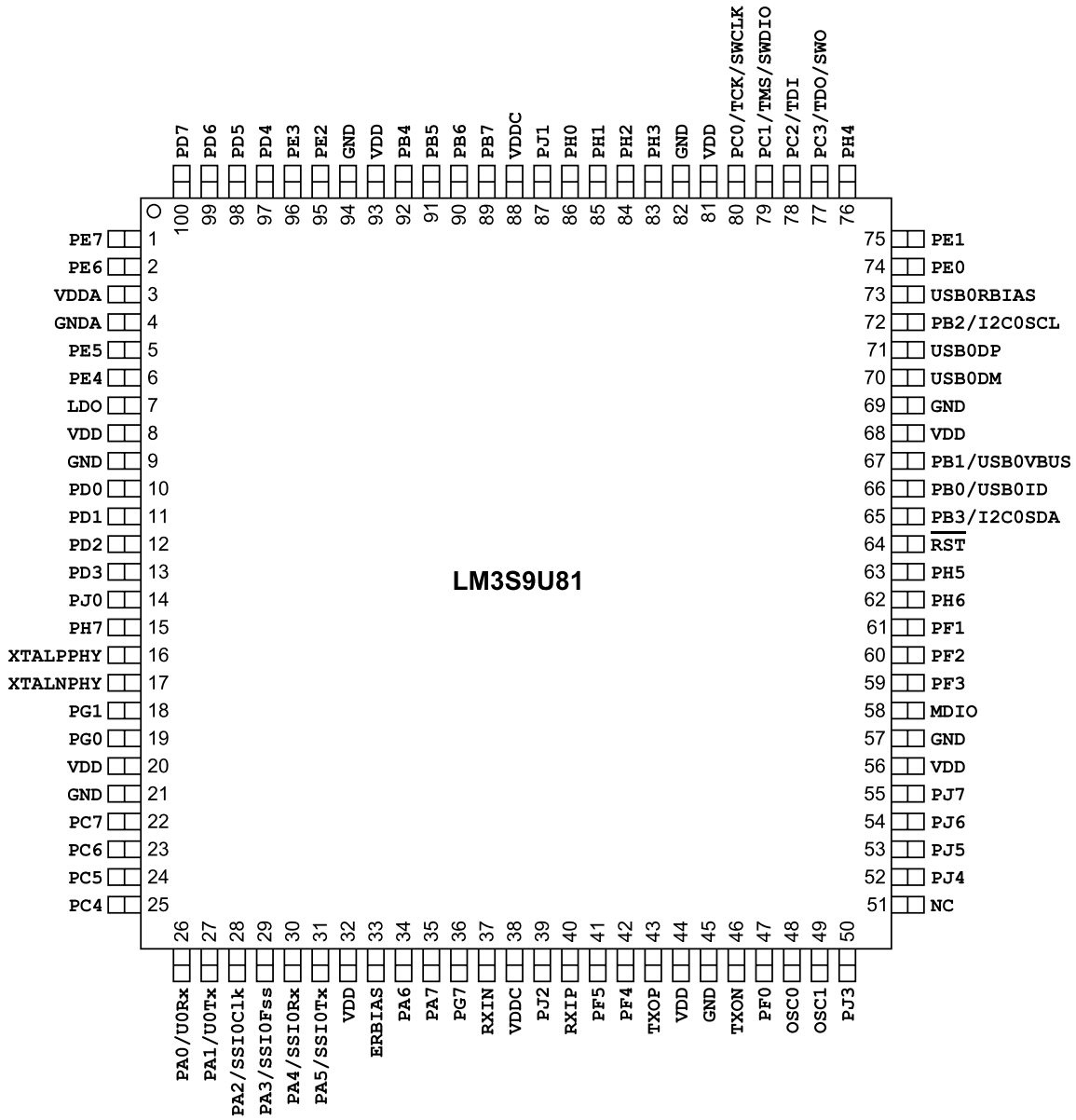
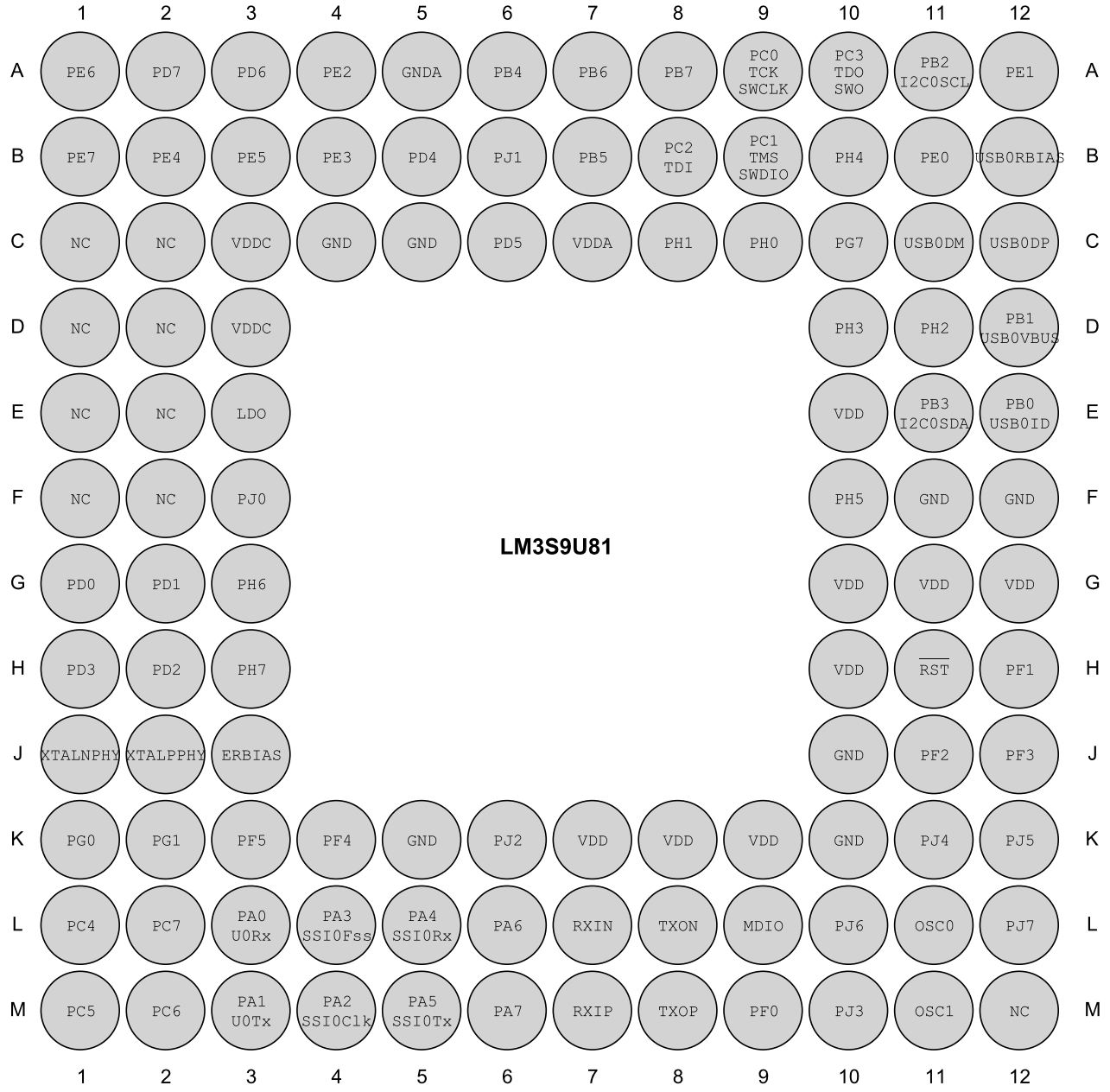


Figure 21-2. 108-Ball BGA Package Pin Diagram (Top View)



22 Signal Tables

The following tables list the signals available for each pin. Signals are configured as GPIOs on reset, except for those noted below. Use the **GPIOAMSEL** register (see page 435) to select analog mode. For a GPIO pin to be used for an alternate digital function, the corresponding bit in the **GPIOAFSEL** register (see page 419) must be set. Further pin muxing options are provided through the PMCx bit field in the **GPIOPCTL** register (see page 437), which selects one of several available peripheral functions for that GPIO.

Important: All GPIO pins are configured as GPIOs by default with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ puts the pins back to their default state.

Table 22-1. GPIO Pins With Default Alternate Functions

GPIO Pin	Default State	GPIOAFSEL Bit	GPIOPCTL PMCx Bit Field
PA[1:0]	UART0	0	0x1
PA[5:2]	SSI0	0	0x1
PB[3:2]	I^2C0	0	0x1
PC[3:0]	JTAG/SWD	1	0x3

Table 22-2 on page 1125 shows the pin-to-signal-name mapping, including functional characteristics of the signals. Each possible alternate analog and digital function is listed for each pin.

Table 22-3 on page 1135 lists the signals in alphabetical order by signal name. If it is possible for a signal to be on multiple pins, each possible pin assignment is listed. The "Pin Mux" column indicates the GPIO and the encoding needed in the PMCx bit field in the **GPIOPCTL** register.

Table 22-4 on page 1144 groups the signals by functionality, except for GPIOs. If it is possible for a signal to be on multiple pins, each possible pin assignment is listed.

Table 22-5 on page 1151 lists the GPIO pins and their analog and digital alternate functions. The AINx and VREFA analog signals are not 5-V tolerant and go through an isolation circuit before reaching their circuitry. These signals are configured by clearing the corresponding DEN bit in the **GPIO Digital Enable (GPIODEN)** register and setting the corresponding AMSEL bit in the **GPIO Analog Mode Select (GPIOAMSEL)** register. Other analog signals are 5-V tolerant and are connected directly to their circuitry (C0-, C0+, C1-, C1+, C2-, C2+, USB0VBUS, USB0ID). These signals are configured by clearing the DEN bit in the **GPIO Digital Enable (GPIODEN)** register. The digital signals are enabled by setting the appropriate bit in the **GPIO Alternate Function Select (GPIOAFSEL)** and **GPIODEN** registers and configuring the PMCx bit field in the **GPIO Port Control (GPIOPCTL)** register to the numeric enoding shown in the table below. Table entries that are shaded gray are the default values for the corresponding GPIO pin.

Table 22-6 on page 1154 lists the signals based on number of possible pin assignments. This table can be used to plan how to configure the pins for a particular functionality. Application Note AN01274 Configuring Stellaris® Microcontrollers with Pin Multiplexing provides an overview of the pin muxing implementation, an explanation of how a system designer defines a pin configuration, and examples of the pin configuration process.

Note: All digital inputs are Schmitt triggered.

22.1 100-Pin LQFP Package Pin Tables

Table 22-2. Signals by Pin Number

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
1	PE7	I/O	TTL	GPIO port E bit 7.
	AIN0	I	Analog	Analog-to-digital converter input 0.
	C2o	O	TTL	Analog comparator 2 output.
	U1DCD	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
2	PE6	I/O	TTL	GPIO port E bit 6.
	AIN1	I	Analog	Analog-to-digital converter input 1.
	C1o	O	TTL	Analog comparator 1 output.
	U1CTS	I	TTL	UART module 1 Clear To Send modem flow control input signal.
3	VDDA	-	Power	The positive supply for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be supplied with a voltage that meets the specification in Table 24-2 on page 1192, regardless of system implementation.
4	GNDA	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
5	PE5	I/O	TTL	GPIO port E bit 5.
	AIN2	I	Analog	Analog-to-digital converter input 2.
	CAN2Tx	O	TTL	CAN module 2 transmit.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	I2S0TXSD	I/O	TTL	I^2S module 0 transmit data.
6	PE4	I/O	TTL	GPIO port E bit 4.
	AIN3	I	Analog	Analog-to-digital converter input 3.
	CAN2Rx	I	TTL	CAN module 2 receive.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	I2S0TXWS	I/O	TTL	I^2S module 0 transmit word select.
	U2Tx	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
7	LDO	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater. The LDO pin must also be connected to the VDDC pins at the board level in addition to the decoupling capacitor(s).
8	VDD	-	Power	Positive supply for I/O and some logic.
9	GND	-	Power	Ground reference for logic and I/O pins.

Table 22-2. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
10	PD0	I/O	TTL	GPIO port D bit 0.
	AIN15	I	Analog	Analog-to-digital converter input 15.
	CAN0Rx	I	TTL	CAN module 0 receive.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	I2S0RXSCK	I/O	TTL	I^2S module 0 receive clock.
	U1CTS	I	TTL	UART module 1 Clear To Send modem flow control input signal.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	U2Rx	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
11	PD1	I/O	TTL	GPIO port D bit 1.
	AIN14	I	Analog	Analog-to-digital converter input 14.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	I2S0RXWS	I/O	TTL	I^2S module 0 receive word select.
	U1DCD	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	U2Tx	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
12	PD2	I/O	TTL	GPIO port D bit 2.
	AIN13	I	Analog	Analog-to-digital converter input 13.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	EPI0S20	I/O	TTL	EPI module 0 signal 20.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
13	PD3	I/O	TTL	GPIO port D bit 3.
	AIN12	I	Analog	Analog-to-digital converter input 12.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	EPI0S21	I/O	TTL	EPI module 0 signal 21.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
14	PJ0	I/O	TTL	GPIO port J bit 0.
	EPI0S16	I/O	TTL	EPI module 0 signal 16.
	I2C1SCL	I/O	OD	I^2C module 1 clock.
15	PH7	I/O	TTL	GPIO port H bit 7.
	EPI0S27	I/O	TTL	EPI module 0 signal 27.
	SSI1Tx	O	TTL	SSI module 1 transmit.
16	XTALPPHY	I	Analog	Ethernet PHY XTALP 25-MHz oscillator crystal input or external clock reference input.

Table 22-2. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
17	XTALNPHY	O	Analog	Ethernet PHY XTALN 25-MHz oscillator crystal output. Leave this pin unconnected when using a single-ended 25-MHz clock input connected to the XTALPPHY pin.
18	PG1	I/O	TTL	GPIO port G bit 1.
	EPI0S14	I/O	TTL	EPI module 0 signal 14.
	I2C1SDA	I/O	OD	I^2C module 1 data.
	U2Tx	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
19	PG0	I/O	TTL	GPIO port G bit 0.
	EPI0S13	I/O	TTL	EPI module 0 signal 13.
	I2C1SCL	I/O	OD	I^2C module 1 clock.
	U2Rx	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
	USB0EPEN	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
20	VDD	-	Power	Positive supply for I/O and some logic.
21	GND	-	Power	Ground reference for logic and I/O pins.
22	PC7	I/O	TTL	GPIO port C bit 7.
	C1o	O	TTL	Analog comparator 1 output.
	C2-	I	Analog	Analog comparator 2 negative input.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	EPI0S5	I/O	TTL	EPI module 0 signal 5.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
23	PC6	I/O	TTL	GPIO port C bit 6.
	C2+	I	Analog	Analog comparator 2 positive input.
	C2o	O	TTL	Analog comparator 2 output.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	EPI0S4	I/O	TTL	EPI module 0 signal 4.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.

Table 22-2. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
24	PC5	I/O	TTL	GPIO port C bit 5.
	C0o	O	TTL	Analog comparator 0 output.
	C1+	I	Analog	Analog comparator 1 positive input.
	C1o	O	TTL	Analog comparator 1 output.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	EPI0S3	I/O	TTL	EPI module 0 signal 3.
	USB0EPEN	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
25	PC4	I/O	TTL	GPIO port C bit 4.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	EPI0S2	I/O	TTL	EPI module 0 signal 2.
26	PA0	I/O	TTL	GPIO port A bit 0.
	I2C1SCL	I/O	OD	I^2C module 1 clock.
	U0Rx	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
27	PA1	I/O	TTL	GPIO port A bit 1.
	I2C1SDA	I/O	OD	I^2C module 1 data.
	U0Tx	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
28	PA2	I/O	TTL	GPIO port A bit 2.
	I2S0RXSD	I/O	TTL	I^2S module 0 receive data.
	SSI0Clk	I/O	TTL	SSI module 0 clock.
29	PA3	I/O	TTL	GPIO port A bit 3.
	I2S0RXMCLK	I/O	TTL	I^2S module 0 receive master clock.
	SSI0Fss	I/O	TTL	SSI module 0 frame.
30	PA4	I/O	TTL	GPIO port A bit 4.
	CAN0Rx	I	TTL	CAN module 0 receive.
	I2S0TXSCK	I/O	TTL	I^2S module 0 transmit clock.
	SSI0Rx	I	TTL	SSI module 0 receive.
31	PA5	I/O	TTL	GPIO port A bit 5.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	I2S0TXWS	I/O	TTL	I^2S module 0 transmit word select.
	SSI0Tx	O	TTL	SSI module 0 transmit.
32	VDD	-	Power	Positive supply for I/O and some logic.
33	ERBIAS	O	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.

Table 22-2. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
34	PA6	I/O	TTL	GPIO port A bit 6.
	CAN0Rx	I	TTL	CAN module 0 receive.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	I2C1SCL	I/O	OD	I^2C module 1 clock.
	U1CTS	I	TTL	UART module 1 Clear To Send modem flow control input signal.
	USB0EPEN	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
35	PA7	I/O	TTL	GPIO port A bit 7.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	I2C1SDA	I/O	OD	I^2C module 1 data.
	U1DCD	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
36	PG7	I/O	TTL	GPIO port G bit 7.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	EPI0S31	I/O	TTL	EPI module 0 signal 31.
37	RXIN	I	Analog	RXIN of the Ethernet PHY.
38	VDDC	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals. The voltage on this pin is 1.3 V and is supplied by the on-chip LDO. The VDDC pins should only be connected to the LDO pin and an external capacitor as specified in Table 24-6 on page 1197.
39	PJ2	I/O	TTL	GPIO port J bit 2.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	EPI0S18	I/O	TTL	EPI module 0 signal 18.
40	RXIP	I	Analog	RXIP of the Ethernet PHY.
41	PF5	I/O	TTL	GPIO port F bit 5.
	C1o	O	TTL	Analog comparator 1 output.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	EPI0S15	I/O	TTL	EPI module 0 signal 15.
	SSI1Tx	O	TTL	SSI module 1 transmit.
42	PF4	I/O	TTL	GPIO port F bit 4.
	C0o	O	TTL	Analog comparator 0 output.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	EPI0S12	I/O	TTL	EPI module 0 signal 12.
	SSI1Rx	I	TTL	SSI module 1 receive.
43	TXOP	O	TTL	TXOP of the Ethernet PHY.
44	VDD	-	Power	Positive supply for I/O and some logic.
45	GND	-	Power	Ground reference for logic and I/O pins.
46	TXON	O	TTL	TXON of the Ethernet PHY.

Table 22-2. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
47	PF0	I/O	TTL	GPIO port F bit 0.
	CAN1Rx	I	TTL	CAN module 1 receive.
	I2S0TXSD	I/O	TTL	I^2S module 0 transmit data.
	U1DSR	I	TTL	UART module 1 Data Set Ready modem output control line.
48	OSC0	I	Analog	Main oscillator crystal input or an external clock reference input.
49	OSC1	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
50	PJ3	I/O	TTL	GPIO port J bit 3.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	EPI0S19	I/O	TTL	EPI module 0 signal 19.
	U1CTS	I	TTL	UART module 1 Clear To Send modem flow control input signal.
51	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
52	PJ4	I/O	TTL	GPIO port J bit 4.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	EPI0S28	I/O	TTL	EPI module 0 signal 28.
	U1DCD	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
53	PJ5	I/O	TTL	GPIO port J bit 5.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	EPI0S29	I/O	TTL	EPI module 0 signal 29.
	U1DSR	I	TTL	UART module 1 Data Set Ready modem output control line.
54	PJ6	I/O	TTL	GPIO port J bit 6.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	EPI0S30	I/O	TTL	EPI module 0 signal 30.
	U1RTS	O	TTL	UART module 1 Request to Send modem flow control output line.
55	PJ7	I/O	TTL	GPIO port J bit 7.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	U1DTR	O	TTL	UART module 1 Data Terminal Ready modem status input signal.
56	VDD	-	Power	Positive supply for I/O and some logic.
57	GND	-	Power	Ground reference for logic and I/O pins.
58	MDIO	I/O	OD	MDIO of the Ethernet PHY.
59	PF3	I/O	TTL	GPIO port F bit 3.
	LED0	O	TTL	Ethernet LED 0.
	SSI1Fss	I/O	TTL	SSI module 1 frame.
60	PF2	I/O	TTL	GPIO port F bit 2.
	LED1	O	TTL	Ethernet LED 1.
	SSI1Clk	I/O	TTL	SSI module 1 clock.
61	PF1	I/O	TTL	GPIO port F bit 1.
	CAN1Tx	O	TTL	CAN module 1 transmit.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	I2S0TXMCLK	I/O	TTL	I^2S module 0 transmit master clock.
	U1RTS	O	TTL	UART module 1 Request to Send modem flow control output line.

Table 22-2. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
62	PH6	I/O	TTL	GPIO port H bit 6.
	EPI0S26	I/O	TTL	EPI module 0 signal 26.
	SSI1Rx	I	TTL	SSI module 1 receive.
63	PH5	I/O	TTL	GPIO port H bit 5.
	EPI0S11	I/O	TTL	EPI module 0 signal 11.
	SSI1Fss	I/O	TTL	SSI module 1 frame.
64	RST	I	TTL	System reset input.
65	PB3	I/O	TTL	GPIO port B bit 3.
	I2C0SDA	I/O	OD	I^2C module 0 data.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
66	PB0	I/O	TTL	GPIO port B bit 0. This pin is not 5-V tolerant.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	USB0ID	I	Analog	This signal senses the state of the USB ID signal. The USB PHY enables an integrated pull-up, and an external element (USB connector) indicates the initial state of the USB controller (pulled down is the A side of the cable and pulled up is the B side).
67	PB1	I/O	TTL	GPIO port B bit 1. This pin is not 5-V tolerant.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	USB0VBUS	I/O	Analog	This signal is used during the session request protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing.
68	VDD	-	Power	Positive supply for I/O and some logic.
69	GND	-	Power	Ground reference for logic and I/O pins.
70	USB0DM	I/O	Analog	Bidirectional differential data pin (D- per USB specification) for USB0.
71	USB0DP	I/O	Analog	Bidirectional differential data pin (D+ per USB specification) for USB0.
72	PB2	I/O	TTL	GPIO port B bit 2.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	I2C0SCL	I/O	OD	I^2C module 0 clock.
	USB0EPEN	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
73	USB0RBIAS	O	Analog	9.1-kΩ resistor (1% precision) used internally for USB analog circuitry.

Table 22-2. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
74	PE0	I/O	TTL	GPIO port E bit 0.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	EPI0S8	I/O	TTL	EPI module 0 signal 8.
	SSI1Clk	I/O	TTL	SSI module 1 clock.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
75	PE1	I/O	TTL	GPIO port E bit 1.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	EPI0S9	I/O	TTL	EPI module 0 signal 9.
	SSI1Fss	I/O	TTL	SSI module 1 frame.
76	PH4	I/O	TTL	GPIO port H bit 4.
	EPI0S10	I/O	TTL	EPI module 0 signal 10.
	SSI1Clk	I/O	TTL	SSI module 1 clock.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
77	PC3	I/O	TTL	GPIO port C bit 3.
	SWO	O	TTL	JTAG TDO and SWO.
	TDO	O	TTL	JTAG TDO and SWO.
78	PC2	I/O	TTL	GPIO port C bit 2.
	TDI	I	TTL	JTAG TDI.
79	PC1	I/O	TTL	GPIO port C bit 1.
	SWDIO	I/O	TTL	JTAG TMS and SWDIO.
	TMS	I	TTL	JTAG TMS and SWDIO.
80	PC0	I/O	TTL	GPIO port C bit 0.
	SWCLK	I	TTL	JTAG/SWD CLK.
	TCK	I	TTL	JTAG/SWD CLK.
81	VDD	-	Power	Positive supply for I/O and some logic.
82	GND	-	Power	Ground reference for logic and I/O pins.
83	PH3	I/O	TTL	GPIO port H bit 3.
	EPI0S0	I/O	TTL	EPI module 0 signal 0.
	USB0EPEN	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
84	PH2	I/O	TTL	GPIO port H bit 2.
	C1o	O	TTL	Analog comparator 1 output.
	EPI0S1	I/O	TTL	EPI module 0 signal 1.
85	PH1	I/O	TTL	GPIO port H bit 1.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	EPI0S7	I/O	TTL	EPI module 0 signal 7.
86	PH0	I/O	TTL	GPIO port H bit 0.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	EPI0S6	I/O	TTL	EPI module 0 signal 6.

Table 22-2. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
87	PJ1	I/O	TTL	GPIO port J bit 1.
	EPI0S17	I/O	TTL	EPI module 0 signal 17.
	I2C1SDA	I/O	OD	I^2C module 1 data.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
88	VDDC	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals. The voltage on this pin is 1.3 V and is supplied by the on-chip LDO. The VDDC pins should only be connected to the LDO pin and an external capacitor as specified in Table 24-6 on page 1197.
89	PB7	I/O	TTL	GPIO port B bit 7.
	NMI	I	TTL	Non-maskable interrupt.
90	PB6	I/O	TTL	GPIO port B bit 6.
	C0+	I	Analog	Analog comparator 0 positive input.
	C0o	O	TTL	Analog comparator 0 output.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	I2S0TXSCK	I/O	TTL	I^2S module 0 transmit clock.
	VREFA	I	Analog	This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AINn signal is converted to 4095. The VREFA input is limited to the range specified in Table 24-23 on page 1208 .
91	PB5	I/O	TTL	GPIO port B bit 5.
	AIN11	I	Analog	Analog-to-digital converter input 11.
	C0o	O	TTL	Analog comparator 0 output.
	C1-	I	Analog	Analog comparator 1 negative input.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	EPI0S22	I/O	TTL	EPI module 0 signal 22.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
92	PB4	I/O	TTL	GPIO port B bit 4.
	AIN10	I	Analog	Analog-to-digital converter input 10.
	C0-	I	Analog	Analog comparator 0 negative input.
	CAN0Rx	I	TTL	CAN module 0 receive.
	EPI0S23	I/O	TTL	EPI module 0 signal 23.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	U2Rx	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.

Table 22-2. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
93	VDD	-	Power	Positive supply for I/O and some logic.
94	GND	-	Power	Ground reference for logic and I/O pins.
95	PE2	I/O	TTL	GPIO port E bit 2.
	AIN9	I	Analog	Analog-to-digital converter input 9.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	EPI0S24	I/O	TTL	EPI module 0 signal 24.
	SSI1Rx	I	TTL	SSI module 1 receive.
96	PE3	I/O	TTL	GPIO port E bit 3.
	AIN8	I	Analog	Analog-to-digital converter input 8.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	EPI0S25	I/O	TTL	EPI module 0 signal 25.
	SSI1Tx	O	TTL	SSI module 1 transmit.
97	PD4	I/O	TTL	GPIO port D bit 4.
	AIN7	I	Analog	Analog-to-digital converter input 7.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	EPI0S19	I/O	TTL	EPI module 0 signal 19.
	I2S0RXSD	I/O	TTL	I^2S module 0 receive data.
	U1RI	I	TTL	UART module 1 Ring Indicator modem status input signal.
98	PD5	I/O	TTL	GPIO port D bit 5.
	AIN6	I	Analog	Analog-to-digital converter input 6.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	EPI0S28	I/O	TTL	EPI module 0 signal 28.
	I2S0RXMCLK	I/O	TTL	I^2S module 0 receive master clock.
	U2Rx	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
99	PD6	I/O	TTL	GPIO port D bit 6.
	AIN5	I	Analog	Analog-to-digital converter input 5.
	EPI0S29	I/O	TTL	EPI module 0 signal 29.
	I2S0TXSCK	I/O	TTL	I^2S module 0 transmit clock.
	U2Tx	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.

Table 22-2. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
100	PD7	I/O	TTL	GPIO port D bit 7.
	AIN4	I	Analog	Analog-to-digital converter input 4.
	C0o	O	TTL	Analog comparator 0 output.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	EPI0S30	I/O	TTL	EPI module 0 signal 30.
	I2S0TXWS	I/O	TTL	I^2S module 0 transmit word select.
	U1DTR	O	TTL	UART module 1 Data Terminal Ready modem status input signal.

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

Table 22-3. Signals by Signal Name

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
AIN0	1	PE7	I	Analog	Analog-to-digital converter input 0.
AIN1	2	PE6	I	Analog	Analog-to-digital converter input 1.
AIN2	5	PE5	I	Analog	Analog-to-digital converter input 2.
AIN3	6	PE4	I	Analog	Analog-to-digital converter input 3.
AIN4	100	PD7	I	Analog	Analog-to-digital converter input 4.
AIN5	99	PD6	I	Analog	Analog-to-digital converter input 5.
AIN6	98	PD5	I	Analog	Analog-to-digital converter input 6.
AIN7	97	PD4	I	Analog	Analog-to-digital converter input 7.
AIN8	96	PE3	I	Analog	Analog-to-digital converter input 8.
AIN9	95	PE2	I	Analog	Analog-to-digital converter input 9.
AIN10	92	PB4	I	Analog	Analog-to-digital converter input 10.
AIN11	91	PB5	I	Analog	Analog-to-digital converter input 11.
AIN12	13	PD3	I	Analog	Analog-to-digital converter input 12.
AIN13	12	PD2	I	Analog	Analog-to-digital converter input 13.
AIN14	11	PD1	I	Analog	Analog-to-digital converter input 14.
AIN15	10	PD0	I	Analog	Analog-to-digital converter input 15.
C0+	90	PB6	I	Analog	Analog comparator 0 positive input.
C0-	92	PB4	I	Analog	Analog comparator 0 negative input.
C0o	24 42 90 91 100	PC5 (3) PF4 (2) PB6 (3) PB5 (1) PD7 (2)	O	TTL	Analog comparator 0 output.
C1+	24	PC5	I	Analog	Analog comparator 1 positive input.
C1-	91	PB5	I	Analog	Analog comparator 1 negative input.
C1o	2 22 24 41 84	PE6 (2) PC7 (7) PC5 (2) PF5 (2) PH2 (2)	O	TTL	Analog comparator 1 output.
C2+	23	PC6	I	Analog	Analog comparator 2 positive input.
C2-	22	PC7	I	Analog	Analog comparator 2 negative input.

Table 22-3. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
C2o	1 23	PE7 (2) PC6 (3)	O	TTL	Analog comparator 2 output.
CAN0Rx	10 30 34 92	PD0 (2) PA4 (5) PA6 (6) PB4 (5)	I	TTL	CAN module 0 receive.
CAN0Tx	11 31 35 91	PD1 (2) PA5 (5) PA7 (6) PB5 (5)	O	TTL	CAN module 0 transmit.
CAN1Rx	47	PF0 (1)	I	TTL	CAN module 1 receive.
CAN1Tx	61	PF1 (1)	O	TTL	CAN module 1 transmit.
CAN2Rx	6	PE4 (2)	I	TTL	CAN module 2 receive.
CAN2Tx	5	PE5 (2)	O	TTL	CAN module 2 transmit.
CCP0	13 22 23 39 42 55 66 72 91 97	PD3 (4) PC7 (4) PC6 (6) PJ2 (9) PF4 (1) PJ7 (10) PB0 (1) PB2 (5) PB5 (4) PD4 (1)	I/O	TTL	Capture/Compare/PWM 0.
CCP1	24 25 34 54 67 90 96 100	PC5 (1) PC4 (9) PA6 (2) PJ6 (10) PB1 (4) PB6 (1) PE3 (1) PD7 (3)	I/O	TTL	Capture/Compare/PWM 1.
CCP2	6 11 25 41 53 67 75 91 95 98	PE4 (6) PD1 (10) PC4 (5) PF5 (1) PJ5 (10) PB1 (1) PE1 (4) PB5 (6) PE2 (5) PD5 (1)	I/O	TTL	Capture/Compare/PWM 2.
CCP3	6 23 24 35 61 72 74 97	PE4 (1) PC6 (1) PC5 (5) PA7 (7) PF1 (10) PB2 (4) PE0 (3) PD4 (2)	I/O	TTL	Capture/Compare/PWM 3.

Table 22-3. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
CCP4	22 25 35 52 95 98	PC7 (1) PC4 (6) PA7 (2) PJ4 (10) PE2 (1) PD5 (2)	I/O	TTL	Capture/Compare/PWM 4.
CCP5	5 12 25 36 90 91	PE5 (1) PD2 (4) PC4 (1) PG7 (8) PB6 (6) PB5 (2)	I/O	TTL	Capture/Compare/PWM 5.
CCP6	10 12 50 75 86 91	PD0 (6) PD2 (2) PJ3 (10) PE1 (5) PH0 (1) PB5 (3)	I/O	TTL	Capture/Compare/PWM 6.
CCP7	11 13 85 90 96	PD1 (6) PD3 (2) PH1 (1) PB6 (2) PE3 (5)	I/O	TTL	Capture/Compare/PWM 7.
EPI0S0	83	PH3 (8)	I/O	TTL	EPI module 0 signal 0.
EPI0S1	84	PH2 (8)	I/O	TTL	EPI module 0 signal 1.
EPI0S2	25	PC4 (8)	I/O	TTL	EPI module 0 signal 2.
EPI0S3	24	PC5 (8)	I/O	TTL	EPI module 0 signal 3.
EPI0S4	23	PC6 (8)	I/O	TTL	EPI module 0 signal 4.
EPI0S5	22	PC7 (8)	I/O	TTL	EPI module 0 signal 5.
EPI0S6	86	PH0 (8)	I/O	TTL	EPI module 0 signal 6.
EPI0S7	85	PH1 (8)	I/O	TTL	EPI module 0 signal 7.
EPI0S8	74	PE0 (8)	I/O	TTL	EPI module 0 signal 8.
EPI0S9	75	PE1 (8)	I/O	TTL	EPI module 0 signal 9.
EPI0S10	76	PH4 (8)	I/O	TTL	EPI module 0 signal 10.
EPI0S11	63	PH5 (8)	I/O	TTL	EPI module 0 signal 11.
EPI0S12	42	PF4 (8)	I/O	TTL	EPI module 0 signal 12.
EPI0S13	19	PG0 (8)	I/O	TTL	EPI module 0 signal 13.
EPI0S14	18	PG1 (8)	I/O	TTL	EPI module 0 signal 14.
EPI0S15	41	PF5 (8)	I/O	TTL	EPI module 0 signal 15.
EPI0S16	14	PJ0 (8)	I/O	TTL	EPI module 0 signal 16.
EPI0S17	87	PJ1 (8)	I/O	TTL	EPI module 0 signal 17.
EPI0S18	39	PJ2 (8)	I/O	TTL	EPI module 0 signal 18.
EPI0S19	50 97	PJ3 (8) PD4 (10)	I/O	TTL	EPI module 0 signal 19.
EPI0S20	12	PD2 (8)	I/O	TTL	EPI module 0 signal 20.
EPI0S21	13	PD3 (8)	I/O	TTL	EPI module 0 signal 21.
EPI0S22	91	PB5 (8)	I/O	TTL	EPI module 0 signal 22.

Table 22-3. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
EPI0S23	92	PB4 (8)	I/O	TTL	EPI module 0 signal 23.
EPI0S24	95	PE2 (8)	I/O	TTL	EPI module 0 signal 24.
EPI0S25	96	PE3 (8)	I/O	TTL	EPI module 0 signal 25.
EPI0S26	62	PH6 (8)	I/O	TTL	EPI module 0 signal 26.
EPI0S27	15	PH7 (8)	I/O	TTL	EPI module 0 signal 27.
EPI0S28	52 98	PJ4 (8) PD5 (10)	I/O	TTL	EPI module 0 signal 28.
EPI0S29	53 99	PJ5 (8) PD6 (10)	I/O	TTL	EPI module 0 signal 29.
EPI0S30	54 100	PJ6 (8) PD7 (10)	I/O	TTL	EPI module 0 signal 30.
EPI0S31	36	PG7 (9)	I/O	TTL	EPI module 0 signal 31.
ERBIAS	33	fixed	O	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
GND	9 21 45 57 69 82 94	fixed	-	Power	Ground reference for logic and I/O pins.
GNDA	4	fixed	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
I2C0SCL	72	PB2 (1)	I/O	OD	I^2C module 0 clock.
I2C0SDA	65	PB3 (1)	I/O	OD	I^2C module 0 data.
I2C1SCL	14 19 26 34	PJ0 (11) PG0 (3) PA0 (8) PA6 (1)	I/O	OD	I^2C module 1 clock.
I2C1SDA	18 27 35 87	PG1 (3) PA1 (8) PA7 (1) PJ1 (11)	I/O	OD	I^2C module 1 data.
I2S0RXMCLK	29 98	PA3 (9) PD5 (8)	I/O	TTL	I^2S module 0 receive master clock.
I2S0RXSCK	10	PD0 (8)	I/O	TTL	I^2S module 0 receive clock.
I2S0RXSD	28 97	PA2 (9) PD4 (8)	I/O	TTL	I^2S module 0 receive data.
I2S0RXWS	11	PD1 (8)	I/O	TTL	I^2S module 0 receive word select.
I2S0TXMCLK	61	PF1 (8)	I/O	TTL	I^2S module 0 transmit master clock.
I2S0TXSCK	30 90 99	PA4 (9) PB6 (9) PD6 (8)	I/O	TTL	I^2S module 0 transmit clock.
I2S0TXSD	5 47	PE5 (9) PF0 (8)	I/O	TTL	I^2S module 0 transmit data.

Table 22-3. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
I2S0TXWS	6 31 100	PE4 (9) PA5 (9) PD7 (8)	I/O	TTL	I^2S module 0 transmit word select.
LDO	7	fixed	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater. The LDO pin must also be connected to the VDDC pins at the board level in addition to the decoupling capacitor(s).
LED0	59	PF3 (1)	O	TTL	Ethernet LED 0.
LED1	60	PF2 (1)	O	TTL	Ethernet LED 1.
MDIO	58	fixed	I/O	OD	MDIO of the Ethernet PHY.
NC	51	fixed	-	-	No connect. Leave the pin electrically unconnected/isolated.
NMI	89	PB7 (4)	I	TTL	Non-maskable interrupt.
OSC0	48	fixed	I	Analog	Main oscillator crystal input or an external clock reference input.
OSC1	49	fixed	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
PA0	26	-	I/O	TTL	GPIO port A bit 0.
PA1	27	-	I/O	TTL	GPIO port A bit 1.
PA2	28	-	I/O	TTL	GPIO port A bit 2.
PA3	29	-	I/O	TTL	GPIO port A bit 3.
PA4	30	-	I/O	TTL	GPIO port A bit 4.
PA5	31	-	I/O	TTL	GPIO port A bit 5.
PA6	34	-	I/O	TTL	GPIO port A bit 6.
PA7	35	-	I/O	TTL	GPIO port A bit 7.
PB0	66	-	I/O	TTL	GPIO port B bit 0. This pin is not 5-V tolerant.
PB1	67	-	I/O	TTL	GPIO port B bit 1. This pin is not 5-V tolerant.
PB2	72	-	I/O	TTL	GPIO port B bit 2.
PB3	65	-	I/O	TTL	GPIO port B bit 3.
PB4	92	-	I/O	TTL	GPIO port B bit 4.
PB5	91	-	I/O	TTL	GPIO port B bit 5.
PB6	90	-	I/O	TTL	GPIO port B bit 6.
PB7	89	-	I/O	TTL	GPIO port B bit 7.
PC0	80	-	I/O	TTL	GPIO port C bit 0.
PC1	79	-	I/O	TTL	GPIO port C bit 1.
PC2	78	-	I/O	TTL	GPIO port C bit 2.
PC3	77	-	I/O	TTL	GPIO port C bit 3.
PC4	25	-	I/O	TTL	GPIO port C bit 4.
PC5	24	-	I/O	TTL	GPIO port C bit 5.
PC6	23	-	I/O	TTL	GPIO port C bit 6.
PC7	22	-	I/O	TTL	GPIO port C bit 7.
PD0	10	-	I/O	TTL	GPIO port D bit 0.

Table 22-3. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
PD1	11	-	I/O	TTL	GPIO port D bit 1.
PD2	12	-	I/O	TTL	GPIO port D bit 2.
PD3	13	-	I/O	TTL	GPIO port D bit 3.
PD4	97	-	I/O	TTL	GPIO port D bit 4.
PD5	98	-	I/O	TTL	GPIO port D bit 5.
PD6	99	-	I/O	TTL	GPIO port D bit 6.
PD7	100	-	I/O	TTL	GPIO port D bit 7.
PE0	74	-	I/O	TTL	GPIO port E bit 0.
PE1	75	-	I/O	TTL	GPIO port E bit 1.
PE2	95	-	I/O	TTL	GPIO port E bit 2.
PE3	96	-	I/O	TTL	GPIO port E bit 3.
PE4	6	-	I/O	TTL	GPIO port E bit 4.
PE5	5	-	I/O	TTL	GPIO port E bit 5.
PE6	2	-	I/O	TTL	GPIO port E bit 6.
PE7	1	-	I/O	TTL	GPIO port E bit 7.
PF0	47	-	I/O	TTL	GPIO port F bit 0.
PF1	61	-	I/O	TTL	GPIO port F bit 1.
PF2	60	-	I/O	TTL	GPIO port F bit 2.
PF3	59	-	I/O	TTL	GPIO port F bit 3.
PF4	42	-	I/O	TTL	GPIO port F bit 4.
PF5	41	-	I/O	TTL	GPIO port F bit 5.
PG0	19	-	I/O	TTL	GPIO port G bit 0.
PG1	18	-	I/O	TTL	GPIO port G bit 1.
PG7	36	-	I/O	TTL	GPIO port G bit 7.
PH0	86	-	I/O	TTL	GPIO port H bit 0.
PH1	85	-	I/O	TTL	GPIO port H bit 1.
PH2	84	-	I/O	TTL	GPIO port H bit 2.
PH3	83	-	I/O	TTL	GPIO port H bit 3.
PH4	76	-	I/O	TTL	GPIO port H bit 4.
PH5	63	-	I/O	TTL	GPIO port H bit 5.
PH6	62	-	I/O	TTL	GPIO port H bit 6.
PH7	15	-	I/O	TTL	GPIO port H bit 7.
PJ0	14	-	I/O	TTL	GPIO port J bit 0.
PJ1	87	-	I/O	TTL	GPIO port J bit 1.
PJ2	39	-	I/O	TTL	GPIO port J bit 2.
PJ3	50	-	I/O	TTL	GPIO port J bit 3.
PJ4	52	-	I/O	TTL	GPIO port J bit 4.
PJ5	53	-	I/O	TTL	GPIO port J bit 5.
PJ6	54	-	I/O	TTL	GPIO port J bit 6.
PJ7	55	-	I/O	TTL	GPIO port J bit 7.
$\overline{\text{RST}}$	64	fixed	I	TTL	System reset input.

Table 22-3. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
RXIN	37	fixed	I	Analog	RXIN of the Ethernet PHY.
RXIP	40	fixed	I	Analog	RXIP of the Ethernet PHY.
SSI0Clk	28	PA2 (1)	I/O	TTL	SSI module 0 clock.
SSI0Fss	29	PA3 (1)	I/O	TTL	SSI module 0 frame.
SSI0Rx	30	PA4 (1)	I	TTL	SSI module 0 receive.
SSI0Tx	31	PA5 (1)	O	TTL	SSI module 0 transmit.
SSI1Clk	60 74 76	PF2 (9) PE0 (2) PH4 (11)	I/O	TTL	SSI module 1 clock.
SSI1Fss	59 63 75	PF3 (9) PH5 (11) PE1 (2)	I/O	TTL	SSI module 1 frame.
SSI1Rx	42 62 95	PF4 (9) PH6 (11) PE2 (2)	I	TTL	SSI module 1 receive.
SSI1Tx	15 41 96	PH7 (11) PF5 (9) PE3 (2)	O	TTL	SSI module 1 transmit.
SWCLK	80	PC0 (3)	I	TTL	JTAG/SWD CLK.
SWDIO	79	PC1 (3)	I/O	TTL	JTAG TMS and SWDIO.
SWO	77	PC3 (3)	O	TTL	JTAG TDO and SWO.
TCK	80	PC0 (3)	I	TTL	JTAG/SWD CLK.
TDI	78	PC2 (3)	I	TTL	JTAG TDI.
TDO	77	PC3 (3)	O	TTL	JTAG TDO and SWO.
TMS	79	PC1 (3)	I	TTL	JTAG TMS and SWDIO.
TXON	46	fixed	O	TTL	TXON of the Ethernet PHY.
TXOP	43	fixed	O	TTL	TXOP of the Ethernet PHY.
U0Rx	26	PA0 (1)	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
U0Tx	27	PA1 (1)	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
U1CTS	2 10 34 50	PE6 (9) PD0 (9) PA6 (9) PJ3 (9)	I	TTL	UART module 1 Clear To Send modem flow control input signal.
U1DCD	1 11 35 52	PE7 (9) PD1 (9) PA7 (9) PJ4 (9)	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
U1DSR	47 53	PF0 (9) PJ5 (9)	I	TTL	UART module 1 Data Set Ready modem output control line.
U1DTR	55 100	PJ7 (9) PD7 (9)	O	TTL	UART module 1 Data Terminal Ready modem status input signal.
U1RI	97	PD4 (9)	I	TTL	UART module 1 Ring Indicator modem status input signal.
U1RTS	54 61	PJ6 (9) PF1 (9)	O	TTL	UART module 1 Request to Send modem flow control output line.

