TMS320C5517 Digital Signal Processor

Technical Reference Manual

Literature Number: SPRUH16B
April 2014–Revised October 2015

Contents

List of Figures

List of Tables

Read This First

About This Manual

This technical reference manual (TRM) details the integration, the environment, the functional description, and the programming models for each peripheral in the device.

Related Documentation From Texas Instruments

For a complete listing of related documentation and development-support tools for the TMS320C5517 Digital Signal Processor, visit the Texas Instruments website at www.ti.com.

Revision History

Changes from A Revision (August 2015) to B Revision **Page**

- Updated/Changed Section 10.2.1 paragraph from "... is capable of operating with a function clock up to 50 MHz" to "The functional clock of MMCSD controller is capable of operating up to system clock frequency."..... 613

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

System Control

Topic **Page**

1.1 Introduction

The digital signal processor (DSP) contains a high-performance, low-power DSP to efficiently handle tasks required by portable audio, wireless audio devices, industrial controls, software defined radio, fingerprint biometrics, and medical applications. The DSP consists of the following primary components:

- A C55x CPU and associated memory
- FFT hardware accelerator
- Four DMA controllers and external memory interface
- Power management module
- A set of I/O peripherals that includes I2S, I2C, SPI, UART, MMC/SD, UHPI, McBSP, McSPI, Timers, EMIF, USB 2.0

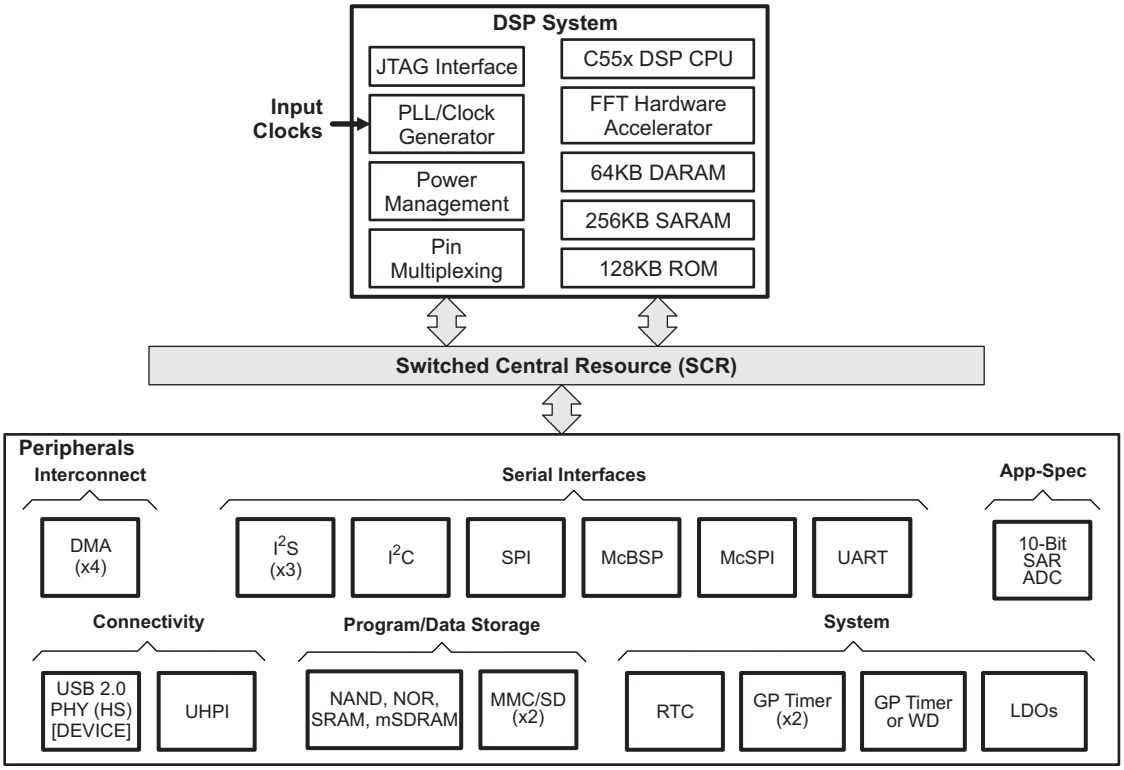
For more information on these components see the following documents:

- *TMS320C55x 3.0 CPU Reference Guide* (SWPU073).
- *TMS320C55x DSP Peripherals Overview Reference Guide* (SPRU317).

1.1.1 Block Diagram

The DSP block diagram is shown in Figure 1-1.

Figure 1-1. Functional Block Diagram



1.1.2 CPU Core

The C55x CPU is responsible for performing the digital signal processing tasks required by the application. In addition, the CPU acts as the overall system controller, responsible for handling many system functions such as system-level initialization, configuration, user interface, user command execution, connectivity functions, and overall system control.

Tightly coupled to the CPU are the following components:

- DSP internal memories
  - Dual-access RAM (DARAM)

  - Single-access RAM (SARAM)
  - Read-only memory (ROM)
- FFT hardware accelerator
- Ports and buses

The CPU also manages/controls all peripherals on the device. Refer to the device-specific data manual for the full list of peripherals.

Figure 1-1 shows the functional block diagram of the DSP and how it connects to the rest of the device. The DSP architecture uses the switched central resource (SCR) to transfer data within the system.

1.1.3 FFT Hardware Accelerator

The C55x CPU includes a tightly-coupled FFT hardware accelerator that communicates with the C55x CPU through the use coprocessor instructions. For ease of use, the ROM has a set of C-callable routines that use these coprocessor instructions to perform 8, 16, 32, 64, 128, or 256-point FFTs. The main features of the FFT hardware accelerator are:

- Support for 8 to 1024-point (in powers of 2) real and complex-valued FFTs and IFFTs.
- An internal twiddle factor generator for optimal use of memory bandwidth and more efficient programming.
- Basic and software-driven auto-scaling feature provides good precision vs cycle count trade-off.
- Single-stage and double-stage modes enabling computation of one or two stages in one pass, thus handling odd power of two FFT widths.

1.1.3.1 Using FFT Accelerator ROM Routines

The device includes C-callable routines in ROM to execute FFT and IFFT using the tightly coupled FFT accelerator. The routines reside in the following address:

Table 1-1. Available HWAFFT Routines

Address	Name	Description	Calling Convention
0x00fefefc	hwafft br	Int32 (32-bit) vector bit-reversal, Strict alignment requirement	void hwafft_br(Int32 *data, Int32 *data_br, Uint16 data_len);
0x00feff10	hwafft 8pts	8-point FFT/IFFT, 1 double-stage, 1 single-stage	Uint16 hwafft_8pts(Int32 *data,Int32 *scratch, Uint16 fft_flag, Uint16 scale_flag);
0x00feffff	hwafft 16pts	16-point FFT/IFFT, 2 double-stages, 0 single-stages	Uint16 hwafft_16pts(Int32 *data,Int32 *scratch, Uint16 fft_flag, Uint16 scale_flag);
0x00ff0155	hwafft 32pts	32-point FFT/IFFT, 2 double-stages, 1 single-stage	Uint16 hwafft_32pts(Int32 *data,Int32 *scratch, Uint16 fft_flag, Uint16 scale_flag);
0x00ff045e	hwafft 64pts	64-point FFT/IFFT, 3 double-stages, 0 single-stages	Uint16 hwafft_64pts(Int32 *data,Int32 *scratch, Uint16 fft_flag, Uint16 scale_flag);
0x00ff05f3	hwafft 128pts	128-point FFT/IFFT, 3 double-stages, 1 single-stage	Uint16 hwafft_128pts(Int32 *data,Int32 *scratch, Uint16 fft_flag, Uint16 scale_flag);
0x00ff0804	hwafft 256pts	256-point FFT/IFFT, 4 double-stages, 0 single-stages	Uint16 hwafft_256pts(Int32 *data,Int32 *scratch, Uint16 fft_flag, Uint16 scale_flag);
0x00ff0a02	hwafft 512pts	512-point FFT/IFFT, 4 double-stages, 1 single-stage	Uint16 hwafft_512pts(Int32 *data,Int32 *scratch, Uint16 fft_flag, Uint16 scale_flag);
0x00ff0c7c	hwafft 1024pts	1024-point FFT/IFFT, 5 double-stages, 0 single-stages	Uint16 hwafft_1024pts(Int32 *data,Int32 *scratch, Uint16 fft_flag, Uint16 scale_flag);

Note that for the FFT routines, output data is dependent on the return value (T0). If return = 0 output data is in-place, meaning the result will overwrite the input buffer. If return =1, output data is placed in the scratch buffer. The 32-bit input and output data consist of 16-bit real and 16-bit imaginary data. If only real data is used, the imaginary part can be zeroed. The Scale flag determines if the butterfly output is divided by 2 to prevent overflow at the expense of resolution. For further information on how to use these routines, see Chapter 2, *FFT Implementation*.

1.1.4 Power Management

Integrated into the DSP are the following power management features:

- One low dropout LDO for power management circuits (V_{DDA_ANA}): ANA_LDO
- One LDO for DSP core (CV_{DD}): DSP_LDO
- One LDO for USB core and PHY (USB_V_{DDA1P3}): USB_LDO
- Idle controller with several clock domains:
  - CPU domain
  - Clock generator domain
  - Peripheral domain
  - USB domain
  - Real-time clock (RTC) domain
- Independent voltage and power domains
- LDOI (LDOs and Bandgap Power Supply)
- Analog POR, and PLL (V_{DDA_ANA} and V_{DDA_PLL})
- Real-time clock core (CV_{DDRTC}) **Note:** CV_{DDRTC} must always be powered by an external power source. None of the on-chip LDOs can be used to power CV_{DDRTC}.
- Digital core (CV_{DD})
- USB core (USB_ V_{DD1P3} and USB_V_{DDA1P3})
- USB PHY and USB PLL (USB_V_{DDOSC}, USB_V_{DDA3P3}, and USB_V_{DDPLL})
- EMIF I/O (DV_{DDEMIF})
- RTC I/O (DV_{DDRTC})
- Rest of the I/O (DV_{DDIO})

1.1.5 Peripherals

The DSP includes the following peripherals:

- Four direct memory access (DMA) controllers, each with four independent channels.
- One external memory interface (EMIF) with 21-bit address and 16-bit data. The EMIF has support for mobile SDRAM and non-mobile SDRAM single-level cell (SCL) NAND with 1-bit ECC, and multi-level cell (MLC) NAND with 4-bit ECC.

NOTE: The DSP can support non-mobile SDRAM under certain circumstances. The DSP always uses mobile SDRAM initialization but it is able to support SDRAM memories that ignore the BA0 and BA1 pins for the 'load mode register' command. During the mobile SDRAM initialization, the device issues the 'load mode register' initialization command to two different addresses that differ in only the BA0 and BA1 address bits. These registers are the Extended Mode register and the Mode register. The Extended mode register exists only in mSDRAM and not in non-mSDRAM. If a non-mobile SDRAM memory ignores bits BA0 and BA1, the second loaded register value overwrites the first, leaving the desired value in the Mode register and the non-mobile SDRAM will work with this DSP.

- Two serial busses:
  - One configurable to support one Multimedia Card (MMC) / Secure Digital (SD/SDIO) controller, one multichannel serial peripheral interface (McSPI), or a full GPIO interface
  - One configurable to support one Multimedia Card (MMC) / Secure Digital (SD/SDIO) controller, one inter IC sound bus (I2S), one multichannel buffered serial port (McBSP), or a full GPIO interface
- One parallel bus configurable to support a universal host port interface (UHPI), or a combination of an SPI, an I2S, a universal asynchronous receiver/transmitter (UART), a mobile SDRAM, and GPIOs.
- One inter-integrated circuit (I2C) multi-master and slave interface with 7-bit and 10-bit addressing modes.
- Three 32-bit timers with 16-bit prescaler; one timer supports watchdog functionality.

- A USB 2.0 slave.
- A 10-bit successive approximation (SAR) analog-to-digital converter with touchscreen conversion capability.
- One real-time clock (RTC) with associated low power mode.

1.2 System Memory

The DSP supports a unified memory map (program code sections and data sections can be mixed and interleaved within the entire memory space) composed of both on-chip and external memory. The on-chip memory consists of 320KB of RAM and 128KB of ROM.

The external memory interface (EMIF) port provides the means for the DSP to access external memory and devices including: mobile and non-mobile single data rate (SDR) SDRAM, (for limitations, see note in Section 1.1.5), NOR Flash, NAND Flash and SRAM.

Separate from the program and data space, the DSP also includes a 64K-byte I/O space for peripheral registers.

1.2.1 Program/Data Memory Map

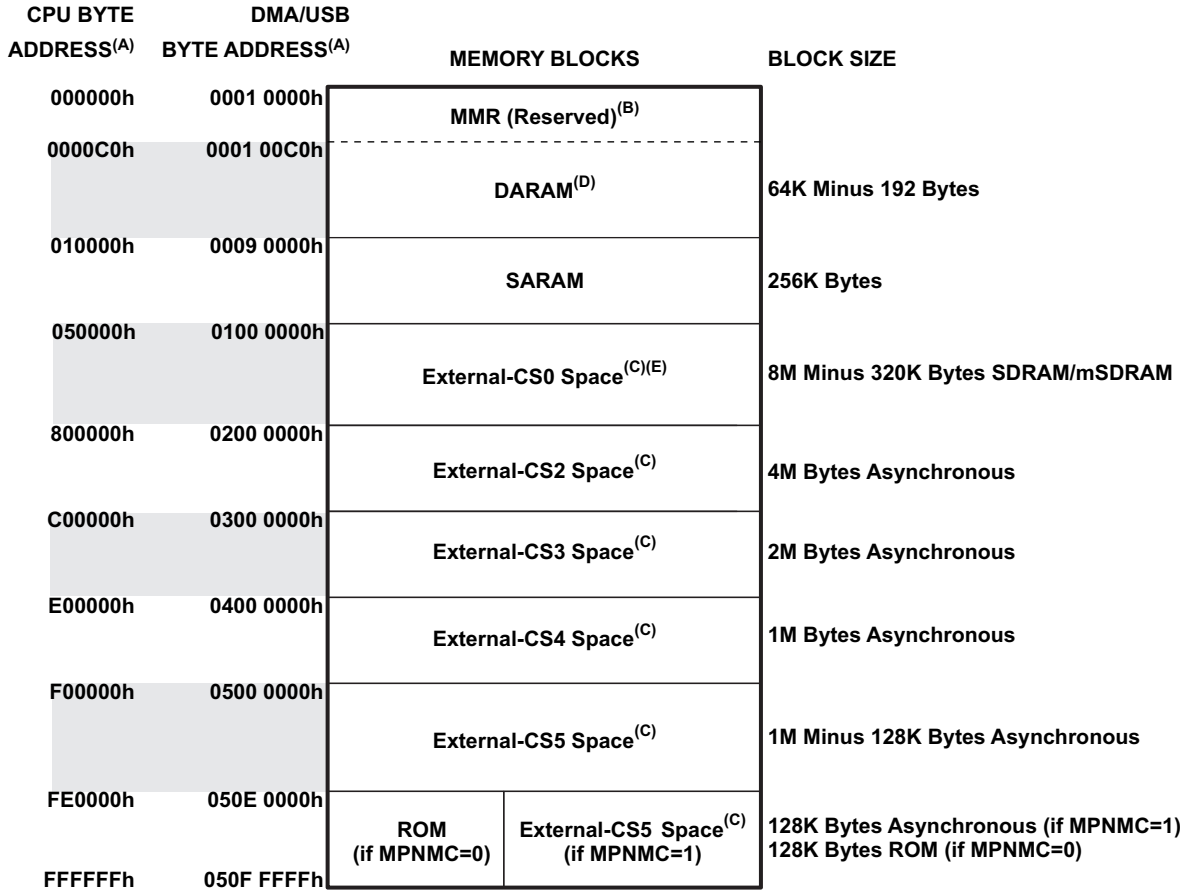
The device provides 16MB of total address space composed of on-chip RAM, on-chip ROM, and external memory space supporting a variety of memory types.

The on-chip, dual-access RAM allows two accesses to a given block during the same cycle. The device has 8 blocks of 8K-bytes of dual-access RAM. The on-chip, single-access RAM allows one access to a given block per cycle. The device has 32 blocks of 8K-bytes of single-access RAM. Attempts to perform two accesses in a cycle to single-access memory will cause one access to stall until the next cycle. An access is defined as either a read or write operation. For the most efficient use of DSP processing power (MIPS), it is important to pay attention to the memory blocks that are being simultaneously accessed by the code and data operations.

The external memory space is divided into five spaces. Each space has a chip select decode signal (called CS) that indicates an access to the selected space. The external memory interface (EMIF) supports access to asynchronous memories such as SRAM Flash, mobile SDRAM and SDRAM.

The DSP memory is accessible by different master modules within the DSP, including the device CPU, the four DMA controllers, and the USB. The DSP memory map as seen by these modules is illustrated in Figure 1-2.

Figure 1-2. DSP Memory Map (A) (B) (C) (D)



A Address shown represents the first byte address in each block.

B The first 192 bytes are reserved for memory-mapped registers (MMRs).

C Out of the four DMA controllers, *only* DMA controller 3 has access to the external memory space.

D The USB controller does not have access to DARAM.

E The CS0 space can be accessed by CS0 *only* or by CS0 *and* CS1.

1.2.1.1 On-Chip Dual-Access RAM (DARAM)

The DARAM is located in the CPU byte address range 00 00C0h - 00 FFFFh and is composed of eight blocks of 4K words each (see Table 1-2). Each DARAM block can perform two accesses per cycle (two reads, two writes, or a read and a write). DARAM can be accessed by the internal program, data, and DMA buses.

As shown in Table 1-2, the DMA controllers access DARAM at an address offset 0x0001_0000 from the CPU memory byte address space.

Table 1-2. DARAM Blocks

Memory Block	CPU Byte Address Range	DMA/USB Controller Byte Address Range
DARAM 0(1)	00 00C0h - 00 1FFFh	0001 00C0h - 0001 1FFFh
DARAM 1	00 2000h - 00 3FFFh	0001 2000h - 0001 3FFFh
DARAM 2	00 4000h - 00 5FFFh	0001 4000h - 0001 5FFFh
DARAM 3	00 6000h - 00 7FFFh	0001 6000h - 0001 7FFFh
DARAM 4	00 8000h - 00 9FFFh	0001 8000h - 0001 9FFFh
DARAM 5	00 A000h - 00 BFFFh	0001 A000h - 0001 BFFFh

(1) First 192 bytes are reserved for memory-mapped registers (MMRs).

Table 1-2. DARAM Blocks (continued)

Memory Block	CPU Byte Address Range	DMA/USB Controller Byte Address Range
DARAM 6	00 C000h - 00 DFFFh	0001 C000h - 0001 DFFFh
DARAM 7	00 E000h - 00 FFFFh	0001 E000h - 0001 FFFFh

1.2.1.2 On-Chip Single-Access RAM (SARAM)

The SARAM is located at the CPU byte address range 01 0000h - 0FFFFh and is composed of 32 blocks of 4K words each (see Table 1-3). Each SARAM block can perform one access per cycle (one read or one write). SARAM can be accessed by the internal program, data, and DMA buses.

As shown in Table 1-3, the DMA controllers access SARAM at an address offset 0x0008_0000 from the CPU memory byte address space.

Table 1-3. SARAM Blocks

Memory Block	CPU Byte Address Range	DMA/USB Controller Byte Address Range
SARAM 0	01 0000h - 01 1FFFh	0009 0000h - 0009 1FFFh
SARAM 1	01 2000h - 01 3FFFh	0009 2000h - 0009 3FFFh
SARAM 2	01 4000h - 01 5FFFh	0009 4000h - 0009 5FFFh
SARAM 3	01 6000h - 01 7FFFh	0009 6000h - 0009 7FFFh
SARAM 4	01 8000h - 01 9FFFh	0009 8000h - 0009 9FFFh
SARAM 5	01 A000h - 01 BFFFh	0009 A000h - 0009 BFFFh
SARAM 6	01 C000h - 01 DFFFh	0009 C000h - 0009 DFFFh
SARAM 7	01 E000h - 01 FFFFh	0009 E000h - 0009 FFFFh
SARAM 8	02 0000h - 02 1FFFh	000A 0000h - 000A 1FFFh
SARAM 9	02 2000h - 02 3FFFh	000A 2000h - 000A 3FFFh
SARAM 10	02 4000h - 02 5FFFh	000A 4000h - 000A 5FFFh
SARAM 11	02 6000h - 02 7FFFh	000A 6000h - 000A 7FFFh
SARAM 12	02 8000h - 02 9FFFh	000A 8000h - 000A 9FFFh
SARAM 13	02 A000h - 02 BFFFh	000A A000h - 000A BFFFh
SARAM 14	02 C000h - 02 DFFFh	000A C000h - 000A DFFFh
SARAM 15	02 E000h - 02 FFFFh	000A E000h - 000A FFFFh
SARAM 16	03 0000h - 03 1FFFh	000B 0000h - 000B 1FFFh
SARAM 17	03 2000h - 03 3FFFh	000B 2000h - 000B 3FFFh
SARAM 18	03 4000h - 03 5FFFh	000B 4000h - 000B 5FFFh
SARAM 19	03 6000h - 03 7FFFh	000B 6000h - 000B 7FFFh
SARAM 20	03 8000h - 03 9FFFh	000B 8000h - 000B 9FFFh
SARAM 21	03 A000h - 03 BFFFh	000B A000h - 000B BFFFh
SARAM 22	03 C000h - 03 DFFFh	000B C000h - 000B DFFFh
SARAM 23	03 E000h - 03 FFFFh	000B E000h - 000B FFFFh
SARAM 24	04 0000h - 04 1FFFh	000C 0000h - 000C 1FFFh
SARAM 25	04 2000h - 04 3FFFh	000C 2000h - 000C 3FFFh
SARAM 26	04 4000h - 04 5FFFh	000C 4000h - 000C 5FFFh
SARAM 27	04 6000h - 04 7FFFh	000C 6000h - 000C 7FFFh
SARAM 28	04 8000h - 04 9FFFh	000C 8000h - 000C 9FFFh
SARAM 29	04 A000h - 04 BFFFh	000C A000h - 000C BFFFh
SARAM 30	04 C000h - 04 DFFFh	000C C000h - 000C DFFFh
SARAM 31	04 E000h - 04 FFFFh	000C E000h - 000C FFFFh

1.2.1.3 On-Chip Single-Access Read-Only Memory (SAROM)

The zero-wait-state ROM is located at the CPU byte address range FE 0000h - FF FFFFh. The ROM is composed of four 16K-word blocks, for a total of 128K-bytes of ROM. Each ROM block can perform one access per cycle (one read or one write). ROM can be accessed by the internal program or data buses, but not the DMA buses. The ROM address space can be mapped by software to the external memory or to the internal ROM via the MPNMC bit in the ST3 status register.

The standard device includes a Bootloader program resident in the ROM and the bootloader code is executed immediately after hardware reset. When the MPNMC bit field of the ST3 status register is set through software, the on-chip ROM is disabled and not present in the memory map, and byte address range FE 0000h - FF FFFFh is directed to external memory space (extends CS5 address reach). A hardware reset always clears the MPNMC bit, so it is not possible to disable the ROM at hardware reset. However, the software reset instruction does not affect the MPNMC bit. The ROM can be accessed by the program and data buses. Each SAROM block can perform one word read access per cycle.

Table 1-4. SAROM Blocks

Memory Block	CPU Byte Address Range	CPU Word Address Range
SAROM0	FE 0000h - FE 7FFFh	7F 0000h - 7F 3FFFh
SAROM1	FE 8000h - FE FFFFh	7F 4000h - 7F 7FFFh
SAROM2	FF 0000h - FF 7FFFh	7F 8000h - 7F BFFFh
SAROM3	FF 8000h - FF FFFFh	7F C000h - 7F FFFFh

1.2.1.4 External Memory

The external memory space of the device is located at the byte address range 05 0000h - FF FFFFh. The external memory space is divided into five chip select spaces. The synchronous space is activated by one chip select pin ($\overline{\text{EM\_CS0}}$) or by a pair of chip selects pins ($\overline{\text{EM\_CS0}}$ and $\overline{\text{EM\_CS1}}$). Each asynchronous chip select space has a corresponding chip select pin (called $\overline{\text{EMIF\_CS}}$[2:5]) that is activated during an access to the chip select space.

The external memory interface (EMIF) provides the means for the DSP to access external memories and other devices including: NOR Flash, NAND Flash, SRAM, mSDRAM, and SDRAM. Before accessing external memory, you must configure the EMIF through its registers. For more detail on the EMIF, see Section 4.1.

As described in Section 1.2.1.3, when the MPNMC bit field of the ST3 status register is cleared (default), the byte address range FE 0000h - FF FFFFh is reserved for the on-chip ROM, which decreases the addressable size for EM_CS5.

The EMIF provides a configurable 16-bit (synchronous or asynchronous) or 8-bit (asynchronous only) data bus, an address bus width of up to 21-bits, and five dedicated chip selects, along with memory control signals. To maximize power savings, the I/O pins of the EMIF can be operated at lower voltage independently of other I/O pins on the DSP. Further power savings may be achieved by setting the EMIF I/O pins to have slow slew rate, as described in Section 1.7.3.5.

1.2.1.4.1 Asynchronous EMIF Interface

The EMIF provides a configurable 16- or 8-bit data bus with address bus width of up to 21-bits, and six dedicated chip selects, along with memory control signals. The cycle timings of the asynchronous interface are fully programmable, allowing for access to a wide range of devices including NAND flash, NOR flash, and SRAM as well as other asynchronous devices such as a TI DSP HPI interface. In NAND mode, the asynchronous interface supports 1-bit ECC for 8- and 16-bit NAND flash and 4-bit ECC for 8-bit NAND flash.

1.2.1.5 Synchronous EMIF Interface

The EMIF provides a 16-bit data bus with one or two dedicated chip selects for mSDRAM. Non-mobile SDRAM can be supported under certain circumstances. The DSP always uses a mobile SDRAM initialization command sequence, but it is able to support SDRAM memories that ignore the BA0 and BA1 pins for the load mode register command. During the mobile SDRAM initialization, the device issues the load mode register initialization command to two different addresses that differ in only the BA0 and BA1 address bits. These registers are the Extended Mode register and the Mode register. The extended mode register exists only in mSDRAM, and not in non-mSDRAM. If a non-mobile SDRAM memory ignores bits BA0 and BA1, the second loaded register value overwrites the first, leaving the desired value in the mode register and the non-mobile SDRAM works with the device.

Some timing parameters are programmable such as the refresh rate and CAS latencies. The EMIF supports up to 100 MHz SDCLK and has the ability to run the SDCLK at half the system clock to meet the EMIF I/O timing requirements and/or at lower power if a slower SDCLK can be used. Detailed information is available in the *Clock Control* section of the EMIF chapter.

1.2.2 I/O Memory Map

The C5x DSP has a separate memory map for peripheral and system registers, called I/O space. This space is 64K-words in length and is accessed via word read and write instructions dedicated for I/O space.

Separate documentation for I/O space registers related to each peripheral exists and is listed in the preface of this guide. System registers, which provide system-level control and status, are described in detail in other sections throughout this guide. Unused addresses in I/O space should be treated as reserved and should not be accessed. Accessing unused I/O space addresses may stall or hang the DSP.

Each of the four DMA controllers has access to a different set of peripherals and their I/O space registers. This is shown in Section 1.7.4.

NOTE: Writing to I/O space registers incurs in at least 2 CPU cycle latency. Thus, when configuring peripheral devices, wait at least two cycles before accessing data from the peripheral. When more than one peripheral register is updated in a sequence, the CPU only needs to wait following the final register write. For example, if the EMIF is being reconfigured, the CPU must wait until the very last EMIF register update takes effect before trying to access the external memory. See the respective peripheral chapter to determine if a peripheral requires additional initialization time.

Before accessing any peripheral register, make sure the peripheral is not held in reset and its internal clock is enabled. The peripheral reset control register (Section 1.7.5.2) and the peripheral clock gating control registers (Section 1.5.3.2.1) control these functions. Accessing a peripheral whose clocks are gated will either return the value of the last address read from the peripheral (when the clocks were last ON) or it may possibly hang the DSP -- depending on the peripheral.

1.3 Device Clocking

1.3.1 Overview

The DSP requires two primary reference clocks: a system reference clock and a USB reference clock. The system clock, which is used by the CPU and most of the DSP peripherals, is controlled by the system clock generator. The system clock generator features a software-programmable PLL multiplier and several dividers. The system clock generator accepts an input reference clock from the CLKIN pin or the 12-MHz on-chip USB reference clock. The selection of the input reference clock is based on the state of the CLK_SEL pin. The CLK_SEL pin is required to be statically tied high or low and cannot change dynamically after reset.

The system clock generator can be used to modify the system reference clock signal according to software-programmable multiplier and dividers. The resulting clock output, the DSP system clock, is passed to the CPU, peripherals, and other modules inside the DSP.

Alternatively, the system clock generator can be fully bypassed and the input reference clock can be passed directly to the DSP system clock. The USB reference clock is generated using a dedicated on-chip oscillator with a 12-MHz external crystal connected to the USB_MXI and USB_MXO pins. This crystal is not required if the USB oscillator is not used as input to the system reference clock and the USB peripheral is not being used.

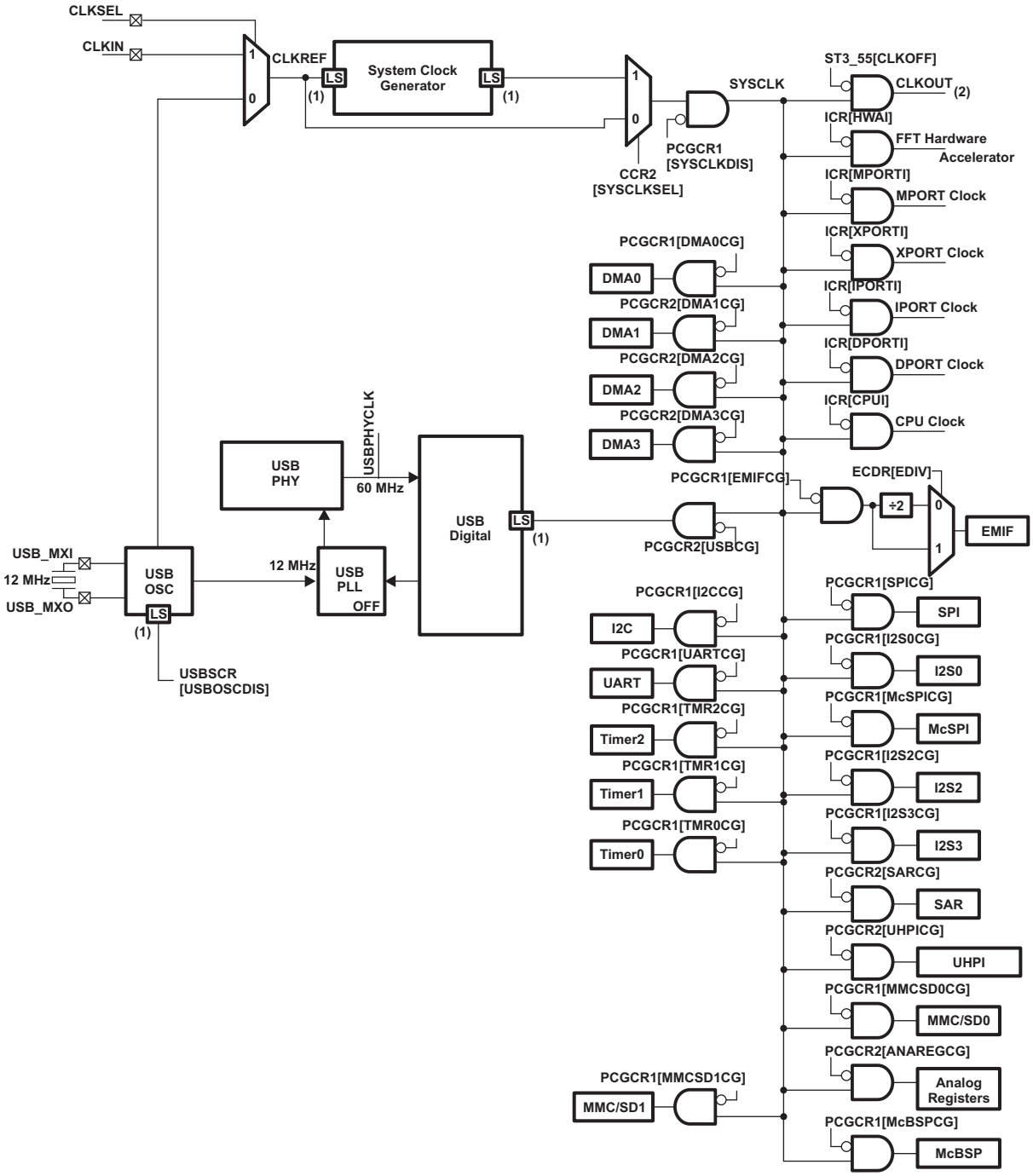
The RTC oscillator generates a clock when a 32.768-KHz crystal is connected to the RTC_XI and RTC_XO pins. RTC core (CV_{DDRTC}) must always be powered by an external power source. None of the on-chip LDOs can power the CV_{DDRTC}.

The 32.768-KHz crystal can be disabled if CLKIN is used as the clock source for the DSP. However, when the RTC oscillator is disabled, the RTC peripheral will not operate and the RTC registers (I/O address range 1900h - 197Fh) will not be accessible. This includes the RTC power management register (RTCPMGT) which controls the RTCLKOUT and WAKEUP pins. To disable the RTC oscillator, connect the RTC_XI pin to CV_{DDRTC} and the RTC_XO pin to ground.

The USB oscillator is powered down at hardware reset only if CLK_SEL is 1. When CLK_SEL is 1, the USB oscillator can be enabled or disabled, by the USBSCR register, and must be allowed to settle for an amount of time specified by USB Oscillator Startup Time parameter in the device specific manual before using the USB peripheral. When CLK_SEL is 0 at reset, the USB LDO is enabled, the USB oscillator is enabled, and the USB oscillator is used as the clock source for the system clock generator. Since the USB oscillator is the system's clock source, it is not possible to disable the USB oscillator when CLK_SEL is 0.

Figure 1-3 shows the overall DSP clock structure. For detailed specifications on clock frequency, voltage requirements, and oscillator/crystal requirements, see the device-specific data manual.

Figure 1-3. DSP Clocking Diagram (1) (2)



(1) LS = Level Shifter

(2) The CLKOUT pin's output driver is enabled/disabled through the CLKOFF bit of the CPU ST3_55 register. At the beginning of the boot sequence, the on-chip Bootloader sets CLKOFF = 1 and CLKOUT pin is disabled (high-impedance). For more information on the ST3_55 register, see the *TMS320C55x 3.0 CPU Reference Guide* (SWPU073).

1.3.2 Clock Domains

The device has many clock domains defined by individually disabled portions of the clock tree structure. Understanding the clock domains and their clock enable/disable control registers is very important for managing power and for ensuring clocks are enabled for domains that are needed. By disabling the clocks and thus the switching current in portions of the chip that are not used, lower dynamic power consumption can be achieved and prolonging battery life.

Figure 1-3 shows the clock tree structure with the clock gating represented by the AND gates. Each AND gate shows the controlling register that allows the downstream clock signal to be enabled/disabled. Once disabled most clock domains can be re-enabled, when the associated clock domain logic is needed, via software running on the CPU. But some domains actually stop the clocks to the CPU and therefore software running on the CPU cannot be responsible for re-enabling those clock domains. Other mechanism must exist for restarting those clocks, and the specific cases are listed below:

- The System Clock Generator (PLL) can be powered-down by writing a 1 to PLLPWRDN bit in the PLL control register, PCR. This stops the PLL from oscillating and shuts down its analog circuits. It is important to bypass the System Clock Generator by writing 0 to SYSCLKSEL bit in CCR2 (clock configuration register 2) prior to powering it down, else the CPU will lose its clock and not be able to recover without hardware reset.

  NOTE: Failsafe logic exists to prevent selecting the PLL clock if it has been powered down but this logic does not protect against powering down the PLL while it is selected as the system clock source. Therefore, software should always maintain responsibility for bypassing the PLL prior to and whenever it is powered down.

- The SYSCLKDIS bit in PCGCR1 (clock gating control register lower) is the master clock gater. Asserting this bit causes the main system clock, SYSCLK, to stop and, therefore, the CPU and all peripherals no longer receive clocks. The WAKEUP pin, INT0 & INT1 pin, or RTC interrupt can be used to re-enable the clock from this condition.
- The ICR bit in CPUI (clock gating control register) gates clocks to the CPU and uses the CPU's *idle* instruction to initiate the clock off mode. Any non-masked interrupt can be used to re-enable the CPU clocks.

1.4 System Clock Generator

1.4.1 Overview

The system clock generator (Figure 1-4) features a software-programmable PLL multiplier and several dividers. The clock generator accepts an input clock from the CLKIN pin or the output clock of the USB oscillator. The clock generator offers flexibility and convenience by way of software-configurable multiplier and divider to modify the clock rate internally. The resulting clock output, SYSCLK, is passed to the CPU, peripherals, and other modules inside the DSP.

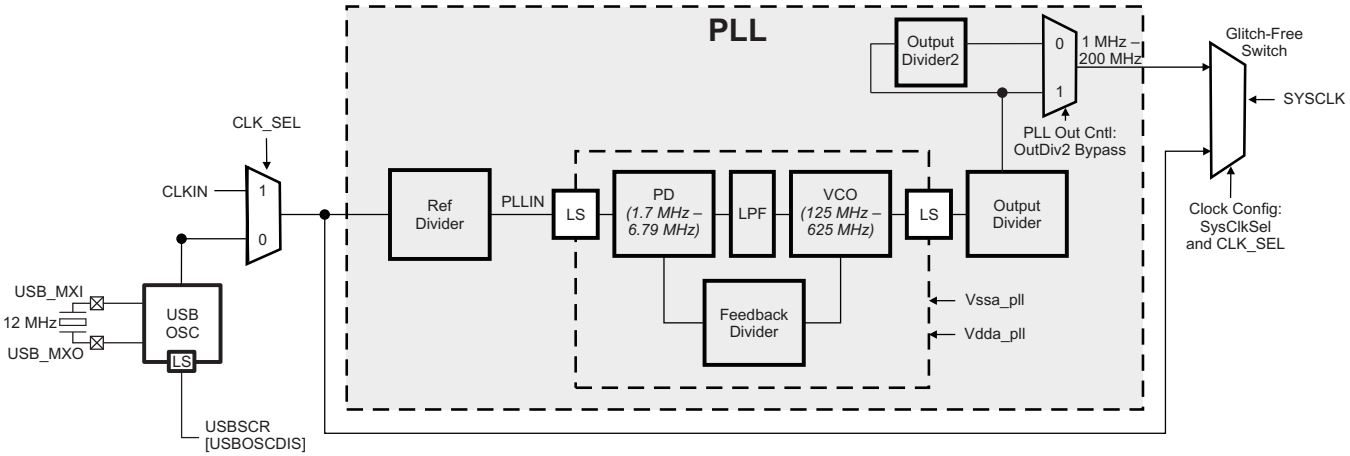
A set of registers are provided for controlling and monitoring the activity of the clock generator. You can write to the SYSCLKSEL bit in CCR2 register to toggle between the two main modes of operation:

- In the BYPASS MODE (see Section 1.4.3.1), the entire clock generator is bypassed, and the frequency of SYSCLK is determined by CLKIN or the USB oscillator output. Once the PLL is bypassed, the PLL can be powered down to save power.
- In the PLL MODE (see Section 1.4.3.2), the input frequency can be both multiplied and divided to produce the desired SYSCLK frequency, and the SYSCLK signal is phase-locked to the input clock signal (CLKREF).

The clock generator bypass mux (controlled by SYSCLKSEL bit in CCR2 register) is a glitch-free mux, which means that clocks will be switched cleanly and not short cycle pulses when switching among the BYPASS MODE and PLL MODE.

For debug purposes, the CLKOUT pin can be used to see different clocks within the clock generator. For details, see Section 1.4.2.3.

Figure 1-4. Clock Generator



1.4.2 Functional Description

The following sections describe the multiplier and dividers of the clock generator.

1.4.2.1 Multiplier and Dividers

The PLL clock generator has one multiplier and three programmable dividers: one before the PLL input and two on the PLL output. The reference clock divider can be programmed to divide the clock generator input clock from a /1 to /64 divider ratio. The Reference Divider must be programmed such that the PLLIN frequency range is 1.7 MHz to 6.79 MHz. At the output of the PD, the output divider can be used to divide the VCO output clock from a /1 to a /8 divider ratio. The output divider2 can be used to further divide the VCO output clock from a /2 to a /64 to further adjust the SYSCLK to meet the maximum operating frequency defined by the datasheet.Keep in mind that programming the output divider with an odd divisor value other than 1 will result in a non-50% duty cycle SYSCLK. This is not a problem for any of the on-chip logic, but the non-50% duty cycle will be visible on chip pins such as EM_SDCLK (in full-rate mode) and CLKOUT.

NOTE: For allowed values of PLLIN, VCO, and SYSCLK, see the datasheet, *TMS320C5517 Fixed-Point Digital Signal Processor* (literature number SPRS727).

The multiplier and divider ratios are controlled through the PLL control registers. The PLLM bits define the multiplier rate. The OD and OD2 bits define the divide ratio of the reference divider and programmable output divider, respectively. The OUTDIV2BY bit is used to enable or bypass the output divider2. Figure 1-5 lists the formulas for the output frequency based on the setting of these bits.

The clock generator must be placed in BYPASS MODE when any PLL dividers or multipliers are changed. Then, it must remain in BYPASS MODE for at least 4 mS before switching to PLL MODE.

Figure 1-5. PLL Output Frequency Configuration

$$\text{SystemClock} = \text{InputClock} \times \left[\frac{\left(\frac{\text{PLLM}}{256} + 1 \right)}{(\text{RD}+1) \times (0\text{D}+1) \times 2(0\text{D2}+1)} \right]$$

1.4.2.2 Powering Down and Powering Up the System PLL

To save power, you can put the PLL in its power down mode. You can power down the PLL by setting the PLLPWRDN = 1 in the PLL control register, PCR. However, before powering down the PLL, you must first place the clock generator in bypass mode.

When the PLL is powered up (PLLPWRDN = 0), the PLL will start its phase-locking sequence. You must keep the clock generator in BYPASS MODE for at least 4 mS while the phase-locking sequence is ongoing. See Section 1.4.3.2 for more details on the PLL_MODE of the clock generator.

1.4.2.3 CLKOUT Pin

For debug purposes, the DSP includes a CLKOUT pin which can be used to tap different clocks within the clock generator to drive other devices on their board. The SRC bits of the CLKOUT configuration register (CLKOUTCR) can be used to specify the source for the CLKOUT pin (see Figure 1-6 and Table 1-5). The CLKOUT_DIV bits provide the various options to divide down the system clock to be used on the CLKOUT pin. The CLKOUT_GZ bit can be used to enable and disable the CLKOUT pin's output buffer.

The CLKOUT pin undergoes clock gating through the CLKOFF bit of the CPU ST3_55 register. At hardware reset, CLKOFF is cleared to 0 so that the clock is visible for debug purposes. But within the bootloader romcode, CLKOFF is set to 1 to conserve power. After the bootloader finishes, the customer application code is free to re-enable CLKOUT. For more information on the ST3_55 register, see the *TMS320C55x 3.0 CPU Reference Guide* (SWPU073). A glitchless clock switching can be implemented by gating CLKOUT, changing the CLKOUT source, and then un-gating CLKOUT.

The slew rate (i.e., dV/dt) of the CLKOUT pin can be controlled by the CLKOUTSR bits in the output slew rate control register (OSRCR). This feature allows for additional power savings when the CLKOUT pin does not need to drive large loads.

Figure 1-6. CLKOUT Configuration Register (CLKOUTCR) [1C24h]

15	14	13	12	11	10	9	8
CLKOUT_GZ	Reserved				CLKOUT_DIV		
R/W-0h	R-0h				R/W-0h		

7	6	5	4	3	2	1	0
Reserved				CLKOUT_SRC			
R-0h				R/W-Bh			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-5. CLKOUT Configuration Register (CLKOUTCR) Field Descriptions

Bit	Field	Type	Reset	Description
15	CLKOUT_GZ	R/W	0h	Output Enable for the CLKOUT pin output buffer: 0x0 = Output driver is enabled 0x1 = Output driver is tristated
14-11	Reserved	R	0h	Reserved.
10-8	CLKOUT_DIV	R/W	0h	CLKOUT Divider value: 000b = CLKOUT represents the CLKOUT clock source divided by 1 001b = CLKOUT represents the CLKOUT clock source divided by 2 010b = CLKOUT represents the CLKOUT clock source divided by 4 011b = CLKOUT represents the CLKOUT clock source divided by 6 100b = CLKOUT represents the CLKOUT clock source divided by 8 101b = CLKOUT represents the CLKOUT clock source divided by 10 110b = CLKOUT represents the CLKOUT clock source divided by 12 111b = CLKOUT represents the CLKOUT clock source divided by 14
7-4	Reserved	R	0h	Reserved.

Table 1-5. CLKOUT Configuration Register (CLKOUTCR) Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	CLKOUT_SRC	R/W	Bh	CLKOUT clock source selector 0000b: Clock source is the System PLL output. 0001b: Clock source is statically held high. 0010b: Clock source is the System PLL output. 0011b: Clock source is statically held low. 0100b: Clock source is the System PLL output. 0101b: Clock source is statically held low. 0110b: Clock source is the System PLL output. 0111b: Clock source is the USB PHY PLL output (48 MHz). 1000b: Clock source is the System PLL output. 1001b: Clock source is the SAR clock. 1010b: Clock source is the System PLL output. 1011b: Clock source is the DSP clock **(reset default)**. 1100b: Clock source is the System PLL output. 1101b: Clock source is the glitch-free clock mux output. 1110b: Clock source is the System PLL output. 1111b: Clock source is the USB PLL output (48 MHz).

1.4.2.4 DSP Reset Conditions of the System Clock Generator

The following sections describe the operation of the system clock generator when the DSP is held in reset state and the DSP is removed from its reset state.

1.4.2.4.1 Clock Generator During Reset

During reset, the PLLPWRDN bit of the PLL Control Register (PCR) is set to 1, and the PLL does not generate an output clock. Furthermore, the SYSCLKSEL bit of the clock configuration register 2 (CCR2) defaults to 0 (BYPASS MODE), and the system clock (SYSCLK) is driven by either the CLKIN pin or the USB oscillator. See Section 1.4.3.1 for more information on the bypass mode of the clock generator.

1.4.2.4.2 Clock Generator After Reset

After reset, the on-chip bootloader programs the system clock generator based on the input clock selected via the CLK_SEL pin. If CLK_SEL = 0, the bootloader programs the system clock generator and sets the system clock to 12 MHz (via the on-chip USB oscillator). If CLK_SEL = 1, the bootloader bypasses the system clock generator altogether and the system clock is driven by the CLKIN pin. In this case, the CLKIN frequency is expected to be 11.2896, 12.0, 12.288, 16.8, or 19.2 MHz. While the bootloader tries to boot from the USB, the clock generator is programmed to multiply the input clock by 3 and adjust TIMER0 setting for 200 ms.

1.4.3 Configuration

1.4.3.1 BYPASS MODE

When the system clock generator is in the BYPASS MODE, the clock generator is not used and the system clock (SYSCLK) is driven by either the CLKIN pin or the USB oscillator.

NOTE: In bypass mode, the PLL is not automatically powered down and will still consume power. For maximum power savings, the PLL should be placed in its power-down mode. See Section 1.4.2.2 for more details.

1.4.3.1.1 Entering and Exiting the BYPASS MODE

To enter the bypass mode, write a 0 to the SYSCLKSEL bit in the clock configuration register 2 (CCR2). In bypass mode, the frequency of the system clock (SYSCLK) is determined by the CLK_SEL pin. If CLK_SEL = 0, SYSCLK is driven by the output of the USB oscillator. Otherwise, SYSCLK will be driven by the CLKIN pin.

To exit the BYPASS MODE, ensure the PLL has completed its phase-locking sequence by waiting at least 4 ms and then write a 1 to the SYSCLKSEL bit. The frequency of SYSCLK will then be determined by the multiplier and divider ratios of the PLL System Clock Generator.

If the clock generator is in the PLL MODE and you want to reprogram the PLL or any of the dividers, you must set the clock generator to BYPASS MODE before changing the PLL and divider settings.

Logic within the clock generator ensures that there are no clock glitches during the transition from PLL MODE to BYPASS MODE and vice versa.

1.4.3.1.2 Register Bits Used in the BYPASS MODE

Table 1-6 describes the bits of the clock generator control registers that are used in the BYPASS MODE. For detailed descriptions of these bits, see Section 1.4.4.

Table 1-6. Clock Generator Control Register Bits Used In BYPASS MODE

Register Bit	Role in BYPASS MODE
SYSCLKSEL	Allows you to switch to the PLL or BYPASS MODES.
PLLPWRDN	Allows you to power down the PLL.

1.4.3.1.3 Setting the System Clock Frequency In the BYPASS MODE

In the BYPASS MODE, the frequency of SYSCLK is determined by the CLK_SEL pin. If CLK_SEL = 0, SYSCLK is driven by the output of the USB oscillator. Otherwise, SYSCLK will be driven by the CLKIN pin.

NOTE: The CLK_SEL pin must be statically tied high or low; it cannot be changed after the device has been powered up.

Table 1-7. Output Frequency in Bypass Mode

CLK_SEL	SYSCLK Source / Frequency
1	CLKIN, expected to be one of the following values by the bootloader: 11.2896, 12.0, 12.288, 16.8, or 19.2 MHz
0	USB clock = 12 MHz

The state of the CLK_SEL pin is read via the CLKSELSTAT bit in the CCR2 register.

1.4.3.2 PLL MODE

In PLL MODE, the frequency of the input clock signal (CLKREF) can be both multiplied and divided to produce the desired output frequency, and the output clock signal is phase-locked to the input clock signal.

1.4.3.2.1 Entering and Exiting the PLL MODE

To enter the PLL_MODE from BYPASS_MODE, first program the PLL to the desired frequency. You must always ensure the PLL has completed its phase-locking sequence before switching to PLL MODE. This PLL has no lock indicator as such indicators are notoriously unreliable. Instead, a fixed amount of time must be allowed to expire while in BYPASS_MODE to allow the PLL to lock. After 4 ms, write a 1 to the SYSCLKSEL bit in the clock configuration register 2 (CCR2) to set the system clock to the output of the PLL.

Whenever PLL needs to be reprogrammed, first the clock generator must be in bypass mode, and then changed to PLL configuration. After waiting 4 ms, write a 1 to the SYSCLKSEL bit to get into the PLL MODE.

Logic within the clock generator ensures that there are no clock glitches during the transition from BYPASS MODE to PLL MODE and vice versa.

1.4.3.2.2 Register Bits Used in the PLL Mode

Table 1-8 describes the bits of the clock generator control registers that are used in the PLL MODE. For detailed descriptions of these bits, see Section 1.4.4.

Table 1-8. Clock Generator Control Register Bits Used In PLL Mode

Register Bit	Role in Bypass Mode
SYSCLKSEL	Allows you to switch to the PLL or bypass modes.
RD	Specifies the divider ratio of the reference divider.
PLLM	Specify the multiplier value for the PLL.
OD	Specify the output divider line.
OutDiv2 Bypass	Determines whether the second output divider is bypassed.
OD2	Specifies the divider ratio of the second output divider.

1.4.3.2.3 Frequency Ranges for Internal Clocks

There are specific minimum and maximum frequencies for all the internal clocks.

For frequencies, see the datasheet: *TMS320C5517 Fixed-Point Digital Signal Processor* (literature number SPRS727).

1.4.3.2.4 Setting the Output Frequency for the PLL MODE

The clock generator output frequency configured based on the settings programmed in the clock generator control registers. The output frequency depends on primarily on three factors: the reference divider value, the PLL multiplier value, and the output divider value (see Figure 1-4). Based on the register settings controlling these divider and multiplier values, you can calculate the frequency of the output clock using the formulas listed in Figure 1-5.

Follow these steps to determine the values for the different dividers and multipliers of the system clock generator:

1. With the desired clock frequency in mind, choose a VCO Output frequency that falls within the range listed in the datasheet: *TMS320C5517 Fixed-Point Digital Signal Processor* (literature number SPRS727). Keep in mind that you can use the programmable output divider to divide the output frequency of the PLL.
2. Determine the divider ratio for the reference divider that will generate the PLLIN frequency that meets the requirements listed in the datasheet: *TMS320C5517 Fixed-Point Digital Signal Processor* (literature number SPRS727). When possible, choose a high value for PLLIN to optimize PLL performance. If the DSP is being clocked by the USB oscillator output, the PLL must be bypassed by setting SYSCLKSEL to 1 and CCR2 and CLK_SEL each to 0.
3. Determine a multiplier value that generates the desired VCO output frequency before output divider OD and OD2.
4. Using the OD and OD2, figure out the values for SYSCLK.

Table 1-9 shows programming examples for different PLL MODE frequencies.

Table 1-9. Examples of Selecting a PLL MODE Frequency, When CLK_SEL=L

USB OSC CLK	RD	PLLM		OD	OD2	PLL Output Frequency
		Hex	Decimal			
12 MHz	1	1456h	5206	15	0	4 MHz
12 MHz	1	1700h	5888	2	0	24 MHz
12 MHz	1	1800h	6144	0	0	75 MHz
12 MHz	1	3456h	13398	0	0	160 MHz
12 MHz	1	3955h	14677	0	0	175 MHz
12 MHz	1	41AAh	16810	0	0	200 MHz

Table 1-10. Examples of Selecting a PLL MODE Frequency, When CLK_SEL=H

USB OSC CLK	RD	PLLM		OD	OD2	PLL Output Frequency
		Hex	Decimal			
11.2896 MHz	1	37B0h	14256	0	0	160 MHz
		3D01h	15617			175 MHz
12.288 MHz	1	3315h	13077	0	0	160 MHz
		37F7h	14327			175 MHz
16.8 MHz	2	3824h	14372	0	0	160 MHz
		3D80h	15744			175 MHz
19.2 MHz	2	3100h	12544	0	0	160 MHz
		35B0h	13744			175 MHz

1.4.3.2.5 Lock Time

As previously discussed, you must place the clock generator in bypass mode before changing the PLL settings. The time it takes the PLL to complete its phase-locking sequence is referred to as the lock time. The PLL has a lock time of 4 ms. Software is responsible for ensuring the PLL remains in BYPASS_MODE for at least 4 ms before switching to PLL_MODE.

1.4.3.2.6 Initializing PLL Mode From PLL Power Down

If the PLL is powered down (PLLPWRDN (bit 13) is set to 1 in PLL Control Register (1C22h)), perform the following procedure to initialize the PLL:

1. The PLL should be held in reset and in bypass mode. If not:
   (a) Set SYSCLKSEL (bit 0) to 0 in Clock Configuration Register 2 (1C1Fh) to switch the PLL to bypass mode.
   (b) Wait 4 clock cycles to ensure that the PLL has switched to bypass mode.
   (c) Set PLLRST (bit 14) to 1 in PLL Control Register.
2. Clear PLLPWRDN in PLL Control Register to power up the analog circuitry in the PLL.
3. Program the desired values for:
   - Multiplier value (bit 15) and reference divider (bits 5–0) in PLL Input Control Register (1C21h)
   - Output Divider (bits 2–0) and Output Divider2 (bits 15–11) in PLL Output Divider Control Register (1C23h)
4. Clear the PLL reset bit in PLL Control Register.
5. Wait for the minimum lock time.
6. Clear SYSCLKSEL in Clock Configuration Register 2 to remove the PLL from bypass mode.

1.4.3.2.7 Changing PLL Multiplier

If the PLL is not powered down (PLL PWRDN (bit 13) is set to 0 in PLL Control Register (1C22h)), perform the following procedure to change the PLL:

1. Set SYSCLKSEL (bit 0) to 0 in Clock Configuration Register 2 (1C1Fh) to switch the PLL to bypass mode.
2. Wait 4 clock cycles to ensure that the PLL has switched to bypass mode.
3. Set PLLRST (bit 14) to 1 in PLL Control Register.
4. Program the desired values for:
   - Multiplier value (bit 15) and reference divider (bits 5–0) in PLL Input Control Register (1C21h)
   - Output Divider (bits 2–0) and Output Divider2 (bits 15–11) in PLL Output Divider Control Register (1C23h)
5. Clear the PLL reset bit in PLL Control Register.
6. Wait for the minimum lock time.
7. Clear SYSCLKSEL in Clock Configuration Register 2 to remove the PLL from bypass mode.

1.4.4 Clock Generator Registers

Table 1-11 lists the registers associated with the clock generator of the DSP. The clock generator registers can be accessed by the CPU at the 16-bit addresses specified in Table 1-11. Note that the CPU accesses all peripheral registers through its I/O space. All other register addresses not listed in Table 1-11 should be considered as reserved locations and the register contents should not be modified.

Table 1-11. Clock Generator Registers

CPU Word Address	Acronym	Register Description	Section
1C20h	PMR	PLL Multiplier Register	Section 1.4.4.1
1C21h	PICR	PLL Input Control Register	Section 1.4.4.2
1C22h	PCR	PLL Control Register	Section 1.4.4.3
1C23h	PODCR	PLL Ouput Divider Control Register	Section 1.4.4.4
1C1Eh	CCR1	Clock Configuration Register 1	Section 1.4.4.5
1C1Fh	CCR2	Clock Configuration Register 2	Section 1.4.4.6

1.4.4.1 PLL Multiplier Register (PMR) [1C20h]

The PLL multiplier register (PMR) is shown in Figure 1-7 and described in Table 1-12.

Figure 1-7. PLL Multiplier Register (PMR) [1C20h]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLLM15_0															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-12. PLL Multiplier Register (PMR) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	PLLM15_0	R/W	0h	PLL Multiplier Multiplier value = PLLM[16:0]/256+1. PLLM[16] bit is in the PLL Input Control Register [1C21h]. The VCO must run between 125 and 625 MHz.

1.4.4.2 PLL Input Control Register (PICR) [1C21h]

The PLL input control register (PICR) is shown in Figure 1-8 and described in Table 1-13.

Figure 1-8. PLL Input Control Register (PICR) [1C21h]

15	14	13	12	11	10	9	8
PLLM16	Reserved						
R/W-0h	R-0h						

7	6	5	4	3	2	1	0
Reserved				RD			
R-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-13. PLL Input Control Register (PICR) Field Descriptions

Bit	Field	Type	Reset	Description
15	PLLM16	R/W	0h	PLL Multiplier Multiplier value = PLLM[16:0]/256+1 The VCO must run between 125 and 625 MHz.
14-6	Reserved	R	0h	Reserved.
5-0	RD	R/W	0h	Reference input frequency divider The frequency inputted to the Phase Detector must fall between 1.7 MHz and 6.79 MHz. Divide the input clock rate to the PLL by the following number: 000000b = Divide-by-1 000001b = Divide-by-2 … 111111b = Divide-by-64

1.4.4.3 PLL Control Register (PCR) [1C22h]

The PLL control register (PCR) is shown in Figure 1-9 and described in Table 1-14.

Figure 1-9. PLL Control Register (PCR) [1C22h]

15	14	13	12	11	10	9	8
PLLBYP	PLLRST	PLLPWRDN	Reserved				
R/W-1h	R/W-1h	R/W-1h	R-0h				

7	6	5	4	3	2	1	0
Reserved							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-14. PLL Control Register (PCR) Field Descriptions

Bit	Field	Type	Reset	Description
15	PLLBYP	R/W	1h	Bypass PLL. 0x0 = PLL is enable 0x1 = PLL is bypassed
14	PLLRST	R/W	1h	Resets the PLL and clears the PLL Multiplier and Reference dividers. 0x0 = PLL is in normal operating mode 0x1 = PLL is reset
13	PLLPWRDN	R/W	1h	PLL power down. 0x0 = PLL is powered 0x1 = PLL is powered down
12-0	Reserved	R	0h	Reserved.

1.4.4.4 PLL Ouput Divider Control Register (PODCR) [1C23h]

The PLL Ouput Divider Control Register (PODCR) is shown in Figure 1-10 and described in Table 1-15.

Figure 1-10. PLL Ouput Divider Register (PODCR) [1C23h]

15	14	13	12	11	10	9	8
OD2					Reserved		
R/W-0h					R-0h		

7	6	5	4	3	2	1	0
Reserved		OUTDIV2BY	Reserved		OD		
R-0h		R/W-0h	R-0h		0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-15. PLL Ouput Divider Control Register (PODCR) Field Descriptions

Bit	Field	Type	Reset	Description
15-11	OD2	R/W	0h	Output Divider 2 value. 00000b = Divide-by-2 00001b = Divide-by-4 00010b = Divide-by-6 00011b = Divide-by-8 ... 11111b = Divide-by-64
10-6	Reserved	R	0h	Reserved
5	OUTDIV2BY	R/W	0h	Output Divider 2 Bypass. Bypasses the second output divider 0x0 = Output divider 2 is enabled 0x1 = Output divider 2 is bypassed

Table 1-15. PLL Ouput Divider Control Register (PODCR) Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4-3	Reserved	R	0h	Reserved.
2-0	OD		0h	Output Divider value. 000b = Divide-by-1 001b = Divide-by-2. and 111b = Divide-by-8

1.4.4.5 Clock Configuration Register 1 (CCR1) [1C1Eh]

The clock configuration register 1 (CCR1) is shown in Figure 1-11 and described in Table 1-16.

Figure 1-11. Clock Configuration Register 1 (CCR1) [1C1Eh]

15	1	0
Reserved		SDCLK_EN
R-0		R/W-0

LEGEND: R = Read only; -*n* = value after reset

Table 1-16. Clock Configuration Register 1 (CCR1) Field Descriptions

Bit	Field	Value	Description
15-1	Reserved	0	Reserved. This bit must be kept as 0 during writes to this register.
0	SDCLK_EN		SDRAM clock enable control. When ON, the EM_SDCLK pin will drive the clock signal at the SYSCLK frequency if in full_rate mode or at SYSCLK frequency divided by 2 if in half_rate mode. When OFF, the EM_SDCLK pin will drive low. Transitions from ON to OFF and OFF to ON are not guaranteed to be glitchless. Therefore, the EMIF should be reset after any change.
		0	EM_SDCLK off (default)
		1	EM_SDCLK on. This bit must be set to 1 before using SDRAM or mSDRAM.

1.4.4.6 Clock Configuration Register 2 (CCR2) [1C1Fh]

The clock configuration register 2 (CCR2) is shown in Figure 1-12 and described in Table 1-17.

Figure 1-12. Clock Configuration Register 2 (CCR2) [1C1Fh]

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		SYSCLKSRC		TIMER0CLKSEL	CLKSELSTAT	Reserved	SYSCLKSEL
R-0h		R-0h		R-0h	R-0h	R-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 1-17. Clock Configuration Register 2 (CCR2) Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	Reserved.
5-4	SYSCLKSRC	R	0h	System clock source bits. These read-only bits reflect the source for the system clock. 0x0 = The system clock generator is in bypass mode; SYSCLK is driven by the RTC oscillator output. 0x1 = The system clock generator is in lock mode; the RTC oscillator output provides the input clock. 0x2 = The system clock generator is in bypass mode; SYSCLK is driven by CLKIN. 0x3 = The system clock generator is in lock mode; the CLKIN pin provides the input clock.
3	TIMER0CLKSEL	R	0h	Timer0 clock source select bit. This bit specifies the input clock to Timer0. 0x0 = Timer0 uses SYSCLK as input clock. 0x1 = Timer0 uses PLL output clock (PLLOUT) as input clock.
2	CLKSELSTAT	R	0h	CLK_SEL pin status bit. This reflects the state of the CLK_SEL pin. 0x0 = CLK_SEL pin is low (RTC input clock selected). 0x1 = CLK_SEL pin is high (CLKIN input clock selected.)
1	Reserved	R	0h	Reserved.
0	SYSCLKSEL	R/W	0h	System clock source select bit. This bit is used to select between the two main clocking modes for the DSP: bypass and lock mode. In bypass mode, the DSP clock generator is byassed and the system clock is set to either CLKIN or the RTC output (as determined by the CLKSEL pin). In lock mode, the system clock is set to the output of the DSP clock generator. Logic in the system clock generator prevents switching from bypass mode to lock mode if the PLL is powered down. After settings SYSCLKSEL = 1, you can check the SYSCLKSRC bits to determine if the system clock generator is in lock mode. 0x0 = Bypass mode is selected. 0x1 = Lock mode is selected.

1.5 Power Management

1.5.1 Overview

In many applications there may be specific requirements to minimize power consumption for both power supply (and battery) and thermal considerations. There are two components to power consumption: active power and leakage power. Active power is the power consumed to perform work and, for digital CMOS circuits, scales roughly with clock frequency and the amount of computations being performed. Active power can be reduced by controlling the clocks in such a way as to either operate at a clock frequency just high enough to complete the required operation in the required time-line or to run at a high enough clock frequency until the work is complete and then drastically cut the clocks (that is, to bypass mode or clock gate) until additional work must be performed.

Leakage power is due to static current leakage and occurs regardless of the clock rate. Leakage, or standby power, is unavoidable while power is applied and scales roughly with the operating junction temperatures. Leakage power can only be avoided by removing power completely.

The DSP has several means of managing the power consumption, as detailed in the following sections. There is extensive use of automatic clock gating in the design as well as software-controlled module clock gating to not only reduce the clock tree power, but to also reduce module power by freezing its state while not operating. Clock management enables you to slow the clocks down on the chip in order to reduce switching power. Independent power domains allow you to shut down parts of the DSP to reduce static power consumption. When not being used, the internal memory of the DSP can also be placed in a low leakage power mode while preserving the memory contents. The operating voltage and drive strength of the I/O pins can also be reduced to decrease I/O power consumption.

Table 1-18 summarizes all of the power management features included in the DSP.

Table 1-18. Power Management Features

Power Management Features	Description
Clock Management	
PLL power-down	The system PLL can be powered-down when not in use to reduce switching and bias power.
Peripheral clock idle	Peripheral clocks can be idled to reduce switching power.
Dynamic Power Management	
Core Voltage Scaling	The DSP logic support two voltage ranges to allow voltage adjustments on-the-fly, increasing voltage during peak processing power demand and decreasing during low demand.
Static Power Management	
DARAM/SARAM low power modes	The internal memory of the DSP can be placed in a low leakage power mode while preserving memory contents.
Independent power domains	DSP Core (CV_{DD}) and USB Core (USB_V_{DD1P3}, USB_V_{DDA1P3}) can be shut off while other supplies remain powered.
I/O Management	
I/O voltage selection	The operating voltage and/or slew rate of the I/O pins can be reduced (at the expense of performance) to decrease I/O power consumption.
USB power-down	The USB peripheral can be powered-down when not being used.

1.5.2 Power Domains

The DSP has separate power domains which provide power to different portions of the device. The separate power domains allow the user to select the optimal voltage to achieve the lowest power consumption at the best possible performance. Note that several power domains have similar voltage requirements and, therefore, could be grouped under a single voltage domain.

Table 1-19. DSP Power Domains

Power Domains	Description
Real-Time Clock Power Domain (CV_{DDRTC})	This domain powers the real-time clock digital circuits and oscillator pins (RTC_XI, RTC_XO). Nominal supply voltage can be 1.05 V. The CV_{DDRTC} pin must always be powered by an external power source even though RTC is not used. None of the on-chip LDOs can power CV_{DDRTC}.
Core Power Domain (CV_{DD})	This domain powers the digital circuits that include the C55x CPU, on-chip memory, and peripherals. Nominal supply voltage is either 1.05 V, 1.3 V, or 1.4 V.
Digital I/O Power Domain 1 (DV_{DDEMIF})	This domain powers all EMIF I/O only. Nominal supply voltage can be 1.8, 2.75, or 3.3 V.
Digital I/O Power Domain 2 (DV_{DDIO})	This domain powers all I/Os, except the EMIF I/O, USB I/O, USB oscillator I/O, some of the analog related digital pins, and the real-time clock power domain I/O. Nominal supply voltage can be 1.8, 2.75, or 3.3 V.
RTC I/O Power Domain (DV_{DDRTC})	This domain powers the WAKEUP and RTC_CLKOUT pins. Nominal supply voltage can be 1.8, 2.75, or 3.3 V.
PLL Power Domain (V_{DDA_PLL})	This domain powers the system clock generator PLL. Nominal supply voltage is 1.3 V. This domain must be powered externally, and it typically consumes ~5 mA at 300 MHz.
Analog Power Domain (V_{DDA_ANA})	This domain powers the power management analog circuits and the 10-bit SAR. Nominal supply voltage is 1.3 V. This domain can be powered from the on-chip analog LDO output pin (ANA_LDOO). **Note:** When externally powered, this domain must always be powered for proper operation.
USB Analog Power Domain (USB_V_{DDA1P3})	This domain powers the USB analog PHY. Nominal supply voltage is 1.3 V.
USB Digital Power Domain (USB_V_{DD1P3})	This domain powers the USB digital module. Nominal supply voltage is 1.3 V. **Note:** This domain must always be powered for proper operation when the on-chip USB oscillator is used to clock the PLL.
USB Oscillator Power Domain (USB_V_{DDOSC})	This domain powers the USB oscillator. Nominal supply voltage is 3.3 V. **Note:** This domain must always be powered for proper operation when the on-chip USB oscillator is used to clock the PLL.
USB Transceiver & Analog Power Domain (USB_V_{DDA3P3})	This domain powers the USB transceiver. Nominal supply voltage is 3.3 V.
USB PLL Power Domain (USB_V_{DDPLL})	This domain powers the USB PLL. Nominal supply voltage is 3.3 V.
LDOI Power Domain (LDOI)	This domain powers LDO, POR comparator, and I/O supply for some pins. Nominal supply voltage is 1.8 V through 3.6 V. **Note:** This domain must be always powered for proper operation.

1.5.3 Clock Management

As mentioned in Section 1.3.2, there are several clock domains within the DSP. The device supports clock gating features that allows software to disable clocks to entire clock domains or modules within a domain in order to reduce the domain's active power consumption to very-near zero (a very small amount of logic will still see a clock).

There are two distinct methods of clock gating. The first uses the ICR CPU register and the CPU's IDLE instruction. This method is used for the following domains: CPU, IPORT, DPORT, MPORT, XPORT & HWA. See Figure 1-3 for a diagram of these domains. In this method, the ICR is written with a value indicating the desired clock gating configuration and then (possibly much later) the IDLE instruction is executed. The contents of the ICR do not become effective until the IDLE instruction is executed. The second method uses system registers, PCGCR1 & PCGCR2. These registers control most of the peripheral clock domains and writes to this register take effect immediately.

The SYSCLKDIS bit in the PCGCR1 register has global effect and, therefore, is a superset of the two methods. When this bit as asserted the whole device is clock gated with the exceptions of the PLL, the USB PLL, the RTC, and the oscillators.

NOTE: Stopping clocks to a domain or a module within that domain only affects active power consumption; it does not affect leakage power consumption.

NOTE: The on-chip Bootloader idles all peripherals and CPU ports at startup, but it enables some peripherals as it uses them. Application code should not assume all peripherals and CPU ports are disabled. To get the minimum power consumption, make sure to disable all peripherals and CPU ports first and then enable only necessary peripherals and CPU ports before using them.