Table 22-3. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
U1Rx	10 12 23 26 66 92	PD0 (5) PD2 (1) PC6 (5) PA0 (9) PB0 (5) PB4 (7)	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
U1Tx	11 13 22 27 67 91	PD1 (5) PD3 (1) PC7 (5) PA1 (9) PB1 (5) PB5 (7)	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
U2Rx	10 19 92 98	PD0 (4) PG0 (1) PB4 (4) PD5 (9)	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
U2Tx	6 11 18 99	PE4 (5) PD1 (4) PG1 (1) PD6 (9)	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
USB0DM	70	fixed	I/O	Analog	Bidirectional differential data pin (D- per USB specification) for USB0.
USB0DP	71	fixed	I/O	Analog	Bidirectional differential data pin (D+ per USB specification) for USB0.
USB0EPEN	19 24 34 72 83	PG0 (7) PC5 (6) PA6 (8) PB2 (8) PH3 (4)	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
USB0ID	66	PB0	I	Analog	This signal senses the state of the USB ID signal. The USB PHY enables an integrated pull-up, and an external element (USB connector) indicates the initial state of the USB controller (pulled down is the A side of the cable and pulled up is the B side).
USB0PFLT	22 23 35 65 74 76 87	PC7 (6) PC6 (7) PA7 (8) PB3 (8) PE0 (9) PH4 (4) PJ1 (9)	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
USB0RBIAS	73	fixed	O	Analog	9.1-kΩ resistor (1% precision) used internally for USB analog circuitry.
USB0VBUS	67	PB1	I/O	Analog	This signal is used during the session request protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing.

Table 22-3. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
VDD	8 20 32 44 56 68 81 93	fixed	-	Power	Positive supply for I/O and some logic.
VDDA	3	fixed	-	Power	The positive supply for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be supplied with a voltage that meets the specification in Table 24-2 on page 1192, regardless of system implementation.
VDDC	38 88	fixed	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals. The voltage on this pin is 1.3 V and is supplied by the on-chip LDO. The VDDC pins should only be connected to the LDO pin and an external capacitor as specified in Table 24-6 on page 1197.
VREFA	90	PB6	I	Analog	This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AINn signal is converted to 4095. The VREFA input is limited to the range specified in Table 24-23 on page 1208 .
XTALNPHY	17	fixed	O	Analog	Ethernet PHY XTALN 25-MHz oscillator crystal output. Leave this pin unconnected when using a single-ended 25-MHz clock input connected to the XTALPPHY pin.
XTALPPHY	16	fixed	I	Analog	Ethernet PHY XTALP 25-MHz oscillator crystal input or external clock reference input.

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

Table 22-4. Signals by Function, Except for GPIO

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
ADC	AIN0	1	I	Analog	Analog-to-digital converter input 0.
	AIN1	2	I	Analog	Analog-to-digital converter input 1.
	AIN2	5	I	Analog	Analog-to-digital converter input 2.
	AIN3	6	I	Analog	Analog-to-digital converter input 3.
	AIN4	100	I	Analog	Analog-to-digital converter input 4.
	AIN5	99	I	Analog	Analog-to-digital converter input 5.
	AIN6	98	I	Analog	Analog-to-digital converter input 6.
	AIN7	97	I	Analog	Analog-to-digital converter input 7.
	AIN8	96	I	Analog	Analog-to-digital converter input 8.
	AIN9	95	I	Analog	Analog-to-digital converter input 9.
	AIN10	92	I	Analog	Analog-to-digital converter input 10.
	AIN11	91	I	Analog	Analog-to-digital converter input 11.
	AIN12	13	I	Analog	Analog-to-digital converter input 12.
	AIN13	12	I	Analog	Analog-to-digital converter input 13.
	AIN14	11	I	Analog	Analog-to-digital converter input 14.
	AIN15	10	I	Analog	Analog-to-digital converter input 15.
	VREFA	90	I	Analog	This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AINn signal is converted to 4095. The VREFA input is limited to the range specified in Table 24-23 on page 1208 .
Analog Comparators	C0+	90	I	Analog	Analog comparator 0 positive input.
	C0-	92	I	Analog	Analog comparator 0 negative input.
	C0o	24 42 90 91 100	O	TTL	Analog comparator 0 output.
	C1+	24	I	Analog	Analog comparator 1 positive input.
	C1-	91	I	Analog	Analog comparator 1 negative input.
	C1o	2 22 24 41 84	O	TTL	Analog comparator 1 output.
	C2+	23	I	Analog	Analog comparator 2 positive input.
	C2-	22	I	Analog	Analog comparator 2 negative input.
	C2o	1 23	O	TTL	Analog comparator 2 output.

Table 22-4. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
Controller Area Network	CAN0Rx	10 30 34 92	I	TTL	CAN module 0 receive.
	CAN0Tx	11 31 35 91	O	TTL	CAN module 0 transmit.
	CAN1Rx	47	I	TTL	CAN module 1 receive.
	CAN1Tx	61	O	TTL	CAN module 1 transmit.
	CAN2Rx	6	I	TTL	CAN module 2 receive.
	CAN2Tx	5	O	TTL	CAN module 2 transmit.
Ethernet	ERBIAS	33	O	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
	LED0	59	O	TTL	Ethernet LED 0.
	LED1	60	O	TTL	Ethernet LED 1.
	MDIO	58	I/O	OD	MDIO of the Ethernet PHY.
	RXIN	37	I	Analog	RXIN of the Ethernet PHY.
	RXIP	40	I	Analog	RXIP of the Ethernet PHY.
	TXON	46	O	TTL	TXON of the Ethernet PHY.
	TXOP	43	O	TTL	TXOP of the Ethernet PHY.
	XTALNPHY	17	O	Analog	Ethernet PHY XTALN 25-MHz oscillator crystal output. Leave this pin unconnected when using a single-ended 25-MHz clock input connected to the XTALPPHY pin.
	XTALPPHY	16	I	Analog	Ethernet PHY XTALP 25-MHz oscillator crystal input or external clock reference input.

Table 22-4. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
External Peripheral Interface	EPI0S0	83	I/O	TTL	EPI module 0 signal 0.
	EPI0S1	84	I/O	TTL	EPI module 0 signal 1.
	EPI0S2	25	I/O	TTL	EPI module 0 signal 2.
	EPI0S3	24	I/O	TTL	EPI module 0 signal 3.
	EPI0S4	23	I/O	TTL	EPI module 0 signal 4.
	EPI0S5	22	I/O	TTL	EPI module 0 signal 5.
	EPI0S6	86	I/O	TTL	EPI module 0 signal 6.
	EPI0S7	85	I/O	TTL	EPI module 0 signal 7.
	EPI0S8	74	I/O	TTL	EPI module 0 signal 8.
	EPI0S9	75	I/O	TTL	EPI module 0 signal 9.
	EPI0S10	76	I/O	TTL	EPI module 0 signal 10.
	EPI0S11	63	I/O	TTL	EPI module 0 signal 11.
	EPI0S12	42	I/O	TTL	EPI module 0 signal 12.
	EPI0S13	19	I/O	TTL	EPI module 0 signal 13.
	EPI0S14	18	I/O	TTL	EPI module 0 signal 14.
	EPI0S15	41	I/O	TTL	EPI module 0 signal 15.
	EPI0S16	14	I/O	TTL	EPI module 0 signal 16.
	EPI0S17	87	I/O	TTL	EPI module 0 signal 17.
	EPI0S18	39	I/O	TTL	EPI module 0 signal 18.
	EPI0S19	50 97	I/O	TTL	EPI module 0 signal 19.
	EPI0S20	12	I/O	TTL	EPI module 0 signal 20.
	EPI0S21	13	I/O	TTL	EPI module 0 signal 21.
	EPI0S22	91	I/O	TTL	EPI module 0 signal 22.
	EPI0S23	92	I/O	TTL	EPI module 0 signal 23.
	EPI0S24	95	I/O	TTL	EPI module 0 signal 24.
	EPI0S25	96	I/O	TTL	EPI module 0 signal 25.
	EPI0S26	62	I/O	TTL	EPI module 0 signal 26.
	EPI0S27	15	I/O	TTL	EPI module 0 signal 27.
	EPI0S28	52 98	I/O	TTL	EPI module 0 signal 28.
	EPI0S29	53 99	I/O	TTL	EPI module 0 signal 29.
	EPI0S30	54 100	I/O	TTL	EPI module 0 signal 30.
	EPI0S31	36	I/O	TTL	EPI module 0 signal 31.

Table 22-4. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
General-Purpose Timers	CCP0	13 22 23 39 42 55 66 72 91 97	I/O	TTL	Capture/Compare/PWM 0.
	CCP1	24 25 34 54 67 90 96 100	I/O	TTL	Capture/Compare/PWM 1.
	CCP2	6 11 25 41 53 67 75 91 95 98	I/O	TTL	Capture/Compare/PWM 2.
	CCP3	6 23 24 35 61 72 74 97	I/O	TTL	Capture/Compare/PWM 3.
	CCP4	22 25 35 52 95 98	I/O	TTL	Capture/Compare/PWM 4.
	CCP5	5 12 25 36 90 91	I/O	TTL	Capture/Compare/PWM 5.
	CCP6	10 12 50 75 86 91	I/O	TTL	Capture/Compare/PWM 6.
	CCP7		I/O	TTL	Capture/Compare/PWM 7.

Table 22-4. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
		11 13 85 90 96			
I2C	I2C0SCL	72	I/O	OD	I^2C module 0 clock.
	I2C0SDA	65	I/O	OD	I^2C module 0 data.
	I2C1SCL	14 19 26 34	I/O	OD	I^2C module 1 clock.
	I2C1SDA	18 27 35 87	I/O	OD	I^2C module 1 data.
I2S	I2S0RXMCLK	29 98	I/O	TTL	I^2S module 0 receive master clock.
	I2S0RXSCK	10	I/O	TTL	I^2S module 0 receive clock.
	I2S0RXSD	28 97	I/O	TTL	I^2S module 0 receive data.
	I2S0RXWS	11	I/O	TTL	I^2S module 0 receive word select.
	I2S0TXMCLK	61	I/O	TTL	I^2S module 0 transmit master clock.
	I2S0TXSCK	30 90 99	I/O	TTL	I^2S module 0 transmit clock.
	I2S0TXSD	5 47	I/O	TTL	I^2S module 0 transmit data.
	I2S0TXWS	6 31 100	I/O	TTL	I^2S module 0 transmit word select.
JTAG/SWD/SWO	SWCLK	80	I	TTL	JTAG/SWD CLK.
	SWDIO	79	I/O	TTL	JTAG TMS and SWDIO.
	SWO	77	O	TTL	JTAG TDO and SWO.
	TCK	80	I	TTL	JTAG/SWD CLK.
	TDI	78	I	TTL	JTAG TDI.
	TDO	77	O	TTL	JTAG TDO and SWO.
	TMS	79	I	TTL	JTAG TMS and SWDIO.

Table 22-4. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
Power	GND	9 21 45 57 69 82 94	-	Power	Ground reference for logic and I/O pins.
	GNDA	4	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
	LDO	7	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater. The LDO pin must also be connected to the VDDC pins at the board level in addition to the decoupling capacitor(s).
	VDD	8 20 32 44 56 68 81 93	-	Power	Positive supply for I/O and some logic.
	VDDA	3	-	Power	The positive supply for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be supplied with a voltage that meets the specification in Table 24-2 on page 1192, regardless of system implementation.
	VDDC	38 88	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals. The voltage on this pin is 1.3 V and is supplied by the on-chip LDO. The VDDC pins should only be connected to the LDO pin and an external capacitor as specified in Table 24-6 on page 1197.
SSI	SSI0Clk	28	I/O	TTL	SSI module 0 clock.
	SSI0Fss	29	I/O	TTL	SSI module 0 frame.
	SSI0Rx	30	I	TTL	SSI module 0 receive.
	SSI0Tx	31	O	TTL	SSI module 0 transmit.
	SSI1Clk	60 74 76	I/O	TTL	SSI module 1 clock.
	SSI1Fss	59 63 75	I/O	TTL	SSI module 1 frame.
	SSI1Rx	42 62 95	I	TTL	SSI module 1 receive.
	SSI1Tx	15 41 96	O	TTL	SSI module 1 transmit.

Table 22-4. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
System Control & Clocks	NMI	89	I	TTL	Non-maskable interrupt.
	OSC0	48	I	Analog	Main oscillator crystal input or an external clock reference input.
	OSC1	49	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
	RST	64	I	TTL	System reset input.
UART	U0Rx	26	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
	U0Tx	27	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
	U1CTS	2 10 34 50	I	TTL	UART module 1 Clear To Send modem flow control input signal.
	U1DCD	1 11 35 52	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
	U1DSR	47 53	I	TTL	UART module 1 Data Set Ready modem output control line.
	U1DTR	55 100	O	TTL	UART module 1 Data Terminal Ready modem status input signal.
	U1RI	97	I	TTL	UART module 1 Ring Indicator modem status input signal.
	U1RTS	54 61	O	TTL	UART module 1 Request to Send modem flow control output line.
	U1Rx	10 12 23 26 66 92	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	U1Tx	11 13 22 27 67 91	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	U2Rx	10 19 92 98	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
	U2Tx	6 11 18 99	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.

Table 22-4. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
USB	USB0DM	70	I/O	Analog	Bidirectional differential data pin (D- per USB specification) for USB0.
	USB0DP	71	I/O	Analog	Bidirectional differential data pin (D+ per USB specification) for USB0.
	USB0EPEN	19 24 34 72 83	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
	USB0ID	66	I	Analog	This signal senses the state of the USB ID signal. The USB PHY enables an integrated pull-up, and an external element (USB connector) indicates the initial state of the USB controller (pulled down is the A side of the cable and pulled up is the B side).
	USB0PFLT	22 23 35 65 74 76 87	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
	USB0RBIAS	73	O	Analog	9.1-kΩ resistor (1% precision) used internally for USB analog circuitry.
	USB0VBUS	67	I/O	Analog	This signal is used during the session request protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing.

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

Table 22-5. GPIO Pins and Alternate Functions

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	10	11
PA0	26	-	U0Rx	-	-	-	-	-	-	I2C1SCL	U1Rx	-	-
PA1	27	-	U0Tx	-	-	-	-	-	-	I2C1SDA	U1Tx	-	-
PA2	28	-	SSI0Clk	-	-	-	-	-	-	-	I2S0RXSD	-	-
PA3	29	-	SSI0Fss	-	-	-	-	-	-	-	I2S0RXMCLK	-	-
PA4	30	-	SSI0Rx	-	-	-	CAN0Rx	-	-	-	I2S0TXSCK	-	-
PA5	31	-	SSI0Tx	-	-	-	CAN0Tx	-	-	-	I2S0TXWS	-	-
PA6	34	-	I2C1SCL	CCP1	-	-	-	CAN0Rx	-	USB0EPEN	U1CTS	-	-
PA7	35	-	I2C1SDA	CCP4	-	-	-	CAN0Tx	CCP3	USB0PFLT	U1DCD	-	-
PB0	66	USB0ID	CCP0	-	-	-	U1Rx	-	-	-	-	-	-
PB1	67	USB0VBUS	CCP2	-	-	CCP1	U1Tx	-	-	-	-	-	-
PB2	72	-	I2C0SCL	-	-	CCP3	CCP0	-	-	USB0EPEN	-	-	-
PB3	65	-	I2C0SDA	-	-	-	-	-	-	USB0PFLT	-	-	-
PB4	92	AIN10 C0-	-	-	-	U2Rx	CAN0Rx	-	U1Rx	EPI0S23	-	-	-
PB5	91	AIN11 C1-	C0o	CCP5	CCP6	CCP0	CAN0Tx	CCP2	U1Tx	EPI0S22	-	-	-

Table 22-5. GPIO Pins and Alternate Functions *(continued)*

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	10	11
PB6	90	VREFA C0+	CCP1	CCP7	C0o	-	-	CCP5	-	-	I2S0TXSCK	-	-
PB7	89	-	-	-	-	NMI	-	-	-	-	-	-	-
PC0	80	-	-	-	TCK SWCLK	-	-	-	-	-	-	-	-
PC1	79	-	-	-	TMS SWDIO	-	-	-	-	-	-	-	-
PC2	78	-	-	-	TDI	-	-	-	-	-	-	-	-
PC3	77	-	-	-	TDO SWO	-	-	-	-	-	-	-	-
PC4	25	-	CCP5	-	-	-	CCP2	CCP4	-	EPI0S2	CCP1	-	-
PC5	24	C1+	CCP1	C1o	C0o	-	CCP3	USB0EPEN	-	EPI0S3	-	-	-
PC6	23	C2+	CCP3	-	C2o	-	U1Rx	CCP0	USB0PFLT	EPI0S4	-	-	-
PC7	22	C2-	CCP4	-	-	CCP0	U1Tx	USB0PFLT	C1o	EPI0S5	-	-	-
PD0	10	AIN15	-	CAN0Rx	-	U2Rx	U1Rx	CCP6	-	I2S0RXSCK	U1CTS	-	-
PD1	11	AIN14	-	CAN0Tx	-	U2Tx	U1Tx	CCP7	-	I2S0RXWS	U1DCD	CCP2	-
PD2	12	AIN13	U1Rx	CCP6	-	CCP5	-	-	-	EPI0S20	-	-	-
PD3	13	AIN12	U1Tx	CCP7	-	CCP0	-	-	-	EPI0S21	-	-	-
PD4	97	AIN7	CCP0	CCP3	-	-	-	-	-	I2S0RXSD	U1RI	EPI0S19	-
PD5	98	AIN6	CCP2	CCP4	-	-	-	-	-	I2S0RXMCLK	U2Rx	EPI0S28	-
PD6	99	AIN5	-	-	-	-	-	-	-	I2S0TXSCK	U2Tx	EPI0S29	-
PD7	100	AIN4	-	C0o	CCP1	-	-	-	-	I2S0TXWS	U1DTR	EPI0S30	-
PE0	74	-	-	SSI1Clk	CCP3	-	-	-	-	EPI0S8	USB0PFLT	-	-
PE1	75	-	-	SSI1Fss	-	CCP2	CCP6	-	-	EPI0S9	-	-	-
PE2	95	AIN9	CCP4	SSI1Rx	-	-	CCP2	-	-	EPI0S24	-	-	-
PE3	96	AIN8	CCP1	SSI1Tx	-	-	CCP7	-	-	EPI0S25	-	-	-
PE4	6	AIN3	CCP3	CAN2Rx	-	-	U2Tx	CCP2	-	-	I2S0TXWS	-	-
PE5	5	AIN2	CCP5	CAN2Tx	-	-	-	-	-	-	I2S0TXSD	-	-
PE6	2	AIN1	-	C1o	-	-	-	-	-	-	U1CTS	-	-
PE7	1	AIN0	-	C2o	-	-	-	-	-	-	U1DCD	-	-
PF0	47	-	CAN1Rx	-	-	-	-	-	-	I2S0TXSD	U1DSR	-	-
PF1	61	-	CAN1Tx	-	-	-	-	-	-	I2S0TXMCLK	U1RTS	CCP3	-
PF2	60	-	LED1	-	-	-	-	-	-	-	SSI1Clk	-	-
PF3	59	-	LED0	-	-	-	-	-	-	-	SSI1Fss	-	-
PF4	42	-	CCP0	C0o	-	-	-	-	-	EPI0S12	SSI1Rx	-	-
PF5	41	-	CCP2	C1o	-	-	-	-	-	EPI0S15	SSI1Tx	-	-
PG0	19	-	U2Rx	-	I2C1SCL	-	-	-	USB0EPEN	EPI0S13	-	-	-
PG1	18	-	U2Tx	-	I2C1SDA	-	-	-	-	EPI0S14	-	-	-
PG7	36	-	-	-	-	-	-	-	-	CCP5	EPI0S31	-	-
PH0	86	-	CCP6	-	-	-	-	-	-	EPI0S6	-	-	-
PH1	85	-	CCP7	-	-	-	-	-	-	EPI0S7	-	-	-

Table 22-5. GPIO Pins and Alternate Functions *(continued)*

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	10	11
PH2	84	-	-	C1o	-	-	-	-	-	EPI0S1	-	-	-
PH3	83	-	-	-	-	USB0EPEN	-	-	-	EPI0S0	-	-	-
PH4	76	-	-	-	-	USB0PFLT	-	-	-	EPI0S10	-	-	SSI1Clk
PH5	63	-	-	-	-	-	-	-	-	EPI0S11	-	-	SSI1Fss
PH6	62	-	-	-	-	-	-	-	-	EPI0S26	-	-	SSI1Rx
PH7	15	-	-	-	-	-	-	-	-	EPI0S27	-	-	SSI1Tx
PJ0	14	-	-	-	-	-	-	-	-	EPI0S16	-	-	I2C1SCL
PJ1	87	-	-	-	-	-	-	-	-	EPI0S17	USB0PFLT	-	I2C1SDA
PJ2	39	-	-	-	-	-	-	-	-	EPI0S18	CCP0	-	-
PJ3	50	-	-	-	-	-	-	-	-	EPI0S19	U1CTS	CCP6	-
PJ4	52	-	-	-	-	-	-	-	-	EPI0S28	U1DCD	CCP4	-
PJ5	53	-	-	-	-	-	-	-	-	EPI0S29	U1DSR	CCP2	-
PJ6	54	-	-	-	-	-	-	-	-	EPI0S30	U1RTS	CCP1	-
PJ7	55	-	-	-	-	-	-	-	-	-	U1DTR	CCP0	-

a. The digital signals that are shaded gray are the power-on default values for the corresponding GPIO pin.

Table 22-6. Possible Pin Assignments for Alternate Functions

# of Possible Assignments	Alternate Function	GPIO Function
one	AIN0	PE7
	AIN1	PE6
	AIN10	PB4
	AIN11	PB5
	AIN12	PD3
	AIN13	PD2
	AIN14	PD1
	AIN15	PD0
	AIN2	PE5
	AIN3	PE4
	AIN4	PD7
	AIN5	PD6
	AIN6	PD5
	AIN7	PD4
	AIN8	PE3
	AIN9	PE2
	C0+	PB6
	C0-	PB4
	C1+	PC5
	C1-	PB5
	C2+	PC6
	C2-	PC7
	CAN1Rx	PF0
	CAN1Tx	PF1
	CAN2Rx	PE4
	CAN2Tx	PE5
	EPI0S0	PH3
	EPI0S1	PH2
	EPI0S10	PH4
	EPI0S11	PH5
	EPI0S12	PF4
	EPI0S13	PG0
	EPI0S14	PG1
	EPI0S15	PF5
	EPI0S16	PJ0
	EPI0S17	PJ1
	EPI0S18	PJ2
	EPI0S2	PC4
	EPI0S20	PD2
	EPI0S21	PD3
	EPI0S22	PB5

Table 22-6. Possible Pin Assignments for Alternate Functions *(continued)*

# of Possible Assignments	Alternate Function	GPIO Function
	EPI0S23	PB4
	EPI0S24	PE2
	EPI0S25	PE3
	EPI0S26	PH6
	EPI0S27	PH7
	EPI0S3	PC5
	EPI0S31	PG7
	EPI0S4	PC6
	EPI0S5	PC7
	EPI0S6	PH0
	EPI0S7	PH1
	EPI0S8	PE0
	EPI0S9	PE1
	I2C0SCL	PB2
	I2C0SDA	PB3
	I2S0RXSCK	PD0
	I2S0RXWS	PD1
	I2S0TXMCLK	PF1
	LED0	PF3
	LED1	PF2
	NMI	PB7
	SSI0Clk	PA2
	SSI0Fss	PA3
	SSI0Rx	PA4
	SSI0Tx	PA5
	SWCLK	PC0
	SWDIO	PC1
	SWO	PC3
	TCK	PC0
	TDI	PC2
	TDO	PC3
	TMS	PC1
	U0Rx	PA0
	U0Tx	PA1
	U1RI	PD4
	USB0ID	PB0
	USB0VBUS	PB1
	VREFA	PB6

Table 22-6. Possible Pin Assignments for Alternate Functions *(continued)*

# of Possible Assignments	Alternate Function	GPIO Function
two	C2o	PC6 PE7
	EPI0S19	PD4 PJ3
	EPI0S28	PD5 PJ4
	EPI0S29	PD6 PJ5
	EPI0S30	PD7 PJ6
	I2S0RXMCLK	PA3 PD5
	I2S0RXSD	PA2 PD4
	I2S0TXSD	PE5 PF0
	U1DSR	PF0 PJ5
	U1DTR	PD7 PJ7
	U1RTS	PF1 PJ6
three	I2S0TXSCK	PA4 PB6 PD6
	I2S0TXWS	PA5 PD7 PE4
	SSI1Clk	PE0 PF2 PH4
	SSI1Fss	PE1 PF3 PH5
	SSI1Rx	PE2 PF4 PH6
	SSI1Tx	PE3 PF5 PH7
four	CAN0Rx	PA4 PA6 PB4 PD0
	CAN0Tx	PA5 PA7 PB5 PD1
	I2C1SCL	PA0 PA6 PG0 PJ0
	I2C1SDA	PA1 PA7 PG1 PJ1
	U1CTS	PA6 PD0 PE6 PJ3
	U1DCD	PA7 PD1 PE7 PJ4
	U2Rx	PB4 PD0 PD5 PG0
	U2Tx	PD1 PD6 PE4 PG1
five	C0o	PB5 PB6 PC5 PD7 PF4
	C1o	PC5 PC7 PE6 PF5 PH2
	CCP7	PB6 PD1 PD3 PE3 PH1
	USB0EPEN	PA6 PB2 PC5 PG0 PH3
six	CCP4	PA7 PC4 PC7 PD5 PE2 PJ4
	CCP5	PB5 PB6 PC4 PD2 PE5 PG7
	CCP6	PB5 PD0 PD2 PE1 PH0 PJ3
	U1Rx	PA0 PB0 PB4 PC6 PD0 PD2
	U1Tx	PA1 PB1 PB5 PC7 PD1 PD3
seven	USB0PFLT	PA7 PB3 PC6 PC7 PE0 PH4 PJ1
eight	CCP1	PA6 PB1 PB6 PC4 PC5 PD7 PE3 PJ6
	CCP3	PA7 PB2 PC5 PC6 PD4 PE0 PE4 PF1
ten	CCP0	PB0 PB2 PB5 PC6 PC7 PD3 PD4 PF4 PJ2 PJ7
	CCP2	PB1 PB5 PC4 PD1 PD5 PE1 PE2 PE4 PF5 PJ5

22.2 108-Ball BGA Package Pin Tables

Table 22-7. Signals by Pin Number

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
A1	PE6	I/O	TTL	GPIO port E bit 6.
	AIN1	I	Analog	Analog-to-digital converter input 1.
	C1o	O	TTL	Analog comparator 1 output.
	U1CTS	I	TTL	UART module 1 Clear To Send modem flow control input signal.
A2	PD7	I/O	TTL	GPIO port D bit 7.
	AIN4	I	Analog	Analog-to-digital converter input 4.
	C0o	O	TTL	Analog comparator 0 output.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	EPI0S30	I/O	TTL	EPI module 0 signal 30.
	I2S0TXWS	I/O	TTL	I^2S module 0 transmit word select.
	U1DTR	O	TTL	UART module 1 Data Terminal Ready modem status input signal.
A3	PD6	I/O	TTL	GPIO port D bit 6.
	AIN5	I	Analog	Analog-to-digital converter input 5.
	EPI0S29	I/O	TTL	EPI module 0 signal 29.
	I2S0TXSCK	I/O	TTL	I^2S module 0 transmit clock.
	U2Tx	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
A4	PE2	I/O	TTL	GPIO port E bit 2.
	AIN9	I	Analog	Analog-to-digital converter input 9.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	EPI0S24	I/O	TTL	EPI module 0 signal 24.
	SSI1Rx	I	TTL	SSI module 1 receive.
A5	GNDA	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
A6	PB4	I/O	TTL	GPIO port B bit 4.
	AIN10	I	Analog	Analog-to-digital converter input 10.
	C0-	I	Analog	Analog comparator 0 negative input.
	CAN0Rx	I	TTL	CAN module 0 receive.
	EPI0S23	I/O	TTL	EPI module 0 signal 23.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	U2Rx	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.

Table 22-7. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
A7	PB6	I/O	TTL	GPIO port B bit 6.
	C0+	I	Analog	Analog comparator 0 positive input.
	C0o	O	TTL	Analog comparator 0 output.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	I2S0TXSCK	I/O	TTL	I^2S module 0 transmit clock.
	VREFA	I	Analog	This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AINn signal is converted to 4095. The VREFA input is limited to the range specified in Table 24-23 on page 1208 .
A8	PB7	I/O	TTL	GPIO port B bit 7.
	NMI	I	TTL	Non-maskable interrupt.
A9	PC0	I/O	TTL	GPIO port C bit 0.
	SWCLK	I	TTL	JTAG/SWD CLK.
	TCK	I	TTL	JTAG/SWD CLK.
A10	PC3	I/O	TTL	GPIO port C bit 3.
	SWO	O	TTL	JTAG TDO and SWO.
	TDO	O	TTL	JTAG TDO and SWO.
A11	PB2	I/O	TTL	GPIO port B bit 2.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	I2C0SCL	I/O	OD	I^2C module 0 clock.
	USB0EPEN	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
A12	PE1	I/O	TTL	GPIO port E bit 1.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	EPI0S9	I/O	TTL	EPI module 0 signal 9.
	SSI1Fss	I/O	TTL	SSI module 1 frame.
B1	PE7	I/O	TTL	GPIO port E bit 7.
	AIN0	I	Analog	Analog-to-digital converter input 0.
	C2o	O	TTL	Analog comparator 2 output.
	U1DCD	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
B2	PE4	I/O	TTL	GPIO port E bit 4.
	AIN3	I	Analog	Analog-to-digital converter input 3.
	CAN2Rx	I	TTL	CAN module 2 receive.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	I2S0TXWS	I/O	TTL	I^2S module 0 transmit word select.
	U2Tx	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.

Table 22-7. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
B3	PE5	I/O	TTL	GPIO port E bit 5.
	AIN2	I	Analog	Analog-to-digital converter input 2.
	CAN2Tx	O	TTL	CAN module 2 transmit.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	I2S0TXSD	I/O	TTL	I^2S module 0 transmit data.
B4	PE3	I/O	TTL	GPIO port E bit 3.
	AIN8	I	Analog	Analog-to-digital converter input 8.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	EPI0S25	I/O	TTL	EPI module 0 signal 25.
	SSI1Tx	O	TTL	SSI module 1 transmit.
B5	PD4	I/O	TTL	GPIO port D bit 4.
	AIN7	I	Analog	Analog-to-digital converter input 7.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	EPI0S19	I/O	TTL	EPI module 0 signal 19.
	I2S0RXSD	I/O	TTL	I^2S module 0 receive data.
	U1RI	I	TTL	UART module 1 Ring Indicator modem status input signal.
B6	PJ1	I/O	TTL	GPIO port J bit 1.
	EPI0S17	I/O	TTL	EPI module 0 signal 17.
	I2C1SDA	I/O	OD	I^2C module 1 data.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
B7	PB5	I/O	TTL	GPIO port B bit 5.
	AIN11	I	Analog	Analog-to-digital converter input 11.
	C0o	O	TTL	Analog comparator 0 output.
	C1-	I	Analog	Analog comparator 1 negative input.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	EPI0S22	I/O	TTL	EPI module 0 signal 22.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
B8	PC2	I/O	TTL	GPIO port C bit 2.
	TDI	I	TTL	JTAG TDI.
B9	PC1	I/O	TTL	GPIO port C bit 1.
	SWDIO	I/O	TTL	JTAG TMS and SWDIO.
	TMS	I	TTL	JTAG TMS and SWDIO.

Table 22-7. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
B10	PH4	I/O	TTL	GPIO port H bit 4.
	EPI0S10	I/O	TTL	EPI module 0 signal 10.
	SSI1Clk	I/O	TTL	SSI module 1 clock.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
B11	PE0	I/O	TTL	GPIO port E bit 0.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	EPI0S8	I/O	TTL	EPI module 0 signal 8.
	SSI1Clk	I/O	TTL	SSI module 1 clock.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
B12	USB0RBIAS	O	Analog	9.1-kΩ resistor (1% precision) used internally for USB analog circuitry.
C1	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
C2	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
C3	VDDC	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals. The voltage on this pin is 1.3 V and is supplied by the on-chip LDO. The VDDC pins should only be connected to the LDO pin and an external capacitor as specified in Table 24-6 on page 1197.
C4	GND	-	Power	Ground reference for logic and I/O pins.
C5	GND	-	Power	Ground reference for logic and I/O pins.
C6	PD5	I/O	TTL	GPIO port D bit 5.
	AIN6	I	Analog	Analog-to-digital converter input 6.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	EPI0S28	I/O	TTL	EPI module 0 signal 28.
	I2S0RXMCLK	I/O	TTL	I^2S module 0 receive master clock.
	U2Rx	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
C7	VDDA	-	Power	The positive supply for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be supplied with a voltage that meets the specification in Table 24-2 on page 1192, regardless of system implementation.
C8	PH1	I/O	TTL	GPIO port H bit 1.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	EPI0S7	I/O	TTL	EPI module 0 signal 7.
C9	PH0	I/O	TTL	GPIO port H bit 0.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	EPI0S6	I/O	TTL	EPI module 0 signal 6.
C10	PG7	I/O	TTL	GPIO port G bit 7.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	EPI0S31	I/O	TTL	EPI module 0 signal 31.

Table 22-7. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
C11	USB0DM	I/O	Analog	Bidirectional differential data pin (D- per USB specification) for USB0.
C12	USB0DP	I/O	Analog	Bidirectional differential data pin (D+ per USB specification) for USB0.
D1	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
D2	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
D3	VDDC	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals. The voltage on this pin is 1.3 V and is supplied by the on-chip LDO. The VDDC pins should only be connected to the LDO pin and an external capacitor as specified in Table 24-6 on page 1197.
D10	PH3	I/O	TTL	GPIO port H bit 3.
	EPI0S0	I/O	TTL	EPI module 0 signal 0.
	USB0EPEN	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
D11	PH2	I/O	TTL	GPIO port H bit 2.
	C1o	O	TTL	Analog comparator 1 output.
	EPI0S1	I/O	TTL	EPI module 0 signal 1.
D12	PB1	I/O	TTL	GPIO port B bit 1. This pin is not 5-V tolerant.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	USB0VBUS	I/O	Analog	This signal is used during the session request protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing.
E1	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
E2	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
E3	LDO	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater. The LDO pin must also be connected to the VDDC pins at the board level in addition to the decoupling capacitor(s).
E10	VDD	-	Power	Positive supply for I/O and some logic.
E11	PB3	I/O	TTL	GPIO port B bit 3.
	I2C0SDA	I/O	OD	I^2C module 0 data.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
E12	PB0	I/O	TTL	GPIO port B bit 0. This pin is not 5-V tolerant.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	USB0ID	I	Analog	This signal senses the state of the USB ID signal. The USB PHY enables an integrated pull-up, and an external element (USB connector) indicates the initial state of the USB controller (pulled down is the A side of the cable and pulled up is the B side).
F1	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
F2	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.

Table 22-7. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
F3	PJ0	I/O	TTL	GPIO port J bit 0.
	EPI0S16	I/O	TTL	EPI module 0 signal 16.
	I2C1SCL	I/O	OD	I^2C module 1 clock.
F10	PH5	I/O	TTL	GPIO port H bit 5.
	EPI0S11	I/O	TTL	EPI module 0 signal 11.
	SSI1Fss	I/O	TTL	SSI module 1 frame.
F11	GND	-	Power	Ground reference for logic and I/O pins.
F12	GND	-	Power	Ground reference for logic and I/O pins.
G1	PD0	I/O	TTL	GPIO port D bit 0.
	AIN15	I	Analog	Analog-to-digital converter input 15.
	CAN0Rx	I	TTL	CAN module 0 receive.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	I2S0RXSCK	I/O	TTL	I^2S module 0 receive clock.
	U1CTS	I	TTL	UART module 1 Clear To Send modem flow control input signal.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	U2Rx	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
G2	PD1	I/O	TTL	GPIO port D bit 1.
	AIN14	I	Analog	Analog-to-digital converter input 14.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	I2S0RXWS	I/O	TTL	I^2S module 0 receive word select.
	U1DCD	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	U2Tx	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
G3	PH6	I/O	TTL	GPIO port H bit 6.
	EPI0S26	I/O	TTL	EPI module 0 signal 26.
	SSI1Rx	I	TTL	SSI module 1 receive.
G10	VDD	-	Power	Positive supply for I/O and some logic.
G11	VDD	-	Power	Positive supply for I/O and some logic.
G12	VDD	-	Power	Positive supply for I/O and some logic.
H1	PD3	I/O	TTL	GPIO port D bit 3.
	AIN12	I	Analog	Analog-to-digital converter input 12.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	EPI0S21	I/O	TTL	EPI module 0 signal 21.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.

Table 22-7. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
H2	PD2	I/O	TTL	GPIO port D bit 2.
	AIN13	I	Analog	Analog-to-digital converter input 13.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	EPI0S20	I/O	TTL	EPI module 0 signal 20.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
H3	PH7	I/O	TTL	GPIO port H bit 7.
	EPI0S27	I/O	TTL	EPI module 0 signal 27.
	SSI1Tx	O	TTL	SSI module 1 transmit.
H10	VDD	-	Power	Positive supply for I/O and some logic.
H11	$\overline{\text{RST}}$	I	TTL	System reset input.
H12	PF1	I/O	TTL	GPIO port F bit 1.
	CAN1Tx	O	TTL	CAN module 1 transmit.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	I2S0TXMCLK	I/O	TTL	I^2S module 0 transmit master clock.
	U1RTS	O	TTL	UART module 1 Request to Send modem flow control output line.
J1	XTALNPHY	O	Analog	Ethernet PHY XTALN 25-MHz oscillator crystal output. Leave this pin unconnected when using a single-ended 25-MHz clock input connected to the XTALPPHY pin.
J2	XTALPPHY	I	Analog	Ethernet PHY XTALP 25-MHz oscillator crystal input or external clock reference input.
J3	ERBIAS	O	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
J10	GND	-	Power	Ground reference for logic and I/O pins.
J11	PF2	I/O	TTL	GPIO port F bit 2.
	LED1	O	TTL	Ethernet LED 1.
	SSI1Clk	I/O	TTL	SSI module 1 clock.
J12	PF3	I/O	TTL	GPIO port F bit 3.
	LED0	O	TTL	Ethernet LED 0.
	SSI1Fss	I/O	TTL	SSI module 1 frame.
K1	PG0	I/O	TTL	GPIO port G bit 0.
	EPI0S13	I/O	TTL	EPI module 0 signal 13.
	I2C1SCL	I/O	OD	I^2C module 1 clock.
	U2Rx	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
	USB0EPEN	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
K2	PG1	I/O	TTL	GPIO port G bit 1.
	EPI0S14	I/O	TTL	EPI module 0 signal 14.
	I2C1SDA	I/O	OD	I^2C module 1 data.
	U2Tx	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.

Table 22-7. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
K3	PF5	I/O	TTL	GPIO port F bit 5.
	C1o	O	TTL	Analog comparator 1 output.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	EPI0S15	I/O	TTL	EPI module 0 signal 15.
	SSI1Tx	O	TTL	SSI module 1 transmit.
K4	PF4	I/O	TTL	GPIO port F bit 4.
	C0o	O	TTL	Analog comparator 0 output.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	EPI0S12	I/O	TTL	EPI module 0 signal 12.
	SSI1Rx	I	TTL	SSI module 1 receive.
K5	GND	-	Power	Ground reference for logic and I/O pins.
K6	PJ2	I/O	TTL	GPIO port J bit 2.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	EPI0S18	I/O	TTL	EPI module 0 signal 18.
K7	VDD	-	Power	Positive supply for I/O and some logic.
K8	VDD	-	Power	Positive supply for I/O and some logic.
K9	VDD	-	Power	Positive supply for I/O and some logic.
K10	GND	-	Power	Ground reference for logic and I/O pins.
K11	PJ4	I/O	TTL	GPIO port J bit 4.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	EPI0S28	I/O	TTL	EPI module 0 signal 28.
	U1DCD	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
K12	PJ5	I/O	TTL	GPIO port J bit 5.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	EPI0S29	I/O	TTL	EPI module 0 signal 29.
	U1DSR	I	TTL	UART module 1 Data Set Ready modem output control line.
L1	PC4	I/O	TTL	GPIO port C bit 4.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	EPI0S2	I/O	TTL	EPI module 0 signal 2.
L2	PC7	I/O	TTL	GPIO port C bit 7.
	C1o	O	TTL	Analog comparator 1 output.
	C2-	I	Analog	Analog comparator 2 negative input.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	EPI0S5	I/O	TTL	EPI module 0 signal 5.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.

Table 22-7. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
L3	PA0	I/O	TTL	GPIO port A bit 0.
	I2C1SCL	I/O	OD	I^2C module 1 clock.
	U0Rx	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
L4	PA3	I/O	TTL	GPIO port A bit 3.
	I2S0RXMCLK	I/O	TTL	I^2S module 0 receive master clock.
	SSI0Fss	I/O	TTL	SSI module 0 frame.
L5	PA4	I/O	TTL	GPIO port A bit 4.
	CAN0Rx	I	TTL	CAN module 0 receive.
	I2S0TXSCK	I/O	TTL	I^2S module 0 transmit clock.
	SSI0Rx	I	TTL	SSI module 0 receive.
L6	PA6	I/O	TTL	GPIO port A bit 6.
	CAN0Rx	I	TTL	CAN module 0 receive.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	I2C1SCL	I/O	OD	I^2C module 1 clock.
	U1CTS	I	TTL	UART module 1 Clear To Send modem flow control input signal.
	USB0EPEN	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
L7	RXIN	I	Analog	RXIN of the Ethernet PHY.
L8	TXON	O	TTL	TXON of the Ethernet PHY.
L9	MDIO	I/O	OD	MDIO of the Ethernet PHY.
L10	PJ6	I/O	TTL	GPIO port J bit 6.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	EPI0S30	I/O	TTL	EPI module 0 signal 30.
	U1RTS	O	TTL	UART module 1 Request to Send modem flow control output line.
L11	OSC0	I	Analog	Main oscillator crystal input or an external clock reference input.
L12	PJ7	I/O	TTL	GPIO port J bit 7.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	U1DTR	O	TTL	UART module 1 Data Terminal Ready modem status input signal.
M1	PC5	I/O	TTL	GPIO port C bit 5.
	C0o	O	TTL	Analog comparator 0 output.
	C1+	I	Analog	Analog comparator 1 positive input.
	C1o	O	TTL	Analog comparator 1 output.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	EPI0S3	I/O	TTL	EPI module 0 signal 3.
	USB0EPEN	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.