1.5.3.1 CPU Domain Clock Gating

Two registers are provided to individually configure and monitor the clock gating modes of the CPU domain: the idle configuration register (ICR) and the idle status register (ISTR).

ICR lets you configure how the CPU domain will respond the next time the idle instruction is executed. When you execute the idle instruction, the content of ICR is copied to ISTR. Then the ISTR values are propagated to the different portions of the CPU domain.

In the CPU domain, there are the following CPU ports:

- IPORT: this port is used by the CPU for fetching instructions from external memory.
- DPORT: this port is used by the CPU when reading and writing data from/to external memory.
- XPORT: this port is used by the CPU when reading and writing from/to IO-space (peripheral) registers.
- MPORT: this port is used by the four DMAs and the USB's CDMA when accessing SARAM or DARAM.
- HWA: this port is the hardware accelerator (FFT coprocessor). It shares all CPU buses.

1.5.3.1.1 Idle Configuration Register (ICR) [0001h] and IDLE Status Register (ISTR) [0002h]

Table 1-20 describes the read/write bits of ICR, and Table 1-21 describes the read-only bits of ISTR.

NOTE: To prevent an emulation lock up, idle requests to these domains may be overridden or ignored when an emulator is connected to the JTAG port of the DSP.

Figure 1-13. Idle Configuration Register (ICR) [0001h]

15				10	9	8
Reserved					HWAI	IPORTI
R/W-0					R/W-0	R/W-0
7	6	5	4		1	0
MPORTI	XPORTI	DPORTI	IDLECFG			CPUI
R/W-0	R/W-0	R/W-0	R/W-0			R/W-0

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 1-20. Idle Configuration Register (ICR) Field Descriptions

Bit	Field	Value	Description
15-10	Reserved	0	Reserved.
9	HWAI		FFT hardware accelerator idle control bit.
		0	Hardware accelerator remains active after execution of an IDLE instruction.
		1	Hardware accelerator is disabled after execution of an IDLE instruction.
8	IPORTI		Instruction port idle control bit. The IPORT is used for all external memory instruction accesses.
		0	IPORT remains active after execution of an IDLE instruction.
		1	IPORT is disabled after execution of an IDLE instruction.
7	MPORTI		Memory port idle control bit. The memory port is used for all DMA, and USB CDMA transactions into on-chip memory.
		0	MPORT remains active after execution of an IDLE instruction.
		1	MPORT is disabled after execution of an IDLE instruction.
6	XPORTI		I/O port idle control bit. The XPORT is used for all CPU I/O memory transactions.
		0	XPORT remains active after execution of an IDLE instruction.
		1	XPORT is disabled after execution of an IDLE instruction.
5	DPORTI		Data port idle control bit. The data port is used for all CPU external memory data accesses.
		0	DPORT remains active after execution of an IDLE instruction.
		1	DPORT is disabled after execution of an IDLE instruction.
4-1	IDLECFG	0111b	Idle configuration bits. You must always set bit 1, 2 and 3 to 1 and bit 4 to 0 before executing the idle instruction.
0	CPUI		CPU idle control bit.
		0	CPU remains active after execution of an IDLE instruction.
		1	CPU is disabled after execution of an IDLE instruction.

Figure 1-14. Idle Status Register (ISTR) [0002h]

15							10	9	8
Reserved								HWAIS	IPORTIS
R-0								R-0	R-0
7	6	5	4				1	0	
MPORTIS	XPORTIS	DPORTIS	Reserved					CPUIS	
R-0	R-0	R-0	R-0					R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 1-21. Idle Status Register (ISTR) Field Descriptions

Bit	Field	Value	Description
15-10	Reserved	0	Reserved.
9	HWAIS		FFT hardware accelerator idle status bit.
		0	Hardware accelerator is active.
		1	Hardware accelerator is disabled.
8	IPORTIS		Instruction port idle status bit. The IPORT is used for all external memory instruction accesses.
		0	IPORT is active.
		1	IPORT is disabled.
7	MPORTIS		Memory port idle status bit. The memory port is used for all DMA, and USB CDMA transactions into on-chip memory.
		0	MPORT is active.
		1	MPORT is disabled.
6	XPORTIS		I/O port idle status bit. The XPORT is used for all CPU I/O memory transactions.
		0	XPORT is active.
		1	XPORT is disabled.
5	DPORTIS		Data port idle status bit. The data port is used for all CPU external memory data accesses.
		0	DPORT is active.
		1	DPORT is disabled.
4-1	Reserved	0	Reserved.
0	CPUIS		CPU idle status bit.
		0	CPU is active.
		1	CPU is disabled.

1.5.3.1.2 Valid Idle Configurations

Not all of the values that you can write to the idle configuration register (ICR) provide valid idle configurations. The valid configurations are limited by dependencies within the system. For example, the IDLECFG bits 1, 2 and 3 of ICR must always be set to 1, and bit 4 must always be cleared to 0. As another example, the XPORT cannot be idled unless the CPU is also idled. Before any part of the CPU domain is idled, you must observe the requirements outlined in Section 1.5.3.2.

A bus error will be generated (BERR = 1 in IFR1) if you execute the idle instruction under any of the following conditions and the idle command will not take effect:

1. If you fail to set IDLECFG = 0111 while setting any of these bits: DPORTI, XPORTI, IPORTI or MPORTI.
2. If you set DPORTI, XPORTI, or IPORTI without also setting CPUI.

Table 1-22. CPU Clock Domain Idle Requirements

To Idle the Following Module/Port	Requirements Before Going to Idle
CPU	No requirements.
FFT Hardware Accelerator	No requirements.
MPORT	DMA controllers, and USB CDMA must not be accessing DARAM or SARAM.
XPORT DPORT IPORT	CPU CPUI must also be set.

1.5.3.1.3 Clock Configuration Process

The clock configuration indicates which portions of the CPU clock domain will be idle, and which will be active. The basic steps to the clock configuration process are:

1. To idle MPORT, DMA controller, and USB CDMA must not be accessing SARAM or DARAM. If any DMA is in active, wait for completion of the DMA transfer.
2. Write the desired configuration to the idle configuration register (ICR). Make sure that you use a valid idle configuration (see Section 1.5.3.1.2).
3. Apply the new idle configuration by executing the IDLE instruction. The content of ICR is copied to the idle status register (ISTR). The bits of ISTR are then propagated through the CPU domain system to enable or disable the specified clocks. If the CPU domain was idled, then program execution will stop immediately after the idle instruction. If the CPU domain was not idled, then program execution will continue past the idle instruction but the appropriate domains will be idle.

The IDLE instruction cannot be executed in parallel with another instruction.

The CPU, DPORT, XPORT, and IPORT domains are enabled automatically by any unmasked interrupts. There is a logic in the DSP core that enables CPU, DPORT, XPORT, and IPORT (clears the bits 0, 5, 6, and 8 of the ISTR register) asynchronously upon detecting an interrupt signal. Therefore, when an unmasked interrupt signal reaches the DSP core, these domains are un-idled automatically. Once the CPU is enabled, it takes 3 CPU cycles to detect the interrupt in the IFR. Note that HWA and MPORT have to be manually enabled after being disabled.

1.5.3.2 Peripheral Domain Clock Gating

The peripheral clock gating allows software to disable clocks to the DSP peripherals, in order to reduce the peripheral's active power consumption to zero. Aside from the analog logic, the DSP is designed in static CMOS; thus, when a peripheral clock stops, the peripheral's state is preserved, and no active current is consumed. When the clock is restarted the peripheral resumes operating from the stopping point.

NOTE: Stopping clocks to a peripheral only affects active power consumption; it does not affect leakage power consumption.

If a peripheral's clock is stopped while being accessed, the access may not occur completely, and could potentially lock-up the device. To avoid this issue, some peripherals have a clock stop request and acknowledge protocol that allows software to ask the peripheral when it is safe to stop the clocks. This is described further in Section 1.5.3.2.2. For the peripherals that do not have the request/acknowledge protocol, the user must ensure that all of the transactions to the peripheral are finished prior to stopping the clocks.

The procedure to turn peripheral clocks on/off is described in Section 1.5.3.2.3.

Some peripherals provide additional power saving features by clock gating components within its peripheral boundary. See the peripheral-specific chapter for more details on these additional power saving features.

1.5.3.2.1 Peripheral Clock Gating Configuration Registers (PCGCR1 and PCGCR2) [1C02 - 1C03h]

The peripheral clock gating configuration registers (PCGCR1 and PCGCR2) are used to disable the clocks of the DSP peripherals. In contrast to the idle control register (ICR), these bits take effect within 6 SYSCLK cycles and do not require an idle instruction.

The peripheral clock gating configuration register 1 (PCGCR1) is shown in Figure 1-15 and described in Table 1-23.

Figure 1-15. Peripheral Clock Gating Configuration Register 1 (PCGCR1) [1C02h]

15	14	13	12	11	10	9	8
SYSCLKDIS	I2S2CG	TMR2CG	TMR1CG	EMIFCG	TMR0CG	MCSPICG	I2S0CG
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
MMCSD1CG	I2CCG	MCBSPCG	MMCSD0CG	DMA0CG	UARTCG	SPICG	I2S3CG
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-23. Peripheral Clock Gating Configuration Register 1 (PCGCR1) Field Descriptions

Bit	Field	Type	Reset	Description
15	SYSCLKDIS	R/W	0h	System clock disable bit. This bit can be used to turn off the system clock. Setting the WAKEUP pin high enables the system clock. Since the WAKEUP pin is used to re-enable the system clock, the WAKEUP pin must be low to disable the system clock. NOTE: Disabling the system clock disables the clock to most parts of the device, including the CPU. 0x0 = System clock is active. 0x1 = System clock is disabled.
14	I2S2CG	R/W	0h	I2S2 clock gate control bit. This bit is used to enable and disable the I2S2 peripheral clock. 0x0 = Peripheral clock is active. 0x1 = Peripheral clock is disabled.
13	TMR2CG	R/W	0h	Timer 2 clock gate control bit. This bit is used to enable and disable the Timer 2 peripheral clock. 0x0 = Peripheral clock is active. 0x1 = Peripheral clock is disabled.
12	TMR1CG	R/W	0h	Timer 1 clock gate control bit. This bit is used to enable and disable the Timer 1 peripheral clock. 0x0 = Peripheral clock is active. 0x1 = Peripheral clock is disabled.
11	EMIFCG	R/W	0h	EMIF clock gate control bit. This bit is used to enable and disable the EMIF peripheral clock. NOTE: You must request permission before stopping the EMIF clock through the peripheral clock stop request/acknowledge register (CLKSTOP). 0x0 = Peripheral clock is active. 0x1 = Peripheral clock is disabled.
10	TMR0CG	R/W	0h	Timer 0 clock gate control bit. This bit is used to enable and disable the Timer 0 peripheral clock. 0x0 = Peripheral clock is active. 0x1 = Peripheral clock is disabled.
9	MCSPICG	R/W	0h	MCSPI clock gate control bit. This bit is used to enable and disable the MCSPI peripheral clock. 0x0 = Peripheral clock is active. 0x1 = Peripheral clock is disabled.

Table 1-23. Peripheral Clock Gating Configuration Register 1 (PCGCR1) Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8	I2S0CG	R/W	0h	I2S0 clock gate control bit. This bit is used to enable and disable the I2S0 peripheral clock. 0x0 = Peripheral clock is active. 0x1 = Peripheral clock is disabled.
7	MMCSD1CG	R/W	0h	MMC/SD1 clock gate control bit. This bit is used to enable and disable the MMC/SD1 peripheral clock. 0x0 = Peripheral clock is active. 0x1 = Peripheral clock is disabled.
6	I2CCG	R/W	0h	I2C clock gate control bit. This bit is used to enable and disable the I2C peripheral clock. 0x0 = Peripheral clock is active. 0x1 = Peripheral clock is disabled.
5	MCBSPCG	R/W	0h	McBSP clock gate control bit. This bit is used to enable the disable the McBSP peripheral clock 0x0 = Peripheral clock is active. 0x1 = Peripheral clock is disabled.
4	MMCSD0CG	R/W	0h	MMC/SD0 clock gate control bit. This bit is used to enable and disable the MMC/SD0 peripheral clock. 0x0 = Peripheral clock is active. 0x1 = Peripheral clock is disabled.
3	DMA0CG	R/W	0h	DMA controller 0 clock gate control bit. This bit is used to enable and disable the peripheral clock the DMA controller 0. 0x0 = Peripheral clock is active. 0x1 = Peripheral clock is disabled.
2	UARTCG	R/W	0h	UART clock gate control bit. This bit is used to enable and disable the UART peripheral clock. NOTE: You must request permission before stopping the UART clock through the peripheral clock stop request/acknowledge register (CLKSTOP). 0x0 = Peripheral clock is active. 0x1 = Peripheral clock is disabled.
1	SPICG	R/W	0h	SPI clock gate control bit. This bit is used to enable and disable the SPI controller peripheral clock. 0x0 = Peripheral clock is active. 0x1 = Peripheral clock is disabled.
0	I2S3CG	R/W	0h	I2S3 clock gate control bit. This bit is used to enable and disable the I2S3 peripheral clock. 0x0 = Peripheral clock is active. 0x1 = Peripheral clock is disabled.

The peripheral clock gating configuration register 2 (PCGCR2) is shown in Figure 1-16 and described in Table 1-24.

Figure 1-16. Peripheral Clock Gating Configuration Register 2 (PCGCR2) [1C03h]

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
McSPISPIREFCG	ANAREGCG	DMA3CG	DMA2CG	DMA1CG	USBCG	SARCG	UHPICG
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-24. Peripheral Clock Gating Configuration Register 2 (PCGCR2) Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0	Reserved.
7	McSPISPIREFCG	R/W	0	McSPI Bridge Clock Gate Control 0 = McSPI Bridge module is active. 1 = McSPI Bridge module is disabled.
6	ANAREGCG	R/W	0	Analog registers clock gate control bit. This bit is used to enable and disable the clock to the registers that control the analog domain of the device, i.e. registers in the 7000h-70FFh I/O space address range. Note: When SARCG = 0, the clocks to the analog domain registers are enabled regardless of the ANAREGCG setting. 0 = Clock is active. 1 = Clock is disabled.
5	DMA3CG	R/W	0	DMA controller 3 clock gate control bit. This bit is used to enable and disable the DMA controller 3 peripheral clock. 0 = Peripheral clock is active. 1 = Peripheral clock is disabled.
4	DMA2CG	R/W	0	DMA controller 2 clock gate control bit. This bit is used to enable and disable the DMA controller 2 peripheral clock. 0 = Peripheral clock is active. 1 = Peripheral clock is disabled.
3	DMA1CG	R/W	0	DMA controller 1 clock gate control bit. This bit is used to enable and disable the DMA controller 1 peripheral clock. 0 = Peripheral clock is active. 1 = Peripheral clock is disabled.
2	USBCG	R/W	0	USB clock gate control bit. This bit is used to enable and disable the USB controller peripheral clock. Note: You must request permission before stopping the USB clock through the peripheral clock stop request and acknowledge register (CLKSTOP). This register does not stop the USB PLL. 0 = Peripheral clock is active. 1 = Peripheral clock is disabled.
1	SARCG	R/W	0	SAR clock gate control bit. This bit is used to enable and disable the SAR peripheral clock. Note: When SARCG = 0, the clock to the analog domain registers is enabled regardless of the ANAREGCG setting. 0 = Peripheral clock is active. 1 = Peripheral clock is disabled.
0	UHPICG	R/W	0	UHPI clock gate control bit. This bit is used to enable and disable the UHPI peripheral clock. 0 = Peripheral clock is active. 1 = Peripheral clock is disabled.

1.5.3.2.2 Peripheral Clock Stop Request/Acknowledge Register (CLKSTOP1 and CLKSTOP2) [1C3Ah and 1C3Bh]

You must execute a handshaking procedure before stopping the clock to the EMIF, USB, McBSP, McSPI, UHPI, and UART. This handshake procedure ensures that current bus transactions are completed before the clock is stopped. The peripheral clock stop request/acknowledge register (CLKSTOP) enables this handshaking mechanism.

To stop the clock to the EMIF or USB, set the corresponding clock stop request bit in the CLKSTOP register, then wait for the peripheral to set the corresponding clock stop acknowledge bit. Once this bit is set, you can idle the corresponding clock in the PCGCR1 and PCGCR2.

To stop the clock to the UHPI, McBSP, or UART, set the corresponding clock stop request bit in the CLKSTOP register, then wait for the peripheral to set the corresponding clock stop acknowledge bit. Once this bit is set, the corresponding clock is also idle at the same time. Setting the corresponding clock gating in the PCGCR1 and PCGCR2 is not required.

To enable the clock to the EMIF, UHPI, McBSP, McSPI, USB, or UART, first enable the clock to the peripheral through PCGCR1 or PCGCR2, then clear the corresponding clock stop request bit in the CLKSTOP register.

The peripheral clock stop request/acknowledge registers (CLKSTOP1 and CLKSTOP2) are shown in Figure 1-17 and Figure 1-18 and described in Table 1-25 and Table 1-26.

Figure 1-17. Peripheral Clock Stop Request/Acknowledge Register 1 (CLKSTOP1) [1C3Ah]

15	14	13	12	11	10	9	8
Reserved				UHPICLKSTPACK	UHPICLKSTPREQ	Reserved	
R-0h				R-0h	R/W-0h	R-0h	

7	6	5	4	3	2	1	0
MBPCLKSTPACK	MBPCLKSTPREQ	URTCLKSTPACK	URTCLKSTPREQ	USBCLKSTPACK	USBCLKSTPREQ	EMIFCLKSTPACK	EMIFCLKSTPREQ
R-0h	R/W-0h	R-0h	R/W-0h	R-0h	R/W-0h	R-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-25. Peripheral Clock Stop Request/Acknowledge Register 1 (CLKSTOP1) Field Descriptions

Bit	Field	Type	Reset	Description
15-12	Reserved	R	0h	Reserved.
11	UHPICLKSTPACK	R	0h	UHPI clock stop acknowledge bit. This bit is set to 1 when the UHPI has acknowledged a request for its clock to be stopped. The UHPI clock should not be stopped until this bit is set to 1. 0x0 = The request to stop the peripheral clock has not been acknowledged. 0x1 = The request to stop the peripheral clock has been acknowledged, the clock can be stopped.
10	UHPICLKSTPREQ	R/W	0h	UHPI peripheral clock stop request bit. When disabling the UHPI internal peripheral clock, you must set this bit to 1 to request permission to stop the clock. After the UHPI acknowledges the request (UHPICLKSTPACK = 1) you can stop the clock through the peripheral clock gating control register 2 (PCGCR2). When enabling the MCSPI internal clock, enable the clock through PCGCR1, then set UHPICLKSTPREQ to 0. 0x0 = Normal operating mode. 0x1 = Request permission to stop the peripheral clock.
9-8	Reserved	R	0h	Reserved.

Table 1-25. Peripheral Clock Stop Request/Acknowledge Register 1 (CLKSTOP1) Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7	MBPCLKSTPACK	R	0h	McBSP clock stop acknowledge bit. This bit is set to 1 when the McBSP has acknowledged a request for its clock to be stopped. The McBSP clock should not be stopped until this bit is set to 1. 0x0 = The request to stop the peripheral clock has not been acknowledged. 0x1 = The request to stop the peripheral clock has been acknowledged, the clock can be stopped.
6	MBPCLKSTPREQ	R/W	0h	McBSP peripheral clock stop request bit. When disabling the McBSP internal peripheral clock, you must set this bit to 1 to request permission to stop the clock. After the McBSP acknowledges the request (MBPCLKSTPACK = 1) you can stop the clock through the peripheral clock gating control register 1 (PCGCR1). When enabling the McBSP internal clock, enable the clock through PCGCR1, then set MBPCKLSTPREQ to 0. 0x0 = Normal operating mode. 0x1 = Request permission to stop the peripheral clock.
5	URTCLKSTPACK	R	0h	UART clock stop acknowledge bit. This bit is set to 1 when the UART has acknowledged a request for its clock to be stopped. The UART clock should not be stopped until this bit is set to 1. 0x0 = The request to stop the peripheral clock has not been acknowledged. 0x1 = The request to stop the peripheral clock has been acknowledged, the clock can be stopped.
4	URTCLKSTPREQ	R/W	0h	UART peripheral clock stop request bit. When disabling the UART internal peripheral clock, you must set this bit to 1 to request permission to stop the clock. After the UART acknowledges the request (URTCLKSTPACK = 1) you can stop the clock through the peripheral clock gating control register 1 (PCGCR1). When enabling the UART internal clock, enable the clock through PCGCR1, then set URTCKLSTPREQ to 0. 0x0 = Normal operating mode. 0x1 = Request permission to stop the peripheral clock.
3	USBCLKSTPACK	R	0h	USB clock stop acknowledge bit. This bit is set to 1 when the USB has acknowledged a request for its clock to be stopped. The USB clock should not be stopped until this bit is set to 1. 0x0 = The request to stop the peripheral clock has not been acknowledged. 0x1 = The request to stop the peripheral clock has been acknowledged, the clock can be stopped.
2	USBCLKSTPREQ	R/W	0h	USB peripheral clock stop request bit. When disabling the USB internal peripheral clock, you must set this bit to 1 to request permission to stop the clock. After the USB acknowledges the request (USBCLKSTPACK = 1) you can stop the clock through the peripheral clock gating control register 2 (PCGCR2). When enabling the USB internal clock, enable the clock through PCGCR2, then set USBCKLSTPREQ to 0. 0x0 = Normal operating mode. 0x1 = Request permission to stop the peripheral clock.
1	EMIFCLKSTPACK	R	0h	EMIF clock stop acknowledge bit. This bit is set to 1 when the EMIF has acknowledged a request for its clock to be stopped. The EMIF clock should not be stopped until this bit is set to 1. 0x0 = The request to stop the peripheral clock has not been acknowledged. 0x1 = The request to stop the peripheral clock has been acknowledged, the clock can be stopped.

Table 1-25. Peripheral Clock Stop Request/Acknowledge Register 1 (CLKSTOP1) Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	EMIFCLKSTPREQ	R/W	0h	EMIF peripheral clock stop request bit. When disabling the EMIF internal peripheral clock, you must set this bit to 1 to request permission to stop the clock. After the EMIF acknowledges the request (EMFCLKSTPACK = 1) you can stop the clock through the peripheral clock gating control register 1 (PCGCR1). When enabling the EMIF internal clock, enable the clock through PCGCR1, then set EMFCKLSTPREQ to 0. 0x0 = Normal operating mode. 0x1 = Request permission to stop the peripheral clock.

Figure 1-18. Peripheral Clock Stop Request/Acknowledge Register 2 (CLKSTOP2) [1C3Bh]

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved			MSPBRIDGECLKSTPACK	MSPBRIDGECLKSTPREQ	MSPCLKSTPACK		MSPCLKSTPREQ
R-0h			R-0h	R/W-0h	R-0h		R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-26. Peripheral Clock Stop Request/Acknowledge Register 2 (CLKSTOP2) Field Descriptions

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0h	Reserved.
4	MSPBRIDGECLKSTPACK	R	0h	MCSPI bridge clock stop acknowledge bit. This bit is set to 1 when the MCSPI bridge has acknowledged a request for its clock to be stopped. The MCSPI bridge clock should not be stopped until this bit is set to 1. 0x0 = The request to stop the peripheral clock has not been acknowledged. 0x1 = The request to stop the peripheral clock has been acknowledged, the clock can be stopped.
3	MSPBRIDGECLKSTPREQ	R/W	0h	MCSPI bridge clock stop request bit. 0x0 = Normal operating mode. 0x1 = Request permission to stop the peripheral clock.
2-1	MSPCLKSTPACK	R	0h	MCSPI peripheral clock stop acknowledge bit. This bit is set to 1 when the MCSPI has acknowledged a request for its clock to be stopped. The MCSPI clock should not be stopped until this bit is set to 1. 0x0 = The request to stop the peripheral clock has not been acknowledged. 0x1 = The request to stop the peripheral clock has been acknowledged, the clock can be stopped.
0	MSPCLKSTPREQ	R/W	0h	MCSPI peripheral clock stop request bit. When disabling the MCSPI internal peripheral clock, you must set this bit to 1 to request permission to stop the clock. After the MCSPI acknowledges the request (MSPCLKSTPACK = 1) you can stop the clock through the peripheral clock gating control register 1 (PCGCR1). When enabling the MCSPI internal clock, enable the clock through PCGCR1, then set MSPCKLSTPREQ to 0. 0x0 = Normal operating mode. 0x1 = Request permission to stop the peripheral clock.

1.5.3.2.3 Clock Configuration Process

The clock configuration indicates which portions of the peripheral clock domain will be idle, and which will be active. The basic steps to the clock configuration process are:

1. Wait for completion of all DMA transfers. You can poll the DMA transfer status and disable DMA transfers through the DMA registers.
2. If idling the EMIF, UHPI, McBSP, McSPI, USB, and UART clock, set the corresponding clock stop request bit in CLKSTOP.
3. Wait for confirmation from the module that its clock can be stopped by polling the clock stop acknowledge bits of CLKSTOP.
4. Set the clock configuration for EMIF and USB through PCGCR1 and PCGCR2. The clock configuration takes place as soon as you write to these registers; the idle instruction is not required.

Note: It is not required for UHPI, McBSP, McSPI, and UART to set the clock configuration for the peripheral domain through PCGCR1 and PCGCR2 (see Section 1.5.3.2.2).

1.5.3.3 Clock Generator Domain Clock Gating

To save power, the system clock generator can be placed in its BYPASS MODE and its PLL can be placed in power down mode. When the system clock generator is in the BYPASS MODE, the clock generator is not used and the system clock (SYSCLK) is driven by either the CLKIN pin or the USB oscillator. For more information entering and exiting the bypass mode of the clock generator, see Section 1.4.3.1.1.

When the clock generator is placed in its bypass mode, the PLL continues to generate a clock output. You can save additional power by powering down the PLL. Section 1.4.2.2 provides more information on powering down the PLL.

1.5.3.4 McSPI Domain Clock Gating

The McSPI peripheral has two peripheral clock domains: the McSPI, which interfaces to the external McSPI bus, and the McSPI bridge, which interfaces to the internal local bus. Each clock domain has its own dedicated clock stop request and acknowledge bits. In order to stop the clock to the McSPI domain, a request to stop the clock must be sent to the McSPI peripheral. Unless the current transaction is complete, the acknowledge will not be received. This ensures the current transaction is complete before the clock shuts off. Once the McSPI peripheral has received the clock stop acknowledge bit, a request to stop must be sent to the McSPI Bridge domain. This acknowledge will be immediate and the clock to the domains are also idled concurrently. It is not required to set the gating bits for the corresponding domain clock.

1.5.3.5 USB Domain Clock Gating

The USB peripheral has two clock domains. The first is a high speed domain that has its clock supplied by a dedicated USB PLL. The reference clock for the USB PLL is the 12.0 MHz USB oscillator. The clock output from the PLL must support the serial data stream that, in high-speed mode, is at a rate of 480 Mb/s. The second clock into the USB peripheral handles the data once it has been packetized and transported in parallel fashion. This clock supports all of the USB registers, CDMA, FIFO, etc., and is clocked by SYSCLK. In order to keep up with the serial data stream, the USB requires SYSCLK to be at least 30 MHz for low-speed/full-speed modes and at least 60 MHz for high-speed mode.

By stopping both of these clocks, it is possible to reduce the USB's active power consumption (in the digital logic) to zero. However, the 12.0-MHz USB oscillator clock must always be powered for proper operation when the on-chip USB oscillator is used to clock the system PLL.

NOTE: Stopping clocks to a peripheral only affects active power consumption; it does not affect leakage power consumption. USB leakage power consumption can be reduced to zero by not powering the USB.

1.5.3.5.1 Clock Configuration Process

The clock configuration process for the USB clock domain consists of disabling the USB peripheral clock followed by disabling the USB on-chip oscillator. This procedure will completely shut off USB module, which does not comply with USB suspend/resume protocol.

To set the clock configuration of the USB clock domain to idle follow these steps:

1. Set the ENSUSPM bit in the POWER register. For more information about the ENSUSPM bit, see Section 17.1.
2. Set the USB clock stop request bit (USBCLKSTREQ) in the CLKSTOP register to request permission to shut off the USB peripheral clock.
3. Wait until the USB acknowledges the clock stop request by polling the USB clock stop acknowledge bit (USBCLKSTPACK) in the CLKSTOP register.
4. Disable the USB peripheral clock by setting USBCG = 1 in the peripheral clock gating control register 2 (PCGCR2).
5. Disable the USB oscillator by setting USBOSCDIS = 1 in the USB system control register (USBSCR).

To enable the USB clock domain, follow these steps:

1. Enable the USB oscillator by setting USBOSCDIS = 0 in USBSCR.
2. Wait for the oscillator to stabilize. Refer to the device-specific data manual for oscillator stabilization time.
3. Enable the USB peripheral clock by setting USBCG = 0 in the peripheral clock gating control register 2 (PCGCR2).
4. Clear the USB clock stop request bit (USBCLKSTREQ) in the CLKSTOP register.
5. Clear the SUSPENDM bit in the POWER register.

1.5.3.5.2 USB System Control Register (USBSCR) [1C32h]

The USB system control register is used to disable the USB on-chip oscillator and to power-down the USB.

Note: The 12.0-MHz USB on-chip oscillator clock must always be powered for proper operation when the on-chip USB oscillator is used to clock the system PLL.

The USB system control register (USBSCR) is shown in Figure 1-19 and described in Table 1-27.

Figure 1-19. USB System Control Register (USBSCR) [1C32h]

15	14	13	12	11	10	9	8
USBPWDN	USBSESSEND	USBVBUSDET	USBPLLEN	Reserved			
R/W-1h	R/W-0h	R/W-1h	R/W-0h	R-0h			

7	6	5	4	3	2	1	0
Reserved	USBDATPOL	Reserved		USBOSCBIASDIS	USBOSCDIS	BYTEMODE	
R-0h	R/W-1h	R-0h		R/W-1h	R/W-1h	R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-27. USB System Control Register (USBSCR) Field Descriptions

Bit	Field	Type	Reset	Description
15	USBPWDN	R/W	1h	USB module power. Asserting USBPWDN puts the USB PHY and PLL in their lowest power state. The USB peripheral is not operational in this state. 0x0 = USB module is powered. 0x1 = USB module is powered-down.

Table 1-27. USB System Control Register (USBSCR) Field Descriptions (continued)

Bit	Field	Type	Reset	Description
14	USBSESSEND	R/W	0h	USB VBUS session end comparator enable. The USB VBUS pin has two comparators that monitor the voltage level on the pin. These comparators can be disabled for power savings when not needed. 0x0 = USB VBUS session end comparator is disabled. 0x1 = USB VBUS session end comparator is enabled.
13	USBVBUSDET	R/W	1h	USB VBUS detect enable. The USB VBUS pin has two comparators that monitor the voltage level on the pin. These comparators can be disabled for power savings when not needed. 0x0 = USB VBUS detect comparator is disabled. 0x1 = USB VBUS detect comparator is enabled.
12	USBPLLEN	R/W	0h	USB PLL enable. This is normally only used for test purposes. 0x0 = Normal USB operation. 0x1 = Override USB suspend end behavior and force release of PLL from suspend state.
11-7	Reserved	R	0h	Reserved. Always write 0 to these bits.
6	USBDATPOL	R/W	1h	USB data polarity bit. Changing this bit can be useful since the data polarity is opposite on type-A and type-B connectors. 0x0 = Reverse polarity on DP and DM signals. 0x1 = Normal polarity (normal polarity matching pin names).
5-4	Reserved	R	0h	Reserved.
3	USBOSCBIASDIS	R/W	1h	USB internal oscillator bias resistor disable. 0x0 = Internal oscillator bias resistor enabled (normal operating mode). 0x1 = Internal oscillator bias resistor disabled. Disabling the internal resistor is primarily for production test purposes. But it can also be used when an external oscillator bias resistor is connected between the USB_MXI and USB_MXO pins (but this is not a recommended configuration).
2	USBOSCDIS	R/W	1h	USB oscillator disable bit. 0x0 = USB internal oscillator enabled. 0x1 = USB internal oscillator disabled. Causes the USB_MXO pin to be tristated and the oscillator's clock into the core is forced low.
1-0	BYTEMODE	R/W	0h	USB byte mode select bits. 0x0 = Word accesses by the CPU are allowed. 0x1 = Byte accesses by the CPU are allowed (high byte is selected). 0x2 = Byte accesses by the CPU are allowed (low byte is selected). 0x3 = Reserved.

1.5.3.6 RTC Domain Clock Gating

Dynamic RTC domain clock gating is not supported. Note that the RTC oscillator, and by extension the RTC domain, can be permanently disabled by not connecting a crystal and tying off the RTC oscillator pins. However, in this configuration, the RTC must still be powered and the RTC registers starting at I/O address 1900h will not be accessible. This includes the RTC Power Management Register (RTCPMGT) that provides power-down control to the on-chip LDOs and control of the WAKEUP and RTC_CLKOUT pins. See the device-specific data manual for more details on permanently disabling the RTC oscillator.

1.5.4 Static Power Management

1.5.4.1 RTC Power Management Register (RTCPMGT) [1930h]

This register enables static power management with power down and wake up register bits as described in the device-specific data sheet and, more generally, below. The RTC power management register (RTCPMGT) is shown in Figure 1-20 and described in Table 1-28.

Figure 1-20. RTC Power Management Register (RTCPMGT) [1930h]

15	5	4	3	2	1	0
Reserved		WU_DOUT	WU_DIR	BG_PD	LDO_PD	RTCCLKOUTEN
R-0		RW-1	RW-0	RW-0	RW-0	RW-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-28. RTC Power Management Register (RTCPMGT) Field Descriptions

Bit	Field	Value	Description
15-5	Reserved	0	Reserved
4	WU_DOUT		Wakeup output, active low/Open-drain.
		0	WAKEUP pin driven low.
		1	WAKEUP pin driver is in high impedance.
3	WU_DIR		Wakeup pin direction control.
		0	WAKEUP pin is configured as input.
		1	WAKEUP pin is configured as output.
			NOTE: The WAKEUP pin, when configured as an input, is active high. When it is configured as an output, it is open-drain and thus it should have an external pullup and it is active low.
2	BG_PD		Powerdown control bit for the bandgap, on-chip LDOs, and the analog POR (power on reset) comparator. This bit shuts down the on-chip LDOs (ANA_LDO, DSP_LDO, and USB_LDO), the Analog POR, and Bandgap reference. BG_PD and LDO_PD are only intended to be used when the internal LDOs supply power to the chip. If the internal LDOs are bypassed and not used then the BG_PD and LDO_PD power down mechanisms should not be used since the POR gets powered down and the POWERGOOD signal would not get generated properly.
			After this bit is asserted, the on-chip LDOs, Analog POR, and the Bandgap reference can only be re-enabled by the WAKEUP pin (being driven HIGH externally) or an enabled RTC alarm or an enabled RTC periodic event interrupt. Once reenabled, the Bandgap circuit takes about 100 msec to charge the external 0.1 µF capacitor on the BG_CAP pin via the the internal resistance of aproxmiately. 320 kΩ.
		0	On-chip LDO, Analog POR, and Bandgap reference are enabled.
		1	On-chip LDO, Analog POR, and Bandgap reference are disabled (shutdown).
1	LDO_PD		On-chip LDOs and Analog POR power down bit. This bit shuts down the on-chip LDOs (ANA_LDO, DSP_LDO, and USB_LDO) and the Analog POR. BG_PD and LDO_PD are only intended to be used when the internal LDOs supply power to the chip. If the internal LDOs are bypassed and not used then the BG_PD and LDO_PD power down mechanisms should not be used since POR gets powered down and the POWERGOOD signal is not generated properly.
			After this bit is asserted, the on-chip LDOs and Analog POR can only be re-enabled by the WAKEUP pin (being driven HIGH externally) or an enabled RTC alarm or an enabled RTC periodic event interrupt. This bit keeps the Bandgap reference turned on to allow a faster wake-up time with the expense power consumption of the Bandgap reference.
		0	On-chip LDO and Analog POR are enabled.
		1	On-chip LDO and Analog POR are disabled (shutdown).
0	RTCCLKOUTEN		Clock-out output enable.
		0	Clock output disabled.
		1	Clock output enabled.

1.5.4.2 RTC Interrupt Flag Register (RTCINTFL) [1920h]

The RTC interrupt flag register (RTCINTFL) is shown in Figure 1-21 and described in Table 1-29.

Figure 1-21. RTC Interrupt Flag Register (RTCINTFL) [1920h]

15	14	13	12	11	10	9	8
ALARMFL	Reserved						
R-0	R-0						

7	6	5	4	3	2	1	0
Reserved		EXTFL	DAYFL	HOURFL	MINFL	SECFL	MSFL
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-29. RTC Interrupt Flag Register (RTCINTFL) Field Descriptions

Bit	Field	Value	Description
15	ALARMFL		Indicates that an alarm interrupt has been generated.
		0	Alarm interrupt did not occur.
		1	Alarm interrupt occurred (write 1 to clear).
14-6	Reserved	0	Reserved.
5	EXTFL		External event (WAKEUP pin assertion) has occurred.
		0	External event interrupt has not occurred.
		1	External event interrupt occurred (write 1 to clear).
4	DAYFL		Day event has occurred.
		0	Periodic Day event has not occurred.
		1	Periodic Day event occurred (write 1 to clear).
3	HOURFL		Hour event has occurred.
		0	Periodic Hour event has not occurred.
		1	Periodic Hour event occurred (write 1 to clear).
2	MINFL		Minute Event has occurred.
		0	Periodic Minute event has not occurred.
		1	Periodic Minute event occurred (write 1 to clear).
1	SECFL		Second Event occurred.
		0	Periodic Second event has not occurred.
		1	Periodic Second event occurred (write 1 to clear).
0	MSFL		Millisecond event occurred.
		0	Periodic Millisecond event has not occurred.
		1	Periodic Millisecond event occurred (write 1 to clear).

1.5.4.3 Locking and Unlocking the RTC Registers

In order to write to the RTC registers, the following steps must be taken:

1. The RTC_ISO bit must be written to with a value of 1.
2. RGKR_LSW and RGKR_MSW must be written to with the fixed key, 0x95A4_F1E0.

After writing to the RTC registers:

1. Clear the gate-keeper registers by writing all zeros.
2. Clear the RSCR register.

The RTC system control and gate-keeper registers are part of a double isolation control for the RTC to protect it from corruption after power-up in case of power glitches. These registers should be cleared at all times except when the RTC registers need to be set.

1.5.4.3.1 RTC System Control Register (RSCR) [1C27h]

Figure 1-22. RTC System Control Register (RSCR) [1C27h]

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved							RTC_ISO
R-0h							R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-30. RTC System Control Register (RSCR) Field Descriptions

Bit	Field	Type	Reset	Description
15-1	Reserved	R	0h	Reserved.
0	RTC_ISO	R/W	0h	RTC Isolation Control. The reset state of this register is low which isolates the CVDD_RTC power domain from the CVDD power domain. This protects the RTC from corruption when CVDD powers up. The DSP must write a 1 to this register before accessing the RTC registers. DSP software should try to minimize the amount of time while this bit is set high so that the RTC is better protected from CVDD powerdown events. 0x0 = RTC is isolated. DSP reads/writes to the RTC registers will be blocked. 0x1 = RTC is not isolated. DSP is able to read/write the RTC registers.

1.5.4.3.2 RTC Gate-Keeper Register LSW (RGKR_LSW) [196Ch]

The RTC Gate-Keeper Register LSW (RGKR_LSW) is shown in Figure 1-23 and described in Table 1-31.

Figure 1-23. RTC Gate-Keeper Register LSW (RGKR_LSW) [196Ch]

15	0
Gate-Keeper_Code[15:0]	
RW-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-31. RTC Gate-Keeper Register LSW (RGKR_LSW) Field Descriptions

Bit	Field	Value	Description
15-0	Gate-Keeper_Code[15:0]		RTC Update Gating Control Register. RTC register writes will be silently blocked unless a specific 32-bit value is written into this register and RGKR_MSW. RTC register reads are unaffected and will occur normally regardless of the value in this register. This register is updated by the 32-kHz clock, so the DSP must allow time after a write for the write to complete. The amount of time that writes are unblocked should be minimized to reduce the chances of RTC time corruption in the event of CV_{DD} powerdown. 0x95A4_F1E0 = DSP writes to RTC registers are **not** blocked.

1.5.4.3.3 RTC Gate-Keeper Register MSW (RGKR_MSW) [196Dh]

The RTC Gate-Keeper Register MSW (RGKR_MSW) is shown in Figure 1-24 and described in Table 1-32.

Figure 1-24. RTC Gate-Keeper Register MSW (RGKR_MSW) [196Dh]

15	0
Gate-Keeper_Code[15:0]	
RW-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-32. RTC Gate-Keeper Register MSW (RGKR_MSW) Field Descriptions

Bit	Field	Value	Description
15-0	Gate-Keeper_Code[15:0]		RTC Update Gating Control Register. RTC register writes will be silently blocked unless a specific 32-bit value is written into this register and RGKR_LSW. RTC register reads are unaffected and will occur normally regardless of the value in this register. This register is updated by the 32-kHz clock, so the DSP must allow time after a write for the write to complete. The amount of time that writes are unblocked should be minimized to reduce the chances of RTC time corruption in the event of CV_{DD} powerdown. 0x95A4_F1E0 = DSP writes to RTC registers are **not** blocked.

1.5.4.4 Internal Memory Low Power Modes

To save power, software can place on-chip memory (DARAM or SARAM) in one of two power modes: memory retention mode and active mode. These power modes are activated through the SLPZVDD and SLPZVSS bits of the RAM Sleep Mode Control Register 1-5 (RAMSLPMDCNTLR[1:5]). To activate memory retention mode, set SLPZVDD bit and clear SLPZVSS bit of each memory bank to be put in retention mode. The retention/active mode of each 4kW DARAM and SARAM bank is independently controllable.

When either type of memory is placed in memory retention, read and write accesses are not allowed. In memory retention mode, the memory is placed in a low power mode while maintaining its contents. The contents are retained as long as there are no access attempts to that memory. In active mode, the memory is readily accessible by the CPU, but consumes more leakage power.

For the entire duration that the memory is in retention mode, there can be no attempts to read or write to the memories address range. This includes accesses by the CPU or any DMA. If an access is attempted while in retention mode then the memory contents will be lost.

NOTE: You must wait at least 10 CPU clock cycles after taking memory out of a low power mode before initiating any read or write access.

Table 1-33 summarizes the power modes for both DARAM and SARAM.

Table 1-33. On-Chip Memory Standby Modes

SLPZVDD	SLPZVSS	Mode	CV_{DD} Voltage
1	1	Active - Normal operational mode - Read and write accesses are allowed	1.05, 1.3, or 1.4 V
1	0	Retention - Low power mode - Contents are retained - No read or write access is allowed	1.05, 1.3, or 1.4 V
0	0	Memory Disabled Mode - Lowest leakage mode - Contents are lost - No read or write access is allowed	1.05, 1.3, or 1.4 V

1.5.4.4.1 RAM Sleep Mode Control Registers (RAMSLPMDCNTLR1–5) [1C28h, 1C2Ah, 1C2Bh, 1C2Ch, 1C2Dh]]

The RAM sleep mode control register 1 (RAMSLPMDCNTLR1) is shown in Figure 1-25 through Figure 1-29.

Figure 1-25. RAM Sleep Mode Control Register 1 (RAMSLPMDCNTLR1) [1C28h]

15	14	13	12	11	10	9	8
DARAM7 SLPZVDD	DARAM7 SLPZVSS	DARAM6 SLPZVDD	DARAM6 SLPZVSS	DARAM5 SLPZVDD	DARAM5 SLPZVSS	DARAM4 SLPZVDD	DARAM4 SLPZVSS
R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1
7	6	5	4	3	2	1	0
DARAM3 SLPZVDD	DARAM3 SLPZVSS	DARAM2 SLPZVDD	DARAM2 SLPZVSS	DARAM1 SLPZVDD	DARAM1 SLPZVSS	DARAM0 SLPZVDD	DARAM0 SLPZVSS
R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 1-26. RAM Sleep Mode Control Register 2 (RAMSLPMDCNTLR2) [1C2Ah]

15	14	13	12	11	10	9	8
SARAM7 SLPZVDD	SARAM7 SLPZVSS	SARAM6 SLPZVDD	SARAM6 SLPZVSS	SARAM5 SLPZVDD	SARAM5 SLPZVSS	SARAM4 SLPZVDD	SARAM4 SLPZVSS
R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1
7	6	5	4	3	2	1	0
SARAM3 SLPZVDD	SARAM3 SLPZVSS	SARAM2 SLPZVDD	SARAM2 SLPZVSS	SARAM1 SLPZVDD	SARAM1 SLPZVSS	SARAM0 SLPZVDD	SARAM0 SLPZVSS
R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 1-27. RAM Sleep Mode Control Register 3 (RAMSLPMDCNTLR3) [1C2Bh]

15	14	13	12	11	10	9	8
SARAM15 SLPZVDD	SARAM15 SLPZVSS	SARAM14 SLPZVDD	SARAM14 SLPZVSS	SARAM13 SLPZVDD	SARAM13 SLPZVSS	SARAM12 SLPZVDD	SARAM12 SLPZVSS
R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1
7	6	5	4	3	2	1	0
SARAM11 SLPZVDD	SARAM11 SLPZVSS	SARAM10 SLPZVDD	SARAM10 SLPZVSS	SARAM9 SLPZVDD	SARAM9 SLPZVSS	SARAM8 SLPZVDD	SARAM8 SLPZVSS
R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 1-28. RAM Sleep Mode Control Register 4 (RAMSLPMDCNTLR4) [1C2Ch]

15	14	13	12	11	10	9	8
SARAM23 SLPZVDD	SARAM23 SLPZVSS	SARAM22 SLPZVDD	SARAM22 SLPZVSS	SARAM21 SLPZVDD	SARAM21 SLPZVSS	SARAM20 SLPZVDD	SARAM20 SLPZVSS
R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1
7	6	5	4	3	2	1	0
SARAM19 SLPZVDD	SARAM19 SLPZVSS	SARAM18 SLPZVDD	SARAM18 SLPZVSS	SARAM17 SLPZVDD	SARAM17 SLPZVSS	SARAM16 SLPZVDD	SARAM16 SLPZVSS
R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 1-29. RAM Sleep Mode Control Register 5 (RAMSLPMDCNTLR5) [1C2Dh]

15	14	13	12	11	10	9	8
SARAM31 SLPZVDD	SARAM31 SLPZVSS	SARAM30 SLPZVDD	SARAM30 SLPZVSS	SARAM29 SLPZVDD	SARAM29 SLPZVSS	SARAM28 SLPZVDD	SARAM28 SLPZVSS
R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1
7	6	5	4	3	2	1	0
SARAM27 SLPZVDD	SARAM27 SLPZVSS	SARAM26 SLPZVDD	SARAM26 SLPZVSS	SARAM25 SLPZVDD	SARAM25 SLPZVSS	SARAM24 SLPZVDD	SARAM24 SLPZVSS
R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1	R/W+1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

1.5.5 Power Configurations

The power-saving features described in the previous sections, such as peripheral clock gating, and on-chip memory power down to name a few, can be combined to form a power configuration. Many different power configurations can be created by enabling and disabling different power domains and clock domains, however, this section defines some basic power configurations that may be useful. These are shown and described in Table 1-34. Please note that there is no single instruction or register that can place the device in these power configurations. Instead, these power configurations are achieved by modifying multiple registers.

NOTE: Before you change the power configuration, make sure that there is a method for the device to exit the power configuration. After exiting a power configuration, your software may have to take additional steps to change the clock and power configuration for other domains.

NOTE: The on-chip Bootloader idles all peripherals and CPU ports at startup. It enables some peripherals as it uses them. Your application code should check the idle configuration of peripherals and CPU ports before using them to be sure these are not idle.

Table 1-34. Power Configurations

Power Configuration	Power Domain State	Clock Domain State	Steps to Enter Clock and Power Configuration	Available Methods for Changing/Exiting Clock and Power Configuration
IDLE3 (Standby Mode)	All power domains on	RTC clock domain enabled Other clock domains disabled. Clock generator domain disabled (BYPASS MODE and PLL powerdown).	Idle peripheral domain Idle CPU domain PLL in BYPASS MODE PLL powerdown Execute idle instruction	A. WAKEUP pin B. RTC interrupt C. External hardware interrupt (INT0 or INT1). D. Any unmasked peripheral interrupt as defined in IER0 and IER1. E. Hardware Reset
IDLE2	All power domains on	RTC clock domain enabled Clock generator domain enabled (PLL_MODE) Other clock domains disabled	Idle peripheral domains Lower PLL frequency Idle CPU domain Execute idle instruction	A. WAKEUP pin B. RTC interrupt C. External hardware interrupt (INT0 or INT1). D. Any unmasked peripheral interrupt as defined in IER0 and IER1. E. Hardware Reset
Active	All power domains on	All clock domains enabled	Turn on all power domains Enable all clock domains	

1.5.5.1 IDLE2 Procedure

In this power configuration all the power domains are turned on, the RTC and clock generator domains are enabled, the CPU domain is disabled, and the DSP peripherals are disabled. When you enter this power configuration all CPU and peripheral activity in the DSP is stopped. Leaving the clock generator domain enabled allows the DSP to quickly exit this power configuration since there is no need to wait for power domains to turn on or for the PLL to re-lock.

Follow these steps to enter the IDLE2 power configuration:

1. Wait for completion of all DMA transfers. You can poll the DMA transfer status and disable DMA transfers through the DMA registers.
2. Disable the USB clock domain as described in Section 1.5.3.5.
3. Idle all the desired peripherals in the peripheral clock domain by modifying the peripheral clock gating configuration registers (PCGCR1 and PCGCR2). See Section 1.5.3.2 for more details on setting the DSP peripherals to idle mode.
4. Clear all interrupts by writing ones to the CPU interrupt flag registers (IFR0 and IFR1).
5. Enable the external event interrupt EXTINTEN = 1 in the RTCINTREG register.
6. Unlock the RTC registers as stated in Section 11.2.3, *Locking and Unlocking the RTC Registers*, to execute the next step.
7. Enable the appropriate wake-up interrupt in the CPU interrupt enable registers (IER0 and IER1).
   - If using the WAKEUP pin to exit this mode:
     (a) Configure the WAKEUP pin as an input by setting WU_DIR = 1 in the RTC power management register (RTCPMGT).
     (b) Enable the external event interrupt by setting EXTINTEN = 1 in the RTCINTREG register.
   - If using the RTC alarm or periodic interrupt as a wake-up event, the RTCINTEN bit must be set in the RTC interrupt enable register (RTCINTEN).
8. Lower the PLL frequency. The lowest frequency for VOC output is 125 MHz.
9. Disable the CPU domain by setting to 1 the CPUI, MPORTI, XPORTI, DPORTI, IPORTI, and CPI bits of the idle configuration register (ICR). Note that the MPORT will not go into idle mode if the USB CDMA or DMA controllers is not idled.
10. Apply the new idle configuration by executing the “IDLE” instruction. The content of ICR is copied to the idle status register (ISTR). The bits of ISTR are then propagated through the CPU domain system to enable or disable the specified clocks.

The IDLE instruction cannot be executed in parallel with another instruction.

To exit the IDLE2 power configuration, follow these steps:

1. Generate the wake-up interrupt you specified during the IDLE2 power down procedure.
2. After the interrupt is generated, the DSP will execute the interrupt service routine.
3. After exiting the interrupt service routine, code execution will resume from the point where the “IDLE” instruction was originally executed.

You can also exit the IDLE2 power configuration by generating a hardware reset. However, in this case, the DSP is completely reset and the state of the DSP before going into IDLE2 is lost.

1.5.5.2 IDLE3 Procedure

In this power configuration all the power domains are turned on, the CPU and clock generator domains are disabled, and the RTC clock domain is enabled. The DSP peripherals and the USB are also disabled in this mode. When you enter this power configuration, all CPU and peripheral activity in the DSP is stopped.

Since the clock generator domain is disabled, you must allow enough time for the PLL to re-lock before exiting this power configuration.

Follow these steps to enter the IDLE3 power configuration:

1. Wait for completion of all DMA transfers. You can poll the DMA transfer status and disable DMA transfers through the DMA registers.
2. Disable the USB clock domain as described in Section 1.5.3.5.
3. Idle all the desired peripherals in the peripheral clock domain by modifying the peripheral clock gating configuration registers (PCGCR1 and PCGCR2). See Section 1.5.3.2 for more details on setting the DSP peripherals to idle mode.
4. Disable the clock generator domain as described in Section 1.5.3.3.
5. Clear all interrupts by writing ones to the CPU interrupt flag registers (IFR0 and IFR1).
6. Enable the external event interrupt EXTINTEN = 1 in the RTCINTREG register.
7. Unlock the RTC registers as stated in Section 11.2.3, *Locking and Unlocking the RTC Registers*, to execute the next step.
8. Enable the appropriate wake-up interrupt in the CPU interrupt enable registers (IER0 and IER1).
   - If using the WAKEUP pin to exit this mode:
     (a) Configure the WAKEUP pin as an input by setting WU_DIR = 1 in the RTC power management register (RTCPMGT).
     (b) Enable the external event interrupt by setting EXTINTEN = 1 in the RTCINTREG register.
   - If using the RTC alarm or periodic interrupt as a wake-up event, the RTCINTEN bit must be set in the RTC interrupt enable register (RTCINTEN).
9. Bypass the PLL by setting SYSCLKSEL = 0 in the CCR2 Register.
10. Power down the PLL by setting the PCR register to 0xA000.
11. Apply the new idle configuration by executing the IDLE instruction. The content of ICR is copied to the idle status register (ISTR). The bits of ISTR are then propagated through the CPU domain system to enable or disable the specified clocks.

The IDLE instruction cannot be executed in parallel with another instruction.

To exit the IDLE3 power configuration, follow these steps:

1. Generate the wake-up interrupt you specified during the IDLE3 power down procedure.
2. After the interrupt is generated, the DSP will execute the interrupt service routine.
3. After exiting the interrupt service routine, code execution will resume from the point where the “IDLE” instruction was originally executed.
4. Enable the clock generator domain as described in Section 1.5.3.3. You can also enable the clock generator domain inside the interrupt service routine.

You can also exit the IDLE3 power configuration by generating a hardware reset, however, in this case the DSP is completely reset and the state of the DSP before going into IDLE3 is lost.

1.5.5.3 Core Voltage Scaling

When the core voltage domain (CV_{DD}) is ON, it can be set to three voltages: 1.4, 1.3 V, or 1.05 V (nominal). The core voltage can be reduced during periods of low processing demand and increased during high demand. Core voltage scaling can be accomplished with an external power management IC (LDO, DC-DC, etc). When the core voltage is decreased (1.4 V to 1.3 V to 1.05 V), care must be taken to ensure device stability. The following rules must be followed to maintain stability:

- When using an external PMIC (power management IC), the board designer must ensure that, for example, from 1.3 V to 1.05 V, the transition does not have ringing that would violate our CV_{DD} minimum rating (1.05 V - 5% = 0.998 V).
- Software must ensure that the clock speed of the device does not exceed the maximum speed of the device at the lower voltage before making the voltage transition. For example, if the device is running at 75 MHz @ 1.3 V, then the PLL must be changed to 75 MHz before changing the core voltage to 1.05 V.

When the core voltage is increased, for example, from 1.05 V to 1.3 V, clock speed is not an issue since the device can operate faster at the higher voltage. However, when switching from 1.05 V to 1.3 V software must allow time for the voltage transition to reach the 1.3 V range. Additionally, external regulators might produce an overshoot that must not pass the maximum operational voltage of the core supply (see the *Recommended Operating Conditions* section in device-specific data manual). Otherwise, the device will be operating out of specification. This could happen if large current draw occurs while the regulator transitions to the higher voltage.

For external PMICs, the step response varies greatly and it is up to the system designer to ensure that the ringing is maintained within the DSP's core supply high voltage operational tolerance (see the *Recommended Operating Conditions* section in device-specific data manual).

1.6 Interrupts

Vector-relative locations and priorities for all internal and external interrupts are shown in Table 1-35.

Table 1-35. Interrupt Table

NAME	SOFTWARE (TRAP) EQUIVALENT	RELATIVE LOCATION (HEX BYTES)(1)	PRIORITY	FUNCTION
RESET	SINT0	0x0	0	Reset (hardware and software)
NMI(2)	SINT1	0x8	1	Non-maskable interrupt
INT0	SINT2	0x10	3	External user interrupt #0
INT1	SINT3	0x18	5	External user interrupt #1
TINT	SINT4	0x20	6	Timer aggregated interrupt
PROG0	SINT5	0x28	7	Programmable transmit interrupt 0 (I2S0 transmit, McBSP transmit, or MMC/SD0 interrupt)
UART	SINT6	0x30	9	UART interrupt
PROG1	SINT7	0x38	10	Programmable receive interrupt 1 (I2S0 receive, McBSP receive, or MMC/SD0 SDIO interrupt)
DMA	SINT8	0x40	11	DMA aggregated interrupt
PROG2	SINT9	0x48	13	Programmable transmit interrupt 1 (McSPI interrupt or MMC/SD1 interrupt)
-	SINT10	0x50	14	Software interrupt
PROG3	SINT11	0x58	15	Programmable receive interrupt 3 (MMC/SD1 SDIO interrupt)
UHPI	SINT12	0x60	17	UHPI interrupt
SAR	SINT13	0x68	18	10-bit SAR A/D conversion or pin interrupt
XMT2	SINT14	0x70	21	I2S2 transmit interrupt
RCV2	SINT15	0x78	22	I2S2 receive interrupt
XMT3	SINT16	0x80	4	I2S3 transmit interrupt
RCV3	SINT17	0x88	8	I2S3 receive interrupt
RTC	SINT18	0x90	12	Wakeup or real-time clock interrupt
SPI	SINT19	0x98	16	SPI interrupt
USB	SINT20	0xA0	19	USB Interrupt
GPIO	SINT21	0xA8	20	GPIO aggregated interrupt
EMIF	SINT22	0xB0	23	EMIF error interrupt
I2C	SINT23	0xB8	24	I2C interrupt
BERR	SINT24	0xC0	2	Bus error interrupt
DLOG	SINT25	0xC8	25	Data log interrupt
RTOS	SINT26	0xD0	26	Real-time operating system interrupt
-	SINT27	0xD8	14	Software interrupt #27
-	SINT28	0xE0	15	Software interrupt #28
-	SINT29	0xE8	16	Software interrupt #29
-	SINT30	0xF0	17	Software interrupt #30
-	SINT31	0xF8	18	Software interrupt #31

(1) Absolute addresses of the interrupt vector locations are determined by the contents of the IVPD and IVPH registers. Interrupt vectors for interrupts 0-15 and 24-31 are relative to IVPD. Interrupt vectors for interrupts 16-23 are relative to IVPH.

(2) The NMI signal is internally tied high (not asserted). However, NMI interrupt vector can be used for SINT1.

1.6.1 IFR and IER Registers

The interrupt flag register 0 (IFR0) and interrupt enable register 0 (IER0) bit layouts are shown in Figure 1-30 and described in Table 1-36.

Figure 1-30. IFR0 and IER0 Bit Locations

15	14	13	12	11	10	9	8
RCV2	XMT2	SAR	UHPI	PROG3	Reserved	PROG2	DMA
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
PROG1	UART	PROG0	TINT	INT1	INT0	Reserved	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-36. IFR0 and IER0 Bit Descriptions

Bit	Field	Value	Description
15	RCV2	1-0	I2S2 receive interrupt flag/mask bit.
14	XMT2	1-0	I2S2 transmit interrupt flag/mask bit.
13	SAR	1-0	SAR interrupt bit.
12	UHPI	1-0	UHPI interrupt bit.
11	PROG3	1-0	Programmable receive interrupt 3 flag/mask bit. This bit is used as the MMC/SD1 SDIO interrupt flag/mask bit. The function of this bit is selected depending on the setting of the SP1MODE bit is in external bus selection register. If SP1MODE = 00b, this bit supports MMC/SD1 SDIO interrupts. If SP1MODE = 01, this bit is a logic high.
10	Reserved	0	Reserved. This bit should always be written with 0.
9	PROG2	1-0	Programmable transmit interrupt 2 flag/mask bit. This bit is used as the MMC/SD1 interrupt flag/mask bit. The function of this bit is selected depending on the setting of the SP1MODE bit in the external bus selection register. If SP1MODE = 00b, this bit supports MMC/SD1 interrupts. If SP1MODE = 01, this bit supports McSPI interrupts.
8	DMA	1-0	DMA aggregated interrupt flag/mask bit
7	PROG1	1-0	Programmable receive interrupt 1 flag/mask bit. This bit is used as either the I2S0 receive interrupt flag/mask bit or the MMC/SD0 SDIO interrupt flag/mask bit. The function of this bit is selected depending on the setting of the SP0MODE bit in the external bus selection register. If SP0MODE = 00b, this bit supports MMC/SD0 SDIO interrupts. If SP0MODE = 11, this bit supports McBSP interrupts.
6	UART	1-0	UART interrupt flag/mask bit
5	PROG0	1-0	Programmable transmit interrupt 0 flag/mask bit. This bit is used as either the I2S0 transmit interrupt flag/mask bit or the MMC/SD0 interrupt flag/mask bit. The function of this bit is selected depending on the setting of the SP0MODE bit in the external bus selection register. If SP0MODE = 00b, this bit supports MMC/SD0 interrupts. If SP0MODE = 11, this bit supports McBSP interrupts.
4	TINT	1-0	Timer aggregated interrupt flag/mask bit.
3	INT1	1-0	External user interrupt #1 flag/mask bit.
2	INT0	1-0	External user interrupt #0 flag/mask bit.
1-0	Reserved	0	Reserved. This bit should always be written with 0.

The interrupt flag register (IFR1) and interrupt enable register 1 (IER1) bit layouts are shown in Figure 1-31 and described in Table 1-37.

Figure 1-31. IFR1 and IER1 Bit Locations

15					11	10	9	8
Reserved						RTOS	DLOG	BERR
R-0						R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0	
I2C	EMIF	GPIO	USB	SPI	RTC	RCV3	XMT3	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-37. IFR1 and IER1 Bit Descriptions

Bit	Field	Value	Description
15-11	Reserved	0	Reserved. This bit should always be written with 0.
10	RTOS	1-0	Real-Time operating system interrupt flag/mask bit.
9	DLOG	1-0	Data log interrupt flag/mask bit.
8	BERR	1-0	Bus error interrupt flag/mask bit.
7	I2C	1-0	I2C interrupt flag/mask bit.
6	EMIF	1-0	EMIF error interrupt flag/mask bit.
5	GPIO	1-0	GPIO aggregated interrupt flag/mask bit.
4	USB	1-0	USB interrupt flag/mask bit.
3	SPI	1-0	SPI interrupt flag/mask bit.
2	RTC	1-0	Wakeup or real-time clock interrupt flag/mask bit.
1	RCV3	1-0	I2S3 receive interrupt flag/mask bit.
0	XMT3	1-0	I2S3 transmit interrupt flag/mask bit.

1.6.2 Interrupt Timing

The interrupt signals on the external interrupts pins ($\overline{INT0}$ and $\overline{INT1}$) are detected with a synchronous negative edge detector circuit. To reliably detect the external interrupts, the interrupt signal must have at least 2 SYSCLK high followed by at least 2 SYSCLK low.

To define the minimum low pulse width in nanoseconds scale, you should take into account that the on-chip PLL of the device is software programmable and that your application may be dynamically changing the frequency of PLL. You should use the slowest frequency that will be used by your application to calculate the minimum interrupt pulse duration in nanoseconds.

When the system master clock is disabled (SYSCLKDIS =1), the external interrupt pins ($\overline{INT0}$ and $\overline{INT1}$) will be asynchronously latched and held low while the clocks are re-enabled. Once the clocks are re-enabled, the DSP will latch the interrupt in the IFR.

1.6.3 Timer Interrupt Aggregation Flag Register (TIAFR) [1C14h]

The CPU has only one interrupt flag that is shared among the three timers. The CPU's interrupt flag is bit 4 (TINT) of the IFR0 & IER0 registers (see Figure 1-31). Since the interrupt flag is shared, software must have a means of determining which timer instance caused the interrupt. Therefore, the timer interrupt aggregation flag register (TIAFR) is a secondary flag register that serves this purpose.

The timer interrupt aggregation flag register (TIAFR) latches each timer (Timer 0, Timer 1, and Timer 2) interrupt signal when the timer counter expires. Using this register, the programmer can determine which timer generated the timer aggregated CPU interrupt signal (TINT). Each Timer flag in TIAFR needs to be cleared by the CPU with a write of 1.

Note that the IFR0[TINT] bit is automatically cleared when entering the interrupt service routine (ISR). Therefore there is no need to manually clear it in the ISR. If two (or more) timers happen to interrupt simultaneously, the TIAFR register will indicate the two (or more) interrupt flags. In this case, the ISR can choose to service both timer interrupts or only one-at-a-time. If the ISR services only one of them, then it should clear only one of the TIAFR flags and upon exiting the ISR, the CPU will immediately be interrupted again to service the second timer flag. If the ISR services all of them, then it should clear all of them in the TIAFR flags and upon exiting the ISR, the CPU won't be interrupted again until a new timer interrupt comes in. For more information, see Section 14.1.

1.6.3.1 Timer Interrupt Selection Register (TISR) [1C3Eh]

The timer interrupt selection register allows the timer interrupt for each timer (Timer0, Timer1 and Timer2) to be configured as a timer interrupt (TINT) or as a non-maskable interrupt (NMI) to the CPU.

The interrupt for the three timers can be routed to CPU TINT or CPU NMI through Timer Interrupt Selection Register [0x1C3E].

Figure 1-32. Timer Interrupt Selection Register (TISR) [1C3Eh]

15	14	13	12	11	10	9	8
RESERVED							
R-0h							

7	6	5	4	3	2	1	0
RESERVED					TINT2SEL	TINT1SEL	TINT0SEL
R-0h					R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-38. Timer Interrupt Selection Register Bit Field Descriptions

Bit	Field	Type	Reset	Description
15-3	RESERVED	R	0h	Reserved
2	TINT2SEL	R/W	0h	Timer2 Interrupt Selection 0x0 = Timer2 interrupt is routed to CPU TINT 0x1 = Timer2 interrupt is routed to NMI
1	TINT1SEL	R/W	0h	Timer1 Interrupt Selection 0x0 = Timer1 interrupt is routed to CPU TINT 0x1 = Timer1 interrupt is routed to NMI
0	TINT0SEL	R/W	0h	Timer0 Interrupt Selection 0x0 = Timer0 interrupt is routed to CPU TINT 0x1 = Timer0 interrupt is routed to NMI

1.6.4 GPIO Interrupt Enable and Aggregation Flag Registers

The CPU has only one interrupt flag that is shared among all GPIO pin interrupt signals. The CPU's interrupt flag is bit 5 (GPIO) of the IFR1 and IER1 registers (see Figure 1-31). Since the interrupt flag is shared, software must have a means of determining which GPIO pin caused the interrupt. Therefore, the GPIO interrupt aggregation flag registers (IOINTFLG1 and IOINTFLG2) are secondary flag registers that serve this purpose.

If any of the GPIO pins are configured as inputs, they can be enabled to accept external signals as interrupts using the GPIO Interrupt Enable Registers (IOINTEN1 and IOINTEN2). The GPIO Interrupt Flag Registers (IOINTFLG1 and IOINTFLG2) can be used to determine which of the 32 GPIO pins triggered the interrupt. Note that the IFR0[GPIO] bit is automatically cleared when entering the interrupt service routine (ISR). Therefore, there is no need to manually clear it in the ISR. If two (or more) GPIO pins happen to interrupt simultaneously, the IOINTFLG1/IOINTFLG2 register indicates the two (or more) interrupt flags. In this case, the ISR can choose to service both/all GPIO interrupts or only one-at-a-time. If the ISR services only one of them, then it should clear only one of the IOINTFLG1/IOINTFLG2 flags and upon exiting the ISR, the CPU is immediately interrupted again to service the others. For more information, see Section 4.1.

1.6.5 DMA Interrupt Enable and Aggregation Flag Registers

The CPU has only one interrupt flag that is shared among the 16 DMA interrupt sources. The CPU's interrupt flag is bit 8 (DMA) of the IFR0 & IER0 registers (see Figure 1-30). Since the interrupt flag is shared, software must have a means of determining which DMA instance caused the interrupt. Therefore, the DMA interrupt aggregation flag registers (DMAIFR) are secondary flag registers that serve this purpose.

Each of the four channels of a DMA controller has its own interrupt, which you can enable or disable a channel interrupt though the DMA*n*CH*m* bits of the DMA Interrupt Enable Register (DMAIER) (see Section 1.7.4.2.1). The interrupts from the four DMA controllers are combined into a single CPU interrupt. You can determine which DMA channel generated the interrupt by reading the bits of the DMA interrupt flag register (DMAIFR). For more information, see Section 4.1.

1.6.6 McSPI Interrupt Aggregation Flag Register

In order to support the slave wake-up interrupts, the McSPI slave wake-up and regular interrupts are aggregated into one register.

Figure 1-33. McSPI Interrupt Aggregation Flag Register (MSIAFR) [1C15h]

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved						MSWAKEUPFLAG	MSINTFLAG
R-0h						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-39. McSPI Interrupt Aggregation Flag Register Bit Field Descriptions

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved.
1	MSWAKEUPFLAG	R/W	0h	McSPI Wake-Up Interrupt Flag. Connected to McSPI WAKEUP. Write a 1 to this register to clear the interrupt flag. 0x0 = McSPI has not generated interrupt. 0x1 = McSPI wake-up interrupt has occurred.

Table 1-39. McSPI Interrupt Aggregation Flag Register Bit Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	MSINTFLAG	R/W	0h	McSPI Interrupt Flag. Connected to McSPI Interrupt. Write a 1 to this register to clear the interrupt flag. 0x0 = McSPI has not generated interrupt. 0x1 = McSPI interrupt has occurred.

1.6.7 McSPI Interrupt Aggregation Enable Register

Figure 1-34. McSPI Interrupt Aggregation Enable Register (MSIAER) [1C3Dh]

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved						MSWAKEUPEN	MSINTEN
R-0h						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-40. McSPI Interrupt Aggregation Enable Register Bit Field Descriptions

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved.
1	MSWAKEUPEN	R/W	0h	McSPI Wake-Up Interrupt Enable 0x0 = McSPI Wake-Up will not interrupt the CPU. 0x1 = McSPI Wake-Up will interrupt the CPU.
0	MSINTEN	R/W	0h	McSPI Interrupt Enable. 0x0 = McSPI will not interrupt the CPU. 0x1 = McSPI will interrupt the CPU.

1.7 System Configuration and Control

1.7.1 Overview

The DSP includes system-level registers for controlling, configuring, and reading the status of the device. These registers are accessible by the CPU and support the following features:

- Device Identification
- Device Configuration
  - Pin multiplexing control
  - Output drive strength configuration
  - Internal pullup and pulldown enable/disable
- DMA Controller Configuration
- Peripheral Reset
- EMIF and USB Byte Access

1.7.2 Device Identification

The DSP includes a set of device ID registers that are intended for use in TI chip manufacturing. These registers are summarized in the following table.

Table 1-41. Die ID Registers

CPU Word Address	Acronym	Register Description	Section
1C40h	DIEIDR0[1]	Die ID Register 0	Section 1.7.2.1
1C41h	DIEIDR1[1]	Die ID Register 1	Section 1.7.2.2
1C42h	DIEIDR2[1]	Die ID Register 2	Section 1.7.2.3
1C43h	DIEIDR3[1]	Die ID Register 3	Section 1.7.2.4
1C44h	DIEIDR4[1]	Die ID Register 4	Section 1.7.2.5
1C45h	DIEIDR5[1]	Die ID Register 5	Section 1.7.2.6
1C46h	DIEIDR6[1]	Die ID Register 6	Section 1.7.2.7
1C47h	DIEIDR7[1]	Die ID Register 7	Section 1.7.2.8
1C58h	JTAGIDLSW	JTAG ID Code LSW Register	Section 1.7.2.9
1C59h	JTAGIDMSW	JTAG ID Code MSW Register	Section 1.7.2.10

(1) This register is reserved.

1.7.2.1 Die ID Register 0 (DIEIDR0) [1C40h]

The die ID register 0 (DIEIDR0) is shown in Figure 1-35 and described in Table 1-42.

Figure 1-35. Die ID Register 0 (DIEIDR0) [1C40h]

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-42. Die ID Register 0 (DIEIDR0) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	Reserved	R	0h	Reserved.

1.7.2.2 Die ID Register 1 (DIEIDR1) [1C41h]

The die ID register 1 (DIEIDR1) is shown in Figure 1-36 and described in Table 1-43.

Figure 1-36. Die ID Register 1 (DIEIDR1) [1C41h]

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-43. Die ID Register 1 (DIEIDR1) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	Reserved	R	0h	Reserved.

1.7.2.3 Die ID Register 2 (DIEIDR2) [1C42h]

The die ID register 2 (DIEIDR2) is shown in Figure 1-37 and described in Table 1-44.

Figure 1-37. Die ID Register 2 (DIEIDR2) [1C42h]

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-44. Die ID Register 2 (DIEIDR2) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	Reserved	R	0h	Reserved.

1.7.2.4 Die ID Register 3 (DIEIDR3) [1C43h]

The die ID register 3 (DIEIDR3) is shown in Figure 1-38 and described in Table 1-45.

Figure 1-38. Die ID Register 3 (DIEIDR3) [1C43h]

15	14	13	12	11	10	9	8
DESIGNREV				Reserved			
R-5h				R-0h			

7	6	5	4	3	2	1	0
Reserved							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-45. Die ID Register 3 (DIEIDR3) Field Descriptions

Bit	Field	Type	Reset	Description
15-12	DESIGNREV	R	0h	Describes the Silicon Revision of Die CPU. 0h = Silicon 1.0. 1h = Silicon 2.0. 2h = Silicon 2.1.
11-0	Reserved	R	0h	Reserved.

1.7.2.5 Die ID Register 4 (DIEIDR4) [1C44h]

The die ID register 4 (DIEIDR4) is shown in Figure 1-39 and described in Table 1-46.

Figure 1-39. Die ID Register 4 (DIEIDR4) [1C44h]

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-46. Die ID Register 4 (DIEIDR4) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	Reserved	R	0h	Reserved.

1.7.2.6 Die ID Register 5 (DIEIDR5) [1C45h]

The die ID register 5 (DIEIDR5) is shown in Figure 1-40 and described in Table 1-47.

Figure 1-40. Die ID Register 5 (DIEIDR5) [1C45h]

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-47. Die ID Register 5 (DIEIDR5) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	Reserved	R	0h	Reserved.

1.7.2.7 Die ID Register 6 (DIEIDR6) [1C46h]

The die ID register 6 (DIEIDR6) is shown in Figure 1-41 and described in Table 1-48.

Figure 1-41. Die ID Register 6 (DIEIDR6) [1C46h]

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-48. Die ID Register 6 (DIEIDR6) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	Reserved	R	0h	Reserved.

1.7.2.8 Die ID Register 7 (DIEIDR7) [1C47h]

The die ID register 7 (DIEIDR7) is shown in Figure 1-42 and described in Table 1-49.

Figure 1-42. Die ID Register 7 (DIEIDR7) [1C47h]

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-49. Die ID Register 7 (DIEIDR7) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	Reserved	R	0h	Reserved.

1.7.2.9 JTAG ID Code LSW Register [1C58h]

Figure 1-43. JTAG ID Code LSW Register (JTAGIDLSW) [1C58h]

15	14	13	12	11	10	9	8
PARTCODE3DOWN0				MFRID			
R-6h				R-17h			

7	6	5	4	3	2	1	0
MFRID							AONE
R-17h							R-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-50. JTAG ID Code LSW Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	PARTCODE3DOWN0	R	6h	Part Code number. Read as 0xB956 (unique code within TI assigned to this device).
11-1	MFRID	R	17h	Manufacturer ID number. Read as 0x17 (this is the TI manufacturer ID).
0	AONE	R	1h	A read only bit which will hold a value of 1.

1.7.2.10 JTAG ID Code MSW Register [1C59h]

Figure 1-44. JTAG ID Code MSW Register (JTAGIDMSW) [1C59h]

15	14	13	12	11	10	9	8
VERSION				PARTCODE15DOWN4			
R-0h				R-B95h			
7	6	5	4	3	2	1	0
PARTCODE15DOWN4							
R-B95h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-51. JTAG ID Code MSW Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	VERSION	R	0h	Version number. Read as 0x0
11-0	PARTCODE15DOWN4	R	B95h	Part Code number. Read as 0xB95 (unique code within TI assigned to this device).

1.7.3 Device Configuration

The DSP includes registers for configuring pin multiplexing, the pin output slew rate, the internal pullups and pulldowns, DSP_LDO voltage selection and USB_LDO enable.

1.7.3.1 External Bus Selection Register (EBSR)

The external bus selection register (EBSR) determines the mapping of the UHPI, I2S2, UART, SPI, McBSP, McSPI, and GPIO signals to 28 of the external parallel port pins. It also determines the mapping of the I2S, McBSP, McSPI, GPIO, or MMC/SD ports to serial port 0 pins and serial port 1 pins. The EBSR register is located at port address 0x1C00. Once the bit fields of this register are changed, the routing of the signals takes place on the next CPU clock cycle.

Additionally, the EBSR controls the function of the upper bits of the EMIF address bus. Pins EM_A[20:15] can be individually configured as GPIO pins through the Axx_MODE bits. When Axx_MODE = 1, the EM_A[xx] pin functions as a GPIO pin. When Axx_MODE = 0, the EM_A[xx] pin retains its EMIF functionality.

NOTE: EM_A[20:15]/GPIO[26:21] are used to select the boot mode during boot.

Before modifying the values of the external bus selection register, you must clock gate all affected peripherals through the Peripheral Clock Gating Control Register (for more information on clock gating peripherals, see Section 1.5.3.2). After the external bus selection register has been modified, you must reset the peripherals before using them through the Peripheral Software Reset Counter Register.

After the boot process is complete, the external bus selection register must be modified only once, during device configuration. Continuously switching the EBSR configuration is not supported.

The external bus selection register (EBSR) is shown in Figure 1-45 and described in Table 1-52.

Figure 1-45. External Bus Selection Register (EBSR) [1C00h]

15	14	12	11	10	9	8
McBSP_CLKS Selection	PPMODE		SP1MODE		SP0MODE	
R/W-0	R/W-001		R/W-00		R/W-00	

7	6	5	4	3	2	1	0
Reserved		A20_MODE	A19_MODE	A18_MODE	A17_MODE	A16_MODE	A15_MODE
R-0		R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-52. EBSR Register Field Descriptions

Bit	Field	Description
15	McBSP_CLKS Selection	McBSP_CLKS Selection 0 = McBSP_CLKR signal is routed to MMC0_D3/GP[5]/McBSPCLKR_CLKS when SP0MODE=3 1 = McBSP_CLKS signal is routed to MMC0_D3/GP[5]/McBSPCLKR_CLKS when SP0MODE=3
14:12	PPMODE	Parallel Port Mode Control Bits. These bits control the pin multiplexing of the UHPI, SPI, UART, I2S2, I2S3, and GP[31:27, 20:12] pins on the parallel port. For more details, see Table 1-53. 000 = Mode 0 (16-bit UHPI bus). All 28 signals of the UHPI bus module are routed to the 28 external signals of the parallel port. Note: SDRAM control signals are multiplexed with UHPI bus control signals. In this mode, UHPI bus signals are routed to the control ports, so SDRAM cannot be accessible. 001 = Mode 1 (SPI, GPIO, UART, I2S2, and SDRAM). 7 signals of the SPI module, 6 GPIO signals, 4 signals of the UART module, 4 signals of the I2S2 module, and 7 SDRAM control signals are routed to the 28 external signals of the parallel port. 010 = Mode 2 (GPIO and SDRAM). 8 GPIO and 7 SDRAM control signals are routed to the 28 external signals of the parallel port. 011 = Mode 3 (SPI, I2S3, and SDRAM). 4 signals of the SPI module, 4 signals of the I2S3 module, and 7 SDRAM control signals are routed to the 28 external signals of the parallel port. 100 = Mode 4 (I2S2, UART, and SDRAM). 4 signals of the I2S2 module, 4 signals of the UART module, and 7 SDRAM control signals are routed to the 28 external signals of the parallel port. 101 = Mode 5 (SPI, UART, and SDRAM). 4 signals of the SPI module, 4 signals of the UART module, and 7 SDRAM control signals are routed to the 28 external signals of the parallel port. 110 = Mode 6 (SPI, I2S2, I2S3, GPIO, and SDRAM). 7 signals of the SPI module, 4 signals of the I2S2 module, 4 signals of the I2S3 module, 6 GPIO, and 7 SDRAM control signals are routed to the 28 external signals of the parallel port. 111 = Reserved.
11:10	SP1MODE	Serial Port 1 Mode Control Bits. The bits control the pin multiplexing of the MMC1, McSPI, and GPIO pins on serial port 1. For more details, see Table 1-54. 00 = Mode 0 (MMC/SD1). All 6 signals of the MMC/SD1 module are routed to the 6 external signals of the serial port 1. 01 = Mode 1 (McSPI). 6 signals of the McSPI module signals are routed to the 6 external signals of the serial port 1. 10 = Mode 2 (GP[11:6]). 6 GPIO signals (GP[11:6]) are routed to the 6 external signals of the serial port 1. 11 = Reserved.
9:8	SP0MODE	Serial Port 0 Mode Control Bits. The bits control the pin multiplexing of the MMC0, I2S0, McBSP, and GPIO pins on serial port 0. For more details, see Section 1.7.3.2.1.3. 00 = Mode 0 (MMC/SD0). All 6 signals of the MMC/SD0 module are routed to the 6 external signals of the serial port 0. 01 = Mode 1 (I2S0 and GP[5:4]). 4 signals of the I2S0 module and 2 GP[5:4] signals are routed to the 6 external signals of the serial port 0. 10 = Mode 2 (GP[5:0]). 6 GPIO signals (GP[5:0]) are routed to the 6 external signals of the serial port 0. 11 = Mode 3 (McBSP). 6 signals of the McBSP module are routed to the 6 external signal port 0.
7-6	Reserved	Reserved. Read-only, writes have no effect.
5	A20_MODE	A20 Pin Mode Bit. This bit controls the pin multiplexing of the EMIF address 20 (EM_A[20]) and general-purpose input/output pin 26 (GP[26]) pin functions. 0 = Pin function is EMIF address pin 20 (EM_A[20]). 1 = Pin function is general-purpose input/output pin 26 (GP[26]). This is the default mode at reset and the pin is configured as an Input. Approximately 10 cycles after the rising edge of RESET, the state on this pin is latched into the BootMode register to specify the boot method.
4	A19_MODE	A19 Pin Mode Bit. This bit controls the pin multiplexing of the EMIF address 19 (EM_A[19]) and general-purpose input/output pin 25 (GP[25]) pin functions. 0 = Pin function is EMIF address pin 19 (EM_A[19]). 1 = Pin function is general-purpose input/output pin 25 (GP[25]). This is the default mode at reset and the pin is configured as an Input. Approximately 10 cycles after the rising edge of RESET, the state on this pin is latched into the BootMode register to specify the boot method.

Table 1-52. EBSR Register Field Descriptions (continued)

Bit	Field	Description
3	A18_MODE	A18 Pin Mode Bit. This bit controls the pin multiplexing of the EMIF address 18 (EM_A[18]) and general-purpose input/output pin 24 (GP[24]) pin functions. 0 = Pin function is EMIF address pin 18 (EM_A[18]). 1 = Pin function is general-purpose input/output pin 24 (GP[24]). This is the default mode at reset and the pin is configured as an Input. Approximately 10 cycles after the rising edge of RESET, the state on this pin is latched into the BootMode register to specify the boot method.
2	A17_MODE	A17 Pin Mode Bit. This bit controls the pin multiplexing of the EMIF address 17 (EM_A[17]) and general-purpose input/output pin 23 (GP[23]) pin functions. For more details, see Table 1-55, *MMC0, I2S0, McBSP, and GP[5:0] Pin Multiplexing*. 0 = Pin function is EMIF address pin 17 (EM_A[17]). 1 = Pin function is general-purpose input/output pin 23 (GP[23]). This is the default mode at reset and the pin is configured as an Input. Approximately 10 cycles after the rising edge of RESET, the state on this pin is latched into the BootMode register to specify the boot method.
1	A16_MODE	A16 Pin Mode Bit. This bit controls the pin multiplexing of the EMIF address 16 (EM_A[16]) and general-purpose input/output pin 22 (GP[22]) pin functions. For more details, see Table 1-55, *MMC0, I2S0, McBSP, and GP[5:0] Pin Multiplexing*. 0 = Pin function is EMIF address pin 16 (EM_A[16]). 1 = Pin function is general-purpose input/output pin 22 (GP[22]). This is the default mode at reset and the pin is configured as an Input. Approximately 10 cycles after the rising edge of RESET, the state on this pin is latched into the BootMode register to specify the boot method.
0	A15_MODE	A15 Pin Mode Bit. This bit controls the pin multiplexing of the EMIF address 15 (EM_A[15]) and general-purpose input/output pin 21 (GP[21]) pin functions. For more details, see Table 1-55, *MMC0, I2S0, McBSP, and GP[5:0] Pin Multiplexing*. 0 = Pin function is EMIF address pin 15 (EM_A[15]). 1 = Pin function is general-purpose input/output pin 21 (GP[21]). This is the default mode at reset and the pin is configured as an Input. Approximately 10 cycles after the rising edge of RESET, the state on this pin is latched into the BootMode register to specify the boot method.

1.7.3.2 Multiplexed Pin Configurations

The device uses pin multiplexing to accommodate a larger number of peripheral functions in the smallest possible package, providing the ultimate flexibility for end applications. The external bus selection register (EBSR) controls all the pin multiplexing functions on the device.

1.7.3.2.1 Pin Multiplexing Details

This section discusses how to program the external bus selection register (EBSR) to select the desired peripheral functions and pin muxing. See the individual subsections for muxing details for a specific muxed pin. After changing any of the pin mux control registers, it will be necessary to reset the peripherals that are affected.

1.7.3.2.1.1 UHPI, SPI, UART, I2S2, I2S3, and GP[31:27, 20:12] Pin Multiplexing [EBSR.PPMODE Bits]

The UHPI, SPI, UART, I2S2, I2S3, and GPIO signal muxing is determined by the value of the PPMODE bit fields in the External Bus Selection Register (EBSR) register. For more details on the actual pin functions, see Table 1-53.

Table 1-53. UHPI, SPI, UART, I2S2, I2S3, and GP[31:27, 20:12] Pin Multiplexing

PULLUP/PULLDOWN CONTROL REGISTER BIT	PIN NUMBER	EBSR PPMODE BITS						
		MODE 0	MODE 1	MODE 2	MODE 3	MODE 4	MODE 5	MODE 6
		000	001 (Reset Default)	010	011	100	101	110
PUDINHIBR7 (1C50h) Bit 12	N3	UHPI_HINT	SPI_CLK	Reserved	Reserved	Reserved	Reserved	SPI_CLK
PUDINHIBR3 (1C19h) Bit 0	P6	UHPI_HD[0]	SPI_RX	Reserved	Reserved	Reserved	Reserved	SPI_RX
PUDINHIBR3 (1C19h) Bit 1	N6	UHPI_HD[1]	SPI_TX	Reserved	Reserved	Reserved	Reserved	SPI_TX
PUDINHIBR3 (1C19h) Bit 2	P7	UHPI_HD[2]	GP[12]	Reserved	Reserved	Reserved	Reserved	GP[12]
PUDINHIBR3 (1C19h) Bit 3	N7	UHPI_HD[3]	GP[13]	Reserved	Reserved	Reserved	Reserved	GP[13]
PUDINHIBR3 (1C19h) Bit 4	N8	UHPI_HD[4]	GP[14]	Reserved	Reserved	Reserved	Reserved	GP[14]
PUDINHIBR3 (1C19h) Bit 5	P9	UHPI_HD[5]	GP[15]	Reserved	Reserved	Reserved	Reserved	GP[15]
PUDINHIBR3 (1C19h) Bit 6	N9	UHPI_HD[6]	GP[16]	Reserved	Reserved	Reserved	Reserved	GP[16]
PUDINHIBR3 (1C19h) Bit 7	P10	UHPI_HD[7]	GP[17]	Reserved	Reserved	Reserved	Reserved	GP[17]
PUDINHIBR3 (1C19h) Bit 8	N10	UHPI_HD[8]	I2S2_CLK	GP[18]	SPI_CLK	I2S2_CLK	SPI_CLK	I2S2_CLK
PUDINHIBR3 (1C19h) Bit 9	P11	UHPI_HD[9]	I2S2_FS	GP[19]	SPI_CS0	I2S2_FS	SPI_CS0	I2S2_FS
PUDINHIBR3 (1C19h) Bit 10	N11	UHPI_HD[10]	I2S2_RX	GP[20]	SPI_RX	I2S2_RX	SPI_RX	I2S2_RX
PUDINHIBR3 (1C19h) Bit 11	P12	UHPI_HD[11]	I2S2_DX	GP[27]	SPI_TX	I2S2_DX	SPI_TX	I2S2_DX
PUDINHIBR3 (1C19h) Bit 12	N12	UHPI_HD[12]	UART_RTS	GP[28]	I2S3_CLK	UART_RTS	UART_RTS	I2S3_CLK
PUDINHIBR3 (1C19h) Bit 13	P13	UHPI_HD[13]	UART_CTS	GP[29]	I2S3_FS	UART_CTS	UART_CTS	I2S3_FS
PUDINHIBR3 (1C19h) Bit 14	N13	UHPI_HD[14]	UART_RXD	GP[30]	I2S3_RX	UART_RXD	UART_RXD	I2S3_RX
PUDINHIBR3 (1C19h) Bit 15	P14	UHPI_HD[15]	UART_TXD	GP[31]	I2S3_DX	UART_TXD	UART_TXD	I2S3_DX
PUDINHIBR7 (1C50h) Bit 8	P4	UHPI_HCNTL0	SPI_CS0	Reserved	Reserved	Reserved	Reserved	SPI_CS0
PUDINHIBR7 (1C50h) Bit 9	N4	UHPI_HCNTL1	SPI_CS1	Reserved	Reserved	Reserved	Reserved	SPI_CS1
PUDINHIBR7 (1C50h) Bit 10	P5	UHPI_HR_RW	SPI_CS2	Reserved	Reserved	Reserved	Reserved	SPI_CS2
PUDINHIBR7 (1C50h) Bit 11	N5	UHPI_HRDY	SPI_CS3	Reserved	Reserved	Reserved	Reserved	SPI_CS3
PUDINHIBR6 (1C4Fh) Bit 7	B5	UHPI_HBE0	EM_DQM0	EM_DQM0	EM_DQM0	EM_DQM0	EM_DQM0	EM_D1M0
PUDINHIBR6 (1C4Fh) Bit 8	P1	UHPI_HBE1	EM_DQM1	EM_DQM1	EM_DQM1	EM_DQM1	EM_DQM1	EM_DQM1
PUDINHIBR7 (1C50h) Bit 3	A6	UHPI_HAS	$\overline{\text{EM\_SDRAS}}$	$\overline{\text{EM\_SDRAS}}$	$\overline{\text{EM\_SDRAS}}$	$\overline{\text{EM\_SDRAS}}$	$\overline{\text{EM\_SDRAS}}$	$\overline{\text{EM\_SDRAS}}$
PUDINHIBR7 (1C50h) Bit 2	B4	UHPI_HCS	$\overline{\text{EM\_SDCAS}}$	$\overline{\text{EM\_SDCAS}}$	$\overline{\text{EM\_SDCAS}}$	$\overline{\text{EM\_SDCAS}}$	$\overline{\text{EM\_SDCAS}}$	$\overline{\text{EM\_SDCAS}}$
PUDINHIBR7 (1C50h) Bit 4	B3	UHPI_HDS1	$\overline{\text{EM\_CS0}}$	$\overline{\text{EM\_CS0}}$	$\overline{\text{EM\_CS0}}$	$\overline{\text{EM\_CS0}}$	$\overline{\text{EM\_CS0}}$	$\overline{\text{EM\_CS0}}$
PUDINHIBR7 (1C50h) Bit 5	A4	UHPI_HDS2	$\overline{\text{EM\_CS1}}$	$\overline{\text{EM\_CS1}}$	$\overline{\text{EM\_CS1}}$	$\overline{\text{EM\_CS1}}$	$\overline{\text{EM\_CS1}}$	$\overline{\text{EM\_CS1}}$
PUDINHIBR7 (1C50h) Bit 1	N2	UHPI_HHWIL	EM_SDCKE	EM_SDCKE	EM_SDCKE	EM_SDCKE	EM_SDCKE	EM_SDCKE

1.7.3.2.1.2 MMC1, McSPI, and GP[11:6] Pin Multiplexing [EBSR.SP1MODE Bits]

The MMC1, McSPI, and GPIO signal muxing is determined by the value of the SP1MODE bit fields in the External Bus Selection Register (EBSR) register. For more details on the actual pin functions, see Table 1-54.

Table 1-54. MMC1, McSPI, and GP[11:6] Pin Multiplexing

PUDINHIBR1 REGISTER BIT[1]	PIN NUMBER	EBSR SP1MODE BITS		
		MODE 0	MODE 1	MODE 2
		00 (Reset Default)	01	10
Bit 8	M13	MMC1_CLK	McSPI_CLK	GP[6]
Bit 9	L14	MMC1_CMD	McSPI_CS0	GP[7]
Bit 10	M14	MMC1_D0	McSPI_SIMO	GP[8]
Bit 11	M12	MMC1_D1	McSPI_SOMI	GP[9]
Bit 12	K14	MMC1_D2	McSPI_CS1	GP[10]
Bit 13	L13	MMC1_D3	McSPI_CS2	GP[11]

(1) The pin names with PUDINHIBR1 (1C17h) register bit field references can have the pulldown register enabled or disabled via this register.

1.7.3.2.1.3 MMC0, I2S0, McBSP, and GP[5:0] Pin Multiplexing [EBSR.SP0MODE Bits]

The MMC0, I2S0, McBSP, and GPIO signal muxing is determined by the value of the SP0MODE bit fields in the External Bus Selection Register (EBSR) register. For more details on the actual pin functions, see Table 1-55.

Table 1-55. MMC0, I2S0, McBSP, and GP[5:0] Pin Multiplexing

PUDINHIBR1 REGISTER BIT[1]	PIN NUMBER	EBSR SP0MODE BITS			
		MODE 0	MODE 1	MODE 2	MODE 3
		00 (Reset Default)	01	10	11
Bit 0	L10	MMC0_CLK	I2S0_CLK	GP[0]	McBSP_CLKX
Bit 1	M11	MMC0_CMD	I2S0_FS	GP[1]	McBSP_FSX
Bit 2	L9	MMC0_D0	I2S0_DX	GP[2]	McBSP_DX
Bit 3	M10	MMC0_D1	I2S0_RX	GP[3]	McBSP_DR
Bit 4	L12	MMC0_D2	GP[4]	GP[4]	McBSP_FSR
Bit 5	L11	MMC0_D3	GP[5]	GP[5]	McBSP_CLKR_CLKS[2]

(1) The pin names with PUDINHIBR1 (1C17H) register bit field references can have the pulldown register enabled or disabled via this register.

(2) Bit 15 of the EBSR register determines this port to be McBSP_CLKR or McBSP_CLKS.

1.7.3.2.1.4 EMIF EM_A[20:15] and GP[26:21] Pin Multiplexing [EBSR.Axx_MODE bits]

The EMIF Address and GPIO signal muxing is determined by the value of the A20_MODE, A19_MODE, A18_MODE, A17_MODE, A16_MODE, and A15_MODE bit fields in the External Bus Selection Register (EBSR) register. For more details on the actual pin functions, see Table 1-56.

Table 1-56. EM_A[20:16] and GP[26:21] Pin Multiplexing

PUDINHIBR2 REGISTER BIT(1)	PIN NUMBER	Axx_MODE BIT	
		0	1
Bit 0	N1	EM_A[15]	GP[21]
Bit 1	E2	EM_A[16]	GP[22]
Bit 2	F2	EM_A[17]	GP[23]
Bit 3	G2	EM_A[18]	GP[24]
Bit 4	G4	EM_A[19]	GP[25]
Bit 5	J3	EM_A[20]	GP[26]

(1) The pin names with PUDINHIBR2 (1C18h) register bit field references can have the pulldown register enabled or disabled via this register.

1.7.3.3 LDO Control Register [7004h]

When the DSP_LDO is enabled by the $\overline{\text{DSP\_LDO\_EN}}$ pin [D12], by default, the DSP_LDOO voltage is set to 1.3 V. The DSP_LDOO voltage can be programmed to be either 1.05 V or 1.3 V via the DSP_LDO_V bit (bit 1) in the LDO Control Register (LDOCNTL).

USB_LDO Enable bit (USBLDOEN bit 0) reset state depends on the setting of CLK_SEL pin at reset. If CLK_SEL is high, the USB LDO is disabled. If CLK_SEL is low, the USB LDO is enabled.

1.7.3.4 LDO Control

All three LDOs can be simultaneously disabled via software by writing to either the BG_PD bit or the LDO_PD bit in the RTCPMGT register (see Figure 1-46). When the LDOs are disabled via this mechanism, the only way to re-enable them is by asserting the WAKEUP signal pin (which must also have been previously enabled to allow wakeup), *or* by a previously enabled and configured RTC alarm, *or* by cycling power to the CV_{DDRTC} pin.

ANA_LDO: Disabled only by the BG_PD and the LDO_PD mechanism, previously described. Otherwise, it is always enabled.

DSP_LDO: Can be statically disabled by the $\overline{\text{DSP\_LDO\_EN}}$ pin. It can be also dynamically disabled via the BG_PD and the LDO_PD mechanism described above. The DSP_LDO can change its output voltage dynamically by software via the DSP_LDO_V bit in the LDOCNTL register (see Figure 1-47). The DSP_LDO output voltage is set to 1.3 V at reset.

USB_LDO: The reset state of the USB_LDO is dependent on the setting of CLK_SEL pin at reset, described in the previous section. If CLK_SEL is high, the USB_LDO is disabled but can be independently and dynamically enabled or disabled by software via the USB_LDO_EN bit in the LDOCNTL register (see Figure 1-47). If CLK_SEL is low, the USB_LDO is enabled at reset and can never be disabled. This is to ensure the USB oscillator has power when it is the source of the system clock.

Table 1-59 shows the ON/OFF control of each LDO and its register control bit configurations.

Figure 1-46. RTC Power Management Register (RTCPMGT) [1930h]

15						8
Reserved						
R-0						
7	5	4	3	2	1	0
Reserved		WU_DOUT	WU_DIR	BG_PD	LDO_PD	RTCCLKOUTEN
R-0		R/W-1	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-57. RTCPMGT Register Bit Descriptions Field Descriptions

Bit	Field	Value	Description
15-5	Reserved	0	Reserved. Read-only, writes have no effect.
4	WU_DOUT		Wakeup output, active low/open-drain.
		0	WAKEUP pin driven low.
		1	WAKEUP pin is in high-impedance (Hi-Z).
3	WU_DIR		Wakeup pin direction control.
		0	WAKEUP pin configured as a input.
		1	WAKEUP pin configured as a output.
			Note: When the WAKEUP pin is configured as an input, it is active high. When the WAKEUP pin is configured as an output, is an open-drain that is active low and should be externally pulled-up via a 10-kΩ resistor to DV_{DDRTC}. WU_DIR must be configured as an input to allow the WAKEUP pin to wake the device up from idle modes.
2	BG_PD		Bandgap, on-chip LDOs, and the analog POR power down bit.
			This bit shuts down the on-chip LDOs (ANA_LDO, DSP_LDO, and USB_LDO), the Analog POR, and Bandgap reference. BG_PD and LDO_PD are only intended to be used when the internal LDOs supply power to the chip. If the internal LDOs are bypassed and not used then the BG_PD and LDO_PD power down mechanisms should not be used since POR gets powered down and the POWERGOOD signal is not generated properly.
			After this bit is asserted, the on-chip LDOs, Analog POR, and the Bandgap reference can be re-enabled by the WAKEUP pin (high) or the RTC alarm interrupt. The Bandgap circuit will take about 100 msec to charge the external 0.1 uF capacitor via the internal 326-kΩ resistor.
		0	On-chip LDOs, Analog POR, and Bandgap reference are enabled.
		1	On-chip LDOs, Analog POR, and Bandgap reference are disabled (shutdown).
1	LDO_PD		On-chip LDOs and Analog POR power down bit.
			This bit shuts down the on-chip LDOs (ANA_LDO, DSP_LDO, and USB_LDO) and the Analog POR. BG_PD and LDO_PD are only intended to be used when the internal LDOs supply power to the chip. If the internal LDOs are bypassed and not used then the BG_PD and LDO_PD power down mechanisms should not be used since POR gets powered down and the POWERGOOD signal is not generated properly.
			After this bit is asserted, the on-chip LDOs and Analog POR can be re-enabled by the WAKEUP pin (high) or the RTC alarm interrupt. This bit keeps the Bandgap reference turned on to allow a faster wake-up time with the expense power consumption of the Bandgap reference.
		0	On-chip LDOs and Analog POR are enabled.
		1	On-chip LDOs and Analog POR are disabled (shutdown).
0	RTCCLKOUTEN		Clockout output enable bit
		0	Clock output disabled
		1	Clock output enabled

Figure 1-47. LDO Control Register (LDOCNTL) [7004h]

15							8
Reserved							
R-0							

7					2	1	0
Reserved						DSP_LDO_V	USB_LDO_EN
R-0						R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-58. LDOCNTL Register Bit Descriptions Field Descriptions

Bit	Field	Value	Description
15-2	Reserved	0	Reserved. Read-only, writes have no effect.
1	DSP_LDO_V		DSP_LDO voltage select bit.
		0	DSP_LDOO is regulated to 1.3 V.
		1	DSP_LDOO is regulated to 1.05 V.
0	USB_LDO_EN		USB_LDO enable bit. The USB_LDO reset state depends on the setting of the CSL_SEL pin at reset. If CLK_SEL is high, the USB LDO is disabled. If CLK_SEL is low, the USB LDO is enabled.
		0	USB_LDO output is disabled. USB_LDOO pin is placed in high-impedance (Hi-Z) state.
		1	USB_LDO output is enabled. USB_LDOO is regulated to 1.3 V.

Table 1-59. LDO Controls Matrix

RTCPMGT Register (1930h)		LDOCNTL Register (7004h)	$\overline{\text{DSP\_LDO\_EN}}$	CLK_SEL	ANA_LDO	DSP_LDO	USB_LDO
BG_PD Bit	LDO_PD Bit	USB_LDO_EN Bit					
1	Don't Care	Don't Care	Don't Care	0	OFF	OFF	OFF
Don't Care	1	Don't Care	Don't Care	0	OFF	OFF	OFF
0	0	Don't Care	Low	0	ON	ON	ON
0	0	Don't Care	High	0	ON	OFF	ON
1	Don't Care	Don't Care	Don't Care	1	OFF	OFF	OFF
Don't Care	1	Don't Care	Don't Care	1	OFF	OFF	OFF
0	0	0	Low	1	ON	ON	OFF
0	0	0	High	1	ON	OFF	OFF
0	0	1	Low	1	ON	ON	ON
0	0	1	High	1	ON	OFF	ON

1.7.3.5 Output Slew Rate Control Register (OSRCR) [1C16h]

To provide the lowest power consumption setting, the DSP has configurable slew rate control on the EMIF and CLKOUT output pins. The output slew rate control register (OSRCR) is used to set a subset of the device I/O pins, namely CLKOUT and EMIF pins, to either fast or slow slew rate. The slew rate feature is implemented by staging/delaying turn-on times of the parallel p-channel drive transistors and parallel n-channel drive transistors of the output buffer. In the slow slew rate configuration, the delay is longer, but ultimately the same number of parallel transistors are used to drive the output high or low; therefore, the drive strength is ultimately the same. The slower slew rate control can be used for power savings and has the greatest effect at lower DV_{DDIO} and DV_{DDEMIF} voltages.

The output slew rate control register (OSRCR) is shown in Figure 1-48 and described in Table 1-60.

Figure 1-48. Output Slew Rate Control Register (OSRCR) [1C16h]

15	3	2	1	0
Reserved		CLKOUTSR	Reserved	EMIFSR
R-0		RW-1	R-0	RW-1

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-60. Output Slew Rate Control Register (OSRCR) Field Descriptions

Bit	Field	Value	Description
15-3	Reserved	0	Reserved.
2	CLKOUTSR		CLKOUT pin output slew rate bits. These bits set the slew rate for the CLKOUT pin.
		0	Slow slew rate
		1	Fast slew rate
1	Reserved	0	Reserved.
0	EMIFSR		EMIF pin output slew rate bits. These bits set the slew rate for the EMIF pins.
		0	Slow slew rate
		1	Fast slew rate

1.7.3.6 Pullup/Pulldown Inhibit Registers (PUDINHIBR1–7)

The device allows you to individually enable or disable the internal pullup and pulldown resistors. You can individually inhibit the pullup and pulldown resistors of the I/O pins through the pullup and pulldown inhibit registers (PUDINHIBRn). There is one pin, TRSTN, that has a pulldown that is permanently enabled and cannot be disabled.

The pullup and pulldown inhibit register 1 (PUDINHIBR1) is shown in Figure 1-49 and described in Table 1-61.

Figure 1-49. Pullup and Pulldown Inhibit Register 1 (PUDINHIBR1) [1C17h]

15	14	13	12	11	10	9	8
Reserved		S15PD	S14PD	S13PD	S12PD	S11PD	S10PD
R-0		R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
7	6	5	4	3	2	1	0
Reserved		S05PD	S04PD	S03PD	S02PD	S01PD	S00PD
R-0		R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-61. Pullup and Pulldown Inhibit Register 1 (PUDINHIBR1) Field Descriptions

Bit	Field	Value	Description
15-14	Reserved	0	Reserved.
13	S15PD		Serial port 1 pin 5 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
12	S14PD		Serial port 1 pin 4 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
11	S13PD		Serial port 1 pin 3 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
10	S12PD		Serial port 1 pin 2 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
9	S11PD		Serial port 1 pin 1 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
8	S10PD		Serial port 1 pin 0 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
7-6	Reserved	0	Reserved.
5	S05PD		Serial port 0 pin 5 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
4	S04PD		Serial port 0 pin 4 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
3	S03PD		Serial port 0 pin 3 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.

Table 1-61. Pullup and Pulldown Inhibit Register 1 (PUDINHIBR1) Field Descriptions (continued)

Bit	Field	Value	Description
2	S02PD		Serial port 0 pin 2 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
1	S01PD		Serial port 0 pin 1 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
0	S00PD		Serial port 0 pin 0 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.

The pullup and pulldown inhibit register 2 (PUDINHIBR2) is shown in Figure 1-50 and described in Table 1-62.

Figure 1-50. Pullup and Pulldown Inhibit Register 2 (PUDINHIBR2) [1C18h]

15	14	13	12	11	10	9	8
CLKINPD	INT1PU	INT0PU	RESETPU	EMU01PU	TDIPU	TMSPU	TCKPU
R/W-1	R/W-1	R/W-1	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
TRSTPD	TDOPU	A20PD	A19PD	A18PD	A17PD	A16PD	A15PD
R/W-0	R/W-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-62. Pullup and Pulldown Inhibit Register 2 (PUDINHIBR2) Field Descriptions

Bit	Field	Value	Description
15	CLKINPD		CLKIN pin pulldown inhibit bit.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
14	INT1PU		Interrupt 1 pin pullup inhibit bit. Setting this bit to 1 disables the pin's internal pullup.
		0	Pin pullup is enabled.
		1	Pin pullup is disabled.
13	INT0PU		Interrupt 0 pin pullup inhibit bit. Setting this bit to 1 disables the pin's internal pullup.
		0	Pin pullup is enabled.
		1	Pin pullup is disabled.
12	RESETPU		Reset pin pullup inhibit bit. Setting this bit to 1 disables the pin's internal pullup.
		0	Pin pullup is enabled.
		1	Pin pullup is disabled.
11	EMU01PU		EMU1 and EMU0 pin pullup inhibit bit. Setting this bit to 1 disables the pin's internal pullup.
		0	Pin pullup is enabled.
		1	Pin pullup is disabled.
10	TDIPU		TDI pin pullup inhibit bit. Setting this bit to 1 disables the pin's internal pullup.
		0	Pin pullup is enabled.
		1	Pin pullup is disabled.
9	TMSPU		TMS pin pullup inhibit bit. Setting this bit to 1 disables the pin's internal pullup.
		0	Pin pullup is enabled.
		1	Pin pullup is disabled.

Table 1-62. Pullup and Pulldown Inhibit Register 2 (PUDINHIBR2) Field Descriptions (continued)

Bit	Field	Value	Description
8	TCKPU		TCK pin pullup inhibit bit. Setting this bit to 1 disables the pin's internal pullup.
		0	Pin pullup is enabled.
		1	Pin pullup is disabled.
7	TRSTPD		TRST pin pulldown inhibit bit.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
6	TDOPU		TDOPU pin pullup inhibit bit.
		0	Pin pullup is enabled.
		1	Pin pulldown is disabled.
5	A20PD		EMIF A[20] pin pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
4	A19PD		EMIF A[19] pin pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
3	A18PD		EMIF A[18] pin pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
2	A17PD		EMIF A[17] pin pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
1	A16PD		EMIF A[16] pin pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
0	A15PD		EMIF A[15] pin pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.

The pullup and pulldown inhibit register 3 (PUDINHIBR3) is shown in Figure 1-51 and described in Table 1-63.

Figure 1-51. Pullup and Pulldown Inhibit Register 3 (PUDINHIBR3) [1C19h]

15	14	13	12	11	10	9	8
PD15PD	PD14PD	PD13PD	PD12PD	PD11PD	PD10PD	PD9PD	PD8PD
R/W-1	R/W-0	R/W-0	R/W-1	R/W-1	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
PD7PD	PD6PD	PD5PD	PD4PD	PD3PD	PD2PD	PD1PD	PD0PD
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-63. Pullup and Pulldown Inhibit Register 3 (PUDINHIBR3) Field Descriptions

Bit	Field	Value	Description
15	PD15PD		Parallel port pin 15 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
14	PD14PD		Parallel port pin 14 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.

Table 1-63. Pullup and Pulldown Inhibit Register 3 (PUDINHIBR3) Field Descriptions (continued)

Bit	Field	Value	Description
13	PD13PD		Parallel port pin 13 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
12	PD12PD		Parallel port pin 12 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
11	PD11PD		Parallel port pin 11 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
10	PD10PD		Parallel port pin 10 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
9	PD9PD		Parallel port pin 9 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
8	PD8PD		Parallel port pin 8 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
7	PD7PD		Parallel port pin 7 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
6	PD6PD		Parallel port pin 6 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
5	PD5PD		Parallel port pin 5 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
4	PD4PD		Parallel port pin 4 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
3	PD3PD		Parallel port pin 3 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
2	PD2PD		Parallel port pin 2 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
1	PD1PD		Parallel port pin 1 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
0	PD0PD		Parallel port pin 0 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.

The pullup and pulldown inhibit register 4 is shown in Figure 1-52 and described in Table 1-64.

Figure 1-52. Pullup and Pulldown Inhibit Register 4 (PUDINHIBR4) [1C4Ch]

15	14	13	12	11	10	9	8
D15PD	D14PD	D13PD	D12PD	D11PD	D10PD	D9PD	D8PD
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
D7PD	D6PD	D5PD	D4PD	D3PD	D2PD	D1PD	D0PD
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-64. Pullup and Pulldown Inhibit Register 4 (PUDINHIBR4) Field Descriptions

Bit	Field	Value	Description
15	D15PD		Parallel port pin 15 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
14	P14PD		Parallel port pin 14 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
13	D13PD		Parallel port pin 13 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
12	D12PD		Parallel port pin 12 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
11	D11PD		Parallel port pin 11 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
10	D10PD		Parallel port pin 10 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
9	D9PD		Parallel port pin 9 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
8	D8PD		Parallel port pin 8 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
7	D7PD		Parallel port pin 7 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
6	D6PD		Parallel port pin 6 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
5	D5PD		Parallel port pin 5 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
4	D4PD		Parallel port pin 4 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
3	D3PD		Parallel port pin 3 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.

Table 1-64. Pullup and Pulldown Inhibit Register 4 (PUDINHIBR4) Field Descriptions (continued)

Bit	Field	Value	Description
2	D2PD		Parallel port pin 2 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
1	D1PD		Parallel port pin 1 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
0	D0PD		Parallel port pin 0 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.

The pullup and pulldown inhibit register 5 is shown in Figure 1-53 and described in Table 1-65.

Figure 1-53. Pullup and Pulldown Inhibit Register 5 (PUDINHIBR5) [1C4Dh]

15	14	13	12	11	10	9	8
Reserved	A14PD	A13PD	A12PD	A11PD	A10PD	A9PD	A8PD
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
A7PD	A6PD	A5PD	A4PD	A3PD	A2PD	A1PD	A0PD
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-65. Pullup and Pulldown Inhibit Register 5 (PUDINHIBR5) Field Descriptions

Bit	Field	Value	Description
15	Reserved	0	Reserved.
14	A14PD		Parallel port pin 14 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
13	A13PD		Parallel port pin 13 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
12	A12PD		Parallel port pin 12 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
11	A11PD		Parallel port pin 11 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
10	A10PD		Parallel port pin 10 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
9	A9PD		Parallel port pin 9 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
8	A8PD		Parallel port pin 8 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
7	A7PD		Parallel port pin 7 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.

Table 1-65. Pullup and Pulldown Inhibit Register 5 (PUDINHIBR5) Field Descriptions (continued)

Bit	Field	Value	Description
6	A6PD		Parallel port pin 6 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
5	A5PD		Parallel port pin 5 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
4	A4PD		Parallel port pin 4 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
3	A3PD		Parallel port pin 3 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
2	A2PD		Parallel port pin 2 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
1	A1PD		Parallel port pin 1 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.
0	A0PD		Parallel port pin 0 pulldown inhibit bit. Setting this bit to 1 disables the pin's internal pulldown.
		0	Pin pulldown is enabled.
		1	Pin pulldown is disabled.

The pullup and pulldown inhibit register 6 is shown in Figure 1-54 and described in Table 1-66.

Figure 1-54. Pullup and Pulldown Inhibit Register 6 (PUDINHIBR6) [1C4Fh]

15	14	13	12	11	10	9	8
Reserved	NANDRDY1PD	NANDRDY0PD	ASYNCRDY1PD	ASYNCRDY0PD	BA1PD	BA0PD	DQM1PD
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

7	6	5	4	3	2	1	0
DQM0PD	RNWPD	OEPD	WEPD	NANDCE1PD	NANDCE0PD	ASYNCCE1PD	ASYNCCE0PD
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-66. Pullup and Pulldown Inhibit Register 6 (PUDINHIBR6) Field Descriptions

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	Reserved.
14	NANDRDY1PD	R/W	0h	EMIF NAND_RDY[1] pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
13	NANDRDY0PD	R/W	0h	EMIF NAND_RDY[0] pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
12	ASYNCRDY1PD	R/W	0h	EMIF ASYNC_RDY[1] pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.

Table 1-66. Pullup and Pulldown Inhibit Register 6 (PUDINHIBR6) Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	ASYNCRDY0PD	R/W	0h	EMIF ASYNC_RDY[0] pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
10	BA1PD	R/W	0h	EMIF BA[1] pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
9	BA0PD	R/W	0h	EMIF BA[0] pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
8	DQM1PD	R/W	0h	EMIF DQM[1] pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
7	DQM0PD	R/W	0h	EMIF DQM[0] pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
6	RNWPD	R/W	0h	EMIF RnW pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
5	OEPD	R/W	0h	EMIF OE pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
4	WEPD	R/W	0h	EMIF WE pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
3	NANDCE1PD	R/W	0h	EMIF NAND_CE[1] pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
2	NANDCE0PD	R/W	0h	EMIF NAND_CE[0] pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
1	ASYNCCE1PD	R/W	0h	EMIF ASYNC_CE[1] PD pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
0	ASYNCCE0PD	R/W	0h	EMIF ASYNC_CE[0] PD pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.

The pullup and pulldown inhibit register 7 is shown in Figure 1-55 and described in Table 1-67.

Figure 1-55. Pullup and Pulldown Inhibit Register 7 (PUDINHIBR7) [1C50h]

15	14	13	12	11	10	9	8
Reserved			PCLKPD	PCNTL3PD	PCNTL2PD	PCNTL1PD	PCNTL0PD
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
Reserved		SDCE1PD	SDCE0PD	SDRASPD	SDCASPD	SDCKEPD	SDCLKPD
R-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 1-67. Pullup and Pulldown Inhibit Register 7 (PUDINHIBR7) Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved.
12	PCLKPD	R/W	0h	PCLK pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
11	PCNTL3PD	R/W	0h	PCNTL[3] pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
10	PCNTL2PD	R/W	0h	PCNTL[2] pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
9	PCNTL1PD	R/W	0h	PCNTL[1] pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
8	PCNTL0PD	R/W	0h	PCNTL[0] pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
7-6	Reserved	R	0h	Reserved.
5	SDCE1PD	R/W	0h	EMIF SD_CE[1] pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
4	SDCE0PD	R/W	0h	EMIF SD_CE[0] pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
3	SDRASPD	R/W	0h	EMIF SDRAS pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
2	SDCASPD	R/W	0h	EMIF SDCAS pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.
1	SDCKEPD	R/W	0h	EMIF SDCKE PD pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.

Table 1-67. Pullup and Pulldown Inhibit Register 7 (PUDINHIBR7) Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	SDCLKPD	R/W	0h	EMIF SDCLK PD pin pulldown inhibit bit. Setting this bit to 1 disables the pin internal pulldown. 0x0 = Pin pulldown is enabled. 0x1 = Pin pulldown is disabled.

1.7.4 DMA Controller Configuration

The DSP includes four DMA controllers that allow movement of blocks of data among internal memory, external memory, and peripherals to occur without intervention from the CPU and in the background of CPU operation. Each DMA has an EVENT input signal (per channel) that can be used to tell it when to start the block transfer. And each DMA has an interrupt output (per channel) that can signal the CPU when the block transfer is completed. The EVENT source and interrupt aggregation is more of a system-level concern and, therefore, they are best described here.

The following sections provide more details on these features. In this section and subsections, the following notations will be used:

- Lowercase, italicized, *n* is an integer, 0-3, representing each of the 4 DMAs.
- Lowercase, italicized, *m* is an integer, 0-3, representing each of the 4 channels within each DMA.

1.7.4.1 DMA Synchronization Events

The DMA controllers allow activity in their channels to be synchronized to selected events. The DSP supports 27 separate synchronization events and each channel can be tied to separate sync events independent of the other channels. Synchronization events are selected by programming the CH*n*EVT field in the DMA*n* channel event source registers (DMA*n*CESR1 and DMA*n*CESR2) (where *n* is an integer, 0-3, representing each of the 4 DMAs). The synchronization events available to each DMA controller are shown in Table 1-68.

Table 1-68. Channel Synchronization Events for DMA Controllers

CH*m*EVT Options	DMA0 Synchronization Event	DMA1 Synchronization Event	DMA2 Synchronization Event	DMA3 Synchronization Event
0000b	Reserved	Reserved	Reserved	Reserved
0001b	I2S0 transmit event	I2S2 transmit event	I2C transmit event	McBSP transmit event
0010b	I2S0 receive event	I2S2 receive event	I2C receive event	McBSP receive event
0011b	McSPI channel 0 read event	Reserved	SAR A/D event	McBSP A-bis mode transmit event
0100b	McSPI channel 0 write event	Reserved	I2S3 transmit event	McBSP A-bis mode receive event
0101b	MMC/SD0 transmit event	UART transmit event	I2S3 receive event	Reserved
0110b	MMC/SD0 receive event	UART receive event	Reserved	Reserved
0111b	MMC/SD1 transmit event	Reserved	Reserved	Reserved
1000b	MMC/SD1 receive event	Reserved	Reserved	Reserved
1001b	McSPI channel 1 read event	Reserved	Reserved	Reserved
1010v	McSPI channel 1 write event	Reserved	Reserved	Reserved
1011b	McSPI channel 2 read event	Reserved	Reserved	Reserved
1100b	Timer 0 event	Timer 0 event	Timer 0 event	Timer 0 event
1101b	Timer 1 event	Timer 1 event	Timer 1 event	Timer 1 event
1110b	Timer 2 event	Timer 2 event	Timer 2 event	Timer 2 event
1111b	McSPI channel 2 write event	Reserved	Reserved	Reserved

1.7.4.2 DMA Configuration Registers

The system-level DMA registers are listed in Table 1-69. The DMA interrupt flag and enable registers (DMAIFR and DMAIER) are used to control the aggregation and CPU interrupt generation for the four DMA controllers and their associated channels. In addition, there are two registers per DMA controller which control event synchronization in each channel; the DMA*n* channel event source registers (DMA*n*CESR1 and DMA*n*CESR2).

Table 1-69. System Registers Related to the DMA Controllers

CPU Word Address	Acronym	Register Description	Section
1C30h	DMAIFR	DMA Interrupt Flag Register	Section 1.7.4.2.1
1C31h	DMAIER	DMA Interrupt Enable Register	Section 1.7.4.2.1
1C1Ah	DMA0CESR1	DMA0 Channel Event Source Register 1	Section 1.7.4.2.2
1C1Bh	DMA0CESR2	DMA0 Channel Event Source Register 2	Section 1.7.4.2.2
1C1Ch	DMA1CESR1	DMA1 Channel Event Source Register 1	Section 1.7.4.2.2
1C1Dh	DMA1CESR2	DMA1 Channel Event Source Register 2	Section 1.7.4.2.2
1C36h	DMA2CESR1	DMA2 Channel Event Source Register 1	Section 1.7.4.2.2
1C37h	DMA2CESR2	DMA2 Channel Event Source Register 2	Section 1.7.4.2.2
1C38h	DMA3CESR1	DMA3 Channel Event Source Register 1	Section 1.7.4.2.2
1C39h	DMA3CESR2	DMA3 Channel Event Source Register 2	Section 1.7.4.2.2

1.7.4.2.1 DMA Interrupt Flag Register (DMAIFR) [1C30h] and DMA Interrupt Enable Register (DMAIER) [1C31h]

The DSP includes two registers for aggregating the four channel interrupts of the four DMA controllers. Use the DMA interrupt enable register (DMAIER) to enable channel interrupts. At the end of a block transfer, if the DMA controller channel interrupt enable (DMA*n*CH*m*IE) bit is 1, an interrupt request is sent to the DSP CPU, where it can be serviced or ignored. Each channel can generate an interrupt, although all channel interrupts are aggregated into a single DMA interrupt signal to the CPU.

To see which channel generated an interrupt, your program can read the DMA interrupt flag register (DMAIFR). The DMA controller channel interrupt flag (DMA*n*CH*m*IF) bits are set to 1 when a DMA channel generates an interrupt. Your program must manually clear the bits of DMAIFR by writing a 1 to the bit positions to be cleared.

Figure 1-56. DMA Interrupt Flag Register (DMAIFR) [1C30h]

15	14	13	12	11	10	9	8
DMA3CH3IF	DMA3CH2IF	DMA3CH1IF	DMA3CH0IF	DMA2CH3IF	DMA2CH2IF	DMA2CH1IF	DMA2CH0IF
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
DMA1CH3IF	DMA1CH2IF	DMA1CH1IF	DMA1CH0IF	DMA0CH3IF	DMA0CH2IF	DMA0CH1IF	DMA0CH0IF
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 1-57. DMA Interrupt Enable Register (DMAIER) [1C31h]

15	14	13	12	11	10	9	8
DMA3CH3IE	DMA3CH2IE	DMA3CH1IE	DMA3CH0IE	DMA2CH3IE	DMA2CH2IE	DMA2CH1IE	DMA2CH0IE
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
DMA1CH3IE	DMA1CH2IE	DMA1CH1IE	DMA1CH0IE	DMA0CH3IE	DMA0CH2IE	DMA0CH1IE	DMA0CH0IE
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-70. DMA Interrupt Flag Register (DMAIFR) Field Descriptions

Bit	Field	Value	Description
15-0	DMA*n*CH*m*IF		Channel interrupt status bits.
		0	DMA controller *n,* channel *m* has not completed its block transfer.
		1	DMA controller *n,* channel *m* block transfer complete.

Table 1-71. DMA Interrupt Enable Register (DMAIER) Field Descriptions

Bit	Field	Value	Description
15-0	DMA*n*CH*m*IE		Channel interrupt enable bits.
		0	DMA controller *n,* channel *m* interrupt is disabled.
		1	DMA controller *n,* channel *m* interrupt is enabled.

1.7.4.2.2 DMAn Channel Event Source Registers (DMAnCESR1 and DMAnCESR2) [1C1Ah, 1C1Bh, 1C1Ch, 1C1Dh, 1C36h, 1C37h, 1C38h, and 1C39h]

When SYNCMODE = 1 in a channel's DMACH*m*TCR2 (see Figure 1-30), activity in the DMA controller is synchronized to a DSP event. You can specify the synchronization event used by the DMA channels by programming the CH*m*EVT bits of the DMA*n*CESR registers.

Each DMA controller contains two channel event source registers (DMA*n*CESR1 and DMA*n*CESR2). DMA*n*CESR1 controls the synchronization event for DMA*n* channel 0 and 1 while DMA*n*CESR2 controls the synchronization event for DMA*n* channel 2 and 3.

The synchronization events available to each DMA controller are shown in Table 1-68. Multiple DMAs and multiple channels within a DMA are allowed to have the same synchronization event.

Figure 1-58. DMA*n* Channel Event Source Register 1 (DMA*n*CESR1) [1C1Ah, 1C1Ch, 1C36h, and 1C38h]

15 12	11 8	7 4	3 0
Reserved	CH1EVT	Reserved	CH0EVT
R-0	RW-0	R-0	RW-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 1-59. DMA*n* Channel Event Source Register 2 (DMA*n*CESR2) [1C1Bh, 1C1Dh, 1C37h, and 1C39h]

15 12	11 8	7 4	3 0
Reserved	CH3EVT	Reserved	CH2EVT
R-0	RW-0	R-0	RW-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-72. DMA*n* Channel Event Source Register 1 (DMA*n*CESR1) Field Descriptions

Bit	Field	Value	Description
15-12	Reserved	0	Reserved.
11-8	CH1EVT	0-Fh	Channel 1 synchronization events. When SYNCMODE = 1 in a channel's DMACH*m*TCR2, the CH1EVT bits in the DMA*n*CESR registers specify the synchronization event for activity in the DMA controller. See Table 1-68 for a list of available synchronization event options.
7-4	Reserved	0	Reserved.
3-0	CH0EVT	0-Fh	Channel 0 synchronization events. when SYNCMODE = 1 in a channel's DMACH*m*TCR2, the CH0EVT bits in the DMA*n*CESR registers specify the synchronization event for activity in the DMA controller. See Table 1-68 for a list of available synchronization event options.

Table 1-73. DMA*n* Channel Event Source Register 2 (DMA*n*CESR2) Field Descriptions

Bit	Field	Value	Description
15-12	Reserved	0	Reserved.
11-8	CH3EVT	0-Fh	Channel 3 synchronization events. When SYNCMODE = 1 in a channel's DMACH*m*TCR2, the CH3EVT bits in the DMA*n*CESR registers specify the synchronization event for activity in the DMA controller. See Table 1-68 for a list of available synchronization event options.
7-4	Reserved	0	Reserved.
3-0	CH2EVT	0-Fh	Channel 2 synchronization events. When SYNCMODE = 1 in a channel's DMACH*m*TCR2, the CH2EVT bits in the DMA*n*CESR registers specify the synchronization event for activity in the DMA controller. See Table 1-68 for a list of available synchronization event options.

1.7.5 Peripheral Reset

All peripherals can be reset through software using the peripheral reset control register (PRCR). The peripheral software reset counter register (PSRCR) controls the duration, in SYSCLK cycles, that the reset signal is asserted low once activated by the bits in PRCR.

To reset a peripheral or group of peripherals, follow these steps:

1. Set COUNT = 08h in PSRCR.
2. Initiate the desired peripheral reset by setting to 1 the bits of PRCR.
3. Do not attempt to access the peripheral for at least the number of clock cycles set in the PSRCR register. A repeated NOP may be necessary.

In some cases, a single reset is used for multiple peripherals. For example, PG4_RST controls the reset to I2S2, I2S3, UART, and SPI.

1.7.5.1 Peripheral Software Reset Counter Register (PSRCR) [1C04h]

The Peripheral Software Reset Counter Register (PSRCR) is shown in Table 1-74 and described in Table 1-74.

Figure 1-60. Peripheral Software Reset Counter Register (PSRCR) [1C04h]

15	0
COUNT	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 1-74. Peripheral Software Reset Counter Register (PSRCR) Field Descriptions

Bit	Field	Value	Description
15-0	COUNT	0-FFFFh	Count bits. These bits specify the number of system clock (SYSCLK) cycles the software reset signals are asserted. When the software counter reaches 0, the software reset bits will be cleared to 0. Always initialize this field with a value of at least 08h.

1.7.5.2 Peripheral Reset Control Register (PRCR) [1C05h]

Writing a 1 to any bits in this register initiates the reset sequence for the associated peripherals. The associated peripherals will be held in reset for the duration of clock cycles set in the PSRCR register and they should not be accessed during that time. Reads of this register return the state of the reset signal for the associated peripherals. In other words, polling may be used to wait for the reset to become de-asserted.

The Peripheral Reset Control Register (PRCR) is shown in Figure 1-61 and described in Table 1-75.

Figure 1-61. Peripheral Reset Control Register (PRCR) [1C05h]

15	14	13	12	11	10	9	8
PG4_CLRDIS	McBSP_CLRDIS	PG3_CLRDIS	DMA_CLRDIS	USB_CLRDIS	SAR_ANA_CLRDIS	EMIF_CLRDIS	I2C_CLRDIS
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
PG4_RST	McBSP_RST	PG3_RST	DMA_RST	USB_RST	SAR_ANA_RST	EMIF_RST	I2C_RST
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-75. Peripheral Reset Control Register (PRCR) Field Descriptions

Bit	Field	Value	Description
15	PG4_CLRDIS		Peripheral Group 4 (UHPI, I2S2, I2S3, UART, and SPI) software reset bit.
		0	Self-clearing of software reset is enabled.
		1	Self-clearing of software reset is disabled.
14	McBSP_CLRDIS		McBSP software reset bit.
		0	Self-clearing of software reset is enabled.
		1	Self-clearing of software reset is disabled.
13	PG3_CLRDIS		Peripheral Group 3 (MMC/SD0, MMC/SD1, I2S0 and McSPI) software reset bit.
		0	Self-clearing of software reset is enabled.
		1	Self-clearing of software reset is disabled.
12	DMA_CLRDIS		DMA0, DMA1, DMA2, and DMA3 software reset bit.
		0	Self-clearing of software reset is enabled.
		1	Self-clearing of software reset is disabled.
11	USB_CLRDIS		USB software reset bit.
		0	Self-clearing of software reset is enabled.
		1	Self-clearing of software reset is disabled.
10	SAR_ANA_CLRDIS		10-bit SAR A/D register and analog registers, including TRIM, software reset bit.
		0	Self-clearing of software reset is enabled.
		1	Self-clearing of software reset is disabled.
9	EMIF_CLRDIS		EMIF software reset bit.
		0	Self-clearing of software reset is enabled.
		1	Self-clearing of software reset is disabled.
8	I2C_CLRDIS		I2C software reset bit.
		0	Self-clearing of software reset is enabled.
		1	Self-clearing of software reset is disabled.
7	PG4_RST		Peripheral group 4 software reset bit. Drives the UHPI, I2S2, I2S3, UART, and SPI reset signal.
		Write 0	Writing zero has no effect
		Write 1	Writing one starts resetting the peripheral group
		Read 0	Reading zero means that peripheral group is out of reset
		Read 1	Reading one means the peripheral group is being held in reset and should not be accessed
6	McBSP_RST		Drives the McBSP module reset signal low (asserted) or high (deasserted):
		0	Reset deasserted, module out of reset.
		1	Reset asserted, module in reset.
5	PG3_RST		Peripheral group 3 software reset bit. Drives the MMC/SD0, MMC/SD1, I2S0, and McSPI reset signal.
		Write 0	Writing zero has no effect
		Write 1	Writing one starts resetting the peripheral group
		Read 0	Reading zero means that peripheral group is out of reset
		Read 1	Reading one means the peripheral group is being held in reset and should not be accessed
4	DMA_RST		DMA software reset bit. Drives the reset signal to all four controllers.
		Write 0	Writing zero has no effect
		Write 1	Writing one starts resetting the peripheral group
		Read 0	Reading zero means that peripheral group is out of reset
		Read 1	Reading one means the peripheral group is being held in reset and should not be accessed
3	USB_RST		USB software reset bit. Drives the USB reset signal.
		Write 0	Writing zero has no effect
		Write 1	Writing one starts resetting the peripheral group
		Read 0	Reading zero means that peripheral group is out of reset
		Read 1	Reading one means the peripheral group is being held in reset and should not be accessed

Table 1-75. Peripheral Reset Control Register (PRCR) Field Descriptions (continued)

Bit	Field	Value	Description
2	SAR_ANA_RST		Drives all Analog Registers and the 10-bit SAR A/D module reset signal low (asserted) or high (deasserted), resetting all analog registers, trim settings, and the SAR clock divider to its maximum divide-by value of 32768. Resetting the trim setting can cause the threshold value of the low-voltage detection circuit to change instantaneously while the DSP_LDO voltage changes slower. Thus, it is possible for the threshold to exceed the DSP_LDOO voltage and the comparator to reset the DSP. This can result in an infinite loop with trim being applied, the DSP booting, the SAR/Analog peripheral reset occurring, and restarting the process.
		Write 0	Writing zero has no effect
		Write 1	Writing one starts resetting the peripheral group
		Read 0	Reading zero means that peripheral group is out of reset
		Read 1	Reading one means the peripheral group is being held in reset and should not be accessed
1	EMIF_RST		EMIF software reset bit. Drives the EMIF reset signal.
		Write 0	Writing zero has no effect
		Write 1	Writing one starts resetting the peripheral group
		Read 0	Reading zero means that peripheral group is out of reset
		Read 1	Reading one means the peripheral group is being held in reset and should not be accessed
0	I2C_RST		I2C software reset bit. Drives the I2C reset signal.
		Write 0	Writing zero has no effect
		Write 1	Writing one starts resetting the peripheral group
		Read 0	Reading zero means that peripheral group is out of reset
		Read 1	Reading one means the peripheral group is being held in reset and should not be accessed

1.7.6 EMIF and USB Byte Access

The C55x CPU architecture cannot generate 8-bit accesses to its data or I/O space. But in some cases specific to the USB and EMIF peripherals, it is necessary to access a single byte of data. For example, when writing byte commands to NAND Flash devices or when enabling the EMIF SDRAM self-refresh mode (to enable this mode you must only write to the upper byte in the SDRAM configuration register 2 to avoid triggering the SDRAM auto-initialization procedure).

For these situations, the upper or lower byte of a CPU word access can be masked using the BYTEMODE bits of the EMIF system control register (ESCR) and the USB system control register (USBSCR). The BYTEMODE bits of ESCR only affect accesses to the external memory and the EMIF registers. The BYTEMODE bits of USBSCR only affect CPU accesses to the USB registers. Table 1-76 and Table 1-77 summarize the effect of the BYTEMODE bits for different CPU operations.

NOTE: The BYTEMODE bits of the EMIF system control register should only be used for controlling CPU accesses to NAND Flash devices and EMIF registers.

Table 1-76. Effect of BYTEMODE Bits on EMIF Accesses

BYTEMODE Setting	CPU Access to EMIF Register	CPU Access To External Memory
BYTEMODE = 00b (16-bit word access)	Entire register contents are accessed	ASIZE = 01b (16-bit data bus): EMIF generates a single 16-bit access to external memory for every CPU word access. ASIZE = 00b (8-bit data bus): EMIF generates two 8-bit accesses to external memory for every CPU word access.
BYTEMODE = 01b (8-bit access with high byte selected)	Only the upper byte of the register is accessed.	ASIZE = 01b (16-bit data bus): EMIF generates a 16-bit access to external memory for every CPU word access; only the high byte of the EMIF data bus is used. ASIZE = 00b (8-bit data bus): EMIF generates a single 8-bit access to external memory for every CPU word access.

Table 1-76. Effect of BYTEMODE Bits on EMIF Accesses (continued)

BYTEMODE Setting	CPU Access to EMIF Register	CPU Access To External Memory
BYTEMODE = 10b (8-bit access with low byte selected)	Only the lower byte of the register is accessed.	ASIZE = 01b (16-bit data bus): EMIF generates a 16-bit access to external memory for every CPU word access; only the low byte of the EMIF data bus is used. ASIZE = 00b (8-bit data bus): EMIF generates a single 8-bit access to external memory for every CPU word access.

The USB system control register (USBSCR) is described in Section 1.5.3.5.2.

Table 1-77. Effect of USBSCR BYTEMODE Bits on USB Access

BYTEMODE Setting	CPU Access to USB Register
BYTEMODE = 00b (16-bit word access)	Entire register contents are accessed
BYTEMODE = 01b (8-bit access with high byte selected)	Only the upper byte of the register is accessed
BYTEMODE = 10b (8-bit access with low byte selected)	Only the lower byte of the register is accessed

1.7.6.1 EMIF System Control Register (ESCR) [1C33h]

The EMIF system control register (ESCR) is shown in Figure 1-62 and described in Table 1-78.

Figure 1-62. EMIF System Control Register (ESCR) [1C33h]

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved						BYTEMODE	
R-0h						R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-78. EMIF System Control Register (ESCR) Field Descriptions

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved.
1-0	BYTEMODE	R/W	0h	EMIF byte mode select bits. These bits control CPU data and program accesses to external memory as well as CPU accesses the EMIF memory-mapped registers. 0x0 = Word accesses by the CPU are allowed. 0x1 = Byte accesses by the CPU are allowed (high byte is selected). 0x2 = Byte accesses by the CPU are allowed (low byte is selected). 0x3 = Reserved.

1.7.7 EMIF Clock Divider Register (ECDR) [1C26h]

The EMIF clock divider register (ECDR) controls the input clock frequency to the EMIF module. When EDIV = 1 (default), the EMIF operates at the same clock rate as the system clock (SYSCLK). When EDIV = 0, the EMIF operates at half the clock rate of the system clock.

This register affects both asynchronous memory mode timing as well as synchronous (mobile SDRAM, SDRAM) mode. But half-rate mode is normally only needed to meet synchronous memory timing. For more information regarding when half-rate mode is required, see the mSDRAM timing sections of the device-specific data sheet.

The EMIF clock divider register (ECDR) is shown in Figure 1-63 and described in Table 1-79.

Figure 1-63. EMIF Clock Divider Register (ECDR) [1C26h]

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved							EDIV
R-0h							R/W-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-79. EMIF Clock Divider Register (ECDR) Field Descriptions

Bit	Field	Type	Reset	Description
15-1	Reserved	R	0h	Reserved.
0	EDIV	R/W	1h	EMIF clock divider select bits. The EMIF module can internally divide its input peripheral clock. When this bit is set to 0, the EMIF operates at half the clock rate of its peripheral clock. When this bit is set to 1 the EMIF operates at the full rate of its peripheral clock. 0x0 = EMIF operates at half the peripheral clock rate. 0x1 = EMIF operates at the same rate as the peripheral clock.

1.7.8 McSPI Functional Clock Divider Register (MSPIFCDR) [1C3Ch]

The system clock is divded by two dividers: a pre-divider that divides by 1, 2 or 4, and another divider that divides from 1, 2, 4–30 to generate the McSPI functional clock.

Figure 1-64. McSPI Functional Clock Divider Register (MSPIFCDR) [1C3Ch]

15	14	13	12	11	10	9	8
OCDIV		RESERVED					
R/W-1h		R-0h					

7	6	5	4	3	2	1	0
RESERVED				FCDIV			
R-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 1-80. McSPI Functional Clock Divider Register (MSPIFCDR) Field Descriptions

Bit	Field	Type	Reset	Description
15-14	OCDIV	R/W	1h	Takes the System Clock and divides it down for the McSPI module functional clock. 0000b = Divide-by-1 0001b = Divide-by-2 0010b = Divide-by-4 0011b = Reserved
13-4	RESERVED	R	0h	Reserved
3-0	FCDIV	R/W	0h	Takes the System Clock and divides it down for the McSPI module functional clock. 0000b = Divide-by-1 0001b = Divide-by-2 0010b = Divide-by-4 0011b = Divide-by-6 0100b = Divide-by-8 0101b = Divide-by-10 0110b = Divide-by-12 0111b = Divide-by-14 1000b = Divide-by-16 1001b = Divide-by-18 1010b = Divide-by-20 1011b = Divide-by-22 1100b = Divide-by-24 1101b = Divide-by-26 1110b = Divide-by-28 1111b = Divide-by-30

1.7.9 UHPI Configuration Register (UHPICR) [1C4Eh]

Figure 1-65. UHPI Configuration Register (UHPICR) [1C4Eh]

15	3	2	1	0
Reserved		BYTEADDR	Reserved	
R-0		R/W-0	R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 1-81. UHPI Configuration Register (UHPICR) Field Descriptions

Bit	Field	Value	Description
15-3	Reserved	0	Reserved.
2	BYTEADDR	0	HPI byte/word address mode select. 0 = Host address corresponds to a word (16-bit) address. 1 = Host address corresponds to a byte (8-bit) address.
1-0	Reserved	0	Reserved. Requires a 0 for proper operation.