Table 22-7. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
M2	PC6	I/O	TTL	GPIO port C bit 6.
	C2+	I	Analog	Analog comparator 2 positive input.
	C2o	O	TTL	Analog comparator 2 output.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	EPI0S4	I/O	TTL	EPI module 0 signal 4.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
M3	PA1	I/O	TTL	GPIO port A bit 1.
	I2C1SDA	I/O	OD	I^2C module 1 data.
	U0Tx	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
M4	PA2	I/O	TTL	GPIO port A bit 2.
	I2S0RXSD	I/O	TTL	I^2S module 0 receive data.
	SSI0Clk	I/O	TTL	SSI module 0 clock.
M5	PA5	I/O	TTL	GPIO port A bit 5.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	I2S0TXWS	I/O	TTL	I^2S module 0 transmit word select.
	SSI0Tx	O	TTL	SSI module 0 transmit.
M6	PA7	I/O	TTL	GPIO port A bit 7.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	I2C1SDA	I/O	OD	I^2C module 1 data.
	U1DCD	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
M7	RXIP	I	Analog	RXIP of the Ethernet PHY.
M8	TXOP	O	TTL	TXOP of the Ethernet PHY.
M9	PF0	I/O	TTL	GPIO port F bit 0.
	CAN1Rx	I	TTL	CAN module 1 receive.
	I2S0TXSD	I/O	TTL	I^2S module 0 transmit data.
	U1DSR	I	TTL	UART module 1 Data Set Ready modem output control line.
M10	PJ3	I/O	TTL	GPIO port J bit 3.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	EPI0S19	I/O	TTL	EPI module 0 signal 19.
	U1CTS	I	TTL	UART module 1 Clear To Send modem flow control input signal.
M11	OSC1	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.

Table 22-7. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
M12	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.

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

Table 22-8. Signals by Signal Name

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
AIN0	B1	PE7	I	Analog	Analog-to-digital converter input 0.
AIN1	A1	PE6	I	Analog	Analog-to-digital converter input 1.
AIN2	B3	PE5	I	Analog	Analog-to-digital converter input 2.
AIN3	B2	PE4	I	Analog	Analog-to-digital converter input 3.
AIN4	A2	PD7	I	Analog	Analog-to-digital converter input 4.
AIN5	A3	PD6	I	Analog	Analog-to-digital converter input 5.
AIN6	C6	PD5	I	Analog	Analog-to-digital converter input 6.
AIN7	B5	PD4	I	Analog	Analog-to-digital converter input 7.
AIN8	B4	PE3	I	Analog	Analog-to-digital converter input 8.
AIN9	A4	PE2	I	Analog	Analog-to-digital converter input 9.
AIN10	A6	PB4	I	Analog	Analog-to-digital converter input 10.
AIN11	B7	PB5	I	Analog	Analog-to-digital converter input 11.
AIN12	H1	PD3	I	Analog	Analog-to-digital converter input 12.
AIN13	H2	PD2	I	Analog	Analog-to-digital converter input 13.
AIN14	G2	PD1	I	Analog	Analog-to-digital converter input 14.
AIN15	G1	PD0	I	Analog	Analog-to-digital converter input 15.
C0+	A7	PB6	I	Analog	Analog comparator 0 positive input.
C0-	A6	PB4	I	Analog	Analog comparator 0 negative input.
C0o	M1 K4 A7 B7 A2	PC5 (3) PF4 (2) PB6 (3) PB5 (1) PD7 (2)	O	TTL	Analog comparator 0 output.
C1+	M1	PC5	I	Analog	Analog comparator 1 positive input.
C1-	B7	PB5	I	Analog	Analog comparator 1 negative input.
C1o	A1 L2 M1 K3 D11	PE6 (2) PC7 (7) PC5 (2) PF5 (2) PH2 (2)	O	TTL	Analog comparator 1 output.
C2+	M2	PC6	I	Analog	Analog comparator 2 positive input.
C2-	L2	PC7	I	Analog	Analog comparator 2 negative input.
C2o	B1 M2	PE7 (2) PC6 (3)	O	TTL	Analog comparator 2 output.
CAN0Rx	G1 L5 L6 A6	PD0 (2) PA4 (5) PA6 (6) PB4 (5)	I	TTL	CAN module 0 receive.

Table 22-8. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
CAN0Tx	G2 M5 M6 B7	PD1 (2) PA5 (5) PA7 (6) PB5 (5)	O	TTL	CAN module 0 transmit.
CAN1Rx	M9	PF0 (1)	I	TTL	CAN module 1 receive.
CAN1Tx	H12	PF1 (1)	O	TTL	CAN module 1 transmit.
CAN2Rx	B2	PE4 (2)	I	TTL	CAN module 2 receive.
CAN2Tx	B3	PE5 (2)	O	TTL	CAN module 2 transmit.
CCP0	H1 L2 M2 K6 K4 L12 E12 A11 B7 B5	PD3 (4) PC7 (4) PC6 (6) PJ2 (9) PF4 (1) PJ7 (10) PB0 (1) PB2 (5) PB5 (4) PD4 (1)	I/O	TTL	Capture/Compare/PWM 0.
CCP1	M1 L1 L6 L10 D12 A7 B4 A2	PC5 (1) PC4 (9) PA6 (2) PJ6 (10) PB1 (4) PB6 (1) PE3 (1) PD7 (3)	I/O	TTL	Capture/Compare/PWM 1.
CCP2	B2 G2 L1 K3 K12 D12 A12 B7 A4 C6	PE4 (6) PD1 (10) PC4 (5) PF5 (1) PJ5 (10) PB1 (1) PE1 (4) PB5 (6) PE2 (5) PD5 (1)	I/O	TTL	Capture/Compare/PWM 2.
CCP3	B2 M2 M1 M6 H12 A11 B11 B5	PE4 (1) PC6 (1) PC5 (5) PA7 (7) PF1 (10) PB2 (4) PE0 (3) PD4 (2)	I/O	TTL	Capture/Compare/PWM 3.
CCP4	L2 L1 M6 K11 A4 C6	PC7 (1) PC4 (6) PA7 (2) PJ4 (10) PE2 (1) PD5 (2)	I/O	TTL	Capture/Compare/PWM 4.

Table 22-8. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
CCP5	B3 H2 L1 C10 A7 B7	PE5 (1) PD2 (4) PC4 (1) PG7 (8) PB6 (6) PB5 (2)	I/O	TTL	Capture/Compare/PWM 5.
CCP6	G1 H2 M10 A12 C9 B7	PD0 (6) PD2 (2) PJ3 (10) PE1 (5) PH0 (1) PB5 (3)	I/O	TTL	Capture/Compare/PWM 6.
CCP7	G2 H1 C8 A7 B4	PD1 (6) PD3 (2) PH1 (1) PB6 (2) PE3 (5)	I/O	TTL	Capture/Compare/PWM 7.
EPI0S0	D10	PH3 (8)	I/O	TTL	EPI module 0 signal 0.
EPI0S1	D11	PH2 (8)	I/O	TTL	EPI module 0 signal 1.
EPI0S2	L1	PC4 (8)	I/O	TTL	EPI module 0 signal 2.
EPI0S3	M1	PC5 (8)	I/O	TTL	EPI module 0 signal 3.
EPI0S4	M2	PC6 (8)	I/O	TTL	EPI module 0 signal 4.
EPI0S5	L2	PC7 (8)	I/O	TTL	EPI module 0 signal 5.
EPI0S6	C9	PH0 (8)	I/O	TTL	EPI module 0 signal 6.
EPI0S7	C8	PH1 (8)	I/O	TTL	EPI module 0 signal 7.
EPI0S8	B11	PE0 (8)	I/O	TTL	EPI module 0 signal 8.
EPI0S9	A12	PE1 (8)	I/O	TTL	EPI module 0 signal 9.
EPI0S10	B10	PH4 (8)	I/O	TTL	EPI module 0 signal 10.
EPI0S11	F10	PH5 (8)	I/O	TTL	EPI module 0 signal 11.
EPI0S12	K4	PF4 (8)	I/O	TTL	EPI module 0 signal 12.
EPI0S13	K1	PG0 (8)	I/O	TTL	EPI module 0 signal 13.
EPI0S14	K2	PG1 (8)	I/O	TTL	EPI module 0 signal 14.
EPI0S15	K3	PF5 (8)	I/O	TTL	EPI module 0 signal 15.
EPI0S16	F3	PJ0 (8)	I/O	TTL	EPI module 0 signal 16.
EPI0S17	B6	PJ1 (8)	I/O	TTL	EPI module 0 signal 17.
EPI0S18	K6	PJ2 (8)	I/O	TTL	EPI module 0 signal 18.
EPI0S19	M10 B5	PJ3 (8) PD4 (10)	I/O	TTL	EPI module 0 signal 19.
EPI0S20	H2	PD2 (8)	I/O	TTL	EPI module 0 signal 20.
EPI0S21	H1	PD3 (8)	I/O	TTL	EPI module 0 signal 21.
EPI0S22	B7	PB5 (8)	I/O	TTL	EPI module 0 signal 22.
EPI0S23	A6	PB4 (8)	I/O	TTL	EPI module 0 signal 23.
EPI0S24	A4	PE2 (8)	I/O	TTL	EPI module 0 signal 24.
EPI0S25	B4	PE3 (8)	I/O	TTL	EPI module 0 signal 25.
EPI0S26	G3	PH6 (8)	I/O	TTL	EPI module 0 signal 26.

Table 22-8. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
EPI0S27	H3	PH7 (8)	I/O	TTL	EPI module 0 signal 27.
EPI0S28	K11 C6	PJ4 (8) PD5 (10)	I/O	TTL	EPI module 0 signal 28.
EPI0S29	K12 A3	PJ5 (8) PD6 (10)	I/O	TTL	EPI module 0 signal 29.
EPI0S30	L10 A2	PJ6 (8) PD7 (10)	I/O	TTL	EPI module 0 signal 30.
EPI0S31	C10	PG7 (9)	I/O	TTL	EPI module 0 signal 31.
ERBIAS	J3	fixed	O	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
GND	C4 C5 K5 K10 J10 F11 F12	fixed	-	Power	Ground reference for logic and I/O pins.
GNDA	A5	fixed	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
I2C0SCL	A11	PB2 (1)	I/O	OD	I^2C module 0 clock.
I2C0SDA	E11	PB3 (1)	I/O	OD	I^2C module 0 data.
I2C1SCL	F3 K1 L3 L6	PJ0 (11) PG0 (3) PA0 (8) PA6 (1)	I/O	OD	I^2C module 1 clock.
I2C1SDA	K2 M3 M6 B6	PG1 (3) PA1 (8) PA7 (1) PJ1 (11)	I/O	OD	I^2C module 1 data.
I2S0RXMCLK	L4 C6	PA3 (9) PD5 (8)	I/O	TTL	I^2S module 0 receive master clock.
I2S0RXSCK	G1	PD0 (8)	I/O	TTL	I^2S module 0 receive clock.
I2S0RXSD	M4 B5	PA2 (9) PD4 (8)	I/O	TTL	I^2S module 0 receive data.
I2S0RXWS	G2	PD1 (8)	I/O	TTL	I^2S module 0 receive word select.
I2S0TXMCLK	H12	PF1 (8)	I/O	TTL	I^2S module 0 transmit master clock.
I2S0TXSCK	L5 A7 A3	PA4 (9) PB6 (9) PD6 (8)	I/O	TTL	I^2S module 0 transmit clock.
I2S0TXSD	B3 M9	PE5 (9) PF0 (8)	I/O	TTL	I^2S module 0 transmit data.
I2S0TXWS	B2 M5 A2	PE4 (9) PA5 (9) PD7 (8)	I/O	TTL	I^2S module 0 transmit word select.

Table 22-8. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
LDO	E3	fixed	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater. The LDO pin must also be connected to the VDDC pins at the board level in addition to the decoupling capacitor(s).
LED0	J12	PF3 (1)	O	TTL	Ethernet LED 0.
LED1	J11	PF2 (1)	O	TTL	Ethernet LED 1.
MDIO	L9	fixed	I/O	OD	MDIO of the Ethernet PHY.
NC	M12 C1 C2 D2 D1 E1 E2 F1 F2	fixed	-	-	No connect. Leave the pin electrically unconnected/isolated.
NMI	A8	PB7 (4)	I	TTL	Non-maskable interrupt.
OSC0	L11	fixed	I	Analog	Main oscillator crystal input or an external clock reference input.
OSC1	M11	fixed	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
PA0	L3	-	I/O	TTL	GPIO port A bit 0.
PA1	M3	-	I/O	TTL	GPIO port A bit 1.
PA2	M4	-	I/O	TTL	GPIO port A bit 2.
PA3	L4	-	I/O	TTL	GPIO port A bit 3.
PA4	L5	-	I/O	TTL	GPIO port A bit 4.
PA5	M5	-	I/O	TTL	GPIO port A bit 5.
PA6	L6	-	I/O	TTL	GPIO port A bit 6.
PA7	M6	-	I/O	TTL	GPIO port A bit 7.
PB0	E12	-	I/O	TTL	GPIO port B bit 0. This pin is not 5-V tolerant.
PB1	D12	-	I/O	TTL	GPIO port B bit 1. This pin is not 5-V tolerant.
PB2	A11	-	I/O	TTL	GPIO port B bit 2.
PB3	E11	-	I/O	TTL	GPIO port B bit 3.
PB4	A6	-	I/O	TTL	GPIO port B bit 4.
PB5	B7	-	I/O	TTL	GPIO port B bit 5.
PB6	A7	-	I/O	TTL	GPIO port B bit 6.
PB7	A8	-	I/O	TTL	GPIO port B bit 7.
PC0	A9	-	I/O	TTL	GPIO port C bit 0.
PC1	B9	-	I/O	TTL	GPIO port C bit 1.
PC2	B8	-	I/O	TTL	GPIO port C bit 2.
PC3	A10	-	I/O	TTL	GPIO port C bit 3.
PC4	L1	-	I/O	TTL	GPIO port C bit 4.
PC5	M1	-	I/O	TTL	GPIO port C bit 5.
PC6	M2	-	I/O	TTL	GPIO port C bit 6.

Table 22-8. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
PC7	L2	-	I/O	TTL	GPIO port C bit 7.
PD0	G1	-	I/O	TTL	GPIO port D bit 0.
PD1	G2	-	I/O	TTL	GPIO port D bit 1.
PD2	H2	-	I/O	TTL	GPIO port D bit 2.
PD3	H1	-	I/O	TTL	GPIO port D bit 3.
PD4	B5	-	I/O	TTL	GPIO port D bit 4.
PD5	C6	-	I/O	TTL	GPIO port D bit 5.
PD6	A3	-	I/O	TTL	GPIO port D bit 6.
PD7	A2	-	I/O	TTL	GPIO port D bit 7.
PE0	B11	-	I/O	TTL	GPIO port E bit 0.
PE1	A12	-	I/O	TTL	GPIO port E bit 1.
PE2	A4	-	I/O	TTL	GPIO port E bit 2.
PE3	B4	-	I/O	TTL	GPIO port E bit 3.
PE4	B2	-	I/O	TTL	GPIO port E bit 4.
PE5	B3	-	I/O	TTL	GPIO port E bit 5.
PE6	A1	-	I/O	TTL	GPIO port E bit 6.
PE7	B1	-	I/O	TTL	GPIO port E bit 7.
PF0	M9	-	I/O	TTL	GPIO port F bit 0.
PF1	H12	-	I/O	TTL	GPIO port F bit 1.
PF2	J11	-	I/O	TTL	GPIO port F bit 2.
PF3	J12	-	I/O	TTL	GPIO port F bit 3.
PF4	K4	-	I/O	TTL	GPIO port F bit 4.
PF5	K3	-	I/O	TTL	GPIO port F bit 5.
PG0	K1	-	I/O	TTL	GPIO port G bit 0.
PG1	K2	-	I/O	TTL	GPIO port G bit 1.
PG7	C10	-	I/O	TTL	GPIO port G bit 7.
PH0	C9	-	I/O	TTL	GPIO port H bit 0.
PH1	C8	-	I/O	TTL	GPIO port H bit 1.
PH2	D11	-	I/O	TTL	GPIO port H bit 2.
PH3	D10	-	I/O	TTL	GPIO port H bit 3.
PH4	B10	-	I/O	TTL	GPIO port H bit 4.
PH5	F10	-	I/O	TTL	GPIO port H bit 5.
PH6	G3	-	I/O	TTL	GPIO port H bit 6.
PH7	H3	-	I/O	TTL	GPIO port H bit 7.
PJ0	F3	-	I/O	TTL	GPIO port J bit 0.
PJ1	B6	-	I/O	TTL	GPIO port J bit 1.
PJ2	K6	-	I/O	TTL	GPIO port J bit 2.
PJ3	M10	-	I/O	TTL	GPIO port J bit 3.
PJ4	K11	-	I/O	TTL	GPIO port J bit 4.
PJ5	K12	-	I/O	TTL	GPIO port J bit 5.
PJ6	L10	-	I/O	TTL	GPIO port J bit 6.

Table 22-8. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
PJ7	L12	-	I/O	TTL	GPIO port J bit 7.
RST	H11	fixed	I	TTL	System reset input.
RXIN	L7	fixed	I	Analog	RXIN of the Ethernet PHY.
RXIP	M7	fixed	I	Analog	RXIP of the Ethernet PHY.
SSI0Clk	M4	PA2 (1)	I/O	TTL	SSI module 0 clock.
SSI0Fss	L4	PA3 (1)	I/O	TTL	SSI module 0 frame.
SSI0Rx	L5	PA4 (1)	I	TTL	SSI module 0 receive.
SSI0Tx	M5	PA5 (1)	O	TTL	SSI module 0 transmit.
SSI1Clk	J11 B11 B10	PF2 (9) PE0 (2) PH4 (11)	I/O	TTL	SSI module 1 clock.
SSI1Fss	J12 F10 A12	PF3 (9) PH5 (11) PE1 (2)	I/O	TTL	SSI module 1 frame.
SSI1Rx	K4 G3 A4	PF4 (9) PH6 (11) PE2 (2)	I	TTL	SSI module 1 receive.
SSI1Tx	H3 K3 B4	PH7 (11) PF5 (9) PE3 (2)	O	TTL	SSI module 1 transmit.
SWCLK	A9	PC0 (3)	I	TTL	JTAG/SWD CLK.
SWDIO	B9	PC1 (3)	I/O	TTL	JTAG TMS and SWDIO.
SWO	A10	PC3 (3)	O	TTL	JTAG TDO and SWO.
TCK	A9	PC0 (3)	I	TTL	JTAG/SWD CLK.
TDI	B8	PC2 (3)	I	TTL	JTAG TDI.
TDO	A10	PC3 (3)	O	TTL	JTAG TDO and SWO.
TMS	B9	PC1 (3)	I	TTL	JTAG TMS and SWDIO.
TXON	L8	fixed	O	TTL	TXON of the Ethernet PHY.
TXOP	M8	fixed	O	TTL	TXOP of the Ethernet PHY.
U0Rx	L3	PA0 (1)	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
U0Tx	M3	PA1 (1)	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
U1CTS	A1 G1 L6 M10	PE6 (9) PD0 (9) PA6 (9) PJ3 (9)	I	TTL	UART module 1 Clear To Send modem flow control input signal.
U1DCD	B1 G2 M6 K11	PE7 (9) PD1 (9) PA7 (9) PJ4 (9)	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
U1DSR	M9 K12	PF0 (9) PJ5 (9)	I	TTL	UART module 1 Data Set Ready modem output control line.
U1DTR	L12 A2	PJ7 (9) PD7 (9)	O	TTL	UART module 1 Data Terminal Ready modem status input signal.

Table 22-8. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
U1RI	B5	PD4 (9)	I	TTL	UART module 1 Ring Indicator modem status input signal.
U1RTS	L10 H12	PJ6 (9) PF1 (9)	O	TTL	UART module 1 Request to Send modem flow control output line.
U1Rx	G1 H2 M2 L3 E12 A6	PD0 (5) PD2 (1) PC6 (5) PA0 (9) PB0 (5) PB4 (7)	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
U1Tx	G2 H1 L2 M3 D12 B7	PD1 (5) PD3 (1) PC7 (5) PA1 (9) PB1 (5) PB5 (7)	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
U2Rx	G1 K1 A6 C6	PD0 (4) PG0 (1) PB4 (4) PD5 (9)	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
U2Tx	B2 G2 K2 A3	PE4 (5) PD1 (4) PG1 (1) PD6 (9)	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
USB0DM	C11	fixed	I/O	Analog	Bidirectional differential data pin (D- per USB specification) for USB0.
USB0DP	C12	fixed	I/O	Analog	Bidirectional differential data pin (D+ per USB specification) for USB0.
USB0EPEN	K1 M1 L6 A11 D10	PG0 (7) PC5 (6) PA6 (8) PB2 (8) PH3 (4)	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
USB0ID	E12	PB0	I	Analog	This signal senses the state of the USB ID signal. The USB PHY enables an integrated pull-up, and an external element (USB connector) indicates the initial state of the USB controller (pulled down is the A side of the cable and pulled up is the B side).
USB0PFLT	L2 M2 M6 E11 B11 B10 B6	PC7 (6) PC6 (7) PA7 (8) PB3 (8) PE0 (9) PH4 (4) PJ1 (9)	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
USB0RBIAS	B12	fixed	O	Analog	9.1-kΩ resistor (1% precision) used internally for USB analog circuitry.
USB0VBUS	D12	PB1	I/O	Analog	This signal is used during the session request protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing.

Table 22-8. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
VDD	K7 G12 K8 K9 H10 G10 E10 G11	fixed	-	Power	Positive supply for I/O and some logic.
VDDA	C7	fixed	-	Power	The positive supply for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be supplied with a voltage that meets the specification in Table 24-2 on page 1192, regardless of system implementation.
VDDC	D3 C3	fixed	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals. The voltage on this pin is 1.3 V and is supplied by the on-chip LDO. The VDDC pins should only be connected to the LDO pin and an external capacitor as specified in Table 24-6 on page 1197.
VREFA	A7	PB6	I	Analog	This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AINn signal is converted to 4095. The VREFA input is limited to the range specified in Table 24-23 on page 1208 .
XTALNPHY	J1	fixed	O	Analog	Ethernet PHY XTALN 25-MHz oscillator crystal output. Leave this pin unconnected when using a single-ended 25-MHz clock input connected to the XTALPPHY pin.
XTALPPHY	J2	fixed	I	Analog	Ethernet PHY XTALP 25-MHz oscillator crystal input or external clock reference input.

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

Table 22-9. Signals by Function, Except for GPIO

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
ADC	AIN0	B1	I	Analog	Analog-to-digital converter input 0.
	AIN1	A1	I	Analog	Analog-to-digital converter input 1.
	AIN2	B3	I	Analog	Analog-to-digital converter input 2.
	AIN3	B2	I	Analog	Analog-to-digital converter input 3.
	AIN4	A2	I	Analog	Analog-to-digital converter input 4.
	AIN5	A3	I	Analog	Analog-to-digital converter input 5.
	AIN6	C6	I	Analog	Analog-to-digital converter input 6.
	AIN7	B5	I	Analog	Analog-to-digital converter input 7.
	AIN8	B4	I	Analog	Analog-to-digital converter input 8.
	AIN9	A4	I	Analog	Analog-to-digital converter input 9.
	AIN10	A6	I	Analog	Analog-to-digital converter input 10.
	AIN11	B7	I	Analog	Analog-to-digital converter input 11.
	AIN12	H1	I	Analog	Analog-to-digital converter input 12.
	AIN13	H2	I	Analog	Analog-to-digital converter input 13.
	AIN14	G2	I	Analog	Analog-to-digital converter input 14.
	AIN15	G1	I	Analog	Analog-to-digital converter input 15.
	VREFA	A7	I	Analog	This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AINn signal is converted to 4095. The VREFA input is limited to the range specified in Table 24-23 on page 1208 .
Analog Comparators	C0+	A7	I	Analog	Analog comparator 0 positive input.
	C0-	A6	I	Analog	Analog comparator 0 negative input.
	C0o	M1 K4 A7 B7 A2	O	TTL	Analog comparator 0 output.
	C1+	M1	I	Analog	Analog comparator 1 positive input.
	C1-	B7	I	Analog	Analog comparator 1 negative input.
	C1o	A1 L2 M1 K3 D11	O	TTL	Analog comparator 1 output.
	C2+	M2	I	Analog	Analog comparator 2 positive input.
	C2-	L2	I	Analog	Analog comparator 2 negative input.
	C2o	B1 M2	O	TTL	Analog comparator 2 output.

Table 22-9. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
Controller Area Network	CAN0Rx	G1 L5 L6 A6	I	TTL	CAN module 0 receive.
	CAN0Tx	G2 M5 M6 B7	O	TTL	CAN module 0 transmit.
	CAN1Rx	M9	I	TTL	CAN module 1 receive.
	CAN1Tx	H12	O	TTL	CAN module 1 transmit.
	CAN2Rx	B2	I	TTL	CAN module 2 receive.
	CAN2Tx	B3	O	TTL	CAN module 2 transmit.
Ethernet	ERBIAS	J3	O	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
	LED0	J12	O	TTL	Ethernet LED 0.
	LED1	J11	O	TTL	Ethernet LED 1.
	MDIO	L9	I/O	OD	MDIO of the Ethernet PHY.
	RXIN	L7	I	Analog	RXIN of the Ethernet PHY.
	RXIP	M7	I	Analog	RXIP of the Ethernet PHY.
	TXON	L8	O	TTL	TXON of the Ethernet PHY.
	TXOP	M8	O	TTL	TXOP of the Ethernet PHY.
	XTALNPHY	J1	O	Analog	Ethernet PHY XTALN 25-MHz oscillator crystal output. Leave this pin unconnected when using a single-ended 25-MHz clock input connected to the XTALPPHY pin.
	XTALPPHY	J2	I	Analog	Ethernet PHY XTALP 25-MHz oscillator crystal input or external clock reference input.

Table 22-9. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
External Peripheral Interface	EPI0S0	D10	I/O	TTL	EPI module 0 signal 0.
	EPI0S1	D11	I/O	TTL	EPI module 0 signal 1.
	EPI0S2	L1	I/O	TTL	EPI module 0 signal 2.
	EPI0S3	M1	I/O	TTL	EPI module 0 signal 3.
	EPI0S4	M2	I/O	TTL	EPI module 0 signal 4.
	EPI0S5	L2	I/O	TTL	EPI module 0 signal 5.
	EPI0S6	C9	I/O	TTL	EPI module 0 signal 6.
	EPI0S7	C8	I/O	TTL	EPI module 0 signal 7.
	EPI0S8	B11	I/O	TTL	EPI module 0 signal 8.
	EPI0S9	A12	I/O	TTL	EPI module 0 signal 9.
	EPI0S10	B10	I/O	TTL	EPI module 0 signal 10.
	EPI0S11	F10	I/O	TTL	EPI module 0 signal 11.
	EPI0S12	K4	I/O	TTL	EPI module 0 signal 12.
	EPI0S13	K1	I/O	TTL	EPI module 0 signal 13.
	EPI0S14	K2	I/O	TTL	EPI module 0 signal 14.
	EPI0S15	K3	I/O	TTL	EPI module 0 signal 15.
	EPI0S16	F3	I/O	TTL	EPI module 0 signal 16.
	EPI0S17	B6	I/O	TTL	EPI module 0 signal 17.
	EPI0S18	K6	I/O	TTL	EPI module 0 signal 18.
	EPI0S19	M10 B5	I/O	TTL	EPI module 0 signal 19.
	EPI0S20	H2	I/O	TTL	EPI module 0 signal 20.
	EPI0S21	H1	I/O	TTL	EPI module 0 signal 21.
	EPI0S22	B7	I/O	TTL	EPI module 0 signal 22.
	EPI0S23	A6	I/O	TTL	EPI module 0 signal 23.
	EPI0S24	A4	I/O	TTL	EPI module 0 signal 24.
	EPI0S25	B4	I/O	TTL	EPI module 0 signal 25.
	EPI0S26	G3	I/O	TTL	EPI module 0 signal 26.
	EPI0S27	H3	I/O	TTL	EPI module 0 signal 27.
	EPI0S28	K11 C6	I/O	TTL	EPI module 0 signal 28.
	EPI0S29	K12 A3	I/O	TTL	EPI module 0 signal 29.
	EPI0S30	L10 A2	I/O	TTL	EPI module 0 signal 30.
	EPI0S31	C10	I/O	TTL	EPI module 0 signal 31.

Table 22-9. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
General-Purpose Timers	CCP0	H1 L2 M2 K6 K4 L12 E12 A11 B7 B5	I/O	TTL	Capture/Compare/PWM 0.
	CCP1	M1 L1 L6 L10 D12 A7 B4 A2	I/O	TTL	Capture/Compare/PWM 1.
	CCP2	B2 G2 L1 K3 K12 D12 A12 B7 A4 C6	I/O	TTL	Capture/Compare/PWM 2.
	CCP3	B2 M2 M1 M6 H12 A11 B11 B5	I/O	TTL	Capture/Compare/PWM 3.
	CCP4	L2 L1 M6 K11 A4 C6	I/O	TTL	Capture/Compare/PWM 4.
	CCP5	B3 H2 L1 C10 A7 B7	I/O	TTL	Capture/Compare/PWM 5.
	CCP6	G1 H2 M10 A12 C9 B7	I/O	TTL	Capture/Compare/PWM 6.
	CCP7		I/O	TTL	Capture/Compare/PWM 7.

Table 22-9. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
		G2 H1 C8 A7 B4			
I2C	I2C0SCL	A11	I/O	OD	I^2C module 0 clock.
	I2C0SDA	E11	I/O	OD	I^2C module 0 data.
	I2C1SCL	F3 K1 L3 L6	I/O	OD	I^2C module 1 clock.
	I2C1SDA	K2 M3 M6 B6	I/O	OD	I^2C module 1 data.
I2S	I2S0RXMCLK	L4 C6	I/O	TTL	I^2S module 0 receive master clock.
	I2S0RXSCK	G1	I/O	TTL	I^2S module 0 receive clock.
	I2S0RXSD	M4 B5	I/O	TTL	I^2S module 0 receive data.
	I2S0RXWS	G2	I/O	TTL	I^2S module 0 receive word select.
	I2S0TXMCLK	H12	I/O	TTL	I^2S module 0 transmit master clock.
	I2S0TXSCK	L5 A7 A3	I/O	TTL	I^2S module 0 transmit clock.
	I2S0TXSD	B3 M9	I/O	TTL	I^2S module 0 transmit data.
	I2S0TXWS	B2 M5 A2	I/O	TTL	I^2S module 0 transmit word select.
JTAG/SWD/SWO	SWCLK	A9	I	TTL	JTAG/SWD CLK.
	SWDIO	B9	I/O	TTL	JTAG TMS and SWDIO.
	SWO	A10	O	TTL	JTAG TDO and SWO.
	TCK	A9	I	TTL	JTAG/SWD CLK.
	TDI	B8	I	TTL	JTAG TDI.
	TDO	A10	O	TTL	JTAG TDO and SWO.
	TMS	B9	I	TTL	JTAG TMS and SWDIO.

Table 22-9. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
Power	GND	C4 C5 K5 K10 J10 F11 F12	-	Power	Ground reference for logic and I/O pins.
	GNDA	A5	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
	LDO	E3	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater. The LDO pin must also be connected to the VDDC pins at the board level in addition to the decoupling capacitor(s).
	VDD	K7 G12 K8 K9 H10 G10 E10 G11	-	Power	Positive supply for I/O and some logic.
	VDDA	C7	-	Power	The positive supply for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be supplied with a voltage that meets the specification in Table 24-2 on page 1192, regardless of system implementation.
	VDDC	D3 C3	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals. The voltage on this pin is 1.3 V and is supplied by the on-chip LDO. The VDDC pins should only be connected to the LDO pin and an external capacitor as specified in Table 24-6 on page 1197.
SSI	SSI0Clk	M4	I/O	TTL	SSI module 0 clock.
	SSI0Fss	L4	I/O	TTL	SSI module 0 frame.
	SSI0Rx	L5	I	TTL	SSI module 0 receive.
	SSI0Tx	M5	O	TTL	SSI module 0 transmit.
	SSI1Clk	J11 B11 B10	I/O	TTL	SSI module 1 clock.
	SSI1Fss	J12 F10 A12	I/O	TTL	SSI module 1 frame.
	SSI1Rx	K4 G3 A4	I	TTL	SSI module 1 receive.
	SSI1Tx	H3 K3 B4	O	TTL	SSI module 1 transmit.

Table 22-9. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
System Control & Clocks	NMI	A8	I	TTL	Non-maskable interrupt.
	OSC0	L11	I	Analog	Main oscillator crystal input or an external clock reference input.
	OSC1	M11	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
	RST	H11	I	TTL	System reset input.
UART	U0Rx	L3	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
	U0Tx	M3	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
	U1CTS	A1 G1 L6 M10	I	TTL	UART module 1 Clear To Send modem flow control input signal.
	U1DCD	B1 G2 M6 K11	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
	U1DSR	M9 K12	I	TTL	UART module 1 Data Set Ready modem output control line.
	U1DTR	L12 A2	O	TTL	UART module 1 Data Terminal Ready modem status input signal.
	U1RI	B5	I	TTL	UART module 1 Ring Indicator modem status input signal.
	U1RTS	L10 H12	O	TTL	UART module 1 Request to Send modem flow control output line.
	U1Rx	G1 H2 M2 L3 E12 A6	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	U1Tx	G2 H1 L2 M3 D12 B7	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	U2Rx	G1 K1 A6 C6	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
	U2Tx	B2 G2 K2 A3	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.

Table 22-9. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
USB	USB0DM	C11	I/O	Analog	Bidirectional differential data pin (D- per USB specification) for USB0.
	USB0DP	C12	I/O	Analog	Bidirectional differential data pin (D+ per USB specification) for USB0.
	USB0EPEN	K1 M1 L6 A11 D10	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
	USB0ID	E12	I	Analog	This signal senses the state of the USB ID signal. The USB PHY enables an integrated pull-up, and an external element (USB connector) indicates the initial state of the USB controller (pulled down is the A side of the cable and pulled up is the B side).
	USB0PFLT	L2 M2 M6 E11 B11 B10 B6	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
	USB0RBIAS	B12	O	Analog	9.1-kΩ resistor (1% precision) used internally for USB analog circuitry.
	USB0VBUS	D12	I/O	Analog	This signal is used during the session request protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing.

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

Table 22-10. GPIO Pins and Alternate Functions

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	10	11
PA0	L3	-	U0Rx	-	-	-	-	-	-	I2C1SCL	U1Rx	-	-
PA1	M3	-	U0Tx	-	-	-	-	-	-	I2C1SDA	U1Tx	-	-
PA2	M4	-	SSI0Clk	-	-	-	-	-	-	-	I2S0RXSD	-	-
PA3	L4	-	SSI0Fss	-	-	-	-	-	-	-	I2S0RXMCLK	-	-
PA4	L5	-	SSI0Rx	-	-	-	CAN0Rx	-	-	-	I2S0TXSCK	-	-
PA5	M5	-	SSI0Tx	-	-	-	CAN0Tx	-	-	-	I2S0TXWS	-	-
PA6	L6	-	I2C1SCL	CCP1	-	-	-	CAN0Rx	-	USB0EPEN	U1CTS	-	-
PA7	M6	-	I2C1SDA	CCP4	-	-	-	CAN0Tx	CCP3	USB0PFLT	U1DCD	-	-
PB0	E12	USB0ID	CCP0	-	-	-	U1Rx	-	-	-	-	-	-
PB1	D12	USB0VBUS	CCP2	-	-	CCP1	U1Tx	-	-	-	-	-	-
PB2	A11	-	I2C0SCL	-	-	CCP3	CCP0	-	-	USB0EPEN	-	-	-
PB3	E11	-	I2C0SDA	-	-	-	-	-	-	USB0PFLT	-	-	-
PB4	A6	AIN10 C0-	-	-	-	U2Rx	CAN0Rx	-	U1Rx	EPI0S23	-	-	-
PB5	B7	AIN11 C1-	C0o	CCP5	CCP6	CCP0	CAN0Tx	CCP2	U1Tx	EPI0S22	-	-	-

Table 22-10. GPIO Pins and Alternate Functions *(continued)*

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	10	11
PB6	A7	VREFA C0+	CCP1	CCP7	C0o	-	-	CCP5	-	-	I2S0TXSCK	-	-
PB7	A8	-	-	-	-	NMI	-	-	-	-	-	-	-
PC0	A9	-	-	-	TCK SWCLK	-	-	-	-	-	-	-	-
PC1	B9	-	-	-	TMS SWDIO	-	-	-	-	-	-	-	-
PC2	B8	-	-	-	TDI	-	-	-	-	-	-	-	-
PC3	A10	-	-	-	TDO SWO	-	-	-	-	-	-	-	-
PC4	L1	-	CCP5	-	-	-	CCP2	CCP4	-	EPI0S2	CCP1	-	-
PC5	M1	C1+	CCP1	C1o	C0o	-	CCP3	USB0EPEN	-	EPI0S3	-	-	-
PC6	M2	C2+	CCP3	-	C2o	-	U1Rx	CCP0	USB0PFLT	EPI0S4	-	-	-
PC7	L2	C2-	CCP4	-	-	CCP0	U1Tx	USB0PFLT	C1o	EPI0S5	-	-	-
PD0	G1	AIN15	-	CAN0Rx	-	U2Rx	U1Rx	CCP6	-	I2S0RXSCK	U1CTS	-	-
PD1	G2	AIN14	-	CAN0Tx	-	U2Tx	U1Tx	CCP7	-	I2S0RXWS	U1DCD	CCP2	-
PD2	H2	AIN13	U1Rx	CCP6	-	CCP5	-	-	-	EPI0S20	-	-	-
PD3	H1	AIN12	U1Tx	CCP7	-	CCP0	-	-	-	EPI0S21	-	-	-
PD4	B5	AIN7	CCP0	CCP3	-	-	-	-	-	I2S0RXSD	U1RI	EPI0S19	-
PD5	C6	AIN6	CCP2	CCP4	-	-	-	-	-	I2S0RXMCLK	U2Rx	EPI0S28	-
PD6	A3	AIN5	-	-	-	-	-	-	-	I2S0TXSCK	U2Tx	EPI0S29	-
PD7	A2	AIN4	-	C0o	CCP1	-	-	-	-	I2S0TXWS	U1DTR	EPI0S30	-
PE0	B11	-	-	SSI1Clk	CCP3	-	-	-	-	EPI0S8	USB0PFLT	-	-
PE1	A12	-	-	SSI1Fss	-	CCP2	CCP6	-	-	EPI0S9	-	-	-
PE2	A4	AIN9	CCP4	SSI1Rx	-	-	CCP2	-	-	EPI0S24	-	-	-
PE3	B4	AIN8	CCP1	SSI1Tx	-	-	CCP7	-	-	EPI0S25	-	-	-
PE4	B2	AIN3	CCP3	CAN2Rx	-	-	U2Tx	CCP2	-	-	I2S0TXWS	-	-
PE5	B3	AIN2	CCP5	CAN2Tx	-	-	-	-	-	-	I2S0TXSD	-	-
PE6	A1	AIN1	-	C1o	-	-	-	-	-	-	U1CTS	-	-
PE7	B1	AIN0	-	C2o	-	-	-	-	-	-	U1DCD	-	-
PF0	M9	-	CAN1Rx	-	-	-	-	-	-	I2S0TXSD	U1DSR	-	-
PF1	H12	-	CAN1Tx	-	-	-	-	-	-	I2S0TXMCLK	U1RTS	CCP3	-
PF2	J11	-	LED1	-	-	-	-	-	-	-	SSI1Clk	-	-
PF3	J12	-	LED0	-	-	-	-	-	-	-	SSI1Fss	-	-
PF4	K4	-	CCP0	C0o	-	-	-	-	-	EPI0S12	SSI1Rx	-	-
PF5	K3	-	CCP2	C1o	-	-	-	-	-	EPI0S15	SSI1Tx	-	-
PG0	K1	-	U2Rx	-	I2C1SCL	-	-	-	USB0EPEN	EPI0S13	-	-	-
PG1	K2	-	U2Tx	-	I2C1SDA	-	-	-	-	EPI0S14	-	-	-
PG7	C10	-	-	-	-	-	-	-	-	CCP5	EPI0S31	-	-
PH0	C9	-	CCP6	-	-	-	-	-	-	EPI0S6	-	-	-
PH1	C8	-	CCP7	-	-	-	-	-	-	EPI0S7	-	-	-

Table 22-10. GPIO Pins and Alternate Functions *(continued)*

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	10	11
PH2	D11	-	-	C1o	-	-	-	-	-	EPI0S1	-	-	-
PH3	D10	-	-	-	-	USB0EPEN	-	-	-	EPI0S0	-	-	-
PH4	B10	-	-	-	-	USB0PFLT	-	-	-	EPI0S10	-	-	SSI1Clk
PH5	F10	-	-	-	-	-	-	-	-	EPI0S11	-	-	SSI1Fss
PH6	G3	-	-	-	-	-	-	-	-	EPI0S26	-	-	SSI1Rx
PH7	H3	-	-	-	-	-	-	-	-	EPI0S27	-	-	SSI1Tx
PJ0	F3	-	-	-	-	-	-	-	-	EPI0S16	-	-	I2C1SCL
PJ1	B6	-	-	-	-	-	-	-	-	EPI0S17	USB0PFLT	-	I2C1SDA
PJ2	K6	-	-	-	-	-	-	-	-	EPI0S18	CCP0	-	-
PJ3	M10	-	-	-	-	-	-	-	-	EPI0S19	U1CTS	CCP6	-
PJ4	K11	-	-	-	-	-	-	-	-	EPI0S28	U1DCD	CCP4	-
PJ5	K12	-	-	-	-	-	-	-	-	EPI0S29	U1DSR	CCP2	-
PJ6	L10	-	-	-	-	-	-	-	-	EPI0S30	U1RTS	CCP1	-
PJ7	L12	-	-	-	-	-	-	-	-	-	U1DTR	CCP0	-

a. The digital signals that are shaded gray are the power-on default values for the corresponding GPIO pin.

Table 22-11. Possible Pin Assignments for Alternate Functions

# of Possible Assignments	Alternate Function	GPIO Function
one	AIN0	PE7
	AIN1	PE6
	AIN10	PB4
	AIN11	PB5
	AIN12	PD3
	AIN13	PD2
	AIN14	PD1
	AIN15	PD0
	AIN2	PE5
	AIN3	PE4
	AIN4	PD7
	AIN5	PD6
	AIN6	PD5
	AIN7	PD4
	AIN8	PE3
	AIN9	PE2
	C0+	PB6
	C0-	PB4
	C1+	PC5
	C1-	PB5
	C2+	PC6
	C2-	PC7
	CAN1Rx	PF0
	CAN1Tx	PF1
	CAN2Rx	PE4
	CAN2Tx	PE5
	EPI0S0	PH3
	EPI0S1	PH2
	EPI0S10	PH4
	EPI0S11	PH5
	EPI0S12	PF4
	EPI0S13	PG0
	EPI0S14	PG1
	EPI0S15	PF5
	EPI0S16	PJ0
	EPI0S17	PJ1
	EPI0S18	PJ2
	EPI0S2	PC4
	EPI0S20	PD2
	EPI0S21	PD3
	EPI0S22	PB5

Table 22-11. Possible Pin Assignments for Alternate Functions *(continued)*

# of Possible Assignments	Alternate Function	GPIO Function
	EPI0S23	PB4
	EPI0S24	PE2
	EPI0S25	PE3
	EPI0S26	PH6
	EPI0S27	PH7
	EPI0S3	PC5
	EPI0S31	PG7
	EPI0S4	PC6
	EPI0S5	PC7
	EPI0S6	PH0
	EPI0S7	PH1
	EPI0S8	PE0
	EPI0S9	PE1
	I2C0SCL	PB2
	I2C0SDA	PB3
	I2S0RXSCK	PD0
	I2S0RXWS	PD1
	I2S0TXMCLK	PF1
	LED0	PF3
	LED1	PF2
	NMI	PB7
	SSI0Clk	PA2
	SSI0Fss	PA3
	SSI0Rx	PA4
	SSI0Tx	PA5
	SWCLK	PC0
	SWDIO	PC1
	SWO	PC3
	TCK	PC0
	TDI	PC2
	TDO	PC3
	TMS	PC1
	U0Rx	PA0
	U0Tx	PA1
	U1RI	PD4
	USB0ID	PB0
	USB0VBUS	PB1
	VREFA	PB6

Table 22-11. Possible Pin Assignments for Alternate Functions *(continued)*

# of Possible Assignments	Alternate Function	GPIO Function
two	C2o	PE7 PC6
	EPI0S19	PJ3 PD4
	EPI0S28	PJ4 PD5
	EPI0S29	PJ5 PD6
	EPI0S30	PJ6 PD7
	I2S0RXMCLK	PA3 PD5
	I2S0RXSD	PA2 PD4
	I2S0TXSD	PE5 PF0
	U1DSR	PF0 PJ5
	U1DTR	PJ7 PD7
	U1RTS	PJ6 PF1
three	I2S0TXSCK	PA4 PB6 PD6
	I2S0TXWS	PE4 PA5 PD7
	SSI1Clk	PF2 PE0 PH4
	SSI1Fss	PF3 PH5 PE1
	SSI1Rx	PF4 PH6 PE2
	SSI1Tx	PH7 PF5 PE3
four	CAN0Rx	PD0 PA4 PA6 PB4
	CAN0Tx	PD1 PA5 PA7 PB5
	I2C1SCL	PJ0 PG0 PA0 PA6
	I2C1SDA	PG1 PA1 PA7 PJ1
	U1CTS	PE6 PD0 PA6 PJ3
	U1DCD	PE7 PD1 PA7 PJ4
	U2Rx	PD0 PG0 PB4 PD5
	U2Tx	PE4 PD1 PG1 PD6
five	C0o	PC5 PF4 PB6 PB5 PD7
	C1o	PE6 PC7 PC5 PF5 PH2
	CCP7	PD1 PD3 PH1 PB6 PE3
	USB0EPEN	PG0 PC5 PA6 PB2 PH3
six	CCP4	PC7 PC4 PA7 PJ4 PE2 PD5
	CCP5	PE5 PD2 PC4 PG7 PB6 PB5
	CCP6	PD0 PD2 PJ3 PE1 PH0 PB5
	U1Rx	PD0 PD2 PC6 PA0 PB0 PB4
	U1Tx	PD1 PD3 PC7 PA1 PB1 PB5
seven	USB0PFLT	PC7 PC6 PA7 PB3 PE0 PH4 PJ1
eight	CCP1	PC5 PC4 PA6 PJ6 PB1 PB6 PE3 PD7
	CCP3	PE4 PC6 PC5 PA7 PF1 PB2 PE0 PD4
ten	CCP0	PD3 PC7 PC6 PJ2 PF4 PJ7 PB0 PB2 PB5 PD4
	CCP2	PE4 PD1 PC4 PF5 PJ5 PB1 PE1 PB5 PE2 PD5

22.3 Connections for Unused Signals

Table 22-12 on page 1189 shows how to handle signals for functions that are not used in a particular system implementation for devices that are in a 100-pin LQFP package. Two options are shown in the table: an acceptable practice and a preferred practice for reduced power consumption and improved EMC characteristics. If a module is not used in a system, and its inputs are grounded, it is important that the clock to the module is never enabled by setting the corresponding bit in the **RCGCx** register.