1.7.10 BootMode Register (BMR) [1C34h]

Figure 1-66. BootMode Register (BMR) [1C34h]

15	11	10	5	4	0
Reserved		BootMode[5:0]		Reserved	
R-0		R-EM_A[20:15]/GP[26:21]		R-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-82. BootMode Register Field Descriptions

BIT	NAME	DESCRIPTION
15:11	Reserved	Reserved
10:9	BootMode[5:4]	Read-only bits that reflect the latched state of the EM_A[20:19]/GP[26:25] pins on the 10th clock edge after RESET pin goes high.(1) The Bootloader reads this register value to determine the frequency of the clock input to the system clock generator. It needs to know this frequency to appropriately program the system clock generator and other peripheral clock dividers. 00: CLK_SEL = 0: 12 MHz via the on-chip USB oscillator CLK_SEL = 1: 11.2896 MHz via the CLK_IN pin 01: CLK_SEL = 0: 12 MHz via the on-chip USB oscillator CLK_SEL = 1: 12.00 MHz or 12.288 MHz via the CLK_IN pin 10: CLK_SEL = 0: 12 MHz via the on-chip USB oscillator CLK_SEL = 1: 16.8 MHz via the CLK_IN pin 11: CLK_SEL = 0: 12 MHz via the on-chip USB oscillator CLK_SEL = 1: 19.2 MHz via the CLK_IN pin
8:5	BootMode[3:0]	Read-only bits that reflect the latched state of the EM_A[18:15]/GP[24:21] pins on the first clock edge after RESET pin goes high. The Bootloader determines boot mode based on this value. 0000: Boot mode: 16-bit NOR flash data boot, system clock generator is in bypass mode. 0001: Boot mode: 16-bit or 8-bit NAND flash data boot, system clock generator is in bypass mode. 0010: Boot mode: UART 9600 baud boot, system clock generator output = input clock x 3 0011: Boot mode: UART 57600 baud boot, system clock generator output = input clock x 3 0100: Boot mode: UART 115200 baud boot, system clock generator output = input clock x 3 0101: Boot mode: SPI 16-bit or 24-bit address Boot (SPI_CLK < 1 MHz), system clock generator output = input clock x 3 0110: Boot mode: SPI 16-bit or 24-bit address Boot (SPI_CLK < 10 MHz), system clock generator output = input clock x 3 0111: Polling Mode 2: Check for valid boot image from peripherals in the following order: NOR, NAND, SPI, I2C, SD/SDHC/MMC/eMMC Controller 0, McSPI, and UART/USB (infinite retry).(2) 1000: Boot mode: I2C 16-bit address Boot, 400 KHz, system clock generator is in bypass mode. 1001: Boot mode: SD, MMC/eMMC Controller 0 card boot, system clock generator is in bypass mode 1010: Boot mode: SD, MMC/eMMC Controller 1 card boot, system clock generator is in bypass mode 1011: Polling Mode 1: Check for valid boot image from peripherals in the following order: NOR, NAND, SPI, I2C, SD/SDHC/MMC/eMMC Controller 0, SD/SDHC/MMC/eMMC Controller 1, and UART/USB (infinite retry).(2) 1100: Boot mode: UHPI 16-bit multiplexed mode boot, system clock generator output = input clock x 3 1101: Boot mode: McSPI 24-bit address serial flash at 10-MHz mode 1110: Boot mode: McSPI 24-bit address serial flash at 40-MHz mode 1111: Boot mode: USB boot, system clock generator output = input clock x 3
4:0	Reserved	Reserved

(1) The $\overline{\text{RESET}}$ pin is asynchronous to the selected system clock (CLKIN or USB_OSC). It could be 10, 11, or even 12 clock cycles after the rising edge of RESETN due to possible metastability.

(2) If MMCx_CMD is low, the bootloader continues to check for a valid boot image in the card controller. MMCx_CMD must be high or toggle in order to move from the card controller to the next peripheral for a valid boot image.

1.8 System Module Registers

Table 1-83. SYSTEM MODULE REGISTERS

CPU Word Address	Acronym	Register Name	Section
1C00h	EBSR	External Bus Selection Register	Section 1.7.3.1
1C02h	PCGCR1	Peripheral Clock Gating Configuration Register 1	Section 1.5.3.2.1
1C03h	PCGCR2	Peripheral Clock Gating Configuration Register 2	Section 1.5.3.2.1
1C04h	PSRCR	Peripheral Software Reset Counter Register	Section 1.7.5.1
1C05h	PRCR	Peripheral Reset Control Register	Section 1.7.5.2
1C14h	TIAFR	Timer Interrupt Aggregation Flag Register	Section 1.6.3
1C15h	MSIAFR	McSPI Interrupt Aggregation Flag Register	Section 1.6.6
1C16h	OSRCR	Output Slew Rate Control Register	Section 1.7.3.5
1C17h	PUDINHIBR1	Pullup and Pulldown Inhibit Register 1	Section 1.7.3.6
1C18h	PUDINHIBR2	Pullup and Pulldown Inhibit Register 2	Section 1.7.3.6
1C19h	PUDINHIBR3	Pullup and Pulldown Inhibit Register 3	Section 1.7.3.6
1C1Ah	DMA0CESR1	Defines the synchronization events for DMA0 channel 1 and 0.	Section 1.7.4.2.2
1C1Bh	DMA0CESR2	Defines the synchronization events for DMA0 channel 3 and 2.	Section 1.7.4.2.2
1C1Ch	DMA1CESR1	Defines the synchronization events for DMA1 channel 1 and 0.	Section 1.7.4.2.2
1C1Dh	DMA1CESR2	Defines the synchronization events for DMA1 channel 3 and 2.	Section 1.7.4.2.2
1C1Eh	CCR1	Clock Configuration Register 1	Section 1.4.4.5
1C1Fh	CCR2	Clock Configuration Register 2	Section 1.4.4.6
1C20h	PMR	PLL Multiplier Register	Section 1.4.4.1
1C21h	PICR	PLL Input Control Register	Section 1.4.4.2
1C22h	PCR	PLL Control Register	Section 1.4.4.3
1C23h	PODCR	PLL Output Divider Control Register	Section 1.4.4.4
1C24h	CLKOUTCR	CLKOUT Configuration Register	Section 1.4.2.3
1C26h	ECDR	EMIF Clock Divider Register	Section 1.7.7
1C27h	RSCR	RTC System Control Register	Section 1.5.4.3.1
1C28h	RAMSLPMDCNTLR1	RAM Sleep Mode Control Register 1	Section 1.5.4.4.1
1C2Ah	RAMSLPMDCNTLR2	RAM Sleep Mode Control Register 2	Section 1.5.4.4.1
1C2Bh	RAMSLPMDCNTLR3	RAM Sleep Mode Control Register 3	Section 1.5.4.4.1
1C2Ch	RAMSLPMDCNTLR4	RAM Sleep Mode Control Register 4	Section 1.5.4.4.1
1C2Dh	RAMSLPMDCNTLR5	RAM Sleep Mode Control Register 5	Section 1.5.4.4.1
1C30h	DMAIFR	Use this register to see which DMA channels have generated interrupts.	Section 1.7.4.2.1
1C31h	DMAIER	Use this register to enable DMA controller channel interrupts.	Section 1.7.4.2.1
1C32h	USBSCR	USB System Control Register	Section 1.5.3.5.2
1C33h	ESCR	EMIF System Control Register	Section 1.7.6.1
1C34h	BMR	Boot Mode Register	Section 1.7.10
1C36h	DMA2CESR1	Defines the synchronization events for DMA2 channel 1 and 0.	Section 1.7.4.2.2
1C37h	DMA2CESR2	Defines the synchronization events for DMA2 channel 3 and 2.	Section 1.7.4.2.2
1C38h	DMA3CESR1	Defines the synchronization events for DMA3 channel 1 and 0.	Section 1.7.4.2.2
1C39h	DMA3CESR2	Defines the synchronization events for DMA3 channel 3 and 2.	Section 1.7.4.2.2

Table 1-83. SYSTEM MODULE REGISTERS (continued)

CPU Word Address	Acronym	Register Name	Section
1C3Ah	CLKSTOP1	Peripheral Clock Stop Request/Acknowledge Register1	Section 1.5.3.2.2
1C3Bh	CLKSTOP2	Peripheral Clock Stop Request/Acknowledge Register2	Section 1.5.3.2.2
1C3Ch	MSPIFCDR	Divides the system clock to obtain the McSPI module functional clocks	Section 1.7.8
1C3Dh	MSIAER	McSPI Interrupt Aggregation Enable Register	Section 1.6.7
1C3Eh	TISR	Timer Interrupt Selection Register	Section 1.6.3.1
1C40h	DIEIDR0	Die ID Register 0	Section 1.7.2.1
1C41h	DIEIDR1	Die ID Register 1	Section 1.7.2.2
1C42h	DIEIDR2	Die ID Register 2	Section 1.7.2.3
1C43h	DIEIDR3	Die ID Register 3	Section 1.7.2.4
1C44h	DIEIDR4	Die ID Register 4	Section 1.7.2.5
1C45h	DIEIDR5	Die ID Register 5	Section 1.7.2.6
1C46h	DIEIDR6	Die ID Register 6	Section 1.7.2.7
1C47h	DIEIDR7	Die ID Register 7	Section 1.7.2.8
1C4Ch	PUDINHIBR4	Pullup and Pulldown Inhibit Register 4	Section 1.7.3.6
1C4Dh	PUDINHIBR5	Pullup and Pulldown Inhibit Register 5	Section 1.7.3.6
1C4Eh	UHPICR	UHPI Configuration Register	Section 1.7.9
1C4Fh	PUDINHIBR6	Pullup and Pulldown Inhibit Register 6	Section 1.7.3.6
1C50h	PUDINHIBR7	Pullup and Pulldown Inhibit Register 7	Section 1.7.3.6
1C58h	JTAGIDLSW	JTAG ID Code LSW Register	Section 1.7.2.9
1C59h	JTAGIDMSW	JTAG ID Code MSW Register	Section 1.7.2.10

FFT Implementation

This chapter describes how to implement the fast fourier transform.

Topic **Page**

2.1 Introduction

The Fast Fourier Transform (FFT) is an efficient means for computing the Discrete Fourier Transform (DFT). It is one of the most widely used computational elements in Digital Signal Processing (DSP) applications. This DSP is ideally suited for such applications. They include an FFT hardware accelerator (HWAFFT) that is tightly coupled with the CPU, allowing high FFT processing performance at very low power. The following sections describe FFT computation on the DSP and covers the following topics:

- Basics of DFT and FFT
- DSP Overview Including the FFT Accelerator
- HWAFFT Description
- HWAFFT Software Interface
- Simple Example to Illustrate the Use of the FFT Accelerator
- FFT Benchmarks
- Description of Open Source FFT Example Software
- Computation of Large (Greater Than 1024-point) FFTs

Project collateral and source code discussed in this application report can be downloaded from: http://www-s.ti.com/sc/techlit/sprabb6.zip.

2.2 Basics of DFT and FFT

The DFT takes an N-point vector of complex data sampled in time and transforms it to an N-point vector of complex data that represents the input signal in the frequency domain. A discussion on the DFT and FFT is provided as background material before discussing the HWAFFT implementation.

The DFT of the input sequence x(n), n = 0, 1, …, N-1 is defined as

$$X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) W_N^{nk}, \; k = 0 \; to \; N-1 \tag{1}$$

Where W_N, the twiddle factor, is defined as

$$W_N = e^{-j2\pi/N}, \; k = 0 \; to \; N-1 \tag{2}$$

The FFT is a class of efficient DFT implementations that produce results identical to the DFT in far fewer cycles. The Cooley-Tukey algorithm is a widely used FFT algorithm that exploits a divide-and-conquer approach to recursively decompose the DFT computation into smaller and smaller DFT computations until the simplest computation remains. One subset of this algorithm called Radix-2 Decimation-in-Time (DIT) breaks each DFT computation into the combination of two DFTs, one for even-indexed inputs and another for odd-indexed inputs. The decomposition continues until a DFT of just two inputs remains. The 2-point DFT is called a butterfly, and it is the simplest computational kernel of Radix-2 FFT algorithms.

2.2.1 Radix-2 Decimation in Time Equations

The Radix-2 DIT decomposition can be seen by manipulating the DFT equation (Equation 1):

Split x(n) into even and odd indices:

$$X(k) = \frac{1}{N} \sum_{n=0}^{N/2-1} x(2n) W_N^{2nk} + \frac{1}{N} \sum_{n=0}^{N/2-1} x(2n+1) W_N^{(2n+1)k}, \; k = 0 \; to \; N-1 \tag{3}$$

Factor W_N^k from the odd indexed summation:

$$X(k) = \frac{1}{N} \sum_{n=0}^{N/2-1} x(2n) W_N^{2nk} + \frac{W_N^k}{N} \sum_{n=0}^{N/2-1} x(2n+1) W_N^{2nk}, \; k = 0 \; to \; N-1 \tag{4}$$

Only twiddle factors from 0 to N/2 are needed:

$$W_N^{2nk} = (e^{-j2\Pi/N})^{2nk} = (e^{-j2\Pi/(N/2)})^{nk} = W_{N/2}^{nk} \text{ and } W_N^{k+N/2} = -W_N^k, k = 0 \text{ to } N/2 - 1 \quad (5)$$

This results in:

$$X(k) = \frac{1}{N}\sum_{n=0}^{N/2-1} x(2n)W_{N/2}^{nk} + \frac{W_N^k}{N}\sum_{n=0}^{N/2-1} x(2n+1)W_{N/2}^{nk}, \; k = 0 \text{ to } N-1 \quad (6)$$

Define $X_{even}(k)$ and $X_{odd}(k)$ such that:

$$X_{even}(k) = \frac{1}{(N/2)}\sum_{n=0}^{N/2-1} x(2n)W_{N/2}^{nk}, \; k = 0 \text{ to } N-1 \quad (7)$$

and

$$X_{odd}(k) = \frac{1}{(N/2)}\sum_{n=0}^{N/2-1} x(2n+1)W_{N/2}^{nk}, \; k = 0 \text{ to } N-1 \quad (8)$$

Finally, Equation 6 is rewritten as:

$$X(k) = \frac{1}{2}(X_{even}(k) + W_N^k X_{odd}(k)), k = 0 \text{ to } N/2 - 1 \quad (9)$$

and

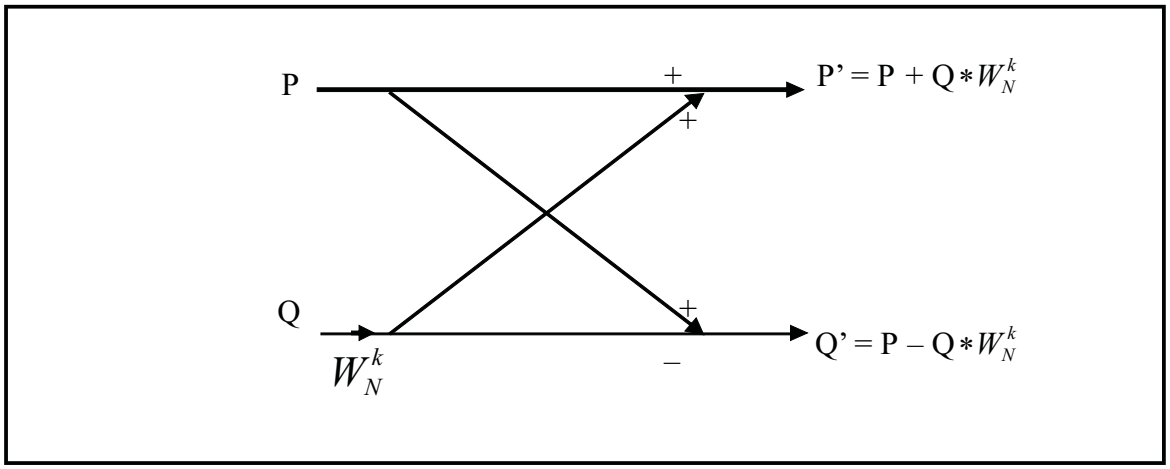
$$X(k+N/2) = \frac{1}{2}(X_{even}(k) - W_N^k X_{odd}(k)), k = 0 \text{ to } N/2 - 1 \quad (10)$$

Equation 9 and Equation 10 show that the N-point DFT can be divided into two smaller N/2-point DFTs. Each smaller DFT is then further divided into smaller DFTs until N = 2. The pair of equations that makeup the 2-point DFT is called the Radix2 DIT Butterfly (see Section 2.2.2). The DIT Butterfly is the core calculation of the FFT and consists of just one complex multiplication and two complex additions.

2.2.2 Radix-2 DIT Butterfly

The Radix-2 Butterfly is illustrated in Figure 2-1. In each butterfly structure, two complex inputs P and Q are operated upon and become complex outputs P' and Q'. Complex multiplication is performed on Q and the twiddle factor, then the product is added to and subtracted from input P to form outputs P' and Q'. The exponent of the twiddle factor W_N^k is dependent on the stage and group of its butterfly. The butterfly is usually represented by its flow graph (Figure 2-1), which looks like a butterfly.

Figure 2-1. DIT Radix 2 Butterfly

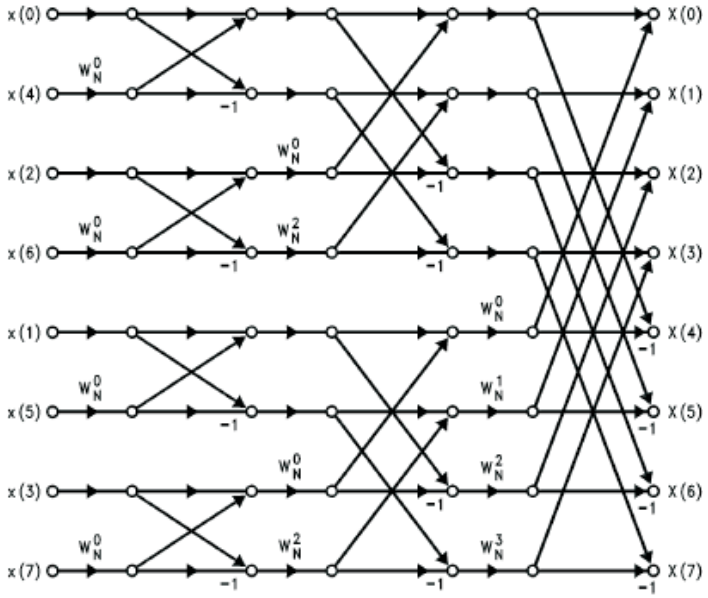


The mathematical meaning of this butterfly is shown below with separate equations for real and imaginary parts:

Complex	Real Part	Imaginary Part
P' = P + Q * W	Pr' = Pr + (Qr * Wr - Qi * Wi)	Pi' = Pi + (Qr * Wi + Qi * Wr)
Q' = P - Q * W	Qr' = Pr - (Qr * Wr - Qi * Wi)	Qi' = Pi - (Qr * Wi + Qi * Wr)

The flow graph in Figure 2-2 shows the interconnected butterflies of an 8-point Radix-2 DIT FFT. Notice that the inputs to the FFT are indexed in bit-reversed order (0, 4, 2, 6, 1, 5, 3, 7) and the outputs are indexed in sequential order (0, 1, 2, 3, 4, 5, 6, 7). Computation of a Radix-2 DIT FFT requires the input vector to be in bit-reversed order, and generates an output vector in sequential order. This bit-reversal is further explained in Section 2.5.3, *Bit-Reverse Function*.

Figure 2-2. DIT Radix 2 8-point FFT



2.2.3 Computational Complexity

The Radix-2 DIT FFT requires $\log_2(N)$ stages, $N/2 * \log_2(N)$ complex multiplications, and $N * \log_2(N)$ complex additions. In contrast, the direct computation of X(k) from the DFT equation (Equation 1) requires N^2 complex multiplications and $(N^2 – N)$ complex additions. Table 2-1 compares the computational complexity for direct DFT versus Radix-2 FFT computations for typical FFT lengths.

Table 2-1. Computational Complexity of Direct DFT Computation versus Radix-2 FFT

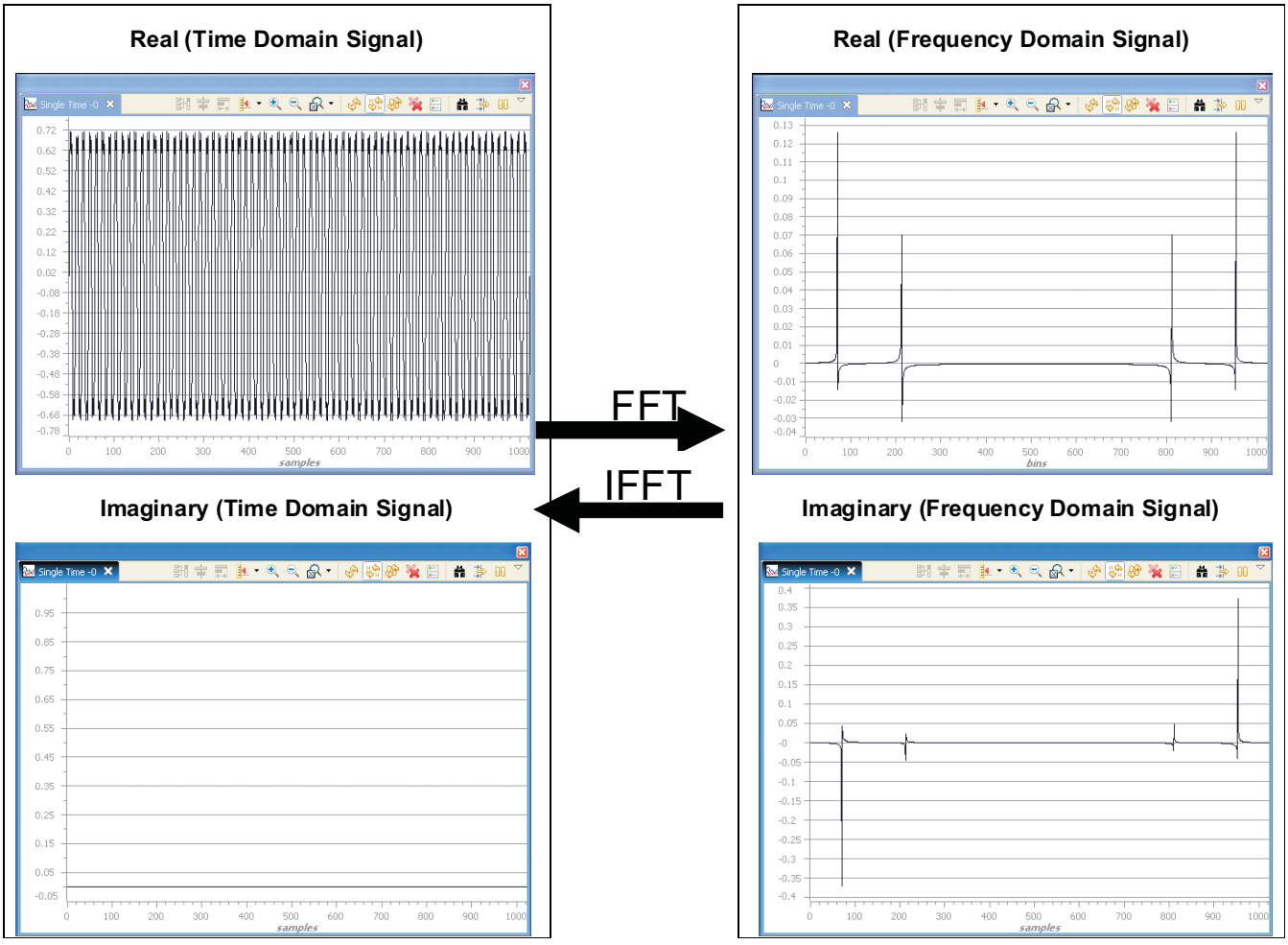
FFT Length	Direct DFT Computation		Radix-2 FFT	
	Complex Multiplications	Complex Additions	Complex Multiplications	Complex Additions
128	16,384	16,256	448	896
256	65,536	65,280	1,024	2,048
512	262,144	264,632	2,304	4,608
1024	1,048,576	1,047,552	5,120	10,240

Table 2-1 clearly shows significant reduction in computational complexity with the Radix-2 FFT, especially for large N. This substantial decrease in computational complexity of the FFT has allowed DSPs to efficiently compute the DFT in reasonable time. For its substantial efficiency improvement over direct computation, the HWAFFT coprocessor in the DSP implements the Radix-2 FFT algorithm.

2.2.4 FFT Graphs

Figure 2-3 is a graphical example of the FFT computation. These results were obtained by using the HWAFFT coprocessor on the DSP. On the left half is the complex time domain signal (real part on top, imaginary part on bottom). On the right half is the complex frequency domain signal produced by the FFT computation (real part on top, imaginary part on bottom). In this example, two sinusoidal tones are present in the time domain. The time domain signal is 1024 points and contains only real data (the imaginary part is all zeros). The two sinusoids are represented in the frequency domain as impulses located at the frequency bins that correspond to the two sinusoidal frequencies. The frequency domain signal is also 1024 points and contains both real parts (top right) and imaginary parts (bottom right).

Figure 2-3. Graphical FFT Computation



2.3 DSP Overview Including the FFT Accelerator

This DSP is a member of TI's TMS320C5000™ fixed-point DSP product family and is designed for low-power applications. These DSPs are optimized for applications characterized by sophisticated processing and portable form factors that require low power and longer battery life. Examples of such applications include portable voice/audio devices, noise cancellation headphones, software-defined radio, musical instruments, medical monitoring devices, wireless microphones, biometrics, industrial instruments, telephony, and audio cards.

Figure 2-4 shows an overview of the DSP consisting of the following primary components:

- Dual MAC, C55x v3.x CPU
  1.05 V @ 75 or 100 MHz, 1.3 V @ 175 or 200 MHz, 1.4 V @ 200 or 225 MHz
- On-Chip memory: 320 KB RAM (64 KB DARAM, 256 KB SARAM), 128 KB ROM
- HWAFFT that supports 8- to 1024-point (powers of 2) real and complex-valued FFTs
- Four DMA controllers and external memory interface
- Power management module
- A set of I/O peripherals that includes I^2C, I^2S, SPI, UART, Timers, EMIF, MMC/SD, GPIO, 10-bit SAR, McBSP, McSPI, UHPI, and USB 2.0
- Three on-chip LDO regulators
- SDRAM and mSDRAM support

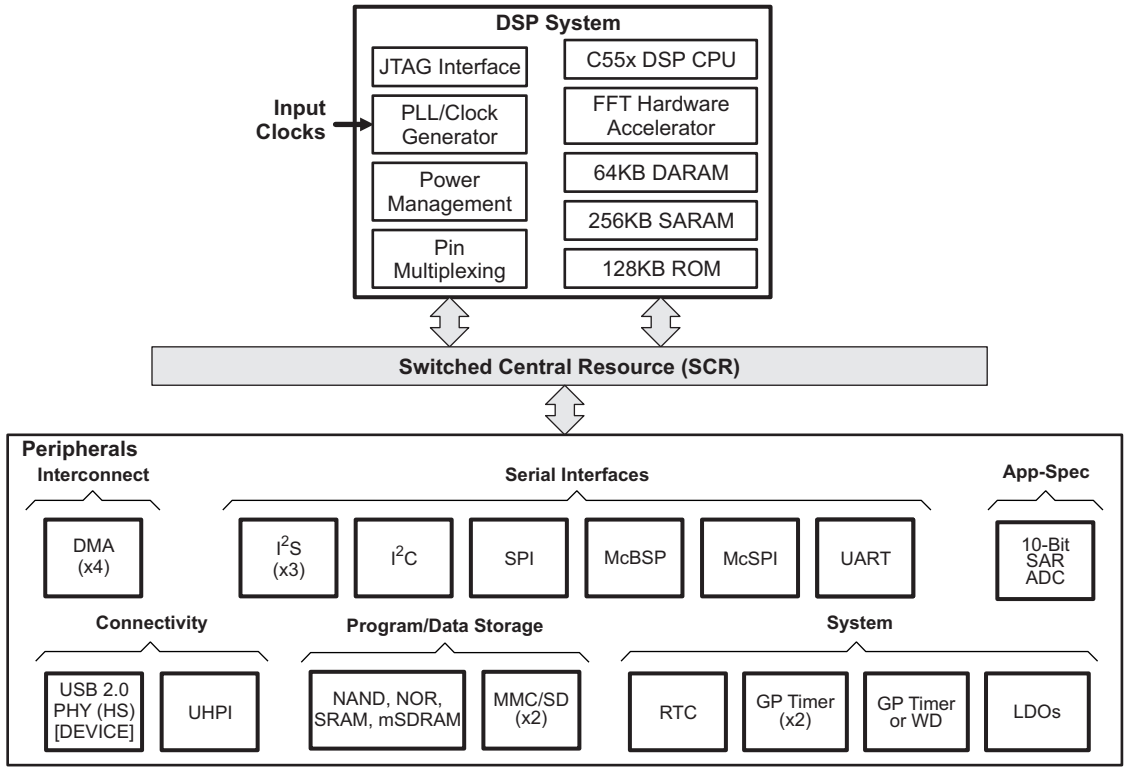


Figure 2-4. Block Diagram

The C55x CPU includes a tightly coupled FFT accelerator that communicates with the C55x CPU through the use of the coprocessor instructions. The main features of this hardware accelerator are:

- Supports 8- to 1024-point (powers of 2) complex-valued FFTs.
- Internal twiddle factor generation for optimal use of memory bandwidth and more efficient programming.
- Basic and software-driven auto-scaling feature provides good precision versus cycle count trade-off.
- Single-stage and double-stage modes enable computation of one or two stages in one pass, and thus better handle the odd power of two FFT widths.
- Is 4 to 6 times more energy efficient and 2.2 to 3.8 times faster than the FFT computations on the CPU.

2.4 FFT Hardware Accelerator Description

The HWAFFT in the DSP is a tightly-coupled, software-controlled coprocessor designed to perform FFT and inverse FFT (IFFT) computations on complex data vectors ranging in length from 8 to 1024 points (powers of 2). It implements a Radix-2 DIT structure that returns the FFT or IFFT result in bit-reversed order.

2.4.1 Tightly-Coupled Hardware Accelerator

The HWAFFT is tightly-coupled with the DSP core which means that it is physically located outside of the DSP core but has access to the core's full memory read bandwidth (busses B, C, and D), access to the core's internal registers and accumulators, and access to its address generation units. The HWAFFT cannot access the data write busses or memory mapped registers (MMRs). Because the HWAFFT is seen as part of the execution unit of the CPU, it must also comply to the core's pipeline exceptions, and in particular those caused by stalls and conditional execution.

2.4.2 Hardware Butterfly, Double-Stage and Single-Stage Mode

The core of the HWAFFT consists of a single Radix-2 DIT Butterfly implemented in hardware. This hardware supports a double-stage mode where two FFT stages are computed a single pass. In this mode the HWAFFT feeds the results from the first stage back into the hardware butterfly to compute the second stage results in a single pass. This double-stage mode offers significant speed-up especially for large FFT lengths. However, when the number of required stages is odd (FFT lengths = 8, 32, 128, or 512 points) the final stage needs to be computed alone and, consequently, at a lower acceleration rate. For this reason a single-stage mode is also provided.

The HWAFFT supports two stage modes:

- Double-Stage Mode – two FFT stages performed in each pass
- Single-Stage Mode – one FFT stage performed in each pass

2.4.3 Pipeline and Latency

The logic of the HWAFFT is pipelined to deliver maximum throughput. Complex multiplication with the twiddle factors is performed in the first pipeline stage, and complex addition and subtraction is performed in the second pipeline stage. Valid results appear some cycles of latency after the first data is read from memory:

- 5 cycles of latency in single-stage mode
- 9 cycles of latency in double-stage mode

There are three states to consider during computation of a single or double stage:

- Prologue: The hardware accelerator is fed with one complex input at a time but does not output any valid data.
- Kernel: Valid outputs appear while new inputs are received and computed upon.
- Epilogue: A few more cycles are needed to flush the pipeline and output the last butterfly results.

Consecutive stages can be overlapped such that the first data points for the next pass are read while the final output values of the current pass are returned. For odd-power-of-two FFT lengths, the last double-stage pass needs to be completed before starting a final single-stage pass. Thus, the double-stage latency is only experienced once for even-powers-of-2 FFT computations and twice for odd-powers-of-2 FFT computations. Latency has little impact on the total computation performance, and less and less so as the FFT size increases.

2.4.4 Software Control

Software is required to communicate between the CPU and the HWAFFT. The CPU instruction set architecture (ISA) includes a class of coprocessor (copr) instructions that allows the CPU to initialize, pass data to, and execute butterfly computations on the HWAFFT. Computation of an FFT/IFFT is performed by executing a sequence of these copr instructions.

C-callable HWAFFT functions are provided with optimized sequences of copr instructions for each available FFT length. To conserve program memory, these functions are located in the DSP's ROM. A detailed explanation of the HWAFFT software interface and its application is provided in Section 2.5, *HWAFFT Software Interface*.

2.4.5 Twiddle Factors

To conserve memory bus bandwidth, twiddle factors are stored in a look-up-table within the HWAFFT coprocessor. The 512 complex twiddle factors (16-bit real, 16-bit imaginary) are available for computing up to 1024-point FFTs. Smaller FFT lengths use a decimated subset of these twiddle factors. Indexing the twiddle table is pipelined and optimized based on the current FFT stage and group being computed. When the IFFT computation is performed, the complex conjugates of the twiddle factors are used.

2.4.6 Scaling

FFT computation with fixed-point numbers is vulnerable to overflow, underflow, and saturation effects. Depending on the dynamic range of the input data, some scaling may be required to avoid these effects during the FFT computation. This scaling can be done before the FFT computation, by computing the dynamic range of the input points and scaling them down accordingly. If the magnitude of each complex input element is less than 1/N, where N = FFT Length, then the N-point FFT computation will not overflow.

Uniformly dividing the input vector elements by N (Pre-scaling) is equivalent to shifting each binary number right by $\log_2(N)$ bits, which introduces significant error (especially for large FFT lengths). When this error propagates through the FFT flow graph, the output noise-to-signal ratio increases by 1 bit per stage or $\log_2(N)$ bits in total. Overflow will not occur if each input's magnitude is less than 1/N.

Alternatively, a simple divide-by-2 and round scaling after each butterfly offers a good trade-off between precision and overflow protection, while minimizing computation cycles. Because the error introduced by early FFT stages is also scaled after each butterfly, the output noise-to-signal ratio increases by just ½ bit per stage or ½ * $\log_2(N)$ bits in total. Overflow is avoided if each input's magnitude is less than 1.

The HWAFFT supports two scale modes:

- NOSCALE
  - Scaling logic disabled
  - Vulnerable to overflow
  - Output dynamic range grows with each stage
  - No overflow if input magnitudes < 1/N
- SCALE
  - Scales each butterfly output by 1/2
  - No overflow if input magnitudes < 1

2.5 HWAFFT Software Interface

The software interface to the HWAFFT is handled through a set of coprocessor instructions. When decoded by the coprocessor, the coprocessor instructions perform initialization, load and store, and execute operations on the HWAFFT coprocessor. C-callable functions are provided that contain the necessary sequences of coprocessor instructions for performing FFT and IFFT computations in the range of 8 to 1024 points (by powers of 2).

In addition, an optimized out-of-place bit-reversal function is provided to perform the complex vector bit-reversal required by Radix-2 FFT computations. These functions are defined in the hwafft.asm source code file. To conserve on-chip RAM, these functions have been placed in the on-chip ROM of the DSP. See Section 2.5.5, *Project Configuration for Calling Functions from ROM*, for steps to configure your project to call the HWAFFT functions from ROM.

2.5.1 Data Types

The input and output vectors of the HWAFFT contain complex numbers. Each real and imaginary part is represented by a two's complement, 16-bit fixed-point number. The most significant bit holds the number's sign value, and the remaining 15 are fraction bits (S16Q15 format). The range of each number is [-1, 1 – $(1/2)^{15}$]. Real and imaginary parts appear in an interleaved order within each vector:

Int16 CMPLX_Vec16[2*N] = …(N = FFT Length)

Real[0]	Imag[0]	Real[1]	Imag[1]	Real[2]	Imag[2]
Bit15,.....,Bit0	Bit15,.....,Bit0	Bit15,.....,Bit0	Bit15,.....,Bit0	Bit15,.....,Bit0	Bit15,.....,Bit0

The HWAFFT functions use an Int32 pointer to reference these complex vectors. Therefore, each 32-bit element contains the 16-bit real part in the most significant 16 bits, and the 16-bit imaginary part in the least significant 16 bits.

Int32 CMPLX_Vec32[N] = …(N = FFT Length)

Real[0] Imag[0]	Real[1] Imag[1]	Real[2] Imag[2]
Bit31,....., Bit16, Bit15,....., Bit0	Bit31,....., Bit16, Bit15,....., Bit0	Bit31,....., Bit16, Bit15,....., Bit0

To extract the real and imaginary parts from the complex vector, it is necessary to mask and shift each 32-bit element into its 16-bit real and imaginary parts:

```
Uint16 Real_Part = CMPLX_Vec32[i] >> 16;
Uint16 Imaginary_Part = CMPLX_Vec32[i] & 0x0000FFFF;
```

2.5.2 HWAFFT Functions

C-Callable HWAFFT Functions are provided for computing FFT/IFFT transforms on the HWAFFT coprocessor. These functions contain optimized sequences of coprocessor instructions for computing scaled or unscaled 8-, 16-, 32-, 64-, 128-, 256-, 512-, and 1024-point FFT/IFFTs. Additionally, an optimized out-of-place bit-reversal function is provided to bit-reverse the input vector before supplying it to the HWAFFT. Computation of a Radix-2 DIT FFT requires the input vector to be in bit-reversed order, and generates an output vector in sequential order.

2.5.2.1 HWAFFT Naming and Format

NOTE: To execute the HWAFFT routines from the ROM of the DSP, the programmer must satisfy memory allocation restrictions for the data and scratch buffers. For an explanation of the restrictions and workarounds, see the device-specific errata:

- *TMS320C5517 Fixed-Point DSP Silicon Errata* (literature number SPRZ383)

The HWAFFT functions are named hwafft_Npts, where N is the FFT length. For example, hwafft_32pts is the name of the function for performing 32-point FFT and IFFT operations. The structure of the HWAFFT functions is:

`Uint16 hwafft_Npts(`	Performs N-point complex FFT/IFFT, where N = {8, 16, 32, 64, 128, 256, 512, 1024}
`Int32 *data,`	Input/output – complex vector
`Int32 *scratch,`	Intermediate/output – complex vector
`Uint16 fft_flag,`	Flag determines whether FFT or IFFT performed, (0 = FFT, 1 = IFFT)
`Uint16 scale_flag`	Flag determines whether butterfly output divided by 2 (0 = Scale, 1 = No Scale)
`); Return value`	Flag determines whether output in data or scratch vector (0 = data, 1 = scratch)

2.5.2.2 HWAFFT Parameters

The following describe the parameters for the HWAFFT functions.

Int32 *data

This is the input vector to the HWAFFT. It contains complex data elements (real part in most significant 16 bits, imaginary part in least significant 16 bits). After the HWAFFT function completes, the result will either be stored in this data vector or in the scratch vector, depending on the status of the return value. The return value is Boolean where 0 indicates that the result is stored in this data vector, and 1 indicates the scratch vector. The data and scratch vectors must reside in separate blocks of RAM (DARAM or SARAM) to maximize memory bandwidth.

```
#pragma DATA_SECTION(data_buf, "data_buf");
     //Static Allocation to Section: "data_buf  : > DARAM" in Linker CMD File
Int32 data_buf[N = FFT Length];
Int32 *data = data_buf;
Int32 *data:
```

The *data parameter is a complex input vector to HWAFFT. It contains the output vector if Return Value = 0 = OUT_SEL_DATA. There is a strict address alignment requirement if *data is shared with a bit-reverse destination vector (**recommended**). See Section 2.5.3.1, *Bit Reverse Destination Vector Alignment Requirement*.

Int32 *scratch

This is the scratch vector used by the HWAFFT to store intermediate results between FFT stages. It contains complex data elements (real part in most significant 16 bits, imaginary part in least significant 16 bits). After the HWAFFT function completes the result will either be stored in the data vector or in this scratch vector, depending on the status of the return value. The return value is Boolean, where 0 indicates that the result is stored in the data vector, and 1 indicates this scratch vector. The data and scratch vectors must reside in separate blocks of RAM (DARAM or SARAM) to maximize memory bandwidth.

```
#pragma DATA_SECTION(scratch_buf, "scratch_buf");
     //Static Allocation to Section: "scratch_buf  : > DARAM" in Linker CMD File
Int32 scratch_buf[N = FFT Length];
Int32 *scratch = scratch_buf;
Int32 *scratch:
```

The *scratch parameter is a complex scratchpad vector to HWAFFT. It contains the output vector if Return Value = 1 = OUT_SEL_SCRATCH.

Uint16 fft_flag

The FFT/IFFT selection is controlled by setting the fft_flag to 0 for FFT and 1 for Inverse FFT.

```
#define FFT_FLAG    ( 0 )   /* HWAFFT to perform FFT  */
#define IFFT_FLAG   ( 1 )   /* HWAFFT to perform IFFT */
Uint16 fft_flag:
```

fft_flag = FFT_FLAG:	FFT Performed
fft_flag = IFFT_FLAG:	Inverse FFT Performed

Uint16 scale_flag

The automatic scaling (divide each butterfly output by 2) feature is controlled by setting the scale_flag to 0 to enable scaling and 1 to disable scaling.

```
#define SCALE_FLAG       ( 0 )   /* HWAFFT to scale butterfly output     */
#define NOSCALE_FLAG     ( 1 )   /* HWAFFT not to scale butterfly output */
Uint16 scale_flag:
```

scale_flag = SCALE_FLAG:	Divide by 2 scaling is performed at the output of each FFT Butterfly.
scale_flag = NOSCALE_FLAG:	No scaling is performed, overflow may occur if the input dynamic is too high.

Uint16 <Return Value>:

This is the Uint16 return value of the HWAFFT functions. After the HWAFFT function completes, the result will either be stored in the data vector or in the scratch vector, depending on the status of this return value. The return value is Boolean where 0 indicates that the result is stored in the data vector, and 1 indicates this scratch vector. The program must check the status of the Return Value to determine the location of the FFT/IFFT result.

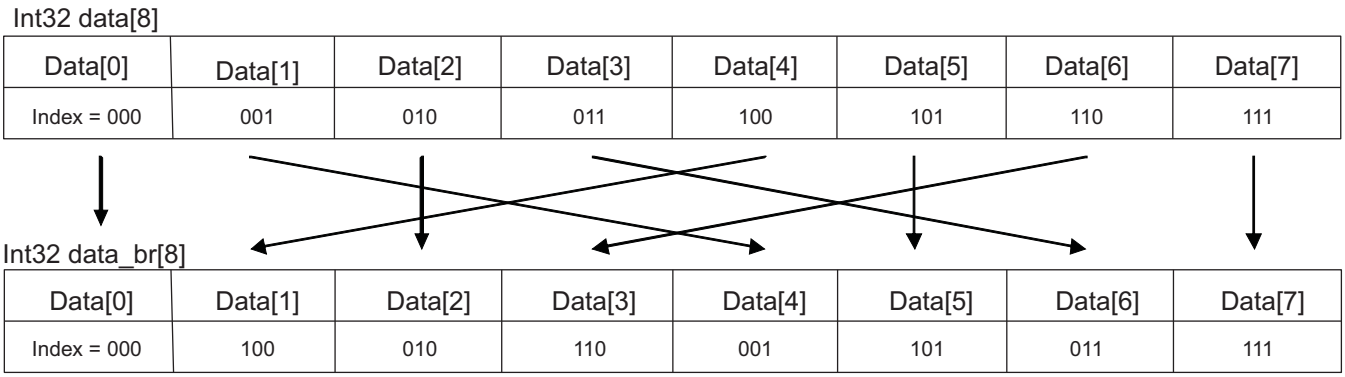
```
#define OUT_SEL_DATA    ( 0 )   /* indicates HWAFFT output located in input data vector   */
#define OUT_SEL_SCRATCH ( 1 )   /* indicates HWAFFT output located in scratch vector */
Uint16 <Return Value>:
```

Return Value = OUT_SEL_DATA:	FFT/IFFT result located in the data vector
Return Value = OUT_SEL_SCRATCH:	FFT/IFFT result located in the scratch vector

2.5.3 Bit Reverse Function

Before computing the FFT/IFFT on the HWAFFT, the input buffer must be bit-reversed to facilitate a Radix-2 DIT computation. This function contains optimized assembly that executes on the CPU to rearrange the Int32 elements of the input vector by placing each element in the destination vector at the index that corresponds to the bit-reversal of its current index. For example, in an 8-element vector, the index of the third element is 011 in binary, then the bit-reversed index is 110 in binary or 6 in decimal, so the third element of the input vector is copied to the sixth element of the bit-reversal destination vector.

Figure 2-5. Bit Reversed Input Buffer



2.5.3.1 Bit Reverse Destination Vector Alignment Requirement

Strict requirements are placed on the address of the bit-reversal destination buffer. This buffer must be aligned in RAM such that $\log_2(4 * N)$ zeros appear in the least significant bits of the byte address (8 bits), where N is the FFT Length. For example, a 1024-point FFT needs to bit-reverse 1024 complex array elements (32-bit elements). The address for the bit-reversed buffer needs to have 12 zeros in the least significant bits of its byte address ($\log_2(4 * 1024) = 12$). Since the word address (16 bits) is the byte address shifted right one bit, the word address requires 11 zeros in the least significant bits. This bit-reverse is considered out-of-place because the inputs and outputs are stored in different vectors. In-place bit-reversal is not supported by this function. There are no alignment requirements for the bit-reverse source vector.

2.5.3.2 Bit Reverse Format and Parameters

The structure of the HWAFFT functions is:

```
void                Performs out-of-place bit-reversal on 32-bit data vector
hwafft_br(
    Int32 *data,    Input – 32-bit data vector
    Int32           Output – bit-reversed data vector
*data_br,
    Uint16          Length of complex data vector
data_len,
);
```

The parameters for the hwafft_br function are:

Int32 *data

This is the input vector to the bit reverse function. It contains complex data elements (real part in most significant 16 bits, imaginary part in least significant 16 bits). There are no specific alignment requirements for this vector.

```
#pragma DATA_SECTION(data_buf, "data_buf");
     //Static Allocation to Section: "data_buf  : > DARAM" in Linker CMD File
Int32 data_buf[N = FFT Length];
Int32 *data = data_buf;
```

Int32 *data_br

This is the destination vector of the bit-reverse function. It contains complex data elements (real part in most significant 16 bits, imaginary part in least significant 16 bits). A strict alignment requirement is placed on this destination vector of the bit-reverse function: This buffer must be aligned in RAM such that $\log_2(4 * N)$ zeros appear in the least significant bits of the byte address (8 bits), where N is the FFT Length. See Section 2.9, *Appendix A Methods for Aligning the Bit-Reverse Destination Vector*, for ways to force the linker to enforce this alignment requirement.

```
#define ALIGNMENT 2*N  // ALIGNS data_br_buf to an address with log2(4*N) zeros in the
                       // least significant bits of the byte address
#pragma DATA_SECTION(data_br _buf, "data_br_buf");
                       // Allocation to Section: "data_br _buf : > DARAM" in Linker CMD File
#pragma DATA_ALIGN (data_br_buf, ALIGNMENT);
Int32 data_br_buf[N = FFT Length];
Int32 * data_br = data_br_buf;
Int32 *data_br:
```

Strict address alignment requirement: This buffer must be aligned in RAM such that ($\log_2$(4 * N) zeros appear in the least significant bits of the byte address (8 bits), where N is the FFT Length. See Section 2.9 for ways to force the linker to enforce this alignment requirement.

Uint16 *data_len

This Uint16 parameter indicates the length of the data and data_br vectors.

```
Uint16 data_len:
```

The data_len parameter indicates the length of the Int32 vector (FFT Length). Valid lengths include powers of two: {8, 16, 32, 64, 128, 256, 512, 1024}.

2.5.4 Function Descriptions and ROM Locations

Table 2-2 shows the available HWAFFT routines with descriptions and respective addresses in ROM.

Table 2-2. Available HWAFFT Routines

Address	Name	Description	Calling Convention
0x00fefefc	hwafft br	Int32 (32-bit) vector bit-reversal, Strict alignment requirement	void hwafft_br(Int32 *data, Int32 *data_br, Uint16 data_len);
0x00feff10	hwafft 8pts	8-point FFT/IFFT, 1 double-stage, 1 single-stage	Uint16 hwafft_8pts(Int32 *data,Int32 *scratch, Uint16 fft_flag, Uint16 scale_flag);
0x00feffff	hwafft 16pts	16-point FFT/IFFT, 2 double-stages, 0 single-stages	Uint16 hwafft_16pts(Int32 *data,Int32 *scratch, Uint16 fft_flag, Uint16 scale_flag);
0x00ff0155	hwafft 32pts	32-point FFT/IFFT, 2 double-stages, 1 single-stage	Uint16 hwafft_32pts(Int32 *data,Int32 *scratch, Uint16 fft_flag, Uint16 scale_flag);
0x00ff045e	hwafft 64pts	64-point FFT/IFFT, 3 double-stages, 0 single-stages	Uint16 hwafft_64pts(Int32 *data,Int32 *scratch, Uint16 fft_flag, Uint16 scale_flag);
0x00ff05f3	hwafft 128pts	128-point FFT/IFFT, 3 double-stages, 1 single-stage	Uint16 hwafft_128pts(Int32 *data,Int32 *scratch, Uint16 fft_flag, Uint16 scale_flag);
0x00ff0804	hwafft 256pts	256-point FFT/IFFT, 4 double-stages, 0 single-stages	Uint16 hwafft_256pts(Int32 *data,Int32 *scratch, Uint16 fft_flag, Uint16 scale_flag);
0x00ff0a02	hwafft 512pts	512-point FFT/IFFT, 4 double-stages, 1 single-stage	Uint16 hwafft_512pts(Int32 *data,Int32 *scratch, Uint16 fft_flag, Uint16 scale_flag);
0x00ff0c7c	hwafft 1024pts	1024-point FFT/IFFT, 5 double-stages, 0 single-stages	Uint16 hwafft_1024pts(Int32 *data,Int32 *scratch, Uint16 fft_flag, Uint16 scale_flag);

2.5.5 Project Configuration for Calling Functions from ROM

The HWAFFT functions occupy approximately 4K Bytes of memory, so to conserve RAM they have been placed in the DSP's 128K Bytes of on-chip ROM. These functions have the same names as the functions stored in hwafft.asm, but the functions in the on-chip ROM do not consume any RAM.

NOTE: To execute the HWAFFT routines from the ROM of the DSP, the programmer must satisfy memory allocation restrictions for the data and scratch buffers. For an explanation of the restrictions and workarounds, see the device-specific errata:

- *TMS320C5517 Fixed-Point DSP Silicon Errata* (literature number SPRZ383)

In order to utilize these HWAFFT routines in ROM, add the following lines to the bottom of the project's linker CMD file and remove the hwafft.asm file from the project (or exclude it from the build). When the project is rebuilt, the HWAFFT functions will reference the ROM locations. The HWAFFT ROM locations for the DSP are shown in Table 2-2.

```
/*** Add the following code to the linker command file to call HWAFFT Routines from ROM ***/

/* HWAFFT Routines ROM Addresses */
_hwafft_br        = 0x00fefefc;
_hwafft_8pts    = 0x00feff10;
_hwafft_16pts    = 0x00feffff;
_hwafft_32pts    = 0x00ff0155;
_hwafft_64pts    = 0x00ff045e;
_hwafft_128pts    = 0x00ff05f3;
_hwafft_256pts    = 0x00ff0804;
_hwafft_512pts    = 0x00ff0a02;
_hwafft_1024pts    = 0x00ff0c7c;
```

2.6 Simple Example to Illustrate the Use of the FFT Accelerator

NOTE: To execute the HWAFFT routines from the ROM of the DSP, the programmer must satisfy memory allocation restrictions for the data and scratch buffers. For an explanation of the restrictions and workarounds, see the device-specific errata:

- *TMS320C5517 Fixed-Point DSP Silicon Errata* (literature number SPRZ383)

The source code below demonstrates typical use of the HWAFFT for the 1024-point FFT and IFFT cases.

The HWAFFT Functions make use of Boolean flag variables to select between FFT and IFFT, Scale and No Scale mode, and Data and Scratch output locations.

```
#define FFT_FLAG        ( 0 )   /* HWAFFT to perform FFT */
#define IFFT_FLAG       ( 1 )   /* HWAFFT to perform IFFT */
#define SCALE_FLAG      ( 0 )   /* HWAFFT to scale butterfly output */
#define NOSCALE_FLAG    ( 1 )   /* HWAFFT not to scale butterfly output */
#define OUT_SEL_DATA    ( 0 )   /* Indicates HWAFFT output located in input data vector */
#define OUT_SEL_SCRATCH ( 1 )   /* Indicates HWAFFT output located in scratch vector */
Int32 *data;
Int32 *data_br;
Uint16 fft_flag;
Uint16 scale_flag;
Int32 *scratch;
Uint16 out_sel;
Int32 *result;
```

2.6.1 1024-Point FFT, Scaling Disabled

Compute 1024-point FFT with Scaling enabled: a ½ scale factor after every stage:

```
fft_flag = FFT_FLAG;
scale_flag = SCALE_FLAG;

data = <1024-point Complex input>;

/* Bit-Reverse 1024-point data, Store into data_br, data_br aligned to
   12-least significant binary zeros*/
hwafft_br(data, data_br, DATA_LEN_1024);  /* bit-reverse input data,
                                             Destination buffer aligned */
data = data_br;

/* Compute 1024-point FFT, scaling enabled. */
out_sel = hwafft_1024pts(data, scratch, fft_flag, scale_flag);

if (out_sel == OUT_SEL_DATA) {
   result = data;
}else {
    result = scratch;
}
```

2.6.2 1024-Point IFFT, Scaling Disabled

Compute 1024-point IFFT with Scaling disabled:

```
fft_flag = IFFT_FLAG;
scale_flag = NOSCALE_FLAG;

data = <1024-point Complex input>;

/* Bit-Reverse 1024-point data, Store into data_br, data_br aligned to
   12-least significant binary zeros */
hwafft_br(data, data_br, DATA_LEN_1024);
data = data_br;

/* Compute 1024-point IFFT, scaling disabled */
out_sel = hwafft_1024pts(data, scratch, fft_flag, scale_flag);

if (out_sel == OUT_SEL_DATA) {
   result = data;
} else {
    result = scratch;
}
```

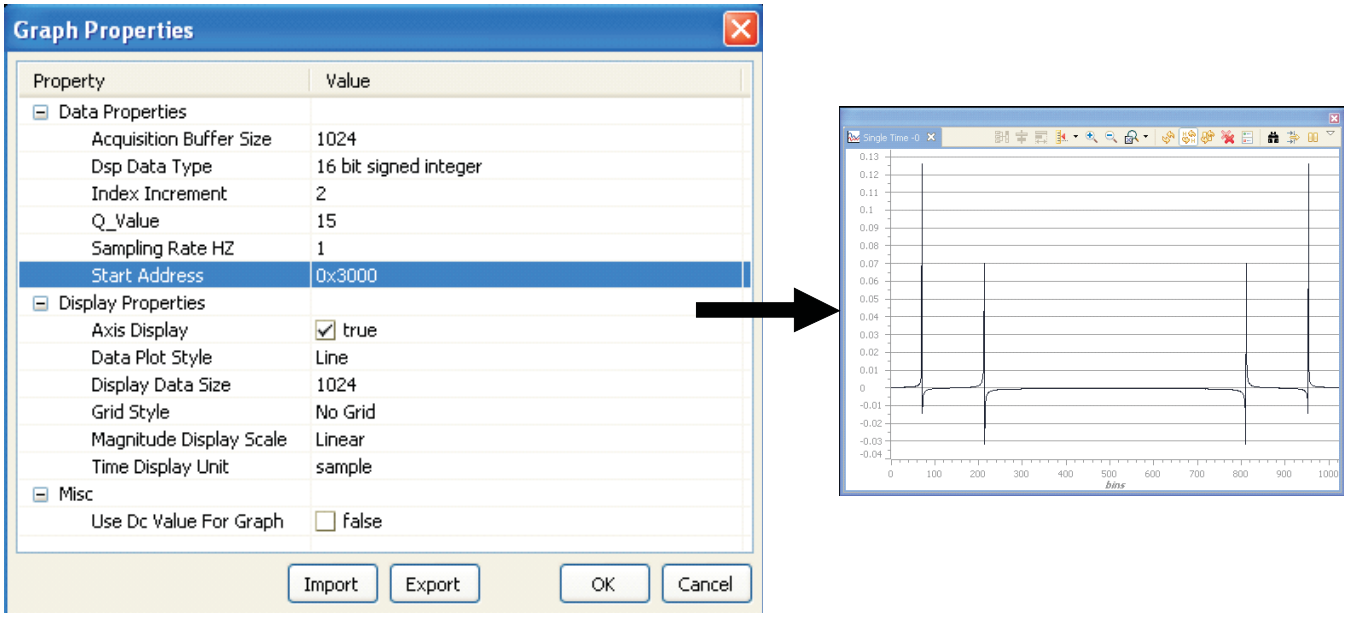
2.6.3 Graphing FFT Results in CCS4

Code Composer includes a graphing utility that makes visualization of the FFT operation quick and easy. The Graph Utility is located in the CCSv4 window, under Tools → Graph → Single Time.

If the FFT Result is stored in scratch (OutSel = 1) and scratch is located at address 0x3000…

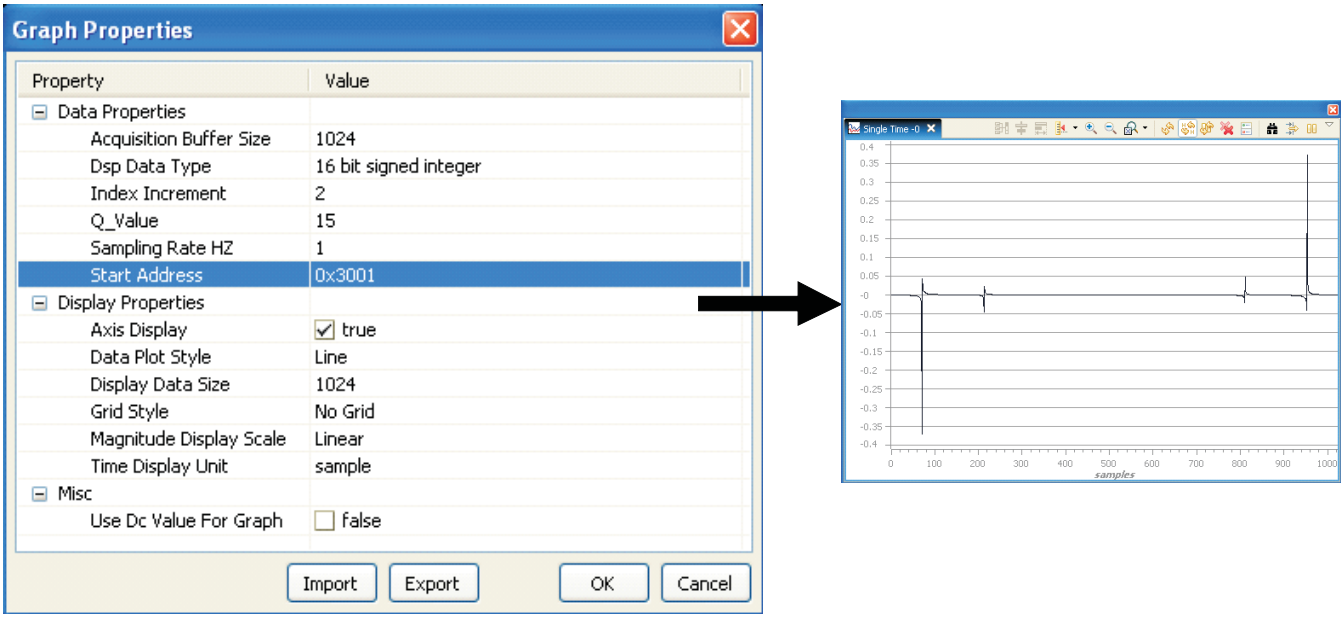
Plot the real part:

Figure 2-6. Graphing the Real Part of the FFT Result in CCS4



Plot the imaginary part:

Figure 2-7. Graphing the Imaginary Part of the FFT Result in CCS4



2.7 FFT Benchmarks

Table 2-3 compares the FFT performance of the HWAFFT versus FFT computation using the CPU under the following conditions:

- Core voltage = 1.05 V
- PLL = 75 or 100 MHz
- Power measurement condition:
  - At room temperature only
  - All peripherals are clock gated
  - Measured at CV_{DD}

Table 2-3. FFT Performance on HWAFFT vs CPU (CV_{DD} = 1.05 V, PLL = 60 MHz)

Complex FFT	FFT with HWA		CPU (Scale)		HWA versus CPU	
	FFT + BR(1) Cycles	Energy/FFT (nJ/FFT)	FFT + BR(1) Cycles	Energy/FFT (nJ/FFT)	x Times Faster (Scale)	x Times Energy Efficient (Scale)
8 pt	92 + 38 = 130	23.6	196 + 95 = 291	95.1	2.2	4
16 pt	115 + 55 = 170	32.1	344 + 117 = 461	157.1	2.7	4.9
32 pt	234 + 87 = 321	69.5	609 + 139 = 748	269.9	2.3	3.9
64 pt	285 + 151 = 436	98.5	1194 + 211 = 1405	531.7	3.2	5.4
128 pt	633 + 279 = 912	219.2	2499 + 299 = 2798	1090.4	3.1	5
256 pt	1133 + 535 = 1668	407.2	5404 + 543 = 5947	2354.2	3.6	5.8
512 pt	2693 + 1047 = 3740	939.7	11829 + 907 = 12736	5097.5	3.4	5.4
1024 pt	5244 + 2071 = 7315	1836.2	25934 + 1783 = 27717	11097.9	3.8	6

(1) BR = Bit Reverse

In summary, Table 2-3 shows that for the test conditions used, HWAFFT is 4 to 6 times more energy efficient and 2.2 to 3.8 times faster than the CPU.

Table 2-4 compares FFT performance of the accelerator versus FFT computation using the CPU under the following conditions:

- Core voltage = 1.3 V
- PLL = 175 or 200 MHz
- Power measurement condition:
  - At room temperature only
  - All peripherals are clock gated
  - Measured at CV_{DD}

Table 2-4. FFT Performance on HWAFFT vs CPU (CV_{DD} = 1.3 V, PLL = 100 MHz)

Complex FFT	FFT with HWA		CPU (Scale)		HWA versus. CPU	
	FFT + BR[1] Cycles	Energy/FFT (nJ/FFT)	FFT + BR[1] Cycles	Energy/FFT (nJ/FFT)	x Times Faster (Scale)	x Times Energy Efficient (Scale)
8 pt	92 + 38 = 130	36.3	196 + 95 = 291	145.9	2.2	4
16 pt	115 + 55 = 170	49.3	344 + 117 = 461	241	2.7	4.9
32 pt	234 + 87 = 321	106.9	609 + 139 = 748	414	2.3	3.9
64 pt	285 + 151 = 436	151.3	1194 + 211 = 1405	815.7	3.2	5.4
128 pt	633 + 279 = 912	336.8	2499 + 299 = 2798	1672.9	3.1	5
256 pt	1133 + 535 = 1668	625.6	5404 + 543 = 5947	3612.9	3.6	5.8
512 pt	2693 + 1047 = 3740	1442.8	11829 + 907 = 12736	7823.8	3.4	5.4
1024 pt	5244 + 2071 = 7315	2820.6	25934 + 1783 = 27717	17032.4	3.8	6

(1) BR = Bit Reverse

In summary, Table 2-4 shows that for the test conditions used, HWAFFT is 4 to 6 times more energy efficient and 2.2 to 3.8 times faster than the CPU.

2.8 Computation of Large (Greater Than 1024-Point) FFTs

The HWAFFT can perform up to 1024-point complex FFTs/IFFTs at a maximum, but if larger FFT sizes (that is, 2048-point) are required, the DSP core can be used to compute extra Radix-2 stages that are too large for the HWAFFT to handle.

Recall the Radix-2 DIT equations:

$$X(k) = \frac{1}{2}(X_{even}(k) + W_N^k X_{odd}(k)), k = 0 \text{ to } N/2 - 1 \quad (11)$$

and

$$X(k + N/2) = \frac{1}{2}(X_{even}(k) - W_N^k X_{odd}(k)), k = 0 \text{ to } N/2 - 1 \quad (12)$$

2.8.1 Procedure for Computing Large FFTs

The procedure for computing an additional Radix-2 DIT stage on the CPU is outlined:

- Split the input signal into even and odd indexed signals, X_{even} and X_{odd}.
- Call N/2 point FFTs for the even and odd indexed inputs.
- Complex Multiply the X_{odd} FFT results with the decimated twiddle factors for that stage.
- Add the X_{odd} * Twiddle product to Xeven to find the first half of the FFT result.
- Subtract X_{odd} * Twiddle to find the second half of the FFT result.

2.8.2 Twiddle Factor Computation

The HWAFFT stores 512 complex twiddle factors enabling FFT/IFFT computations up to 1024 points. Recall Equation 5 states that only twiddle factors from 0 to N/2 are needed. To compute FFT/IFFTs larger than 1024 points, you must supply N/2 complex twiddle factors, where N is the FFT length (powers of 2).

The following MATLAB code creates real and imaginary parts of the twiddle factors for any N:

```
N = 2048;
n = 0:(N/2-1);
twid_r = cos(2*pi*n/N);
twid_i = -sin(2*pi*n/N);
```

2.8.3 Bit-Reverse Separates Even and Odd Indexes

A nice property of the bit-reversal process is the automatic separation of odd-indexed data from even-indexed data. Before the bit-reverse, even indexes have a 0 in the least significant bit and odd indexes have a 1 in the least significant bit. After the bit-reverse, even indexes have a 0 in the most significant bit, and odd indexes have a 1 in the most significant bit. Therefore, all even indexed data resides in the first half of the bit-reversed vector, and all odd indexed data resides in the second half of the bit-reversed vector. This process meets two needs: separation of even and odd indexed-data vectors and bit-reversing both vectors.

2.8.4 2048-point FFT Source Code

The following C source code demonstrates a 2048-point FFT using this approach. Two 1024-point FFTs are computed on the HWAFFT, and a final Radix-2 stage is performed on the CPU to generate a 2048-point FFT result:

```
#define FFT_FLAG         ( 0 )       /* HWAFFT to perform FFT */
#define IFFT_FLAG        ( 1 )       /* HWAFFT to perform IFFT */
#define SCALE_FLAG       ( 0 )       /* HWAFFT to scale butterfly output */
#define NOSCALE_FLAG     ( 1 )       /* HWAFFT not to scale butterfly output */
#define OUT_SEL_DATA     ( 0 )       /* Indicates HWAFFT output located in input data vector */
#define OUT_SEL_SCRATCH  ( 1 )       /* Indicates HWAFFT output located in scratch vector  */
#define DATA_LEN_2048    ( 2048 )
#define TEST_DATA_LEN    (DATA_LEN_2048)

// Static Memory Allocations and Alignment:
#pragma DATA_SECTION(data_br_buf, "data_br_buf");
```

```
#pragma DATA_ALIGN (data_br_buf, 4096);
   // Align 2048-pt bit-reverse dest vector to byte addr w/ 13 least sig zeros
Int32 data_br_buf[TEST_DATA_LEN];

#pragma DATA_SECTION(data_even_buf, "data_even_buf");
Int32 data_even_buf[TEST_DATA_LEN/2];

#pragma DATA_SECTION(data_odd_buf, "data_odd_buf");
Int32 data_odd_buf[TEST_DATA_LEN/2];

#pragma DATA_SECTION(scratch_even_buf, "scratch_even_buf");
Int32 scratch_even_buf[TEST_DATA_LEN/2];

#pragma DATA_SECTION(scratch_odd_buf, "scratch_odd_buf");
Int32 scratch_odd_buf[TEST_DATA_LEN/2];

// Function Prototypes:
Int32 CPLX_Mul(Int32 op1, Int32 op2);
   // Yr = op1_r*op2*r - op1_i*op2_i, Yi = op1_r*op2_i + op1_i*op2_r
Int32 CPLX_Add(Int32 op1, Int32 op2, Uint16 scale_flag);
   // Yr = 1/2 * (op1_r + op2_r), Yi = 1/2 *(op1_i + op2_i)
Int32 CPLX_Subtract(Int32 op1, Int32 op2, Uint16 scale_flag);
   // Yr = 1/2 * (op1_r - op2_r), Yi = 1/2 *(op1_i - op2_i)

// Declare Variables
Int32 *data_br;
Int32 *data;
Int32 *data_even, *data_odd;
Int32 *scratch_even, *scratch_odd;
Int32 *twiddle;
Int32 twiddle_times_data_odd;
Uint16 fft_flag;
Uint16 scale_flag;
Uint16 out_sel;
Uint16 k;

// Assign pointers to static memory allocations
data_br = data_br_buf;
data_even = data_even_buf;
data_odd = data_odd_buf;
scratch_even = scratch_even_buf;
scratch_odd = scratch_odd_buf;
twiddle = twiddle_buf;     // 1024-pt Complex Twiddle Table
data = invec_fft_2048pts;  // 2048-pt Complex Input Vector

// HWAFFT flags:
fft_flag = FFT_FLAG;       // HWAFFT to perform FFT (not IFFT)
scale_flag = SCALE_FLAG;   // HWAFFT to scale by 2 after each butterfly stage

// Bit-reverse input data for DIT FFT calculation
hwafft_br(data, data_br, DATA_LEN_2048);
   // data_br aligned to log2(4*2048) = 13 zeros in least sig bits
data = data_br;

// Split data into even-indexed data & odd-indexed data
// data is already bit-reversed, so even-indexed data = first half & odd-
indexed data = second half
for(k=0; k<DATA_LEN_2048/2; k++)
{
    data_even[k] = data[k];
    data_odd[k]  = data[k+DATA_LEN_2048/2];
}

// 1024-pt FFT the even data on the FFT Hardware Accelerator
out_sel = hwafft_1024pts(data_even, scratch_even, fft_flag, scale_flag);
if(out_sel == OUT_SEL_SCRATCH) data_even = scratch_even;
```

```
// 1024-pt FFT the odd data on the FFT Hardware Accelerator
out_sel = hwafft_1024pts(data_odd, scratch_odd, fft_flag, scale_flag);
if(out_sel == OUT_SEL_SCRATCH) data_odd = scratch_odd;

// Combine the even and odd FFT results with a final Radix-2 Butterfly stage on the CPU
for(k=0; k<DATA_LEN_2048/2; k++)  // Computes 2048-point FFT
{
    // X(k)     = 1/2*(X_even[k] + Twiddle[k]*X_odd(k))
    // X(k+N/2) = 1/2*(X_even[k] - Twiddle[k]*X_odd(k))
    // Twiddle[k]*X_odd(k):

    twiddle_times_data_odd = CPLX_Mul(twiddle[k], data_odd[k]);

    // X(k):
    data[k] = CPLX_Add(data_even[k], twiddle_times_data_odd, SCALE_FLAG);  // Add then scale by 2

    // X(k+N/2):
    data[k+DATA_LEN_2048/2] = CPLX_Subtract(data_even[k], twiddle_times_data_odd, SCALE_FLAG);
         //Sub then scale
}

result = data;   //2048-pt FFT result

/* END OF 2048-POINT FFT SOURCE CODE */
```

2.9 Appendix A Methods for Aligning the Bit-Reverse Destination Vector

The optimized bit-reverse function hwafft_br requires the destination vector to be data aligned such that the starting address of the destination vector, data_br, contains $\log_2(4 * N)$ zeros in the least significant bits of the binary address. There are a few different ways to force the linker map the bit-reverse destination vector to an address with $\log_2(4 * N)$ zeros in the least significant bits. Three different methods are shown here. For further details, refer to the *TMS320C55x C/C++ Compiler User's Guide* (SPRU280).