Table 22-12. Connections for Unused Signals (100-Pin LQFP)

Function	Signal Name	Pin Number	Acceptable Practice	Preferred Practice
Ethernet	ERBIAS	33	Connect to GND through 12.4-kΩ resistor.	Connect to GND through 12.4-kΩ resistor.
	RXIN	37	NC	GND
	RXIP	40	NC	GND
	TXON	46	NC	GND
	TXOP	43	NC	GND
	XTALNPHY[a]	17	NC	NC
	XTALPPHY[a]	16	NC	GND
GPIO	All unused GPIOs	-	NC	GND
No Connects	NC	-	NC	NC
System Control	OSC0	48	NC	GND
	OSC1	49	NC	NC
	$\overline{\text{RST}}$	64	Pull up as shown in Figure 5-1 on page 189	Connect through a capacitor to GND as close to pin as possible
USB	USB0DM	70	NC	GND
	USB0DP	71	NC	GND
	USB0RBIAS	73	Connect to GND through 10-kΩ resistor.	Connect to GND through 10-kΩ resistor.

a. Note that the Ethernet PHY is powered up by default. The PHY cannot be powered down unless a clock source is provided and the MDIO pin is pulled up through a 10-kΩ resistor.

Table 22-13 on page 1190 shows how to handle signals for functions that are not used in a particular system implementation for devices that are in a 108-ball BGA package. Two options are shown in the table: an acceptable practice and a preferred practice for reduced power consumption and improved EMC characteristics. If a module is not used in a system, and its inputs are grounded, it is important that the clock to the module is never enabled by setting the corresponding bit in the **RCGCx** register.

Table 22-13. Connections for Unused Signals (108-Ball BGA)

Function	Signal Name	Pin Number	Acceptable Practice	Preferred Practice
Ethernet	ERBIAS	J3	Connect to GND through 12.4-kΩ resistor.	Connect to GND through 12.4-kΩ resistor.
	RXIN	L7	NC	GND
	RXIP	M7	NC	GND
	TXON	L8	NC	GND
	TXOP	M8	NC	GND
	XTALNPHY[a]	J1	NC	NC
	XTALPPHY[a]	J2	NC	GND
GPIO	All unused GPIOs	-	NC	GND
No Connects	NC	-	NC	NC
System Control	OSC0	L11	NC	GND
	OSC1	M11	NC	NC
	$\overline{\text{RST}}$	H11	Pull up as shown in Figure 5-1 on page 189	Connect through a capacitor to GND as close to pin as possible
USB	USB0RBIAS	B12	Connect to GND through 10-kΩ resistor.	Connect to GND through 10-kΩ resistor.
	USB0DM	C11	NC	GND
	USB0DP	C12	NC	GND

a. Note that the Ethernet PHY is powered up by default. The PHY cannot be powered down unless a clock source is provided and the MDIO pin is pulled up through a 10-kΩ resistor.

23 Operating Characteristics

Table 23-1. Temperature Characteristics

Characteristic	Symbol	Value	Unit
Industrial operating temperature range	T_A	-40 to +85	°C
Unpowered storage temperature range	T_S	-65 to +150	°C

Table 23-2. Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance (junction to ambient)[a]	Θ_{JA}	33 (100LQFP) 31 (108BGA)	°C/W
Junction temperature, -40 to +125[b]	T_J	$T_A + (P \bullet \Theta_{JA})$	°C

a. Junction to ambient thermal resistance θ_{JA} numbers are determined by a package simulator.

b. Power dissipation is a function of temperature.

Table 23-3. ESD Absolute Maximum Ratings[a]

Parameter Name	Min	Nom	Max	Unit
V_{ESDHBM}	-	-	2.0	kV
V_{ESDCDM}	-	-	500	V

a. All Stellaris® parts are ESD tested following the JEDEC standard.

24 Electrical Characteristics

24.1 Maximum Ratings

The maximum ratings are the limits to which the device can be subjected without permanently damaging the device. Device reliability may be adversely affected by exposure to absolute-maximum ratings for extended periods.

Note: The device is not guaranteed to operate properly at the maximum ratings.

Table 24-1. Maximum Ratings

Parameter	Parameter Name[a]	Value		Unit
		Min	Max	
V_{DD}	V_{DD} supply voltage	0	4	V
V_{DDA}	V_{DDA} supply voltage	0	4	V
V_{IN_GPIO}	Input voltage[b]	-0.3	5.5	V
	Input voltage for PB0 and PB1 when configured as GPIO	-0.3	V_{DD} + 0.3	V
$I_{GPIOMAX}$	Maximum current per output pin	-	25	mA
V_{NON}	Maximum input voltage on a non-power pin when the microcontroller is unpowered	-	300	mV

a. Voltages are measured with respect to GND.

b. Applies to static and dynamic signals including overshoot.

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 (see “Connections for Unused Signals” on page 1189).

24.2 Recommended Operating Conditions

For special high-current applications, the GPIO output buffers may be used with the following restrictions. With the GPIO pins configured as 8-mA output drivers, a total of four GPIO outputs may be used to sink current loads up to 18 mA each. At 18-mA sink current loading, the V_{OL} value is specified as 1.2 V. The high-current GPIO package pins must be selected such that there are only a maximum of two per side of the physical package or BGA pin group with the total number of high-current GPIO outputs not exceeding four for the entire package.

Table 24-2. Recommended DC Operating Conditions

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{DD}	V_{DD} supply voltage	3.0	3.3	3.6	V
V_{DDA}	V_{DDA} supply voltage	3.0	3.3	3.6	V
V_{DDC}	V_{DDC} supply voltage, run mode	1.235	1.3	1.365	V
V_{IH}	High-level input voltage	2.1	-	5.0	V
V_{IL}	Low-level input voltage	-0.3	-	1.2	V
V_{OH}	High-level output voltage	2.4	-	-	V

Table 24-2. Recommended DC Operating Conditions *(continued)*

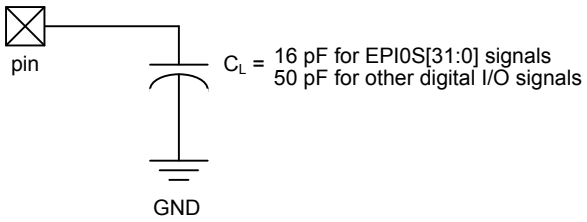
Parameter	Parameter Name	Min	Nom	Max	Unit
V_{OL}	Low-level output voltage	-	-	0.4	V
I_{OH}	High-level source current, V_{OH}=2.4 V[a]				
	2-mA Drive	-2.0	-	-	mA
	4-mA Drive	-4.0	-	-	mA
	8-mA Drive	-8.0	-	-	mA
I_{OL}	Low-level sink current, V_{OL}=0.4 V[a]				
	2-mA Drive	2.0	-	-	mA
	4-mA Drive	4.0	-	-	mA
	8-mA Drive	8.0	-	-	mA
	8-mA Drive, V_{OL}=1.2 V	18.0	-	-	mA

a. I_O specifications reflect the maximum current where the corresponding output voltage meets the V_{OH} /V_{OL} thresholds. I_O current can exceed these limits (subject to absolute maximum ratings).

24.3 Load Conditions

Unless otherwise specified, the following conditions are true for all timing measurements.

Figure 24-1. Load Conditions



24.4 JTAG and Boundary Scan

Table 24-3. JTAG Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
J1	F_{TCK}	TCK operational clock frequency[a]	0	-	10	MHz
J2	T_{TCK}	TCK operational clock period	100	-	-	ns
J3	T_{TCK_LOW}	TCK clock Low time	-	t_{TCK}/2	-	ns
J4	T_{TCK_HIGH}	TCK clock High time	-	t_{TCK}/2	-	ns
J5	T_{TCK_R}	TCK rise time	0	-	10	ns
J6	T_{TCK_F}	TCK fall time	0	-	10	ns
J7	T_{TMS_SU}	TMS setup time to TCK rise	20	-	-	ns
J8	T_{TMS_HLD}	TMS hold time from TCK rise	20	-	-	ns
J9	T_{TDI_SU}	TDI setup time to TCK rise	25	-	-	ns
J10	T_{TDI_HLD}	TDI hold time from TCK rise	25	-	-	ns

Table 24-3. JTAG Characteristics *(continued)*

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
J11	T_{TDO_ZDV}	TCK fall to Data Valid from High-Z, 2-mA drive	-	23	35	ns
		TCK fall to Data Valid from High-Z, 4-mA drive		15	26	ns
		TCK fall to Data Valid from High-Z, 8-mA drive		14	25	ns
		TCK fall to Data Valid from High-Z, 8-mA drive with slew rate control		18	29	ns
J12	T_{TDO_DV}	TCK fall to Data Valid from Data Valid, 2-mA drive	-	21	35	ns
		TCK fall to Data Valid from Data Valid, 4-mA drive		14	25	ns
		TCK fall to Data Valid from Data Valid, 8-mA drive		13	24	ns
		TCK fall to Data Valid from Data Valid, 8-mA drive with slew rate control		18	28	ns
J13	T_{TDO_DVZ}	TCK fall to High-Z from Data Valid, 2-mA drive	-	9	11	ns
		TCK fall to High-Z from Data Valid, 4-mA drive		7	9	ns
		TCK fall to High-Z from Data Valid, 8-mA drive		6	8	ns
		TCK fall to High-Z from Data Valid, 8-mA drive with slew rate control		7	9	ns

a. A ratio of at least 8:1 must be kept between the system clock and TCK.

Figure 24-2. JTAG Test Clock Input Timing

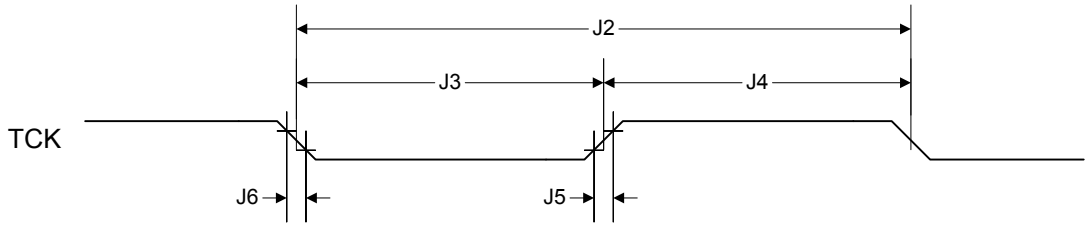
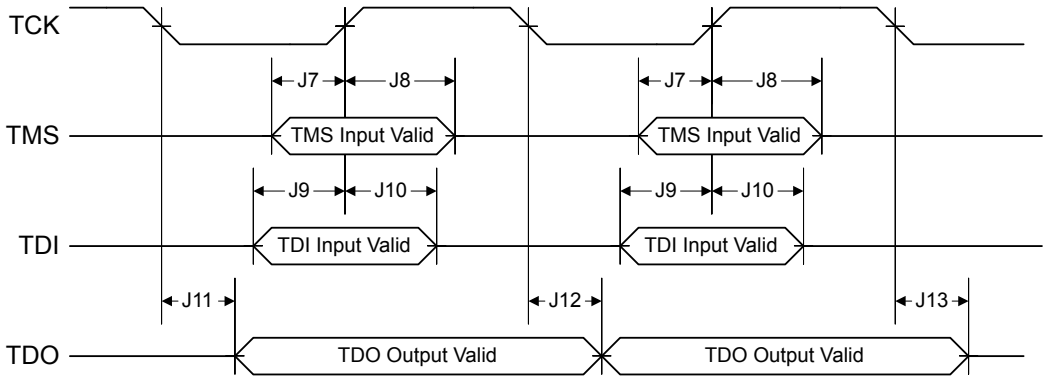


Figure 24-3. JTAG Test Access Port (TAP) Timing



24.5 Power and Brown-Out

Table 24-4. Power Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
P1	V_{TH}	Power-On Reset threshold	-	2	-	V
P2	V_{BTH}	Brown-Out Reset threshold	2.85	2.9	2.95	V
P3	T_{POR}	Power-On Reset timeout	6	-	18	ms
P4	T_{BOR}	Brown-Out timeout	-	500	-	µs
P5	T_{IRPOR}	Internal reset timeout after POR	-	-	2	ms
P6	T_{IRBOR}	Internal reset timeout after BOR	-	-	2	ms
P7	$T_{VDDRISE}$	Supply voltage (V_{DD}) rise time (0V-3.0V)	-	-	10	ms
P8	T_{VDD2_3}	Supply voltage (V_{DD}) rise time (2.0V-3.0V)	-	-	6	ms

Figure 24-4. Power-On Reset Timing

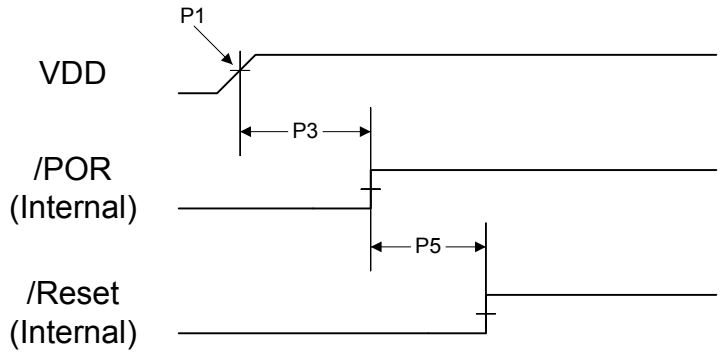


Figure 24-5. Brown-Out Reset Timing

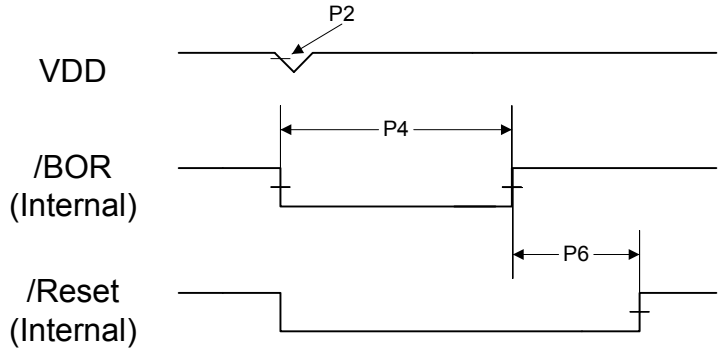
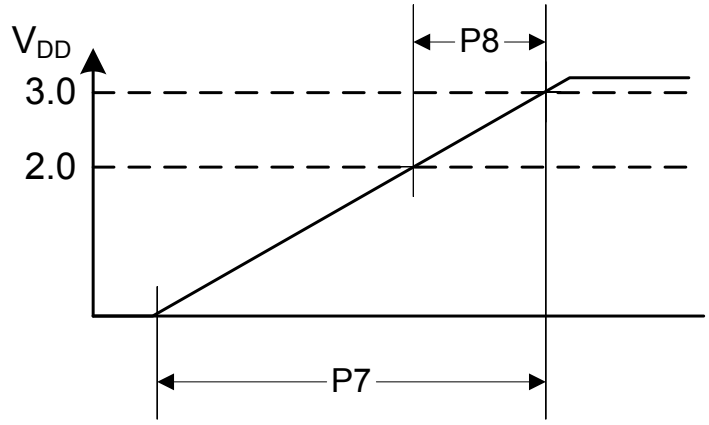


Figure 24-6. Power-On Reset and Voltage Parameters



24.6 Reset

Table 24-5. Reset Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
R1	T_{IRHWR}	Internal reset timeout after hardware reset ($\overline{RST}$ pin)	-	-	2	ms
R2	T_{IRSWR}	Internal reset timeout after software-initiated system reset	-	-	2	ms
R3	T_{IRWDR}	Internal reset timeout after watchdog reset	-	-	2	ms
R4	T_{IRMFR}	Internal reset timeout after MOSC failure reset	-	-	2	ms
R5	T_{MIN}	Minimum $\overline{RST}$ pulse width[a]	2	-	-	µs

a. This specification must be met in order to guarantee proper reset operation.

Figure 24-7. External Reset Timing ($\overline{RST}$)

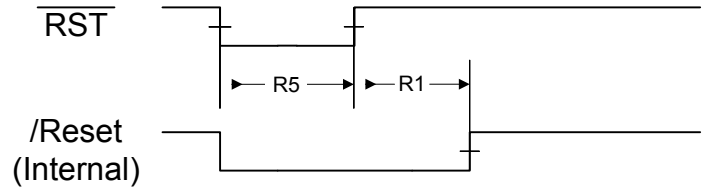


Figure 24-8. Software Reset Timing

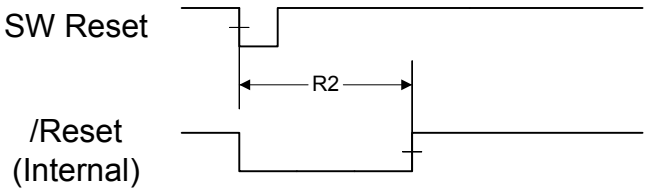


Figure 24-9. Watchdog Reset Timing

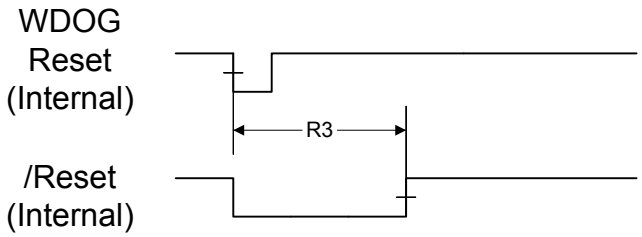
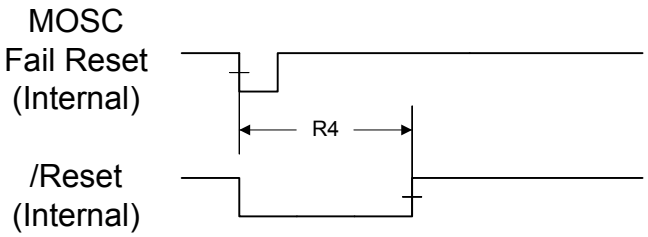


Figure 24-10. MOSC Failure Reset Timing



24.7 On-Chip Low Drop-Out (LDO) Regulator

Table 24-6. LDO Regulator Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
C_{LDO}	External filter capacitor size for internal power supply[a]	1.0	-	3.0	µF
V_{LDO}	LDO output voltage	1.235	1.3	1.365	V

a. The capacitor should be connected as close as possible to pin 86.

24.8 Clocks

The following sections provide specifications on the various clock sources and mode.

24.8.1 PLL Specifications

The following tables provide specifications for using the PLL.

Table 24-7. Phase Locked Loop (PLL) Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
F_{REF_XTAL}	Crystal reference[a]	3.579545	-	16.384	MHz
F_{REF_EXT}	External clock reference[a]	3.579545	-	16.384	MHz
F_{PLL}	PLL frequency[b]	-	400	-	MHz
T_{READY}	PLL lock time	0.562[c]	-	1.38[d]	ms

a. The exact value is determined by the crystal value programmed into the XTAL field of the **Run-Mode Clock Configuration (RCC)** register.

b. PLL frequency is automatically calculated by the hardware based on the XTAL field of the **RCC** register.

c. Using a 16.384-MHz crystal

d. Using 3.5795-MHz crystal

Table 24-8 on page 1198 shows the actual frequency of the PLL based on the crystal frequency used (defined by the XTAL field in the **RCC** register).

Table 24-8. Actual PLL Frequency

XTAL	Crystal Frequency (MHz)	PLL Frequency (MHz)	Error
0x04	3.5795	400.904	0.0023%
0x05	3.6864	398.1312	0.0047%
0x06	4.0	400	-
0x07	4.096	401.408	0.0035%
0x08	4.9152	398.1312	0.0047%
0x09	5.0	400	-
0x0A	5.12	399.36	0.0016%
0x0B	6.0	400	-
0x0C	6.144	399.36	0.0016%
0x0D	7.3728	398.1312	0.0047%
0x0E	8.0	400	-
0x0F	8.192	398.6773333	0.0033%
0x10	10.0	400	-
0x11	12.0	400	-
0x12	12.288	401.408	0.0035%
0x13	13.56	397.76	0.0056%
0x14	14.318	400.90904	0.0023%
0x15	16.0	400	-
0x16	16.384	404.1386667	0.010%

24.8.2 PIOSC Specifications

Table 24-9. PIOSC Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
$F_{PIOSC25}$	Internal 16-MHz precision oscillator frequency variance, factory calibrated at 25 °C	-	±0.25%	±1%	-
F_{PIOSCT}	Internal 16-MHz precision oscillator frequency variance, factory calibrated at 25 °C, across specified temperature range	-	-	±3%	-
$F_{PIOSCUCAL}$	Internal 16-MHz precision oscillator frequency variance, user calibrated at a chosen temperature	-	±0.25%	±1%	-

24.8.3 Internal 30-kHz Oscillator Specifications

Table 24-10. 30-kHz Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
$F_{IOSC30KHZ}$	Internal 30-KHz oscillator frequency	15	30	45	KHz

24.8.4 Main Oscillator Specifications

Table 24-11. Main Oscillator Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
F_{MOSC}	Main oscillator frequency	1	-	16.384	MHz
T_{MOSC_PER}	Main oscillator period	61	-	1000	ns
T_{MOSC_SETTLE}	Main oscillator settling time[a]	17.5	-	20	ms
$F_{REF\_XTAL\_BYPASS}$	Crystal reference using the main oscillator (PLL in BYPASS mode)[b]	1	-	16.384	MHz
$F_{REF\_EXT\_BYPASS}$	External clock reference (PLL in BYPASS mode)[b]	0	-	50	MHz
DC_{MOSC_EXT}	External clock reference duty cycle	45	-	55	%

a. This parameter is highly sensitive to PCB layout and trace lengths, which may make this parameter time longer. Care must be taken in PCB design to minimize trace lengths and RLC (resistance, inductance, capacitance).

b. If the ADC is used, the crystal reference must be 16 MHz ± .03% when the PLL is bypassed.

Table 24-12. Supported MOSC Crystal Frequencies[a]

Crystal Frequency (MHz) Not Using the PLL	Crystal Frequency (MHz) Using the PLL
1.000 MHz	reserved
1.8432 MHz	reserved
2.000 MHz	reserved
2.4576 MHz	reserved
3.579545 MHz	
3.6864 MHz	
4 MHz (USB)	
4.096 MHz	
4.9152 MHz	
5 MHz (USB)	
5.12 MHz	
6 MHz (reset value)(USB)	
6.144 MHz	
7.3728 MHz	
8 MHz (USB)	
8.192 MHz	
10.0 MHz (USB)	
12.0 MHz (USB)	
12.288 MHz	
13.56 MHz	
14.31818 MHz	
16.0 MHz (USB)	
16.384 MHz	

a. Frequencies that may be used with the USB interface are indicated in the table.

24.8.5 System Clock Specification with ADC Operation

Table 24-13. System Clock Characteristics with ADC Operation

Parameter	Parameter Name	Min	Nom	Max	Unit
F_{sysadc}	System clock frequency when the ADC module is operating (when PLL is bypassed).[a]	15.9952	16	16.0048	MHz

a. Clock frequency (plus jitter) must be stable inside specified range. ADC can be clocked from the PLL or directly from an external clock source, as long as frequency absolute precision is inside specified range.

24.8.6 System Clock Specification with USB Operation

Table 24-14. System Clock Characteristics with USB Operation

Parameter	Parameter Name	Min	Nom	Max	Unit
F_{sysusb}	System clock frequency when the USB module is operating (note that MOSC must be the clock source, either with or without using the PLL)	30	-	-	MHz

24.9 Sleep Modes

Table 24-15. Sleep Modes AC Characteristics[a]

Parameter No	Parameter	Parameter Name	Min	Nom	Max	Unit
D1	T_{WAKE_S}	Time to wake from interrupt in sleep mode, not using the PLL[b]	-	-	2	system clocks
	T_{WAKE_DS}	Time to wake from interrupt deep-sleep mode, not using the PLL[b]	-	-	7	system clocks
D2	$T_{WAKE\_PLL\_S}$	Time to wake from interrupt in sleep or deep-sleep mode when using the PLL[b]	-	-	T_{READY}	ms
D3	T_{ENTER_DS}	Time to enter deep-sleep mode from sleep request	-	0	35[c]	ms

a. Values in this table assume the IOSC is the clock source during sleep or deep-sleep mode.

b. Specified from registering the interrupt to first instruction.

c. Nominal specification occurs 99.9995% of the time.

24.10 Flash Memory

Table 24-16. Flash Memory Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
PE_{CYC}	Number of guaranteed program/erase cycles before failure[a]	15,000	-	-	cycles
T_{RET}	Data retention, -40˚C to +85˚C	10	-	-	years
T_{PROG}	Word program time	-	-	1	ms
T_{BPROG}	Buffer program time	-	-	1	ms
T_{ERASE}	Page erase time	-	-	12	ms
T_{ME}	Mass erase time	-	-	16	ms

a. A program/erase cycle is defined as switching the bits from 1-> 0 -> 1.

24.11 Input/Output Characteristics

Note: All GPIOs are 5-V tolerant, except PB0 and PB1. See “Signal Description” on page 395 for more information on GPIO configuration.

Table 24-17. GPIO Module Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
R_{GPIOPU}	GPIO internal pull-up resistor	100	-	300	kΩ
R_{GPIOPD}	GPIO internal pull-down resistor	200	-	500	kΩ
I_{LKG}	GPIO input leakage current[a]	-	-	2	µA
T_{GPIOR}	GPIO Rise Time, 2-mA drive[b]	-	14	20	ns
	GPIO Rise Time, 4-mA drive[b]		7	10	ns
	GPIO Rise Time, 8-mA drive[b]		4	5	ns
	GPIO Rise Time, 8-mA drive with slew rate control[b]		6	8	ns
T_{GPIOF}	GPIO Fall Time, 2-mA drive[c]	-	14	21	ns
	GPIO Fall Time, 4-mA drive[c]		7	11	ns
	GPIO Fall Time, 8-mA drive[c]		4	6	ns
	GPIO Fall Time, 8-mA drive with slew rate control[c]		6	8	ns

a. The leakage current is measured with GND or VDD applied to the corresponding pin(s). The leakage of digital port pins is measured individually. The port pin is configured as an input and the pullup/pulldown resistor is disabled.

b. Time measured from 20% to 80% of V_{DD}.

c. Time measured from 80% to 20% of V_{DD}.

24.12 External Peripheral Interface (EPI)

When the EPI module is in SDRAM mode, the drive strength must be configured to 8 mA. Table 24-18 on page 1201 shows the rise and fall times in SDRAM mode with 16 pF load conditions. When the EPI module is in Host-Bus or General-Purpose mode, the values in “Input/Output Characteristics” on page 1201 should be used.

Table 24-18. EPI SDRAM Characteristics

Parameter	Parameter Name	Condition	Min	Nom	Max	Unit
T_{SDRAMR}	EPI Rise Time (from 20% to 80% of V_{DD})	8-mA drive, C_L = 16 pF	-	2	3	ns
T_{SDRAMF}	EPI Fall Time (from 80% to 20% of V_{DD})	8-mA drive, C_L = 16 pF	-	2	3	ns

Table 24-19. EPI SDRAM Interface Characteristics[a]

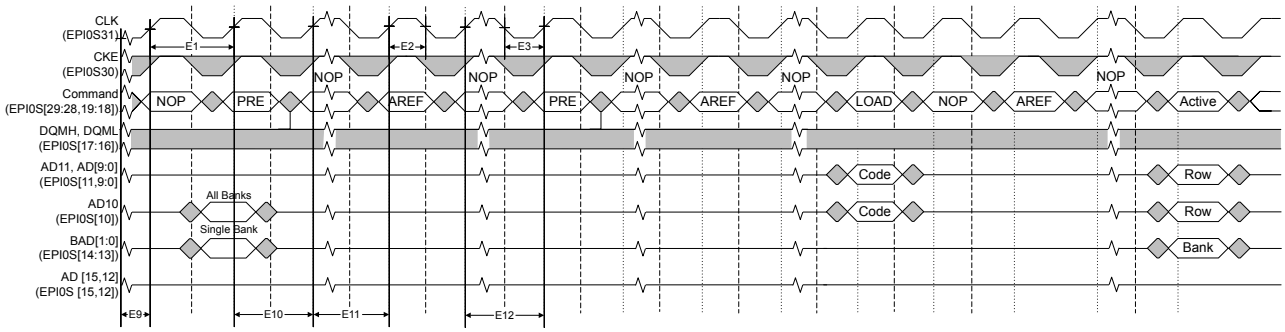
Parameter No	Parameter	Parameter Name	Min	Nom	Max	Unit
E1	T_{CK}	SDRAM Clock period	20	-	-	ns
E2	T_{CH}	SDRAM Clock high time	10	-	-	ns
E3	T_{CL}	SDRAM Clock low time	10	-	-	ns
E4	T_{COV}	CLK to output valid	-5	-	5	ns
E5	T_{COI}	CLK to output invalid	-5	-	5	ns
E6	T_{COT}	CLK to output tristate	-5	-	5	ns
E7	T_S	Input set up to CLK	10	-	-	ns

Table 24-19. EPI SDRAM Interface Characteristics *(continued)*

Parameter No	Parameter	Parameter Name	Min	Nom	Max	Unit
E8	T_H	CLK to input hold	0	-	-	ns
E9	T_{PU}	Power-up time	100	-	-	µs
E10	T_{RP}	Precharge all banks	20	-	-	ns
E11	T_{RFC}	Auto refresh	66	-	-	ns
E12	T_{MRD}	Program mode register	40	-	-	ns

a. The EPI SDRAM interface must use 8-mA drive.

Figure 24-11. SDRAM Initialization and Load Mode Register Timing



Notes:
1. If CS is high at clock high time, all applied commands are NOP.
2. The **Mode** register may be loaded prior to the auto refresh cycles if desired.
3. JEDEC and PC100 specify three clocks.
4. Outputs are guaranteed High-Z after command is issued.

Figure 24-12. SDRAM Read Timing

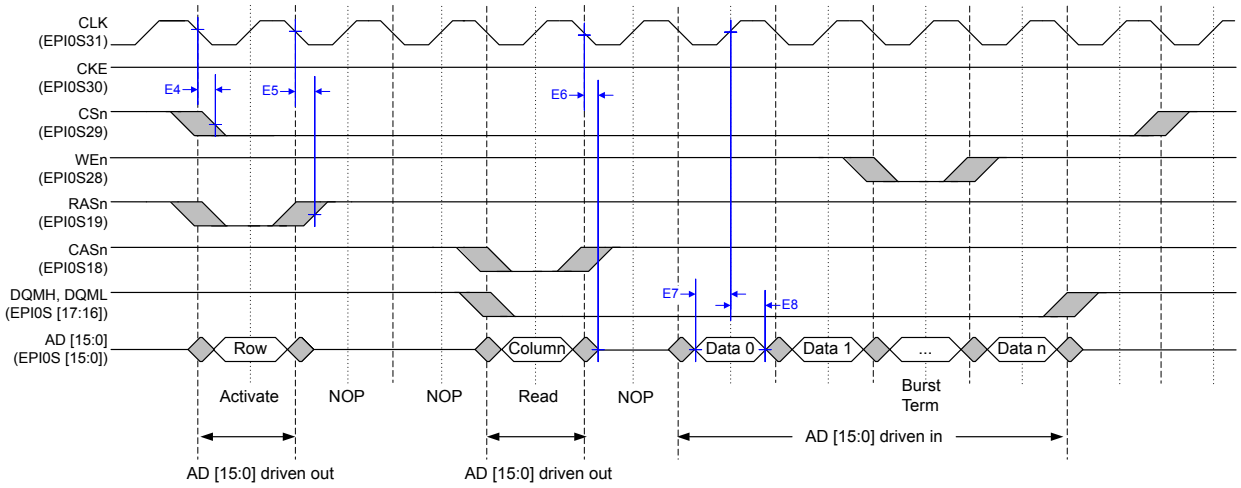


Figure 24-13. SDRAM Write Timing

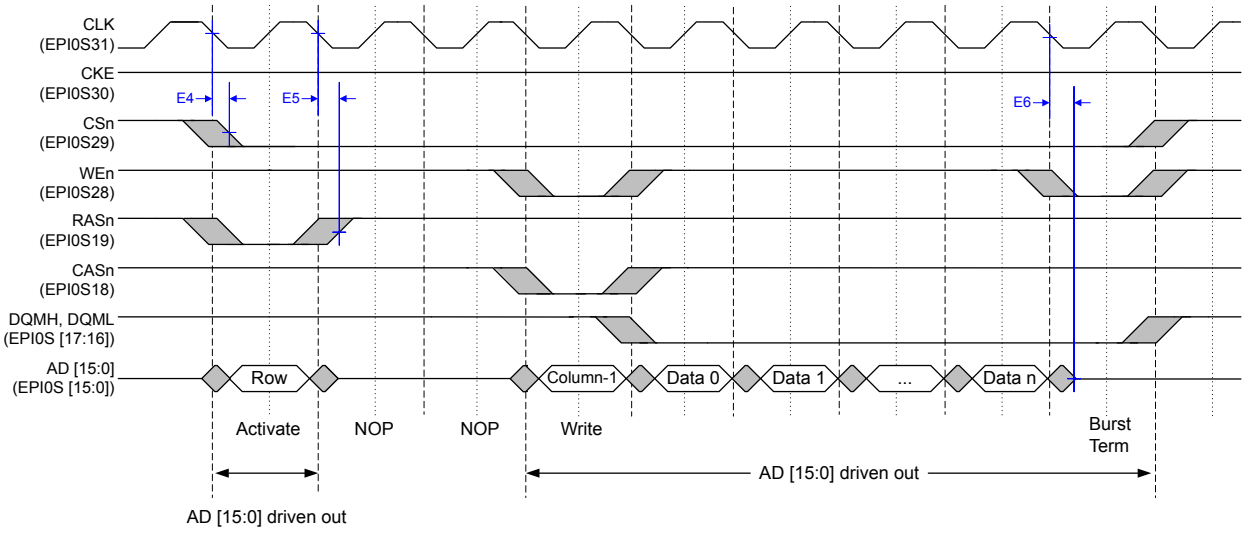
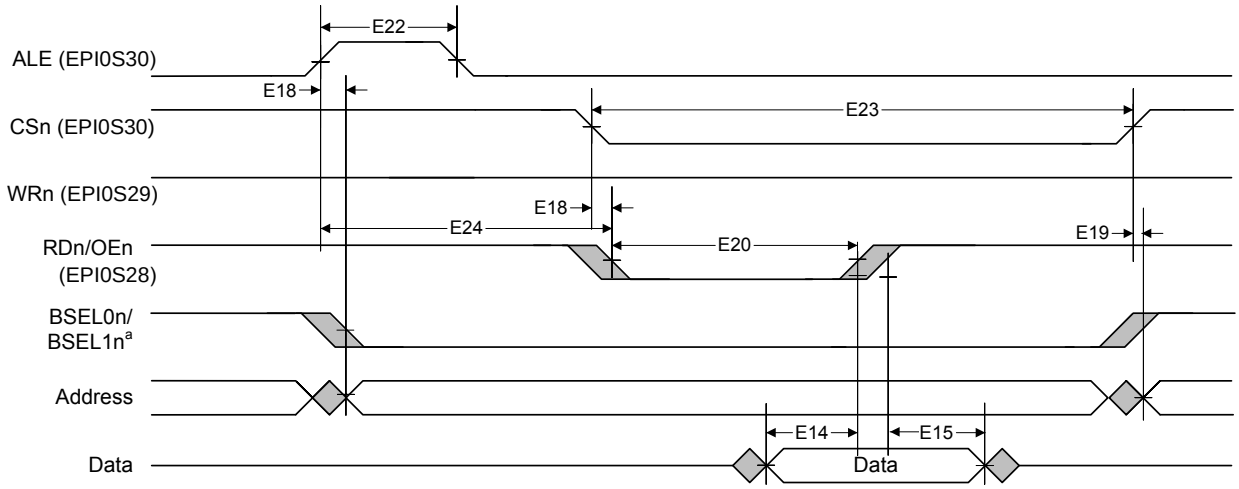


Table 24-20. EPI Host-Bus 8 and Host-Bus 16 Interface Characteristics

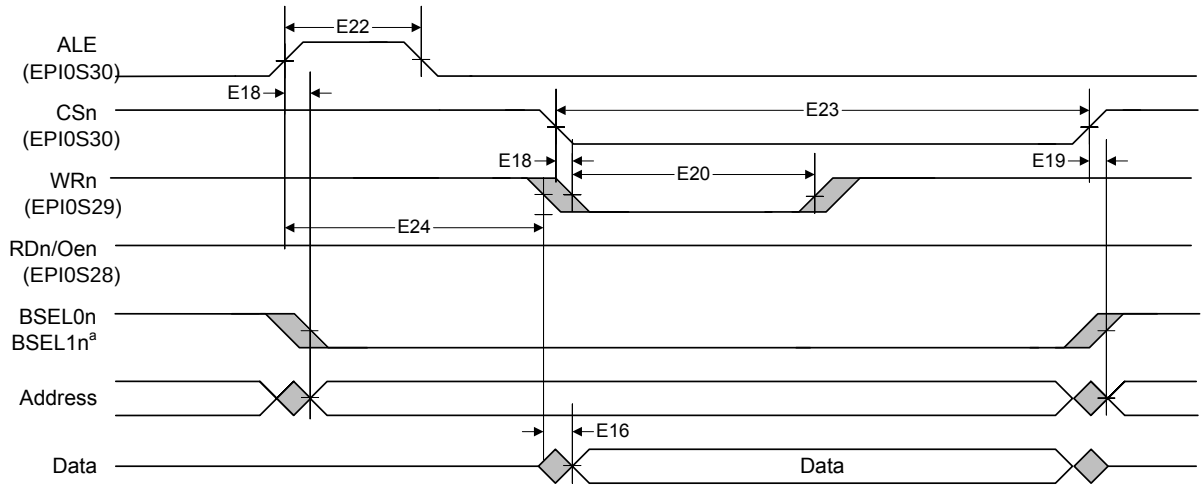
Parameter No	Parameter	Parameter Name	Min	Nom	Max	Unit
E14	T_{ISU}	Read data set up time	10	-	-	ns
E15	T_{IH}	Read data hold time	0	-	-	ns
E16	T_{DV}	WEn to write data valid	-	-	5	ns
E17	T_{DI}	Data hold from WEn invalid	2	-	-	EPI Clocks
E18	T_{OV}	CSn to output valid	-5	-	5	ns
E19	T_{OINV}	CSn to output invalid	-5	-	5	ns
E20	T_{STLOW}	WEn / RDn strobe width low	2	-	-	EPI Clocks
E21	T_{FIFO}	FEMPTY and FFULL setup time to clock edge	2	-	-	System Clocks
E22	$T_{ALEHIGH}$	ALE width high	-	1	-	EPI Clocks
E23	T_{CSLOW}	CSn width low	4	-	-	EPI Clocks
E24	T_{ALEST}	ALE rising to WEn / RDn strobe falling	2	-	-	EPI Clocks
E25	T_{ALEADD}	ALE falling to ADn tristate	1	-	-	EPI Clocks

Figure 24-14. Host-Bus 8/16 Mode Read Timing



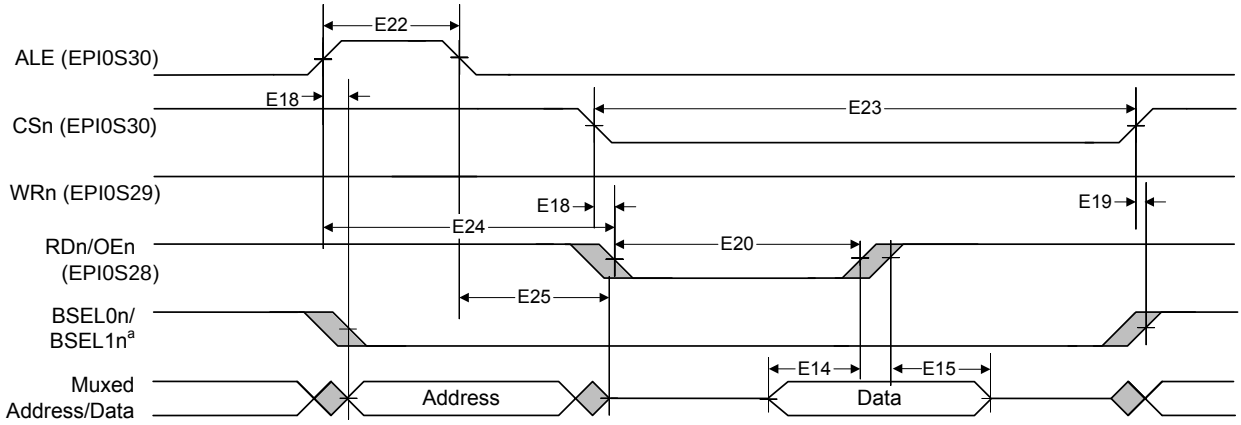
[a] BSEL0n and BSEL1n are available in Host-Bus 16 mode only.

Figure 24-15. Host-Bus 8/16 Mode Write Timing



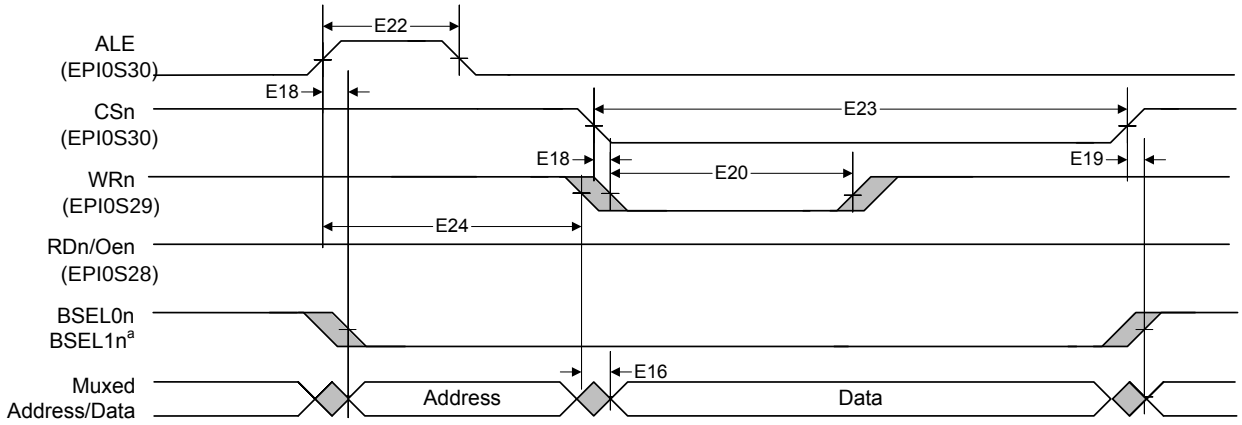
[a] BSEL0n and BSEL1n are available in Host-Bus 16 mode only.

Figure 24-16. Host-Bus 8/16 Mode Muxed Read Timing



[a] BSEL0n and BSEL1n are available in Host-Bus 16 mode only.

Figure 24-17. Host-Bus 8/16 Mode Muxed Write Timing

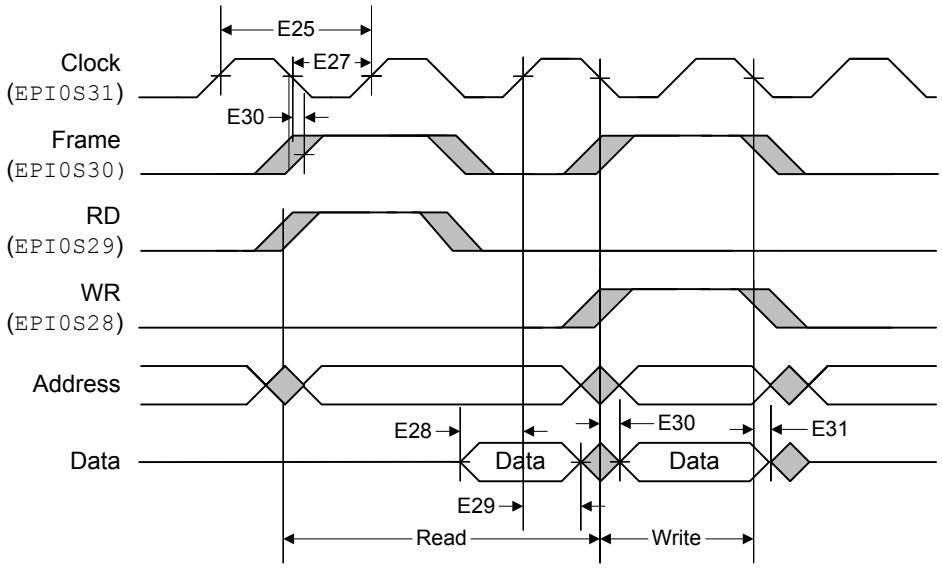


[a] BSEL0n and BSEL1n are available in Host-Bus 16 mode only.

Table 24-21. EPI General-Purpose Interface Characteristics

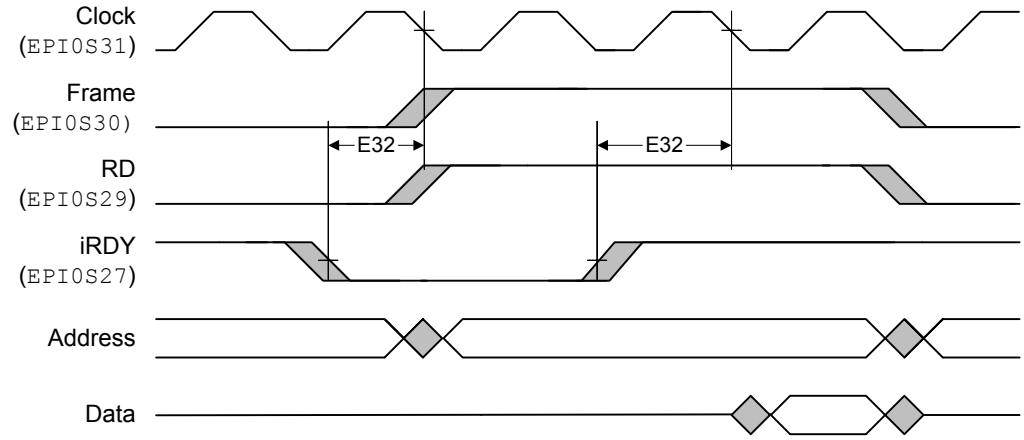
Parameter No	Parameter	Parameter Name	Min	Nom	Max	Unit
E25	T_{CK}	General-Purpose Clock period	20	-	-	ns
E26	T_{CH}	General-Purpose Clock high time	10	-	-	ns
E27	T_{CL}	General-Purpose Clock low time	10	-	-	ns
E28	T_{ISU}	Input signal set up time to rising clock edge	10	-	-	ns
E29	T_{IH}	Input signal hold time from rising clock edge	0	-	-	ns
E30	T_{DV}	Falling clock edge to output valid	-5	-	5	ns
E31	T_{DI}	Falling clock edge to output invalid	-5	-	5	ns
E32	T_{RDYSU}	iRDY assertion or deassertion set up time to falling clock edge	10	-	-	ns

Figure 24-18. General-Purpose Mode Read and Write Timing



The above figure illustrates accesses where the FRM50 bit is clear, the FRMCNT field is 0x0, the RD2CYC bit is clear, and the WR2CYC bit is clear.

Figure 24-19. General-Purpose Mode iRDY Timing



24.13 Analog-to-Digital Converter (ADC)

Table 24-22. ADC Characteristics[a]

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{ADCIN}	Maximum single-ended, full-scale analog input voltage, using internal reference	-	-	3.0	V
	Maximum single-ended, full-scale analog input voltage, using external reference	-	-	V_{REFA}	V
	Minimum single-ended, full-scale analog input voltage	0.0	-	-	V
	Maximum differential, full-scale analog input voltage, using internal reference	-	-	1.5	V
	Maximum differential, full-scale analog input voltage, using external reference	-	-	$V_{REFA}/2$	V
	Minimum differential, full-scale analog input voltage	0.0	-	-	V
N	Resolution		12		bits
F_{ADC}	ADC internal clock frequency[b]	15.9952	16	16.0048	MHz
$T_{ADCCONV}$	Conversion time[c]		1		µs
$F_{ADCCONV}$	Conversion rate[c]		1000		k samples/s
$T_{ADCSAMP}$	Sample time	125	-	-	ns
T_{LT}	Latency from trigger to start of conversion	-	2	-	system clocks
I_L	ADC input leakage	-	-	2.0	µA
R_{ADC}	ADC equivalent resistance	-	-	10	kΩ
C_{ADC}	ADC equivalent capacitance	0.9	1.0	1.1	pF
E_L	Integral nonlinearity (INL) error, 12-bit mode	-	-	±8	LSB
	Integral nonlinearity (INL) error, 10-bit mode	-	-	±2	LSB
E_D	Differential nonlinearity (DNL) error, 12-bit mode	-	-	±4	LSB
	Differential nonlinearity (DNL) error, 10-bit mode	-	-	±2	LSB
E_O	Offset error, 12-bit mode	-	-	±40	LSB
	Offset error, 10-bit mode	-	-	±10	LSB
E_G	Full-scale gain error, 12-bit mode	-	-	±100	LSB
	Full-scale gain error, 10-bit mode	-	-	±25	LSB
E_{TS}	Temperature sensor accuracy[d]	-	-	±5	°C

a. The ADC reference voltage is 3.0 V. This reference voltage is internally generated from the 3.3 VDDA supply by a band gap circuit.

b. The ADC must be clocked from the PLL or directly from an external clock source to operate properly.

c. The conversion time and rate scale from the specified number if the ADC internal clock frequency is any value other than 16 MHz.

d. Note that this parameter does not include ADC error.

Figure 24-20. ADC Input Equivalency Diagram

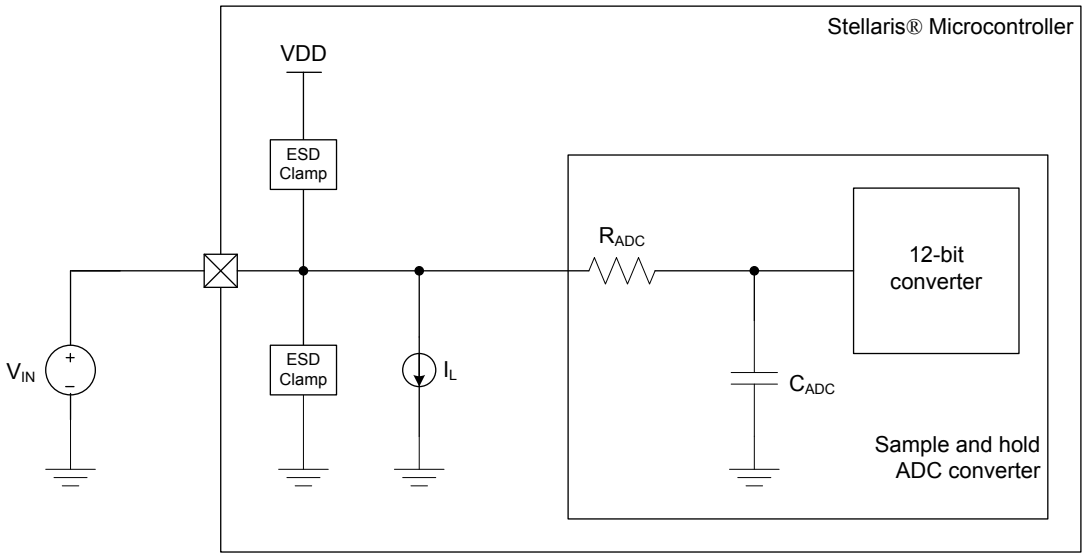


Table 24-23. ADC Module External Reference Characteristics[a]

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{REFA}	External voltage reference for ADC, when the VREF field in the **ADCCTL** register is 0x1[b]	2.97	-	3.03	V
	External voltage reference for ADC, when the VREF field in the **ADCCTL** register is 0x3[c]	0.99	-	1.01	V
I_L	External voltage reference leakage current	-	-	2.0	µA

a. Care must be taken to supply a reference voltage of acceptable quality.

b. Ground is always used as the reference level for the minimum conversion value.

c. Ground is always used as the reference level for the minimum conversion value.

Table 24-24. ADC Module Internal Reference Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{REFI}	Internal voltage reference for ADC	-	3.0	-	V

24.14 Synchronous Serial Interface (SSI)

Table 24-25. SSI Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
S1	T_{CLK_PER}	SSIClk cycle time[a]	40	-	-	ns
S2	T_{CLK_HIGH}	SSIClk high time	-	0.5	-	t clk_per
S3	T_{CLK_LOW}	SSIClk low time	-	0.5	-	t clk_per
S4	T_{CLKRF}	SSIClk rise/fall time[b]	-	4	6	ns
S5	T_{DMD}	Data from master valid delay time	0	-	1	system clocks
S6	T_{DMS}	Data from master setup time	1	-	-	system clocks
S7	T_{DMH}	Data from master hold time	2	-	-	system clocks
S8	T_{DSS}	Data from slave setup time	1	-	-	system clocks

Table 24-25. SSI Characteristics *(continued)*

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
S9	T_{DSH}	Data from slave hold time	2	-	-	system clocks

a. In master mode, the system clock must be at least twice as fast as the SSIClk; in slave mode, the system clock must be at least 12 times faster than the SSIClk.

b. Note that the delays shown are using 8-mA drive strength.

Figure 24-21. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement

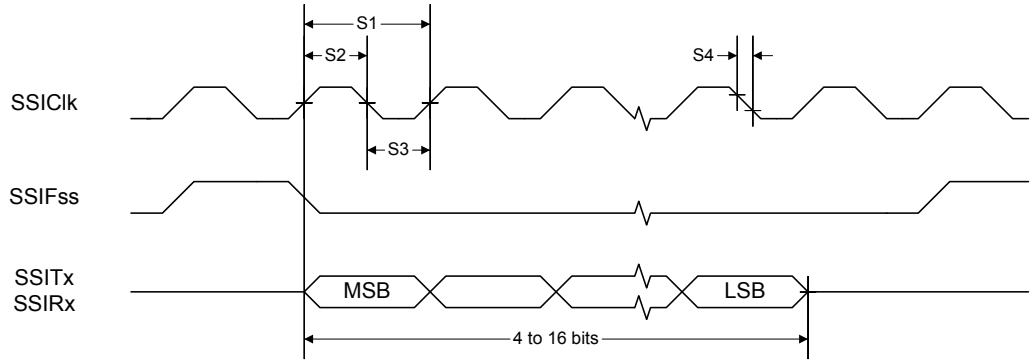


Figure 24-22. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer

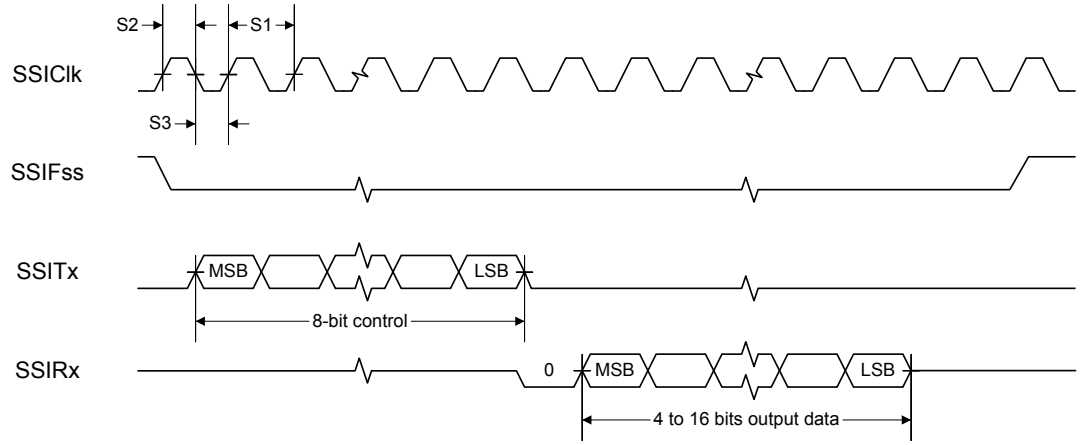
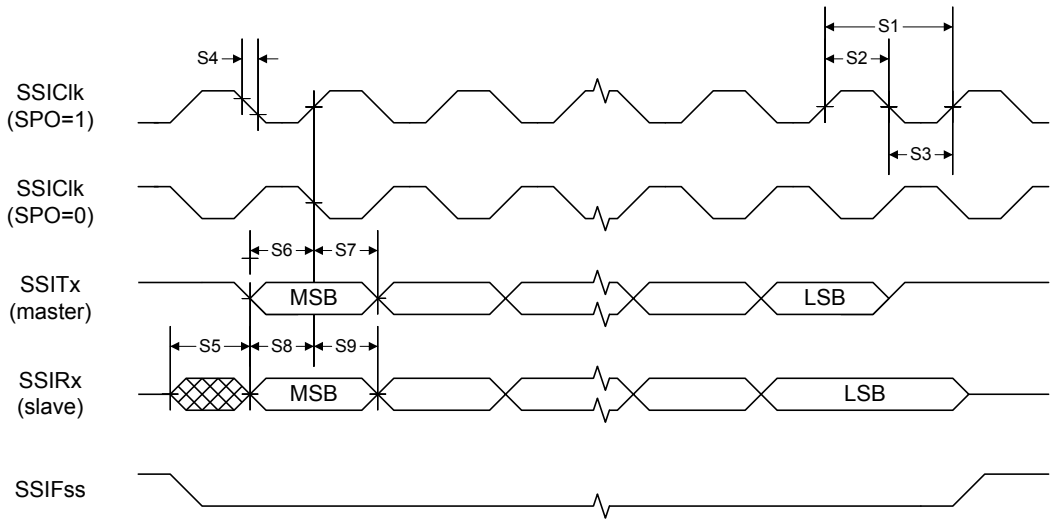


Figure 24-23. SSI Timing for SPI Frame Format (FRF=00), with SPH=1



24.15 Inter-Integrated Circuit (I^2C) Interface

Table 24-26. I^2C Characteristics

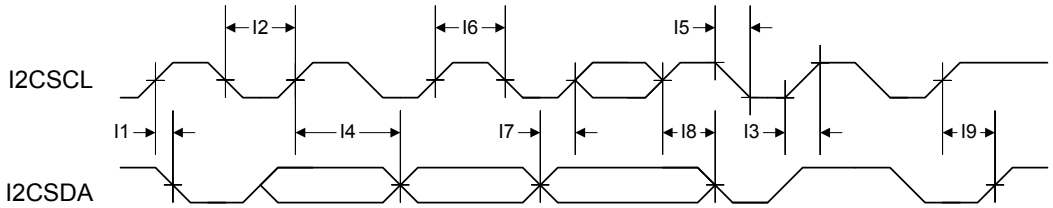
Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
I1[a]	T_{SCH}	Start condition hold time	36	-	-	system clocks
I2[a]	T_{LP}	Clock Low period	36	-	-	system clocks
I3[b]	T_{SRT}	I2CSCL/I2CSDA rise time (V_{IL} =0.5 V to V_{IH} =2.4 V)	-	-	(see note b)	ns
I4[a]	T_{DH}	Data hold time	2	-	-	system clocks
I5[c]	T_{SFT}	I2CSCL/I2CSDA fall time (V_{IH} =2.4 V to V_{IL} =0.5 V)	-	9	10	ns
I6[a]	T_{HT}	Clock High time	24	-	-	system clocks
I7[a]	T_{DS}	Data setup time	18	-	-	system clocks
I8[a]	T_{SCSR}	Start condition setup time (for repeated start condition only)	36	-	-	system clocks
I9[a]	T_{SCS}	Stop condition setup time	24	-	-	system clocks

a. Values depend on the value programmed into the TPR bit in the **I^2C Master Timer Period (I2CMTPR)** register; a TPR programmed for the maximum I2CSCL frequency (TPR=0x2) results in a minimum output timing as shown in the table above. The I^2C interface is designed to scale the actual data transition time to move it to the middle of the I2CSCL Low period. The actual position is affected by the value programmed into the TPR; however, the numbers given in the above values are minimum values.

b. Because I2CSCL and I2CSDA are open-drain-type outputs, which the controller can only actively drive Low, the time I2CSCL or I2CSDA takes to reach a high level depends on external signal capacitance and pull-up resistor values.

c. Specified at a nominal 50 pF load.

Figure 24-24. I²C Timing



24.16 Inter-Integrated Circuit Sound (I²S) Interface

Table 24-27. I²S Master Clock (Receive and Transmit)

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
M1	T_{MCLK_PER}	Cycle time	20.3	-	-	ns
M2	T_{MCLKRF}	Rise/fall time	See "Input/Output Characteristics" on page 1201.			ns
M3	T_{MCLK_HIGH}	High time	10	-	-	ns
M4	T_{MCLK_LOW}	Low time	10	-	-	ns
M5	T_{MDC}	Duty cycle	48	-	52	%
M6	$T_{MJITTER}$	Jitter	-	-	2.5	ns

Table 24-28. I²S Slave Clock (Receive and Transmit)

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
M7	T_{SCLK_PER}	Cycle time	80	-	-	ns
M8	T_{SCLK_HIGH}	High time	40	-	-	ns
M9	T_{SCLK_LOW}	Low time	40	-	-	ns
M10	T_{SDC}	Duty cycle	-	50	-	%

Table 24-29. I²S Master Mode

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
M11	T_{MSWS}	SCK fall to WS valid	-	-	10	ns
M12	T_{MSD}	SCK fall to TXSD valid	-	-	10	ns
M13	T_{MSDS}	RXSD setup time to SCK rise	10	-	-	ns
M14	T_{MSDH}	RXSD hold time from SCK rise	10	-	-	ns

Figure 24-25. I²S Master Mode Transmit Timing

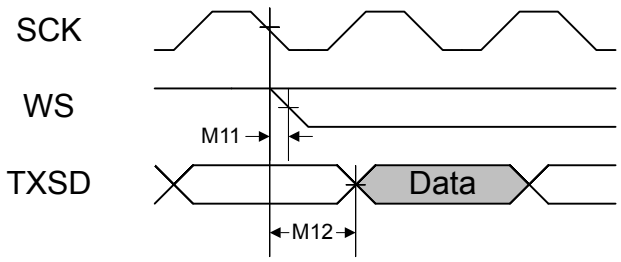


Figure 24-26. I²S Master Mode Receive Timing

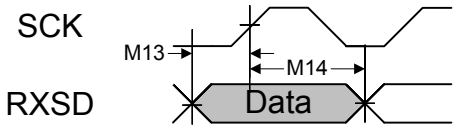


Table 24-30. I²S Slave Mode

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
M15	T_{SCLK_PER}	Cycle time	80	-	-	ns
M16	T_{SCLK_HIGH}	High time	40	-	-	ns
M17	T_{SCLK_LOW}	Low time	40	-	-	ns
M18	T_{SDC}	Duty cycle	-	50	-	%
M19	T_{SSETUP}	WS setup time to SCK rise	-	-	25	ns
M20	T_{SHOLD}	WS hold time from SCK rise	-	-	10	ns
M21	T_{SSD}	SCK fall to TXSD valid	-	-	20	ns
M22	T_{SLSD}	Left-justified mode, WS to TXSD	-	-	20	ns
M23	T_{SSDS}	RXSD setup time to SCK rise	10	-	-	ns
M24	T_{SSDH}	RXSD hold time from SCK rise	10	-	-	ns

Figure 24-27. I²S Slave Mode Transmit Timing

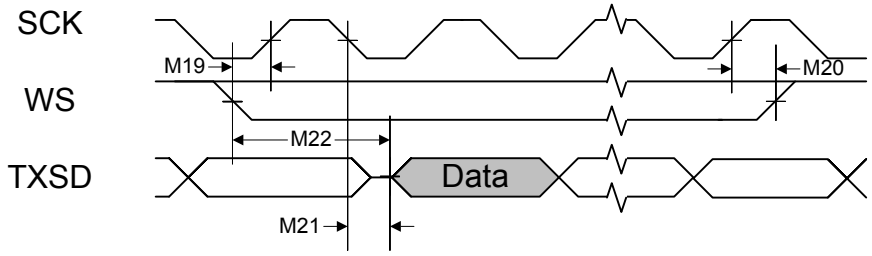
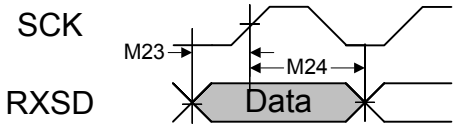


Figure 24-28. I²S Slave Mode Receive Timing



24.17 Ethernet Controller

Table 24-31. Ethernet Controller DC Characteristics

Parameter	Parameter Name	Value	Unit
R_{EBIAS}	Value of the pull-down resistor on the ERBIAS pin	12.4K ± 1 %	Ω

Table 24-32. 100BASE-TX Transmitter Characteristics[a]

Parameter Name	Min	Nom	Max	Unit
Peak output amplitude	950	-	1050	mVpk
Output amplitude symmetry	98	-	102	%
Output overshoot	-	-	5	%
Rise/Fall time	3	-	5	ns
Rise/Fall time imbalance	-	-	500	ps
Duty cycle distortion	-	-	±250	ps
Jitter	-	-	1.4	ns

a. Measured at the line side of the transformer.

Table 24-33. 100BASE-TX Transmitter Characteristics (informative)[a]

Parameter Name	Min	Nom	Max	Unit
Return loss	16	-	-	dB
Open-circuit inductance	350	-	-	µH

a. The specifications in this table are included for information only. They are mainly a function of the external transformer and termination resistors used for measurements.

Table 24-34. 100BASE-TX Receiver Characteristics

Parameter Name	Min	Nom	Max	Unit
Signal detect assertion threshold	600	700	-	mVppd
Signal detect de-assertion threshold	350	425	-	mVppd
Differential input resistance	-	3.6	-	kΩ
Jitter tolerance (pk-pk)	4	-	-	ns
Baseline wander tracking	-80	-	+80	%
Signal detect assertion time	-	-	1000	µs
Signal detect de-assertion time	-	-	4	µs

Table 24-35. 10BASE-T Transmitter Characteristics[a]

Parameter Name	Min	Nom	Max	Unit
Peak differential output signal	2.2	-	2.7	V
Harmonic content	27	-	-	dB
Link pulse width	-	100	-	ns
Start-of-idle pulse width, Last bit 0	-	300	-	ns
Start-of-idle pulse width, Last bit 1	-	350	-	ns

a. The Manchester-encoded data pulses, the link pulse and the start-of-idle pulse are tested against the templates and using the procedures found in Clause 14 of *IEEE 802.3*.

Table 24-36. 10BASE-T Transmitter Characteristics (informative)[a]

Parameter Name	Min	Nom	Max	Unit
Output return loss	15	-	-	dB
Output impedance balance	29-17log(f/10)	-	-	dB
Peak common-mode output voltage	-	-	50	mV
Common-mode rejection	-	-	100	mV

Table 24-36. 10BASE-T Transmitter Characteristics (informative) *(continued)*

Parameter Name	Min	Nom	Max	Unit
Common-mode rejection jitter	-	-	1	ns

a. The specifications in this table are included for information only. They are mainly a function of the external transformer and termination resistors used for measurements.

Table 24-37. 10BASE-T Receiver Characteristics

Parameter Name	Min	Nom	Max	Unit
Jitter tolerance (pk-pk)	30	26	-	ns
Input squelched threshold	340	440	540	mVppd
Differential input resistance	-	3.6	-	kΩ
Common-mode rejection	25	-	-	V

Table 24-38. Isolation Transformers[a]

Name	Value	Condition
Turns ratio	1 CT : 1 CT	+/- 5%
Open-circuit inductance	350 uH (min)	@ 10 mV, 10 kHz
Leakage inductance	0.40 uH (max)	@ 1 MHz (min)
Inter-winding capacitance	25 pF (max)	
DC resistance	0.9 Ohm (max)	
Insertion loss	0.4 dB (typ)	0-65 MHz
HIPOT	1500	Vrms

a. Two simple 1:1 isolation transformers are required at the line interface. Transformers with integrated common-mode chokes are recommended for exceeding FCC requirements. This table gives the recommended line transformer characteristics.

Note: The 100Base-TX amplitude specifications assume a transformer loss of 0.4 dB.

Table 24-39. Ethernet Reference Crystal

Parameter	Parameter Name	Min	Nom	Max	Unit
$F_{XTALPHYOSC}$	Ethernet PHY oscillator frequency	-	25	-	MHz
$TOL_{XTALPHYOSC}$	Ethernet PHY oscillator frequency tolerance[a]	-	±50	-	PPM
$MODE_{XTALPHYOSC}$	Ethernet PHY oscillation mode	Parallel resonance, fundamental mode			-

a. This tolerance provides a guard band for temperature stability and aging drift.

Figure 24-29. External XTLP Oscillator Characteristics

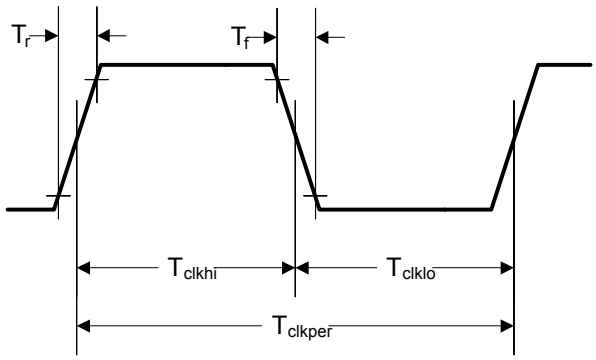


Table 24-40. External XTLP Oscillator Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
$XTLN_{ILV}$	XTLN Input Low Voltage	-	-	0.8	-
$XTLP_F$	XTLP Frequency[a]	-	25.0	-	-
T_{CLKPER}	XTLP Period[a]	-	40	-	-
$XTLP_{DC}$	XTLP Duty Cycle	40 40	-	60 60	%
T_R , T_F	Rise/Fall Time	-	-	4.0	ns
T_{JITTER}	Absolute Jitter	-	-	0.1	ns

a. IEEE 802.3 frequency tolerance ±50 ppm.

24.18 Universal Serial Bus (USB) Controller

The Stellaris® USB controller electrical specifications are compliant with the *Universal Serial Bus Specification Rev. 2.0* (full-speed and low-speed support) and the *On-The-Go Supplement to the USB 2.0 Specification Rev. 1.0*. Some components of the USB system are integrated within the LM3S9U81 microcontroller and specific to the Stellaris microcontroller design. An external component resistor is needed as specified in Table 24-41.

Table 24-41. USB Controller Characteristics

Parameter	Parameter Name	Value	Unit
R_{UBIAS}	Value of the pull-down resistor on the USB0RBIAS pin	9.1K ± 1 %	Ω

24.19 Analog Comparator

Table 24-42. Analog Comparator Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{INP},V_{INN}	Input voltage range	GND	-	V_{DD}	V
V_{CM}	Input common mode voltage range	GND	-	V_{DD}-1.5	V
V_{OS}	Input offset voltage	-	±10	±25	mV

Table 24-42. Analog Comparator Characteristics *(continued)*

Parameter	Parameter Name	Min	Nom	Max	Unit
C_{MRR}	Common mode rejection ratio	50	-	-	dB
T_{RT}	Response time	-	-	1.0	µs
T_{MC}	Comparator mode change to Output Valid	-	-	10	µs

Table 24-43. Analog Comparator Voltage Reference Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
R_{HR}	Resolution in high range	-	$V_{DDA}/31$	-	V
R_{LR}	Resolution in low range	-	$V_{DDA}/23$	-	V
A_{HR}	Absolute accuracy high range	-	-	$\pm R_{HR}/2$	V
A_{LR}	Absolute accuracy low range	-	-	$\pm R_{LR}/4$	V

24.20 Current Consumption

This section provides information on typical and maximum power consumption under various conditions. Unless otherwise indicated, current consumption numbers include use of the on-chip LDO regulator and therefore include I_{DDC}.

24.20.1 Nominal Power Consumption

The following table provides nominal figures for current consumption.

Table 24-44. Nominal Power Consumption

Parameter	Parameter Name	Conditions	Nom	Unit
I_{DD_RUN}	Run mode 1 (Flash loop)	V_{DD} = 3.3 V Code= while(1){} executed out of Flash Peripherals = All ON System Clock = 80 MHz (with PLL) Temp = 25°C	101[a] 159[b]	mA
I_{DD_SLEEP}	Sleep mode	V_{DD} = 3.3 V Peripherals = All clock gated System Clock = 80 MHz (with PLL) Temp = 25°C	20	mA
$I_{DD\_DEEPSLEEP}$	Deep-sleep mode	Peripherals = All OFF System Clock = IOSC30KHZ/64 Temp = 25°C	550	µA

a. Ethernet MAC and PHY powered down by software.

b. Auto-negotiate enabled. If an Ethernet cable is attached to the connector, the consumption increases by 7-10 mA.

24.20.2 Maximum Current Consumption

The current measurements specified in the table that follows are maximum values under the following conditions:

- V_{DD} = 3.6 V
- V_{DDC} = 1.3 V

- V_{DDA} = 3.6 V
- Temperature = 25°C
- Clock source (MOSC) = 16.348-MHz crystal oscillator

Table 24-45. Detailed Current Specifications

Parameter	Parameter Name	Conditions	Max	Unit
I_{DD_RUN}	Run mode 1 (Flash loop)	V_{DD} = 3.6 V Code= while(1){} executed out of Flash Peripherals = All ON System Clock = 80 MHz (with PLL) Temperature = 85°C	210[a] 138[b]	mA
I_{DD_SLEEP}	Sleep mode	V_{DD} = 3.6 V Peripherals = All Clock Gated System Clock = 80 MHz (with PLL) Temperature = 85°C	46	mA
$I_{DD\_DEEPSLEEP}$	Deep-Sleep mode	V_{DD} = 3.6 V Peripherals = All Clock Gated System Clock = IOSC30/64 Temperature = 85°C	1.8	mA

a. Auto-negotiate enabled. If an Ethernet cable is attached to the connector, the consumption increases by 7-10 mA.

b. Ethernet MAC and PHY powered down by software.

A Register Quick Reference

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
The Cortex-M3 Processor															
R0, type R/W, , reset - (see page 72)															
DATA															
DATA															
R1, type R/W, , reset - (see page 72)															
DATA															
DATA															
R2, type R/W, , reset - (see page 72)															
DATA															
DATA															
R3, type R/W, , reset - (see page 72)															
DATA															
DATA															
R4, type R/W, , reset - (see page 72)															
DATA															
DATA															
R5, type R/W, , reset - (see page 72)															
DATA															
DATA															
R6, type R/W, , reset - (see page 72)															
DATA															
DATA															
R7, type R/W, , reset - (see page 72)															
DATA															
DATA															
R8, type R/W, , reset - (see page 72)															
DATA															
DATA															
R9, type R/W, , reset - (see page 72)															
DATA															
DATA															
R10, type R/W, , reset - (see page 72)															
DATA															
DATA															
R11, type R/W, , reset - (see page 72)															
DATA															
DATA															
R12, type R/W, , reset - (see page 72)															
DATA															
DATA															
SP, type R/W, , reset - (see page 73)															
SP															
SP															
LR, type R/W, , reset 0xFFFF.FFFF (see page 74)															
LINK															
LINK															
PC, type R/W, , reset - (see page 75)															
PC															
PC															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSR, type R/W, , reset 0x0100.0000 (see page 76)															
N	Z	C	V	Q	ICI / IT		THUMB								
ICI / IT									ISRNUM						
PRIMASK, type R/W, , reset 0x0000.0000 (see page 80)															
															PRIMASK
FAULTMASK, type R/W, , reset 0x0000.0000 (see page 81)															
															FAULTMASK
BASEPRI, type R/W, , reset 0x0000.0000 (see page 82)															
								BASEPRI							
CONTROL, type R/W, , reset 0x0000.0000 (see page 83)															
														ASP	TMPL

Cortex-M3 Peripherals

System Timer (SysTick) Registers

Base 0xE000.E000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STCTRL, type R/W, offset 0x010, reset 0x0000.0004															
															COUNT
													CLK_SRC	INTEN	ENABLE
STRELOAD, type R/W, offset 0x014, reset 0x0000.0000															
								RELOAD							
RELOAD															
STCURRENT, type R/WC, offset 0x018, reset 0x0000.0000															
								CURRENT							
CURRENT															