2.9.1 Statically Allocate Buffer at Beginning of Suitable RAM Block

NOTE: To execute the HWAFFT routines from the ROM of the DSP, the programmer must satisfy memory allocation restrictions for the data and scratch buffers. For an explanation of the restrictions and workarounds, see the device-specific errata:

- *TMS320C5517 Fixed-Point DSP Silicon Errata* (literature number SPRZ383)

Place the buffer at the beginning of a DARAM or SARAM block with $\log_2(4 * N)$ zeros in the least significant bits of its byte address. For example, memory section DARAM2_3 below starts at address 0x0004000, which contains 14 zeros in the least significant bits of its binary address (0x0004000 = 0b0100 0000 0000 0000). Therefore, this address is a suitable bit-reverse destination vector for FFT Lengths up to 4096-points because $\log_2(4 * 4096) = 14$.

In the Linker CMD File...

```
MEMORY
{
  MMR    (RWIX): origin = 0000000h, length = 0000c0h    /* MMRs */
  DARAM0 (RWIX): origin = 00000c0h, length = 001f40h    /* on-chip DARAM 0,   4000 words */
  DARAM1 (RWIX): origin = 0002000h, length = 002000h    /* on-chip DARAM 1,   4096 words */
  DARAM2_3 (RWIX): origin = 0004000h, length = 004000h  /* on-chip DARAM 2_3, 8192 words */
  DARAM4 (RWIX): origin = 0008000h, length = 002000h    /* on-chip DARAM 4,   4096 words */
  ... (leaving out rest of memory sections)
}

SECTIONS
{
   data_br_buf : > DARAM2_3  /* ADDR = 0x004000, Aligned to addr with 14 least-sig zeros */
```

```
}
```

2.9.2 Use the ALIGN Descriptor to Force $log_2(4 * N)$ Zeros in the Least Significant Bits

The ALIGN descriptor forces the alignment of a specific memory section, while providing the linker with added flexibility to allocate sections across the entire DARAM or SARAM because no blocks are statically allocated. It aligns the memory section to an address with log_2(ALIGN Value) zeros in the least significant bits of the binary address.

For example, the following code aligns data_br_buf to an address with 12 zeros in the least significant bits, suitable for a 1024-point bit-reverse destination vector.

In the Linker CMD File...

```
MEMORY
{
  MMR    (RWIX): origin = 0000000h, length = 0000c0h  /* MMRs */
  DARAM (RWIX): origin = 00000c0h, length = 00ff40h   /* on-chip DARAM  32 Kwords */
  SARAM  (RWIX): origin = 0010000h, length = 040000h  /* on-chip SARAM 128 Kwords */
}

SECTIONS
{
   data_br_buf   : > DARAM   ALIGN = 4096
                          /* 2^12 = 4096 , Aligned to addr with 12 least-sig zeros */
}
```

2.9.3 Use the DATA_ALIGN Pragma

The DATA_ALIGN pragma is placed in the source code where the vector is defined. The syntax is shown below.

#pragma DATA_ALIGN (*symbol***,** *constant***);**

The DATA_ALIGN pragma aligns the symbol to an alignment boundary. The boundary is the value of the constant in words. For example, a constant of 4 specifies a 64-bit alignment. The constant must be a power of 2.

In this example, a constant of 2048 aligns the data_br_buf symbol to an address with 12 zeros in the least significant bits, suitable for a 1024-point bit-reverse destination vector.

In the source file where data_br is declared (e.g., main.c):

```
#pragma DATA_SECTION(data_br_buf, "data_br_buf");
#pragma DATA_ALIGN (data_br_buf, 2048);
Int32 data_br_buf[TEST_DATA_LEN];
```

Direct Memory Access (DMA) Controller

Topic **Page**

3.1 Introduction

The following sections describe the features and operation of the direct memory access (DMA) controller in the digital signal processor (DSP). The DMA controller allows movement of data between internal/external memory and other peripherals without CPU intervention.

3.1.1 Purpose of the DMA Controller

The DMA controller is used to move data between internal memory, external memory, and peripherals without intervention from the CPU and in the background of CPU operation.

The DSP includes four DMA controllers with four DMA channels each for a total of 16 DMA channels. Aside from the DSP resources they can access, all four DMA controllers are identical. Throughout this document the general operation of each DMA controller will be described. Differences between each DMA controller will be noted when necessary.

3.1.2 Key Features of the DMA Controller

The DMA controller has the following features:

- Operation that is independent of the CPU.
- Four channels per DMA controller, which allow the DMA controller to keep track of the context of four independent block transfers.
- Event synchronization. DMA transfers in each channel can be made dependent on the occurrence of selected events. For details, see Section 3.2.7.
- An interrupt for each channel. Each channel can send an interrupt to the CPU on completion of the programmed transfer. See Interrupt Support in Section 3.2.14.
- A dedicated clock idle domain. The user can put the four device DMA controllers into a low-power state by turning off their input clock. See Power Management in Section 3.2.15.
- Ping-Pong mode for DMA transfer. This mode provides double buffering capability fully implemented in hardware. For details, see Section 3.2.9.

To read about the registers used to program the DMA controller, see Section 3.4.

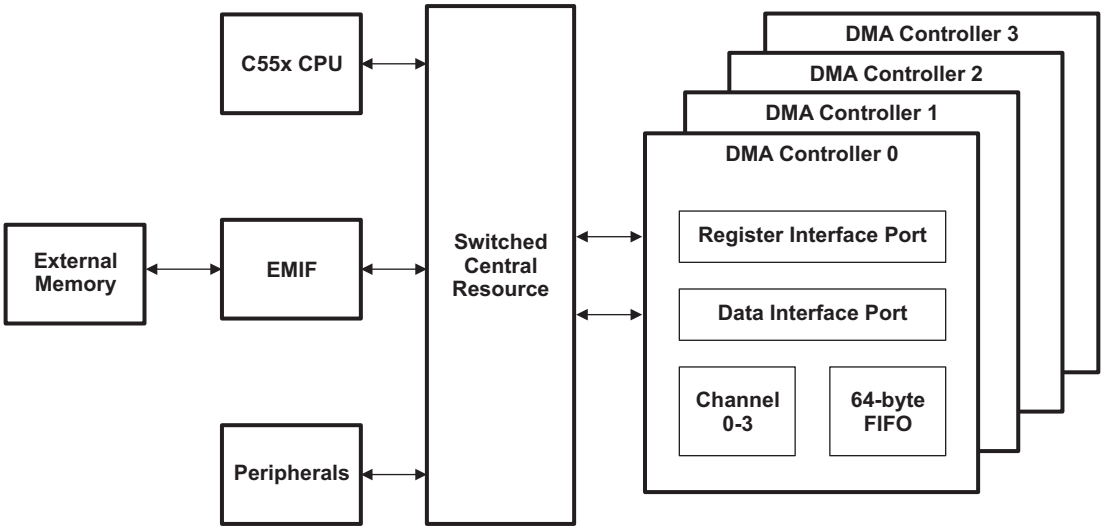
3.1.3 Block Diagram of the DMA Controller

Figure 3-1 is a conceptual diagram of connections between the DMA controller and other parts of the DSP. The DMA controller is made up of the following blocks:

- Register interface port. The CPU uses this port to access the DMA controller registers.
- Data interface port. The DMA controller accesses internal dual-access RAM (DARAM), internal single-access RAM (SARAM), external memory, and on-chip peripherals through its data interface port.
- Data transfers are carried out by the four DMA channels. (The DMA channels are described in Section 3.2.3)
- 64-byte FIFO. As data is read from the source address, it is placed in the DMA controller FIFO. The four DMA channels must share the DMA controller FIFO; the FIFO can only be accessed by a single channel at a time.

It is possible for multiple channels to request access to the DMA controller FIFO at the same time. In this case the DMA controller arbitrates amongst the DMA channels using a round-robin arbitration scheme.

Figure 3-1. Conceptual Block Diagram of the DMA Controller



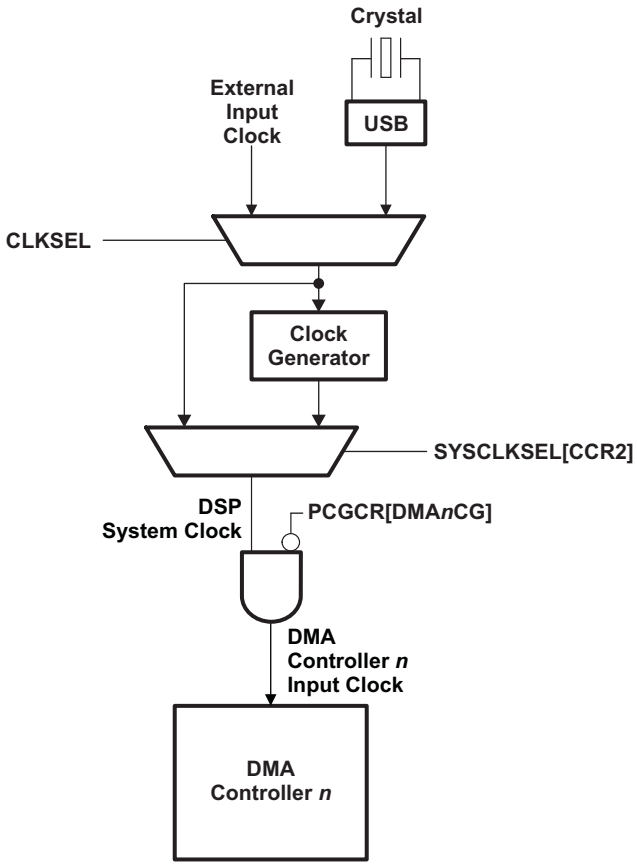
3.2 DMA Controller Architecture

3.2.1 Clock Control

As shown in Figure 3-2, the clock generator receives either the USB oscillator clock or a signal from an external clock source and produces the DSP system clock. This clock is used by the DSP CPU and peripherals.

The DSP includes logic which can be used to gate the clock to its on-chip peripherals, including each of the four DMA controllers. The input clock to the DMA controllers can be enabled and disabled through the peripheral clock gating configuration registers (PCGCR1 and PCGCR2).

Figure 3-2. Clocking Diagram for the DMA Controller



3.2.2 Memory Map

On the DSP, although all DMA controllers can access internal dual-access RAM (DARAM) and single-access RAM (SARAM), each DMA controller can only access a subset of on-chip peripherals. Also, only DMA controller 3 has access to external memory. The addresses from the point of view of the CPU as compared to the DMA controller are different for DARAM, SARAM and external memory. The DMA controller reads on-chip and off-chip memory by adding the offsets introduced in . The memory map, as seen by the DMA controllers and the CPU, is shown in Table 3-1. Peripherals not shown in Table 3-1 are not accessible by the DMA controllers.

Table 3-1. DMA Controller Memory Map

DMA Start *Byte* Address	CPU Start *Word* Address (I/O Space)	CPU Start *Word* Address (Data Space)	DSP Memory Map	DMA Controller 0 Memory Map	DMA Controller 1 Memory Map	DMA Controller 2 Memory Map	DMA Controller 3 Memory Map
0000 2800h	00 2800h	-	I2S0	I2S0	Reserved	Reserved	Reserved
0000 3400h	00 3400h	-	McSPI	McSPI			
0000 3A00h	00 3A00h	-	MMC/SD0	MMC/SD0			
0000 3B00h	00 3B00h	-	MMC/SD1	MMCSD1			
0001 0000h[1]	-	00 0000h [1]	DARAM	DARAM	DARAM	DARAM	DARAM
0009 0000h	-	00 8000h	SARAM	SARAM	SARAM	SARAM	SARAM
0000 1B00h	00 1B00h	-	UART	Reserved	UART	Reserved	Reserved
0000 2A00h	00 2A00h	-	I2S2		I2S2		
0000 1A00h	00 1A00h	-	I2C		Reserved	I2C	
0000 2B00h	00 2B00h	-	I2S3			I2S3	
0000 7000h	00 7000h	-	10-bit SAR			10-bit SAR	
0000 6000h	00 4000h	-	McBSP			Reserved	McBSP
0100 0000h	-	02 8000h	SDRAM				SDRAM
0200 0000h	-	40 0000h	EMIF CS2				EMIF CS2
0300 0000h	-	60 0000h	EMIF CS3				EMIF CS3
0400 0000h	-	70 0000h	EMIF CS4				EMIF CS4
0500 0000h	-	78 0000h	EMIF CS5				EMIF CS5

(1) Word addresses 00 0000h-00 005Fh (which correspond to byte addresses 00 0000h-00 00BFh) are reserved for the memory-mapped registers (MMRs) of the DSP CPU.

3.2.3 DMA Channels

Each DMA controller has four channels to transfer data among the DSP resources (DARAM, SARAM, external memory, and peripherals). Each channel reads data from the source address and writes data to the destination address.

The DMA first in, first out (FIFO) buffer is used by the channels to store transfer data; this allows the data transfer to occur in two stages (see Figure 3-3).

Data read access Transfer of data from the source address to the DMA FIFO buffer.

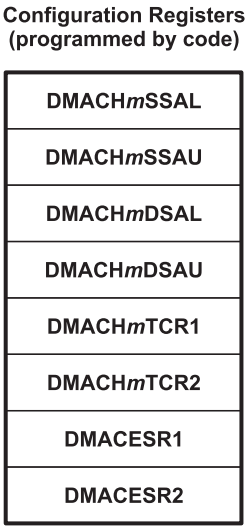
Data write access Transfer of data from the DMA FIFO buffer to the destination address.

Figure 3-3. Two-Part DMA Transfer

The set of conditions under which transfers occur in a channel is called the channel context. Each of the four channels contains a register structure for programming and updating the channel context (see Figure 3-4). The user code modifies the configuration registers. The DMA channel becomes active when the channel is enabled (EN = 1 in DMACH*m*TCR2).

The channel configuration registers cannot be programmed while the channel is active (EN = 1 in DMACH*m*TCR2). Modifying channel registers while the DMA channel is running may cause unpredictable operation of the channel. To change a DMA channel configuration, the channel must first be disabled (EN = 0 in DMACH*m*TCR2). The DMA controller will always complete any on-going burst transfer before stopping channel activity. Note that a block transfer may consist of a number of burst transfers. The channel is considered to be active until it completes the burst transfer during which the channel is disabled. After a channel has been disabled, the channel context must be fully reloaded.

Figure 3-4. Registers for Controlling the Context of a Channel



3.2.4 Channel Source and Destination Start Addresses

During a data transfer in a DMA channel, the first address at which data is read is called the source start address. The first address to which the data is written is called the destination start address. These are byte addresses. Each channel contains the following registers for specifying the start addresses.

In Ping-Pong mode, the source or destination start address is the start address of the Ping buffer. The length of the **Ping** and the **Pong** buffers should be half of the DMA transfer size as programmed in TCR1. The first half is assumed to be the **Ping** buffer and the second half is assumed to be the **Pong** buffer. These two buffers are required to be contiguous in the memory space and are of equal size. The programmer is responsible for allocating these buffers contiguously. It is recommended to consider the **Ping** and the **Pong** buffers to be a single data buffer in the memory space so that the compiler always allocates them next to each other.

Table 3-2. Registers Used to Define the Start Addresses for a DMA Transfer

Register	Load with...
DMACH*m*SSAL	Source start address (least-significant part)
DMACH*m*SSAU	Source start address (most-significant part)
DMACH*m*DSAL	Destination start address (least-significant part)
DMACH*m*DSAU	Destination start address (most-significant part)

Section 3.2.2 shows a high-level memory map of the DSP as seen by the DMA controllers and the CPU. The table shows both the word addresses (23-bit addresses) used by the CPU and byte addresses (32-bit addresses) used by the DMA controller.

The following sections explain how to determine the start address for memory accesses and I/O accesses.

CAUTION

All data buffers in on-chip or off-chip memory should be 32-bit aligned. For more information on managing memory, see the *TMS320C55x Assembly Language Tools User Guide* (SPRU280).

Additionally, the amount of data (in bytes) to be transferred as programmed in the LENGTH field in DMACH*m*TCR1 should be a multiple of 4 bytes x $2^{BURSTMODE}$ field in DMACH*m*TCR2, that is, LENGTH = (4 x $2^{BURSTMODE}$) bytes.

For an explanation of this workaround, see the *TMS320C5517 Fixed-Point DSP Silicon Errata* [literature number: SPRZ383].

3.2.4.1 Start Address for On-Chip Memory

The CPU uses word addresses and the DMA uses byte addresses. Furthermore, an offset must be added to CPU addresses to generate DMA addresses.

Follow these steps to program the DMA controller with a byte address corresponding to a CPU word address:

1. Identify the correct start address. If you have a word address, shift it left by 1 bit to form a byte address of 32 bits. For example, the CPU word address for SARAM block 0 (00 8000h) should be converted to byte address 0001 0000h.
2. Add correct offset to the CPU byte address. For DARAM, add a value of 01 0000h to the desired byte address. For SARAM, add a value of 08 0000h. For example, since byte address 0001 0000h corresponds to SARAM block 0, a value of 0008 0000h should be added to the byte address to generate 0009 0000h.
3. Load the 16 least significant bits (LSBs) of the byte address into DMACH*m*SSAL (for source) or DMACH*m*DSAL (for destination).
4. Load the 16 most significant bits (MSBs) of the byte address into DMACH*m*SSAU (for source) or DMACH*m*DSAU (for destination).

3.2.4.2 Start Address for External Memory Space

The CPU uses word addresses and the DMA uses byte addresses. Furthermore, the CPU and DMA controller use different starting addresses for the external memory chip select spaces.

1. Calculate the address offset. Determine the starting word address for the chip select space being used, and then subtract it from the CPU word address (see for starting addresses of all chip select spaces). For example, if you are using word address 40 8000h (EMIF CS2 space in external memory), then subtract 40 0000h (starting word address for EMIF CS2 space in external memory) to generate 00 8000h (40 8000h - 40 0000h).
2. Convert the offset to a byte address offset. Shift the offset left by 1 bit to form a byte address offset of 32 bits. For example, offset 00 8000h should be converted to byte address offset 0001 0000h.
3. Calculate the correct DMA byte address. Use to identify the starting DMA byte address of chip select space being used, and then add it to the byte address offset to generate the DMA byte address. For example, byte address offset 0001 0000h becomes 0201 0000h (0200 0000h + 0001 0000h).
4. Load the 16 least significant bits (LSBs) of the byte address into DMACH*m*SSAL (for source) or DMACH*m*DSAL (for destination).
5. Load the 16 most significant bits (MSBs) of the byte address into DMACH*m*SSAU (for source) or DMACH*m*DSAU (for destination).

3.2.4.3 Start Address for I/O Space

The CPU uses word addresses and the DMA uses byte addresses. The following steps describe how to program the DMA controller with a byte address for a peripheral memory-mapped register:

1. Identify the correct DMA byte address for the peripheral memory-mapped register. The starting DMA byte addresses for the memory-mapped register space of the DSP peripherals are given in Table 3-1.

2. Load the 16 least significant bits (LSBs) of the byte address into DMACH*m*SSAL (for source) or DMACH*m*DSAL (for destination).
3. Load the 16 most significant bits (MSBs) of the byte address into DMACH*m*SSAU (for source) or DMACH*m*DSAU (for destination).

3.2.5 Updating Addresses in a Channel

During data transfers in a DMA channel, the DMA controller begins its read and write accesses at the start addresses you specify (as described in Section 3.2.4). Each time the DMA controller services a channel, it transfers the number of double words specified in the BURSTMODE of the channel's transfer control register (DMACH*m*TCR2). If constant addressing mode is selected (DST/SRCAMODE = 10b), the channel does not update the addressing registers (DMACH*m*SSAU/L and DMACH*m*DSAU/L). Otherwise, if the channel is set to automatic-post increment addressing mode (DST/SRCAMODE = 00b), the channel increments the value in the addressing registers by the total number of bytes transferred.

To change the source or destination address of a channel, the channel must first be disabled by setting EN = 0. The CPU can then update the Source/Destination Address registers and the Transfer Control registers before restarting the channel.

3.2.6 Data Burst Capability

During a read-write transaction (from source address to destination address) through the Switched Central Resource (see), the DMA controller moves data one double word at a time by default. Since every transaction request involves cycle overheads (see Section 3.2.11), the DMA throughput can be increased by programming the DMA channel to move multiple double words during a transaction, provided both the source and destination targets associated with the transfer support burst capability. The DMA controller can burst to and from SARAM, DARAM, external memory (EMIF) and UART. For a list of DSP resources that support data bursting, see Table 3-3.

The BURSTMODE bits of the Transfer Control Register 2 (DMACH*m*TCR2) specify the number of double words the DMA controller moves each time it services a channel, that is, the DMA controller executes a burst of n double words (n = 2, 4, 8, or 16) each time a channel is serviced instead of moving only 1 double word.

Note that the DMA controller services one channel at a time. Each time the DMA services a channel it must transfer the number of double words specified by the burst mode bits. Therefore, care must be taken when programming a channel to use a high burst count since this may impact the minimum amount of time it takes the DMA controller to service other channels. DMA channels are serviced in a round-robin fashion.

Table 3-3. Destinations/Sources That Support DMA Bursting

Destination/Source Address	Burst Mode Supported
DARAM	1, 2, 4, 8, 16 double words
SARAM	1, 2, 4, 8, 16 double words
UART (1)	1, 2, 4, 8 double words
EMIF	1, 2, 4, 8, 16 double words

(1) The UART treats each double word transfer as a single byte. Therefore, an 8 double word transfer from the DMA to the UART will yield 8 new bytes in the UART FIFO. For more information, see Section 15.1, *Universal Asynchronous Receiver/Transmitter (UART)*.

3.2.7 Synchronizing Channel Activity to DSP Peripheral Events

Activity in a channel can be synchronized to an event in a DSP peripheral. Synchronization is enabled by setting SYNCMODE = 1 in DMACH*m*TCR2. Using the CH*n*EVT bits of DMACESR1 and DMACESR2, the user can specify which synchronization event triggers channel activity. Note that synchronization to an event signaled by the driving of an external interrupt pin is not supported.

If event synchronization is enabled, the channel will wait for the event from the peripheral as programmed in the DMACESR1 and/or DMACESR2 registers before reading from the source address into the DMA FIFO. The synchronization event will trigger the channel to transfer the number of bytes specified by the BURSTMODE bits into the FIFO. Once the FIFO has been filled, the DMA channel will begin writing to the destination address to empty the FIFO.

In non-synchronized transfers (SYNCMODE = 0), the channel sends an access request to the source address as soon as the channel is enabled (EN = 1 in DMACH*m*TCR2). The channel will transfer data from the source address to the FIFO, and then to the destination address until the entire block transfer has been completed or until the user program disables the channel.

The synchronization events are not buffered. Hence, if a synchronization event occurs when the DMA channel is still servicing the previous event, the overlapping (second) event is dropped. In this scenario, the DMA controller does not disable the affected channel nor does it signal an error to the CPU. Therefore, care must be taken when programming the DSP to ensure that the DMA has enough clock cycles to transfer the required number of data bytes on each synchronization event.

Note that some peripherals can generate interrupts whenever a data underrun or overrun condition occurs. The user program can use these interrupts to detect dropped synchronization events.

CAUTION

When using synchronization events, you must set EN = 1 and SYNCMODE = 1 during the same write cycle to DMACH*m*TCR2. The DMA channel will transfer the first data value when it receives the synchronization event specified by CH*n*EVT in DMACESR1 and DMACESR2. Also, when disabling the channel, you must set EN = 0 and SYNCMODE = 0 during the same write cycle to DMACH*m*TCR2.

3.2.8 Channel Auto-Initialization Capability

After a block transfer is completed (all of the bytes specified by LENGTH in DMACH*m*TCR1 have been moved), the DMA controller automatically disables the channel (EN = 0). If it is necessary for the channel to be used again, the CPU can reprogram the new channel context and re-enable the DMA channel, or the DMA controller can automatically initialize the new context and re-enable the channel.

When auto-initialization is used, after each block transfer is completed, the DMA controller automatically reloads the transfer control register and the source and destination start address registers and re-enables the channel allowing the channel to run again. Auto-initialization is enabled by setting the AUTORLD = 1 in the transfer control register (DMACH*m*TCR2).

CAUTION

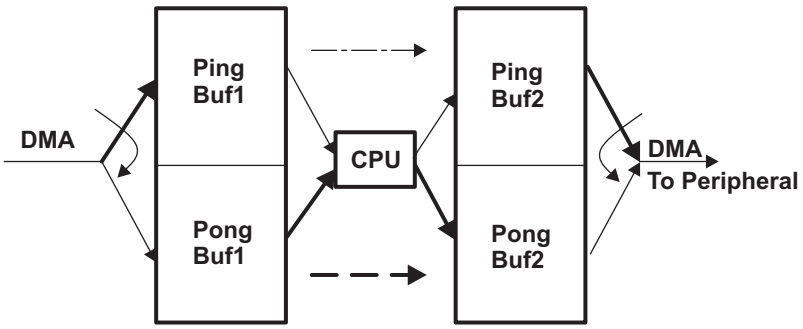
The auto-initialization feature can only be used when event synchronization is used (SYNCMODE = 1 in DMACH*m*TCR2).

Using auto-initialization feature without event synchronization can lead to unintended behavior of the DMA controller.

3.2.9 Ping-Pong DMA Mode

The Ping-Pong mode for DMA transfer can be used to do continuous processing of incoming data without losing any samples. Every channel in the DMA controller has the capability of being configured in the Ping-Pong mode. This mode can be initiated by setting PING_PONG_EN bit of the Transfer Control Register 2 (TCR2) to 1. For more information, see Section 3.4.3. You should also define the transfer buffer length in bytes in the Transfer Control Register 1 (TCR1). The length of the **Ping** and the **Pong** buffers should be half of the DMA transfer size as programmed in TCR1. The first half is assumed to be the **Ping** buffer and the second half is assumed to be the **Pong** buffer. These two buffers are required to be contiguous in the memory space and of equal size. The programmer is responsible for allocating these buffers contiguously. It is recommended to consider the **Ping** and the **Pong** buffers to be a single data buffer in the memory space so that the compiler always allocates them next to each other.

Figure 3-5. Ping-Pong Mode for DMA Data Transfer



As shown in Figure 3-5, in Ping-Pong mode, DMA starts filling up the **Ping** buffer first. Once the **Ping** buffer is full, a DMA interrupt is generated to the CPU and the LAST_XFER bit (bit 1 in TCR2) is set to 0, which indicates that the data in the **Ping** buffer can be processed. The CPU can start processing the samples in the **Ping** buffer while the DMA is transferring the data to the **Pong** buffer. When the **Pong** buffer is full, another DMA interrupt is sent to the CPU to indicate the availability of data in the **Pong** buffer and the LAST_XFER bit in TCR2 is set to 1. If the AUTORLD bit = 1 in TCR2 (), then the DMA automatically reinitiates the DMA transfer until either EN or AUTORLD bit is set to 0. In the case that the AUTORLD bit is set to 0, DMA stops data transfer after the **Pong** buffer is full and the interrupt is generated. It also resets the EN bit to 0 in TCR2.

At any time during the DMA transfer, LAST_XFER bit of TCR2 (Section 3.4.3) can be polled to find whether the last completed transfer was the **Ping** or **Pong** buffer.

3.2.10 Monitoring Channel Activity

The DMA controller can send an interrupt to the CPU whenever a channel has completed a block transfer. Each channel has an interrupt enable (DMA*n*CH*m*IE) bit in the interrupt enable register (DMAIER) and corresponding status bits in the interrupt flag register (DMAIFR). When a channel completes a block transfer, the DMA controller checks the corresponding DMA*n*CH*m*IE bit and acts accordingly:

- If the DMA*n*CH*m*IE bit is 1 (the interrupt is enabled), the DMA controller sets the corresponding status bit and sends the associated interrupt request to the CPU. Your program must manually clear bits in DMAIFR by writing a 1 to them.
- If the DMA*n*CH*m*IE bit is 0, no interrupt is sent and the status bit is not affected.

Each channel also includes a STATUS bit in DMACH*m*TCR2 to indicate the state of the channel transfer. The DMA controller sets the channel STATUS bit to 1 if:

- A nonzero value is written on LENGTH in DMACH*m*TCR1.
- A write access is performed to DMACH*m*TCR2 and LENGTH has a nonzero value.

The DMA controller clears the STATUS bit to 0 if:

- All the bytes specified by LENGTH in DMACH*m*TCR1 have been transferred.
- A value of 0 is written to LENGTH in DMACH*m*TCR1.

The LAST_XFER bit which is bit 1 in TCR2 (Section 3.4.3) indicates whether the last completed transfer was the **Ping** or the **Pong** buffer. This bit is valid only when Ping-Pong DMA mode is enabled (PING_PONG_EN bit in TCR2 is 1).

LAST_XFER bit in TCR2 is set to 0 if:

- The last completed transfer was the **Ping** buffer.

LAST_XFER bit in TCR2 is set to 1 if:

- The last completed transfer was the **Pong** buffer.

3.2.11 Latency in DMA Transfers

Each element transfer in a channel is composed of a read access (a transfer from the source location to the DMA controller FIFO) and a write access (a transfer from the DMA controller FIFO to the destination location). The time to complete this activity depends on factors such as:

- The selected frequency of the CPU clock signal. This signal, as propagated to the DMA controller, determines the timing for all DMA transfers.
- Wait states or other extra cycles added by or resulting from an interface.
- Activity on other channels. Since channels are serviced in a sequential order, the number of pending DMA service requests in the other channels affects how often a given channel can be serviced.
- Competition from the CPU or other DMA controllers. If a DMA controller and the CPU request access to the same internal memory block or peripheral in the same cycle and the memory block or peripheral cannot service both requests at the same time, the CPU request has higher priority. The DMA request is serviced as soon as there are no pending CPU requests.
- The timing of synchronization events (if the channel is synchronized). The DMA controller cannot service a synchronized channel until the synchronization event has occurred. For more details on synchronization, see Section 3.2.7.

The minimum (best-case) latency for a burst DMA transfer can be summarized as follows:

- For transfers initiating from internal memory: the first access for word read and write takes 8 cycles, while consecutive accesses take 2 more cycles. Thus the DMA takes 2N + 6 system clock cycles to complete a burst transfer, where N corresponds to the burst size in words.
- For transfers initiating from a peripheral source: the first access for word read and write takes 6 cycles, while consecutive accesses take 2 more cycles. Thus the DMA takes 2N + 4 system clock cycles to complete a burst transfer, where N corresponds to the burst size in words.

The burst size of the DMA is specified through the BURSTMODE bits. Note that a block transfer may consist of a number of burst transfers. For accesses to external memory, the number of cycles the DMA controller takes to read or write data is determined by the EMIF settings, including the memory type used, programmed timings, and any delays caused by the memory itself (such as control of the wait pin).

3.2.12 Reset Considerations

The DMA controller has one reset source: a hardware reset. This reset is always initiated during a full chip reset. Alternatively, software can force a hardware reset on all DMA controllers through the DMA_RST bit of the peripheral reset control register (PRCR). See the device data manual for more details on PRCR. Please note that the DMA controller input clock must be enabled when using DMA_RST (see Section 3.2.1).

When a hardware reset occurs, all the registers of the DMA controllers are set to their default values. The DMA controllers remain inactive until programmed by software.

3.2.13 Initialization

To initialize the DMA controller follow these steps:

1. Ensure the DMA controller is out of reset by setting the DMA_RST bit to 0 in the peripheral reset control register (PRCR). PRCR is a chip configuration register, it is not part of the DMA controller, see the device data manual for more details.
2. Enable the DMA controller input clock by setting the corresponding DMA*n*CG bit to 0 in the peripheral clock gating configuration registers (PCGCR1 and PCGCR2). PCGCR1 and PCGCR2 are chip configuration registers, they are not part of the DMA controller, see the device data manual for more details.
3. Ensure that all DMA channel interrupt flags are cleared by writing a 1 to the bits of the DMA interrupt flag register (DMAIFR). Also, ensure all DMA interrupt flags in the CPU interrupt flag registers (IFR0 and IFR1) are cleared.
4. If using interrupts, enable the desired channel interrupt by setting the DMA*n*CH*m*IE bits of the interrupt enable register (DMAIER). The CPU interrupt enable bit (INTEN) in the transfer control register 2 (DMACH*m*TCR2) must also be set.
5. If using synchronization events, select the event to be used through the CH*m*EVT bits of the channel event source registers (DMA*n*CESR1 and DMA*n*CESR2). The synchronization mode bit (SYNCMODE) of DMACH*m*TCR2 must also be set, although this should be done only when the channel is ready to be enabled.
6. Load the source address to the source start address registers (DMACH*m*SSAL and DMACH*m*SSAU). See Section 3.2.4, Start Address in a Channel, for more information on calculating the correct source start address.
7. Load the destination address to the destination start address registers (DMACH*m*DSAL and DMACH*m*DSAU). See Section 3.2.4, Start Address in a Channel, for more information on calculating the correct destination start address.
8. Load the DMA transfer control register 1 (DMACH*m*TCR1) with the number of double words to transfer. Note that the number of double words must be specified in bytes. For example, for a 256 double word transfer, program this field with 1024 (256 x 4 = 1024). When Ping-Pong DMA mode is enabled, this is the size of the Ping and the Pong buffer combined. For more details, see Section 3.2.4.
9. Configure DMACH*m*TCR2 accordingly. Through this register you can specify the source and destination addressing modes and burst mode. You can also enable automatic reload, event synchronization, CPU interrupts and Ping-Pong mode. Note that you must keep EN = 0 and SYNCMODE = 0 during this step.
10. If the DMA channel is servicing a peripheral, ensure that the corresponding peripheral is not active and hence not generating synchronization events.
11. Enable the DMA channel by setting EN = 1 (and SYNCMODE = 1 if using synchronization events).
12. If necessary, enable peripheral being serviced the DMA channel.

If using synchronization events, the DMA channel will start a data transfer when an event is received. Otherwise, the DMA channel will start the transfer immediately. At the end of the block transfer, if interrupts are enabled, the DMA controller will generate a CPU interrupt. If interrupts are not enabled, your program can poll DMACH*m*TCR1 until either EN or STATUS are cleared to 0 by the DMA controller to determine when the DMA has finished a block transfer. If AUTORLD is set the DMA controller will restart the specified transfer (for more details, see Section 3.2.8.

For more specific examples of programming the DMA controller, see Section 3.3, *DMA Transfer Examples*.

NOTE: If a DMA controller is programmed to access on-chip memory, ensure that the MPORT is not idled in the Idle Configuration Register (ICR). Note that the value programmed in the ICR takes effect only on running the 'idle' instruction on the CPU. For more information on these registers, see the Section 1.1, *System Control*.

3.2.14 Interrupt Support

3.2.14.1 Interrupt Events and Requests

Each of the four channels of a DMA controller has its own interrupt which the user can enable or disable a channel interrupt though the DMA*n*CH*m* bits of the DMA interrupt enable register (DMAIER). The interrupts from the four DMA controllers are combined into a single CPU interrupt. You can determine which DMA channel generated the interrupt by reading the bits of the DMA interrupt flag register (DMAIFR). Your program must manually clear bits in DMAIFR by writing a 1 to them.

3.2.14.2 Interrupt Multiplexing

As described in the previous section, on the DSP, the interrupts from each of the four DMA controllers are combined into a single CPU interrupt. However, the resulting DMA interrupt is not multiplexed with any other interrupt source.

3.2.15 Power Management

Each DMA controller can be idled independently to conserve power if it is not being actively used. This is achieved by turning off the peripheral clock of each DMA controller in the peripheral clock gating configuration register (PCGCR). For more information on the PCGCR register, see Section 1.1, *System Control*.

3.2.16 Emulation Considerations

The DMA controller is not interrupted by emulation events such as an emulation breakpoint. However, an emulation suspend may halt activity in a peripheral being serviced by the DMA controller. In this case, the DMA controller activity will be indirectly suspended.

3.3 DMA Transfer Examples

The DMA controller can be used to perform two basic types of transfers: block transfers and peripheral servicing transfers. The following sections provide examples for these two typical use case scenarios.

3.3.1 Block Move Example

The most basic transfer performed by the DMA is a block move. During device operation it is often necessary to transfer a block of data from one location to another, usually between on-chip and off-chip memory.

In this example, data is copied from the external asynchronous memory . A data block of 256 double words (1024 bytes) residing at CPU word address 78 0000h (external memory CS5 space) needs to be transferred to internal CPU word address 02 7000h (SARAM, block 31), as shown in Figure 3-6.

The source address for the transfer is set to the equivalent DMA byte address of the data block in external memory, and the destination address is set to the equivalent DMA byte address of the data block in SARAM. More specifically the equivalent DMA byte addresses for source and destination buffers described in this example are 0500 0000h and 000C E000h, respectively. For more information on DMA byte addresses, see Section 3.2.2.

Figure 3-7 shows the DMA channel register contents during the transfer after the channel is enabled.

Figure 3-6. Block Move Example

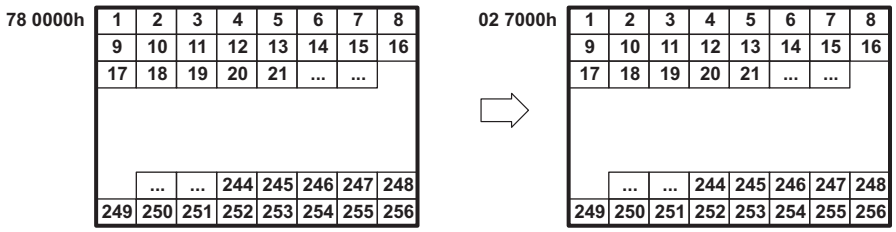


Figure 3-7. Block Move Example DMA Configuration

(a) DMA Register Contents

Register Contents	
0500h	0000h
000Ch	E000h
E020h	0400h

→

Parameter	
DMACH*m*SSAU	DMACH*m*SSAL
DMACH*m*DSAU	DMACH*m*DSAL
DMACH*m*TCR2	DMACH*m*TCR1

(b) Channel Transfer Control Options

15	14	13	12	11	10	9	8
1	1	1	1	00		00	
EN	STATUS	INTEN	AUTORLD	RSV		DSTAMODE	

7	6	5	3	2	1	0
10		000		1	0	0
SRCAMODE		BURSTMODE		SYNCMODE	LAST_XFER	PING_PONG_EN

3.3.2 Peripheral Servicing Example

The DMA controllers can service peripherals in the background of CPU operation, without requiring any CPU intervention. Through proper initialization of the DMA channels, they can be configured to continuously service on-chip and off-chip peripherals throughout device operation. Each DMA controller has a set of synchronization events which can trigger activity in DMA channels specified by the user. When programming a DMA channel to service a peripheral, it is necessary to know how data is to be presented to the DSP. Data is always provided with some kind of synchronization event as either one data sample per event (non-bursting) or multiple data samples per event (bursting).

3.3.2.1 Non-Bursting Peripherals

Non-bursting peripherals include the on-chip inter-integrated circuit (I2C) module and many external devices, such as codecs. Regardless of the peripheral, the DMA channel configuration is the same.

The I2C transmit and receive data streams are treated independently by the DMA. The transmit and receive data streams can have completely different counts, data sizes, and formats. Figure 3-8 shows DMA servicing incoming I2C data.

To transfer the incoming data stream to its proper location in internal memory, the DMA channel must be set up for a non-burst transfer with synchronization enabled. Since a receive event (ICREVT) is generated for every data sample as it arrives, it is necessary to have the DMA transfer each data sample individually. Figure 3-8 shows the DMA channel register contents for this transfer after the channel is enabled.

The source address of the DMA channel is set to the data receive register (ICDRR) address for the I2C, and the destination address is set to the start of the data block in internal memory. Since the address of ICDRR is fixed, the source address mode is set to 10b (constant address) and the destination address mode is set to 00b (automatic post-increment). Note that in this example the destination address is set to the DMA byte address 000C E000h, which corresponds to SARAM block 31. For more information on DMA byte addresses, see Section 3.2.2.

Note that the DMA will transfer a full double word from ICDRR to the destination address every time a receive synchronization event is generated by the I2C. When allocating memory for the receive buffer, two 16-bit words must be allocated for every I2C data sample.

Figure 3-8. Servicing Incoming I2C Data Example

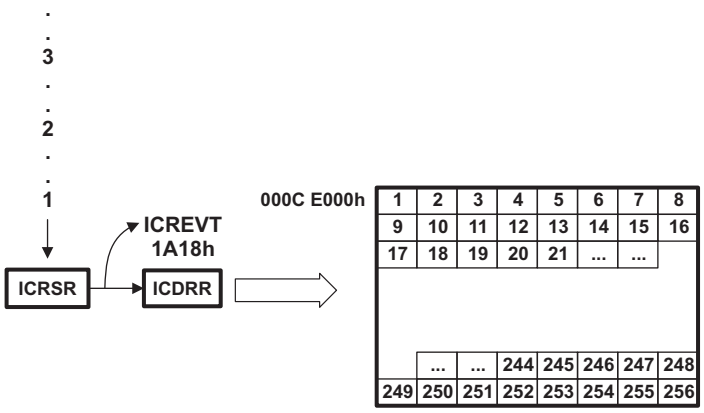


Figure 3-9. Servicing Incoming I2C Data Example DMA Configuration

(a) DMA Register Contents

Register Contents	
0000h	1A18h
000Ch	E000h
F084h	0400h

→

Parameter	
DMACH*m*SSAU	DMACH*m*SSAL
DMACH*m*DSAU	DMACH*m*DSAL
DMACH*m*TCR2	DMACH*m*TCR1

(b) Channel Transfer Control Options

15	14	13	12	11 10	9 8
1	1	1	1	00	00
EN	STATUS	INTEN	AUTORLD	RSV	DSTAMODE

7 6	5 3	2	1	0
10	000	1	0	0
SRCAMODE	BURSTMODE	SYNCMODE	LAST_XFER	PING_PONG_EN

3.3.2.2 Bursting Peripherals

Bursting peripherals include the universal asynchronous receiver/transmitter (UART) and the external memory controller (EMIF). For these peripherals, the DMA can be configured to transfer multiple data samples every time the channel is serviced.

The UART transmit and receive data streams are treated independently by the DMA. The transmit and receive data streams can have completely different counts, data sizes, and formats. Furthermore, the DMA burst size feature can be used to empty or fill the UART FIFO every time the UART generates a synchronization event. Figure 3-10 shows DMA servicing incoming UART data.

To transfer the incoming data in the UART FIFO to its proper location in internal memory, the DMA channel must be set up for a burst transfer with synchronization enabled. Since a receive event (URXEVT) is generated every time the FIFO trigger level is reached, it is necessary to have the DMA channel burst transfer size match the UART FIFO trigger level. For example, if the UART FIFO trigger level is set to 8 bytes, the DMA channel burst size must be set to 8 double words. Note that although the DMA always transfers double words, the UART treats each double word request as a single byte request. Also, when allocating memory for the receive buffer, four bytes must be allocated for every UART data sample.

Figure 3-11 shows the DMA channel register contents for this transfer after the channel is enabled. The source address of the DMA channel is set to the receive buffer register (RBR) address for the UART, and the destination address is set to the start of the data block in internal memory. Since the address of RBR is fixed, the source address mode is set to 10b (constant address) and the destination address mode is set to 00b (automatic post-increment).

Note that in this example the destination address is set to the DMA byte address 000C E000h, which corresponds to SARAM block 31 .

For more information on DMA byte addresses, see Section 3.2.2.

Figure 3-10. Servicing Incoming UART Data Example

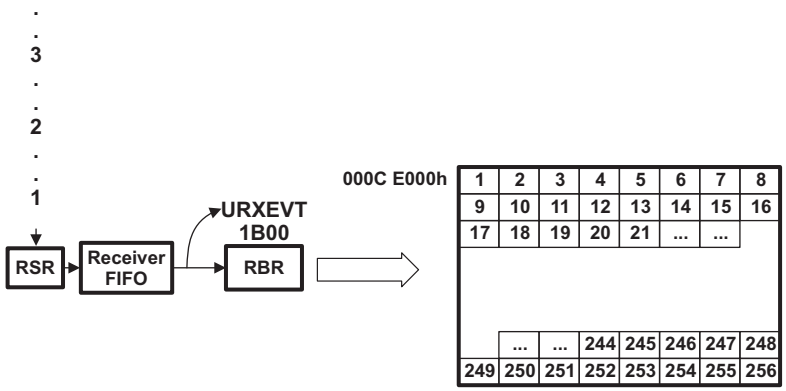


Figure 3-11. Servicing Incoming UART Data Example DMA Configuration

(a) DMA Register Contents

Register Contents	
0000h	1B00h
000Ch	E000h
F09Ch	0400h

→

Parameter	
DMACH*m*SSAU	DMACH*m*SSAL
DMACH*m*DSAU	DMACH*m*DSAL
DMACH*m*TCR2	DMACH*m*TCR1

(b) Channel Transfer Control Options

15	14	13	12	11 10	9 8
1	1	1	1	00	00
EN	STATUS	INTEN	AUTORLD	RSV	DSTAMODE

7 6	5 3	2	1	0
10	000	1	0	0
SRCAMODE	BURSTMODE	SYNCMODE	LAST_XFER	PING_PONG_EN

3.3.3 Ping-Pong DMA Example

The example here describes Ping-Pong transfer from an external codec (non-bursting peripheral) to the internal memory. Figure 3-12 shows DMA transfer of incoming data from the codec through the I2S0 in a Ping-Pong fashion. To transfer the incoming data stream to its destination in the internal memory, the DMA channel must be set up for a non-burst transfer with synchronization enabled. A receive synchronization event is generated for every data sample as it arrives; hence it is necessary to have the DMA transfer each data sample individually.

Figure 3-12 shows the DMA channel register contents for this transfer after the channel is enabled. The source address of the DMA channel is set to the left data0 receive register (I2S0RXLT0) address for I2S0, and the destination address is set to the start of the data block in internal memory. Since the address of I2S0RXLT0 is fixed, the source address mode is set to 10b (constant address) and the destination address mode is set to 00b (automatic post-increment). Note that in this example the destination address is set to the DMA byte address 000C E000h, which corresponds to SARAM block 31 at 0004 E000h and DMA transfer length is 1K bytes. For more information on DMA byte addresses, see Section 3.2.2. When allocating memory for the receive buffer, two 16-bit words must be allocated for every I2S data sample. It is also assumed that 1K bytes data buffer has been allocated at 000C E000h.

On every receive event generated by the I2S, the DMA will transfer a full double word from I2S0RXLT0 to the destination address. After the DMA has transferred the 128th sample, it sends an interrupt to the CPU, sets the LAST_XFER bit to 0 and continues the transfer. Once the 256th sample is transferred, the DMA controller again generates an interrupt to the CPU and sets the LAST_XFER bit to 1. Since the AUTORLD bit has been set to 1, the destination address is reloaded and the transfer resumes.

Figure 3-12. Servicing Incoming I2S Data Example in Ping-Pong DMA Mode

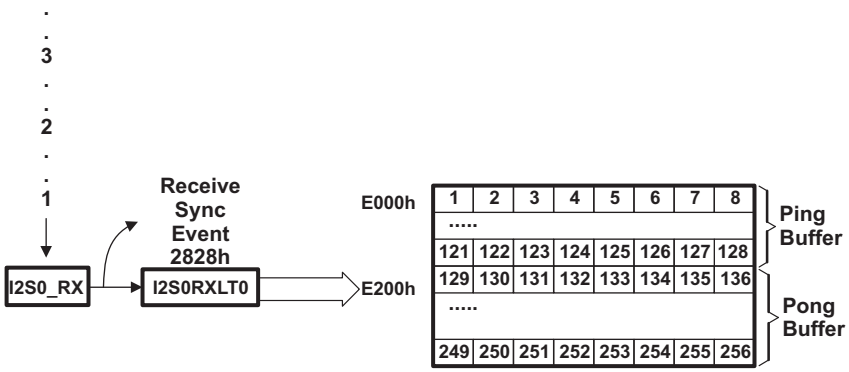


Figure 3-13. Servicing Incoming I2S Data Example DMA Configuration

(a) DMA Register Contents

Register Contents	
0000h	2828h
000Ch	E000h
F085h	0400h

→

Parameter	
DMACH*m*SSAU	DMACH*m*SSAL
DMACH*m*DSAU	DMACH*m*DSAL
DMACH*m*TCR2	DMACH*m*TCR1

(b) Channel Transfer Control Options

15	14	13	12	11 10	9 8
1	1	1	1	00	00
EN	STATUS	INTEN	AUTORLD	RSV	DSTAMODE

7 6	5 3	2	1	0
10	000	1	0	1
SRCAMODE	BURSTMODE	SYNCMODE	LAST_XFER	PING_PONG_EN

3.4 Registers

Table 3-4 through Table 3-8 list the memory-mapped registers associated with the four direct memory access (DMA) controllers. The DMA controller registers can be accessed by the CPU at the word addresses specified in each table. Note that the CPU accesses all peripheral registers through its I/O space. All other register addresses not listed in the tables below should be considered as reserved locations and the register contents should not be modified.

There are several other registers that affect the operation of the DMA controllers. The DMA interrupt flag and enable registers (DMAIFR and DMAIER) are used to control the interrupt generation of the four DMA controllers. In addition, there are two registers per DMA controller which control event synchronization in each channel—the DMA*n* channel event source registers (DMA*n*CESRL and DMA*n*CESRU). These registers are not part of the DMA controllers; they are part of the DSP system. For more information on these registers, see Chapter 1, *System Control*.

Table 3-4. System Registers Related to the DMA Controllers(1)

CPU Word Address	Acronym	Register Description
1C30h	DMAIFR	DMA Interrupt Flag Register
1C31h	DMAIER	DMA Interrupt Enable Register
1C1Ah	DMA0CESR1	DMA0 Channel Event Source Register 1
1C1Bh	DMA0CESR2	DMA0 Channel Event Source Register 2
1C1Ch	DMA1CESR1	DMA1 Channel Event Source Register 1
1C1Dh	DMA1CESR2	DMA1 Channel Event Source Register 2
1C36h	DMA2CESR1	DMA2 Channel Event Source Register 1
1C37h	DMA2CESR2	DMA2 Channel Event Source Register 2
1C38h	DMA3CESR1	DMA3 Channel Event Source Register 1
1C39h	DMA3CESR2	DMA3 Channel Event Source Register 2

(1) For more information on these registers, see Chapter 1, *System Control*.

Table 3-5. DMA Controller 0 (DMA0) Registers

CPU Word Address	Acronym	Register Description	Section
0C00h	DMACH0SSAL	Channel 0 Source Start Address Register Lower	Section 3.4.1
0C01h	DMACH0SSAU	Channel 0 Source Start Address Register Upper	Section 3.4.1
0C02h	DMACH0DSAL	Channel 0 Destination Start Address Register Lower	Section 3.4.2
0C03h	DMACH0DSAU	Channel 0 Destination Start Address Register Upper	Section 3.4.2
0C04h	DMACH0TCRL	Channel 0 Transfer Control Register Lower	Section 3.4.3
0C05h	DMACH0TCRU	Channel 0 Transfer Control Register Upper	Section 3.4.3
0C20h	DMACH1SSAL	Channel 1 Source Start Address Register Lower	Section 3.4.1
0C21h	DMACH1SSAU	Channel 1 Source Start Address Register Upper	Section 3.4.1
0C22h	DMACH1DSAL	Channel 1 Destination Start Address Register Lower	Section 3.4.2
0C23h	DMACH1DSAU	Channel 1 Destination Start Address Register Upper	Section 3.4.2
0C24h	DMACH1TCRL	Channel 1 Transfer Control Register Lower	Section 3.4.3
0C25h	DMACH1TCRU	Channel 1 Transfer Control Register Upper	Section 3.4.3
0C40h	DMACH2SSAL	Channel 2 Source Start Address Register Lower	Section 3.4.1
0C41h	DMACH2SSAU	Channel 2 Source Start Address Register Upper	Section 3.4.1
0C42h	DMACH2DSAL	Channel 2 Destination Start Address Register Lower	Section 3.4.2
0C43h	DMACH2DSAU	Channel 2 Destination Start Address Register Upper	Section 3.4.2
0C44h	DMACH2TCRL	Channel 2 Transfer Control Register Lower	Section 3.4.3
0C45h	DMACH2TCRU	Channel 2 Transfer Control Register Upper	Section 3.4.3
0C60h	DMACH3SSAL	Channel 3 Source Start Address Register Lower	Section 3.4.1
0C61h	DMACH3SSAU	Channel 3 Source Start Address Register Upper	Section 3.4.1

Table 3-5. DMA Controller 0 (DMA0) Registers (continued)

CPU Word Address	Acronym	Register Description	Section
0C62h	DMACH3DSAL	Channel 3 Destination Start Address Register Lower	Section 3.4.2
0C63h	DMACH3DSAU	Channel 3 Destination Start Address Register Upper	Section 3.4.2
0C64h	DMACH3TCRL	Channel 3 Transfer Control Register Lower	Section 3.4.3
0C65h	DMACH3TCRU	Channel 3 Transfer Control Register Upper	Section 3.4.3

Table 3-6. DMA Controller 1 (DMA1) Registers

CPU Word Address	Acronym	Register Description	Section
0D00h	DMACH0SSAL	Channel 0 Source Start Address Register Lower	Section 3.4.1
0D01h	DMACH0SSAU	Channel 0 Source Start Address Register Upper	Section 3.4.1
0D02h	DMACH0DSAL	Channel 0 Destination Start Address Register Lower	Section 3.4.2
0D03h	DMACH0DSAU	Channel 0 Destination Start Address Register Upper	Section 3.4.2
0D04h	DMACH0TCRL	Channel 0 Transfer Control Register Lower	Section 3.4.3
0D05h	DMACH0TCRU	Channel 0 Transfer Control Register Upper	Section 3.4.3
0D20h	DMACH1SSAL	Channel 1 Source Start Address Register Lower	Section 3.4.1
0D21h	DMACH1SSAU	Channel 1 Source Start Address Register Upper	Section 3.4.1
0D22h	DMACH1DSAL	Channel 1 Destination Start Address Register Lower	Section 3.4.2
0D23h	DMACH1DSAU	Channel 1 Destination Start Address Register Upper	Section 3.4.2
0D24h	DMACH1TCRL	Channel 1 Transfer Control Register Lower	Section 3.4.3
0D25h	DMACH1TCRU	Channel 1 Transfer Control Register Upper	Section 3.4.3
0D40h	DMACH2SSAL	Channel 2 Source Start Address Register Lower	Section 3.4.1
0D41h	DMACH2SSAU	Channel 2 Source Start Address Register Upper	Section 3.4.1
0D42h	DMACH2DSAL	Channel 2 Destination Start Address Register Lower	Section 3.4.2
0D43h	DMACH2DSAU	Channel 2 Destination Start Address Register Upper	Section 3.4.2
0D44h	DMACH2TCRL	Channel 2 Transfer Control Register Lower	Section 3.4.3
0D45h	DMACH2TCRU	Channel 2 Transfer Control Register Upper	Section 3.4.3
0D60h	DMACH3SSAL	Channel 3 Source Start Address Register Lower	Section 3.4.1
0D61h	DMACH3SSAU	Channel 3 Source Start Address Register Upper	Section 3.4.1
0D62h	DMACH3DSAL	Channel 3 Destination Start Address Register Lower	Section 3.4.2
0D63h	DMACH3DSAU	Channel 3 Destination Start Address Register Upper	Section 3.4.2
0D64h	DMACH3TCRL	Channel 3 Transfer Control Register Lower	Section 3.4.3
0D65h	DMACH3TCRU	Channel 3 Transfer Control Register Upper	Section 3.4.3

Table 3-7. DMA Controller 2 (DMA2) Registers

CPU Word Address	Acronym	Register Description	Section
0E00h	DMACH0SSAL	Channel 0 Source Start Address Register Lower	Section 3.4.1
0E01h	DMACH0SSAU	Channel 0 Source Start Address Register Upper	Section 3.4.1
0E02h	DMACH0DSAL	Channel 0 Destination Start Address Register Lower	Section 3.4.2
0E03h	DMACH0DSAU	Channel 0 Destination Start Address Register Upper	Section 3.4.2
0E04h	DMACH0TCRL	Channel 0 Transfer Control Register Lower	Section 3.4.3
0E05h	DMACH0TCRU	Channel 0 Transfer Control Register Upper	Section 3.4.3
0E20h	DMACH1SSAL	Channel 1 Source Start Address Register Lower	Section 3.4.1
0E21h	DMACH1SSAU	Channel 1 Source Start Address Register Upper	Section 3.4.1
0E22h	DMACH1DSAL	Channel 1 Destination Start Address Register Lower	Section 3.4.2
0E23h	DMACH1DSAU	Channel 1 Destination Start Address Register Upper	Section 3.4.2

Table 3-7. DMA Controller 2 (DMA2) Registers (continued)

CPU Word Address	Acronym	Register Description	Section
0E24h	DMACH1TCRL	Channel 1 Transfer Control Register Lower	Section 3.4.3
0E25h	DMACH1TCRU	Channel 1 Transfer Control Register Upper	Section 3.4.3
0E40h	DMACH2SSAL	Channel 2 Source Start Address Register Lower	Section 3.4.1
0E41h	DMACH2SSAU	Channel 2 Source Start Address Register Upper	Section 3.4.1
0E42h	DMACH2DSAL	Channel 2 Destination Start Address Register Lower	Section 3.4.2
0E43h	DMACH2DSAU	Channel 2 Destination Start Address Register Upper	Section 3.4.2
0E44h	DMACH2TCRL	Channel 2 Transfer Control Register Lower	Section 3.4.3
0E45h	DMACH2TCRU	Channel 2 Transfer Control Register Upper	Section 3.4.3
0E60h	DMACH3SSAL	Channel 3 Source Start Address Register Lower	Section 3.4.1
0E61h	DMACH3SSAU	Channel 3 Source Start Address Register Upper	Section 3.4.1
0E62h	DMACH3DSAL	Channel 3 Destination Start Address Register Lower	Section 3.4.2
0E63h	DMACH3DSAU	Channel 3 Destination Start Address Register Upper	Section 3.4.2
0E64h	DMACH3TCRL	Channel 3 Transfer Control Register Lower	Section 3.4.3
0E65h	DMACH3TCRU	Channel 3 Transfer Control Register Upper	Section 3.4.3

Table 3-8. DMA Controller 3 (DMA3) Registers

CPU Word Address	Acronym	Register Description	Section
0F00h	DMACH0SSAL	Channel 0 Source Start Address Register Lower	Section 3.4.1
0F01h	DMACH0SSAU	Channel 0 Source Start Address Register Upper	Section 3.4.1
0F02h	DMACH0DSAL	Channel 0 Destination Start Address Register Lower	Section 3.4.2
0F03h	DMACH0DSAU	Channel 0 Destination Start Address Register Upper	Section 3.4.1
0F04h	DMACH0TCRL	Channel 0 Transfer Control Register Lower	Section 3.4.3
0F05h	DMACH0TCRU	Channel 0 Transfer Control Register Upper	Section 3.4.3
0F20h	DMACH1SSAL	Channel 1 Source Start Address Register Lower	Section 3.4.1
0F21h	DMACH1SSAU	Channel 1 Source Start Address Register Upper	Section 3.4.1
0F22h	DMACH1DSAL	Channel 1 Destination Start Address Register Lower	Section 3.4.2
0F23h	DMACH1DSAU	Channel 1 Destination Start Address Register Upper	Section 3.4.2
0F24h	DMACH1TCRL	Channel 1 Transfer Control Register Lower	Section 3.4.3
0F25h	DMACH1TCRU	Channel 1 Transfer Control Register Upper	Section 3.4.3
0F40h	DMACH2SSAL	Channel 2 Source Start Address Register Lower	Section 3.4.1
0F41h	DMACH2SSAU	Channel 2 Source Start Address Register Upper	Section 3.4.1
0F42h	DMACH2DSAL	Channel 2 Destination Start Address Register Lower	Section 3.4.2
0F43h	DMACH2DSAU	Channel 2 Destination Start Address Register Upper	Section 3.4.2
0F44h	DMACH2TCRL	Channel 2 Transfer Control Register Lower	Section 3.4.3
0F45h	DMACH2TCRU	Channel 2 Transfer Control Register Upper	Section 3.4.3
0F60h	DMACH3SSAL	Channel 3 Source Start Address Register Lower	Section 3.4.1
0F61h	DMACH3SSAU	Channel 3 Source Start Address Register Upper	Section 3.4.1
0F62h	DMACH3DSAL	Channel 3 Destination Start Address Register Lower	Section 3.4.2
0F63h	DMACH3DSAU	Channel 3 Destination Start Address Register Upper	Section 3.4.2
0F64h	DMACH3TCRL	Channel 3 Transfer Control Register Lower	Section 3.4.3
0F65h	DMACH3TCRU	Channel 3 Transfer Control Register Upper	Section 3.4.3

3.4.1 Source Start Address Registers (DMACHmSSAL and DMACHmSSAU)

Each channel has two source start address registers, which are shown in Figure 3-14 and Figure 3-15 and described in Table 3-9 and Table 3-10. For the first access to the source port of the channel, the DMA controller generates a byte address by concatenating the contents of the two I/O-mapped registers. DMACH*m*SSAU supplies the upper bits, and DMACH*m*SSAL supplies the lower bits:

Source start address = DMACH*m*SSAU:DMACH*m*SSAL

The channel updates the source start address registers every time it is serviced the DMA controller. The amount of data transferred each time the channel is serviced is specified by the BURSTMODE bits of DMACH*m*TCRU.

The destination start address is supplied by DMACH*m*DSAL and DMACH*m*DSAU, which are described in Section 3.4.2.

NOTE:

1. You must load the source start address registers with a byte address. For more details, see Section 3.2.4.
2. There are four DMA controllers in the DSP, although all DMA controllers can access DARAM and SARAM, each DMA controller can only access a subset of on-chip peripherals. Also, only DMA controller 3 has access to external memory. For more details, see Section 3.2.2.
3. All data buffers in on-chip or off-chip memory should be aligned on an even boundary. For more information on managing memory, see the *TMS320C55x Assembly Language Tools User Guide* (SPRU280).

Figure 3-14. Source Start Address Register - Lower Part (DMACH*m*SSAL)

15	0
SSAL	
RW-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 3-15. Source Start Address Register - Upper Part (DMACH*m*SSAU)

15	0
SSAU	
RW-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-9. Source Start Address Register - Lower Part (DMACH*m*SSAL) Field Description

Bit	Field	Type	Reset	Description
15-0	SSAL	RW	0	Lower part of source start address (byte address). Value is 0 to FFFFh.

Table 3-10. Source Start Address Register - Upper Part (DMACH*m*SSAU) Field Description

Bit	Field	Type	Reset	Description
15-0	SSAU	RW	0	Upper part of source start address (byte address). Value is 0 to FFFFh.

3.4.2 Destination Start Address Registers (DMACHmDSAL and DMACHmDSAU)

Each channel has two destination start address registers, which are shown in Figure 3-16 as well as Figure 3-17 and described in Table 3-11 and Table 3-12. For the first access to the destination port of the channel, the DMA controller generates a byte address by concatenating the contents of the two I/O-mapped registers. DMACH*m*DSAU supplies the upper bits, and DMACH*m*DSAL supplies the lower bits:

Destination start address = DMACH*m*DSAU:DMACH*m*DSAL

The channel updates the source start address registers every time it is serviced the DMA controller. The amount of data transferred each time the channel is serviced is specified by the BURSTMODE bits of DMACH*m*TCRU.

The source start address is supplied by DMACH*m*SSAL and DMACH*m*SSAU, which are described in Section 3.4.1.

NOTE:

1. You must load the source start address registers with a byte address. For more details, see Section 3.2.4.
2. There are four DMA controllers in the DSP, although all DMA controllers can access DARAM and SRAM, each DMA controller can only access a subset of on-chip peripherals. Also, only DMA controller 3 has access to external memory. For more details, see Section 3.2.2.
3. All data buffers in on-chip or off-chip memory should be aligned on an even boundary. For more information on managing memory, see the *TMS320C55x Assembly Language Tools User Guide* (SPRU280).

Figure 3-16. Destination Start Address Register - Lower Part (DMACH*m*DSAL)

15	0
DSAL	
RW-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 3-17. Destination Start Address Register - Upper Part (DMACH*m*DSAU)

15	0
DSAU	
RW-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-11. DMA Destination Start Address Register - Lower Part (DMACH*m*DSAL) Field Description

Bit	Field	Type	Reset	Description
15-0	DSAL	RW	0	Lower part of destination start address (byte address). Value is 0 to FFFFh.

Table 3-12. DMA Destination Start Address Register - Upper Part (DMACH*m*DSAU) Field Description

Bit	Field	Type	Reset	Description
15-0	DSAU	RW	0	Upper part of destination start address (byte address). Value is 0 to FFFFh.

3.4.3 Transfer Control Registers (DMACHmTCRL and DMACHmTCRU)

Each channel has two transfer control registers shown in Figure 3-18 and Figure 3-19. These I/O-mapped register enables you to specify the size of the transfer, enable event synchronization and auto-initialization, select the source and destination addressing mode, and specify the burst mode of the channel. You can also monitor the status of the DMA channel through these registers. Table 3-13 and Table 3-14 describes the fields of these registers.

CAUTION

When using synchronization events, you must set EN = 1 and SYNCMODE = 1 during the same write cycle to DMACH*m*TCRU. The DMA channel will transfer the first data value when it receives the synchronization event specified by CH*n*EVT in DMACESRL and DMACESRU. Also, when disabling the channel, you must set EN = 0 and SYNCMODE = 0 during the same write cycle to DMACH*m*TCRU.

The amount of data (in bytes) to be transferred as programmed in the LENGTH field in DMACH*m*TCRL should be a multiple of 4 bytes x $2^{BURSTMODE}$ field in DMACH*m*TCRU, that is, LENGTH = (4 x $2^{BURSTMODE}$) bytes.

For an explanation of this workaround, see the *TMS320C5517 Fixed-Point DSP Silicon Errata* [literature number: SPRZ383].

Figure 3-18. Transfer Control Register Lower (DMACH*m*TCRL)

15	0
LENGTH	
RW-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 3-19. Transfer Control Register Upper (DMACH*m*TCRU)

15	14	13	12	11	10	9	8
EN	STATUS	INTEN	AUTORLD	RSV		DSTAMODE	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0		R/W-0	

7	6	5	3	2	1	0
SRCAMODE		BURSTMODE		SYNCMODE	LAST_XFER	PING_PONG_EN
R/W-0		R/W-0		R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-13. Transfer Control Register Lower (DMACH*m*TCRL) Field Description

Bit	Field	Type	Reset	Description
15-0	LENGTH	RW	0	Size of transfer. When Ping-Pong mode is enabled, this is the size of the **Ping** and the **Pong** transfer combined. Value 0000h to 0003h: Reserved, do not use. Value 0004h to FFFCh: These bits specify the number of double words (specified in bytes) to be transferred in multiples of (4 x $2^{BURSTMODE}$) bytes. **Note:** These bits are not updated by the channel each time it is serviced by the DMA controller.

Table 3-14. Transfer Control Register Upper (DMACH*m*TCRU) Field Descriptions

Bit	Field	Type	Reset	Description
15	EN	R/W	0	Channel enable bit. Use EN to enable or disable transfers in the channel. The DMA controller clears EN once a block transfer in the channel is complete. 0 = The channel is disabled. the channel cannot be serviced by the DMA controller. IF DMA burst transfer is ongoing in the channel, the DMA controller allows the burst transfer to complete. **Note:** When you write a 0 to this bit, you must also write a 0 to the SYNCMODE bit. 1 = The channel is enabled. the channel can be serviced by the DMA in the next available time slot.
14	STATUS	R/W	0	Channel status bit. This bit indicates the status o the DMA channel transfer. The DMA controller sets the channel STATUS bit to 1 if: - A nonzero value is written on LENGTH in DMACH*m*TCRL - A write access is performed to DMACH*m*TCRU and LENGTH has a nonzero value. The DMA controller clears the STATUS bit to 0 if: - All the bytes specified by LENGTH in DMACH*m*TCRL have been transferred. - A value of 0 is written to LENGTH in DMACH*m*TCRL 0 = Corresponding DMA channel has transferred all the bytes specified by LENGTH in DMACH*m*TCRL. 1 = Corresponding DMA channel has not finished transferring all the bytes specified by LENGTH in DMACH*m*TCRL.
13	INTEN	R/W	0	CPU interrupt enable bit. The DMA channel is capable of generating a CPU interrupt when a block transfer is finished. In order for the CPU to receive the interrupt, the corresponding channel interrupt mask bit in the interrupt mask register (DMAIMR) must be set to 1. 0 = Disable channel interrupt. 1 = Enabled channel interrupt.
12	AUTORLD	R/W	0	Automatic reload bit. Once a transfer is finished, the DMA automatically reloads the transfer control register and the source and destination start address registers and restarts the transfer. **Note:** Automatic reload can only be used when SYNCMODE = 1. 0 = DMA transfer does not automatically reload. 1 = Upon completion of a full transfer, the registers are reloaded and the transfer is restarted.
11-10	Reserved	R/W	0	Reserved, always write zeroes to these bits.
9-8	DSTAMODE	R/W	0	Destination addressing mode bits. DSTAMODE determines the addressing mode used by the DMA controller when it writes to the destination address. 0 = Automatic post increment. The destination byte address is incremented by four each transfer. 1h = Reserved, do not use. 2h = Constant address. 3h = Reserved, do not use.
7-6	SRAMODE	R/W	0	Source addressing mode bits. SRCAMODE determines the addressing mode used by the DMA controller when it reads from the source address. 0 = Automatic post increment. The source byte address is incremented by four after each transfer. 1h = Reserved, do not use. 2h = Constant address. 3h = Reserved, do not use.
5-3	BURSTMODE	R/W	0	Burst mode bits. These bits specify the number of double word transfers that each channel performs at once before the DMA controller moves on to the active channel. **Note:** The burst mode selected must always be less than or equal to the number of bytes specified in DMACH*m*TCRL. 0 = 1 double word (4 bytes). 1h = 2 double words (8 bytes). 2h = 4 double words (16 bytes). 3h = 8 double words (32 bytes). 4h = 16 double words (64 bytes). 5h to 7h = Reserved, do not use.

Table 3-14. Transfer Control Register Upper (DMACH*m*TCRU) Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	SYNCMODE	R/W	0	Synchronization mode bit. CH*n*EVT bits in DMACESR*n* determine which event in the DSP (for example, a timer countdown) initiates a DMA transfer in the channel. Multiple channels can have the same SYNCEVT value; in other words, one synchronization event can initiate activity in multiple channels. On each sync event, the DMA transfers the number of double words specified by the BURSTMODE bits. For example, BURSTMODE = 010b, the DMA will transfer a total of 4 double words per sync event. A DSP reset selects SYNCMODE = 0 (no synchronization). When SYNCMODE = 0, the DMA controller does not wait for a synchronization event before beginning a DMA transfer in the channel; channel activity begins as soon as the channel is enabled (EN = 1). 0 = When synchronization is disabled, the DMA controller does not wait for a synchronization event before beginning a DMA transfer. **Note:** When you write a 0 to the EN bit, you must also write a 0 to this bit. 1 = Activity in the DMA controller is synchronized to the event specified in the CH*n*EVT bits of DMACESR*n*. **Note:** When you set this bit to 1, you must also write a 1 to the EN bit.
1	LAST_XFER	R/W	0	Indicates whether the most recent completed transfer was the **Ping** buffer or the **Pong** buffer. This status bit is only valid when the PING_PONG_EN bit is set to 1. 0 = The last completed transfer was the **Ping** buffer 1 = The last completed transfer was the **Pong** buffer
0	PING_PONG_EN	R/W	0	Enable Ping-Pong DMA transfer mode. 0 = Ping-Pong mode is disabled 1 = Ping-Pong mode is enabled

External Memory Interface (EMIF)

The following sections describe the operation of the External Memory Interface (EMIF) in the Digital Signal Processor (DSP).