Cortex-M3 Peripherals

Nested Vectored Interrupt Controller (NVIC) Registers

Base 0xE000.E000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN0, type R/W, offset 0x100, reset 0x0000.0000															
INT															
INT															
EN1, type R/W, offset 0x104, reset 0x0000.0000															
									INT						
INT															
DIS0, type R/W, offset 0x180, reset 0x0000.0000															
INT															
INT															
DIS1, type R/W, offset 0x184, reset 0x0000.0000															
									INT						
INT															
PEND0, type R/W, offset 0x200, reset 0x0000.0000															
INT															
INT															
PEND1, type R/W, offset 0x204, reset 0x0000.0000															
									INT						
INT															
UNPEND0, type R/W, offset 0x280, reset 0x0000.0000															
INT															
INT															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNPEND1, type R/W, offset 0x284, reset 0x0000.0000															
									INT						
INT															
ACTIVE0, type RO, offset 0x300, reset 0x0000.0000															
INT															
INT															
ACTIVE1, type RO, offset 0x304, reset 0x0000.0000															
									INT						
INT															
PRI0, type R/W, offset 0x400, reset 0x0000.0000															
INTD								INTC							
INTB								INTA							
PRI1, type R/W, offset 0x404, reset 0x0000.0000															
INTD								INTC							
INTB								INTA							
PRI2, type R/W, offset 0x408, reset 0x0000.0000															
INTD								INTC							
INTB								INTA							
PRI3, type R/W, offset 0x40C, reset 0x0000.0000															
INTD								INTC							
INTB								INTA							
PRI4, type R/W, offset 0x410, reset 0x0000.0000															
INTD								INTC							
INTB								INTA							
PRI5, type R/W, offset 0x414, reset 0x0000.0000															
INTD								INTC							
INTB								INTA							
PRI6, type R/W, offset 0x418, reset 0x0000.0000															
INTD								INTC							
INTB								INTA							
PRI7, type R/W, offset 0x41C, reset 0x0000.0000															
INTD								INTC							
INTB								INTA							
PRI8, type R/W, offset 0x420, reset 0x0000.0000															
INTD								INTC							
INTB								INTA							
PRI9, type R/W, offset 0x424, reset 0x0000.0000															
INTD								INTC							
INTB								INTA							
PRI10, type R/W, offset 0x428, reset 0x0000.0000															
INTD								INTC							
INTB								INTA							
PRI11, type R/W, offset 0x42C, reset 0x0000.0000															
INTD								INTC							
INTB								INTA							
PRI12, type R/W, offset 0x430, reset 0x0000.0000															
INTD								INTC							
INTB								INTA							
PRI13, type R/W, offset 0x434, reset 0x0000.0000															
INTD								INTC							
INTB								INTA							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWTRIG, type WO, offset 0xF00, reset 0x0000.0000															
										INTID					
Cortex-M3 Peripherals **System Control Block (SCB) Registers** Base 0xE000.E000															
ACTLR, type R/W, offset 0x008, reset 0x0000.0000															
													DISFOLD	DISWBUF	DISMCYC
CPUID, type RO, offset 0xD00, reset 0x412F.C230															
IMP								VAR				CON			
PARTNO												REV			
INTCTRL, type R/W, offset 0xD04, reset 0x0000.0000															
NMISET			PENDSV	UNPENDSV	PENDSTSET	PENDSTCLR		ISRPRE	ISRPEND				VECPEND		
VECPEND				RETBASE					VECACT						
VTABLE, type R/W, offset 0xD08, reset 0x0000.0000															
		BASE	OFFSET												
OFFSET															
APINT, type R/W, offset 0xD0C, reset 0xFA05.0000															
VECTKEY															
ENDIANESS					PRIGROUP								SYSRESREQ	VECTCLRACT	VECTRESET
SYSCTRL, type R/W, offset 0xD10, reset 0x0000.0000															
											SEVONPEND		SLEEPDEEP	SLEEPEXIT	
CFGCTRL, type R/W, offset 0xD14, reset 0x0000.0200															
						STKALIGN	BFHFNMIGN				DIV0	UNALIGNED		MAINPEND	BASETHR
SYSPRI1, type R/W, offset 0xD18, reset 0x0000.0000															
								USAGE							
BUS								MEM							
SYSPRI2, type R/W, offset 0xD1C, reset 0x0000.0000															
SVC															
SYSPRI3, type R/W, offset 0xD20, reset 0x0000.0000															
TICK								PENDSV							
								DEBUG							
SYSHNDCTRL, type R/W, offset 0xD24, reset 0x0000.0000															
													USAGE	BUS	MEM
SVC	BUSP	MEMP	USAGEP	TICK	PNDSV		MON	SVCA				USGA		BUSA	MEMA
FAULTSTAT, type R/W1C, offset 0xD28, reset 0x0000.0000															
						DIV0	UNALIGN					NOCP	INVPC	INVSTAT	UNDEF
BFARV			BSTKE	BUSTKE	IMPRE	PRECISE	IBUS	MMARV			MSTKE	MUSTKE		DERR	IERR
HFAULTSTAT, type R/W1C, offset 0xD2C, reset 0x0000.0000															
DBG	FORCED														
														VECT	
MMADDR, type R/W, offset 0xD34, reset -															
ADDR															
ADDR															
FAULTADDR, type R/W, offset 0xD38, reset -															
ADDR															
ADDR															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Cortex-M3 Peripherals **Memory Protection Unit (MPU) Registers** Base 0xE000.E000															
MPUTYPE, type RO, offset 0xD90, reset 0x0000.0800															
								IREGION							
DREGION															SEPARATE
MPUCTRL, type R/W, offset 0xD94, reset 0x0000.0000															
													PRIVDEFEN	HFNMIENA	ENABLE
MPUNUMBER, type R/W, offset 0xD98, reset 0x0000.0000															
													NUMBER		
MPUBASE, type R/W, offset 0xD9C, reset 0x0000.0000															
ADDR															
ADDR											VALID		REGION		
MPUBASE1, type R/W, offset 0xDA4, reset 0x0000.0000															
ADDR															
ADDR											VALID		REGION		
MPUBASE2, type R/W, offset 0xDAC, reset 0x0000.0000															
ADDR															
ADDR											VALID		REGION		
MPUBASE3, type R/W, offset 0xDB4, reset 0x0000.0000															
ADDR															
ADDR											VALID		REGION		
MPUATTR, type R/W, offset 0xDA0, reset 0x0000.0000															
			XN		AP					TEX			S	C	B
SRD										SIZE					ENABLE
MPUATTR1, type R/W, offset 0xDA8, reset 0x0000.0000															
			XN		AP					TEX			S	C	B
SRD										SIZE					ENABLE
MPUATTR2, type R/W, offset 0xDB0, reset 0x0000.0000															
			XN		AP					TEX			S	C	B
SRD										SIZE					ENABLE
MPUATTR3, type R/W, offset 0xDB8, reset 0x0000.0000															
			XN		AP					TEX			S	C	B
SRD										SIZE					ENABLE
System Control Base 0x400F.E000															
DID0, type RO, offset 0x000, reset - (see page 204)															
	VER							CLASS							
MAJOR								MINOR							
PBORCTL, type R/W, offset 0x030, reset 0x0000.0002 (see page 206)															
														BORIOR	
RIS, type RO, offset 0x050, reset 0x0000.0000 (see page 207)															
							MOSCPUPRIS	USBPLLLRIS	PLLLRIS					BORRIS	
IMC, type R/W, offset 0x054, reset 0x0000.0000 (see page 209)															
							MOSCPUPIM	USBPLLLIM	PLLLIM					BORIM	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISC, type R/W1C, offset 0x058, reset 0x0000.0000 (see page 211)															
							MOSCPUPMIS	USBPLLLMIS	PLLLMIS					BORMIS	
RESC, type R/W, offset 0x05C, reset - (see page 213)															
															MOSCFAIL
										WDT1	SW	WDT0	BOR	POR	EXT
RCC, type R/W, offset 0x060, reset 0x0780.3AD1 (see page 215)															
				ACG	SYSDIV				USESYSDIV						
		PWRDN		BYPASS	XTAL					OSCSRC				IOSCDIS	MOSCDIS
PLLCFG, type RO, offset 0x064, reset - (see page 219)															
		F									R				
GPIOHBCTL, type R/W, offset 0x06C, reset 0x0000.0000 (see page 220)															
							PORTJ	PORTH	PORTG	PORTF	PORTE	PORTD	PORTC	PORTB	PORTA
RCC2, type R/W, offset 0x070, reset 0x07C0.6810 (see page 222)															
USERCC2	DIV400		SYSDIV2						SYSDIV2LSB						
	USBPWRDN	PWRDN2		BYPASS2					OSCSRC2						
MOSCCTL, type R/W, offset 0x07C, reset 0x0000.0000 (see page 225)															
															CVAL
DSLPCLKCFG, type R/W, offset 0x144, reset 0x0780.0000 (see page 226)															
			DSDIVORIDE												
									DSOSCSRC						
PIOSCCAL, type R/W, offset 0x150, reset 0x0000.0000 (see page 228)															
UTEN															
							UPDATE		UT						
I2SMCLKCFG, type R/W, offset 0x170, reset 0x0000.0000 (see page 229)															
RXEN		RXI										RXF			
TXEN		TXI										TXF			
DID1, type RO, offset 0x004, reset - (see page 231)															
VER				FAM				PARTNO							
PINCOUNT								TEMP			PKG		ROHS	QUAL	
DC0, type RO, offset 0x008, reset 0x017F.00BF (see page 233)															
SRAMSZ															
FLASHSZ															
DC1, type RO, offset 0x010, reset - (see page 234)															
			WDT1		CAN2	CAN1	CAN0							ADC1	ADC0
MINSYSDIV				MAXADC1SPD		MAXADC0SPD		MPU		TEMPSNS	PLL	WDT0	SWO	SWD	JTAG
DC2, type RO, offset 0x014, reset 0x570F.5037 (see page 237)															
	EPI0		I2S0		COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
	I2C1		I2C0							SSI1	SSI0		UART2	UART1	UART0
DC3, type RO, offset 0x018, reset 0xBFFF.7FC0 (see page 239)															
32KHZ		CCP5	CCP4	CCP3	CCP2	CCP1	CCP0	ADC0AIN7	ADC0AIN6	ADC0AIN5	ADC0AIN4	ADC0AIN3	ADC0AIN2	ADC0AIN1	ADC0AIN0
	C2O	C2PLUS	C2MINUS	C1O	C1PLUS	C1MINUS	C0O	C0PLUS	C0MINUS						
DC4, type RO, offset 0x01C, reset 0x5004.F1FF (see page 241)															
	EPHY0		EMAC0										PICAL		
CCP7	CCP6	UDMA	ROM				GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
DC5, type RO, offset 0x020, reset 0x0000.0000 (see page 243)															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DC6, type RO, offset 0x024, reset 0x0000.0013 (see page 244)															
											USB0PHY			USB0	
DC7, type RO, offset 0x028, reset 0xFFFF.FFFF (see page 245)															
	DMACH30	DMACH29	DMACH28	DMACH27	DMACH26	DMACH25	DMACH24	DMACH23	DMACH22	DMACH21	DMACH20	DMACH19	DMACH18	DMACH17	DMACH16
DMACH15	DMACH14	DMACH13	DMACH12	DMACH11	DMACH10	DMACH9	DMACH8	DMACH7	DMACH6	DMACH5	DMACH4	DMACH3	DMACH2	DMACH1	DMACH0
DC8, type RO, offset 0x02C, reset 0xFFFF.FFFF (see page 249)															
ADC1AIN15	ADC1AIN14	ADC1AIN13	ADC1AIN12	ADC1AIN11	ADC1AIN10	ADC1AIN9	ADC1AIN8	ADC1AIN7	ADC1AIN6	ADC1AIN5	ADC1AIN4	ADC1AIN3	ADC1AIN2	ADC1AIN1	ADC1AIN0
ADC0AIN15	ADC0AIN14	ADC0AIN13	ADC0AIN12	ADC0AIN11	ADC0AIN10	ADC0AIN9	ADC0AIN8	ADC0AIN7	ADC0AIN6	ADC0AIN5	ADC0AIN4	ADC0AIN3	ADC0AIN2	ADC0AIN1	ADC0AIN0
DC9, type RO, offset 0x190, reset 0x00FF.00FF (see page 252)															
								ADC1DC7	ADC1DC6	ADC1DC5	ADC1DC4	ADC1DC3	ADC1DC2	ADC1DC1	ADC1DC0
								ADC0DC7	ADC0DC6	ADC0DC5	ADC0DC4	ADC0DC3	ADC0DC2	ADC0DC1	ADC0DC0
NVMSTAT, type RO, offset 0x1A0, reset 0x0000.0001 (see page 254)															
															FWB
RCGC0, type R/W, offset 0x100, reset 0x00000040 (see page 255)															
			WDT1		CAN2	CAN1	CAN0							ADC1	ADC0
				MAXADC1SPD		MAXADC0SPD						WDT0			
SCGC0, type R/W, offset 0x110, reset 0x00000040 (see page 258)															
			WDT1		CAN2	CAN1	CAN0							ADC1	ADC0
				MAXADC1SPD		MAXADC0SPD						WDT0			
DCGC0, type R/W, offset 0x120, reset 0x00000040 (see page 261)															
			WDT1		CAN2	CAN1	CAN0							ADC1	ADC0
												WDT0			
RCGC1, type R/W, offset 0x104, reset 0x00000000 (see page 263)															
	EPI0		I2S0		COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
	I2C1		I2C0							SSI1	SSI0		UART2	UART1	UART0
SCGC1, type R/W, offset 0x114, reset 0x00000000 (see page 266)															
	EPI0		I2S0		COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
	I2C1		I2C0							SSI1	SSI0		UART2	UART1	UART0
DCGC1, type R/W, offset 0x124, reset 0x00000000 (see page 269)															
	EPI0		I2S0		COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
	I2C1		I2C0							SSI1	SSI0		UART2	UART1	UART0
RCGC2, type R/W, offset 0x108, reset 0x00000000 (see page 272)															
	EPHY0		EMAC0												USB0
		UDMA					GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
SCGC2, type R/W, offset 0x118, reset 0x00000000 (see page 275)															
	EPHY0		EMAC0												USB0
		UDMA					GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
DCGC2, type R/W, offset 0x128, reset 0x00000000 (see page 278)															
	EPHY0		EMAC0												USB0
		UDMA					GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
SRCR0, type R/W, offset 0x040, reset 0x00000000 (see page 281)															
			WDT1		CAN2	CAN1	CAN0							ADC1	ADC0
												WDT0			
SRCR1, type R/W, offset 0x044, reset 0x00000000 (see page 283)															
	EPI0		I2S0		COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
	I2C1		I2C0							SSI1	SSI0		UART2	UART1	UART0
SRCR2, type R/W, offset 0x048, reset 0x00000000 (see page 286)															
	EPHY0		EMAC0												USB0
		UDMA					GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Internal Memory **Flash Memory Registers (Flash Control Offset)** Base 0x400F.D000															
FMA, type R/W, offset 0x000, reset 0x0000.0000															
													OFFSET		
OFFSET															
FMD, type R/W, offset 0x004, reset 0x0000.0000															
DATA															
DATA															
FMC, type R/W, offset 0x008, reset 0x0000.0000															
WRKEY															
												COMT	MERASE	ERASE	WRITE
FCRIS, type RO, offset 0x00C, reset 0x0000.0000															
														PRIS	ARIS
FCIM, type R/W, offset 0x010, reset 0x0000.0000															
														PMASK	AMASK
FCMISC, type R/W1C, offset 0x014, reset 0x0000.0000															
														PMISC	AMISC
FMC2, type R/W, offset 0x020, reset 0x0000.0000															
WRKEY															
															WRBUF
FWBVAL, type R/W, offset 0x030, reset 0x0000.0000															
FWB[n]															
FWB[n]															
FCTL, type R/W, offset 0x0F8, reset 0x0000.0000															
														USDACK	USDREQ
FWBn, type R/W, offset 0x100 - 0x17C, reset 0x0000.0000															
DATA															
DATA															
Internal Memory **Memory Registers (System Control Offset)** Base 0x400F.E000															
RMCTL, type R/W1C, offset 0x0F0, reset -															
															BA
FMPRE0, type R/W, offset 0x130 and 0x200, reset 0xFFFF.FFFF															
READ_ENABLE															
READ_ENABLE															
FMPPE0, type R/W, offset 0x134 and 0x400, reset 0xFFFF.FFFF															
PROG_ENABLE															
PROG_ENABLE															
BOOTCFG, type R/W, offset 0x1D0, reset 0xFFFF.FFFE															
NW															
PORT				PIN			POL	EN						DBG1	DBG0
USER_REG0, type R/W, offset 0x1E0, reset 0xFFFF.FFFF															
NW	DATA														
DATA															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USER_REG1, type R/W, offset 0x1E4, reset 0xFFFF.FFFF															
NW	DATA														
DATA															
USER_REG2, type R/W, offset 0x1E8, reset 0xFFFF.FFFF															
NW	DATA														
DATA															
USER_REG3, type R/W, offset 0x1EC, reset 0xFFFF.FFFF															
NW	DATA														
DATA															
FMPRE1, type R/W, offset 0x204, reset 0xFFFF.FFFF															
READ_ENABLE															
READ_ENABLE															
FMPRE2, type R/W, offset 0x208, reset 0xFFFF.FFFF															
READ_ENABLE															
READ_ENABLE															
FMPRE3, type R/W, offset 0x20C, reset 0xFFFF.FFFF															
READ_ENABLE															
READ_ENABLE															
FMPRE4, type R/W, offset 0x210, reset 0xFFFF.FFFF															
READ_ENABLE															
READ_ENABLE															
FMPRE5, type R/W, offset 0x214, reset 0xFFFF.FFFF															
READ_ENABLE															
READ_ENABLE															
FMPRE6, type R/W, offset 0x218, reset 0x0000.0000															
READ_ENABLE															
READ_ENABLE															
FMPRE7, type R/W, offset 0x21C, reset 0x0000.0000															
READ_ENABLE															
READ_ENABLE															
FMPPE1, type R/W, offset 0x404, reset 0xFFFF.FFFF															
PROG_ENABLE															
PROG_ENABLE															
FMPPE2, type R/W, offset 0x408, reset 0xFFFF.FFFF															
PROG_ENABLE															
PROG_ENABLE															
FMPPE3, type R/W, offset 0x40C, reset 0xFFFF.FFFF															
PROG_ENABLE															
PROG_ENABLE															
FMPPE4, type R/W, offset 0x410, reset 0xFFFF.FFFF															
PROG_ENABLE															
PROG_ENABLE															
FMPPE5, type R/W, offset 0x414, reset 0xFFFF.FFFF															
PROG_ENABLE															
PROG_ENABLE															
FMPPE6, type R/W, offset 0x418, reset 0x0000.0000															
PROG_ENABLE															
PROG_ENABLE															
FMPPE7, type R/W, offset 0x41C, reset 0x0000.0000															
PROG_ENABLE															
PROG_ENABLE															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Micro Direct Memory Access (µDMA) **µDMA Channel Control Structure (Offset from Channel Control Table Base)** Base n/a															
DMASRCENDP, type R/W, offset 0x000, reset -															
ADDR															
ADDR															
DMADSTENDP, type R/W, offset 0x004, reset -															
ADDR															
ADDR															
DMACHCTL, type R/W, offset 0x008, reset -															
DSTINC		DSTSIZE		SRCINC		SRCSIZE								ARBSIZE	
ARBSIZE				XFERSIZE								NXTUSEBURST		XFERMODE	
Micro Direct Memory Access (µDMA) **µDMA Registers (Offset from µDMA Base Address)** Base 0x400F.F000															
DMASTAT, type RO, offset 0x000, reset 0x001F.0000															
											DMACHANS				
								STATE							MASTEN
DMACFG, type WO, offset 0x004, reset -															
															MASTEN
DMACTLBASE, type R/W, offset 0x008, reset 0x0000.0000															
ADDR															
ADDR															
DMAALTBASE, type RO, offset 0x00C, reset 0x0000.0200															
ADDR															
ADDR															
DMAWAITSTAT, type RO, offset 0x010, reset 0xFFFF.FFC0															
WAITREQ[n]															
WAITREQ[n]															
DMASWREQ, type WO, offset 0x014, reset -															
SWREQ[n]															
SWREQ[n]															
DMAUSEBURSTSET, type R/W, offset 0x018, reset 0x0000.0000															
SET[n]															
SET[n]															
DMAUSEBURSTCLR, type WO, offset 0x01C, reset -															
CLR[n]															
CLR[n]															
DMAREQMASKSET, type R/W, offset 0x020, reset 0x0000.0000															
SET[n]															
SET[n]															
DMAREQMASKCLR, type WO, offset 0x024, reset -															
CLR[n]															
CLR[n]															
DMAENASET, type R/W, offset 0x028, reset 0x0000.0000															
SET[n]															
SET[n]															
DMAENACLR, type WO, offset 0x02C, reset -															
CLR[n]															
CLR[n]															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAALTSET, type R/W, offset 0x030, reset 0x0000.0000															
SET[n]															
SET[n]															
DMAALTCLR, type WO, offset 0x034, reset -															
CLR[n]															
CLR[n]															
DMAPRIOSET, type R/W, offset 0x038, reset 0x0000.0000															
SET[n]															
SET[n]															
DMAPRIOCLR, type WO, offset 0x03C, reset -															
CLR[n]															
CLR[n]															
DMAERRCLR, type R/W, offset 0x04C, reset 0x0000.0000															
															ERRCLR
DMACHASGN, type R/W, offset 0x500, reset 0x0000.0000															
CHASGN[n]															
CHASGN[n]															
DMACHIS, type R/W1C, offset 0x504, reset 0x0000.0000															
CHIS[n]															
CHIS[n]															
DMAPeriphID0, type RO, offset 0xFE0, reset 0x0000.0030															
								PID0							
DMAPeriphID1, type RO, offset 0xFE4, reset 0x0000.00B2															
								PID1							
DMAPeriphID2, type RO, offset 0xFE8, reset 0x0000.000B															
								PID2							
DMAPeriphID3, type RO, offset 0xFEC, reset 0x0000.0000															
								PID3							
DMAPeriphID4, type RO, offset 0xFD0, reset 0x0000.0004															
								PID4							
DMAPCellID0, type RO, offset 0xFF0, reset 0x0000.000D															
								CID0							
DMAPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0															
								CID1							
DMAPCellID2, type RO, offset 0xFF8, reset 0x0000.0005															
								CID2							
DMAPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1															
								CID3							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
General-Purpose Input/Outputs (GPIOs) GPIO Port A (APB) base: 0x4000.4000 GPIO Port A (AHB) base: 0x4005.8000 GPIO Port B (APB) base: 0x4000.5000 GPIO Port B (AHB) base: 0x4005.9000 GPIO Port C (APB) base: 0x4000.6000 GPIO Port C (AHB) base: 0x4005.A000 GPIO Port D (APB) base: 0x4000.7000 GPIO Port D (AHB) base: 0x4005.B000 GPIO Port E (APB) base: 0x4002.4000 GPIO Port E (AHB) base: 0x4005.C000 GPIO Port F (APB) base: 0x4002.5000 GPIO Port F (AHB) base: 0x4005.D000 GPIO Port G (APB) base: 0x4002.6000 GPIO Port G (AHB) base: 0x4005.E000 GPIO Port H (APB) base: 0x4002.7000 GPIO Port H (AHB) base: 0x4005.F000 GPIO Port J (APB) base: 0x4003.D000 GPIO Port J (AHB) base: 0x4006.0000															
GPIODATA, type R/W, offset 0x000, reset 0x0000.0000 (see page 409)															
								DATA							
GPIODIR, type R/W, offset 0x400, reset 0x0000.0000 (see page 410)															
								DIR							
GPIOIS, type R/W, offset 0x404, reset 0x0000.0000 (see page 411)															
								IS							
GPIOIBE, type R/W, offset 0x408, reset 0x0000.0000 (see page 412)															
								IBE							
GPIOIEV, type R/W, offset 0x40C, reset 0x0000.0000 (see page 413)															
								IEV							
GPIOIM, type R/W, offset 0x410, reset 0x0000.0000 (see page 414)															
								IME							
GPIORIS, type RO, offset 0x414, reset 0x0000.0000 (see page 415)															
								RIS							
GPIOMIS, type RO, offset 0x418, reset 0x0000.0000 (see page 416)															
								MIS							
GPIOICR, type W1C, offset 0x41C, reset 0x0000.0000 (see page 418)															
								IC							
GPIOAFSEL, type R/W, offset 0x420, reset - (see page 419)															
								AFSEL							
GPIODR2R, type R/W, offset 0x500, reset 0x0000.00FF (see page 421)															
								DRV2							
GPIODR4R, type R/W, offset 0x504, reset 0x0000.0000 (see page 422)															
								DRV4							
GPIODR8R, type R/W, offset 0x508, reset 0x0000.0000 (see page 423)															
								DRV8							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOODR, type R/W, offset 0x50C, reset 0x0000.0000 (see page 424)															
								ODE							
GPIOPUR, type R/W, offset 0x510, reset - (see page 425)															
								PUE							
GPIOPDR, type R/W, offset 0x514, reset 0x0000.0000 (see page 427)															
								PDE							
GPIOSLR, type R/W, offset 0x518, reset 0x0000.0000 (see page 429)															
								SRL							
GPIODEN, type R/W, offset 0x51C, reset - (see page 430)															
								DEN							
GPIOLOCK, type R/W, offset 0x520, reset 0x0000.0001 (see page 432)															
LOCK															
LOCK															
GPIOCR, type -, offset 0x524, reset - (see page 433)															
								CR							
GPIOAMSEL, type R/W, offset 0x528, reset 0x0000.0000 (see page 435)															
								GPIOAMSEL							
GPIOPCTL, type R/W, offset 0x52C, reset - (see page 437)															
PMC7				PMC6				PMC5				PMC4			
PMC3				PMC2				PMC1				PMC0			
GPIOPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000 (see page 439)															
								PID4							
GPIOPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000 (see page 440)															
								PID5							
GPIOPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000 (see page 441)															
								PID6							
GPIOPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000 (see page 442)															
								PID7							
GPIOPeriphID0, type RO, offset 0xFE0, reset 0x0000.0061 (see page 443)															
								PID0							
GPIOPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000 (see page 444)															
								PID1							
GPIOPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018 (see page 445)															
								PID2							
GPIOPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001 (see page 446)															
								PID3							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOPCellID0, type RO, offset 0xFF0, reset 0x0000.000D (see page 447)															
								CID0							
GPIOPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0 (see page 448)															
								CID1							
GPIOPCellID2, type RO, offset 0xFF8, reset 0x0000.0005 (see page 449)															
								CID2							
GPIOPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1 (see page 450)															
								CID3							
External Peripheral Interface (EPI) Base 0x400D.0000															
EPICFG, type R/W, offset 0x000, reset 0x0000.0000 (see page 483)															
											BLKEN	MODE			
EPIBAUD, type R/W, offset 0x004, reset 0x0000.0000 (see page 484)															
COUNT1															
COUNT0															
EPISDRAMCFG, type R/W, offset 0x010, reset 0x82EE.0000 (see page 486)															
FREQ					RFSH										
						SLEEP								SIZE	
EPIHB8CFG, type R/W, offset 0x010, reset 0x0000.FF00 (see page 488)															
								XFFEN	XFEEN	WRHIGH	RDHIGH				
MAXWAIT								WRWS		RDWS				MODE	
EPIHB16CFG, type R/W, offset 0x010, reset 0x0000.FF00 (see page 491)															
								XFFEN	XFEEN	WRHIGH	RDHIGH				
MAXWAIT								WRWS		RDWS			BSEL	MODE	
EPIGPCFG, type R/W, offset 0x010, reset 0x0000.0000 (see page 495)															
CLKPIN	CLKGATE		RDYEN	FRMPIN	FRM50	FRMCNT				RW		WR2CYC	RD2CYC		
MAXWAIT										ASIZE				DSIZE	
EPIHB8CFG2, type R/W, offset 0x014, reset 0x0000.0000 (see page 500)															
WORD					CSBAUD	CSCFG				WRHIGH	RDHIGH				
								WRWS		RDWS					
EPIHB16CFG2, type R/W, offset 0x014, reset 0x0000.0000 (see page 503)															
WORD					CSBAUD	CSCFG				WRHIGH	RDHIGH				
								WRWS							
EPIGPCFG2, type R/W, offset 0x014, reset 0x0000.0000 (see page 506)															
WORD															
EPIADDRMAP, type R/W, offset 0x01C, reset 0x0000.0000 (see page 507)															
								EPSZ		EPADR		ERSZ		ERADR	
EPIRSIZE0, type R/W, offset 0x020, reset 0x0000.0003 (see page 509)															
														SIZE	
EPIRSIZE1, type R/W, offset 0x030, reset 0x0000.0003 (see page 509)															
														SIZE	
EPIRADDR0, type R/W, offset 0x024, reset 0x0000.0000 (see page 510)															
			ADDR												
ADDR															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPIRADDR1, type R/W, offset 0x034, reset 0x0000.0000 (see page 510)															
			ADDR												
ADDR															
EPIRPSTD0, type R/W, offset 0x028, reset 0x0000.0000 (see page 511)															
			POSTCNT												
EPIRPSTD1, type R/W, offset 0x038, reset 0x0000.0000 (see page 511)															
			POSTCNT												
EPISTAT, type RO, offset 0x060, reset 0x0000.0000 (see page 513)															
						CELOW	XFFULL	XFEMPTY	INITSEQ	WBUSY	NBRBUSY				ACTIVE
EPIRFIFOCNT, type RO, offset 0x06C, reset - (see page 515)															
												COUNT			
EPIREADFIFO, type RO, offset 0x070, reset - (see page 516)															
DATA															
DATA															
EPIREADFIFO1, type RO, offset 0x074, reset - (see page 516)															
DATA															
DATA															
EPIREADFIFO2, type RO, offset 0x078, reset - (see page 516)															
DATA															
DATA															
EPIREADFIFO3, type RO, offset 0x07C, reset - (see page 516)															
DATA															
DATA															
EPIREADFIFO4, type RO, offset 0x080, reset - (see page 516)															
DATA															
DATA															
EPIREADFIFO5, type RO, offset 0x084, reset - (see page 516)															
DATA															
DATA															
EPIREADFIFO6, type RO, offset 0x088, reset - (see page 516)															
DATA															
DATA															
EPIREADFIFO7, type RO, offset 0x08C, reset - (see page 516)															
DATA															
DATA															
EPIFIFOLVL, type R/W, offset 0x200, reset 0x0000.0033 (see page 517)															
														WFERR	RSERR
									WRFIFO				RDFIFO		
EPIWFIFOCNT, type RO, offset 0x204, reset 0x0000.0004 (see page 519)															
													WTAV		
EPIIM, type R/W, offset 0x210, reset 0x0000.0000 (see page 520)															
													WRIM	RDIM	ERRIM
EPIRIS, type RO, offset 0x214, reset 0x0000.0004 (see page 521)															
													WRRIS	RDRIS	ERRRIS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPIMIS, type RO, offset 0x218, reset 0x0000.0000 (see page 523)															
													WRMIS	RDMIS	ERRMIS
EPIEISC, type R/W1C, offset 0x21C, reset 0x0000.0000 (see page 524)															
													WTFULL	RSTALL	TOUT
General-Purpose Timers Timer 0 base: 0x4003.0000 Timer 1 base: 0x4003.1000 Timer 2 base: 0x4003.2000 Timer 3 base: 0x4003.3000															
GPTMCFG, type R/W, offset 0x000, reset 0x0000.0000 (see page 543)															
													GPTMCFG		
GPTMTAMR, type R/W, offset 0x004, reset 0x0000.0000 (see page 544)															
								TASNAPS	TAWOT	TAMIE	TACDIR	TAAMS	TACMR	TAMR	
GPTMTBMR, type R/W, offset 0x008, reset 0x0000.0000 (see page 546)															
								TBSNAPS	TBWOT	TBMIE	TBCDIR	TBAMS	TBCMR	TBMR	
GPTMCTL, type R/W, offset 0x00C, reset 0x0000.0000 (see page 548)															
	TBPWML	TBOTE		TBEVENT		TBSTALL	TBEN		TAPWML	TAOTE	RTCEN	TAEVENT		TASTALL	TAEN
GPTMIMR, type R/W, offset 0x018, reset 0x0000.0000 (see page 551)															
				TBMIM	CBEIM	CBMIM	TBTOIM				TAMIM	RTCIM	CAEIM	CAMIM	TATOIM
GPTMRIS, type RO, offset 0x01C, reset 0x0000.0000 (see page 553)															
				TBMRIS	CBERIS	CBMRIS	TBTORIS				TAMRIS	RTCRIS	CAERIS	CAMRIS	TATORIS
GPTMMIS, type RO, offset 0x020, reset 0x0000.0000 (see page 556)															
				TBMMIS	CBEMIS	CBMMIS	TBTOMIS				TAMMIS	RTCMIS	CAEMIS	CAMMIS	TATOMIS
GPTMICR, type W1C, offset 0x024, reset 0x0000.0000 (see page 559)															
				TBMCINT	CBECINT	CBMCINT	TBTOCINT				TAMCINT	RTCCINT	CAECINT	CAMCINT	TATOCINT
GPTMTAILR, type R/W, offset 0x028, reset 0xFFFF.FFFF (see page 561)															
								TAILR							
								TAILR							
GPTMTBILR, type R/W, offset 0x02C, reset 0x0000.FFFF (see page 562)															
								TBILR							
								TBILR							
GPTMTAMATCHR, type R/W, offset 0x030, reset 0xFFFF.FFFF (see page 563)															
								TAMR							
								TAMR							
GPTMTBMATCHR, type R/W, offset 0x034, reset 0x0000.FFFF (see page 564)															
								TBMR							
								TBMR							
GPTMTAPR, type R/W, offset 0x038, reset 0x0000.0000 (see page 565)															
								TAPSR							
GPTMTBPR, type R/W, offset 0x03C, reset 0x0000.0000 (see page 566)															
								TBPSR							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPTMTAPMR, type R/W, offset 0x040, reset 0x0000.0000 (see page 567)															
								TAPSMR							
GPTMTBPMR, type R/W, offset 0x044, reset 0x0000.0000 (see page 568)															
								TBPSMR							
GPTMTAR, type RO, offset 0x048, reset 0xFFFF.FFFF (see page 569)															
TAR															
TAR															
GPTMTBR, type RO, offset 0x04C, reset 0x0000.FFFF (see page 570)															
TBR															
TBR															
GPTMTAV, type RW, offset 0x050, reset 0xFFFF.FFFF (see page 571)															
TAV															
TAV															
GPTMTBV, type RW, offset 0x054, reset 0x0000.FFFF (see page 572)															
TBV															
TBV															
Watchdog Timers WDT0 base: 0x4000.0000 WDT1 base: 0x4000.1000															
WDTLOAD, type R/W, offset 0x000, reset 0xFFFF.FFFF (see page 577)															
WDTLOAD															
WDTLOAD															
WDTVALUE, type RO, offset 0x004, reset 0xFFFF.FFFF (see page 578)															
WDTVALUE															
WDTVALUE															
WDTCTL, type R/W, offset 0x008, reset 0x0000.0000 (WDT0) and 0x8000.0000 (WDT1) (see page 579)															
WRC															
														RESEN	INTEN
WDTICR, type WO, offset 0x00C, reset - (see page 581)															
WDTINTCLR															
WDTINTCLR															
WDTRIS, type RO, offset 0x010, reset 0x0000.0000 (see page 582)															
															WDTRIS
WDTMIS, type RO, offset 0x014, reset 0x0000.0000 (see page 583)															
															WDTMIS
WDTTEST, type R/W, offset 0x418, reset 0x0000.0000 (see page 584)															
							STALL								
WDTLOCK, type R/W, offset 0xC00, reset 0x0000.0000 (see page 585)															
WDTLOCK															
WDTLOCK															
WDTPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000 (see page 586)															
								PID4							
WDTPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000 (see page 587)															
								PID5							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDTPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000 (see page 588)															
								PID6							
WDTPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000 (see page 589)															
								PID7							
WDTPeriphID0, type RO, offset 0xFE0, reset 0x0000.0005 (see page 590)															
								PID0							
WDTPeriphID1, type RO, offset 0xFE4, reset 0x0000.0018 (see page 591)															
								PID1							
WDTPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018 (see page 592)															
								PID2							
WDTPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001 (see page 593)															
								PID3							
WDTPCellID0, type RO, offset 0xFF0, reset 0x0000.000D (see page 594)															
								CID0							
WDTPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0 (see page 595)															
								CID1							
WDTPCellID2, type RO, offset 0xFF8, reset 0x0000.0006 (see page 596)															
								CID2							
WDTPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1 (see page 597)															
								CID3							
Analog-to-Digital Converter (ADC) ADC0 base: 0x4003.8000 ADC1 base: 0x4003.9000															
ADCACTSS, type R/W, offset 0x000, reset 0x0000.0000 (see page 621)															
												ASEN3	ASEN2	ASEN1	ASEN0
ADCRIS, type RO, offset 0x004, reset 0x0000.0000 (see page 622)															
															INRDC
												INR3	INR2	INR1	INR0
ADCIM, type R/W, offset 0x008, reset 0x0000.0000 (see page 624)															
												DCONSS3	DCONSS2	DCONSS1	DCONSS0
												MASK3	MASK2	MASK1	MASK0
ADCISC, type R/W1C, offset 0x00C, reset 0x0000.0000 (see page 626)															
												DCINSS3	DCINSS2	DCINSS1	DCINSS0
												IN3	IN2	IN1	IN0
ADCOSTAT, type R/W1C, offset 0x010, reset 0x0000.0000 (see page 629)															
												OV3	OV2	OV1	OV0
ADCEMUX, type R/W, offset 0x014, reset 0x0000.0000 (see page 631)															
EM3				EM2				EM1				EM0			

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCUSTAT, type R/W1C, offset 0x018, reset 0x0000.0000 (see page 636)															
												UV3	UV2	UV1	UV0
ADCSSPRI, type R/W, offset 0x020, reset 0x0000.3210 (see page 637)															
		SS3				SS2				SS1				SS0	
ADCSPC, type R/W, offset 0x024, reset 0x0000.0000 (see page 639)															
												PHASE			
ADCPSSI, type R/W, offset 0x028, reset - (see page 641)															
GSYNC				SYNCWAIT											
												SS3	SS2	SS1	SS0
ADCSAC, type R/W, offset 0x030, reset 0x0000.0000 (see page 643)															
													AVG		
ADCDCISC, type R/W1C, offset 0x034, reset 0x0000.0000 (see page 644)															
								DCINT7	DCINT6	DCINT5	DCINT4	DCINT3	DCINT2	DCINT1	DCINT0
ADCCTL, type R/W, offset 0x038, reset 0x0000.0000 (see page 646)															
											RES			VREF	
ADCSSMUX0, type R/W, offset 0x040, reset 0x0000.0000 (see page 647)															
MUX7				MUX6				MUX5				MUX4			
MUX3				MUX2				MUX1				MUX0			
ADCSSCTL0, type R/W, offset 0x044, reset 0x0000.0000 (see page 649)															
TS7	IE7	END7	D7	TS6	IE6	END6	D6	TS5	IE5	END5	D5	TS4	IE4	END4	D4
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
ADCSSFIFO0, type RO, offset 0x048, reset - (see page 652)															
				DATA											
ADCSSFIFO1, type RO, offset 0x068, reset - (see page 652)															
				DATA											
ADCSSFIFO2, type RO, offset 0x088, reset - (see page 652)															
				DATA											
ADCSSFIFO3, type RO, offset 0x0A8, reset - (see page 652)															
				DATA											
ADCSSFSTAT0, type RO, offset 0x04C, reset 0x0000.0100 (see page 653)															
			FULL				EMPTY	HPTR				TPTR			
ADCSSFSTAT1, type RO, offset 0x06C, reset 0x0000.0100 (see page 653)															
			FULL				EMPTY	HPTR				TPTR			
ADCSSFSTAT2, type RO, offset 0x08C, reset 0x0000.0100 (see page 653)															
			FULL				EMPTY	HPTR				TPTR			
ADCSSFSTAT3, type RO, offset 0x0AC, reset 0x0000.0100 (see page 653)															
			FULL				EMPTY	HPTR				TPTR			

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCSSOP0, type R/W, offset 0x050, reset 0x0000.0000 (see page 655)															
			S7DCOP				S6DCOP				S5DCOP				S4DCOP
			S3DCOP				S2DCOP				S1DCOP				S0DCOP
ADCSSDC0, type R/W, offset 0x054, reset 0x0000.0000 (see page 657)															
S7DCSEL				S6DCSEL				S5DCSEL				S4DCSEL			
S3DCSEL				S2DCSEL				S1DCSEL				S0DCSEL			
ADCSSMUX1, type R/W, offset 0x060, reset 0x0000.0000 (see page 659)															
MUX3				MUX2				MUX1				MUX0			
ADCSSMUX2, type R/W, offset 0x080, reset 0x0000.0000 (see page 659)															
MUX3				MUX2				MUX1				MUX0			
ADCSSCTL1, type R/W, offset 0x064, reset 0x0000.0000 (see page 660)															
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
ADCSSCTL2, type R/W, offset 0x084, reset 0x0000.0000 (see page 660)															
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
ADCSSOP1, type R/W, offset 0x070, reset 0x0000.0000 (see page 662)															
			S3DCOP				S2DCOP				S1DCOP				S0DCOP
ADCSSOP2, type R/W, offset 0x090, reset 0x0000.0000 (see page 662)															
			S3DCOP				S2DCOP				S1DCOP				S0DCOP
ADCSSDC1, type R/W, offset 0x074, reset 0x0000.0000 (see page 663)															
S3DCSEL				S2DCSEL				S1DCSEL				S0DCSEL			
ADCSSDC2, type R/W, offset 0x094, reset 0x0000.0000 (see page 663)															
S3DCSEL				S2DCSEL				S1DCSEL				S0DCSEL			
ADCSSMUX3, type R/W, offset 0x0A0, reset 0x0000.0000 (see page 665)															
												MUX0			
ADCSSCTL3, type R/W, offset 0x0A4, reset 0x0000.0002 (see page 666)															
												TS0	IE0	END0	D0
ADCSSOP3, type R/W, offset 0x0B0, reset 0x0000.0000 (see page 667)															
															S0DCOP
ADCSSDC3, type R/W, offset 0x0B4, reset 0x0000.0000 (see page 668)															
												S0DCSEL			
ADCDCRIC, type R/W, offset 0xD00, reset 0x0000.0000 (see page 669)															
								DCTRIG7	DCTRIG6	DCTRIG5	DCTRIG4	DCTRIG3	DCTRIG2	DCTRIG1	DCTRIG0
								DCINT7	DCINT6	DCINT5	DCINT4	DCINT3	DCINT2	DCINT1	DCINT0
ADCDCCTL0, type R/W, offset 0xE00, reset 0x0000.0000 (see page 674)															
											CIE	CIC		CIM	
ADCDCCTL1, type R/W, offset 0xE04, reset 0x0000.0000 (see page 674)															
											CIE	CIC		CIM	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCDCCTL2, type R/W, offset 0xE08, reset 0x0000.0000 (see page 674)															
											CIE	CIC		CIM	
ADCDCCTL3, type R/W, offset 0xE0C, reset 0x0000.0000 (see page 674)															
											CIE	CIC		CIM	
ADCDCCTL4, type R/W, offset 0xE10, reset 0x0000.0000 (see page 674)															
											CIE	CIC		CIM	
ADCDCCTL5, type R/W, offset 0xE14, reset 0x0000.0000 (see page 674)															
											CIE	CIC		CIM	
ADCDCCTL6, type R/W, offset 0xE18, reset 0x0000.0000 (see page 674)															
											CIE	CIC		CIM	
ADCDCCTL7, type R/W, offset 0xE1C, reset 0x0000.0000 (see page 674)															
											CIE	CIC		CIM	
ADCDCCMP0, type R/W, offset 0xE40, reset 0x0000.0000 (see page 676)															
				COMP1											
				COMP0											
ADCDCCMP1, type R/W, offset 0xE44, reset 0x0000.0000 (see page 676)															
				COMP1											
				COMP0											
ADCDCCMP2, type R/W, offset 0xE48, reset 0x0000.0000 (see page 676)															
				COMP1											
				COMP0											
ADCDCCMP3, type R/W, offset 0xE4C, reset 0x0000.0000 (see page 676)															
				COMP1											
				COMP0											
ADCDCCMP4, type R/W, offset 0xE50, reset 0x0000.0000 (see page 676)															
				COMP1											
				COMP0											
ADCDCCMP5, type R/W, offset 0xE54, reset 0x0000.0000 (see page 676)															
				COMP1											
				COMP0											
ADCDCCMP6, type R/W, offset 0xE58, reset 0x0000.0000 (see page 676)															
				COMP1											
				COMP0											
ADCDCCMP7, type R/W, offset 0xE5C, reset 0x0000.0000 (see page 676)															
				COMP1											
				COMP0											
Universal Asynchronous Receivers/Transmitters (UARTs) UART0 base: 0x4000.C000 UART1 base: 0x4000.D000 UART2 base: 0x4000.E000															
UARTDR, type R/W, offset 0x000, reset 0x0000.0000 (see page 693)															
				OE	BE	PE	FE	DATA							
UARTRSR/UARTECR, type RO, offset 0x004, reset 0x0000.0000 (Read-Only Status Register) (see page 695)															
												OE	BE	PE	FE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UARTRSR/UARTECR, type WO, offset 0x004, reset 0x0000.0000 (Write-Only Error Clear Register) (see page 695)															
								DATA							
UARTFR, type RO, offset 0x018, reset 0x0000.0090 (see page 698)															
							RI	TXFE	RXFF	TXFF	RXFE	BUSY	DCD	DSR	CTS
UARTILPR, type R/W, offset 0x020, reset 0x0000.0000 (see page 701)															
								ILPDVSR							
UARTIBRD, type R/W, offset 0x024, reset 0x0000.0000 (see page 702)															
DIVINT															
UARTFBRD, type R/W, offset 0x028, reset 0x0000.0000 (see page 703)															
										DIVFRAC					
UARTLCRH, type R/W, offset 0x02C, reset 0x0000.0000 (see page 704)															
								SPS	WLEN		FEN	STP2	EPS	PEN	BRK
UARTCTL, type R/W, offset 0x030, reset 0x0000.0300 (see page 706)															
CTSEN	RTSEN			RTS	DTR	RXE	TXE	LBE	LIN	HSE	EOT	SMART	SIRLP	SIREN	UARTEN
UARTIFLS, type R/W, offset 0x034, reset 0x0000.0012 (see page 710)															
										RXIFLSEL			TXIFLSEL		
UARTIM, type R/W, offset 0x038, reset 0x0000.0000 (see page 712)															
LME5IM	LME1IM	LMSBIM			OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM	DSRIM	DCDIM	CTSIM	RIIM
UARTRIS, type RO, offset 0x03C, reset 0x0000.000F (see page 716)															
LME5RIS	LME1RIS	LMSBRIS			OERIS	BERIS	PERIS	FERIS	RTRIS	TXRIS	RXRIS	DSRRIS	DCDRIS	CTSRIS	RIRIS
UARTMIS, type RO, offset 0x040, reset 0x0000.0000 (see page 720)															
LME5MIS	LME1MIS	LMSBMIS			OEMIS	BEMIS	PEMIS	FEMIS	RTMIS	TXMIS	RXMIS	DSRMIS	DCDMIS	CTSMIS	RIMIS
UARTICR, type W1C, offset 0x044, reset 0x0000.0000 (see page 724)															
LME5IC	LME1IC	LMSBIC			OEIC	BEIC	PEIC	FEIC	RTIC	TXIC	RXIC	DSRMIC	DCDMIC	CTSMIC	RIMIC
UARTDMACTL, type R/W, offset 0x048, reset 0x0000.0000 (see page 726)															
													DMAERR	TXDMAE	RXDMAE
UARTLCTL, type R/W, offset 0x090, reset 0x0000.0000 (see page 727)															
										BLEN					MASTER
UARTLSS, type RO, offset 0x094, reset 0x0000.0000 (see page 728)															
TSS															
UARTLTIM, type RO, offset 0x098, reset 0x0000.0000 (see page 729)															
TIMER															
UARTPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000 (see page 730)															
								PID4							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UARTPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000 (see page 731)															
								PID5							
UARTPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000 (see page 732)															
								PID6							
UARTPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000 (see page 733)															
								PID7							
UARTPeriphID0, type RO, offset 0xFE0, reset 0x0000.0060 (see page 734)															
								PID0							
UARTPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000 (see page 735)															
								PID1							
UARTPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018 (see page 736)															
								PID2							
UARTPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001 (see page 737)															
								PID3							
UARTPCellID0, type RO, offset 0xFF0, reset 0x0000.000D (see page 738)															
								CID0							
UARTPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0 (see page 739)															
								CID1							
UARTPCellID2, type RO, offset 0xFF8, reset 0x0000.0005 (see page 740)															
								CID2							
UARTPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1 (see page 741)															
								CID3							
Synchronous Serial Interface (SSI) SSI0 base: 0x4000.8000 SSI1 base: 0x4000.9000															
SSICR0, type R/W, offset 0x000, reset 0x0000.0000 (see page 757)															
SCR								SPH	SPO	FRF		DSS			
SSICR1, type R/W, offset 0x004, reset 0x0000.0000 (see page 759)															
											EOT	SOD	MS	SSE	LBM
SSIDR, type R/W, offset 0x008, reset 0x0000.0000 (see page 761)															
DATA															
SSISR, type RO, offset 0x00C, reset 0x0000.0003 (see page 762)															
											BSY	RFF	RNE	TNF	TFE
SSICPSR, type R/W, offset 0x010, reset 0x0000.0000 (see page 764)															
								CPSDVSR							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSIIM, type R/W, offset 0x014, reset 0x0000.0000 (see page 765)															
												TXIM	RXIM	RTIM	RORIM
SSIRIS, type RO, offset 0x018, reset 0x0000.0008 (see page 766)															
												TXRIS	RXRIS	RTRIS	RORRIS
SSIMIS, type RO, offset 0x01C, reset 0x0000.0000 (see page 768)															
												TXMIS	RXMIS	RTMIS	RORMIS
SSIICR, type W1C, offset 0x020, reset 0x0000.0000 (see page 770)															
														RTIC	RORIC
SSIDMACTL, type R/W, offset 0x024, reset 0x0000.0000 (see page 771)															
														TXDMAE	RXDMAE
SSIPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000 (see page 772)															
								PID4							
SSIPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000 (see page 773)															
								PID5							
SSIPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000 (see page 774)															
								PID6							
SSIPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000 (see page 775)															
								PID7							
SSIPeriphID0, type RO, offset 0xFE0, reset 0x0000.0022 (see page 776)															
								PID0							
SSIPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000 (see page 777)															
								PID1							
SSIPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018 (see page 778)															
								PID2							
SSIPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001 (see page 779)															
								PID3							
SSIPCellID0, type RO, offset 0xFF0, reset 0x0000.000D (see page 780)															
								CID0							
SSIPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0 (see page 781)															
								CID1							
SSIPCellID2, type RO, offset 0xFF8, reset 0x0000.0005 (see page 782)															
								CID2							
SSIPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1 (see page 783)															
								CID3							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Inter-Integrated Circuit (I^2C) Interface **I^2C Master** I2C 0 base: 0x4002.0000 I2C 1 base: 0x4002.1000															
I2CMSA, type R/W, offset 0x000, reset 0x0000.0000															
								SA							R/S
I2CMCS, type RO, offset 0x004, reset 0x0000.0020 (Read-Only Status Register)															
									BUSBSY	IDLE	ARBLST	DATACK	ADRACK	ERROR	BUSY
I2CMCS, type WO, offset 0x004, reset 0x0000.0020 (Write-Only Control Register)															
												ACK	STOP	START	RUN
I2CMDR, type R/W, offset 0x008, reset 0x0000.0000															
								DATA							
I2CMTPR, type R/W, offset 0x00C, reset 0x0000.0001															
									TPR						
I2CMIMR, type R/W, offset 0x010, reset 0x0000.0000															
															IM
I2CMRIS, type RO, offset 0x014, reset 0x0000.0000															
															RIS
I2CMMIS, type RO, offset 0x018, reset 0x0000.0000															
															MIS
I2CMICR, type WO, offset 0x01C, reset 0x0000.0000															
															IC
I2CMCR, type R/W, offset 0x020, reset 0x0000.0000															
										SFE	MFE				LPBK
Inter-Integrated Circuit (I^2C) Interface **I^2C Slave** I2C 0 base: 0x4002.0000 I2C 1 base: 0x4002.1000															
I2CSOAR, type R/W, offset 0x800, reset 0x0000.0000															
									OAR						
I2CSCSR, type RO, offset 0x804, reset 0x0000.0000 (Read-Only Status Register)															
													FBR	TREQ	RREQ
I2CSCSR, type WO, offset 0x804, reset 0x0000.0000 (Write-Only Control Register)															
															DA
I2CSDR, type R/W, offset 0x808, reset 0x0000.0000															
								DATA							
I2CSIMR, type R/W, offset 0x80C, reset 0x0000.0000															
													STOPIM	STARTIM	DATAIM

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2CSRIS, type RO, offset 0x810, reset 0x0000.0000															
													STOPRIS	STARTRIS	DATARIS
I2CSMIS, type RO, offset 0x814, reset 0x0000.0000															
													STOPMIS	STARTMIS	DATAMIS
I2CSICR, type WO, offset 0x818, reset 0x0000.0000															
													STOPIC	STARTIC	DATAIC

Inter-Integrated Circuit Sound (I^2S) Interface

Base 0x4005.4000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2STXFIFO, type WO, offset 0x000, reset 0x0000.0000 (see page 835)															
TXFIFO															
TXFIFO															
I2STXFIFOCFG, type R/W, offset 0x004, reset 0x0000.0000 (see page 836)															
														CSS	LRS
I2STXCFG, type R/W, offset 0x008, reset 0x1400.7DF0 (see page 837)															
		JST	DLY	SCP	LRP	WM		FMT	MSL						
SSZ						SDSZ									
I2STXLIMIT, type R/W, offset 0x00C, reset 0x0000.0000 (see page 839)															
											LIMIT				
I2STXISM, type R/W, offset 0x010, reset 0x0000.0000 (see page 840)															
															FFI
															FFM
I2STXLEV, type RO, offset 0x018, reset 0x0000.0000 (see page 841)															
											LEVEL				
I2SRXFIFO, type RO, offset 0x800, reset 0x0000.0000 (see page 842)															
RXFIFO															
RXFIFO															
I2SRXFIFOCFG, type R/W, offset 0x804, reset 0x0000.0000 (see page 843)															
													FMM	CSS	LRS
I2SRXCFG, type R/W, offset 0x808, reset 0x1400.7DF0 (see page 844)															
		JST	DLY	SCP	LRP		RM		MSL						
SSZ						SDSZ									
I2SRXLIMIT, type R/W, offset 0x80C, reset 0x0000.7FFF (see page 847)															
											LIMIT				
I2SRXISM, type R/W, offset 0x810, reset 0x0000.0000 (see page 848)															
															FFI
															FFM
I2SRXLEV, type RO, offset 0x818, reset 0x0000.0000 (see page 849)															
											LEVEL				
I2SCFG, type R/W, offset 0xC00, reset 0x0000.0000 (see page 850)															
										RXSLV	TXSLV			RXEN	TXEN

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2SIM, type R/W, offset 0xC10, reset 0x0000.0000 (see page 852)															
										RXREIM	RXSRIM			TXWEIM	TXSRIM
I2SRIS, type RO, offset 0xC14, reset 0x0000.0000 (see page 854)															
										RXRERIS	RXSRRIS			TXWERIS	TXSRRIS
I2SMIS, type RO, offset 0xC18, reset 0x0000.0000 (see page 856)															
										RXREMIS	RXSRMIS			TXWEMIS	TXSRMIS
I2SIC, type WO, offset 0xC1C, reset 0x0000.0000 (see page 858)															
										RXREIC				TXWEIC	
Controller Area Network (CAN) Module CAN0 base: 0x4004.0000 CAN1 base: 0x4004.1000 CAN2 base: 0x4004.2000															
CANCTL, type R/W, offset 0x000, reset 0x0000.0001 (see page 881)															
								TEST	CCE	DAR		EIE	SIE	IE	INIT
CANSTS, type R/W, offset 0x004, reset 0x0000.0000 (see page 883)															
								BOFF	EWARN	EPASS	RXOK	TXOK	LEC		
CANERR, type RO, offset 0x008, reset 0x0000.0000 (see page 886)															
RP	REC							TEC							
CANBIT, type R/W, offset 0x00C, reset 0x0000.2301 (see page 887)															
	TSEG2			TSEG1				SJW		BRP					
CANINT, type RO, offset 0x010, reset 0x0000.0000 (see page 888)															
INTID															
CANTST, type R/W, offset 0x014, reset 0x0000.0000 (see page 889)															
								RX	TX		LBACK	SILENT	BASIC		
CANBRPE, type R/W, offset 0x018, reset 0x0000.0000 (see page 891)															
												BRPE			
CANIF1CRQ, type R/W, offset 0x020, reset 0x0000.0001 (see page 892)															
BUSY								MNUM							
CANIF2CRQ, type R/W, offset 0x080, reset 0x0000.0001 (see page 892)															
BUSY								MNUM							
CANIF1CMSK, type R/W, offset 0x024, reset 0x0000.0000 (see page 893)															
								WRNRD	MASK	ARB	CONTROL	CLRINTPND	NEWDAT / TXRQST	DATAA	DATAB
CANIF2CMSK, type R/W, offset 0x084, reset 0x0000.0000 (see page 893)															
								WRNRD	MASK	ARB	CONTROL	CLRINTPND	NEWDAT / TXRQST	DATAA	DATAB
CANIF1MSK1, type R/W, offset 0x028, reset 0x0000.FFFF (see page 896)															
MSK															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CANIF2MSK1, type R/W, offset 0x088, reset 0x0000.FFFF (see page 896)															
MSK															
CANIF1MSK2, type R/W, offset 0x02C, reset 0x0000.FFFF (see page 897)															
MXTD	MDIR		MSK												
CANIF2MSK2, type R/W, offset 0x08C, reset 0x0000.FFFF (see page 897)															
MXTD	MDIR		MSK												
CANIF1ARB1, type R/W, offset 0x030, reset 0x0000.0000 (see page 899)															
ID															
CANIF2ARB1, type R/W, offset 0x090, reset 0x0000.0000 (see page 899)															
ID															
CANIF1ARB2, type R/W, offset 0x034, reset 0x0000.0000 (see page 900)															
MSGVAL	XTD	DIR	ID												
CANIF2ARB2, type R/W, offset 0x094, reset 0x0000.0000 (see page 900)															
MSGVAL	XTD	DIR	ID												
CANIF1MCTL, type R/W, offset 0x038, reset 0x0000.0000 (see page 902)															
NEWDAT	MSGLST	INTPND	UMASK	TXIE	RXIE	RMTEN	TXRQST	EOB				DLC			
CANIF2MCTL, type R/W, offset 0x098, reset 0x0000.0000 (see page 902)															
NEWDAT	MSGLST	INTPND	UMASK	TXIE	RXIE	RMTEN	TXRQST	EOB				DLC			
CANIF1DA1, type R/W, offset 0x03C, reset 0x0000.0000 (see page 905)															
DATA															
CANIF1DA2, type R/W, offset 0x040, reset 0x0000.0000 (see page 905)															
DATA															
CANIF1DB1, type R/W, offset 0x044, reset 0x0000.0000 (see page 905)															
DATA															
CANIF1DB2, type R/W, offset 0x048, reset 0x0000.0000 (see page 905)															
DATA															
CANIF2DA1, type R/W, offset 0x09C, reset 0x0000.0000 (see page 905)															
DATA															
CANIF2DA2, type R/W, offset 0x0A0, reset 0x0000.0000 (see page 905)															
DATA															
CANIF2DB1, type R/W, offset 0x0A4, reset 0x0000.0000 (see page 905)															
DATA															
CANIF2DB2, type R/W, offset 0x0A8, reset 0x0000.0000 (see page 905)															
DATA															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CANTXRQ1, type RO, offset 0x100, reset 0x0000.0000 (see page 906)															
TXRQST															
CANTXRQ2, type RO, offset 0x104, reset 0x0000.0000 (see page 906)															
TXRQST															
CANNWDA1, type RO, offset 0x120, reset 0x0000.0000 (see page 907)															
NEWDAT															
CANNWDA2, type RO, offset 0x124, reset 0x0000.0000 (see page 907)															
NEWDAT															
CANMSG1INT, type RO, offset 0x140, reset 0x0000.0000 (see page 908)															
INTPND															
CANMSG2INT, type RO, offset 0x144, reset 0x0000.0000 (see page 908)															
INTPND															
CANMSG1VAL, type RO, offset 0x160, reset 0x0000.0000 (see page 909)															
MSGVAL															
CANMSG2VAL, type RO, offset 0x164, reset 0x0000.0000 (see page 909)															
MSGVAL															
Ethernet Controller **Ethernet MAC (Ethernet Offset)** Base 0x4004.8000															
MACRIS/MACIACK, type R/W1C, offset 0x000, reset 0x0000.0000															
									PHYINT	MDINT	RXER	FOV	TXEMP	TXER	RXINT
MACIM, type R/W, offset 0x004, reset 0x0000.007F															
									PHYINTM	MDINTM	RXERM	FOVM	TXEMPM	TXERM	RXINTM
MACRCTL, type R/W, offset 0x008, reset 0x0000.0008															
											RSTFIFO	BADCRC	PRMS	AMUL	RXEN
MACTCTL, type R/W, offset 0x00C, reset 0x0000.0000															
											DUPLEX		CRC	PADEN	TXEN
MACDATA, type RO, offset 0x010, reset 0x0000.0000 (Reads)															
RXDATA															
RXDATA															
MACDATA, type WO, offset 0x010, reset 0x0000.0000 (Writes)															
TXDATA															
TXDATA															
MACIA0, type R/W, offset 0x014, reset 0x0000.0000															
MACOCT4								MACOCT3							
MACOCT2								MACOCT1							
MACIA1, type R/W, offset 0x018, reset 0x0000.0000															
MACOCT6								MACOCT5							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACTHR, type R/W, offset 0x01C, reset 0x0000.003F															
										THRESH					
MACMCTL, type R/W, offset 0x020, reset 0x0000.0000															
								REGADR						WRITE	START
MACMDV, type R/W, offset 0x024, reset 0x0000.0080															
								DIV							
MACMTXD, type R/W, offset 0x02C, reset 0x0000.0000															
MDTX															
MACMRXD, type R/W, offset 0x030, reset 0x0000.0000															
MDRX															
MACNP, type RO, offset 0x034, reset 0x0000.0000															
										NPR					
MACTR, type R/W, offset 0x038, reset 0x0000.0000															
															NEWTX
MACLED, type R/W, offset 0x040, reset 0x0000.0100															
				LED1								LED0			
MDIX, type R/W, offset 0x044, reset 0x0000.0000															
															EN
Ethernet Controller **MII Management (Accessed through the MACMCTL register)**															
MR0, type R/W, address 0x00, reset 0x1000															
RESET	LOOPBK	SPEEDSL	ANEGEN	PWRDN	ISO	RANEG	DUPLEX	COLT							
MR1, type RO, address 0x01, reset 0x7809															
	100X_F	100X_H	10T_F	10T_H						ANEGC	RFAULT	ANEGA	LINK	JAB	EXTD
MR2, type RO, address 0x02, reset 0x0161															
OUI[21:6]															
MR3, type RO, address 0x03, reset 0xB410															
OUI[5:0]						MN						RN			
MR4, type R/W, address 0x04, reset 0x01E1															
NP		RF					A3	A2	A1	A0	S				
MR5, type RO, address 0x05, reset 0x0001															
NP	ACK	RF	A								S				
MR6, type RO, address 0x06, reset 0x0000															
											PDF	LPNPA		PRX	LPANEGA
MR16, type RO, address 0x10, reset 0x0040															
						SR									
MR17, type R/W, address 0x11, reset 0x0002															
	FASTRIP	EDPD		LSQE			FASTEST						FGLS	ENON	
MR27, type RO, address 0x1B, reset -															
											XPOL				
MR29, type RO, address 0x1D, reset 0x0000															
								EONIS	ANCOMPIS	RFLTIS	LDIS	LPACKIS	PDFIS	PRXIS	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR30, type R/W, address 0x1E, reset 0x0000															
								EONIM	ANCOMPIM	RFLTIM	LDIM	LPACKIM	PDFIM	PRXIM	
MR31, type R/W, address 0x1F, reset 0x0040															
			AUTODONE								SPEED				SCRDIS
Universal Serial Bus (USB) Controller Base 0x4005.0000															
USBFADDR, type R/W, offset 0x000, reset 0x00 (see page 997)															
									FUNCADDR						
USBPOWER, type R/W, offset 0x001, reset 0x20 (OTG A / Host Mode) (see page 998)															
												RESET	RESUME	SUSPEND	PWRDNPHY
USBPOWER, type R/W, offset 0x001, reset 0x20 (OTG B / Device Mode) (see page 998)															
								ISOUP	SOFTCONN			RESET	RESUME	SUSPEND	PWRDNPHY
USBTXIS, type RO, offset 0x002, reset 0x0000 (see page 1001)															
EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
USBRXIS, type RO, offset 0x004, reset 0x0000 (see page 1003)															
EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	
USBTXIE, type R/W, offset 0x006, reset 0xFFFF (see page 1005)															
EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
USBRXIE, type R/W, offset 0x008, reset 0xFFFE (see page 1007)															
EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	
USBIS, type RO, offset 0x00A, reset 0x00 (OTG A / Host Mode) (see page 1009)															
								VBUSERR	SESREQ	DISCON	CONN	SOF	BABBLE	RESUME	
USBIS, type RO, offset 0x00A, reset 0x00 (OTG B / Device Mode) (see page 1009)															
										DISCON		SOF	RESET	RESUME	SUSPEND
USBIE, type R/W, offset 0x00B, reset 0x06 (OTG A / Host Mode) (see page 1012)															
								VBUSERR	SESREQ	DISCON	CONN	SOF	BABBLE	RESUME	
USBIE, type R/W, offset 0x00B, reset 0x06 (OTG B / Device Mode) (see page 1012)															
										DISCON		SOF	RESET	RESUME	SUSPEND
USBFRAME, type RO, offset 0x00C, reset 0x0000 (see page 1015)															
										FRAME					
USBEPIDX, type R/W, offset 0x00E, reset 0x00 (see page 1016)															
													EPIDX		
USBTEST, type R/W, offset 0x00F, reset 0x00 (OTG A / Host Mode) (see page 1017)															
								FORCEH	FIFOACC	FORCEFS					
USBTEST, type R/W, offset 0x00F, reset 0x00 (OTG B / Device Mode) (see page 1017)															
									FIFOACC	FORCEFS					
USBFIFO0, type R/W, offset 0x020, reset 0x0000.0000 (see page 1019)															
								EPDATA							
								EPDATA							
USBFIFO1, type R/W, offset 0x024, reset 0x0000.0000 (see page 1019)															
								EPDATA							
								EPDATA							
USBFIFO2, type R/W, offset 0x028, reset 0x0000.0000 (see page 1019)															
								EPDATA							
								EPDATA							
USBFIFO3, type R/W, offset 0x02C, reset 0x0000.0000 (see page 1019)															
								EPDATA							
								EPDATA							
USBFIFO4, type R/W, offset 0x030, reset 0x0000.0000 (see page 1019)															
								EPDATA							
								EPDATA							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBFIFO5, type R/W, offset 0x034, reset 0x0000.0000 (see page 1019)															
EPDATA															
EPDATA															
USBFIFO6, type R/W, offset 0x038, reset 0x0000.0000 (see page 1019)															
EPDATA															
EPDATA															
USBFIFO7, type R/W, offset 0x03C, reset 0x0000.0000 (see page 1019)															
EPDATA															
EPDATA															
USBFIFO8, type R/W, offset 0x040, reset 0x0000.0000 (see page 1019)															
EPDATA															
EPDATA															
USBFIFO9, type R/W, offset 0x044, reset 0x0000.0000 (see page 1019)															
EPDATA															
EPDATA															
USBFIFO10, type R/W, offset 0x048, reset 0x0000.0000 (see page 1019)															
EPDATA															
EPDATA															
USBFIFO11, type R/W, offset 0x04C, reset 0x0000.0000 (see page 1019)															
EPDATA															
EPDATA															
USBFIFO12, type R/W, offset 0x050, reset 0x0000.0000 (see page 1019)															
EPDATA															
EPDATA															
USBFIFO13, type R/W, offset 0x054, reset 0x0000.0000 (see page 1019)															
EPDATA															
EPDATA															
USBFIFO14, type R/W, offset 0x058, reset 0x0000.0000 (see page 1019)															
EPDATA															
EPDATA															
USBFIFO15, type R/W, offset 0x05C, reset 0x0000.0000 (see page 1019)															
EPDATA															
EPDATA															
USBDEVCTL, type R/W, offset 0x060, reset 0x80 (see page 1021)															
								DEV	FSDEV	LSDEV	VBUS		HOST	HOSTREQ	SESSION
USBTXFIFOSZ, type R/W, offset 0x062, reset 0x00 (see page 1023)															
											DPB	SIZE			
USBRXFIFOSZ, type R/W, offset 0x063, reset 0x00 (see page 1023)															
											DPB	SIZE			
USBTXFIFOADD, type R/W, offset 0x064, reset 0x0000 (see page 1024)															
							ADDR								
USBRXFIFOADD, type R/W, offset 0x066, reset 0x0000 (see page 1024)															
							ADDR								
USBCONTIM, type R/W, offset 0x07A, reset 0x5C (see page 1025)															
								WTCON				WTID			
USBVPLEN, type R/W, offset 0x07B, reset 0x3C (see page 1026)															
								VPLEN							
USBFSEOF, type R/W, offset 0x07D, reset 0x77 (see page 1027)															
								FSEOFG							
USBLSEOF, type R/W, offset 0x07E, reset 0x72 (see page 1028)															
								LSEOFG							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBTXFUNCADDR0, type R/W, offset 0x080, reset 0x00 (see page 1029)															
									ADDR						
USBTXFUNCADDR1, type R/W, offset 0x088, reset 0x00 (see page 1029)															
									ADDR						
USBTXFUNCADDR2, type R/W, offset 0x090, reset 0x00 (see page 1029)															
									ADDR						
USBTXFUNCADDR3, type R/W, offset 0x098, reset 0x00 (see page 1029)															
									ADDR						
USBTXFUNCADDR4, type R/W, offset 0x0A0, reset 0x00 (see page 1029)															
									ADDR						
USBTXFUNCADDR5, type R/W, offset 0x0A8, reset 0x00 (see page 1029)															
									ADDR						
USBTXFUNCADDR6, type R/W, offset 0x0B0, reset 0x00 (see page 1029)															
									ADDR						
USBTXFUNCADDR7, type R/W, offset 0x0B8, reset 0x00 (see page 1029)															
									ADDR						
USBTXFUNCADDR8, type R/W, offset 0x0C0, reset 0x00 (see page 1029)															
									ADDR						
USBTXFUNCADDR9, type R/W, offset 0x0C8, reset 0x00 (see page 1029)															
									ADDR						
USBTXFUNCADDR10, type R/W, offset 0x0D0, reset 0x00 (see page 1029)															
									ADDR						
USBTXFUNCADDR11, type R/W, offset 0x0D8, reset 0x00 (see page 1029)															
									ADDR						
USBTXFUNCADDR12, type R/W, offset 0x0E0, reset 0x00 (see page 1029)															
									ADDR						
USBTXFUNCADDR13, type R/W, offset 0x0E8, reset 0x00 (see page 1029)															
									ADDR						
USBTXFUNCADDR14, type R/W, offset 0x0F0, reset 0x00 (see page 1029)															
									ADDR						
USBTXFUNCADDR15, type R/W, offset 0x0F8, reset 0x00 (see page 1029)															
									ADDR						
USBTXHUBADDR0, type R/W, offset 0x082, reset 0x00 (see page 1031)															
									ADDR						
USBTXHUBADDR1, type R/W, offset 0x08A, reset 0x00 (see page 1031)															
									ADDR						
USBTXHUBADDR2, type R/W, offset 0x092, reset 0x00 (see page 1031)															
									ADDR						
USBTXHUBADDR3, type R/W, offset 0x09A, reset 0x00 (see page 1031)															
									ADDR						
USBTXHUBADDR4, type R/W, offset 0x0A2, reset 0x00 (see page 1031)															
									ADDR						
USBTXHUBADDR5, type R/W, offset 0x0AA, reset 0x00 (see page 1031)															
									ADDR						
USBTXHUBADDR6, type R/W, offset 0x0B2, reset 0x00 (see page 1031)															
									ADDR						
USBTXHUBADDR7, type R/W, offset 0x0BA, reset 0x00 (see page 1031)															
									ADDR						
USBTXHUBADDR8, type R/W, offset 0x0C2, reset 0x00 (see page 1031)															
									ADDR						
USBTXHUBADDR9, type R/W, offset 0x0CA, reset 0x00 (see page 1031)															
									ADDR						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBTXHUBADDR10, type R/W, offset 0x0D2, reset 0x00 (see page 1031)															
									ADDR						
USBTXHUBADDR11, type R/W, offset 0x0DA, reset 0x00 (see page 1031)															
									ADDR						
USBTXHUBADDR12, type R/W, offset 0x0E2, reset 0x00 (see page 1031)															
									ADDR						
USBTXHUBADDR13, type R/W, offset 0x0EA, reset 0x00 (see page 1031)															
									ADDR						
USBTXHUBADDR14, type R/W, offset 0x0F2, reset 0x00 (see page 1031)															
									ADDR						
USBTXHUBADDR15, type R/W, offset 0x0FA, reset 0x00 (see page 1031)															
									ADDR						
USBTXHUBPORT0, type R/W, offset 0x083, reset 0x00 (see page 1033)															
									PORT						
USBTXHUBPORT1, type R/W, offset 0x08B, reset 0x00 (see page 1033)															
									PORT						
USBTXHUBPORT2, type R/W, offset 0x093, reset 0x00 (see page 1033)															
									PORT						
USBTXHUBPORT3, type R/W, offset 0x09B, reset 0x00 (see page 1033)															
									PORT						
USBTXHUBPORT4, type R/W, offset 0x0A3, reset 0x00 (see page 1033)															
									PORT						
USBTXHUBPORT5, type R/W, offset 0x0AB, reset 0x00 (see page 1033)															
									PORT						
USBTXHUBPORT6, type R/W, offset 0x0B3, reset 0x00 (see page 1033)															
									PORT						
USBTXHUBPORT7, type R/W, offset 0x0BB, reset 0x00 (see page 1033)															
									PORT						
USBTXHUBPORT8, type R/W, offset 0x0C3, reset 0x00 (see page 1033)															
									PORT						
USBTXHUBPORT9, type R/W, offset 0x0CB, reset 0x00 (see page 1033)															
									PORT						
USBTXHUBPORT10, type R/W, offset 0x0D3, reset 0x00 (see page 1033)															
									PORT						
USBTXHUBPORT11, type R/W, offset 0x0DB, reset 0x00 (see page 1033)															
									PORT						
USBTXHUBPORT12, type R/W, offset 0x0E3, reset 0x00 (see page 1033)															
									PORT						
USBTXHUBPORT13, type R/W, offset 0x0EB, reset 0x00 (see page 1033)															
									PORT						
USBTXHUBPORT14, type R/W, offset 0x0F3, reset 0x00 (see page 1033)															
									PORT						
USBTXHUBPORT15, type R/W, offset 0x0FB, reset 0x00 (see page 1033)															
									PORT						
USBRXFUNCADDR1, type R/W, offset 0x08C, reset 0x00 (see page 1035)															
									ADDR						
USBRXFUNCADDR2, type R/W, offset 0x094, reset 0x00 (see page 1035)															
									ADDR						
USBRXFUNCADDR3, type R/W, offset 0x09C, reset 0x00 (see page 1035)															
									ADDR						
USBRXFUNCADDR4, type R/W, offset 0x0A4, reset 0x00 (see page 1035)															
									ADDR						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBRXFUNCADDR5, type R/W, offset 0x0AC, reset 0x00 (see page 1035)															
									ADDR						
USBRXFUNCADDR6, type R/W, offset 0x0B4, reset 0x00 (see page 1035)															
									ADDR						
USBRXFUNCADDR7, type R/W, offset 0x0BC, reset 0x00 (see page 1035)															
									ADDR						
USBRXFUNCADDR8, type R/W, offset 0x0C4, reset 0x00 (see page 1035)															
									ADDR						
USBRXFUNCADDR9, type R/W, offset 0x0CC, reset 0x00 (see page 1035)															
									ADDR						
USBRXFUNCADDR10, type R/W, offset 0x0D4, reset 0x00 (see page 1035)															
									ADDR						
USBRXFUNCADDR11, type R/W, offset 0x0DC, reset 0x00 (see page 1035)															
									ADDR						
USBRXFUNCADDR12, type R/W, offset 0x0E4, reset 0x00 (see page 1035)															
									ADDR						
USBRXFUNCADDR13, type R/W, offset 0x0EC, reset 0x00 (see page 1035)															
									ADDR						
USBRXFUNCADDR14, type R/W, offset 0x0F4, reset 0x00 (see page 1035)															
									ADDR						
USBRXFUNCADDR15, type R/W, offset 0x0FC, reset 0x00 (see page 1035)															
									ADDR						
USBRXHUBADDR1, type R/W, offset 0x08E, reset 0x00 (see page 1037)															
									ADDR						
USBRXHUBADDR2, type R/W, offset 0x096, reset 0x00 (see page 1037)															
									ADDR						
USBRXHUBADDR3, type R/W, offset 0x09E, reset 0x00 (see page 1037)															
									ADDR						
USBRXHUBADDR4, type R/W, offset 0x0A6, reset 0x00 (see page 1037)															
									ADDR						
USBRXHUBADDR5, type R/W, offset 0x0AE, reset 0x00 (see page 1037)															
									ADDR						
USBRXHUBADDR6, type R/W, offset 0x0B6, reset 0x00 (see page 1037)															
									ADDR						
USBRXHUBADDR7, type R/W, offset 0x0BE, reset 0x00 (see page 1037)															
									ADDR						
USBRXHUBADDR8, type R/W, offset 0x0C6, reset 0x00 (see page 1037)															
									ADDR						
USBRXHUBADDR9, type R/W, offset 0x0CE, reset 0x00 (see page 1037)															
									ADDR						
USBRXHUBADDR10, type R/W, offset 0x0D6, reset 0x00 (see page 1037)															
									ADDR						
USBRXHUBADDR11, type R/W, offset 0x0DE, reset 0x00 (see page 1037)															
									ADDR						
USBRXHUBADDR12, type R/W, offset 0x0E6, reset 0x00 (see page 1037)															
									ADDR						
USBRXHUBADDR13, type R/W, offset 0x0EE, reset 0x00 (see page 1037)															
									ADDR						
USBRXHUBADDR14, type R/W, offset 0x0F6, reset 0x00 (see page 1037)															
									ADDR						
USBRXHUBADDR15, type R/W, offset 0x0FE, reset 0x00 (see page 1037)															
									ADDR						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBRXHUBPORT1, type R/W, offset 0x08F, reset 0x00 (see page 1039)															
									PORT						
USBRXHUBPORT2, type R/W, offset 0x097, reset 0x00 (see page 1039)															
									PORT						
USBRXHUBPORT3, type R/W, offset 0x09F, reset 0x00 (see page 1039)															
									PORT						
USBRXHUBPORT4, type R/W, offset 0x0A7, reset 0x00 (see page 1039)															
									PORT						
USBRXHUBPORT5, type R/W, offset 0x0AF, reset 0x00 (see page 1039)															
									PORT						
USBRXHUBPORT6, type R/W, offset 0x0B7, reset 0x00 (see page 1039)															
									PORT						
USBRXHUBPORT7, type R/W, offset 0x0BF, reset 0x00 (see page 1039)															
									PORT						
USBRXHUBPORT8, type R/W, offset 0x0C7, reset 0x00 (see page 1039)															
									PORT						
USBRXHUBPORT9, type R/W, offset 0x0CF, reset 0x00 (see page 1039)															
									PORT						
USBRXHUBPORT10, type R/W, offset 0x0D7, reset 0x00 (see page 1039)															
									PORT						
USBRXHUBPORT11, type R/W, offset 0x0DF, reset 0x00 (see page 1039)															
									PORT						
USBRXHUBPORT12, type R/W, offset 0x0E7, reset 0x00 (see page 1039)															
									PORT						
USBRXHUBPORT13, type R/W, offset 0x0EF, reset 0x00 (see page 1039)															
									PORT						
USBRXHUBPORT14, type R/W, offset 0x0F7, reset 0x00 (see page 1039)															
									PORT						
USBRXHUBPORT15, type R/W, offset 0x0FF, reset 0x00 (see page 1039)															
									PORT						
USBTXMAXP1, type R/W, offset 0x110, reset 0x0000 (see page 1041)															
					MAXLOAD										
USBTXMAXP2, type R/W, offset 0x120, reset 0x0000 (see page 1041)															
					MAXLOAD										
USBTXMAXP3, type R/W, offset 0x130, reset 0x0000 (see page 1041)															
					MAXLOAD										
USBTXMAXP4, type R/W, offset 0x140, reset 0x0000 (see page 1041)															
					MAXLOAD										
USBTXMAXP5, type R/W, offset 0x150, reset 0x0000 (see page 1041)															
					MAXLOAD										
USBTXMAXP6, type R/W, offset 0x160, reset 0x0000 (see page 1041)															
					MAXLOAD										
USBTXMAXP7, type R/W, offset 0x170, reset 0x0000 (see page 1041)															
					MAXLOAD										
USBTXMAXP8, type R/W, offset 0x180, reset 0x0000 (see page 1041)															
					MAXLOAD										
USBTXMAXP9, type R/W, offset 0x190, reset 0x0000 (see page 1041)															
					MAXLOAD										
USBTXMAXP10, type R/W, offset 0x1A0, reset 0x0000 (see page 1041)															
					MAXLOAD										
USBTXMAXP11, type R/W, offset 0x1B0, reset 0x0000 (see page 1041)															
					MAXLOAD										

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBTXMAXP12, type R/W, offset 0x1C0, reset 0x0000 (see page 1041)															
					MAXLOAD										
USBTXMAXP13, type R/W, offset 0x1D0, reset 0x0000 (see page 1041)															
					MAXLOAD										
USBTXMAXP14, type R/W, offset 0x1E0, reset 0x0000 (see page 1041)															
					MAXLOAD										
USBTXMAXP15, type R/W, offset 0x1F0, reset 0x0000 (see page 1041)															
					MAXLOAD										
USBCSRL0, type W1C, offset 0x102, reset 0x00 (OTG A / Host Mode) (see page 1043)															
								NAKTO	STATUS	REQPKT	ERROR	SETUP	STALLED	TXRDY	RXRDY
USBCSRL0, type W1C, offset 0x102, reset 0x00 (OTG B / Device Mode) (see page 1043)															
								SETENDC	RXRDYC	STALL	SETEND	DATAEND	STALLED	TXRDY	RXRDY
USBCSRH0, type W1C, offset 0x103, reset 0x00 (OTG A / Host Mode) (see page 1047)															
													DTWE	DT	FLUSH
USBCSRH0, type W1C, offset 0x103, reset 0x00 (OTG B / Device Mode) (see page 1047)															
															FLUSH
USBCOUNT0, type RO, offset 0x108, reset 0x00 (see page 1049)															
									COUNT						
USBTYPE0, type R/W, offset 0x10A, reset 0x00 (see page 1050)															
								SPEED							
USBNAKLMT, type R/W, offset 0x10B, reset 0x00 (see page 1051)															
											NAKLMT				
USBTXCSRL1, type R/W, offset 0x112, reset 0x00 (OTG A / Host Mode) (see page 1052)															
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
USBTXCSRL2, type R/W, offset 0x122, reset 0x00 (OTG A / Host Mode) (see page 1052)															
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
USBTXCSRL3, type R/W, offset 0x132, reset 0x00 (OTG A / Host Mode) (see page 1052)															
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
USBTXCSRL4, type R/W, offset 0x142, reset 0x00 (OTG A / Host Mode) (see page 1052)															
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
USBTXCSRL5, type R/W, offset 0x152, reset 0x00 (OTG A / Host Mode) (see page 1052)															
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
USBTXCSRL6, type R/W, offset 0x162, reset 0x00 (OTG A / Host Mode) (see page 1052)															
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
USBTXCSRL7, type R/W, offset 0x172, reset 0x00 (OTG A / Host Mode) (see page 1052)															
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
USBTXCSRL8, type R/W, offset 0x182, reset 0x00 (OTG A / Host Mode) (see page 1052)															
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
USBTXCSRL9, type R/W, offset 0x192, reset 0x00 (OTG A / Host Mode) (see page 1052)															
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
USBTXCSRL10, type R/W, offset 0x1A2, reset 0x00 (OTG A / Host Mode) (see page 1052)															
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
USBTXCSRL11, type R/W, offset 0x1B2, reset 0x00 (OTG A / Host Mode) (see page 1052)															
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
USBTXCSRL12, type R/W, offset 0x1C2, reset 0x00 (OTG A / Host Mode) (see page 1052)															
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
USBTXCSRL13, type R/W, offset 0x1D2, reset 0x00 (OTG A / Host Mode) (see page 1052)															
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
USBTXCSRL14, type R/W, offset 0x1E2, reset 0x00 (OTG A / Host Mode) (see page 1052)															
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
USBTXCSRL15, type R/W, offset 0x1F2, reset 0x00 (OTG A / Host Mode) (see page 1052)															
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBTXCSRL1, type R/W, offset 0x112, reset 0x00 (OTG B / Device Mode) (see page 1052)															
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
USBTXCSRL2, type R/W, offset 0x122, reset 0x00 (OTG B / Device Mode) (see page 1052)															
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
USBTXCSRL3, type R/W, offset 0x132, reset 0x00 (OTG B / Device Mode) (see page 1052)															
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
USBTXCSRL4, type R/W, offset 0x142, reset 0x00 (OTG B / Device Mode) (see page 1052)															
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
USBTXCSRL5, type R/W, offset 0x152, reset 0x00 (OTG B / Device Mode) (see page 1052)															
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
USBTXCSRL6, type R/W, offset 0x162, reset 0x00 (OTG B / Device Mode) (see page 1052)															
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
USBTXCSRL7, type R/W, offset 0x172, reset 0x00 (OTG B / Device Mode) (see page 1052)															
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
USBTXCSRL8, type R/W, offset 0x182, reset 0x00 (OTG B / Device Mode) (see page 1052)															
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
USBTXCSRL9, type R/W, offset 0x192, reset 0x00 (OTG B / Device Mode) (see page 1052)															
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
USBTXCSRL10, type R/W, offset 0x1A2, reset 0x00 (OTG B / Device Mode) (see page 1052)															
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
USBTXCSRL11, type R/W, offset 0x1B2, reset 0x00 (OTG B / Device Mode) (see page 1052)															
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
USBTXCSRL12, type R/W, offset 0x1C2, reset 0x00 (OTG B / Device Mode) (see page 1052)															
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
USBTXCSRL13, type R/W, offset 0x1D2, reset 0x00 (OTG B / Device Mode) (see page 1052)															
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
USBTXCSRL14, type R/W, offset 0x1E2, reset 0x00 (OTG B / Device Mode) (see page 1052)															
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
USBTXCSRL15, type R/W, offset 0x1F2, reset 0x00 (OTG B / Device Mode) (see page 1052)															
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
USBTXCSRH1, type R/W, offset 0x113, reset 0x00 (OTG A / Host Mode) (see page 1057)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH2, type R/W, offset 0x123, reset 0x00 (OTG A / Host Mode) (see page 1057)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH3, type R/W, offset 0x133, reset 0x00 (OTG A / Host Mode) (see page 1057)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH4, type R/W, offset 0x143, reset 0x00 (OTG A / Host Mode) (see page 1057)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH5, type R/W, offset 0x153, reset 0x00 (OTG A / Host Mode) (see page 1057)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH6, type R/W, offset 0x163, reset 0x00 (OTG A / Host Mode) (see page 1057)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH7, type R/W, offset 0x173, reset 0x00 (OTG A / Host Mode) (see page 1057)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH8, type R/W, offset 0x183, reset 0x00 (OTG A / Host Mode) (see page 1057)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH9, type R/W, offset 0x193, reset 0x00 (OTG A / Host Mode) (see page 1057)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH10, type R/W, offset 0x1A3, reset 0x00 (OTG A / Host Mode) (see page 1057)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH11, type R/W, offset 0x1B3, reset 0x00 (OTG A / Host Mode) (see page 1057)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBTXCSRH12, type R/W, offset 0x1C3, reset 0x00 (OTG A / Host Mode) (see page 1057)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH13, type R/W, offset 0x1D3, reset 0x00 (OTG A / Host Mode) (see page 1057)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH14, type R/W, offset 0x1E3, reset 0x00 (OTG A / Host Mode) (see page 1057)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH15, type R/W, offset 0x1F3, reset 0x00 (OTG A / Host Mode) (see page 1057)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH1, type R/W, offset 0x113, reset 0x00 (OTG B / Device Mode) (see page 1057)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH2, type R/W, offset 0x123, reset 0x00 (OTG B / Device Mode) (see page 1057)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH3, type R/W, offset 0x133, reset 0x00 (OTG B / Device Mode) (see page 1057)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH4, type R/W, offset 0x143, reset 0x00 (OTG B / Device Mode) (see page 1057)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH5, type R/W, offset 0x153, reset 0x00 (OTG B / Device Mode) (see page 1057)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH6, type R/W, offset 0x163, reset 0x00 (OTG B / Device Mode) (see page 1057)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH7, type R/W, offset 0x173, reset 0x00 (OTG B / Device Mode) (see page 1057)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH8, type R/W, offset 0x183, reset 0x00 (OTG B / Device Mode) (see page 1057)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH9, type R/W, offset 0x193, reset 0x00 (OTG B / Device Mode) (see page 1057)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH10, type R/W, offset 0x1A3, reset 0x00 (OTG B / Device Mode) (see page 1057)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH11, type R/W, offset 0x1B3, reset 0x00 (OTG B / Device Mode) (see page 1057)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH12, type R/W, offset 0x1C3, reset 0x00 (OTG B / Device Mode) (see page 1057)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH13, type R/W, offset 0x1D3, reset 0x00 (OTG B / Device Mode) (see page 1057)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH14, type R/W, offset 0x1E3, reset 0x00 (OTG B / Device Mode) (see page 1057)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH15, type R/W, offset 0x1F3, reset 0x00 (OTG B / Device Mode) (see page 1057)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBRXMAXP1, type R/W, offset 0x114, reset 0x0000 (see page 1061)															
					MAXLOAD										
USBRXMAXP2, type R/W, offset 0x124, reset 0x0000 (see page 1061)															
					MAXLOAD										
USBRXMAXP3, type R/W, offset 0x134, reset 0x0000 (see page 1061)															
					MAXLOAD										
USBRXMAXP4, type R/W, offset 0x144, reset 0x0000 (see page 1061)															
					MAXLOAD										
USBRXMAXP5, type R/W, offset 0x154, reset 0x0000 (see page 1061)															
					MAXLOAD										
USBRXMAXP6, type R/W, offset 0x164, reset 0x0000 (see page 1061)															
					MAXLOAD										
USBRXMAXP7, type R/W, offset 0x174, reset 0x0000 (see page 1061)															
					MAXLOAD										

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBRXMAXP8, type R/W, offset 0x184, reset 0x0000 (see page 1061)															
					MAXLOAD										
USBRXMAXP9, type R/W, offset 0x194, reset 0x0000 (see page 1061)															
					MAXLOAD										
USBRXMAXP10, type R/W, offset 0x1A4, reset 0x0000 (see page 1061)															
					MAXLOAD										
USBRXMAXP11, type R/W, offset 0x1B4, reset 0x0000 (see page 1061)															
					MAXLOAD										
USBRXMAXP12, type R/W, offset 0x1C4, reset 0x0000 (see page 1061)															
					MAXLOAD										
USBRXMAXP13, type R/W, offset 0x1D4, reset 0x0000 (see page 1061)															
					MAXLOAD										
USBRXMAXP14, type R/W, offset 0x1E4, reset 0x0000 (see page 1061)															
					MAXLOAD										
USBRXMAXP15, type R/W, offset 0x1F4, reset 0x0000 (see page 1061)															
					MAXLOAD										
USBRXCSRL1, type R/W, offset 0x116, reset 0x00 (OTG A / Host Mode) (see page 1063)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL2, type R/W, offset 0x126, reset 0x00 (OTG A / Host Mode) (see page 1063)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL3, type R/W, offset 0x136, reset 0x00 (OTG A / Host Mode) (see page 1063)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL4, type R/W, offset 0x146, reset 0x00 (OTG A / Host Mode) (see page 1063)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL5, type R/W, offset 0x156, reset 0x00 (OTG A / Host Mode) (see page 1063)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL6, type R/W, offset 0x166, reset 0x00 (OTG A / Host Mode) (see page 1063)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL7, type R/W, offset 0x176, reset 0x00 (OTG A / Host Mode) (see page 1063)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL8, type R/W, offset 0x186, reset 0x00 (OTG A / Host Mode) (see page 1063)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL9, type R/W, offset 0x196, reset 0x00 (OTG A / Host Mode) (see page 1063)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL10, type R/W, offset 0x1A6, reset 0x00 (OTG A / Host Mode) (see page 1063)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL11, type R/W, offset 0x1B6, reset 0x00 (OTG A / Host Mode) (see page 1063)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL12, type R/W, offset 0x1C6, reset 0x00 (OTG A / Host Mode) (see page 1063)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL13, type R/W, offset 0x1D6, reset 0x00 (OTG A / Host Mode) (see page 1063)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBRXCSRL14, type R/W, offset 0x1E6, reset 0x00 (OTG A / Host Mode) (see page 1063)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL15, type R/W, offset 0x1F6, reset 0x00 (OTG A / Host Mode) (see page 1063)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL1, type R/W, offset 0x116, reset 0x00 (OTG B / Device Mode) (see page 1063)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL2, type R/W, offset 0x126, reset 0x00 (OTG B / Device Mode) (see page 1063)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL3, type R/W, offset 0x136, reset 0x00 (OTG B / Device Mode) (see page 1063)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL4, type R/W, offset 0x146, reset 0x00 (OTG B / Device Mode) (see page 1063)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL5, type R/W, offset 0x156, reset 0x00 (OTG B / Device Mode) (see page 1063)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL6, type R/W, offset 0x166, reset 0x00 (OTG B / Device Mode) (see page 1063)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL7, type R/W, offset 0x176, reset 0x00 (OTG B / Device Mode) (see page 1063)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL8, type R/W, offset 0x186, reset 0x00 (OTG B / Device Mode) (see page 1063)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL9, type R/W, offset 0x196, reset 0x00 (OTG B / Device Mode) (see page 1063)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL10, type R/W, offset 0x1A6, reset 0x00 (OTG B / Device Mode) (see page 1063)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL11, type R/W, offset 0x1B6, reset 0x00 (OTG B / Device Mode) (see page 1063)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL12, type R/W, offset 0x1C6, reset 0x00 (OTG B / Device Mode) (see page 1063)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL13, type R/W, offset 0x1D6, reset 0x00 (OTG B / Device Mode) (see page 1063)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL14, type R/W, offset 0x1E6, reset 0x00 (OTG B / Device Mode) (see page 1063)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL15, type R/W, offset 0x1F6, reset 0x00 (OTG B / Device Mode) (see page 1063)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRH1, type R/W, offset 0x117, reset 0x00 (OTG A / Host Mode) (see page 1068)															
								AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	
USBRXCSRH2, type R/W, offset 0x127, reset 0x00 (OTG A / Host Mode) (see page 1068)															
								AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	
USBRXCSRH3, type R/W, offset 0x137, reset 0x00 (OTG A / Host Mode) (see page 1068)															
								AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	
USBRXCSRH4, type R/W, offset 0x147, reset 0x00 (OTG A / Host Mode) (see page 1068)															
								AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	
USBRXCSRH5, type R/W, offset 0x157, reset 0x00 (OTG A / Host Mode) (see page 1068)															
								AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	
USBRXCSRH6, type R/W, offset 0x167, reset 0x00 (OTG A / Host Mode) (see page 1068)															
								AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	
USBRXCSRH7, type R/W, offset 0x177, reset 0x00 (OTG A / Host Mode) (see page 1068)															
								AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	
USBRXCSRH8, type R/W, offset 0x187, reset 0x00 (OTG A / Host Mode) (see page 1068)															
								AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBRXCSRH9, type R/W, offset 0x197, reset 0x00 (OTG A / Host Mode) (see page 1068)															
								AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	
USBRXCSRH10, type R/W, offset 0x1A7, reset 0x00 (OTG A / Host Mode) (see page 1068)															
								AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	
USBRXCSRH11, type R/W, offset 0x1B7, reset 0x00 (OTG A / Host Mode) (see page 1068)															
								AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	
USBRXCSRH12, type R/W, offset 0x1C7, reset 0x00 (OTG A / Host Mode) (see page 1068)															
								AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	
USBRXCSRH13, type R/W, offset 0x1D7, reset 0x00 (OTG A / Host Mode) (see page 1068)															
								AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	
USBRXCSRH14, type R/W, offset 0x1E7, reset 0x00 (OTG A / Host Mode) (see page 1068)															
								AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	
USBRXCSRH15, type R/W, offset 0x1F7, reset 0x00 (OTG A / Host Mode) (see page 1068)															
								AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	
USBRXCSRH1, type R/W, offset 0x117, reset 0x00 (OTG B / Device Mode) (see page 1068)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCSRH2, type R/W, offset 0x127, reset 0x00 (OTG B / Device Mode) (see page 1068)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCSRH3, type R/W, offset 0x137, reset 0x00 (OTG B / Device Mode) (see page 1068)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCSRH4, type R/W, offset 0x147, reset 0x00 (OTG B / Device Mode) (see page 1068)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCSRH5, type R/W, offset 0x157, reset 0x00 (OTG B / Device Mode) (see page 1068)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCSRH6, type R/W, offset 0x167, reset 0x00 (OTG B / Device Mode) (see page 1068)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCSRH7, type R/W, offset 0x177, reset 0x00 (OTG B / Device Mode) (see page 1068)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCSRH8, type R/W, offset 0x187, reset 0x00 (OTG B / Device Mode) (see page 1068)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCSRH9, type R/W, offset 0x197, reset 0x00 (OTG B / Device Mode) (see page 1068)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCSRH10, type R/W, offset 0x1A7, reset 0x00 (OTG B / Device Mode) (see page 1068)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCSRH11, type R/W, offset 0x1B7, reset 0x00 (OTG B / Device Mode) (see page 1068)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCSRH12, type R/W, offset 0x1C7, reset 0x00 (OTG B / Device Mode) (see page 1068)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCSRH13, type R/W, offset 0x1D7, reset 0x00 (OTG B / Device Mode) (see page 1068)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCSRH14, type R/W, offset 0x1E7, reset 0x00 (OTG B / Device Mode) (see page 1068)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBRXCSRH15, type R/W, offset 0x1F7, reset 0x00 (OTG B / Device Mode) (see page 1068)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCOUNT1, type RO, offset 0x118, reset 0x0000 (see page 1073)															
			COUNT												
USBRXCOUNT2, type RO, offset 0x128, reset 0x0000 (see page 1073)															
			COUNT												
USBRXCOUNT3, type RO, offset 0x138, reset 0x0000 (see page 1073)															
			COUNT												
USBRXCOUNT4, type RO, offset 0x148, reset 0x0000 (see page 1073)															
			COUNT												
USBRXCOUNT5, type RO, offset 0x158, reset 0x0000 (see page 1073)															
			COUNT												
USBRXCOUNT6, type RO, offset 0x168, reset 0x0000 (see page 1073)															
			COUNT												
USBRXCOUNT7, type RO, offset 0x178, reset 0x0000 (see page 1073)															
			COUNT												
USBRXCOUNT8, type RO, offset 0x188, reset 0x0000 (see page 1073)															
			COUNT												
USBRXCOUNT9, type RO, offset 0x198, reset 0x0000 (see page 1073)															
			COUNT												
USBRXCOUNT10, type RO, offset 0x1A8, reset 0x0000 (see page 1073)															
			COUNT												
USBRXCOUNT11, type RO, offset 0x1B8, reset 0x0000 (see page 1073)															
			COUNT												
USBRXCOUNT12, type RO, offset 0x1C8, reset 0x0000 (see page 1073)															
			COUNT												
USBRXCOUNT13, type RO, offset 0x1D8, reset 0x0000 (see page 1073)															
			COUNT												
USBRXCOUNT14, type RO, offset 0x1E8, reset 0x0000 (see page 1073)															
			COUNT												
USBRXCOUNT15, type RO, offset 0x1F8, reset 0x0000 (see page 1073)															
			COUNT												
USBTXTYPE1, type R/W, offset 0x11A, reset 0x00 (see page 1075)															
								SPEED		PROTO		TEP			
USBTXTYPE2, type R/W, offset 0x12A, reset 0x00 (see page 1075)															
								SPEED		PROTO		TEP			
USBTXTYPE3, type R/W, offset 0x13A, reset 0x00 (see page 1075)															
								SPEED		PROTO		TEP			
USBTXTYPE4, type R/W, offset 0x14A, reset 0x00 (see page 1075)															
								SPEED		PROTO		TEP			
USBTXTYPE5, type R/W, offset 0x15A, reset 0x00 (see page 1075)															
								SPEED		PROTO		TEP			
USBTXTYPE6, type R/W, offset 0x16A, reset 0x00 (see page 1075)															
								SPEED		PROTO		TEP			
USBTXTYPE7, type R/W, offset 0x17A, reset 0x00 (see page 1075)															
								SPEED		PROTO		TEP			
USBTXTYPE8, type R/W, offset 0x18A, reset 0x00 (see page 1075)															
								SPEED		PROTO		TEP			
USBTXTYPE9, type R/W, offset 0x19A, reset 0x00 (see page 1075)															
								SPEED		PROTO		TEP			