Topic **Page**

4.1 Introduction

4.1.1 Purpose of the External Memory Interface

The purpose of the EMIF is to provide a means to connect to a variety of external asynchronous devices including:

- NOR Flash, NAND Flash, and SRAM
- Mobile single data rate (SDR) SDRAM and SDRAM devices

NOTE: Non-mobile SDRAM can be supported under certain circumstances. This device always uses mobile SDRAM initialization but is able to support SDRAM memories that ignore the BA0 and BA1 pins for the *load mode register* command. During the mobile SDRAM initialization, the device issues the *load mode register* initialization command to two different addresses that differ in only the BA0 and BA1 address bits. These registers are the Extended Mode register and the Mode register. The extended mode register exists only in mSDRAM and not in non-mSDRAM. If a non-mobile SDRAM memory ignores bits BA0 and BA1, the second loaded register value overwrites the first, leaving the desired value in the mode register and the non-mobile SDRAM will work with the device.

Section 4.3 contains examples of connecting the EMIF to mobile SDRAM devices.

4.1.2 Features

The EMIF has the following features:

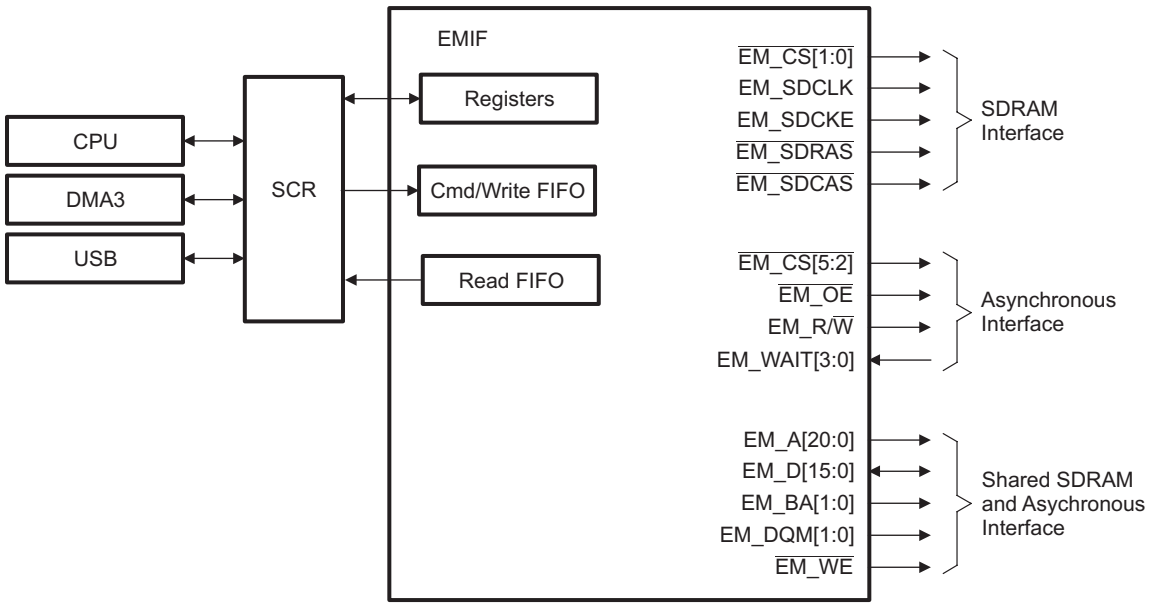
- Supports asynchronous devices (e.g., RAM, ROM, NOR Flash).
  - Up to 8MB asynchronous address range over 4 chip selects.
  - Supports 8- and 16-bit data bus widths.
  - Programmable cycle timings for each chip select.
  - Page mode for NOR Flash.
  - Supports extended wait cycles.
  - Supports select strobe mode.
- Supports NAND Flash on 4 asynchronous chip selects.
  - Supports 8 and 16-bit data bus widths.
  - Programmable cycle timings for each chip select.
  - Supports 1-bit ECC for 8 and 16-bit NAND Flash.
  - Supports 4-bit ECC for 8-bit and 16-bit NAND Flash.
  - Does not perform error correction.

- Supports mobile SDR SDRAM devices.
  - 8MB SDRAM address range over two chip select spaces.
  - 16-bit data bus width.
  - Supports CAS latencies of 2 and 3 (4 not supported).
  - Supports 1, 2, and 4 internal banks.
  - Supports 256, 512, 1024, 2048-word page sizes.
  - Supports 4 and 8 burst lengths (1 and 2 not supported).
  - Supports sequential burst type (interleave burst type not supported).
  - SDRAM auto initialization from reset or configuration change.
  - Bank interleaving across both the chip selects.
  - Self refresh and pre-charge power down modes for low power.
  - Partial array self-refresh and temperature controlled self refresh modes for low power in mobile SDR.
    - Temperature controlled self-refresh is only supported for mobile SDR having on-chip temperature sensor.
  - Supports prioritized refresh.
  - Programmable SDRAM refresh rate and backlog counter.
  - Programmable SDRAM timing parameters.
  - Auto pre-charge not supported for better bank interleaving performance.
  - Clock Suspend not supported for SDR SDRAM.

4.1.3 Functional Block Diagram

Figure 4-1 illustrates a high-level view of the EMIF and its connections within the device. The CPU and DMA controller access the EMIF through a switched central resource. Section 4.2.3 describes the EMIF external pins and summarizes their purpose when interfacing with external devices.

Figure 4-1. EMIF Functional Block Diagram



4.2 Architecture

This section provides details about the architecture and operation of the EMIF. Both SDRAM and asynchronous interface are covered, along with other system-related topics such as clock control and pin multiplexing.

4.2.1 Clock Control

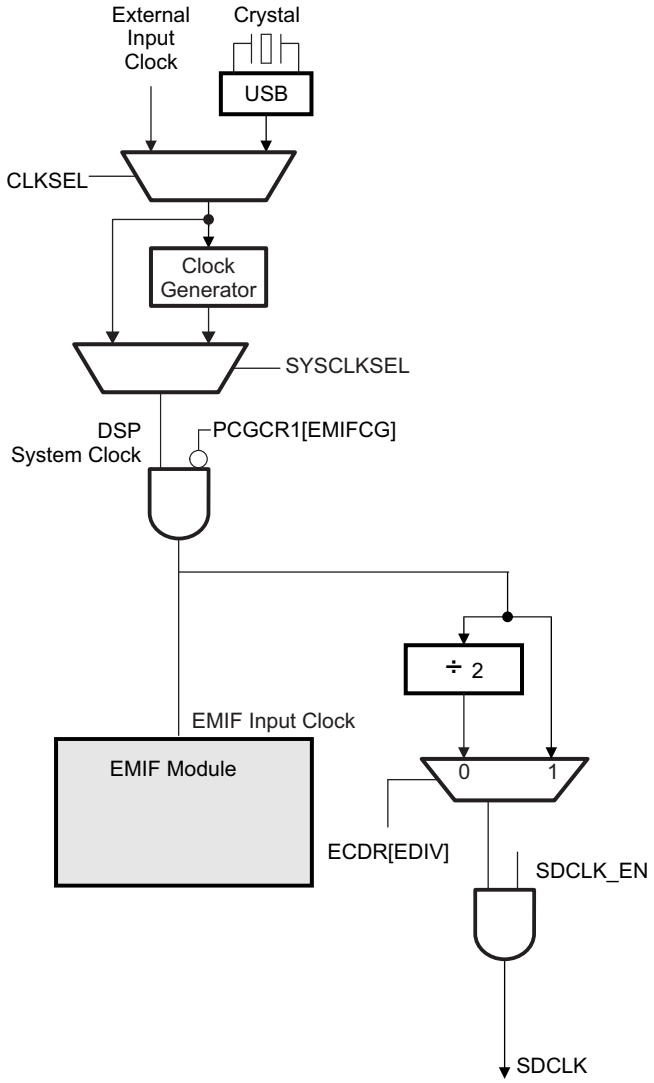
As shown in Figure 4-2, the clock generator receives either the USB oscillator or a signal from an external clock source and produces the DSP system clock. This clock is used by the DSP CPU and peripherals. The EMIF input clock is used to source the interface clock in synchronous mode and to generate the access cycles in asynchronous mode. The SDCLK can be generated from the EMIF CLK or a divide by two version of the EMIF CLK through EMIF Clock Divide Register (ECDR) register.

This device has limitations to the clock frequency on the EM_SDCLK pin (driven by SDCLK) based on the CV_{DD} and DV_{DDEMIF}.

- When CV_{DD} = 1.3 V, and DV_{DDEMIF} = 3.3 V or 2.75 V, the maximum clock frequency on the EM_SDCLK pin is limited to 100 MHz (EM_SDCLK ≤ 100 MHz). Therefore, if SYSCLK ≤ 100 MHz, the EM_SDCLK can be configured either as SYSCLK or SYSCLK/2, but if SYSCLK > 100 MHz, the EM_SDCLK must be configured as SYSCLK/2.
- When CV_{DD} = 1.05 V, and DV_{DDEMIF} = 3.3 V or 2.75 V, the maximum clock frequency on the EM_SDCLK pin is limited to 75 MHz (EM_SDCLK ≤ 75 MHz). Therefore, if SYSCLK ≤ 75 MHz, the EM_SDCLK can be configured as either SYSCLK or SYSCLK/2, but if SYSCLK > 75 MHz, the EM_SDCLK must be configured as SYSCLK/2.
- When DV_{DDEMIF} = 1.8 V, regardless of the CV_{DD} voltage, the clock frequency on the EM_SDCLK pin must be configured as SYSCLK/2 and ≤ 50 MHz.

The device includes logic which can be used to gate the clock to its on-chip peripherals, including the EMIF. The input clock to the EMIF can be enabled and disabled through the peripheral clock gating configuration register 1 (PCGCR1).

Figure 4-2. Clocking Diagram for the EMIF



4.2.2 EMIF Requests

Different modules within the DSP can make requests to the EMIF. These requests consist of accesses to SDRAM memory, asynchronous memory, and EMIF registers. Because the EMIF can process only one request at a time, a switch central resource (SCR) exists within the DSP to provide prioritized requests from the different sources to the EMIF.

1. CPU (peripheral register access)
2. CPU (data access)
3. CPU (instruction fetch)
4. DMA Controller 3
5. USB

If a request is submitted to the EMIF from two or more sources simultaneously, the SCR will arbitrate between the difference sources using a round-robin approach. Upon completion of a request, the SCR again evaluates the pending requests and forwards the next pending request to the EMIF.

When the EMIF receives a request, it may or may not be immediately processed. In some cases, the EMIF will perform one or more auto refresh cycles before processing the request. For details on the internal arbitration of the EMIF between performing requests and performing auto refresh cycles, see Section 4.2.10.

CAUTION

The EMIF does not support constant addressing mode. All request serviced by the EMIF must use a linear (incrementing) addressing mode.

4.2.3 Memory Map

External memory is divided into several chip select spaces. On the device CPU, DMA controller 3 (DMA3) and USB can access external memory. The starting address for each of the chip select spaces in external memory is different from the point-of-view of these modules. The memory map as seen by these modules are shown in the following table.

Table 4-1. EMIF Memory Map

DSP Memory Map	Size (Bytes)	CPU Start *Word* Address	DMA3 Start *Byte* Address	USB Start *Byte* Address
EMIF CS0	8M (1)	02 8000h	0100 0000h	0100 0000h
EMIF CS2	4M	40 0000h	0200 0000h	0200 0000h
EMIF CS3	2M	60 0000h	0300 0000h	0300 0000h
EMIF CS4	1M	70 0000h	0400 0000h	0400 0000h
EMIF CS5 (2)	1M	78 0000h	0500 0000h	0500 0000h

(1) This is an approximate value.
(2) When MP/MC = 0 the upper 128K bytes of EMIF CS5 is used for ROM data.

4.2.4 Signal Descriptions

Table 4-2 describes the EMIF pins that are used to interface to external devices.

Table 4-2. EMIF Pins Used to Access Both SDRAM and Asynchronous Devices

Pin(s)	Type	Description
EM_A[20:0]	Output	EMIF address bus. When interfacing to an SDRAM device, these pins are primarily used to provide the row and column address to the SDRAM. When interfacing to an asynchronous device, these pins are used in conjunction with the EM_BA[1:0] pins to form the address that is sent to the device.
EM_D[15:0]	Input/Output	EMIF data bus.
EM_BA[1:0]	Output	When interfacing to an SDRAM device, these pins are used to provide the bank address inputs to the SDRAM. When interfacing to an asynchronous device, these pins are used in conjunction with the EM_A[20:0] pins to form the address that is sent to the device.
$\overline{\text{EM\_DQM[1:0]}}$	Output	Active-low byte enables. When interfacing to SDRAM, these pins are connected to the $\overline{\text{DQM}}$ pins of the SDRAM to individually enable/disable each of the bytes in a data access. When interfacing to an asynchronous device, these pins are connected to byte enables.
$\overline{\text{EM\_WE}}$	Output	Active-low write enable. When interfacing to SDRAM, this pin is connected to the WE pin of the SDRAM and is used to send commands to the device. When interfacing to an asynchronous device, this pin provides a signal that is active-low during the strobe period of an asynchronous write access cycle.

Table 4-3. EMIF Pins Specific to SDRAM

Pin(s)	Type	Description
EM_CS[1:0]	Output	Active-low chip select pin for SDRAM devices. This pin is connected to the chip-select pin of the attached SDRAM device and is used for enabling/disabling commands. By default, the EMIF keeps this SDRAM chip select active, even if the EMIF is not interfaced with an SDRAM device. This pin is deactivated when accessing the asynchronous memory bank and is reactivated on completion of the asynchronous assess.
EM_SDRAS	Output	Active-low row address strobe pin. This pin is connected to the RAS pin of the attached SDRAM device and is used for sending commands to the device.
EM_SDCAS	Output	Active-low column address strobe pin. This pin is connected to the CAS pin of the attached SDRAM device and is used for sending commands to the device.
EM_SDCKE	Output	Clock enable pin. This pin is connected to the CKE pin of the attached SDRAM device and is used for issuing the SELF REFRESH command which places the device in self refresh mode.
EM_SDCLK	Output	SDRAM clock pin. This pin is connected to the CLK pin of the attached SDRAM device.

Table 4-4. EMIF Pins Specific to Asynchronous Devices

Pin(s)	Type	Description
EM_CS[5:2]	Output	Active-low chip select pins for asynchronous devices. These pins are meant to be connected to the chip-select pins of the attached asynchronous device. These pins are active only during accesses to the asynchronous memory.
EM_OE	Output	Active-low pin enable for asynchronous devices. This pin provides a signal which is active-low during the strobe period of an asynchronous read access cycle.
EM_R/W	Output	Active-low read/write select pin. This pin is high for the duration of an asynchronous read access cycle and low for the duration of an asynchronous write cycle.
EM_WAIT[3:0]	Input	Wait input with programmable polarity / NAND Flash ready input. An asynchronous device can extend the strobe period of an access cycle by asserting the wait input pins. When connected to NAND flash devices these pins function as NAND Flash ready inputs.

4.2.5 Pin Multiplexing

The EMIF address pins EM_A[20:15] are multiplexed with GPIO pins GPIO[26:21]. The external bus selection register (EBSR) controls the functionality of these pins. For more details on this register, see the device-specific data manual.

4.2.6 SDRAM Controller and Interface

The EMIF can gluelessly interface to mobile SDR SDRAM devices and supports such features as self refresh mode and prioritized refresh. In addition, it provides flexibility through programmable parameters such as the refresh rate, CAS latency, and many SDRAM timing parameters. The following sections include details on how to interface and properly configure the EMIF to perform read and write operations to externally connected SDR SDRAM devices. The SDCLK_EN bit must be set in the Clock Configuration register 1 (CCR1) 0x1C1E to turn on the SDRAM clock.

Non-mobile SDRAM can be supported under certain circumstances. This device always uses mobile SDRAM initialization but is able to support SDRAM memories that ignore the BA0 and BA1 pins for the *load mode register* command. During the mobile SDRAM initialization, the device issues the *load mode register* initialization command to two different addresses that differ in only the BA0 and BA1 address bits. These registers are the Extended Mode register and the Mode register. The extended mode register exists only in mSDRAM and not in non-mSDRAM. If a non-mobile SDRAM memory ignores bits BA0 and BA1, the second loaded register value overwrites the first, leaving the desired value in the mode register and the non-mobile SDRAM will work with device.

4.2.6.1 SDRAM Commands

The EMIF supports the SDRAM commands described in Table 4-5. Table 4-6 shows the truth table for the SDRAM commands, and an example timing waveform of the PRE command is shown in Figure 4-3. The address pin EM_A[10] is pulled low in this example to deactivate only the bank specified by the EM_BA[1:0] pins.

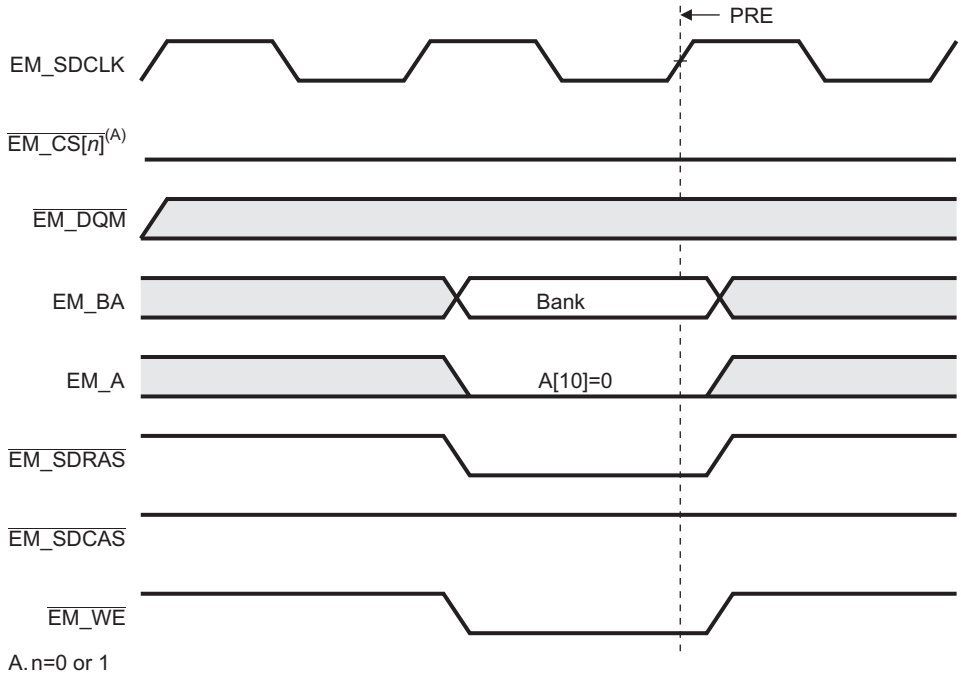
Table 4-5. EMIF SDRAM Commands

Command	Description
PRE	Pre-charge command. Depending on the value of EM_A[10], the PRE command either deactivates the open row in all banks (EM_A[10] = 1) or only the bank specified by the EM_BA[1:0] pins (EM_A[10] = 0).
ACTV	Activate command. The ACTV command activates the selected row in a particular bank for the current access.
READ	Read command. The READ command outputs the starting column address and signals the SDRAM to begin the burst read operation. Address EM_A[10] is always pulled low to avoid auto pre-charge. This allows for better bank interleaving performance.
WRT	Write command. The WRT command outputs the starting column address and signals the SDRAM to begin the burst write operation. Address EM_A[10] is always pulled low to avoid auto pre-charge. This allows for better bank interleaving performance.
BT	Burst-terminate command. The BT command is used to truncate the current read or write burst request.
LMR	Load mode register command. The LMR command sets the mode register of the attached SDRAM devices and is only issued during the SDRAM initialization sequence described in Section 4.2.6.4.
REFR	Auto-refresh command. The REFR command signals the SDRAM to perform an auto refresh according to its internal address.
SLFR	Self-refresh command. The self refresh command places the SDRAM into self-refresh mode, during which it provides its own clock signal and auto refresh cycles.
NOP	No operation command. The NOP command is issued during all cycles in which one of the above commands is not issued.
POWER DOWN	Power-down command. The POWER DOWN command signals the SDRAM to deactivate its input and output buffers, excluding CKE. The EMIF closes (pre-charge) all internal banks of the SDRAM prior to issuing the POWER DOWN command, therefore, the EMIF only supports pre-charge power down.

Table 4-6. Truth Table for SDRAM Commands

SDRAM Pins:	CKE	$\overline{CS}$	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$	BA[1:0]	A[12:11]	A[10]	A[9:0]
EMIF Pins:	EM_SDCKE	$\overline{EM\_CS[n]}$	$\overline{EM\_SDRAS}$	$\overline{EM\_SDCAS}$	$\overline{EM\_WE}$	EM_BA[1:0]	EM_A[12:11]	EM_A[10]	EM_A[9:0]
PRE	H	L	L	H	L	Bank/X	X	L/H	X
ACTV	H	L	L	H	H	Bank	Row	Row	Row
READ	H	L	H	L	H	Bank	Column	L	Column
WRT	H	L	H	L	L	Bank	Column	L	Column
BT	H	L	H	H	L	X	X	X	X
LMR	H	L	L	L	L	X	Mode	Mode	Mode
REFR	H	L	L	L	H	X	X	X	X
SLFR	L	L	L	L	H	X	X	X	X
NOP	H	L	H	H	H	X	X	X	X

Figure 4-3. Timing Waveform for SDRAM PRE Command



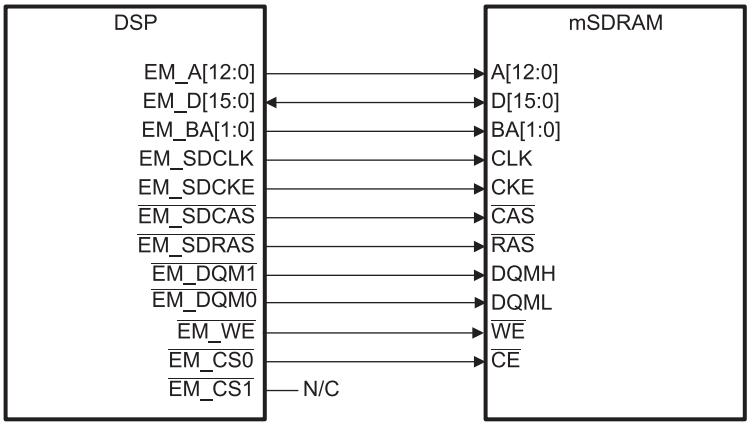
4.2.6.2 Interfacing to Mobile SDRAM

The EMIF supports a glueless interface to mobile SDRAM devices with the following characteristics:

- Pre-charge bit is A[10]
- The number of column address bits is 8, 9,10, or 11
- The number of row address bits is less than or equal to 14
- The number of internal banks is 1, 2, or 4

Figure 4-4 shows an interface between the EMIF and a single mobile SDRAM device. The mobile SDRAM device can be connected to either $\overline{EM\_CS0}$ or $\overline{EM\_CS1}$, however the proper external memory address must be used to activate the correct chip select pin. For more details, see Section 4.2.6.11. Note that when operated in SDRAM mode, the EMIF bus is always 16 bits wide.

Figure 4-4. EMIF to mSDRAM Connection



4.2.6.3 SDRAM Configuration Registers

The operation of the EMIF's SDRAM interface is controlled by programming the appropriate configuration registers. This section describes the purpose and function of each configuration register, but should be referenced for a more detailed description of each register, including the default registers values and bit-field positions. The following tables list the main SDRAM configuration registers, along with a description of each of their programmable fields.

NOTE: Writing to SDCR1 or the lower byte of SDCR2 causes the EMIF to abandon whatever it is currently doing and trigger the SDRAM initialization procedure described in Section 4.2.6.4 .

Table 4-7. Description of Bit Fields in SDRAM Configuration Registers (SDCR1 and SDCR2)

Parameter	Description
SR	This bit controls entering and exiting of the Self-Refresh mode. The field should be written using a byte-write to the upper byte of SDCR2 to avoid triggering the SDRAM initialization sequence.
PD	This bit controls entering and exiting of the Power down mode. The field should be written using a byte-write to the upper byte of SDCR2 to avoid triggering the SDRAM initialization sequence. If both SR and PD bits are set, the EMIF will go into Self Refresh. See Section 4.2.6.4
PDWR	Perform refreshes during Power Down. Writing a 1 to this bit will cause the EMIF to exit the power down state and issue an AUTO REFRESH command every time Refresh May level is set.
CL	**CAS latency.** This field defines the number of clock cycles between when an SDRAM issues a READ command and when the first piece of data appears on the bus. The value in this field is sent to the attached SDRAM device via the LOAD MODE REGISTER command during the SDRAM initialization procedure as described in Section 4.2.6.4. Only values of 2h (CAS latency = 2) and 3h (CAS latency = 3) are supported and should be written to this field. A 1 must be simultaneously written to the BIT11_9LOCK bit field of SDCR1 in order to write to the CL bit field.
IBANK	**Number of Internal SDRAM Banks.** This field defines the number of banks inside the attached SDRAM devices in the following way: • When IBANK = 0, 1 internal bank is used • When IBANK = 1h, 2 internal banks are used • When IBANK = 2h, 4 internal banks are used This field value affects the mapping of logical addresses to SDRAM row, column, and bank addresses.
PAGESIZE	**Page Size.** This field defines the internal page size of the attached SDRAM devices in the following way: • When PAGESIZE = 0, 256-word pages are used • When PAGESIZE = 1h, 512-word pages are used • When PAGESIZE = 2h, 1024-word pages are used • When PAGESIZE = 3h, 2048-word pages are used This field value affects the mapping of logical addresses to SDRAM row, column, and bank addresses.

Table 4-8. Description of Bit Fields in SDRAM Refresh Control Register (SDRCR)

Parameter	Description
REFRATE	Refresh Rate. This field controls the rate at which attached SDRAM devices will be refreshed. The equation below can be used to determine the required value of REFRATE for an SDRAM device: REFRATE <= f_{EM_SDCLK} / (Required SDRAM Refresh Rate). More information about the operation of the SDRAM refresh controller can be found in Section 4.2.6.6.

Table 4-9. Description of Bit Fields in SDRAM Timing Registers (SDTIMR1 and SDTIMR2)

Parameter	Description
T_RFC T_RP T_RCD T_WR T_RAS T_RC T_RRD	SDRAM Timing Parameters. These fields configure the EMIF to comply with the AC timing constraints and to more efficiently schedule its operations. More details about each of these parameters can be found in the register description in section 4. These parameters should be set to satisfy the corresponding timing requirements found in the SDRAM's datasheet.

Table 4-10. Description of Bit Fields in SDRAM Self Refresh Exit Timing Register (SDSRETR)

Parameter	Description
T_XS	Self Refresh Exit Parameter. The T_XS field of this register informs the EMIF about the minimum number of EM_SDCLK cycles required between exiting Self Refresh and issuing any command. This parameter should be set to satisfy the t_{XSR} value for the attached SDRAM device.

4.2.6.4 Mobile SDRAM Auto-Initialization Sequence

The EMIF automatically performs an mobile SDRAM initialization sequence, regardless of whether it is interfaced to an SDRAM device. No memory accesses to the SDRAM and asynchronous interfaces are performed until this auto-initialization is complete. The auto-initialization sequence is performed by the EMIF when either of the following two events occur:

- The EMIF is taken out of reset. See Section 4.2.12 for more information on EMIF reset.
- A write is performed to SDRAM configuration register 1 (SDCR1) or the least significant byte of SDRAM configuration register 2 (SDCR2). In either of these cases the EMIF starts from step 2 below, without waiting for the eight SDRAM refresh intervals.

An SDRAM initialization sequence consists of the following steps:

1. The EMIF drives EM_SDCKE high and begins continuously issuing NOP commands.
2. After eight SDRAM refresh intervals, the EMIF issues a PRE command with EM_A[10] held high to indicate all banks. The SDRAM refresh interval is as defined by the REFRATE field in the SDRAM refresh control register (SDRCR).
3. After eight REFR commands, the EMIF issues an LMR command to the extended mode register with the settings as described in Table 4-11. (This step is needed for mobile SDRAM; because the chip does not have a pin to select if mobile SDRAM or SDRAM is used, this command is always sent during initialization. Therefore, SDRAM support is limited to memories that can handle this extra initialization command.)
4. The EMIF issues another LMR command, this time to the mode register with the settings as described in Table 4-12.
5. Finally, the EMIF performs an auto refresh cycle (see Section 4.2.6.6).

Table 4-11. Extended Mode Register Settings During Auto-Init Sequence

Mode Register Bit	Mode Register Field	Init Value	Description
15-7	Reserved	00h	Reserved
6-5	Drive Strength	SDRAM_DRIVE	Sets the drive strength of the memory data pins. The setting for these bits is taken from the SDRAM_DRIVE bits of the SDCR2 register.
4-3	Internal Temperature Compensated Self-Refresh	0h	Specifies the maximum case temperature for self-refresh interval adjustment.
2-0	Partial Array Self-Refresh	PASR	Specifies the amount of memory that will be refreshed during a SLFR command. The setting for these bits is taken from the PASR bits of the SDCR2 register.

Table 4-12. Mode Register Settings During Auto-Init Sequence

Mode Register Bit	Mode Register Field	Init Value	Description
15:10	Reserved	00h	Reserved.
9	Write Burst Mode	0h	Specifies the write burst mode. This bit is set to 0 to indicate the burst type setting specified by bits 2-0 applies to both reads and writes.
8-7	Standard Operating Mode	0h	Selects standard operating mode.
6-4	CAS Latency	CL	Specifies the CAS latency. The value for these bits is taken from the CL bits of the SDCR1 register.
3	Burst Type	0h	Specifies the burst type. This bit is set to 0 to indicate sequential bursts.
2-0	Burst Length	3h	Specifies the burst length. These bits are set to 3 to indicate a burst length of 8.

4.2.6.5 SDRAM Configuration Procedure

The EMIF automatically performs the mSDRAM initialization sequence described in Section 4.2.6.4. However, you must follow the procedure listed below before performing any SDRAM requests.

The steps in the initialization procedure are as follows:

1. Turn SD clock on by writing 0x0001 to **Clock Configuration Register 1 (CCR1) [0x1C1Eh]**.
2. Program SDTIMR1, SDTIMR2, and SDSRETR to satisfy the timing requirements for the attached SDRAM device. The timing parameters should be taken from the SDRAM datasheet.
3. Program the REFRATE field of SDRCR such that the following equation is satisfied: (REFRATE x 8) / (f_{EM_SDCLK}) > 100 microseconds or 200 microseconds, depending on the power-up constraint specified by the SDRAM memory. The SDRAM power-up constraint specifies that 200 microseconds (sometimes 100 microseconds) should exist between receiving stable VDD and CLK and the issuing of a PRE command. For example, an EM_SDCLK frequency of 100 MHz would require setting REFRATE to 1251 (4E3h) or higher to meet a 100 microseconds constraint.
4. Program SDCR1 and SDCR2 to match the characteristics of the attached SDRAM device. This will cause the auto-initialization sequence in Section 4.2.6.4 to be re-run with the new value of REFRATE.
5. Perform a read from the SDRAM to guarantee that step 3 will occur after the initialization process has completed. Alternatively, wait for 200 microseconds.
6. Finally, program the REFRATE field to match that of the attached device's refresh interval. See Section 4.2.6.6.2 for details on determining the appropriate value.

After following the above procedure, the EMIF is ready to perform accesses to the attached SDRAM device. See Section 4.3 for an example of configuring the SDRAM interface.

NOTE: You must follow the steps in Section 4.2.13 before starting the steps outlined in this section.

4.2.6.6 SDRAM Refresh Controller

An SDRAM device requires that each of its rows be refreshed at a minimum required rate. The EMIF can meet this constraint by performing auto refresh cycles at or above this required rate. An auto refresh cycle consists of issuing a PRE command to all banks of the SDRAM device followed by issuing a REFR command. To inform the EMIF of the required rate for performing auto refresh cycles, the REFRATE field of the SDRAM Refresh Control Register (SDRCR) must be programmed. The EMIF will use this value along with two internal counters to automatically perform auto refresh cycles at the required rate. The auto refresh cycles cannot be disabled, even if the EMIF is not interfaced with an SDRAM. The remainder of this section details the EMIF's refresh scheme and provides an example for determining the appropriate value to place in the REFRATE field of SDRCR.

4.2.6.6.1 EMIF Refresh Scheme

The EMIF uses two counters to schedule REFR command: a 13-bit decrementing refresh interval counter and a 4-bit refresh backlog counter. The interval counter is loaded with the REFRATE field value at reset. The interval counter decrements by one each cycle until it reaches zero at which point it reloads from REFRATE and restarts decrementing. The counter also reloads and restarts decrementing whenever the REFRATE field is updated.

The refresh backlog counter records the number of REFR commands the EMIF currently has outstanding. The backlog counter increments by one each time the interval counter reloads (unless it has reached its maximum value of 15). The backlog counter decrements by one each time the EMIF issues a REFR command (unless it is already at zero).

The EMIF uses the refresh backlog counter to determine the urgency with which an auto refresh cycle should be performed. The four levels of urgency are described in Table 4-13. This refresh scheme allows the required refreshes to be performed with minimal impact on memory access requests.

Table 4-13. Refresh Urgency Levels

Urgency Level	Refresh Backlog Counter Range	Action Taken
Refresh May	1-3	An auto-refresh cycle is performed only if the EMIF has no requests pending and none of the SDRAM banks are open.
Refresh Release	4-7	An auto-refresh cycle is performed if the EMIF has no requests pending, regardless of whether any SDRAM banks are open.
Refresh Need	8-11	An auto-refresh cycle is performed at the completion of the current access unless there are read requests pending.
Refresh Must	12-15	Multiple auto-refresh cycles are performed at the completion of the current access until the Refresh Release urgency level is reached. At that point, the EMIF can begin servicing any new read or write requests.

The refresh counters do not operate when SDRAM has been put into self-refresh mode.

The EMIF issues REFR commands within auto refresh cycles. An auto refresh cycle consists of issuing an REFR command and waiting T_RFC (see the SDRAM timing register 2) cycles before re-checking the refresh levels. If the Refresh Release level is not reached, the EMIF starts another auto refresh cycle, otherwise it returns to the idle state.

4.2.6.6.2 Determining the Appropriate Value for the REFRATE Field

The value that should be programmed into the refresh-rate (REFRATE) field of SDRCR can be calculated by using the frequency of the EM_SDCLK signal (f_{EM_SDCLK}) and the required refresh rate of the SDRAM ($f_{Refresh}$).

The following formula can be used to calculate REFRATE:

REFRATE $\leq f_{EM\_SDCLK} / f_{Refresh}$

The SDRAM datasheet often communicates the required SDRAM Refresh Rate in terms of the number of REFR commands required in a given time interval. The required SDRAM Refresh Rate in the formula above can be therefore be calculated by dividing the number of required cycles per time interval (n_{cycles}) by the time interval given in the datasheet ($t_{Refresh\ Period}$):

$f_{Refresh} = n_{cycles} / t_{Refresh\ Period}$

Combining these formulas, the value that should be programmed into the REFRATE field can be computed as:

$REFRATE \leq f_{EM\_SDCLK} \times t_{Refresh\ Period} / n_{cycles}$

The following example illustrates calculating the value of REFRATE. Given that:

- f_{EM_SDCLK} = 100 MHz
- $t_{Refresh\ Period}$ = 64 ms (required refresh interval of the SDRAM)
- n_{cycles} = 8192 (number of cycles in a refresh interval for the SDRAM)

REFRATE can be calculated as:

REFRATE ≤ 100 MHz × 64 ms/8192

REFRATE ≤ 781.25

REFRATE = 782 cycles = 30Eh cycles

4.2.6.7 Self Refresh Mode

To reduce the power consumption of the SDRAM device, you can request the EMIF to issue a SLRF command by setting the SR bit of SDCR2 to 1. When this bit is set, the EMIF will continue normal operation until all outstanding memory access requests have been serviced and the SDRAM refresh backlog has been cleared.

NOTE: The SR bit should be set and cleared using a byte-write to the upper byte of the SDCR2 to avoid triggering the SDRAM initialization sequence. The BYTEMODE bits in the EMIF system control register can be used to limit writes to EMIF registers to the upper or lower byte. See Section 4.2.8 for more details.

While in the self-refresh state, the EMIF continues to service asynchronous bank requests and register accesses as normal with one caveat: the EMIF will not park the data bus following a read to asynchronous memory while in the self-refresh state; instead, the EMIF tri-states the data bus. Therefore, it is not recommended to perform asynchronous read operations while the EMIF is in the self-refresh state in order to prevent floating inputs on the data bus. More information about data bus parking can be found in Section 4.2.9.

The EMIF will come out of the self-refresh state if the SR bit is cleared or an SDRAM access is requested while in the self-refresh state and T_RAS + 1 cycles have elapsed since the SLFR command was issued. The value of T_RAS is taken from SDRAM timing register 2. The EMIF exits from the self-refresh state by driving EM_SDCKE high, waiting T_XS + 1 cycles, and performing an auto refresh cycle. The value for T_XS is taken from the SDRAM self refresh exit timing register.

While in self-refresh mode, the SDRAM device consumes a minimal amount of power while performing its own refresh cycles. The SDRAM device should also be placed into self-refresh mode when changing the frequency of EM_SDCLK. If the frequency of EM_SDCLK changes while the SDRAM is not in self-refresh mode, the memory must be re-initialized following the procedure described in Section 4.2.6.5.

To use partial array self-refresh for mobile SDRAM, the PASR bits in the SDCR2 register must be appropriately programmed. The EMIF performs bank interleaving when IBANK_POS = 0 in SDCR2. Since the SDRAM is partially refreshed during partial array self refresh, it is recommended that IBANK_POS be set to 1 for software ease. If IBANK_POS is set to 0, it is the responsibility of software to move critical data into the banks that are going to be refreshed during partial array self-refresh.

4.2.6.8 Power Down Mode

To support low power modes, the EMIF can be requested to issue a POWER DOWN command to the SDRAM by setting the PD bit in SDCR2. When this bit is set, the EMIF will continue normal operation until all outstanding memory access requests have been serviced and the SDRAM refresh backlog has been cleared. At this point the EMIF will enter the power down state.

Upon entering this state, the EMIF will issue a POWER DOWN command (same as a NOP command but driving EM_SDCKE low on the same cycle). The EMIF then maintains EM_SDCKE low until it exits the power down state.

Since the EMIF services the refresh backlog before it enters the power down state, all internal banks of the SDRAM are closed (pre-charged) prior to issuing the POWER DOWN command. Therefore, the EMIF only supports pre-charge power down. The EMIF does not support active power down where internal banks of the SDRAM are open (active) before the POWER DOWN command is issued.

During the power down state, the EMIF services the SDRAM, asynchronous memory, and register accesses as normal, returning to the power down state upon completion.

The PDWR bit in the SDRAM configuration register indicates whether the EMIF should perform refreshes in power down state. If the PDWR bit is set, the EMIF exits power down state every time the Refresh Must level is reached, it performs REFR commands to the SDRAM, and then it returns back to the power down state. This evenly distributes the refreshes to the SDRAM in power down state. If the PDWR bit is not set, the EMIF does not perform any refreshes to the SDRAM. Therefore the data integrity of the SDRAM is not guaranteed upon power down exit if the PDWR bit is not set.

If the PD bit is cleared while in the power down state, the EMIF will come out of the power down state. To exit the power down state the EMIF drives EM_SDCKE high and enters its idle state.

4.2.6.9 SDRAM Read Operation

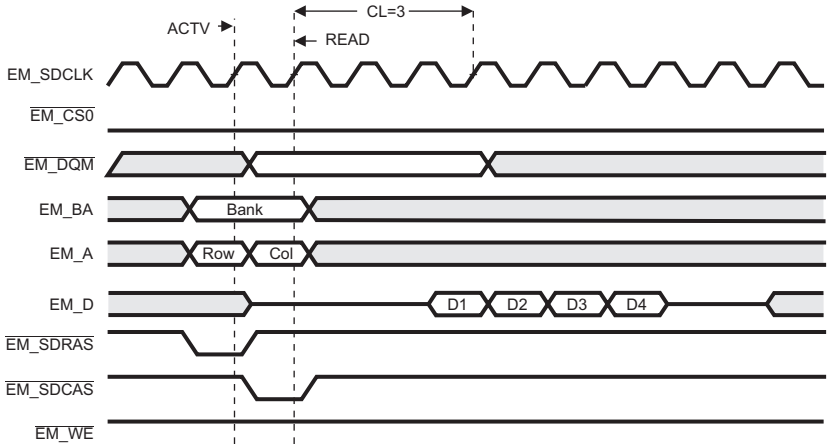
When the EMIF receives a read request to SDRAM, it performs one or more read access cycles. A read access cycle begins with the issuing of the ACTV command to select the desired bank and row of the SDRAM device. After the row has been opened, the EMIF proceeds to issue a READ command while specifying the desired bank and column address. The address pin EM_A[10] is held low during the READ command to avoid auto-pre-charging. The READ command signals the SDRAM device to start bursting data from the specified address while the EMIF issues NOP commands. Following a READ command, the CL field of SDCR1 defines how many delay cycles will be present before the read data appears on the data bus. This is referred to as the CAS latency.

Figure 4-5 shows the signal waveforms for a basic SDRAM read operation in which a burst of data is read from a single page. A burst size of eight is always used for SDRAM accesses.

The EMIF will truncate a series of bursting data if the remaining addresses of the burst are not required to complete the request. The EMIF can truncate the burst in three ways:

- By issuing another READ to the same page in the same bank.
- By issuing a PRE command in order to prepare for accessing a different page of the same bank.
- By issuing a BT command in order to prepare for accessing a page in a different bank.

Figure 4-5. Timing Waveform for Basic SDRAM Read Operation



Several other pins are also active during a read access. The EM_DQM[1:0] pins are driven low during the READ commands and are kept low during the NOP commands that correspond to the burst request. The state of the other EMIF pins during each command can be found in Table 4-6.

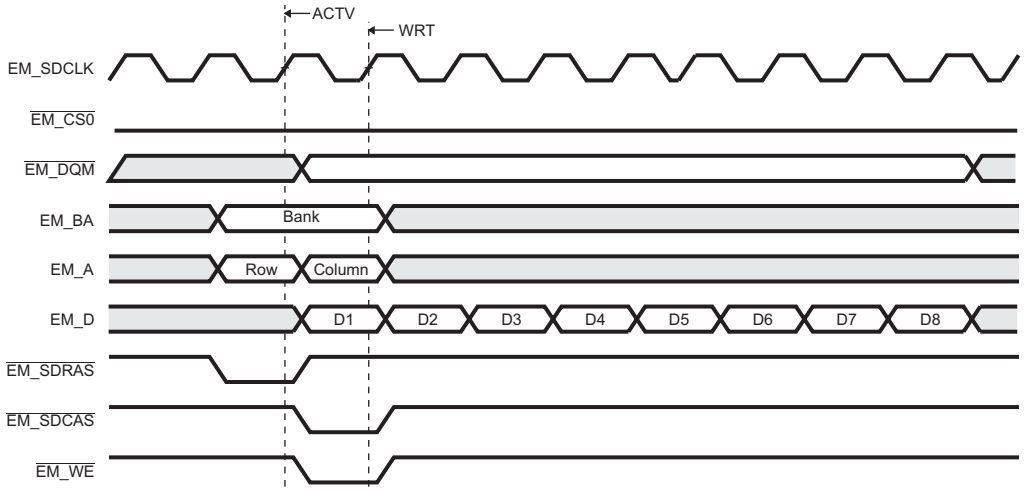
The EMIF schedules its commands based on the timing information that is provided to it in the SDRAM timing registers (SDTIMR1 and SDTIMR2). The values for the timing parameters in these registers should be chosen to satisfy the timing requirements listed in the SDRAM datasheet. The EMIF uses this timing information to avoid violating any timing constraints related to issuing commands. This is commonly accomplished by inserting NOP commands between various commands during an access. For more details on the various timing parameters, see the register description of SDTIMR1 and SDTIMR2 in .

4.2.6.10 SDRAM Write Operation

When the EMIF receives a write request to SDRAM it performs one or more write-access cycles. A write-access cycle begins with the issuing of the ACTV command to select the desired bank and row of the SDRAM device. After the row has been opened, the EMIF proceeds to issue a WRT command while specifying the desired bank and column address. The address pin EM_A[10] is held low during the WRT command to avoid automatic pre-charging. The WRT command signals the SDRAM device to start writing a burst of data to the specified address while the EMIF issues NOP commands. The associated write data will be placed on the data bus in the cycle concurrent with the WRT command and with subsequent burst continuation NOP commands.

Figure 4-6 shows the signal waveforms for a basic SDRAM write operation in which a burst of data is written to a single page. A burst size of eight is always used for SDRAM accesses.

Figure 4-6. Timing Waveform for Basic SDRAM Write Operation



The EMIF will truncate a series of bursting data if the remaining addresses of the burst are not part of the write request. The EMIF can truncate the burst in three ways:

- By issuing another WRT to the same page.
- By issuing a PRE command in order to prepare for accessing a different page of the same bank.
- By issuing a BT command in order to prepare for accessing a page in a different bank.

Several other pins are also active during a write access. The EM_DQM pins are driven to select which bytes of the data word will be written to the SDRAM device. They are also used to mask out entire undesired data words during a burst access. The state of the other EMIF pins during each command can be found in Table 4-6.

The EMIF schedules its commands based on the timing information that is provided to it in the SDRAM timing registers (SDTIMR1 and SDTIMR2). The values for the timing parameters in these registers should be chosen to satisfy the timing requirements listed in the SDRAM datasheet. The EMIF uses this timing information to avoid violating any timing constraints related to issuing commands. This is commonly accomplished by inserting NOP commands during various cycles of an access. For more details on the various timing parameters, see the register description of SDTIMR1 and SDTIMR2 in .

4.2.6.11 Mapping from Logical Address to SDRAM Address

The EMIF interleaves the internal banks for SDRAM connected to both the chip selects. From the system point of view, the external SDRAM is seen as one block of SDRAM. If smaller devices are used, the memory is seen to rollover.

When addressing SDRAM, if the IBANK_POS bit in SDCR2 is set to 0 (recommended), the EMIF uses the three fields IBANK, EBANK and PAGESIZE in SDCR1 to determine the mapping from source address to SDRAM row, column, bank and chip select. If the IBANK_POS bit is set to 1, the EMIF uses the four fields IBANK, EBANK, PAGESIZE, and ROWSIZE in SDCR1 and SDCR2 to determine the mapping from source address to SDRAM row, column, bank, and chip select. In all cases the EMIF considers its SDRAM address space to be a single logical block regardless of the number of physical devices or whether the devices are mapped across one or two EMIF chip selects.

For IBANK_POS = 0, Table 4-14 shows which logical address bits map to the SDRAM row, column, bank and chip select bits for all combinations of IBANK, EBANK, and PAGESIZE. For IBANK_POS = 1, Table 4-15 shows which logical address bits map to the SDRAM row, column, bank, and chip select bits for all combinations of IBANK, EBANK, PAGESIZE, and ROWSIZE.

Table 4-14. Mapping of Logical Address to SDRAM Address (IBANK_POS = 0)[1] [2]

EBANK	IBANK	PAGESIZE	Logical Address								
			30:23:00	22:15	14	13	12	11	10	9	8:01
0	0	0	x	nrb=14							ncb=8
1	0	0	x	nrb=13						nce=1	ncb=8
0	1	0	x	nrb=13						nbb=1	ncb=8
1	1	0	x	nrb=12					nce=1	nbb=1	ncb=8
0	2	0	x	nrb=12					nbb=2		ncb=8
1	2	0	x	nrb=11				nce=1	nbb=2		ncb=8
0	0	1	x	nrb=13						ncb=9	
1	0	1	x	nrb=12					nce=1	ncb=9	
0	1	1	x	nrb=12					nbb=1	ncb=9	
1	1	1	x	nrb=11				nce=1	nbb=1	ncb=9	
0	2	1	x	nrb=11				nbb=2		ncb=9	
1	2	1	x	nrb=10			nce=1	nbb=2		ncb=9	
0	0	2	x	nrb=12					ncb=10		
1	0	2	x	nrb=11				nce=1	ncb=10		
0	1	2	x	nrb=11				nbb=1	ncb=10		
1	1	2	x	nrb=10			nce=1	nbb=1	ncb=10		
0	2	2	x	nrb=10			nbb=2		ncb=10		
1	2	2	x	nrb=9		nce=1	nbb=2		ncb=10		
0	0	3	x	nrb=11				ncb=11			
1	0	3	x	nrb=10			nce=1	ncb=11			
0	1	3	x	nrb=10			nbb=1	ncb=11			
1	1	3	x	nrb=9		nce=1	nbb=1	ncb=11			
0	2	3	x	nrb=9		nbb=2		ncb=11			
1	2	3	x	nrb=8		nce=1	nbb=2	ncb=11			

(1) LEGEND: nrb = number of row address bits; ncb = number of column address bits; nbb = number of bank address bits; nce = number of chip select bits.

(2) Not all combinations of IBANK, ROWSIZE, EBANK, and PAGESIZE are valid. Table 4-14 shows only those combinations that are valid.

For IBANK_POS = 0, the effect of the above address-mapping scheme is that as the logical address increments across SDRAM page boundaries, the EMIF moves onto the same page in the next bank in the current device ($\overline{EM\_CS0}$). This movement along the banks of the current device continues until the page has been accessed in all banks in the current device. The EMIF then proceeds to the same page in the next device (if EBANK = 1, $\overline{EM\_CS1}$) and proceeds through the same page in all its banks before moving over to the next page in the first device ($\overline{EM\_CS0}$). The EMIF exploits this traversal across internal banks and chip selects while remaining on the same page to maximize the number of open SDRAM banks within the overall SDRAM space. Thus, the EMIF can keep a maximum of 8 banks (4 internal banks across 2 chip selects) open at a time.

Table 4-15. Mapping of Logical Address to SDRAM Address (IBANK_POS = 1) (1) (2)

IBANK	ROWSIZE	EBANK	PAGESIZE	Logical Address											
				30:23:00	22	21	20	19	18	17:13	12	11	10	9	8:01
0	0	0	0	x	x	x	x	x	x	nrb=9					ncb=8
0	0	1	0	x	x	x	x	x	nrb=9					nce=1	ncb=8
0	1	0	0	x	x	x	x	x	nrb=10						ncb=8
0	1	1	0	x	x	x	x	nrb=10						nce=1	ncb=8
0	2	0	0	x	x	x	x	nrb=11							ncb=8
0	2	1	0	x	x	x	nrb=11							nce=1	ncb=8
0	3	0	0	x	x	x	nrb=12								ncb=8
0	3	1	0	x	x	nrb=12								nce=1	ncb=8
0	4	0	0	x	x	nrb=13									ncb=8
0	4	1	0	x	nrb=13									nce=1	ncb=8
0	5	0	0	x	nrb=14										ncb=8
1	0	0	0	x	x	x	x	x	nbb=1	nrb=9					ncb=8
1	0	1	0	x	x	x	x	nbb=1	nrb=9					nce=1	ncb=8
1	1	0	0	x	x	x	x	nbb=1	nrb=10						ncb=8
1	1	1	0	x	x	x	nbb=1	nrb=10						nce=1	ncb=8
1	2	0	0	x	x	x	nbb=1	nrb=11							ncb=8
1	2	1	0	x	x	nbb=1	nrb=11							nce=1	ncb=8
1	3	0	0	x	x	nbb=1	nrb=12								ncb=8
1	3	1	0	x	nbb=1	nrb=12								nce=1	ncb=8
1	4	0	0	x	nbb=1	nrb=13									ncb=8
2	0	0	0	x	x	x	x	x	nbb=2	nrb=9					ncb=8
2	0	1	0	x	x	x	x	nbb=2	nrb=9					nce=1	ncb=8
2	1	0	0	x	x	x	x	nbb=2	nrb=10						ncb=8
2	1	1	0	x	x	x	nbb=2	nrb=10						nce=1	ncb=8
2	2	0	0	x	x	x	nbb=2	nrb=11							ncb=8
2	2	1	0	x	nbb=2	nrb=11								nce=1	ncb=8
2	3	0	0	x	nbb=2	nrb=12									ncb=8
0	0	0	1	x	x	x	x	x	nrb=9					ncb=9	
0	0	1	1	x	x	x	x	nrb=9					nce=1	ncb=9	
0	1	0	1	x	x	x	x	nrb=10						ncb=9	
0	1	1	1	x	x	x	nrb=10						nce=1	ncb=9	
0	2	0	1	x	x	x	nrb=11							ncb=9	
0	2	1	1	x	x	nrb=11							nce=1	ncb=9	
0	3	0	1	x	x	nrb=12								ncb=9	
0	3	1	1	x	nrb=12								nce=1	ncb=9	
0	4	0	1	x	nrb=13									ncb=9	
1	0	0	1	x	x	x	x	nbb=1	nrb=9					ncb=9	
1	0	1	1	x	x	x	nbb=1	nrb=9					nce=1	ncb=9	
1	1	0	1	x	x	x	nbb=1	nrb=10						ncb=9	
1	1	1	1	x	x	nbb=1	nrb=10						nce=1	ncb=9	
1	2	0	1	x	x	nbb=1	nrb=11							ncb=9	
1	2	1	1	x	nbb=1	nrb=11							nce=1	ncb=9	
1	3	0	1	x	nbb=1	nrb=12								ncb=9	
2	0	0	1	x	x	x	nbb=2	nrb=9						ncb=9	
2	0	1	1	x	x	nbb=2	nrb=9						nce=1	ncb=9	
2	1	0	1	x	x	nbb=2	nrb=10							ncb=9	
2	1	1	1	x	nbb=2	nrb=10							nce=1	ncb=9	
2	2	0	1	x	nbb=2	nrb=11								ncb=9	
0	0	0	2	x	x	x	x	nrb=9					ncb=10		
0	0	1	2	x	x	x	nrb=9					nce=1	ncb=10		
0	1	0	2	x	x	x	nrb=10						ncb=10		
0	1	1	2	x	x	nrb=10						nce=1	ncb=10		
0	2	0	2	x	x	nrb=11							ncb=10		

(1) LEGEND: nrb = number of row address bits; ncb = number of column address bits; nbb = number of bank address bits; nce = number of chip select bits.

(2) Not all combinations of IBANK, ROWSIZE, EBANK, and PAGESIZE are valid. Table 4-15 shows only those combinations that are valid.

Table 4-15. Mapping of Logical Address to SDRAM Address (IBANK_POS = 1) [(1)] [(2)] (continued)

IBANK	ROWSIZE	EBANK	PAGESIZE	Logical Address										
				30:23:00	22	21	20	19	18	17:13	12	11	10	9	8:01
0	2	1	2	x	nrb=11							nce=1	ncb=10		
0	3	0	2	x	nrb=12								ncb=10		
1	0	0	2	x	x	x	nbb=1	nrb=9					ncb=10		
1	0	1	2	x	x	nbb=1	nrb=9					nce=1	ncb=10		
1	1	0	2	x	x	nbb=1	nrb=10						ncb=10		
1	1	1	2	x	nbb=1	nrb=10						nce=1	ncb=10		
1	2	0	2	x	nbb=1	nrb=11							ncb=10		
2	0	0	2	x	x	nbb=2	nrb=9						ncb=10		
2	0	1	2	x	nbb=2	nrb=9						nce=1	ncb=10		
2	1	0	2	x	nbb=2	nrb=10							ncb=10		
0	0	0	3	x	x	x	nrb=9					ncb=11			
0	0	1	3	x	x	nrb=9					nce=1	ncb=11			
0	1	0	3	x	x	nrb=10						ncb=11			
0	1	1	3	x	nrb=10						nce=1	ncb=11			
0	2	0	3	x	nrb=11							ncb=11			
1	0	0	3	x	x	nbb=1	nrb=9					ncb=11			
1	0	1	3	x	nbb=1	nrb=9					nce=1	ncb=11			
1	1	0	3	x	nbb=1	nrb=10						ncb=11			
2	0	0	3	x	nbb=2	nrb=9						ncb=11			

For IBANK_POS = 1, the effect of the address-mapping scheme is that as the source address increments across SDRAM page boundaries, the EMIF moves onto the same page in the same bank in the next device (if EBANK = 1, $\overline{\text{EM\_CS1}}$). The EMIF then proceeds to the next page in the same bank in the first device ($\overline{\text{EM\_CS0}}$). This movement along the same banks of both devices continues until all the pages have been accessed in the same bank in both the devices. The EMIF then proceeds to the next bank in the first device ($\overline{\text{EM\_CS0}}$). Thus, the EMIF can keep a maximum of 2 banks across 2 chip selects open at a time. Since the EMIF can keep less number of banks open, this case is lower in performance than the IBANK_POS = 0 case. Thus this case is only recommended to be used along with Partial Array Self Refresh for mobile SDR SDRAM where performance can be traded off for power savings.

4.2.7 Asynchronous Controller and Interface

The EMIF easily interfaces to a variety of asynchronous devices including: NOR Flash, NAND Flash, and SRAM. It can be operated in two major modes (see Table 4-16):

- Normal Mode
- Select Strobe Mode

The default mode of operation is Normal Mode, in which the $\overline{\text{EM\_DQM}}$ pins of the EMIF function as byte enables. In this mode, the $\overline{\text{EM\_CS[5:2]}}$ pins behave as typical chip select signals, remaining active for the duration of the asynchronous access. See Section 4.2.7.1 for an example interface with multiple 8-bit devices.

In Select Strobe Mode the $\overline{\text{EM\_CS[5:2]}}$ pins act as a strobe-active only during the strobe period of an access. In this mode, the $\overline{\text{EM\_DQM}}$ pins of the EMIF function as standard byte enables for reads and writes.

A summary of the differences between the two modes of operation are shown in Table 4-16. For the details of asynchronous operations in Normal Mode, see Section 4.2.7.4 and for the details of asynchronous operations in Select Strobe Mode, see Section 4.2.7.5.

The EMIF hardware defaults to Normal Mode for each chip select space, but can be manually switched to Select Strobe Mode by setting the SS bit of the Asynchronous CSn Configuration Register 2 (ACSnCR2).

Table 4-16. Normal Mode vs. Select Strobe Mode

Mode	Function of $\overline{\text{EM\_DQM}}$ Pins	Operation of $\overline{\text{EM\_CS[5:2]}}$ Pins
Normal Mode	Byte enables	Active during the entire asynchronous access cycle
Select Strobe Mode	Byte enables	Active only during the strobe period of an access cycle

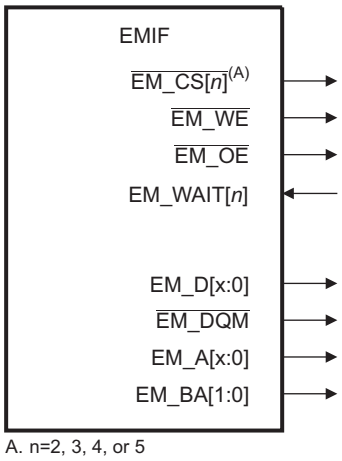
In both Normal Mode and Select Strobe Mode, the EMIF can be configured to operate in a sub-mode called NAND Flash Mode. In NAND Flash Mode, the EMIF is able to calculate an error correction code (ECC) for transfers up to 512 bytes.

The EMIF also provides configurable cycle timing parameters and an Extended Wait Mode that allows the connected device to extend the strobe period of an access cycle. The following sections describe the features related to interfacing with external asynchronous devices.

4.2.7.1 Interfacing to Asynchronous Memory

Figure 4-7 shows the EMIF's external pins used in interfacing with an asynchronous device. The pin $\overline{\text{EM\_CS[n]}}$ can be any of these chip select pins: $\overline{\text{EM\_CS[5:2]}}$.

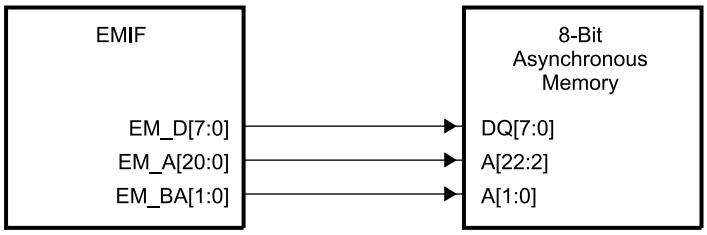
Figure 4-7. EMIF Asynchronous Interface



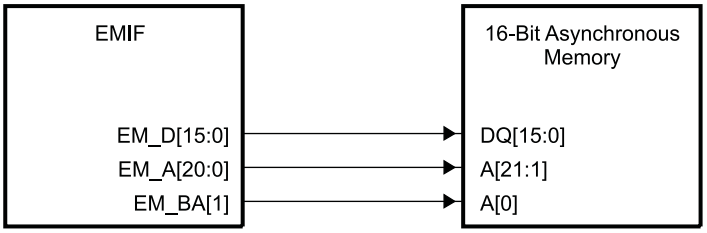
A. n=2, 3, 4, or 5

Of special note is the connection between the EMIF and the external device's address bus. The EMIF address pins EM_A[20:0] always provide the least significant bits of a double-word (32-bit) address. The EM_BA[1:0] pins provide word (16-bit) and byte selection functionality according to the data bus width configured in the Asynchronous CSn Configuration Register 1 (ACSnCR1). Figure 4-8 shows the mapping between the EMIF's and the connected device's data and address pins for an 8- and 16-bit data bus configuration.

Figure 4-8. Connecting Data and Address Bus to Asynchronous Memory Devices



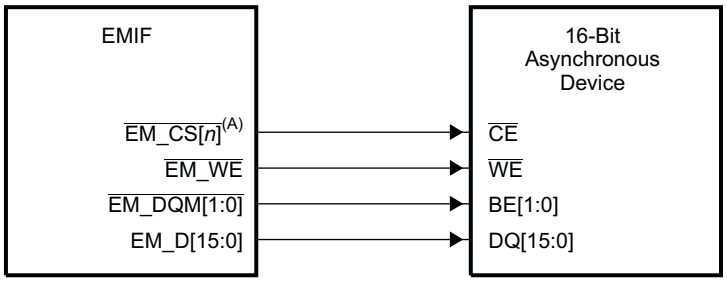
a) EMIF to 8-bit memory interface



b) EMIF to 16-bit memory interface

Figure 4-9 shows a common interface between the EMIF and an external asynchronous memory with byte enables. The EMIF should be operated in either Normal Mode or Select Strobe Mode when using this interface so that the $\overline{\text{EM\_DQM}}$ signals operate as byte enables.

Figure 4-9. Common Asynchronous Interface



A. n=2, 3, 4, or 5

4.2.7.2 Accessing Larger Asynchronous Memories

The external memory address space within the DSP's memory map and the number of dedicated EMIF address pins limits the maximum size of any asynchronous memory device connected to the EMIF. If an asynchronous memory device with a size larger than that allowed by these restrictions is needed, then GPIO pins may be used to control the upper address lines of the memory device.

An approach like this is useful in a system which requires that application code be loaded from slower/larger flash memory into faster/smaller internal or SDRAM memory. In this type of system, the DSP's ROM Bootloader can load a secondary Bootloader from flash. The secondary Bootloader can then finish the boot process by loading the actual application code.

When the ROM Bootloader copies the secondary Bootloader from the lower portion of the flash it does not need to manipulate the upper address lines. Only the secondary Bootloader, which is board-specific and is stored in the external flash, needs to know which GPIO pins have been assigned to the function of upper address lines. Therefore, the secondary Bootloader can perform the task of configuring the selected pins as GPIO and loading the remainder of the code from the upper flash memory.

The ROM Bootloader assumes that any GPIO pins used to control the upper address lines of the flash memory will be pulled to '0' after reset. This means that normally the GPIO pins selected for this function will be either spare or used as outputs only by the application, and therefore can be pulled to '0' at reset with an external pull-down resistor. The GPIO pins chosen should be tri-stated by default on device reset. For details on which GPIO-capable pins are tri-stated on device reset, see the device Data Manual.

4.2.7.3 Configuring the EMIF for Asynchronous Accesses

The operation of the EMIF's asynchronous interface can be configured by programming the appropriate register fields. The tables below list the register fields that can be programmed and describe the purpose of each field. These registers must be programmed prior to accessing the external memory. A transfer following a write to these registers will use the new configuration.

Note: provides the reset value and bit position for each register field. However, the Bootloader documentation should be consulted to determine if the fields are programmed during boot.

Table 4-17. Description of the Asynchronous CSn Configuration Registers (ACSnCR1 and ACSnCR2)

Parameter	Description
SS	**Select Strobe mode**. This bit selects the EMIF's mode of operation in the following way: • SS = 0h selects Normal Mode – $\overline{\text{EM\_DQM}}$ pins function as byte enables – $\overline{\text{EM\_CS}}$[5:2] pins are active for duration of access • SS = 1h selects Select Strobe Mode – $\overline{\text{EM\_DQM}}$ pins function as byte enables – $\overline{\text{EM\_CS}}$[5:2] pins acts as a strobe
EW	**Extended Wait Mode enable**. • EW = 0h disables Extended Wait Mode • EW = 1h enables Extended Wait Mode When set to 1, the EMIF enables its Extended Wait Mode in which the strobe width of an access cycle can be extended in response to the assertion of the wait pins (EM_WAIT[3:0]). The WPn bits in the Asynchronous Wait Cycle Configuration Register 2 (AWCCR2) controls to polarity of wait pins. **NOTE:** Extended Wait Mode should not be used while in NAND Flash Mode. See Section 4.2.7.6 for more details on this mode of operation
W_SETUP/R_SETUP	**Ready/Write setup widths**. These fields define the number of EMIF clock cycles of setup time for the address pins (EM_A[20:0] and EM_BA[1:0]), byte enables ($\overline{\text{EM\_DQM}}$[1:0]), and asynchronous chip select pins ($\overline{\text{EM\_CS}}$[5:2]) before the read strobe pin ($\overline{\text{EM\_OE}}$) or write strobe pin ($\overline{\text{EM\_WE}}$) falls, minus one cycle. For writes, the W_SETUP field also defines the setup time for the data pins (EM_D[15:0]). See the datasheet of the external asynchronous device to determine the appropriate setting for this field.
W_STROBE/R_STROBE	**Read/Write strobe widths**. These fields define the number of EMIF clock cycles between the falling and rising of the read strobe pin ($\overline{\text{EM\_OE}}$) or write strobe pin ($\overline{\text{EM\_WE}}$), minus one cycle. If Extended Wait Mode is enabled (EW = 1), these fields must be set to a value greater than zero. See the datasheet of the external asynchronous device to determine the appropriate setting for this field.
W_HOLD/R_HOLD	**Read/Write hold widths**. These fields define the number of EMIF clock cycles of hold time for the address pins (EM_A[20:0] and EM_BA[1:0]), byte enables ($\overline{\text{EM\_DQM}}$[1:0]), and asynchronous chip select pins ($\overline{\text{EM\_CS}}$[5:2]) after the read strobe pin ($\overline{\text{EM\_OE}}$) or write strobe pin ($\overline{\text{EM\_WE}}$) rises, minus one cycle. For writes, the W_HOLD field also defines the hold time for the data pins (EM_D[15:0]). See the datasheet of the external asynchronous device to determine the appropriate setting for this field.
TA	**Minimum turnaround time.** This field defines the minimum number of EMIF clock cycles between asynchronous reads and writes, minus one cycle. The purpose of this feature is to avoid contention on the bus. The value written to this field also determines the number of cycles that will be inserted between asynchronous accesses and SDRAM accesses. See the datasheet of the external asynchronous device to determine the appropriate setting for this field.

Table 4-17. Description of the Asynchronous CSn Configuration Registers (ACSnCR1 and ACSnCR2) (continued)

Parameter	Description
ASIZE	**Asynchronous device bus width.** This field determines the data bus width of the asynchronous interface in the following way: • ASIZE = 0h selects an 8-bit bus • ASIZE = 1h selects a 16-bit bus The configuration of ASIZE determines the function of the EM_A[20:0] and EM_BA[1:0] pins as described in Section 4.2.7.1. This field also determines the number of external accesses required to fulfill a request generated by one of the sources mentioned in Section 4.2.6.7 . For example, a request for a double-word (32 bits) would require four external access when ASIZE = 0h. See the datasheet of the external asynchronous device to determine the appropriate setting for this field.

Table 4-18. Description of the Asynchronous Wait Cycle Configuration Registers (AWCCR1 and AWCCR2)

Parameter	Description
WP[3:0]	**Wait pin polarity.** • WP*n* = 0h selects active-high polarity • WP*n* = 1h selects active-low polarity When set to 1, the EMIF will wait if the wait pin is high. When set to 0, the EMIF will wait if the wait pin is low. The EMIF must have the Extended Wait Mode enabled for the wait pins to affect the width of the strobe period. The polarity of the wait pins is programmable. **NOTE:** The polarity of the wait pins is not programmable in NAND Flash Mode.
MEWC	**Maximum Extended Wait Cycles.** This field configures the number of EMIF clock cycles the EMIF will wait for the wait pins (EM_WAIT[3:0]) to be deactivated during the strobe period of an access cycle. The maximum number of EMIF clock cycles it will wait is determined by the following formula: Maximum Extended Wait Cycles = (MEWC + 1) * 16 If the wait pin is not deactivated within the time specified by this field, the EMIF resumes the access cycle, registering whatever data is on the bus and preceding to the hold period of the access cycle. This situation is referred to as an Asynchronous Timeout. An Asynchronous Timeout generates an interrupt if it has been enabled in the EMIF Interrupt Mask Set Register (EIMSR). For more information about the EMIF's interrupts, see Section 4.2.14. **NOTE:** Extended Wait Mode should not be used while in NAND Flash Mode. See Section 4.2.7.6 for more details on this mode of operation.

Table 4-19. Description of the EMIF Interrupt Mask Set Register (EIMSR)

Parameter	Description
WRMSET	**Wait Rise Mask Set.** Writing a 1 to this bit enables an interrupt to be generated when a rising edge on a wait pin occurs while in NAND Flash Mode
ATMSET	**Asynchronous Timeout Mask Set.** Writing a 1 to this bit enables an interrupt to be generated when an Asynchronous Timeout occurs.

Table 4-20. Description of the EMIF Interrupt Mask Clear Register (EIMCR)

Parameter	Description
WRMCLR	**Wait Rise Mask Clear.** Writing a 1 to this bit disables the interrupt, clearing the WRMSET bit in the EMIF interrupt mask set register (EIMSR).
ATMCLR	**Asynchronous Timeout Mask Clear.** Writing a 1 to this bit enables an interrupt to be generated when an Asynchronous Timeout occurs.

4.2.7.4 Read and Write Operations in Normal Mode

Normal Mode is the asynchronous interface's default mode of operation. It is selected when the SS bit of the Asynchronous CSn Configuration Register 2 (ACSnCR2) is cleared to 0. In this mode, the EM_DQM pins operate as byte enables. Section 4.2.7.4.1 and Section 4.2.7.4.2 explain the details of read and write operations while in Normal Mode.

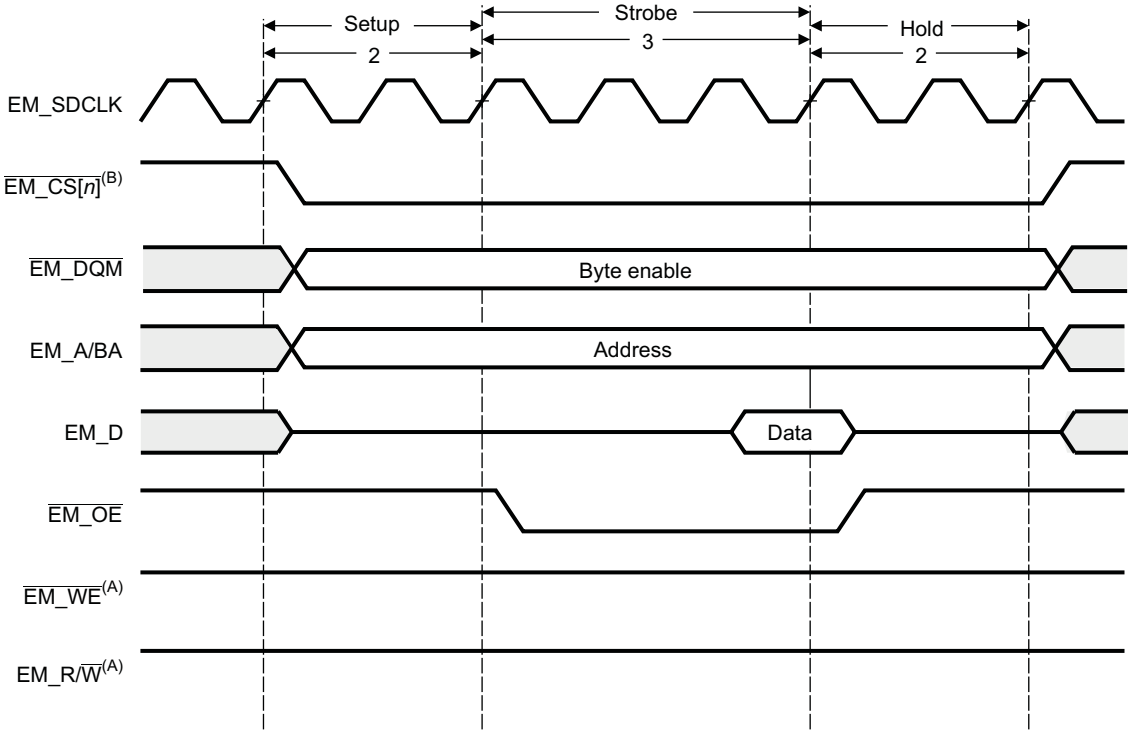
4.2.7.4.1 Asynchronous Read Operations (Normal Mode)

An asynchronous read is performed when any of the requesters mentioned in Section 4.2.2 request a read from the attached asynchronous memory. After the request is received, a read operation is initiated once it becomes the EMIF's highest priority task, according to the priority scheme detailed in Section 4.2.10. In the event that the read request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIF until the entire request is fulfilled. The details of an asynchronous read operation in Normal Mode are described in Table 4-21. Also, Figure 4-10 shows an example timing diagram of a basic read operation.

Table 4-21. Asynchronous Read Operation in Normal Mode

Time Interval	Pin Activity in Normal Mode
Turn-around period	Once the read operation becomes the highest priority task for the EMIF, the EMIF waits for the programmed number of turn-around cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the Asynchronous CSn Configuration Register 1 (ACSnCR1). There are two exceptions to this rule: • If the current read operation was directly proceeded by another read operation, no turnaround cycles are inserted. • If the current read operation was directly proceeded by a write operation and the TA field has been set to 0h, one turn-around cycle will be inserted. After the EMIF has waited for the turnaround cycles to complete, it again checks to make sure that the read operation is still its highest priority task. If so, the EMIF proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIF terminates the operation.
Start of the setup period	The following actions occur at the start of the setup period: • The setup, strobe, and hold values are set according to the R_SETUP, R_STROBE, and R_HOLD values in ACSnCR1 and ACSNCR2. • The address pins EM_A[20:0] and EM_BA[1:0] become valid and carry the values described in Figure 4-10. • EM_DQM[1:0] becomes valid. • EM_CS[n] (where n = 2, 3, 4, or 5) falls to enable the external device (if not already low from a previous operation).
Strobe period	The following actions occur during the strobe period of a read operation: 1. EM_OE falls at the start of the strobe period 2. On the rising edge of the clock which is concurrent with the end of the strobe period: • EM_OE rises • The data on the EM_D[15:0] bus is sampled by the EMIF. In Figure 4-10, the wait pins (EM_WAIT[3:0]) are inactive. If wait pins are instead activated, the strobe period can be extended by the external device to give it more time to provide the data. Section 4.2.5 contains more details on using the wait pins.
End of the hold period	At the end of the hold period: • The address pins EM_A[20:0], EM_BA[1:0], and EM_DQM[1:0] become invalid • EM_CS[n] rises (if no more operations are required to complete the current request) EMIF may be required to issue additional read operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIF immediately re-enters the setup period to begin another operation without incurring the turn-round cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIF returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIF instead enters directly into the turnaround period for the pending read or write operation.

Figure 4-10. Timing Waveform of an Asynchronous Read Cycle in Normal Mode



A. During the entirety of an asychronous read operation, the EM_WE and EM_R/W pins are driven high.
B. n=2, 3, 4, or 5

4.2.7.4.2 Asynchronous Write Operations (Normal Mode)

An asynchronous write is performed when any of the requesters mentioned in Section 4.2.2 request a write to memory in the asynchronous bank of the EMIF. After the request is received, a write operation is initiated once it becomes the EMIF's highest priority task, according to the priority scheme detailed in Section 4.2.10. In the event that the write request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIF until the entire request is fulfilled. The details of an asynchronous write operation in Normal Mode are described in Table 4-22. Also, Figure 4-11 shows an example timing diagram of a basic write operation.

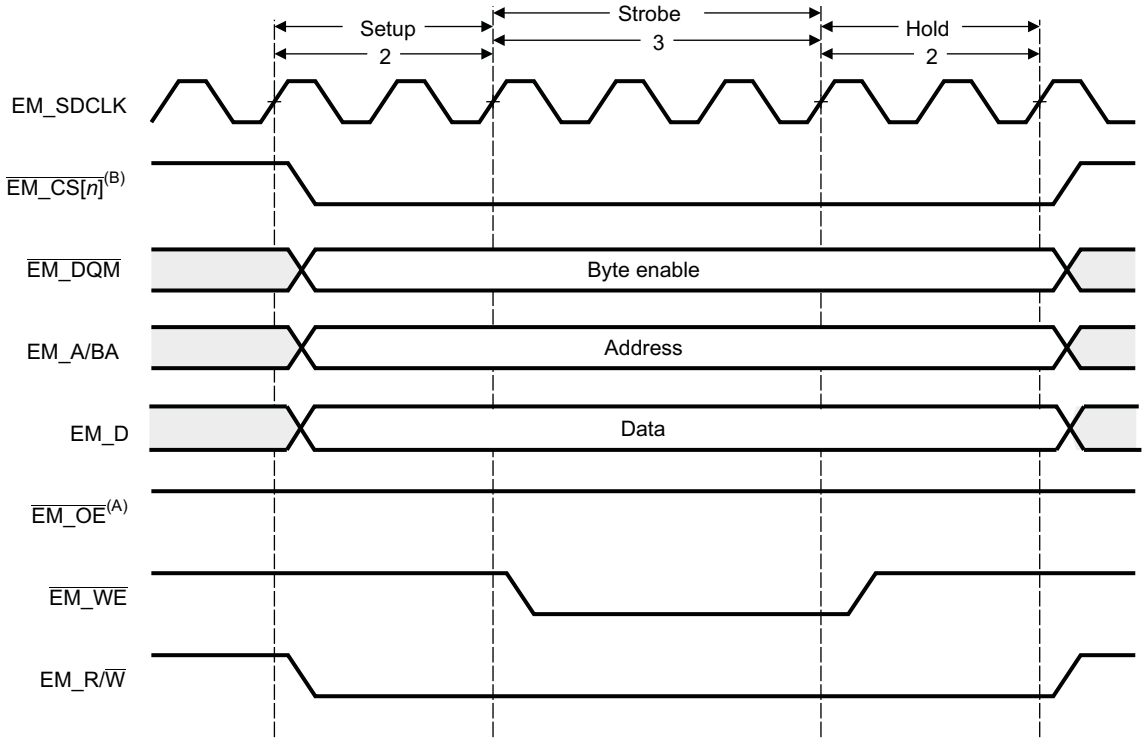
Table 4-22. Asynchronous Write Operation in Normal Mode

Time Interval	Pin Activity in Normal Mode
Turnaround period	Once the write operation becomes the highest priority task for the EMIF, the EMIF waits for the programmed number of turn-around cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the Asynchronous CSn Configuration Register 1 (ACSnCR1). There are two exceptions to this rule: • If the current write operation was directly proceeded by another write operation, no turn-around cycles are inserted. • If the current write operation was directly proceeded by a read operation and the TA field has been set to 0h, one turnaround cycle will be inserted. After the EMIF has waited for the turn-around cycles to complete, it again checks to make sure that the write operation is still its highest priority task. If so, the EMIF proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIF terminates the operation.

Table 4-22. Asynchronous Write Operation in Normal Mode (continued)

Time Interval	Pin Activity in Normal Mode
Start of the setup period	The following actions occur at the start of the setup period: • The setup, strobe, and hold values are set according to the W_SETUP, W_STROBE, and W_HOLD values in ACSnCR2. • The address pins EM_A[20:0] and EM_BA[1:0] and the data pins EM_D[15:0] become valid. The EM_A[20:0] and EM_BA[1:0] pins carry the values described in Section 4.2.7.1. • $\overline{\text{EM\_DQM}}$[1:0] become valid. • The EM_R/$\overline{\text{W}}$ pin falls to indicate a write (if not already low from a previous operation). • $\overline{\text{EM\_CS}}$[n] (where n = 2, 3, 4, or 5) falls to enable the external device (if not already low from a previous operation).
Strobe period	The following actions occur at the start of the strobe period of a write operation: • $\overline{\text{EM\_WE}}$ falls • Normal Mode The following actions occur on the rising edge of the clock which is concurrent with the end of the strobe period: • $\overline{\text{EM\_WE}}$ rises • Normal Mode In Figure 4-11, the wait pins (EM_WAIT[3:0]) are inactive. If the wait pins are instead activated, the strobe period can be extended by the external device to give it more time to accept the data. Section 4.2.7.7 contains more details on using the wait pins.
End of the hold period	At the end of the hold period: • The address pins EM_A[20:0] and EM_BA[1:0] become invalid • The data pins become invalid • The EM_R/$\overline{\text{W}}$ pin rises (if no more operations are required to complete the current request) • $\overline{\text{EM\_CS}}$[n] rises (if no more operations are required to complete the current request) The EMIF may be required to issue additional write operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIF immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIF returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIF instead enters directly into the turnaround period for the pending read or write operation.

Figure 4-11. Timing Waveform of an Asynchronous Write Cycle in Normal Mode



A. During the entirety of an asychronous write operation, the EMA_OE pin is driven high.
B. n=2, 3, 4, 5

4.2.7.5 Read and Write Operation in Select Strobe Mode

Select Strobe Mode is the EMIF's second mode of operation. It is selected when the SS bit of the Asynchronous CSn Configuration Register 2 (ACSnCR2) is set to 1. In this mode, the EM_DQM[1:0] pins operate as byte enables and the EM_CS[5:2] pins are only active during the strobe period of an access cycle. Section 4.2.7.5.1 and Section 4.2.7.5.2 explain the details of read and write operations while in Select Strobe Mode.

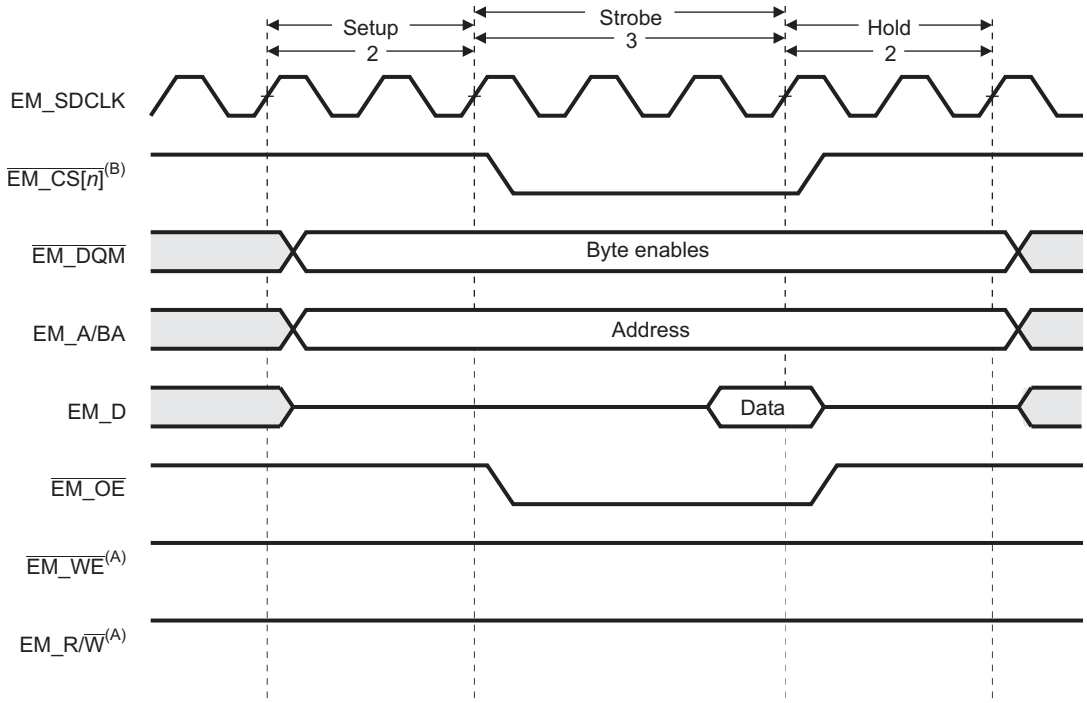
4.2.7.5.1 Asynchronous Read Operations (Select Strobe Mode)

An asynchronous read is performed when any of the requesters mentioned in Section 4.2.2 request a read from the attached asynchronous memory. After the request is received, a read operation is initiated once it becomes the EMIF's highest priority task, according to the priority scheme detailed in Section 4.2.10. In the event that the read request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIF until the entire request is fulfilled. The details of an asynchronous read operation in Select Strobe Mode are described in Table 4-23. Also, Figure 4-12 shows an example timing diagram of a basic read operation.

Table 4-23. Asynchronous Read Operation in Select Strobe Mode

Time Interval	Pin Activity in Select Strobe Mode
Turnaround period	Once the read operation becomes the highest priority task for the EMIF, the EMIF waits for the programmed number of turn-around cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the Asynchronous CSn Configuration Register 1 (ACSnCR1). There are two exceptions to this rule: • If the current read operation was directly proceeded by another read operation, no turn-around cycles are inserted. • If the current read operation was directly proceeded by a write operation and the TA field has been set to 0h, one turn-around cycle will be inserted. After the EMIF has waited for the turn-around cycles to complete, it again checks to make sure that the read operation is still its highest priority task. If so, the EMIF proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIF terminates the operation.
Start of the setup period	The following actions occur at the start of the setup period: • The setup, strobe, and hold values are set according to the R_SETUP, R_STROBE, and R_HOLD values in ACSnCR1 and ACSnCR2. • The address pins EM_A[20:0] and EM_BA[1:0] become valid and carry the values described in Section 4.2.7.1. • The $\overline{\text{EM\_DQM}}$ pins become valid as byte enables.
Strobe period	The following actions occur during the strobe period of a read operation: • $\overline{\text{EM\_CS}}$[n] (where n = 2, 3, 4, or 5) and $\overline{\text{EM\_OE}}$ fall at the start of the strobe period • On the rising edge of the clock which is concurrent with the end of the strobe period: – $\overline{\text{EM\_CS}}$[n] and $\overline{\text{EM\_OE}}$ rise – The data on the EM_D[15:0] bus is sampled by the EMIF. In Figure 4-12, the wait pins (EM_WAIT[3:0]) are inactive. If the wait pins are instead activated, the strobe period can be extended by the external device to give it more time to provide the data. Section 4.2.7.7 contains more details on using the wait pins.
End of the hold period	At the end of the hold period: • The address pins EM_A[20:0] and EM_BA[1:0] become invalid • The $\overline{\text{EM\_DQM}}$ pins become invalid The EMIF may be required to issue additional read operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIF immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIF returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIF instead enters directly into the turnaround period for the pending read or write operation.

Figure 4-12. Timing Waveform of an Asynchronous Read Cycle in Select Strobe Mode



A. During the entirety of an asychronous read operation, the EM_WE and EM_R/W pins are driven high.
B. n=2, 3, 4, or 5

4.2.7.5.2 Asynchronous Write Operations (Select Strobe Mode)

An asynchronous write is performed when any of the requesters mentioned in Section 4.2.2 request a write to memory in the asynchronous bank of the EMIF. After the request is received, a write operation is initiated once it becomes the EMIF's highest priority task, according to the priority scheme detailed in Section 4.2.10 . In the event that the write request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIF until the entire request is fulfilled. The details of an asynchronous write operation in Select Strobe Mode are described in Table 4-24 . Also, Figure 4-13 shows an example timing diagram of a basic write operation.

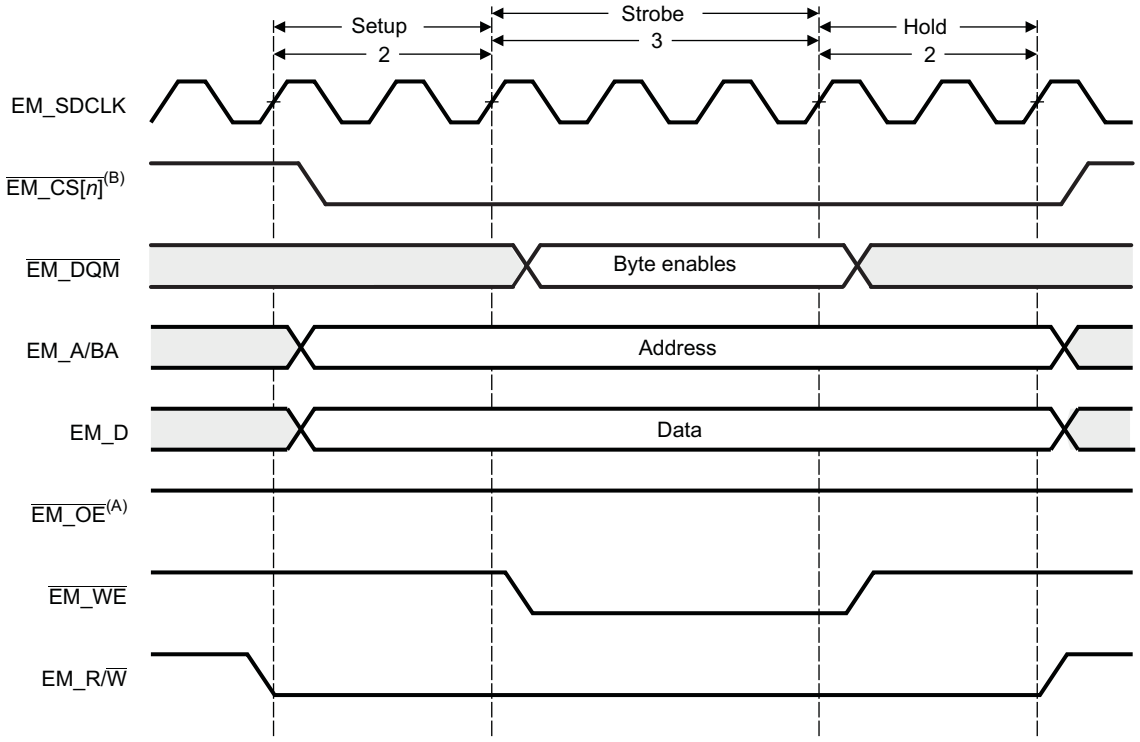
Table 4-24. Asynchronous Write Operation in Select Strobe Mode

Time Interval	Pin Activity in Select Strobe Mode
Turnaround period	Once the write operation becomes the highest priority task for the EMIF, the EMIF waits for the programmed number of turn-around cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the Asynchronous CSn Configuration Register 1 (ACSnCR1). There are two exceptions to this rule: • If the current write operation was directly proceeded by another write operation, no turn-around cycles are inserted. • If the current write operation was directly proceeded by a read operation and the TA field has been set to 0h, one turnaround cycle will be inserted. After the EMIF has waited for the turn-around cycles to complete, it again checks to make sure that the write operation is still its highest priority task. If so, the EMIF proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIF terminates the operation.
Start of the setup period	The following actions occur at the start of the setup period: • The setup, strobe, and hold values are set according to the W_SETUP, W_STROBE, and W_HOLD values in ACSnCR2. • The address pins EM_A[20:0] and EM_BA[1:0] and the data pins EM_D[15:0] become valid. The EM_A[20:0] and EM_BA[1:0] pins carry the values described in Section 4.2.7.1. • The EM_R/W pin falls to indicate a write (if not already low from a previous operation). • EM_CS[n] (where n = 2, 3, 4, or 5) falls to enable the external device (if not already low from a previous operation).

Table 4-24. Asynchronous Write Operation in Select Strobe Mode (continued)

Time Interval	Pin Activity in Select Strobe Mode
Strobe period	The following actions occur at the start of the strobe period of a write operation: • $\overline{\text{EM\_WE}}$ falls • The $\overline{\text{EM\_DQM}}$ pins become active as write strobes. The following actions occur on the rising edge of the clock which is concurrent with the end of the strobe period: • $\overline{\text{EM\_WE}}$ rises • The $\overline{\text{EM\_DQM}}$ pins deactivate In Figure 4-13, the wait pins (EM_WAIT[3:0]) are inactive. If the wait pins are instead activated, the strobe period can be extended by the external device to give it more time to accept the data. Section 4.2.7.7 contains more details on using the wait pins.
End of the hold period	At the end of the hold period: • The address pins EM_A[20:0] and EM_BA[1:0] become invalid • The data pins become invalid • The EM_R/$\overline{\text{W}}$ pin rises (if no more operations are required to complete the current request) • $\overline{\text{EM\_CS[n]}}$ rises (if no more operations are required to complete the current request) The EMIF may be required to issue additional write operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIF immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIF returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIF instead enters directly into the turnaround period for the pending read or write operation.

Figure 4-13. Timing Waveform of an Asynchronous Write Cycle in Select Strobe Mode



A. During the entirety of an asychronous write operation, the EM_OE pin is driven high.
B. n=2, 3, 4, or 5

4.2.7.6 NAND Flash Mode

NAND Flash Mode is a sub-mode of both Normal Mode and Select Strobe Mode. The chip select pins (EM_CS[5:2]) may be placed in NAND Flash mode by setting the CSn_USE_NAND bits in the NAND Flash control register (NANDFCR). Note that the NAND Flash Mode can be independently enabled for each chip select space. Table 4-25 displays the bit fields present in NANDFCR and briefly describes their use.

When a chip select space is configured to operate in NAND Flash mode, the EMIF hardware can calculate the error correction code (ECC) for each 512 byte data transfer to that chip select space. The EMIF hardware will not automatically generate a NAND access cycle, which includes the command, address, and data phases, necessary to complete a transfer to NAND Flash. All NAND Flash operations can be divided into single asynchronous cycles and with the help of software, the EMIF can execute a complete NAND access cycle.

NOTE: By default, the EM_CS2 pin is set to NAND Flash mode after reset. The NAND Flash mode is disabled for the other chip select pins (EM_CS[3:5]).

Table 4-25. Description of the NAND Flash Control Register (NANDFCR)

Parameter	Description
CS2_ECC_START	**NAND Flash ECC state for EM_CS2.** Set to 1 to start an ECC calculation for NAND Flash connected to EM_CS2. This bit is cleared to 0 when the NAND Flash 1-Bit ECC Registers for EMIF CS2 are read.
CS2_USE_NAND	**NAND Flash mode for EM_CS2.** Set to 1 to enable NAND Flash mode for EM_CS2.
CS3_ECC_START	**NAND Flash ECC state for EM_CS3.** Set to 1 to start an ECC calculation for NAND Flash connected to EM_CS3. This bit is cleared to 0 when the NAND Flash 1-Bit ECC Registers for EMIF CS3 are read.
CS3_USE_NAND	**NAND Flash mode for EM_CS3.** Set to 1 to enable NAND Flash mode for EM_CS3.
CS4_ECC_START	**NAND Flash ECC state for EM_CS4.** Set to 1 to start an ECC calculation for NAND Flash connected to EM_CS4. This bit is cleared to 0 when the NAND Flash 1-Bit ECC Registers for EMIF CS4 are read.
CS4_USE_NAND	**NAND Flash mode for EM_CS4.** Set to 1 to enable NAND Flash mode for EM_CS4.
CS5_ECC_START	**NAND Flash ECC state for EM_CS5.** Set to 1 to start an ECC calculation for NAND Flash connected to EM_CS5. This bit is cleared to 0 when the NAND Flash 1-Bit ECC Registers for EMIF CS5 are read.
CS5_USE_NAND	**NAND Flash mode for EM_CS5.** Set to 1 to enable NAND Flash mode for EM_CS5.

4.2.7.6.1 Configuring for NAND Flash Mode

Similar to the asynchronous accesses previously described, the EMIF's registers must be programmed appropriately to interface to a NAND Flash device. In addition to the fields listed in Table 4-25, you should set the CSn_USE_NAND bits of the NAND Flash Control Register (NANDFCR) to 1 for the chip select spaces you want to operate in NAND Flash Mode. Note that the Extended Wait Mode of any chip select space being used in NAND Flash Mode should be disabled by setting EW = 0 in the Asynchronous CSn Configuration Register 2 (ACSnCR2).

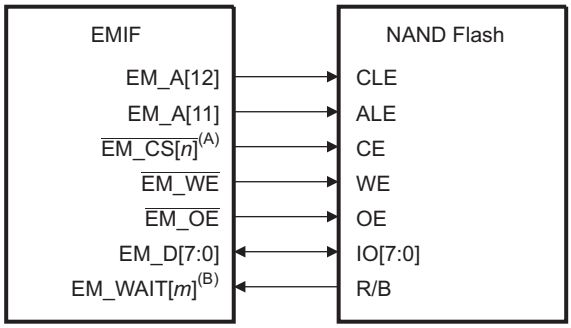
4.2.7.6.2 Connecting to NAND Flash

Figure 4-14 shows the EMIF external pins used to interface with a NAND Flash device. EMIF address lines are used to drive the NAND Flash device's command latch enable (CLE) and address latch enable (ALE) signals. Note that you can use any EMIF address lines to drive the CLE and ALE signals of the NAND Flash.

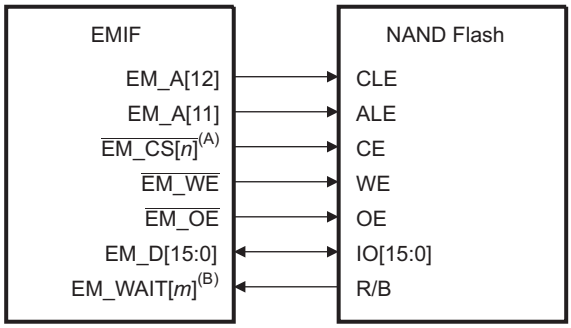
NOTE: The EMIF will not control the NAND Flash device's write protect pin. The write protect pin must be controlled outside of the EMIF.

NOTE: By default, the $\overline{\text{EM\_CS2}}$ pin is set to NAND Flash mode after reset. The NAND Flash mode is disabled for the other chip select pins ($\overline{\text{EM\_CS}}$[3:5]).

Figure 4-14. EMIF to NAND Flash Interface



a) Connection to 8-Bit NAND Device



b) Connection to 16-Bit NAND Device

A. n=2, 3, 4, or 5
B. m=0, 1, 2, or 3

4.2.7.6.3 Driving CLE and ALE

As described in Section 4.2.7.6.2, any of the EMIF address lines can be used to drive the ALE and CLE pins of the NAND Flash device. The following must be considered when using this approach:

- The CPU cannot generate 8-bit accesses to its data or I/O space. However, the EMIF can be programmed to access a single byte for every word access initiated by the CPU (see Section 4.2.8 for more details).
- As stated in Section 4.2.7.1, the EMIF always drives the least significant bit of a double-word (32-bit) address on EM_A[0].

Table 4-26 shows how a CPU word address (EM_A[21:0]) is driven on the EMIF address pins. Table 4-26 also shows how a DMA byte address is driven on the EMIF address pins.

Table 4-26. CPU and DMA Address to EMIF Address Pin Mapping

CPU Word Address	DMA Byte Address	EMIF Address Pins
A21	A22	EM_A[20]
…	…	…
A13	A14	EM_A[12]
A12	A13	EM_A[11]
…	…	…
A1	A2	EM_A[0]
A0	A1	EM_BA[1]

Table 4-26. CPU and DMA Address to EMIF Address Pin Mapping (continued)

CPU Word Address	DMA Byte Address	EMIF Address Pins
0	A0	EM_BA[0]

Notes:

- The EM_BA[1:0] pins are not required when interfacing to NAND Flash memory, however, they are included in this figure to provide a complete picture of CPU address to EMIF address pin mapping.
- The EM_BA[0] pin is only used if the EMIF data bus is configured as an 8-bit bus (ASIZE = 00b).

As an example, suppose that the EMIF address pins EM_A[12] and EM_A[11] are used to drive the CLE and ALE pins of the NAND Flash memory and that the $\overline{\text{EM\_CS2}}$ is used to drive the CE pin. In this situation, a write or read from the following CPU word addresses can be used to manipulate the CLE and ALE pins:

- 40 0000h (40 0000h + 00 0000h): a CPU write or read from this address drives CLE and ALE low.
- 40 2000h (40 0000h + 00 2000h): a CPU write or read from this address drives CLE high and ALE low.
- 40 1000h (40 0000h + 00 1000h): a CPU write or read from this address drives CLE low and ALE high.

Note that in this example the CPU must access EMIF CS2 to initiate accesses with the memory device. As indicated by Table 4-1, EMIF CS2 starts at CPU word address 40 0000h. When using other chip select spaces to interface to NAND, simply replace base address used in this example with that of chip select being used.

4.2.7.6.4 NAND Read and Program Operations

A NAND Flash access cycle is composed of a command, address, and data phase. The EMIF will not automatically generate these three phases to complete a NAND access with one transfer request. To complete a NAND access cycle, multiple single asynchronous access cycles (as described above) must be completed by the EMIF. Software must be used to request the appropriate asynchronous accesses to complete a NAND Flash access cycle. This software must be developed to the specification of the chosen NAND Flash device.

Since NAND operations are divided into single asynchronous access cycles, the chip select signal will not remain activated for the duration of the NAND operation. Instead, the chip select signal will deactivate between each asynchronous access cycle. For this reason, the EMIF does not support NAND Flash devices that require the chip select signal to remain low during the t_R time for a read. See Section 4.2.7.6.9 for workaround.

Care must be taken when performing a NAND read or write operation via the DMA controller. See Section 4.2.7.6.5 for more details.

NOTE: The EMIF does not support NAND Flash devices that require the chip select signal to remain low during the t_R time for a read. See Section 4.2.7.6.9 for workaround.

4.2.7.6.5 NAND Data Read and Write via DMA Controller

When performing NAND accesses, the DMA controller is most efficiently used for the data phase of the access. The command and address phases of the NAND access require only a few words of data to be transferred and therefore do not take advantage of the DMA controller's ability to transfer larger quantities of data with a single request. Furthermore, since the minimum amount of data the DMA controller can transfer is one double-word (32-bits), it cannot be used during NAND command and address phases. In this section we will focus on using the DMA controller for the data phase of a NAND access.

There are two conditions that require care to be taken when performing NAND reads and writes via the DMA controller. These are:

- The address lines used to drive CLE and ALE signals must be driven low.
- The EMIF does not support a constant address mode, but only supports linear incrementing address modes.

Since the EMIF does not support a constant addressing mode, when programming the DMA, a linear incrementing address mode must be used. When using a linear incrementing address mode, care must be taken not to increase the address into a range that drives CLE and/or ALE high. To prevent the address from incrementing into a range that drives CLE and/or ALE high, the DMA start address and transfer size must be carefully considered.

Consider a system in which EM_A[12] is connected to CLE and EM_A[11] is connected to ALE. In this case, per Figure 4-15, the transfer size of the DMA must stay below 8192 bytes (2^13) to avoid driving the CLE and ALE pins to 1. The DMA setup for a NAND Flash data read would look as follows:

- 4 ≤ LENGTH ≤ 8192 bytes
- DSTAMODE = 00b, automatic post increment
- SRCAMODE = 00b, automatic post increment
- SSA (source start address) = CE base address
- DSA (destination start address) = address of internal memory buffer

Similarly, the DMA setup for a NAND Flash data write would be as follows:

- 4 ≤ LENGTH ≤ 8192 bytes
- DSTAMODE = 00b, automatic post increment
- SRCAMODE = 00b, automatic post increment
- SSA (source start address) = address of internal memory buffer
- DSA (destination start address) = CE base address

4.2.7.6.6 1-Bit ECC Generation

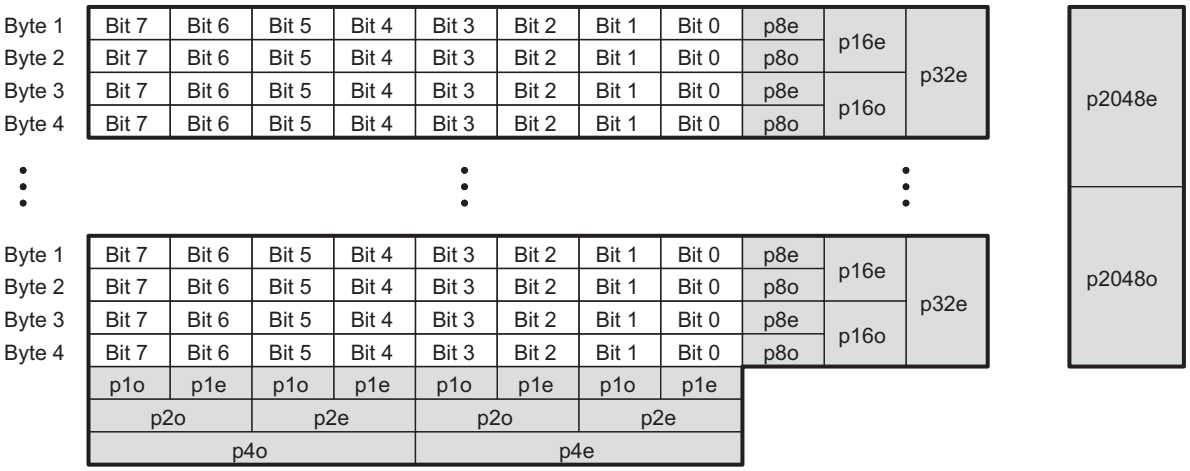
If the CSn_USE_NAND bits in the NAND Flash control register (NANDFCR) are set to 1, the EMIF supports ECC calculation for up to 512 bytes for the corresponding chip select space. To perform the ECC calculation, the corresponding CSn_ECC_START bits in NANDFCR must be set to 1. It is the responsibility of the software to start the ECC calculation prior to issuing a write or read to NAND Flash. It is also the responsibility of the software to read the calculated ECC from the NAND Flash 1-Bit ECC registers once the transfer to NAND Flash has completed. If the software writes or reads more than 512 bytes, the ECC will be incorrect.

Reading a NAND Flash 1-Bit ECC register clears the corresponding CSn_ECC_START bit in NANDFCR. Also, the NAND Flash 1-Bit ECC registers are cleared upon writing a 1 to the corresponding CSn_ECC_START bit in NANDFCR.

Figure 4-15 shows the algorithm used to calculate the ECC value for an 8-bit NAND Flash.

For an 8-bit NAND Flash, p1o through p4e are column parities and p8e through p2048o are row parities. Similarly, the algorithm can be extended to a 16-bit NAND Flash. For a 16-bit NAND Flash p1o through p8e are column parities and p16e through p2048o are row parities. The software must ignore the unwanted parity bits if ECC is desired for less than 512 bytes of data. For example, p2048e and p2048o are not required for ECC on 256 bytes of data. Similarly, p1024e, p1024o, p2048e, and p2048o are not required for ECC on 128 bytes of data.

Figure 4-15. ECC Value for 8-Bit NAND Flash



4.2.7.6.7 4-Bit ECC Generation

The EMIF supports 4-bit ECC for 8-bit and 16-bit NAND Flash. In NAND mode, if the NAND Flash 4-bit ECC start bit (4BIT_ECC_START) in the NAND Flash Control register (NANDFCR) is set, the EMIF calculates 4-bit ECC for the selected chip select. Only one chip select can be selected for the 4-bit ECC calculation at one time. The selection of the chip select is done by programming the 4-bit ECC CS select bit field (4BIT_ECC_SEL) in NANDFCR.

The calculated parity (for writes) and syndrome (for reads) can be read from the NAND Flash 4-Bit ECC registers (NAND4BITECC1[8:1]). The 4-bit ECC start bit (4BIT_ECC_START) is cleared upon reading any of the NAND Flash 4-Bit ECC registers. The NAND Flash 4-Bit ECC registers are cleared upon writing 1 to the 4-bit ECC start bit (4BIT_ECC_START).

The 4-bit ECC algorithm works on a 10-bit data bus. Since the 4-bit ECC is calculated over an 8-bit value for 8- and 16-bit NAND Flash, the EMIF zeroes the upper two bits. However, the parity and the syndrome value read from the NAND Flash 4-Bit ECC registers are 10 bits wide. It is the responsibility of software to convert 10-bit parity values to 8 or 16 bits before writing to the spare location of the NAND Flash after a write operation. Similarly, it is the responsibility of the software to convert the 8- or 16-bit parity values read from the spare location of the NAND Flash after a read operation to 10 bits before writing the NAND Flash 4-Bit ECC Load register (NAND4BITECCLOAD).

At the end of the syndrome calculation after read, the error address and the error value can be calculated by setting the address and error value calculation start bit (ADDR_CALC_ST) in the NAND Flash Control register (NANDFCR). The end of address calculation is flagged by the 4-bit ECC correction state field (CORRSTATE) in the NAND Flash Status register 1 (NANDFSR1). The number of errors can be read from the 4-bit number of errors field (ERRNUM) in the NAND Flash Status register 2 (NANDFSR2). The error address value can be read from the NAND Flash 4-Bit ECC Error Address registers (NANDERRADD[4:1]). The error value can be read from the NAND Flash 4-Bit ECC Error Value registers (NANDERRVAL1[4:1]).

The address and error value start bit (ADDR_CALC_ST) are cleared upon reading any of the NAND Flash 4-Bit ECC Error Address registers or the NAND Flash 4-Bit ECC Error Value registers. The EMIF records the syndrome value internally before the error address and error value calculation. Therefore, a new read operation can be performed simultaneously with the error address calculation.

The EMIF supports 4-bit ECC calculation up to 518 bytes. The software needs to follow the following procedure for 4-bit ECC calculation:

For writes:

1. Set the (4BIT_ECC_START) bit in the NAND Flash Control register (NANDFCR) to 1.
2. Write 518 bytes of data to the NAND Flash.
3. Read the parity from the NAND Flash 4-Bit ECC registers (NAND4BITECC1[8:1]).

4. Convert the 10-bit parity values to 8 or 16 bits depending on the width of the connected NAND Flash memory. All 10-bit parity values can be concatenated together with ECC value 1 (4BIT_ECC_VAL1) as the LSB and ECC value 8 (4BIT_ECC_VAL8) as the MSB. Then the concatenated value can be broken down into ten 8-bit or five 16-bit values.
5. Store the parity to a spare location in the NAND Flash.

For reads:

1. Set the (4BIT_ECC_START) bit in the NAND Flash Control register (NANDFCR) to 1.
2. Read 518 bytes of data from the NAND Flash.
3. Clear the (4BIT_ECC_START) bit in NANDFCR by reading any of the NAND Flash 4-Bit ECC registers.
4. Read the parity stored in the spare location in the NAND Flash.
5. Convert the 8-bit or 16-bit parity values to 10-bits. Reverse of the conversion that was done during writes.
6. Write the parity values in the NAND Flash 4-bit ECC Load register (NAND4BITECCLOAD). Write each parity value one at a time starting from 4BIT_ECC_VAL8 to 4BIT_ECC_VAL1.
7. Perform a dummy read to the EMIF revision register (REV). This is only required to ensure time for syndrome calculation after writing the ECC values in step 6.

8. Read the syndrome from the NAND Flash 4-Bit ECC registers (NAND4BITECC1[8:1]). A syndrome value of 0 means no bit errors. If the syndrome is non-zero continue to step 9.
9. Set the (ADDR_CALC_ST) it in NANDFCR to 1.
10. Start another read from NAND if required (a new thread from step 1).
11. Wait for the 4-bit ECC correction state field (CORRSTATE) in NANDFSR1 to be equal to 1h, 2h, or 3h.
12. Read the number of errors from the 4-bit number of errors field (ERRNUM) in the NANDFSR2.
13. Read the error address from the NAND Flash 4-Bit ECC Error Address registers (NANDERRADD[4:1]). The address for the word in error is equal to: total_words_read + 7 - address_value. For 518 bytes, the address will be equal to: 525 - address_value.
14. Read the error value from NAND Flash 4-Bit ECC Error Value registers (NANDERRVAL[4:1]). The values from these registers can be XORed with the errored word to correct the errors.

NOTE: You must perform step 13 or 14. Otherwise the ADDR_CALC_ST bit will not be cleared and you will not be able to start a new address and value error calculation.

4.2.7.6.8 NAND Flash Status Registers (NANDFSR1 and NANDFSR2)

The NAND Flash status register 1 (NANDFSR1) indicates the raw status of the EM_WAIT[3:0] pins while in NAND Flash Mode. A wait pin should be connected to the NAND Flash device's R/B signal, so that it can indicate whether or not the NAND Flash device is busy.

During a read, the R/B signal will transition and remain low while the NAND Flash retrieves the data requested. Once the R/B signal transitions high, the requested data is ready and should be read by the EMIF. During a write/program operation, the R/B signal transitions and remains low while the NAND Flash is programming the Flash with the data it has received from the EMIF. Once the R/B signal transitions high, the data has been written to the Flash and the next phase of the transaction may be performed.

From this explanation, you can see that the NAND Flash status register is useful to the software for indicating the status of the NAND Flash device and determining when to proceed to the next phase of a NAND Flash operation.

When a rising edge occurs on a wait pin, the EMIF sets the corresponding WR (wait rise) bit in the EMIF Interrupt Raw Register (EIRR). Therefore, the EMIF Wait Rise interrupt may be used to indicate the status of the NAND Flash device. The WPn bits in the Asynchronous Wait Cycle Configuration Register (AWCCR) do not affect the NAND Flash status register (NANDFSR) or the WR bits in EIRR. See Section 4.2.14 for more a detailed description of the wait rise interrupt.

4.2.7.6.9 Interfacing to a Non-CE Don't Care NAND Flash

As explained in Section 4.2.7.6.4, the EMIF does not support NAND Flash devices that require the chip select signal to remain low during the t_R time for a read. One way to work around this limitation is to use a GPIO pin to drive the CE signal of the NAND Flash device. If this work around is implemented, software will configure the selected GPIO to be low, then begin the NAND Flash operation, starting with the command phase. Once the NAND Flash operation has completed the software can then configure the selected GPIO to be high.

4.2.7.7 Extended Wait Mode and the Wait Pins

The EMIF supports the Extend Wait Mode. This is a mode in which the external asynchronous device may assert control over the length of the strobe period. The Extended Wait Mode can be enabled on a per chip select basis by setting the EW bit in the Asynchronous CSn Configuration register 2 (ACSnCR2). When this bit is set, the EMIF monitors the wait pin corresponding to the chip select being accessed to determine if the attached device wishes to extend the strobe period of the current access cycle beyond the programmed number of clock cycles.

If the EMIF detects that the wait pin has been asserted, it will begin inserting extra strobe cycles into the operation until the wait pin is deactivated by the external device. The EMIF will then return to the last cycle of the programmed strobe period and the operation will proceed as usual from this point. Please refer to the device data manual for details on the timing requirements of the wait pins.

The wait pins cannot be used to extend the strobe period indefinitely. The programmable MEWC field in the Asynchronous Wait Cycle Configuration Register 1 (AWCCR1) determines the maximum number of EMIF clock cycles the strobe period may be extended beyond the programmed length. When the counter expires, the EMIF proceeds to the hold period of the operation regardless of the state of the wait pin. The EMIF can also generate an interrupt upon expiration of this counter. See Section 4.2.14 for details on enabling this interrupt.

For the EMIF to function properly in the Extended Wait mode, the wait pin polarity bits (WPn) of AWCCR2 must be programmed to set the polarity of the wait pins. In its reset state of 1, the EMIF will insert wait cycles when the wait pin is sampled high. When set to 0, the EMIF will insert wait cycles only when wait pin is sampled low. This programmability allows for a glueless connection to larger variety of asynchronous devices.

By default, a wait pin is assigned to each chip select space. You can change the default wait pin assignment using the CSn_WAIT bits of AWCCR2. You are allowed to assign a single wait pin to more than one chip select space.

Finally, a restriction is placed on the strobe period timing parameters when operating in Extended Wait mode. Specifically, the W_STROBE and R_STROBE fields must not be set to 0 for proper operation.

4.2.8 BYTEMODE Bits of The EMIF System Control Register

WARNING

The BYTEMODE bits affect CPU program and data accesses to external memory as well as CPU accesses to the EMIF registers. Therefore, to avoid data corruption issues you must always set BYTEMODE = 00b (16-bit access) after you have completed all 8-bit CPU accesses to external memory or EMIF registers. External memory accesses by other modules are not affected by these bits.

The CPU cannot generate 8-bit accesses to its data or I/O space. This presents a problem given that it is often necessary to access a single byte when communicating with NAND Flash devices or, in the case of the EMIF SDRAM self-refresh mode, you must only access a single byte in the SDRAM Configuration Register 2 (SDCR2) to avoid triggering the SDRAM auto-initialization procedure.

For these situations, the BYTEMODE bits of the EMIF System Control Register (ESCR) can be used to program the DSP switched central resource (SCR) such that a CPU word access generates a single byte access when reading or writing from external memory or when accessing the EMIF registers. Table 4-27 summarizes the effect of the BYTEMODE bits for different CPU operations. For more details on ESCR, see the *DSP System User's Guide* chapter.

NOTE: The BYTEMODE bits should only be used for controlling CPU accesses to NAND Flash devices and EMIF registers.

Table 4-27. Effect of BYTEMODE Bits on EMIF Accesses

BYTEMODE Setting	CPU Access to EMIF Register	CPU Access To External Memory
BYTEMODE = 00b (16-bit word access)	Entire register contents are accessed	ASIZE = 01b (16-bit data bus): EMIF generates a single 16-bit access to external memory for every CPU word access. ASIZE = 00b (8-bit data bus): EMIF generates two 8-bit accesses to external memory for every CPU word access.

Table 4-27. Effect of BYTEMODE Bits on EMIF Accesses (continued)

BYTEMODE Setting	CPU Access to EMIF Register	CPU Access To External Memory
BYTEMODE = 01b (8-bit access with high byte selected)	Only the upper byte of the register is accessed.	ASIZE = 01b (16-bit data bus): EMIF generates a 16-bit access to external memory for every CPU word access; only the high byte of the EMIF data bus is used. ASIZE = 00b (8-bit data bus): EMIF generates a single 8-bit access to external memory for every CPU word access.
BYTEMODE = 10b (8-bit access with low byte selected)	Only the lower byte of the register is accessed.	ASIZE = 01b (16-bit data bus): EMIF generates a 16-bit access to external memory for every CPU word access; only the low byte of the EMIF data bus is used. ASIZE = 00b (8-bit data bus): EMIF generates a single 8-bit access to external memory for every CPU word access.

4.2.9 Data Bus Parking

The EMIF always drives the data bus to the previous write data value when it is idle. This feature is called data bus parking. Only when the EMIF issues a read command to the external memory does it stop driving the data bus. After the EMIF latches the last read data, it immediately parks the data bus again.

The one exception to this behavior occurs after performing an asynchronous read operation while the EMIF is in the self-refresh state. In this situation, the read operation is not followed by the EMIF parking the data bus, instead the EMIF tri-states the data bus. Therefore, it is not recommended to perform asynchronous read operations while the EMIF is in the self-refresh state, in order to prevent floating inputs on the data bus. External pull-up or pull-down resistors should be placed on the EMIF data bus pins if it is required to perform reads in this situation. The precise resistor value should be chosen so that the worst case combined off-state leakage currents do not cause the voltage levels on the associated pins to drop below the high-level input voltage requirement.

More information about the self-refresh state can be found in Section 4.2.6.7.

4.2.10 Priority and Arbitration

Section 4.2.2 of this document describes the external prioritization and arbitration among requests from different sources within the DSP. The result of this external arbitration is that only one request is presented to the EMIF at a time. Once the EMIF completes a request, the external arbiter then provides the EMIF with the next pending request.

Internally, the EMIF undertakes memory device transactions according to a strict priority scheme. The highest priority events are:

- A hardware reset (see Section 4.2.12).
- A write to SDRAM configuration register 1 (SDCR1) or the least significant byte of SDRAM configuration register 2 (SDCR2).

Either of these will cause the EMIF to immediately commence its initialization sequence as described in Section 4.2.6.4 .

Once the EMIF has completed its initialization sequence, it performs memory transactions according to the following priority scheme (highest priority listed first):

1. If the EMIF's backlog refresh counter is at the Refresh Must urgency level, the EMIF performs multiple SDRAM auto refresh cycles until the Refresh Release urgency level is reached.
2. If an SDRAM or asynchronous read has been requested, the EMIF performs a read operation.
3. If the EMIF's backlog refresh counter is at the Refresh Need urgency level, the EMIF performs an SDRAM auto refresh cycle.
4. If an SDRAM or asynchronous write has been requested, the EMIF performs a write operation.
5. If the EMIF's backlog refresh counter is at the Refresh May or Refresh Release urgency level, the EMIF performs an SDRAM auto refresh cycle.

6. If the value of the SR bit in SDCR2 has been set to 1, the EMIF will enter the self-refresh state as described in Section 4.2.6.7.

After taking one of the actions listed above, the EMIF then returns to the top of the priority list to determine its next action.

Because the EMIF does not issue auto-refresh cycles when in the self-refresh state, the above priority scheme does not apply when in this state. See Section 4.2.6.7 for details on the operation of the EMIF when in the self-refresh state.

4.2.11 System Considerations

In a system that interfaces to both SDRAM and asynchronous memory, the asynchronous requests must not take longer than the smaller of the following two values:

- t_{RAS} (typically 120us): to avoid violating the maximum time allowed between issuing an ACTV and PRE command to the SDRAM.
- $t_{Refresh\_Rate}$ x 11 (typically 15.7 ms = 11 = 172.7 ms): to avoid refresh violations on the SDRAM.

The length of an asynchronous request is controlled by multiple factors, the primary factor being the number of access cycles required to complete the request. For example, an asynchronous request for 4 bytes will require four access cycles using an 8-bit data bus configuration and only two access cycles using a 16-bit data bus configuration.

The length of the individual access cycles that make up the asynchronous request is determined by the programmed setup, strobe, hold, and turnaround values, but can also be extended with the assertion of the EM_WAIT[3:0] input signals up to a programmed maximum limit. It is up to the user to make sure that an entire asynchronous request does not exceed the timing values listed above when also interfacing to an SDRAM device. This can be done by limiting the asynchronous timing parameters.

4.2.12 Reset Considerations

The EMIF has one reset source: a hardware reset. This reset is always initiated during a full chip reset. Alternatively, software can force a hardware reset on the EMIF through the PG1_RST bit of the peripheral reset control register (PRCR). See the device data manual for more details on PRCR. Please note that the EMIF input clock must be enabled for PG1_RST to have an affect on the EMIF (see Section 4.2.1). Also note that the PG1_RST bit resets other modules in the DSP.

When a hardware reset occurs, the EMIF state machine and registers are reset. Command and data stored in the EMIF memory controller FIFOs is lost.

NOTE: External memory accesses and EMIF register accesses should not be performed while PG1_RST is asserted.

When the EMIF is taken out of reset it automatically begins running the SDRAM initialization sequence described in Section 4.2.6.4. Note that even though the initialization procedure is automatically started, you must still follow the configuration procedure described Section 4.2.6.5.

4.2.13 Initialization

The following initialization procedure describes the basic setup of the EMIF.

1. Perform the necessary device pin multiplexing setup (see Section 4.2.5 for more details).
2. Program the DSP clock generator to provide the desired EM_SDCLK clock frequency. For details on programming the DSP clock generator, see the *DSP System Guide* chapter .
3. Set the EDIV bit of the EMIF clock divider register (ECDR) to select the frequency of EM_SDCLK: half the CPU clock rate or equal to the CPU clock rate. The frequency of EM_SDCLK must meet the timing requirements in the SDRAM manufacturer's documentation and the timing limitations described in electrical specifications of the device data manual. For this device's EM_SDCLK limitation, see section Section 4.2.1.
4. Reset the EMIF using the PG1_RST bit of the peripheral reset control register (PRCR). See Section 4.2.12 for more details on this bit.
5. If you want to use interrupts, you can enable them in this step. The EMIF can generate interrupts on three conditions: the rising edge of the wait signals (EM_WAIT[3:0]), asynchronous memory access time-outs, and addressing errors. More information on interrupts is given in Section 4.2.14.
6. If using the EMIF to access SDRAM, follow the steps outlined in Section 4.2.6.5 (including turning on the SDCLK as outline in that section)..
7. If using the EMIF to access asynchronous memories or other devices, configure the EMIF as described in Section 4.2.7.3.

4.2.14 Interrupt Support

The EMIF supports a single interrupt to the CPU. Section 4.2.14.1 details the generation and internal masking of EMIF interrupts, and Section 4.2.14.2 describes how the EMIF interrupts are sent to the CPU.

4.2.14.1 Interrupt Events and Requests

There are three conditions that may cause the EMIF to generate an interrupt to the CPU. These conditions are:

- A rising edge on the EM_WAIT[3:0] signals (wait interrupt).
- An asynchronous memory access time out.
- Usage of unsupported addressing mode (line trap interrupt).

The wait interrupt is not affected by the wait pin polarity configuration bits (WPn) in the asynchronous wait cycle configuration register 2 (AWCCR2). The asynchronous time out interrupt condition occurs when the attached asynchronous device fails to dessert the EM_WAIT pin within the number of cycles defined by the MEWC bits in the asynchronous wait cycle configuration register 1 (AWCCR1).

The EMIF supports only linear incrementing addressing mode. If an access request for an unsupported addressing mode is received, the EMIF will set the LT bit in interrupt raw register and treat the request as a linear incrementing request.

Only when the interrupt is enabled by setting the appropriate bit (WRMASKSET/ATMASKSET/LTMASKSET) in the EMIF interrupt mask set register (EIMSR) to 1, will the interrupt be sent to the CPU. Once enabled, the interrupt may be disabled by writing a 1 to the corresponding bit in the EMIF interrupt mask clear register (EIMCR). The bit fields in both the EIMSR and EIMCR may be used to indicate whether the interrupt is enabled. When the interrupt is enabled, the corresponding bit field in both the EIMSR and EIMCR will have a value of 1; when the interrupt is disabled, the corresponding bit field will have a value of 0.

The EMIF interrupt raw register (EIRR) and the EMIF interrupt mask register (EIMR) indicate the status of each interrupt. The appropriate bit (WR/AT/LT) in EIRR is set when the interrupt condition occurs, whether or not the interrupt has been enabled. However, the appropriate bit (WRMASKED/ATMASKED/LTMASKED) in EIMR is set only when the interrupt condition occurs and the interrupt is enabled. Writing a 1 to the bit in EIRR clears the EIRR bit as well as the corresponding bit in EIMR.

Table 4-28 contains a brief summary of the interrupt status and control bit fields. See for complete details on the register fields.

Table 4-28. Interrupt Monitor and Control Bit Fields

Register Name	Bit Name	Description
EMIF interrupt raw register (EIRR)	WR[3:0]	These bits are set when an rising edge on the wait signals (EM_WAIT[3:0]) occurs. Writing a 1 clears the WR bits as well as the WRMASKED bits in EIMR. Each WR bit corresponds to an EM_WAIT pin.
	AT	This bit is set when an asynchronous timeout occurs. Writing a 1 clears the AT bit as well as the ATMASKED bit in EIMR.
	LT	This bit is set when an unsupported addressing mode is used. Writing a 1 clears LT bit as well as the LTMASKED bit in EIMR.
EMIF interrupt mask register (EIMR)	WRMASKED[3:0]	These bits are set only when a rising edge on the wait signals (EM_WAIT[3:0]) occurs and the interrupt has been enabled by writing a 1 to the WRMASKSET bits in EIMSR. Each WRMASKED bit corresponds to an EM_WAIT pin.
	ATMASKED	This bit is set only when an asynchronous timeout occurs and the interrupt has been enabled by writing a 1 to the ATMASKSET bit in EIMSR.
	LTMASKED	This bit is set only when line trap interrupt occurs and the interrupt has been enabled by writing a 1 to the LTMASKSET bit in EIMSR.
EMIF interrupt mask set register (EIMSR)	WRMASKSET[3:0]	Writing a 1 to these bits enables the wait rise interrupt of the corresponding wait pin. Each WRMASKSET bit corresponds to an EM_WAIT pin.
	ATMASKSET	Writing a 1 to this bit enables the asynchronous timeout interrupt.
	LTMASKSET	Writing a 1 to this bit enables the line trap interrupt.
EMIF interrupt mask clear register (EIMCR)	WRMASKCLR[3:0]	Writing a 1 to these bits disables the wait rise interrupt of the corresponding wait pin. Each WRMASKCLR bit corresponds to an EM_WAIT pin.
	ATMASKCLR	Writing a 1 to this bit disables the asynchronous timeout interrupt.
	LTMASKCLR	Writing a 1 to this bit disables the line trap interrupt.

4.2.14.2 Interrupt Multiplexing

The EMIF interrupt to the DSP CPU is not multiplexed with any other interrupt source.

4.2.15 DMA Event Support

The EMIF does not generate any DMA events.

4.2.16 Power Management

There are several ways to reduce the power consumption of the device EMIF. First, the EMIF power domain voltage can be selected during the board design phase based on the requirements of the devices connected to the EMIF. Running at lower voltages consumes less power. The lowest voltage allowable by the memory device should be used to minimize power consumption.

Second, there are several clock options. the input clock of the EMIF can be turned off by using the peripheral clock gating configuration register (PCGCR). For detailed information on PCGCR see the device-specific data manual.

- The input clock of the EMIF can be turned off by using the peripheral clock gating configuration register (PCGCR). For detailed information on PCGCR, see the device-specific data manual.
- SDCLK can be turned off if SDRAM is not used by using the Clock Configuration Register 1 (CCR1). For detail information on CCR1, see the device-specific data manual.
- EMIF clock divider should be used if SDRAM operates at less than half the CPU frequency. The EMIF clock divider divides the EMIF clock by two for SDCLK. This is set using the EMIF Clock Divider Register (ECDR). For detail information on ECDR, see the device-specific data manual and Section 4.2.1 for this device EM_SDCLK limitation.

Third, when connected to SDRAM, the EMIF can enter a self-refresh mode. While in the self-refresh state, the EMIF continues to service asynchronous bank requests and register accesses as normal. Self-refresh mode must be used with care as bus parking is disabled following a read to asynchronous memory while in this mode. This could lead to floating inputs which would lead to higher power consumption. For more information on the EMIF self-refresh mode please see Section 4.2.6.7.

Lastly, the EMIF can be requested to issue a power down command to an SDRAM device. When placed in the power down mode the memory device deactivates its input and output buffers, excluding CKE, to maximize power savings. While in the power down state, the EMIF services the SDRAM, asynchronous memory, and register accesses as normal, returning to the power down state upon completion. For more information on the EMIF power down mode, see Section 4.2.6.8 and Section 4.2.1 for the device's EM_SDCLK limitations.

4.2.17 Emulation Considerations

The EMIF will remain fully functional during emulation halts to allow emulation access to external memory.

4.2.18 CPU Instruction Pipeline Considerations

This section explains two special cases of pipeline operation that could impact external memory accesses.

As described in the *TMS320C55x DSP CPU Reference Guide* (SPRU371), the CPU uses instruction pipelining. Multiple instructions are processed simultaneously in the pipeline, and different instructions may access external memory during different phases of completion. The pipeline consists of a number of phases during which different, designated tasks are performed.

4.2.18.1 A Write Followed by a Read at a Different Address

The read phase of the pipeline is used to read operands and other data needed to complete the execution of an instruction. The results of an instruction may be written to external memory in the write phase of the pipeline. The read phase occurs earlier in the pipeline than the write phase. For this reason, the read and write requests made to the EMIF may occur in an order which is different than the order in which the instructions entered the pipeline. For example, consider the following code segment:

```
I1: MOV T0, *(#External_Address_1) ; Instruction 1 writes to external memory
I2: MOV *(#External_Address_2), T1 ; Instruction 2 reads from external memory
```

Although the code shows the write followed by the read, the read actually occurs first on the EMIF because of the pipelining effect, as shown in Table 4-29. The figure shows the two instructions passing through the read (R), execute (X), and write (W) phases.

For some applications, maintaining the proper write-read order is critical. In such cases, NOP (no operation) instructions (or other instructions that do not perform external memory accesses) can be inserted between the original instructions to delay the read operation. This technique is shown in Table 4-30.

Table 4-29. Partial Pipeline Diagram of Consecutive Instructions That Write and Read at Different Addresses

R	X	W	Cycle	Commnet
I1			n	
I2	I1		n+1	Read initiated by instruction 2
	I2	I1	n+2	Write initiated by instruction 1
		I2	n+3	

Table 4-30. NOP Instructions Inserted in the Code of Figure 1−3 to Make the Write Occur Before the Read

R	X	W	Cycle	Commnet
I1			n	
NOP	I1		n+1	
NOP	NOP	I1	n+2	Write initiated by instruction 1
I2	NOP	NOP	n+3	Read initiated by instruction 2
	I2	NOP	n+4	
		I2	n+5	

4.2.18.2 A Write Followed by a Read at the Same Address

In most cases, when a write to memory is followed immediately by a read at the same address, the data written is the same data expected back during the read. The C55x CPU takes advantage of this fact with a special memory-bypass feature. During the read, the CPU gets a copy of the data directly from the write bus(es) instead of accessing memory.

When the CPU is accessing external devices, there may be cases in which the memory-bypass feature would lead to unwanted results. For example, suppose two physical memory locations X and Y are mapped to the same address. Writing modifies location X, and reading gets data from location Y. If the memory-bypass feature takes effect, location Y is not read.

To prevent the memory bypass, insert three or more NOP instructions (or other instructions) between the instructions that perform the write and the read. For example:

```
MOVT0,*(#External_Address_1)  ; Write to address 1
NOP                           ; 3-cycle delay
NOP
NOP
MOV*(#External_Address_1), T1 ; Read from address 1
```

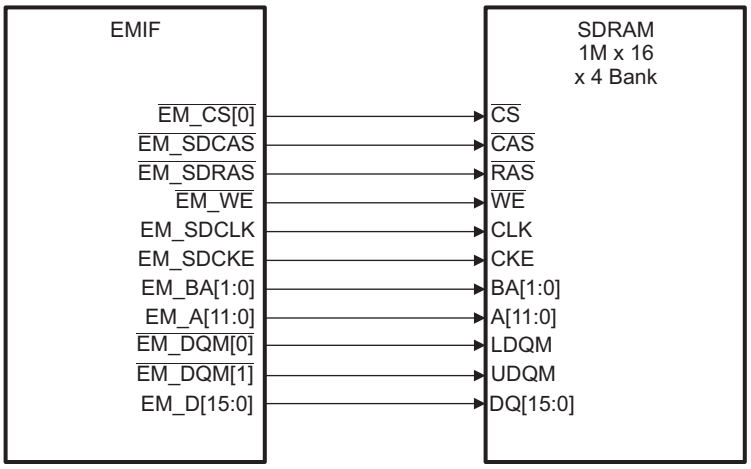
4.3 Interfacing the EMIF to Mobile SDRAM

This section presents an example of interfacing the EMIF to a mobile SDRAM device. The 16Mbit mobile SDRAM Micron MT48H4M16LF-8 device is used for this example.

4.3.1 Hardware Interface

Figure 4-16 shows the hardware interface between the EMIF and the 16Mbit mobile SDRAM Micron MT48H4M16LF-8 device.

Figure 4-16. Hardware Interface to Mobile SDRAM Device



4.3.2 SW Configuration

This section describes how to configure the EMIF registers to interface with the Micron MT48H4M16LF-8 device. The SDRAM clock frequency is assumed to be 100 MHz (f_{EM_SDCLK} = 100 MHz).

4.3.2.1 Settings for SDRAM Timing Registers (SDTIMR1 and SDTIMR2)

The fields of SDTIMR1 and SDTIMR2 control various timing parameters used by the EMIF. Table 4-31 shows the values that should be used when interfacing to the Micron MT48H4M16LF device. These values were derived using timing specifications obtained from the memory's datasheet. Figure 4-17 and Figure 4-18 show a graphical representation of the contents of SDTIMR1 and SDTIMR2 after being programmed with the derived values.

Table 4-31. SDTIMR1 and SDTIMR2 Field Calculations

Field Name	Formula	Value from MT48H4M16LF-8 Datasheet	Value Calculated for Field
T_RFC	T_RFC ≥ (t_{RFC}*f_{EM_SDCLK}) - 1	t_{RFC} = 80 ns (min)	7
T_RP	T_RP ≥ (t_{RP}*f_{EM_SDCLK}) – 1	t_{RP} = 19 ns (min)	1
T_RCD	T_RCD ≥ (t_{RCD}*f_{EM_SDCLK}) – 1	t_{RCD} = 19 ns (min)	1
T_WR	T_WR ≥ (t_{WR}*f_{EM_SDCLK}) – 1	t_{RDL} **(1)** = 2 clocks = 20ns (min)	1
T_RAS	T_RAS ≥ (t_{RAS}*f_{EM_SDCLK}) - 1	t_{RAS} = 48 ns (min)	4
T_RC	T_RC ≥ (t_{RC}*f_{EM_SDCLK}) - 1	t_{RFC} = 80 ns (min)	7
T_RRD	T_RRD ≥ (t_{RRD}*f_{EM_SDCLK}) - 1	t_{RRD} = 16 ns (min)	1

(1) The Micron MT48H4M16LF data sheet does not specify a t_{WR}, instead is specifies t_{RDL} as the last data in to PRECHARGE command.

Figure 4-17. SDTIMR1 Contents

15–12	11–8	7	6–4	3–0
0100	0111	0	001	0000
TRAS	TRC	Reserved	TRRD	Reserved

Figure 4-18. SDTIMR2 Contents

15–11	10–8	7	6–4	3	2–0
0 0111	001	0	001	0	001
TRFC	TRP	Reserved	TRCD	Reserved	TWR

4.3.2.2 Settings for SDRAM Self Refresh Exit Timing Register (SDSRETR)

The TXS field of SDSRETR should be programmed satisfy the t_{XSR} timing requirement of the memory. Table 4-32 shows the calculation of the value for TXS and Figure 4-19 shows a graphical representation of SDSRETR after being programmed with this value.

Table 4-32. SDSRETR Field Calculations

Field Name	Formula	Value from MT48H4M16LF-8 Datasheet	Value Calculated for Field
TXS	TXS ≥ (t_{XSR}*f_{EM_SDCLK}) - 1	t_{XSR} = 80 ns (min)	7

Figure 4-19. SDRETR Contents

15	5	4	0
0		0 0111	
Reserved		TXS	

4.3.2.3 Settings for SDRAM Refresh Control Register (SDRCR)

SDRCR should next be programmed to satisfy the required refresh rate of the memory device. Table 4-33 shows the calculation of the proper value to program into the REFRATE field of this register. Based on this calculation, a value of 61Ah should be written to SDRCR. Figure 4-20 shows a graphical representation of the value that should be programmed into SDRCR.

Table 4-33. SDRCR Field Calculations

Field Name	Formula	Value From MT48H4M16LF-8 Data Sheet	Value Calculated for Field
REFRATE	REFRATE ≤ f_{EM_SDCLK} * $t_{Refresh\_Period}$ / ncycles	$t_{Refresh\_Period}$ = 64 ms; ncycles = 4096	1562 = 61Ah

Figure 4-20. SDRCR Contents

15	13	12	0
0		0 0110 0001 1010 (61Ah)	
Reserved		REFRATE	

4.3.2.4 Settings for SDRAM Configuration Registers (SDCR1 and SDCR2)

The fields of SDCR1 and SDCR2 should be programmed as described in Table 4-34 to properly interface with the memory device. Based on these settings, a value of 4720h should be written to SDCR1 and a value of 0001h should be written to SDCR2. Figure 4-21 and Figure 4-22 show a graphical representation of the values that should be programmed into SDCR1 and SDCR2.

Table 4-34. SDCR1 and SDCR2 Field Calculations

Field Name	Value	Purpose
SR	0	To avoid placing the EMIF in Self-Refresh Mode.
PD	0	To avoid placing the EMIF in Power Down Mode.
PDWR	0	To disable refreshes during power down.
PASR	0	To select 4 banks for refresh during SLFR command.
ROWSIZE	Don't care	This field is only used when IBANK_POS = 1.
IBANK_POS	0	To configure the EMIF to access the page in all banks before moving on to the next page. You can set this bit to 1 if you want the EMIF to access all the pages in a single bank before moving on to the next page.

Table 4-34. SDCR1 and SDCR2 Field Calculations (continued)

Field Name	Value	Purpose
SDRAM_DRIVE	0	To select full drive strength when initializing the memory device.
BIT_9_1_UNLOCK	1	To allow writes to PASR, ROWSIZE, IBANK_POS, and SDRAM_DRIVE.
NM	1	This bit should always be set to 1.
CL	3	To select a CAS latency of 3.
BIT_11_9_LOCK	1	To allow the CL field to be written.
IBANK	2	To select 4 internal SDRAM banks.
EBANK	0	To use a single chip select when communicating with the memory device.
PAGESIZE	0	To select a page size of 256 words.

Figure 4-21. SDCR1 Contents

15	14	13	12	11	9	8
0	1	0		011		1
Reserved	NM	Reserved		CL		BIT_11_9_LOCK

7	6	4	3	2	0
0	010		0	000	
Reserved	IBANK		EBANK	PAGESIZE	

Figure 4-22. SDCR2 Contents

15	14	13	12	10	9	8
0	0	0	000		000	
SR	PD	PDWR	Reserved		PASR	

7	6	4	3	2	1	0
0	000		0	00		1
PASR	ROWSIZE		IBANK_POS	SDRAM_DRIVE		BIT_9_1_LOCK

4.3.2.5 Software Sequence for Programming EMIF Registers

There are several steps involved in setting up the EMIF for SDRAM accesses. These steps are described in Section 4.2.6.4 (steps applicable for all types of memories) and Section 4.2.6.5 (steps specific for SDRAM accesses). For reference, Table 4-35 summarizes the values that should be written to the EMIF registers when interfacing to a 16 Mbit mobile SDRAM Micron MT48H4M16LF-8 device using an EM_SDCLK frequency of 100 MHz.