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBTXTYPE10, type R/W, offset 0x1AA, reset 0x00 (see page 1075)															
								SPEED		PROTO		TEP			
USBTXTYPE11, type R/W, offset 0x1BA, reset 0x00 (see page 1075)															
								SPEED		PROTO		TEP			
USBTXTYPE12, type R/W, offset 0x1CA, reset 0x00 (see page 1075)															
								SPEED		PROTO		TEP			
USBTXTYPE13, type R/W, offset 0x1DA, reset 0x00 (see page 1075)															
								SPEED		PROTO		TEP			
USBTXTYPE14, type R/W, offset 0x1EA, reset 0x00 (see page 1075)															
								SPEED		PROTO		TEP			
USBTXTYPE15, type R/W, offset 0x1FA, reset 0x00 (see page 1075)															
								SPEED		PROTO		TEP			
USBTXINTERVAL1, type R/W, offset 0x11B, reset 0x00 (see page 1077)															
								TXPOLL / NAKLMT							
USBTXINTERVAL2, type R/W, offset 0x12B, reset 0x00 (see page 1077)															
								TXPOLL / NAKLMT							
USBTXINTERVAL3, type R/W, offset 0x13B, reset 0x00 (see page 1077)															
								TXPOLL / NAKLMT							
USBTXINTERVAL4, type R/W, offset 0x14B, reset 0x00 (see page 1077)															
								TXPOLL / NAKLMT							
USBTXINTERVAL5, type R/W, offset 0x15B, reset 0x00 (see page 1077)															
								TXPOLL / NAKLMT							
USBTXINTERVAL6, type R/W, offset 0x16B, reset 0x00 (see page 1077)															
								TXPOLL / NAKLMT							
USBTXINTERVAL7, type R/W, offset 0x17B, reset 0x00 (see page 1077)															
								TXPOLL / NAKLMT							
USBTXINTERVAL8, type R/W, offset 0x18B, reset 0x00 (see page 1077)															
								TXPOLL / NAKLMT							
USBTXINTERVAL9, type R/W, offset 0x19B, reset 0x00 (see page 1077)															
								TXPOLL / NAKLMT							
USBTXINTERVAL10, type R/W, offset 0x1AB, reset 0x00 (see page 1077)															
								TXPOLL / NAKLMT							
USBTXINTERVAL11, type R/W, offset 0x1BB, reset 0x00 (see page 1077)															
								TXPOLL / NAKLMT							
USBTXINTERVAL12, type R/W, offset 0x1CB, reset 0x00 (see page 1077)															
								TXPOLL / NAKLMT							
USBTXINTERVAL13, type R/W, offset 0x1DB, reset 0x00 (see page 1077)															
								TXPOLL / NAKLMT							
USBTXINTERVAL14, type R/W, offset 0x1EB, reset 0x00 (see page 1077)															
								TXPOLL / NAKLMT							
USBTXINTERVAL15, type R/W, offset 0x1FB, reset 0x00 (see page 1077)															
								TXPOLL / NAKLMT							
USBRXTYPE1, type R/W, offset 0x11C, reset 0x00 (see page 1079)															
								SPEED		PROTO		TEP			
USBRXTYPE2, type R/W, offset 0x12C, reset 0x00 (see page 1079)															
								SPEED		PROTO		TEP			
USBRXTYPE3, type R/W, offset 0x13C, reset 0x00 (see page 1079)															
								SPEED		PROTO		TEP			
USBRXTYPE4, type R/W, offset 0x14C, reset 0x00 (see page 1079)															
								SPEED		PROTO		TEP			
USBRXTYPE5, type R/W, offset 0x15C, reset 0x00 (see page 1079)															
								SPEED		PROTO		TEP			

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBRXTYPE6, type R/W, offset 0x16C, reset 0x00 (see page 1079)															
								SPEED		PROTO		TEP			
USBRXTYPE7, type R/W, offset 0x17C, reset 0x00 (see page 1079)															
								SPEED		PROTO		TEP			
USBRXTYPE8, type R/W, offset 0x18C, reset 0x00 (see page 1079)															
								SPEED		PROTO		TEP			
USBRXTYPE9, type R/W, offset 0x19C, reset 0x00 (see page 1079)															
								SPEED		PROTO		TEP			
USBRXTYPE10, type R/W, offset 0x1AC, reset 0x00 (see page 1079)															
								SPEED		PROTO		TEP			
USBRXTYPE11, type R/W, offset 0x1BC, reset 0x00 (see page 1079)															
								SPEED		PROTO		TEP			
USBRXTYPE12, type R/W, offset 0x1CC, reset 0x00 (see page 1079)															
								SPEED		PROTO		TEP			
USBRXTYPE13, type R/W, offset 0x1DC, reset 0x00 (see page 1079)															
								SPEED		PROTO		TEP			
USBRXTYPE14, type R/W, offset 0x1EC, reset 0x00 (see page 1079)															
								SPEED		PROTO		TEP			
USBRXTYPE15, type R/W, offset 0x1FC, reset 0x00 (see page 1079)															
								SPEED		PROTO		TEP			
USBRXINTERVAL1, type R/W, offset 0x11D, reset 0x00 (see page 1081)															
								TXPOLL / NAKLMT							
USBRXINTERVAL2, type R/W, offset 0x12D, reset 0x00 (see page 1081)															
								TXPOLL / NAKLMT							
USBRXINTERVAL3, type R/W, offset 0x13D, reset 0x00 (see page 1081)															
								TXPOLL / NAKLMT							
USBRXINTERVAL4, type R/W, offset 0x14D, reset 0x00 (see page 1081)															
								TXPOLL / NAKLMT							
USBRXINTERVAL5, type R/W, offset 0x15D, reset 0x00 (see page 1081)															
								TXPOLL / NAKLMT							
USBRXINTERVAL6, type R/W, offset 0x16D, reset 0x00 (see page 1081)															
								TXPOLL / NAKLMT							
USBRXINTERVAL7, type R/W, offset 0x17D, reset 0x00 (see page 1081)															
								TXPOLL / NAKLMT							
USBRXINTERVAL8, type R/W, offset 0x18D, reset 0x00 (see page 1081)															
								TXPOLL / NAKLMT							
USBRXINTERVAL9, type R/W, offset 0x19D, reset 0x00 (see page 1081)															
								TXPOLL / NAKLMT							
USBRXINTERVAL10, type R/W, offset 0x1AD, reset 0x00 (see page 1081)															
								TXPOLL / NAKLMT							
USBRXINTERVAL11, type R/W, offset 0x1BD, reset 0x00 (see page 1081)															
								TXPOLL / NAKLMT							
USBRXINTERVAL12, type R/W, offset 0x1CD, reset 0x00 (see page 1081)															
								TXPOLL / NAKLMT							
USBRXINTERVAL13, type R/W, offset 0x1DD, reset 0x00 (see page 1081)															
								TXPOLL / NAKLMT							
USBRXINTERVAL14, type R/W, offset 0x1ED, reset 0x00 (see page 1081)															
								TXPOLL / NAKLMT							
USBRXINTERVAL15, type R/W, offset 0x1FD, reset 0x00 (see page 1081)															
								TXPOLL / NAKLMT							
USBRQPKTCOUNT1, type R/W, offset 0x304, reset 0x0000 (see page 1083)															
COUNT															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBRQPKTCOUNT2, type R/W, offset 0x308, reset 0x0000 (see page 1083)															
COUNT															
USBRQPKTCOUNT3, type R/W, offset 0x30C, reset 0x0000 (see page 1083)															
COUNT															
USBRQPKTCOUNT4, type R/W, offset 0x310, reset 0x0000 (see page 1083)															
COUNT															
USBRQPKTCOUNT5, type R/W, offset 0x314, reset 0x0000 (see page 1083)															
COUNT															
USBRQPKTCOUNT6, type R/W, offset 0x318, reset 0x0000 (see page 1083)															
COUNT															
USBRQPKTCOUNT7, type R/W, offset 0x31C, reset 0x0000 (see page 1083)															
COUNT															
USBRQPKTCOUNT8, type R/W, offset 0x320, reset 0x0000 (see page 1083)															
COUNT															
USBRQPKTCOUNT9, type R/W, offset 0x324, reset 0x0000 (see page 1083)															
COUNT															
USBRQPKTCOUNT10, type R/W, offset 0x328, reset 0x0000 (see page 1083)															
COUNT															
USBRQPKTCOUNT11, type R/W, offset 0x32C, reset 0x0000 (see page 1083)															
COUNT															
USBRQPKTCOUNT12, type R/W, offset 0x330, reset 0x0000 (see page 1083)															
COUNT															
USBRQPKTCOUNT13, type R/W, offset 0x334, reset 0x0000 (see page 1083)															
COUNT															
USBRQPKTCOUNT14, type R/W, offset 0x338, reset 0x0000 (see page 1083)															
COUNT															
USBRQPKTCOUNT15, type R/W, offset 0x33C, reset 0x0000 (see page 1083)															
COUNT															
USBRXDPKTBUFDIS, type R/W, offset 0x340, reset 0x0000 (see page 1085)															
EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	
USBTXDPKTBUFDIS, type R/W, offset 0x342, reset 0x0000 (see page 1087)															
EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	
USBEPC, type R/W, offset 0x400, reset 0x0000.0000 (see page 1089)															
						PFLTACT			PFLTAEN	PFLTSEN	PFLTEN		EPENDE	EPEN	
USBEPCRIS, type RO, offset 0x404, reset 0x0000.0000 (see page 1092)															
															PF
USBEPCIM, type R/W, offset 0x408, reset 0x0000.0000 (see page 1093)															
															PF
USBEPCISC, type R/W, offset 0x40C, reset 0x0000.0000 (see page 1094)															
															PF
USBDRRIS, type RO, offset 0x410, reset 0x0000.0000 (see page 1095)															
															RESUME
USBDRIM, type R/W, offset 0x414, reset 0x0000.0000 (see page 1096)															
															RESUME

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBDRISC, type W1C, offset 0x418, reset 0x0000.0000 (see page 1097)															
															RESUME
USBGPCS, type R/W, offset 0x41C, reset 0x0000.0001 (see page 1098)															
														DEVMODOTG	DEVMOD
USBVDC, type R/W, offset 0x430, reset 0x0000.0000 (see page 1099)															
															VBDEN
USBVDCRIS, type RO, offset 0x434, reset 0x0000.0000 (see page 1100)															
															VD
USBVDCIM, type R/W, offset 0x438, reset 0x0000.0000 (see page 1101)															
															VD
USBVDCISC, type R/W, offset 0x43C, reset 0x0000.0000 (see page 1102)															
															VD
USBIDVRIS, type RO, offset 0x444, reset 0x0000.0000 (see page 1103)															
															ID
USBIDVIM, type R/W, offset 0x448, reset 0x0000.0000 (see page 1104)															
															ID
USBIDVISC, type R/W1C, offset 0x44C, reset 0x0000.0000 (see page 1105)															
															ID
USBDMASEL, type R/W, offset 0x450, reset 0x0033.2211 (see page 1106)															
								DMACTX				DMACRX			
DMABTX				DMABRX				DMAATX				DMAARX			
Analog Comparators Base 0x4003.C000															
ACMIS, type R/W1C, offset 0x000, reset 0x0000.0000 (see page 1115)															
													IN2	IN1	IN0
ACRIS, type RO, offset 0x004, reset 0x0000.0000 (see page 1116)															
													IN2	IN1	IN0
ACINTEN, type R/W, offset 0x008, reset 0x0000.0000 (see page 1117)															
													IN2	IN1	IN0
ACREFCTL, type R/W, offset 0x010, reset 0x0000.0000 (see page 1118)															
						EN	RNG					VREF			
ACSTAT0, type RO, offset 0x020, reset 0x0000.0000 (see page 1119)															
														OVAL	
ACSTAT1, type RO, offset 0x040, reset 0x0000.0000 (see page 1119)															
														OVAL	
ACSTAT2, type RO, offset 0x060, reset 0x0000.0000 (see page 1119)															
														OVAL	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCTL0, type R/W, offset 0x024, reset 0x0000.0000 (see page 1120)															
				TOEN	ASRCP			TSLVAL	TSEN		ISLVAL	ISEN		CINV	
ACCTL1, type R/W, offset 0x044, reset 0x0000.0000 (see page 1120)															
				TOEN	ASRCP			TSLVAL	TSEN		ISLVAL	ISEN		CINV	
ACCTL2, type R/W, offset 0x064, reset 0x0000.0000 (see page 1120)															
				TOEN	ASRCP			TSLVAL	TSEN		ISLVAL	ISEN		CINV	

B Ordering and Contact Information

B.1 Ordering Information

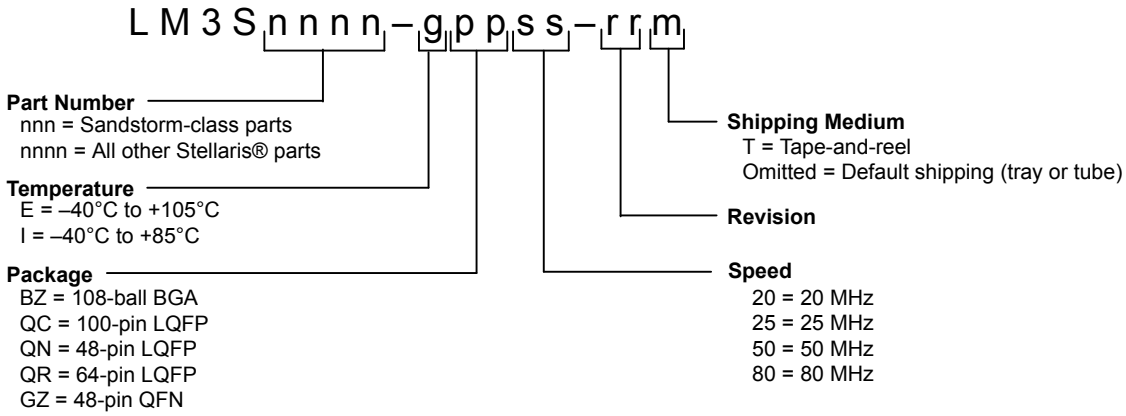


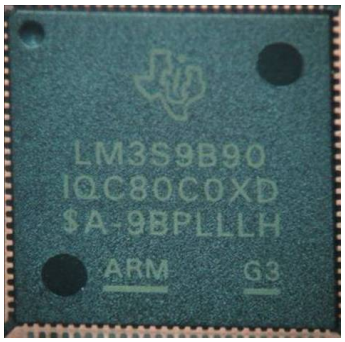
Table B-1. Part Ordering Information

Orderable Part Number	Description
LM3S9U81-IQC80-A2	Stellaris® LM3S9U81 Microcontroller Industrial Temperature 100-pin LQFP
LM3S9U81-IBZ80-A2	Stellaris LM3S9U81 Microcontroller Industrial Temperature 108-ball BGA
LM3S9U81-IQC80-A2T	Stellaris LM3S9U81 Microcontroller Industrial Temperature 100-pin LQFP Tape-and-reel
LM3S9U81-IBZ80-A2T	Stellaris LM3S9U81 Microcontroller Industrial Temperature 108-ball BGA Tape-and-reel

B.2 Part Markings

The Stellaris microcontrollers are marked with an identifying number. This code contains the following information:

- The first line indicates the part number, for example, LM3S9B90.
- In the second line, the first eight characters indicate the temperature, package, speed, revision, and product status. For example in the figure below, IQC80C0X indicates an Industrial temperature (I), 100-pin LQFP package (QC), 80-MHz (80), revision C0 (C0) device. The letter immediately following the revision indicates product status. An X indicates experimental and requires a waiver; an S indicates the part is fully qualified and released to production.
- The remaining characters contain internal tracking numbers.



B.3 Kits

The Stellaris Family provides the hardware and software tools that engineers need to begin development quickly.

- Reference Design Kits accelerate product development by providing ready-to-run hardware and comprehensive documentation including hardware design files
- Evaluation Kits provide a low-cost and effective means of evaluating Stellaris microcontrollers before purchase
- Development Kits provide you with all the tools you need to develop and prototype embedded applications right out of the box

See the website at www.ti.com/stellaris for the latest tools available, or ask your distributor.

B.4 Support Information

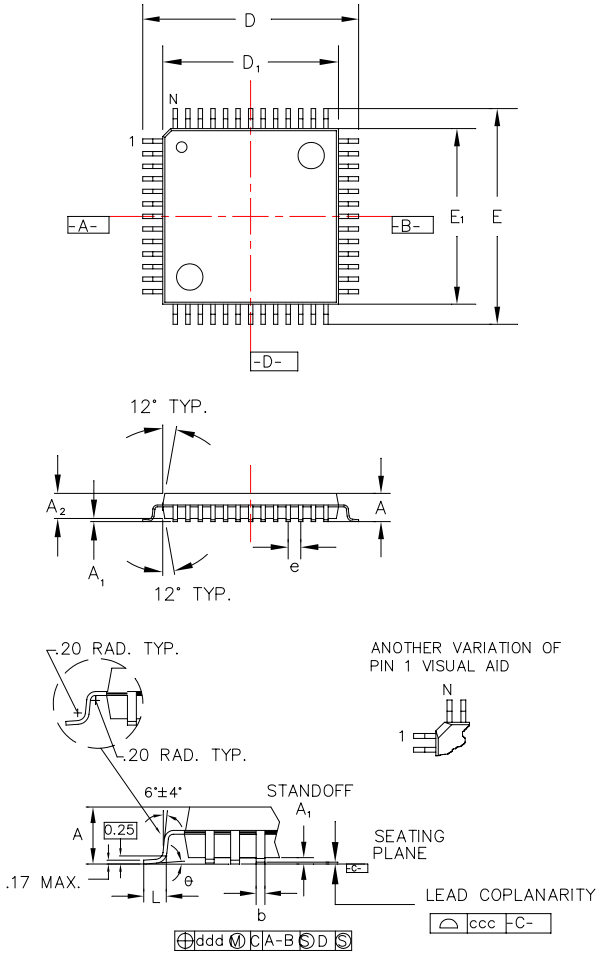
For support on Stellaris products, contact the TI Worldwide Product Information Center nearest you: http://www-k.ext.ti.com/sc/technical-support/product-information-centers.htm.

C Package Information

C.1 100-Pin LQFP Package

C.1.1 Package Dimensions

Figure C-1. Stellaris LM3S9U81 100-Pin LQFP Package Dimensions



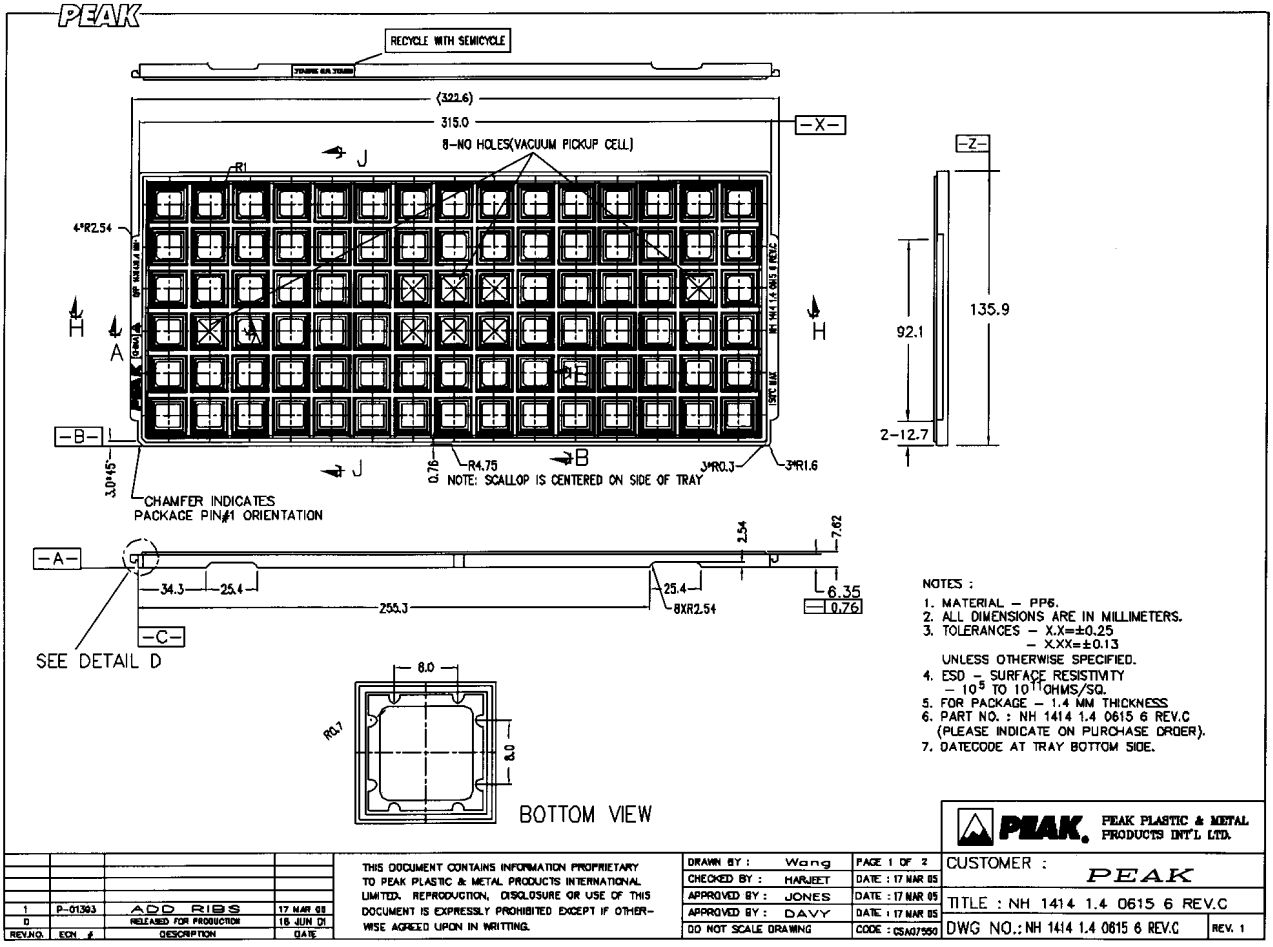
Note: The following notes apply to the package drawing.

1. All dimensions shown in mm.
2. Dimensions shown are nominal with tolerances indicated.
3. Foot length 'L' is measured at gage plane 0.25 mm above seating plane.

Body +2.00 mm Footprint, 1.4 mm package thickness		
Symbols	Leads	100L
A	Max.	1.60
A_1	-	0.05 Min./0.15 Max.
A_2	±0.05	1.40
D	±0.20	16.00
D_1	±0.05	14.00
E	±0.20	16.00
E_1	±0.05	14.00
L	+0.15/-0.10	0.60
e	Basic	0.50
b	+0.05	0.22
θ	-	0˚-7˚
ddd	Max.	0.08
ccc	Max.	0.08
JEDEC Reference Drawing		MS-026
Variation Designator		BED

C.1.2 Tray Dimensions

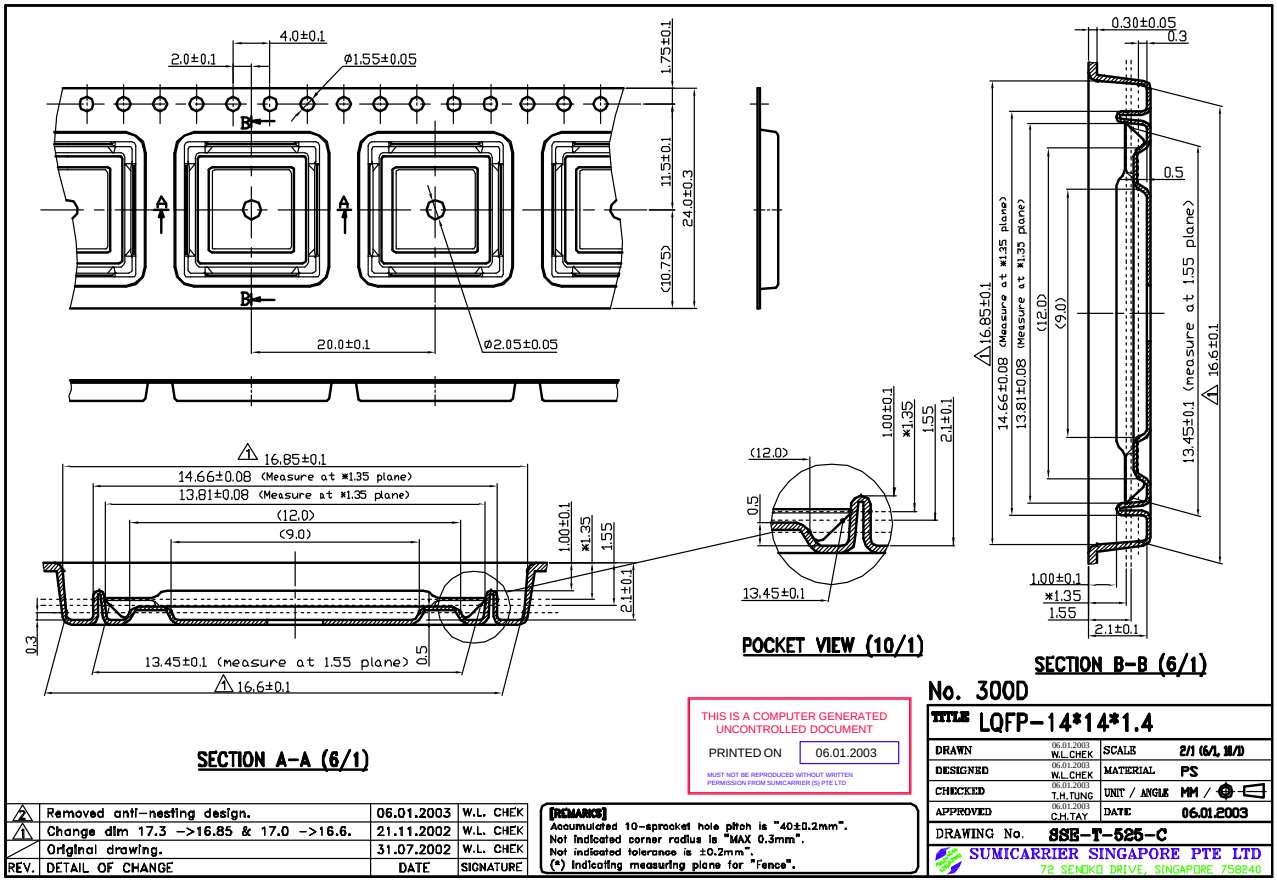
Figure C-2. 100-Pin LQFP Tray Dimensions



C.1.3 Tape and Reel Dimensions

Note: In the figure that follows, pin 1 is located in the top right corner of the device.

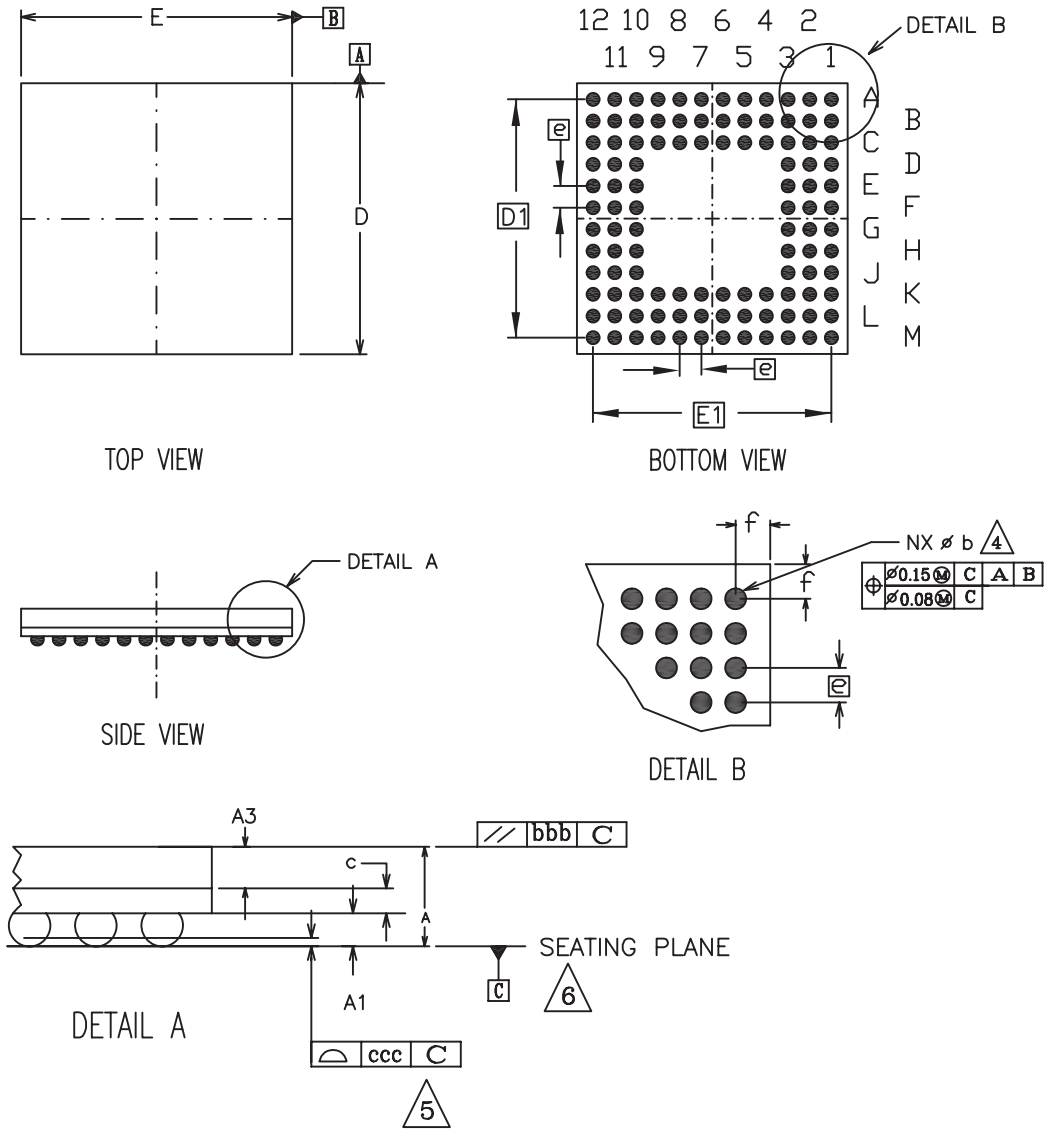
Figure C-3. 100-Pin LQFP Tape and Reel Dimensions



C.2 108-Ball BGA Package

C.2.1 Package Dimensions

Figure C-4. Stellaris LM3S9U81 108-Ball BGA Package Dimensions



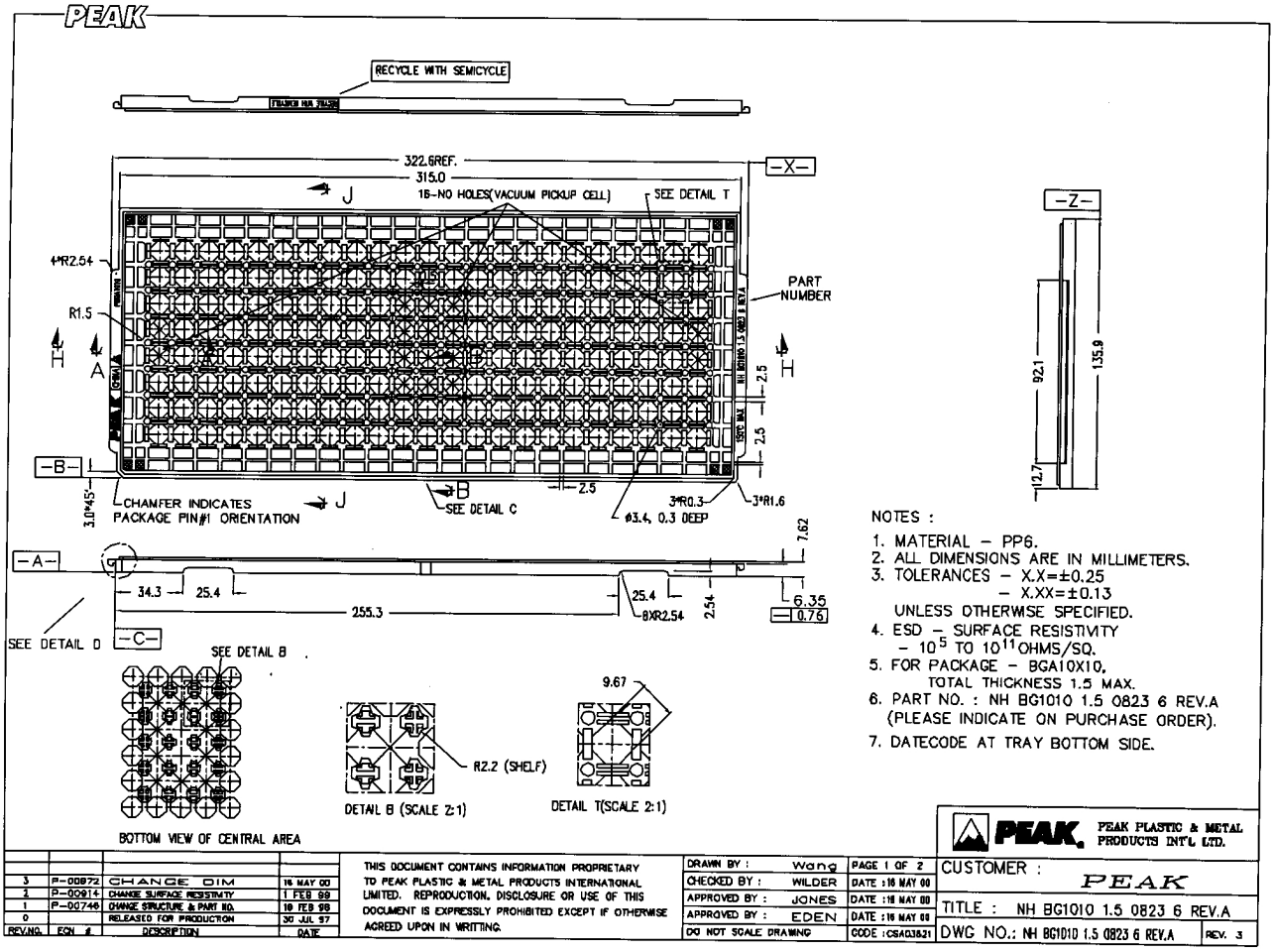
Note: The following notes apply to the package drawing.

1. ALL DIMENSIONS ARE IN MILLIMETERS.
2. 'e' REPRESENTS THE BASIC SOLDER BALL GRID PITCH.
3. 'M' REPRESENTS THE BASIC SOLDER BALL MATRIX SIZE.
   AND SYMBOL 'N' IS THE NUMBER OF BALLS AFTER DEPOPULATING.
4. 'b' IS MEASURABLE AT THE MAXIMUM SOLDER BALL DIAMETER AFTER REFLOW
   PARALLEL TO PRIMARY DAIUM [C] .
5. DIMENSION 'ccc' IS MEASURED PARALLEL TO PRIMARY DATUM [C].
6. PRIMARY DATUM [C] AND SEATING PLANE ARE DEFINED BY THE SPHERICAL
   CROWNS OF THE SOLDER BALLS.
7. PACKAGE SURFACE SHALL BE MATTE FINISH CHARMILLES 24 TO 27.
8. SUBSTRATE MATERIAL BASE IS BT RESIN.
9. THE OVERALL PACKAGE THICKNESS "A" ALREADY CONSIDERS COLLAPSE BALLS
10. DIMENSIONING AND TOLERANCING PER ASME Y14.5M 1994.
11. EXCEPT DIMENSION b.

Symbols	MIN	NOM	MAX
A	1.22	1.36	1.50
A1	0.29	0.34	0.39
A3	0.65	0.70	0.75
c	0.28	0.32	0.36
D	9.85	10.00	10.15
D1		8.80 BSC	
E	9.85	10.00	10.15
E1		8.80 BSC	
b	0.43	0.48	0.53
bbb		.20	
ddd		.12	
e		0.80 BSC	
f	-	0.60	-
M		12	
n		108	
REF: JEDEC MO-219F			

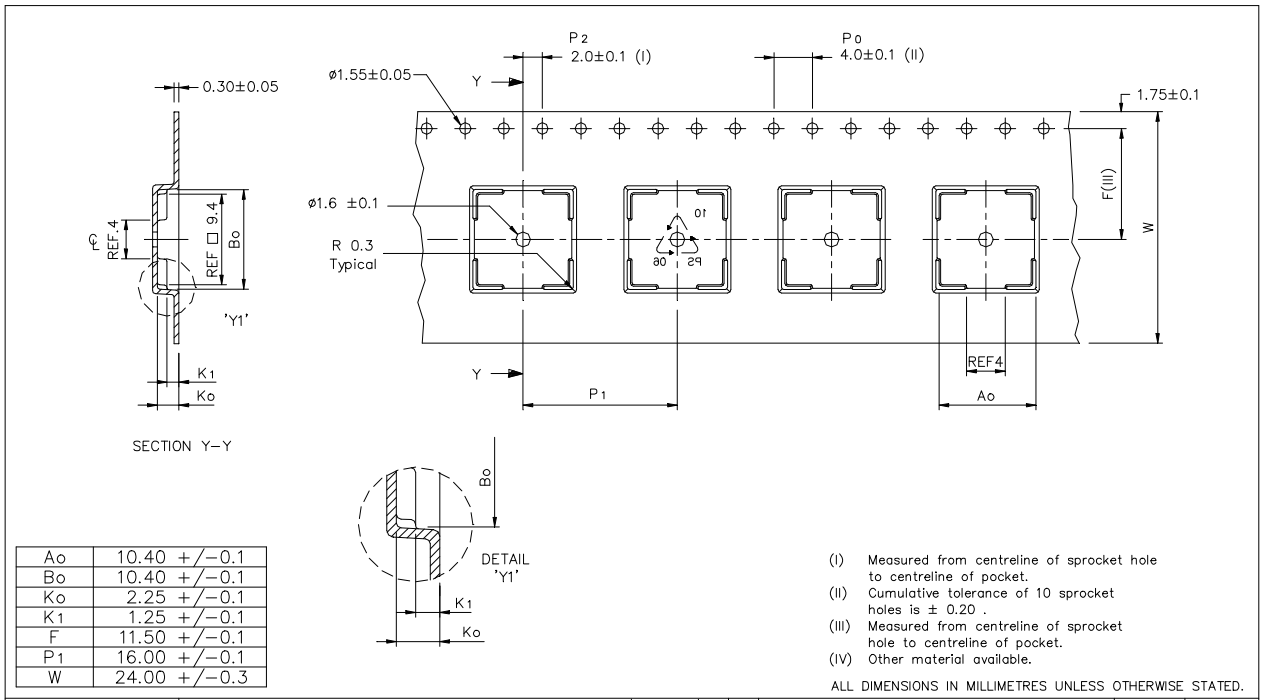
C.2.2 Tray Dimensions

Figure C-5. 108-Ball BGA Tray Dimensions



C.2.3 Tape and Reel Dimensions

Figure C-6. 108-Ball BGA Tape and Reel Dimensions



Ao	10.40 +/–0.1
Bo	10.40 +/–0.1
Ko	2.25 +/–0.1
K1	1.25 +/–0.1
F	11.50 +/–0.1
P1	16.00 +/–0.1
W	24.00 +/–0.3

PACKAGING INFORMATION

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/ Ball Finish	MSL Peak Temp (3)	Samples (Requires Login)
LM3S9U81-IBZ80-A2	ACTIVE	NFBGA	ZCR	108	184	Green (RoHS & no Sb/Br)	SNAGCU	Level-3-260C-168 HR	
LM3S9U81-IBZ80-A2T	ACTIVE	NFBGA	ZCR	108	1500	Green (RoHS & no Sb/Br)	SNAGCU	Level-3-260C-168 HR	
LM3S9U81-IQC80-A2	ACTIVE	LQFP	PZ	100	90	Green (RoHS & no Sb/Br)		Level-3-260C-168 HR	
LM3S9U81-IQC80-A2T	ACTIVE	LQFP	PZ	100	1000	Green (RoHS & no Sb/Br)		Level-3-260C-168 HR	

(1) The marketing status values are defined as follows:
ACTIVE: Product device recommended for new designs.
LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.
PREVIEW: Device has been announced but is not in production. Samples may or may not be available.
OBSOLETE: TI has discontinued the production of the device.

(2) Eco Plan - The planned eco-friendly classification: Pb-Free (RoHS), Pb-Free (RoHS Exempt), or Green (RoHS & no Sb/Br) - please check http://www.ti.com/productcontent for the latest availability information and additional product content details.
TBD: The Pb-Free/Green conversion plan has not been defined.
Pb-Free (RoHS): TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.
Pb-Free (RoHS Exempt): This component has a RoHS exemption for either 1) lead-based flip-chip solder bumps used between the die and package, or 2) lead-based die adhesive used between the die and leadframe. The component is otherwise considered Pb-Free (RoHS compatible) as defined above.
Green (RoHS & no Sb/Br): TI defines "Green" to mean Pb-Free (RoHS compatible), and free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material)

(3) MSL, Peak Temp. -- The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Mobile Processors	www.ti.com/omap		
Wireless Connectivity	www.ti.com/wirelessconnectivity		
	TI E2E Community Home Page		e2e.ti.com

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265