Table 4-35. EMIF Register Values for Micron MT48H4M16LF-8 Mobile SDRAM Device

Register	Value
SDTIMR1	4710h
SDTIMR2	3911h
SDRETR	0007h
SDRCR	061Ah
SDCR1	4720h
SDCR2	0001h
CCR1	0001h

4.4 EMIF Registers

Table 4-36 lists the memory-mapped registers for the EMIF. All register offset addresses not listed in Table 4-36 should be considered as reserved locations and the register contents should not be modified.

Table 4-36. EMIF REGISTERS

CPU Word Address	Acronym	Register Name	Section
1000h	REV	Revision Register	Section 4.4.1
1001h	STATUS	Status Register	Section 4.4.2
1004h	AWCCR1	Asynchronous Wait Cycle Configuration Register 1	Section 4.4.3
1005h	AWCCR2	Asynchronous Wait Cycle Configuration Register 2	Section 4.4.4
1008h	SDCR1	SDRAM Configuration Register 1	Section 4.4.5
1009h	SDCR2	SDRAM Configuration Register 2	Section 4.4.6
100Ch	SDRCR	SDRAM Refresh Control Register	Section 4.4.7
1010h	ACS2CR1	Asynchronous CS2 Configuration Register 1	Section 4.4.8
1011h	ACS2CR2	Asynchronous CS2 Configuration Register 2	Section 4.4.9
1014h	ACS3CR1	Asynchronous CS3 Configuration Register 1	Section 4.4.10
1015h	ACS3CR2	Asynchronous CS3 Configuration Register 2	Section 4.4.11
1018h	ACS4CR1	Asynchronous CS4 Configuration Register 1	Section 4.4.12
1019h	ACS4CR2	Asynchronous CS4 Configuration Register 2	Section 4.4.13
101Ch	ACS5CR1	Asynchronous CS5 Configuration Register 1	Section 4.4.14
101Dh	ACS5CR2	Asynchronous CS5 Configuration Register 2	Section 4.4.15
1020h	SDTIMR1	SDRAM Timing Register 1	Section 4.4.16
1021h	SDTIMR2	SDRAM Timing Register 1	Section 4.4.17
103Ch	SDSRETR	SDRAM Self Refresh Exit Timing Register	Section 4.4.18
1040h	EIRR	EMIF Interrupt Raw Register	Section 4.4.19
1044h	EIMR	EMIF Interrupt Mask Register	Section 4.4.20
1048h	EIMSR	EMIF Interrupt Mask Set Register	Section 4.4.21
104Ch	EIMCR	EMIF Interrupt Mask clear Register	Section 4.4.22
1060h	NANDFCR	NAND Flash Control Register	Section 4.4.23
1064h	NANDFSR1	NAND Flash Status Register 1	Section 4.4.24
1065h	NANDFSR2	NAND Flash Status Register 2	Section 4.4.25
1068h	PAGEMODCTRL1	Page Mode Control Register 1	Section 4.4.26
1069h	PAGEMODCTRL2	Page Mode Control Register 2	Section 4.4.27
1070h	NCS2ECC1	NAND Flash CS2 1-Bit ECC Register 1	Section 4.4.28
1071h	NCS2ECC2	NAND Flash CS2 1-Bit ECC Register 2	Section 4.4.29
1074h	NCS3ECC1	NAND Flash CS3 1-Bit ECC Register 1	Section 4.4.30
1075h	NCS3ECC2	NAND Flash CS3 1-Bit ECC Register 2	Section 4.4.31
1078h	NCS4ECC1	NAND Flash CS4 1-Bit ECC Register 1	Section 4.4.32
1079h	NCS4ECC2	NAND Flash CS4 1-Bit ECC Register 2	Section 4.4.33
107Ch	NCS5ECC1	NAND Flash CS5 1-Bit ECC Register 1	Section 4.4.34
107Dh	NCS5ECC2	NAND Flash CS5 1-Bit ECC Register 2	Section 4.4.35
10BCh	NAND4BITECCLOAD	NAND Flash 4-Bit ECC Load Register	Section 4.4.36
10C0h	NAND4BITECC1	NAND Flash 4-Bit ECC Register 1	Section 4.4.37
10C1h	NAND4BITECC2	NAND Flash 4-Bit ECC Register 2	Section 4.4.38
10C4h	NAND4BITECC3	NAND Flash 4-Bit ECC Register 3	Section 4.4.39
10C5h	NAND4BITECC4	NAND Flash 4-Bit ECC Register 4	Section 4.4.40
10C8h	NAND4BITECC5	NAND Flash 4-Bit ECC Register 5	Section 4.4.41
10C9h	NAND4BITECC6	NAND Flash 4-Bit ECC Register 6	Section 4.4.42
10CCh	NAND4BITECC7	NAND Flash 4-Bit ECC Register 7	Section 4.4.43

Table 4-36. EMIF REGISTERS (continued)

CPU Word Address	Acronym	Register Name	Section
10CDh	NAND4BITECC8	NAND Flash 4-Bit ECC Register 8	Section 4.4.44
10D0h	NANDERRADD1	NAND Flash 4-Bit ECC Error Address Register 1	Section 4.4.45
10D1h	NANDERRADD2	NAND Flash 4-Bit ECC Error Address Register 2	Section 4.4.46
10D4h	NANDERRADD3	NAND Flash 4-Bit ECC Error Address Register 3	Section 4.4.47
10D5h	NANDERRADD4	NAND Flash 4-Bit ECC Error Address Register 4	Section 4.4.48
10D8h	NANDERRVAL1	NAND Flash 4-Bit ECC Error Value Register 1	Section 4.4.49
10D9h	NANDERRVAL2	NAND Flash 4-Bit ECC Error Value Register 2	Section 4.4.50
10DCh	NANDERRVAL3	NAND Flash 4-Bit ECC Error Value Register 3	Section 4.4.51
10DDh	NANDERRVAL4	NAND Flash 4-Bit ECC Error Value Register 4	Section 4.4.52

4.4.1 REV Register (offset = 1000h) [reset = 205h]

REV is shown in Figure 4-23 and described in Table 4-37.

This is a read only register indicating the major and minor revision of the EMIF.

Figure 4-23. REV Register

15	14	13	12	11	10	9	8
MAJOR_REVISION							
R-2h							
7	6	5	4	3	2	1	0
MINOR_REVISION							
R-5h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-37. REV Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	MAJOR_REVISION	R	2h	Major revision code for EMIF.
7-0	MINOR_REVISION	R	5h	Minor revision code for EMIF.

4.4.2 STATUS Register (offset = 1001h) [reset = C000h]

STATUS is shown in Figure 4-24 and described in Table 4-38.

Status Register

Figure 4-24. STATUS Register

15	14	13	12	11	10	9	8
BIGENDIAN	FULLRATE	MODULE_ID					
R-1h	R-1h	R-0h					

7	6	5	4	3	2	1	0
MODULE_ID							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-38. STATUS Register Field Descriptions

Bit	Field	Type	Reset	Description
15	BIGENDIAN	R	1h	Big endian value, defining big endian mode (data R/W order) 0x0 = EMIF is running in little endian. 0x1 = EMIF is running in big endian.
14	FULLRATE	R	1h	Full rate status bit. This bit reflects the clock rate being used by the EMIF. 0x0 = EMIF is running at half the CPU clock frequency. 0x1 = EMIF is running at the same speed as the CPU clock.
13-0	MODULE_ID	R	0h	Module ID code for EMIF.

4.4.3 AWCCR1 Register (offset = 1004h) [reset = 40h]

AWCCR1 is shown in Figure 4-25 and described in Table 4-39.

Asynchronous Wait Cycle Configuration Register 1

Figure 4-25. AWCCR1 Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
MEWC							
R/W-40h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-39. AWCCR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	MEWC	R/W	40h	Maximum extended wait cycles. The EMIF will wait for a maximum of (MEWC + 1) x 16 clock cycles before it stops inserting asynchronous wait cycles and proceeds to the hold period of the access. This setting applies to all the wait pins.

4.4.4 AWCCR2 Register (offset = 1005h) [reset = F0E4h]

AWCCR2 is shown in Figure 4-26 and described in Table 4-40.

Asynchronous Wait Cycle Configuration Register 2

Figure 4-26. AWCCR2 Register

15	14	13	12	11	10	9	8
WP3	WP2	WP1	WP0	Reserved			
R/W-1h	R/W-1h	R/W-1h	R/W-1h	R-0h			

7	6	5	4	3	2	1	0
CS5_WAIT		CS4_WAIT		CS3_WAIT		CS2_WAIT	
R/W-3h		R/W-2h		R/W-1h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-40. AWCCR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	WP3	R/W	1h	EM_WAIT3 polarity bit. This bit defines the polarity of the EM_WAIT3 pin. 0x0 = Insert wait cycles if wait pin is low. 0x1 = Insert wait cycles if wait pin is high.
14	WP2	R/W	1h	EM_WAIT2 polarity bit. This bit defines the polarity of the EM_WAIT2 pin. 0x0 = Insert wait cycles if wait pin is low. 0x1 = Insert wait cycles if wait pin is high.
13	WP1	R/W	1h	EM_WAIT1 polarity bit. This bit defines the polarity of the EM_WAIT1 pin. 0x0 = Insert wait cycles if wait pin is low. 0x1 = Insert wait cycles if wait pin is high.
12	WP0	R/W	1h	EM_WAIT0 polarity bit. This bit defines the polarity of the EM_WAIT0 pin. 0x0 = Insert wait cycles if wait pin is low. 0x1 = Insert wait cycles if wait pin is high.
11-8	Reserved	R	0h	Reserved
7-6	CS5_WAIT	R/W	3h	Wait pin mapping bits for EMIF CS5. By default, each asynchronous chip select space is assigned a wait input pin. You can use the wait pin mapping bits to change the default assignment. 0x0 = Use the EM_WAIT0 pin. 0x1 = Use the EM_WAIT1 pin. 0x2 = Use the EM_WAIT2 pin. 0x3 = Use the EM_WAIT3 pin.
5-4	CS4_WAIT	R/W	2h	Wait pin mapping bits for EMIF CS4. By default, each asynchronous chip select space is assigned a wait input pin. You can use the wait pin mapping bits to change the default assignment. 0x0 = Use the EM_WAIT0 pin. 0x1 = Use the EM_WAIT1 pin. 0x2 = Use the EM_WAIT2 pin. 0x3 = Use the EM_WAIT3 pin.

Table 4-40. AWCCR2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-2	CS3_WAIT	R/W	1h	Wait pin mapping bits for EMIF CS3. By default, each asynchronous chip select space is assigned a wait input pin. You can use the wait pin mapping bits to change the default assignment. 0x0 = Use the EM_WAIT0 pin. 0x1 = Use the EM_WAIT1 pin. 0x2 = Use the EM_WAIT2 pin. 0x3 = Use the EM_WAIT3 pin.
1-0	CS2_WAIT	R/W	0h	Wait pin mapping bits for EMIF CS2. By default, each asynchronous chip select space is assigned a wait input pin. You can use the wait pin mapping bits to change the default assignment. 0x0 = Use the EM_WAIT0 pin. 0x1 = Use the EM_WAIT1 pin. 0x2 = Use the EM_WAIT2 pin. 0x3 = Use the EM_WAIT3 pin.

4.4.5 SDCR1 Register (offset = 1008h) [reset = 620h]

SDCR1 is shown in Figure 4-27 and described in Table 4-41.

SDRAM Configuration Register 1

Figure 4-27. SDCR1 Register

15	14	13	12	11	10	9	8
Reserved	NM	Reserved		CL			BIT_11_9_LOCK
R-0h	R/W-0h	R-0h		R/W-3h			R/W-0h

7	6	5	4	3	2	1	0
Reserved	IBANK			EBANK	PAGESIZE		
R-0h	R/W-2h			R/W-0h	R/W-0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-41. SDCR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	Reserved
14	NM	R/W	0h	Narrow mode bit. This bit should always be set to 1. A write to this field will cause the EMIF to start the SDRAM initialization sequence.
13-12	Reserved	R	0h	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
11-9	CL	R/W	3h	CAS latency. This field defines the CAS latency to be used when accessing connected SDRAM devices. A 1 must be simultaneously written to the BIT_11_9_LOCK bit field of this register in order to write to the CL bit field. Writing to this field triggers the SDRAM initialization sequence. 0x0 = Reserved. 0x1 = Reserved. 0x2 = CAS latency set to 2 EM_SDCLK cycles. 0x3 = CAS latency set to 3 EM_SDCLK cycles. 0x4 = Reserved. 0x5 = Reserved. 0x6 = Reserved. 0x7 = Reserved.
8	BIT_11_9_LOCK	R/W	0h	Bits 11 to 9 lock. CL can only be written if BIT_11_9_LOCK is simultaneously written with a 1. BIT_11_9_LOCK is always read as 0. 0x0 = Writes to bits 11 through 9 are disabled. 0x1 = Writes to bits 11 through 9 are enabled.
7	Reserved	R	0h	Reserved
6-4	IBANK	R/W	2h	Internal SDRAM Bank size. This field defines number of banks inside the connected SDRAM devices. Writing to this field triggers the SDRAM initialization sequence. 0x0 = 1 bank SDRAM devices. 0x1 = 2 bank SDRAM devices. 0x2 = 4 bank SDRAM devices. 0x3 = Reserved. 0x4 = Reserved. 0x5 = Reserved. 0x6 = Reserved. 0x7 = Reserved.

Table 4-41. SDCR1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	EBANK	R/W	0h	External chip select setup. Defines whether SDRAM accesses will use 1 or 2 chip select lines. A write to this field will cause the EMIF to start the SDRAM initialization sequence. 0x0 = EMIF will use SD_CE0 for all SDRAM accesses. 0x1 = EMIF will use SD_CE0 and SD_CE1 for all SDRAM accesses.
2-0	PAGESIZE	R/W	0h	Page size. This field defines the internal page size of connected SDRAM devices. Writing to this field triggers the SDRAM initialization sequence. 0x0 = 8 column address bits (256 elements per row) 0x1 = 9 column address bits (512 elements per row) 0x2 = 10 column address bits (1024 elements per row) 0x3 = 11 column address bits (2048 elements per row) 0x4 = Reserved. 0x5 = Reserved. 0x6 = Reserved. 0x7 = Reserved.

4.4.6 SDCR2 Register (offset = 1009h) [reset = 0h]

SDCR2 is shown in Figure 4-28 and described in Table 4-42.

SDRAM Configuration Register 2

Figure 4-28. SDCR2 Register

15	14	13	12	11	10	9	8
SR	PD	PDWR	Reserved			PASR	
R/W-0h	R/W-0h	R/W-0h	R-0h			R/W-0h	

7	6	5	4	3	2	1	0
PASR	ROWSIZE			IBANK_POS	SDRAM_DRIVE		BIT_9_1_LOCK
R/W-0h	R/W-0h			R/W-0h	R/W-0h		R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-42. SDCR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	SR	R/W	0h	Self-Refresh mode bit. This bit controls entering and exiting of the Self-Refresh mode. The field should be written using a byte-write to the upper byte of SDCR to avoid triggering the SDRAM initialization sequence. 0x0 = Writing a 0 to this bit will cause connected SDRAM devices and the EMIF to exit the self-refresh mode. 0x1 = Writing a 1 to this bit will cause connected SDRAM devices and the EMIF to enter the self-refresh mode.
14	PD	R/W	0h	Power down bit. This bit controls entering and exiting of the power down mode. The field should be written using a byte-write to the upper byte of SDCR to avoid triggering the SDRAM initialization sequence. If both SR and PD bits are set, the EMIF will go into Self Refresh. 0x0 = Writing a 0 to this bit will cause connected SDRAM devices and the EMIF to exit the power down mode. 0x1 = Writing a 1 to this bit will cause connected SDRAM devices and the EMIF to enter the power down mode.
13	PDWR	R/W	0h	Perform refreshes during power down bit. Writing a 1 to this bit will cause EMIF to exit power down state and issue and AUTO REFRESH command every time Refresh May level is set. 0x0 = Do not perform refreshes during power down. 0x1 = Perform refreshes during power down.
12-10	Reserved	R	0h	Reserved
9-7	PASR	R/W	0h	Partial array self refresh. These bits get loaded into the Extended Mode Register of a mobile SDR during initialization. A write to this field will cause the EMIF to start the SDRAM initialization sequence. To write to this field you must simultaneously set BIT_9_1_LOCK to 1. 0x0 = 4 banks will be refreshed. 0x1 = 2 banks will be refreshed. 0x2 = 1 bank will be refreshed. 0x3 = Reserved. 0x4 = Reserved. 0x5 = Half a bank will be refreshed. 0x6 = One quarter of a bank will be refreshed. 0x7 = Reserved.

Table 4-42. SDCR2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6-4	ROWSIZE	R/W	0h	Row size. Defines the number of row address bits of connected SDRAM devices. This field is only used when IBANK_POS is set to 1. A write to this field will cause the EMIF to start the SDRAM initialization sequence. To write to this field you must simultaneously set BIT_9_1_LOCK to 1. 0x0 = 9 row address bits. 0x1 = 10 row address bits. 0x2 = 11 row address bits. 0x3 = 12 row address bits. 0x4 = 13 row address bits. 0x5 = 14 row address bits. 0x6 = Reserved. 0x7 = Reserved.
3	IBANK_POS	R/W	0h	Internal bank position. This bit specifies the mapping of the logical address to the SDRAM address. A write to this field will cause the EMIF to start the SDRAM initialization sequence. To write to this field you must simultaneously set BIT_9_1_LOCK to 1.
2-1	SDRAM_DRIVE	R/W	0h	SDRAM drive strength. These bits specify the the SDRAM drive strength configuration the EMIF uses while initializing the SDRAM device. A write to this field will cause the EMIF to start the SDRAM initialization sequence. To write to this field you must simultaneously set BIT_9_1_LOCK to 1. 0x0 = Full drive strength. 0x1 = Half drive strength. 0x2 = One fourth drive strength. 0x3 = One eighth drive strength.
0	BIT_9_1_LOCK	R/W	0h	Bits 9 to 1 lock. Bits 9 through 1 can only be written if BIT_9_1_LOCK is simultaneously written with a 1. BIT_9_1_LOCK is always read as a 0. 0x0 = Writes to bits 9 through 1 are disabled. 0x1 = Writes to bits 9 through 1 are enabled.

4.4.7 SDRCR Register (offset = 100Ch) [reset = 7Dh]

SDRCR is shown in Figure 4-29 and described in Table 4-43.

SDRAM Refresh Control Register

Figure 4-29. SDRCR Register

15	14	13	12	11	10	9	8
Reserved			REFRATE				
R-0h			R/W-7Dh				
7	6	5	4	3	2	1	0
REFRATE							
R/W-7Dh							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-43. SDRCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved
12-0	REFRATE	R/W	7Dh	Refresh Rate. This field is used to define the SDRAM refresh period in terms of EM_SDCLK cycles. Writing a value less than 0x0020 to this field will cause it to be loaded with (2 * T_RFC) + 1 value from SDRAM timing register.

4.4.8 ACS2CR1 Register (offset = 1010h) [reset = 7FFCh]

ACS2CR1 is shown in Figure 4-30 and described in Table 4-44.

Asynchronous CS2 Configuration Register 1

Figure 4-30. ACS2CR1 Register

15	14	13	12	11	10	9	8
RSETUPLSB			RSTROBE				
R/W-3h			R/W-3Fh				

7	6	5	4	3	2	1	0
RSTROBE	RHOLD			TA		ASIZE	
R/W-3Fh	R/W-7h			R/W-3h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-44. ACS2CR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	RSETUPLSB	R/W	3h	These bits in conjunction with RSETUPMSB in NCE0CR2 define the read setup timing in EM_SDCLK cycles, minus one cycle.
12-7	RSTROBE	R/W	3Fh	Read strobe width in EM_SDCLK cycles, minus one cycle.
6-4	RHOLD	R/W	7h	Read hold width in EM_SDCLK cycles, minus one cycle.
3-2	TA	R/W	3h	Minimum turn-around time. This field defines the minimum number of EM_SDCLK cycles between reads and writes, minus one cycle.
1-0	ASIZE	R/W	0h	Asynchronous data bus width. This field defines the width of the asynchronous device's data bus. 0x0 = 8 bit data bus. 0x1 = 16 bit data bus. 0x2 = Reserved. 0x3 = Reserved.

4.4.9 ACS2CR2 Register (offset = 1011h) [reset = 3FFFh]

ACS2CR2 is shown in Figure 4-31 and described in Table 4-45.

Asynchronous CS2 Configuration Register 2

Figure 4-31. ACS2CR2 Register

15	14	13	12	11	10	9	8
SS	EW	WSETUP				WSTROBE	
R/W-0h	R/W-0h	R/W-Fh				R/W-3Fh	

7	6	5	4	3	2	1	0
WSTROBE				WHOLD			RSETUPMSB
R/W-3Fh				R/W-7h			R/W-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-45. ACS2CR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	SS	R/W	0h	Select Strobe bit. This bit defines whether the asynchronous interface operates in WE Strobe Mode or Select Strobe Mode. 0x0 = WE Strobe Mode enabled. 0x1 = Select Strobe Mode enabled.
14	EW	R/W	0h	Extended wait cycles enable bit. This bit defines whether extended wait cycles will be enabled. 0x0 = Extended wait cycles disabled. 0x1 = Extended wait cycles enabled.
13-10	WSETUP	R/W	Fh	Write setup width in EM_SDCLK cycles, minus one cycle.
9-4	WSTROBE	R/W	3Fh	Write strobe width in EM_SDCLK cycles, minus one cycle.
3-1	WHOLD	R/W	7h	Write hold width in EM_SDCLK cycles, minus one cycle.
0	RSETUPMSB	R/W	1h	These bits in conjunction with RSETUPLSB in NCE0CR1 define the read setup timing in EM_SDCLK cycles, minus one cycle.

4.4.10 ACS3CR1 Register (offset = 1014h) [reset = 7FFDh]

ACS3CR1 is shown in Figure 4-32 and described in Table 4-46.

Asynchronous CS3 Configuration Register 1

Figure 4-32. ACS3CR1 Register

15	14	13	12	11	10	9	8
RSETUPLSB			RSTROBE				
R/W-3h			R/W-3Fh				
7	6	5	4	3	2	1	0
RSTROBE	RHOLD			TA		ASIZE	
R/W-3Fh	R/W-7h			R/W-3h		R/W-1h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-46. ACS3CR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	RSETUPLSB	R/W	3h	These bits in conjunction with RSETUPMSB in NCE1CR2 define the read setup timing in EM_SDCLK cycles, minus one cycle.
12-7	RSTROBE	R/W	3Fh	Read strobe width in EM_SDCLK cycles, minus one cycle.
6-4	RHOLD	R/W	7h	Read hold width in EM_SDCLK cycles, minus one cycle.
3-2	TA	R/W	3h	Minimum turn-around time. This field defines the minimum number of EM_SDCLK cycles between reads and writes, minus one cycle.
1-0	ASIZE	R/W	1h	Asynchronous data bus width. This field defines the width of the asynchronous device's data bus. 0x0 = 8 bit data bus. 0x1 = 16 bit data bus. 0x2 = Reserved. 0x3 = Reserved.

4.4.11 ACS3CR2 Register (offset = 1015h) [reset = 3FFFh]

ACS3CR2 is shown in Figure 4-33 and described in Table 4-47.

Asynchronous CS3 Configuration Register 2

Figure 4-33. ACS3CR2 Register

15	14	13	12	11	10	9	8
SS	EW	WSETUP				WSTROBE	
R/W-0h	R/W-0h	R/W-Fh				R/W-3Fh	

7	6	5	4	3	2	1	0
WSTROBE				WHOLD			RSETUPMSB
R/W-3Fh				R/W-7h			R/W-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-47. ACS3CR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	SS	R/W	0h	Select Strobe bit. This bit defines whether the asynchronous interface operates in WE Strobe Mode or Select Strobe Mode. 0x0 = WE Strobe Mode enabled. 0x1 = Select Strobe Mode enabled.
14	EW	R/W	0h	Extended wait cycles enable bit. This bit defines whether extended wait cycles will be enabled. 0x0 = Extended wait cycles disabled. 0x1 = Extended wait cycles enabled.
13-10	WSETUP	R/W	Fh	Write setup width in EM_SDCLK cycles, minus one cycle.
9-4	WSTROBE	R/W	3Fh	Write strobe width in EM_SDCLK cycles, minus one cycle.
3-1	WHOLD	R/W	7h	Write hold width in EM_SDCLK cycles, minus one cycle.
0	RSETUPMSB	R/W	1h	These bits in conjunction with RSETUPLSB in NCE1CR1 define the read setup timing in EM_SDCLK cycles, minus one cycle.

4.4.12 ACS4CR1 Register (offset = 1018h) [reset = 7FFCh]

ACS4CR1 is shown in Figure 4-34 and described in Table 4-48.

Asynchronous CS4 Configuration Register 1

Figure 4-34. ACS4CR1 Register

15	14	13	12	11	10	9	8
RSETUPLSB			RSTROBE				
R/W-3h			R/W-3Fh				
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RSTROBE	RHOLD			TA		ASIZE	
R/W-3Fh	R/W-7h			R/W-3h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-48. ACS4CR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	RSETUPLSB	R/W	3h	These bits in conjunction with RSETUPMSB in ACE0CR2 define the read setup timing in EM_SDCLK cycles, minus one cycle.
12-7	RSTROBE	R/W	3Fh	Read strobe width in EM_SDCLK cycles, minus one cycle.
6-4	RHOLD	R/W	7h	Read hold width in EM_SDCLK cycles, minus one cycle.
3-2	TA	R/W	3h	Minimum turn-around time. This field defines the minimum number of EM_SDCLK cycles between reads and writes, minus one cycle.
1-0	ASIZE	R/W	0h	Asynchronous data bus width. This field defines the width of the asynchronous device's data bus. 0x0 = 8 bit data bus. 0x1 = 16 bit data bus. 0x2 = Reserved. 0x3 = Reserved.

4.4.13 ACS4CR2 Register (offset = 1019h) [reset = 3FFFh]

ACS4CR2 is shown in Figure 4-35 and described in Table 4-49.

Asynchronous CS4 Configuration Register 2

Figure 4-35. ACS4CR2 Register

15	14	13	12	11	10	9	8
SS	EW	WSETUP				WSTROBE	
R/W-0h	R/W-0h	R/W-Fh				R/W-3Fh	

7	6	5	4	3	2	1	0
WSTROBE				WHOLD			RSETUPMSB
R/W-3Fh				R/W-7h			R/W-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-49. ACS4CR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	SS	R/W	0h	Select Strobe bit. This bit defines whether the asynchronous interface operates in WE Strobe Mode or Select Strobe Mode. 0x0 = WE Strobe Mode enabled. 0x1 = Select Strobe Mode enabled.
14	EW	R/W	0h	Extended wait cycles enable bit. This bit defines whether extended wait cycles will be enabled. 0x0 = Extended wait cycles disabled. 0x1 = Extended wait cycles enabled.
13-10	WSETUP	R/W	Fh	Write setup width in EM_SDCLK cycles, minus one cycle.
9-4	WSTROBE	R/W	3Fh	Write strobe width in EM_SDCLK cycles, minus one cycle.
3-1	WHOLD	R/W	7h	Write hold width in EM_SDCLK cycles, minus one cycle.
0	RSETUPMSB	R/W	1h	These bits in conjunction with RSETUPLSB in ACE0CR1 define the read setup timing in EM_SDCLK cycles, minus one cycle.

4.4.14 ACS5CR1 Register (offset = 101Ch) [reset = 7FFDh]

ACS5CR1 is shown in Figure 4-36 and described in Table 4-50.

Asynchronous CS5 Configuration Register 1

Figure 4-36. ACS5CR1 Register

15	14	13	12	11	10	9	8
RSETUPLSB			RSTROBE				
R/W-3h			R/W-3Fh				

7	6	5	4	3	2	1	0
RSTROBE	RHOLD			TA		ASIZE	
R/W-3Fh	R/W-7h			R/W-3h		R/W-1h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-50. ACS5CR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	RSETUPLSB	R/W	3h	These bits in conjunction with RSETUPMSB in ACE1CR2 define the read setup timing in EM_SDCLK cycles, minus one cycle.
12-7	RSTROBE	R/W	3Fh	Read strobe width in EM_SDCLK cycles, minus one cycle.
6-4	RHOLD	R/W	7h	Read hold width in EM_SDCLK cycles, minus one cycle.
3-2	TA	R/W	3h	Minimum turn-around time. This field defines the minimum number of EM_SDCLK cycles between reads and writes, minus one cycle.
1-0	ASIZE	R/W	1h	Asynchronous data bus width. This field defines the width of the asynchronous device's data bus. 0x0 = 8 bit data bus. 0x1 = 16 bit data bus. 0x2 = Reserved. 0x3 = Reserved.

4.4.15 ACS5CR2 Register (offset = 101Dh) [reset = 3FFFh]

ACS5CR2 is shown in Figure 4-37 and described in Table 4-51.

Asynchronous CS5 Configuration Register 2

Figure 4-37. ACS5CR2 Register

15	14	13	12	11	10	9	8
SS	EW	WSETUP				WSTROBE	
R/W-0h	R/W-0h	R/W-Fh				R/W-3Fh	

7	6	5	4	3	2	1	0
WSTROBE				WHOLD			RSETUPMSB
R/W-3Fh				R/W-7h			R/W-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-51. ACS5CR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	SS	R/W	0h	Select Strobe bit. This bit defines whether the asynchronous interface operates in WE Strobe Mode or Select Strobe Mode. 0x0 = WE Strobe Mode enabled. 0x1 = Select Strobe Mode enabled.
14	EW	R/W	0h	Extended wait cycles enable bit. This bit defines whether extended wait cycles will be enabled. 0x0 = Extended wait cycles disabled. 0x1 = Extended wait cycles enabled.
13-10	WSETUP	R/W	Fh	Write setup width in EM_SDCLK cycles, minus one cycle.
9-4	WSTROBE	R/W	3Fh	Write strobe width in EM_SDCLK cycles, minus one cycle.
3-1	WHOLD	R/W	7h	Write hold width in EM_SDCLK cycles, minus one cycle.
0	RSETUPMSB	R/W	1h	These bits in conjunction with RSETUPLSB in ACE1CR1 define the read setup timing in EM_SDCLK cycles, minus one cycle.

4.4.16 SDTIMR1 Register (offset = 1020h) [reset = 5820h]

SDTIMR1 is shown in Figure 4-38 and described in Table 4-52.

SDRAM Timing Register 1

Figure 4-38. SDTIMR1 Register

15	14	13	12	11	10	9	8
TRAS				TRC			
R/W-5h				R/W-8h			
7	6	5	4	3	2	1	0
Reserved	TRRD			Reserved			
R-0h	R/W-2h			R-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-52. SDTIMR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	TRAS	R/W	5h	Specifies the Tras value of the SDRAM. This defines the minimum number of EM_SDCLK clock cycles from Activate (ACTV) to Precharge (PRE), minus one: T_RAS = (Tras/tEM_SDCLK) - 1
11-8	TRC	R/W	8h	Specifies the Trc value of the SDRAM. This defines the minimum number of EM_SDCLK clock cycles from Activate (ACTV) to Activate (ACTV), minus one: T_RC = (Trc/tEM_SDCLK) - 1
7	Reserved	R	0h	Reserved
6-4	TRRD	R/W	2h	Specifies the Trrd value of the SDRAM. This defines the minimum number of EM_SDCLK clock cycles from Activate (ACTV) to Activate (ACTV) for a different bank, minus one: T_RRD = (Trrd/tEM_SDCLK) - 1
3-0	Reserved	R	0h	Reserved

4.4.17 SDTIMR2 Register (offset = 1021h) [reset = 4221h]

SDTIMR2 is shown in Figure 4-39 and described in Table 4-53.

SDRAM Timing Register 1

Figure 4-39. SDTIMR2 Register

15	14	13	12	11	10	9	8
TRFC						TRP	
R/W-8h						R/W-2h	

7	6	5	4	3	2	1	0
Reserved	TRCD			Reserved	TWR		
R-0h	R/W-2h			R-0h	R/W-1h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-53. SDTIMR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-11	TRFC	R/W	8h	Specifies the Trfc value of the SDRAM. This defines the minimum number of EM_SDCLK cycles from Refresh (REFR) to Refresh (REFR), minus one: T_RFC = (Trfc/tEM_SDCLK) - 1
10-8	TRP	R/W	2h	Specifies the Trp value of the SDRAM. This defines the minimum number of EM_SDCLK cycles from Precharge (PRE) to Activate (ACTV) or Refresh (REFR) command, minus one: T_RP = (Trp/tEM_SDCLK) - 1
7	Reserved	R	0h	Reserved
6-4	TRCD	R/W	2h	Specifies the Trcd value of the SDRAM. This defines the minimum number of EM_SDCLK cycles from Active (ACTV) to Read (READ) or Write (WRT), minus one: T_RCD = (Trcd/tEM_SDCLK) - 1
3	Reserved	R	0h	Reserved
2-0	TWR	R/W	1h	Specifies the Twr value of the SDRAM. This defines the minimum number of EM_SDCLK cycles from last Write (WRT) to Precharge (PRE), minus one: T_WR = (Twr/tEM_SDCLK) - 1

4.4.18 SDSRETR Register (offset = 103Ch) [reset = 5h]

SDSRETR is shown in Figure 4-40 and described in Table 4-54.

SDRAM Self Refresh Exit Timing Register

Figure 4-40. SDSRETR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved			TXS				
R-0h			R/W-5h				

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-54. SDSRETR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0h	Reserved
4-0	TXS	R/W	5h	This field specifies the minimum number of EM_SDCLK cycles from Self-Refresh exit to any command, minus one. T_XS = Txsr / tEM_SDCLK - 1

4.4.19 EIRR Register (offset = 1040h) [reset = 0h]

EIRR is shown in Figure 4-41 and described in Table 4-55.

Interrupt Raw Register

Figure 4-41. EIRR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		WR				LT	AT
R-0h		R/W1C-0h				R/W1C-0h	R/W1C-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-55. EIRR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	Reserved
5-2	WR	R/W1C	0h	Wait rise bits. These bits are set to 1 by hardware when a rising edge is detected on the wait pins. The polarity bits of AWCCR2 have not effect on these bits. Writing a 1 will clear these bits as well as the WRMASKED bits in the interrupt masked register (EMIR). Writing a 0 has no effect.
1	LT	R/W1C	0h	Line trap. These bits are set to 1 by hardware to indicate illegal memory access. Writing a 1 will clear this bit as well as the LTMASKED bit in the interrupt masked register (EMIR). Writing a 0 has no effect.
0	AT	R/W1C	0h	Asynchronous timeout. This bit is set to 1 by hardware to indicate that during an extended asynchronous memory access cycle, a wait pin did not go inactive within the number of cycles defined by the MEWC field in asynchronous wait cycle configuration register 1 (AWCCR1). Writing a 1 will clear these bits as well as the ATMASKED bit in the interrupt masked register (EMIR). Writing a 0 has no effect.

4.4.20 EIMR Register (offset = 1044h) [reset = 0h]

EIMR is shown in Figure 4-42 and described in Table 4-56.

Interrupt Mask Register

Figure 4-42. EIMR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		WRMASKED				LTMASKED	ATMASKED
R-0h		R/W1C-0h				R/W1C-0h	R/W1C-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-56. EIMR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	Reserved
5-2	WRMASKED	R/W1C	0h	Masked wait rise. These bits are set to 1 by hardware when a rising edge is detected on the wait pins. The polarity bits of AWCCR2 have not effect on these bits. Writing a 1 will clear these bits as well as the WR bits in the interrupt raw register (EIRR). Writing a 0 has no effect.
1	LTMASKED	R/W1C	0h	Masked line trap. These bits are set to 1 by hardware to indicate illegal memory access, only if the LTMASKSET bit in the interrupt mask set register (EIMSR) is set to 1. Writing a 1 will clear these bits as well as the LT bit in the interrupt raw register (EIRR). Writing a 0 has no effect.
0	ATMASKED	R/W1C	0h	Masked asynchronous timeout. This bit is set to 1 by hardware to indicate that during an extended asynchronous memory access cycle, a wait pin did not go inactive within the number of cycles defined by the MEWC field in asynchronous wait cycle configuration register 1 (AWCCR1), only if the ATMASKSET bit in the interrupt mask set register (EIMSR) is set to 1. Writing a 1 will clear these bits as well as the AT bit in the interrupt raw register (EIRR). Writing a 0 has no effect.

4.4.21 EIMSR Register (offset = 1048h) [reset = 0h]

EIMSR is shown in Figure 4-43 and described in Table 4-57.

Interrupt Mask Set Register

Figure 4-43. EIMSR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		WRMASKSET				LTMASKSET	ATMASKSET
R-0h		R/W1S-0h				R/W1S-0h	R/W1S-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-57. EIMSR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	Reserved
5-2	WRMASKSET	R/W1S	0h	Mask set for WRMASKED bits in interrupt mask register (EIMR). Writing a 1 will enable the wait rise interrupts and set these bits as well as the WRMASKCLR bits in the interrupt mask clear register (EIMCR). Writing a 0 has no effect.
1	LTMASKSET	R/W1S	0h	Mask set for LTMASKED bit in interrupt mask register (EIMR). Writing a 1 will enable the interrupt and set this bit as well as the LTMASKCLR bit in the interrupt mask clear register (EIMCR). Writing a 0 has no effect.
0	ATMASKSET	R/W1S	0h	Mask set for ATMASKED bit in interrupt mask register (EIMR). Writing a 1 will enable the interrupt and set this bit as well as the ATMASKCLR bit in the interrupt mask clear register (EIMCR). Writing a 0 has no effect.

4.4.22 EIMCR Register (offset = 104Ch) [reset = 0h]

EIMCR is shown in Figure 4-44 and described in Table 4-58.

Interrupt Mask clear Register

Figure 4-44. EIMCR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		WRMASKCLR				LTMASKCLR	ATMASKCLR
R-0h		R/W1C-0h				R/W1C-0h	R/W1C-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-58. EIMCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	Reserved
5-2	WRMASKCLR	R/W1C	0h	Mask clear for WRMASKED bits in interrupt mask register (EIMR). Writing a 1 will disable the wait rise interrupts and clear these bits as well as the WRMASKSET bits in the interrupt mask set register (EIMSR). Writing a 0 has no effect.
1	LTMASKCLR	R/W1C	0h	Mask clear for LTMASKED bit in interrupt mask register (EIMR). Writing a 1 will disable the interrupt and clear this bit as well as the LTMASKSET bit in the interrupt mask set register (EIMSR). Writing a 0 has no effect.
0	ATMASKCLR	R/W1C	0h	Mask clear for ATMASKED bit in interrupt mask register (EIMR). Writing a 1 will disable the interrupt and clear this bit as well as the ATMASKSET bit in the interrupt mask set register (EIMSR). Writing a 0 has no effect.

4.4.23 NANDFCR Register

NANDFCR is shown in Figure 4-45 and described in Table 4-59.

NAND Flash Control Register

Figure 4-45. NANDFCR Register

15	14	13	12	11	10	9	8
Reserved		ADDR_CALC_ST	4BIT_ECC_START	CS5_ECC_START	CS4_ECC_START	CS3_ECC_START	CS2_ECC_START
R-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
Reserved		4BIT_ECC_SEL		CS5_USE_NAND	CS4_USE_NAND	CS3_USE_NAND	CS2_USE_NAND
R-0h		R/W-0h		R/W-0h	R/W-0h	R/W-1h	R/W-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-59. NANDFCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	Reserved	R	0h	Reserved
13	ADDR_CALC_ST	R/W	0h	NAND flash 4-Bit ECC error address and error value calculation start. Set to 1 to start 4-Bit ECC error address and error value calculation on read syndrome. This bit is cleared when any of the NAND flash error address registers or NAND flash error value registers are read. Writing a 0 has no effect.
12	4BIT_ECC_START	R/W	0h	NAND flash 4-Bit ECC start for the selected chip select. Set to 1 to start 4-Bit ECC calculation on data for NAND flash on chip select selected by 4BIT_ECC_SEL field. This bit is cleared when any of the NAND flash 4-Bit ECC registers are read. Writing a 0 has no effect.
11	CS5_ECC_START	R/W	0h	NAND flash 1-Bit ECC start for EMIF CS5. Set to 1 to start 1-Bit ECC calculation on data for NAND flash on EMIF CS5. This bit is cleared when NAND flash CS5 1-Bit ECC register is read. Writing a 0 has no effect.
10	CS4_ECC_START	R/W	0h	NAND flash 1-Bit ECC start for EMIF CS4. Set to 1 to start 1-Bit ECC calculation on data for NAND flash on EMIF CS4. This bit is cleared when NAND flash CS4 1-Bit ECC register is read. Writing a 0 has no effect.
9	CS3_ECC_START	R/W	0h	NAND flash 1-Bit ECC start for EMIF CS3. Set to 1 to start 1-Bit ECC calculation on data for NAND flash on EMIF CS3. This bit is cleared when NAND flash CS3 1-Bit ECC register is read. Writing a 0 has no effect.
8	CS2_ECC_START	R/W	0h	NAND flash 1-Bit ECC start for EMIF CS2. Set to 1 to start 1-Bit ECC calculation on data for NAND flash on EMIF CS2. This bit is cleared when NAND flash CS2 1-Bit ECC register is read. Writing a 0 has no effect.
7-6	Reserved	R	0h	Reserved
5-4	4BIT_ECC_SEL	R/W	0h	NAND flash 4-Bit ECC chip select selection. This field selects the chip select on which the 4-Bit ECC will be calculated. 0x0 = Select EMIF CS2 for 4-Bit ECC calculation. 0x1 = Select EMIF CS3 for 4-Bit ECC calculation. 0x2 = Select EMIF CS4 for 4-Bit ECC calculation. 0x3 = Select EMIF CS5 for 4-Bit ECC calculation.

Table 4-59. NANDFCR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	CS5_USE_NAND	R/W	0h	NAND flash mode for EM_CS5. Set to 1 if using NAND flash on EM_CS5. 0x0 = Not using NAND flash. 0x1 = Using NAND flash on chip select pin.
2	CS4_USE_NAND	R/W	0h	NAND flash mode for EM_CS4. Set to 1 if using NAND flash on EM_CS4. 0x0 = Not using NAND flash. 0x1 = Using NAND flash on chip select pin.
1	CS3_USE_NAND	R/W	1h	NAND flash mode for EM_CS3. Set to 1 if using NAND flash on EM_CS3. 0x0 = Not using NAND flash. 0x1 = Using NAND flash on chip select pin.
0	CS2_USE_NAND	R/W	1h	NAND flash mode for EM_CS2. Set to 1 if using NAND flash on EM_CS2. 0x0 = Not using NAND flash. 0x1 = Using NAND flash on chip select pin.

4.4.24 NANDFSR1 Register (offset = 1064h) [reset = 0h]

NANDFSR1 is shown in Figure 4-46 and described in Table 4-60.

NAND Flash Status Register 1

Figure 4-46. NANDFSR1 Register

15	14	13	12	11	10	9	8
Reserved				CORRSTATE			
R-0h				R-0h			

7	6	5	4	3	2	1	0
Reserved				WAITSTAT			
R-0h				R-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-60. NANDFSR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	Reserved	R	0h	Reserved
11-8	CORRSTATE	R	0h	4-Bit ECC state value when performing error address and error value calculation. 0x0 = No error. 0x1 = Errors cannot be corrected (five or more errors). 0x2 = Error correction complete (errors on bit 8 or 9). 0x3 = Error correction complete (error exists). 0x4 = Reserved. 0x5 = Calculating number of errors. 0x6 = Preparing for error search. 0x7 = Preparing for error search. 0x8 = Searching for errors. 0x9 = Reserved. 0x10 = Reserved. 0x11 = Reserved. 0x12 = Calculating error value. 0x13 = Calculating error value. 0x14 = Calculating error value. 0x15 = Calculating error value.
7-4	Reserved	R	0h	Reserved
3-0	WAITSTAT	R	0h	These bits shows the raw status of the four wait input signals (EM_WAIT[3:0]). The polarity bits in the asynchronous wait cycle configuration register 2 (AWCCR2) have no effect on these bits. WAITSTAT[0] corresponds to EM_WAIT[0], WAITSTAT[1] to EM_WAIT[1], and so on.

4.4.25 NANDFSR2 Register (offset = 1065h) [reset = 0h]

NANDFSR2 is shown in Figure 4-47 and described in Table 4-61.

NAND Flash Status Register 2

Figure 4-47. NANDFSR2 Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved						ERRNUM	
R-0h						R-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-61. NANDFSR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1-0	ERRNUM	R	0h	4-Bit ECC error number. This field shows the number for errors found after the error address calculation and error value calculation is done. 0x0 = 1 error found. 0x1 = 2 errors found. 0x2 = 3 errors found. 0x3 = 4 errors found.

4.4.26 PAGEMODCTRL1 Register (offset = 1068h) [reset = FCFEh]

PAGEMODCTRL1 is shown in Figure 4-48 and described in Table 4-62.

Page Mode Control Register 1

Figure 4-48. PAGEMODCTRL1 Register

15	14	13	12	11	10	9	8
CS3_PAGE_DELAY						CS3_PAGE_SIZE	CS3_PAGEMOD_EN
R/W-3Fh						R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
CS2_PAGE_DELAY						CS2_PAGE_SIZE	CS2_PAGEMOD_EN
R/W-3Fh						R/W-1h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-62. PAGEMODCTRL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	CS3_PAGE_DELAY	R/W	3Fh	Page access delay for NOR flash connected on EM_CS3. Number of EMIF clock cycles required for the page read data to be valid, minus one cycle. This value must not be set to 0.
9	CS3_PAGE_SIZE	R/W	0h	Page size for NOR flash connected on EM_CS3. 0x0 = 4-word page. 0x1 = 8-word page.
8	CS3_PAGEMOD_EN	R/W	0h	Page mode enable for NOR flash connected on EM_CS3. 0x0 = Disable page mode. 0x1 = Enable page mode.
7-2	CS2_PAGE_DELAY	R/W	3Fh	Page access delay for NOR flash connected on EM_CS2. Number of EMIF clock cycles required for the page read data to be valid, minus one cycle. This value must not be set to 0.
1	CS2_PAGE_SIZE	R/W	1h	Page size for NOR flash connected on EM_CS2. 0x0 = 4-word page. 0x1 = 8-word page.
0	CS2_PAGEMOD_EN	R/W	0h	Page mode enable for NOR flash connected on EM_CS2. 0x0 = Disable page mode. 0x1 = Enable page mode.

4.4.27 PAGEMODCTRL2 Register (offset = 1069h) [reset = FCFCh]

PAGEMODCTRL2 is shown in Figure 4-49 and described in Table 4-63.

Page Mode Control Register 2

Figure 4-49. PAGEMODCTRL2 Register

15	14	13	12	11	10	9	8
CS5_PAGE_DELAY						CS5_PAGE_SIZE	CS5_PAGEMOD_EN
R/W-3Fh						R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
CS4_PAGE_DELAY						CS4_PAGE_SIZE	CS4_PAGEMOD_EN
R/W-3Fh						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-63. PAGEMODCTRL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	CS5_PAGE_DELAY	R/W	3Fh	Page access delay for NOR flash connected on EM_CS5. Number of EMIF clock cycles required for the page read data to be valid, minus one cycle. This value must not be set to 0.
9	CS5_PAGE_SIZE	R/W	0h	Page size for NOR flash connected on EM_CS5. 0x0 = 4-word page. 0x1 = 8-word page.
8	CS5_PAGEMOD_EN	R/W	0h	Page mode enable for NOR flash connected on EM_CS5. 0x0 = Disable page mode. 0x1 = Enable page mode.
7-2	CS4_PAGE_DELAY	R/W	3Fh	Page access delay for NOR flash connected on EM_CS4. Number of EMIF clock cycles required for the page read data to be valid, minus one cycle. This value must not be set to 0.
1	CS4_PAGE_SIZE	R/W	0h	Page size for NOR flash connected on EM_CS4. 0x0 = 4-word page. 0x1 = 8-word page.
0	CS4_PAGEMOD_EN	R/W	0h	Page mode enable for NOR flash connected on EM_CS4. 0x0 = Disable page mode. 0x1 = Enable page mode.

4.4.28 NCS2ECC1 Register (offset = 1070h) [reset = 0h]

NCS2ECC1 is shown in Figure 4-50 and described in Table 4-64.

NAND Flash CS2 1-Bit ECC Register 1

Figure 4-50. NCS2ECC1 Register

15	14	13	12	11	10	9	8
Reserved				P2048E	P1024E	P512E	P256E
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
P128E	P64E	P32E	P16E	P8E	P4E	P2E	P1E
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-64. NCS2ECC1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	Reserved	R	0h	Reserved
11	P2048E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
10	P1024E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
9	P512E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
8	P256E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
7	P128E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
6	P64E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
5	P32E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
4	P16E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.

Table 4-64. NCS2ECC1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	P8E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
2	P4E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
1	P2E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
0	P1E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.

4.4.29 NCS2ECC2 Register (offset = 1071h) [reset = 0h]

NCS2ECC2 is shown in Figure 4-51 and described in Table 4-65.

NAND Flash CS2 1-Bit ECC Register 2

Figure 4-51. NCS2ECC2 Register

15	14	13	12	11	10	9	8
Reserved				P2048O	P1024O	P512O	P256O
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
P128O	P64O	P32O	P16O	P8O	P4O	P2O	P1O
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-65. NCS2ECC2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	Reserved	R	0h	Reserved
11	P2048O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
10	P1024O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
9	P512O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
8	P256O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
7	P128O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
6	P64O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
5	P32O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
4	P16O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.

Table 4-65. NCS2ECC2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	P8O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
2	P4O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
1	P2O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
0	P1O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.

4.4.30 NCS3ECC1 Register (offset = 1074h) [reset = 0h]

NCS3ECC1 is shown in Figure 4-52 and described in Table 4-66.

NAND Flash CS3 1-Bit ECC Register 1

Figure 4-52. NCS3ECC1 Register

15	14	13	12	11	10	9	8
Reserved				P2048E	P1024E	P512E	P256E
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
P128E	P64E	P32E	P16E	P8E	P4E	P2E	P1E
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-66. NCS3ECC1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	Reserved	R	0h	Reserved
11	P2048E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
10	P1024E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
9	P512E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
8	P256E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
7	P128E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
6	P64E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
5	P32E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
4	P16E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.

Table 4-66. NCS3ECC1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	P8E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
2	P4E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
1	P2E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
0	P1E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.

4.4.31 NCS3ECC2 Register (offset = 1075h) [reset = 0h]

NCS3ECC2 is shown in Figure 4-53 and described in Table 4-67.

NAND Flash CS3 1-Bit ECC Register 2

Figure 4-53. NCS3ECC2 Register

15	14	13	12	11	10	9	8
Reserved				P2048O	P1024O	P512O	P256O
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
P128O	P64O	P32O	P16O	P8O	P4O	P2O	P1O
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-67. NCS3ECC2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	Reserved	R	0h	Reserved
11	P2048O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
10	P1024O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
9	P512O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
8	P256O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
7	P128O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
6	P64O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
5	P32O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
4	P16O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.

Table 4-67. NCS3ECC2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	P8O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
2	P4O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
1	P2O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
0	P1O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.

4.4.32 NCS4ECC1 Register (offset = 1078h) [reset = 0h]

NCS4ECC1 is shown in Figure 4-54 and described in Table 4-68.

NAND Flash CS4 1-Bit ECC Register 1

Figure 4-54. NCS4ECC1 Register

15	14	13	12	11	10	9	8
Reserved				P2048E	P1024E	P512E	P256E
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
P128E	P64E	P32E	P16E	P8E	P4E	P2E	P1E
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-68. NCS4ECC1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	Reserved	R	0h	Reserved
11	P2048E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
10	P1024E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
9	P512E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
8	P256E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
7	P128E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
6	P64E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
5	P32E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
4	P16E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.

Table 4-68. NCS4ECC1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	P8E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
2	P4E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
1	P2E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
0	P1E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.

4.4.33 NCS4ECC2 Register (offset = 1079h) [reset = 0h]

NCS4ECC2 is shown in Figure 4-55 and described in Table 4-69.

NAND Flash CS4 1-Bit ECC Register 2

Figure 4-55. NCS4ECC2 Register

15	14	13	12	11	10	9	8
Reserved				P2048O	P1024O	P512O	P256O
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
P128O	P64O	P32O	P16O	P8O	P4O	P2O	P1O
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-69. NCS4ECC2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	Reserved	R	0h	Reserved
11	P2048O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
10	P1024O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
9	P512O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
8	P256O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
7	P128O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
6	P64O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
5	P32O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
4	P16O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.

Table 4-69. NCS4ECC2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	P8O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
2	P4O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
1	P2O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
0	P1O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.

4.4.34 NCS5ECC1 Register (offset = 107Ch) [reset = 0h]

NCS5ECC1 is shown in Figure 4-56 and described in Table 4-70.

NAND Flash CS5 1-Bit ECC Register 1

Figure 4-56. NCS5ECC1 Register

15	14	13	12	11	10	9	8
Reserved				P2048E	P1024E	P512E	P256E
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
P128E	P64E	P32E	P16E	P8E	P4E	P2E	P1E
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-70. NCS5ECC1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	Reserved	R	0h	Reserved
11	P2048E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
10	P1024E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
9	P512E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
8	P256E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
7	P128E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
6	P64E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
5	P32E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
4	P16E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.

Table 4-70. NCS5ECC1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	P8E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
2	P4E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
1	P2E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.
0	P1E	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1E, P2E, and P4E are column parities and P8E to P2048E are row parities. For 16 bit NAND Flash, P1E, P2E, P4E, and P8E are column parities and P16E to P2048E are row parities.

4.4.35 NCS5ECC2 Register (offset = 107Dh) [reset = 0h]

NCS5ECC2 is shown in Figure 4-57 and described in Table 4-71.

NAND Flash CS5 1-Bit ECC Register 2

Figure 4-57. NCS5ECC2 Register

15	14	13	12	11	10	9	8
Reserved				P2048O	P1024O	P512O	P256O
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
P128O	P64O	P32O	P16O	P8O	P4O	P2O	P1O
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-71. NCS5ECC2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	Reserved	R	0h	Reserved
11	P2048O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
10	P1024O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
9	P512O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
8	P256O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
7	P128O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
6	P64O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
5	P32O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
4	P16O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.

Table 4-71. NCS5ECC2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	P8O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
2	P4O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
1	P2O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.
0	P1O	R/W	0h	1-Bit ECC code calculated while reading/writing NAND Flash. For 8 bit NAND Flash, P1O, P2O, and P4O are column parities and P8O to P2048O are row parities. For 16 bit NAND Flash, P1O, P2O, P4O, and P8O are column parities and P16O to P2048O are row parities.

4.4.36 NAND4BITECCLOAD Register (offset = 10BCh) [reset = 0h]

NAND4BITECCLOAD is shown in Figure 4-58 and described in Table 4-72.

NAND Flash 4-Bit ECC Load Register

Figure 4-58. NAND4BITECCLOAD Register

15	14	13	12	11	10	9	8
Reserved						4BIT_ECC_LOAD	
R-0h						R/W-0h	

7	6	5	4	3	2	1	0
4BIT_ECC_LOAD							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-72. NAND4BITECCLOAD Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	4BIT_ECC_LOAD	R/W	0h	4-Bit ECC Load. This register is used to load 4-Bit ECC values when performing syndrome calculation during NAND Flash reads.

4.4.37 NAND4BITECC1 Register (offset = 10C0h) [reset = 0h]

NAND4BITECC1 is shown in Figure 4-59 and described in Table 4-73.

NAND Flash 4-Bit ECC Register 1

Figure 4-59. NAND4BITECC1 Register

15	14	13	12	11	10	9	8
Reserved						4BIT_ECC_VAL1	
R-0h						R-0h	

7	6	5	4	3	2	1	0
4BIT_ECC_VAL1							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-73. NAND4BITECC1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	4BIT_ECC_VAL1	R	0h	4-Bit ECC or syndrome value 1 calculated while writing or reading NAND Flash.

4.4.38 NAND4BITECC2 Register (offset = 10C1h) [reset = 0h]

NAND4BITECC2 is shown in Figure 4-60 and described in Table 4-74.

NAND Flash 4-Bit ECC Register 2

Figure 4-60. NAND4BITECC2 Register

15	14	13	12	11	10	9	8
Reserved						4BIT_ECC_VAL2	
R-0h						R-0h	

7	6	5	4	3	2	1	0
4BIT_ECC_VAL2							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-74. NAND4BITECC2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	4BIT_ECC_VAL2	R	0h	4-Bit ECC or syndrome value 2 calculated while writing or reading NAND Flash.

4.4.39 NAND4BITECC3 Register (offset = 10C4h) [reset = 0h]

NAND4BITECC3 is shown in Figure 4-61 and described in Table 4-75.

NAND Flash 4-Bit ECC Register 3

Figure 4-61. NAND4BITECC3 Register

15	14	13	12	11	10	9	8
Reserved						4BIT_ECC_VAL3	
R-0h						R-0h	

7	6	5	4	3	2	1	0
4BIT_ECC_VAL3							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-75. NAND4BITECC3 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	4BIT_ECC_VAL3	R	0h	4-Bit ECC or syndrome value 3 calculated while writing or reading NAND Flash.

4.4.40 NAND4BITECC4 Register (offset = 10C5h) [reset = 0h]

NAND4BITECC4 is shown in Figure 4-62 and described in Table 4-76.

NAND Flash 4-Bit ECC Register 4

Figure 4-62. NAND4BITECC4 Register

15	14	13	12	11	10	9	8
Reserved						4BIT_ECC_VAL4	
R-0h						R-0h	

7	6	5	4	3	2	1	0
4BIT_ECC_VAL4							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-76. NAND4BITECC4 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	4BIT_ECC_VAL4	R	0h	4-Bit ECC or syndrome value 4 calculated while writing or reading NAND Flash.

4.4.41 NAND4BITECC5 Register (offset = 10C8h) [reset = 0h]

NAND4BITECC5 is shown in Figure 4-63 and described in Table 4-77.

NAND Flash 4-Bit ECC Register 5

Figure 4-63. NAND4BITECC5 Register

15	14	13	12	11	10	9	8
Reserved						4BIT_ECC_VAL5	
R-0h						R-0h	

7	6	5	4	3	2	1	0
4BIT_ECC_VAL5							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-77. NAND4BITECC5 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	4BIT_ECC_VAL5	R	0h	4-Bit ECC or syndrome value 5 calculated while writing or reading NAND Flash.

4.4.42 NAND4BITECC6 Register (offset = 10C9h) [reset = 0h]

NAND4BITECC6 is shown in Figure 4-64 and described in Table 4-78.

NAND Flash 4-Bit ECC Register 6

Figure 4-64. NAND4BITECC6 Register

15	14	13	12	11	10	9	8
Reserved						4BIT_ECC_VAL6	
R-0h						R-0h	

7	6	5	4	3	2	1	0
4BIT_ECC_VAL6							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-78. NAND4BITECC6 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	4BIT_ECC_VAL6	R	0h	4-Bit ECC or syndrome value 6 calculated while writing or reading NAND Flash.

4.4.43 NAND4BITECC7 Register (offset = 10CCh) [reset = 0h]

NAND4BITECC7 is shown in Figure 4-65 and described in Table 4-79.

NAND Flash 4-Bit ECC Register 7

Figure 4-65. NAND4BITECC7 Register

15	14	13	12	11	10	9	8
Reserved						4BIT_ECC_VAL7	
R-0h						R-0h	

7	6	5	4	3	2	1	0
4BIT_ECC_VAL7							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-79. NAND4BITECC7 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	4BIT_ECC_VAL7	R	0h	4-Bit ECC or syndrome value 7 calculated while writing or reading NAND Flash.

4.4.44 NAND4BITECC8 Register (offset = 10CDh) [reset = 0h]

NAND4BITECC8 is shown in Figure 4-66 and described in Table 4-80.

NAND Flash 4-Bit ECC Register 8

Figure 4-66. NAND4BITECC8 Register

15	14	13	12	11	10	9	8
Reserved						4BIT_ECC_VAL8	
R-0h						R-0h	

7	6	5	4	3	2	1	0
4BIT_ECC_VAL8							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-80. NAND4BITECC8 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	4BIT_ECC_VAL8	R	0h	4-Bit ECC or syndrome value 8 calculated while writing or reading NAND Flash.

4.4.45 NANDERRADD1 Register (offset = 10D0h) [reset = 0h]

NANDERRADD1 is shown in Figure 4-67 and described in Table 4-81.

NAND Flash 4-Bit ECC Error Address Register 1

Figure 4-67. NANDERRADD1 Register

15	14	13	12	11	10	9	8
Reserved						ERR_ADDR1	
R-0h						R-0h	

7	6	5	4	3	2	1	0
ERR_ADDR1							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-81. NANDERRADD1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	ERR_ADDR1	R	0h	4-Bit ECC error address 1.

4.4.46 NANDERRADD2 Register (offset = 10D1h) [reset = 0h]

NANDERRADD2 is shown in Figure 4-68 and described in Table 4-82.

NAND Flash 4-Bit ECC Error Address Register 2

Figure 4-68. NANDERRADD2 Register

15	14	13	12	11	10	9	8
Reserved						ERR_ADDR2	
R-0h						R-0h	

7	6	5	4	3	2	1	0
ERR_ADDR2							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-82. NANDERRADD2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	ERR_ADDR2	R	0h	4-Bit ECC error address 2.

4.4.47 NANDERRADD3 Register (offset = 10D4h) [reset = 0h]

NANDERRADD3 is shown in Figure 4-69 and described in Table 4-83.

NAND Flash 4-Bit ECC Error Address Register 3

Figure 4-69. NANDERRADD3 Register

15	14	13	12	11	10	9	8
Reserved						ERR_ADDR3	
R-0h						R-0h	

7	6	5	4	3	2	1	0
ERR_ADDR3							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-83. NANDERRADD3 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	ERR_ADDR3	R	0h	4-Bit ECC error address 3.

4.4.48 NANDERRADD4 Register (offset = 10D5h) [reset = 0h]

NANDERRADD4 is shown in Figure 4-70 and described in Table 4-84.

NAND Flash 4-Bit ECC Error Address Register 4

Figure 4-70. NANDERRADD4 Register

15	14	13	12	11	10	9	8
Reserved						ERR_ADDR4	
R-0h						R-0h	

7	6	5	4	3	2	1	0
ERR_ADDR4							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-84. NANDERRADD4 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	ERR_ADDR4	R	0h	4-Bit ECC error address 4.

4.4.49 NANDERRVAL1 Register (offset = 10D8h) [reset = 0h]

NANDERRVAL1 is shown in Figure 4-71 and described in Table 4-85.

NAND Flash 4-Bit ECC Error Value Register 1

Figure 4-71. NANDERRVAL1 Register

15	14	13	12	11	10	9	8
Reserved						ERR_VALUE1	
R-0h						R-0h	

7	6	5	4	3	2	1	0
ERR_VALUE1							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-85. NANDERRVAL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	ERR_VALUE1	R	0h	4-Bit ECC error value 1.

4.4.50 NANDERRVAL2 Register (offset = 10D9h) [reset = 0h]

NANDERRVAL2 is shown in Figure 4-72 and described in Table 4-86.

NAND Flash 4-Bit ECC Error Value Register 2

Figure 4-72. NANDERRVAL2 Register

15	14	13	12	11	10	9	8
Reserved						ERR_VALUE2	
R-0h						R-0h	

7	6	5	4	3	2	1	0
ERR_VALUE2							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-86. NANDERRVAL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	ERR_VALUE2	R	0h	4-Bit ECC error value 2.

4.4.51 NANDERRVAL3 Register (offset = 10DCh) [reset = 0h]

NANDERRVAL3 is shown in Figure 4-73 and described in Table 4-87.

NAND Flash 4-Bit ECC Error Value Register 3

Figure 4-73. NANDERRVAL3 Register

15	14	13	12	11	10	9	8
Reserved						ERR_VALUE3	
R-0h						R-0h	

7	6	5	4	3	2	1	0
ERR_VALUE3							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-87. NANDERRVAL3 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	ERR_VALUE3	R	0h	4-Bit ECC error value 3.

4.4.52 NANDERRVAL4 Register (offset = 10DDh) [reset = 0h]

NANDERRVAL4 is shown in Figure 4-74 and described in Table 4-88.

NAND Flash 4-Bit ECC Error Value Register 4

Figure 4-74. NANDERRVAL4 Register

15	14	13	12	11	10	9	8
Reserved						ERR_VALUE4	
R-0h						R-0h	

7	6	5	4	3	2	1	0
ERR_VALUE4							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-88. NANDERRVAL4 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	ERR_VALUE4	R	0h	4-Bit ECC error value 4.

General-Purpose Input/Output (GPIO)

Topic **Page**

5.1 Introduction

5.1.1 Purpose of the Peripheral

The GPIO peripheral provides general-purpose pins that can be configured as either inputs or outputs. When configured as an output, you can write to an internal register to control the state driven on the output pin. When configured as an input, you can detect the state of the input by reading the state of the internal register. The GPIO can also be used to send interrupts to the CPU.

External input clocks on certain GPIOs can also be used to drive the Timers on this device.

5.1.2 Features

The GPIO peripheral supports the following:

- 32 GPIOs plus 1 general-purpose output (XF) and 4 Special-Purpose Outputs for Use With SAR
  - Configure up to 26 GPIO pins simultaneously.
- Each GPIO pin has internal pulldowns (IPDs) which can be individually disabled.
- Each GPIO pin can be configured to generate edge detected interrupts to the CPU on either the rising or falling edge.

5.1.3 Industry Standard(s) Compliance Statement

The GPIO peripheral connects to external devices. While it is possible that the software implements some standard connectivity protocol over GPIO, the GPIO peripheral itself is not compliant with any such standards.

5.2 Peripheral Architecture

The following sections describe the GPIO peripheral.

5.2.1 Clock Control

The input clock to the GPIO peripheral is driven by the system clock.

5.2.2 Signal Descriptions

The device supports up to 26 signals, GPIO[31:27], GPIO[20:0], simultaneously. All GPIO pins are muxed with other signals and have an optional internal pull-down resistor. The mux is controlled in the External Bus Selection Register (EBSR) located at port address 1C00h. The routing of the signals takes place on the next CPU clock cycle. Before modifying the values of EBSR, you must first clock gated all affected peripherals via the Peripheral Clock Gating Control Register (PCGCR1 and PCGCR2) at addresses 1C02h and 1C03h.

The EBSR can be modified only once after boot process is complete. Continuously switching the EBSR is not supported. Pulldowns are disabled or enabled by the Pulldown Inhibit Register (1C17h, 1C18h and 1C19h). See Section 1.7.3.6, *Pullup/Pulldown Inhibit Registers (PDINHIBR1–7)*. These GPIO are muxed with other signals. For more information on the package pinout and muxing of each GPIO signal, see the device-specific data manual.

Due to a variety of different technology devices that can be connected to the GPIO, the levels of logic 0 (low) and logic 1 (high) are not fixed and depend on the supply level. See the device-specific data manual for more information.

5.2.3 GPIO Register Structure

The GPIO configuration registers are grouped into two 16-bit registers for each function. Control of the general-purpose I/O is maintained through a set of I/O memory mapped registers. For detailed information on the GPIO registers, see Section 5.3.

5.2.4 Using a GPIO Signal as an Output

GPIO signals are configured to operate as inputs or outputs by writing the appropriate value to the GPIO direction registers (IODIR1 and IODIR2). This section describes using the GPIO signal as an output.

5.2.4.1 Configuring a GPIO Output Signal

To configure a given GPIO signal as an output, set the bit in (IODIR1 and IODIR2) that is associated with the desired GPIO signal. For detailed information on the GPIO direction registers, see Section 5.3.1 and Section 5.3.2.

5.2.4.2 Controlling the GPIO Output Signal State

The GPIO output is controlled by the setting or clearing the OUT bit in the GPIO data out registers (IODATAOUT1 or IODATAOUT2) for the desired GPIO. When the GPIO is configured for output, a write of "1" will make the output high and a write of "0" will make the output low. For detailed information on the GPIO data out registers, see and .

5.2.5 Using a GPIO Signal as an Input

GPIO signals are configured to operate as inputs or outputs by writing the appropriate value to the IODIR1 or IODIR2 registers. This section describes using the GPIO signal as an input.

5.2.5.1 Configuring a GPIO Input Signal

To configure a given GPIO signal as an input, clear the bit in the GPIO direct register (IODIR1 or IODIR2) that is associated with the desired GPIO signal. For detailed information on GPIO direction registers, see and .

5.2.5.2 Controlling the GPIO Input Signal State

The current state of the GPIO signals are read using the GPIO data in registers (IOINDATA1 & IOINDATA2).

- For GPIO signals configured as inputs, reading input data register returns the state of the input signal synchronized to the GPIO peripheral (system clock).
- For GPIO signals configured as outputs, reading input data register returns the output value being driven by the device.

To use GPIO input signals as interrupt sources, see Section 5.2.7.

For detailed information on GPIO data in registers, see and .

5.2.6 Reset Considerations

The GPIO peripheral is only reset by a hardware reset.

5.2.6.1 Software Reset Considerations

A software reset does not modify the configuration and state of the GPIO signals. Software resets include reset initiated through the emulator, the device's software reset instruction, and the Peripheral Reset Control Register (PRCR) (1C05h). For a detailed description, see Section 1.7.5.2, *Peripheral Reset Control Register (PRCR) [1C05h]*.

5.2.6.2 Hardware Reset Considerations

A hardware reset will cause the GPIO configuration to return to the default state, including GPIO pin selection (GPIOs default to input), and data registers to their default states, therefore affecting the configuration and state of the GPIO signals.

5.2.7 Interrupt Support

GPIO peripheral can individually generate an interrupt. The GPIO interrupts are falling or rising edge triggered interrupts.

5.2.7.1 Interrupt Events and Requests

The GPIO signals can be configured to generate an interrupt. The device supports interrupts from the GPIO signals. All GPIO are tied a single interrupt. To determine which GPIO caused the interrupt, the GPIO interrupt flag registers (IOINTFLG1 and IOINTFLG2) must be read. For detailed information on the GPIO interrupt flag registers, see and .

5.2.7.2 Enabling GPIO Interrupt Events

GPIO interrupt events are enabled in GPIO interrupt enable registers (IOINTEN1 and IOINTEN2). Interrupts can be enabled for each of the GPIO. Setting a "1" to the appropriate GPIO will enable external interrupts for this pin. For detailed information on GPIO interrupt enable registers, see and .

5.2.7.3 Configuring GPIO Interrupt Edge Triggering

Each GPIO interrupt source can be configured to generate an interrupt on the rising or the falling edge. The edge detection is synchronized to the GPIO peripheral module clock. This is controlled in the GPIO interrupt edge trigger enable registers (IOINTEDG1 and IOINTEDG2). Setting a "0" to this register will use Rising edge to trigger the interrupt and a "1" will use Falling Edge to trigger the interrupt if interrupts are enabled for this GPIO. For detailed information on the GPIO interrupt edge trigger enable registers, see and .

5.2.7.4 GPIO Interrupt Status

When an interrupt occurs on an enabled GPIO pin, the GPIO interrupt flag registers (IOINTFLG1 and IOINTFLG2) latch the corresponding bit to a "1". The interrupt signal to the CPU will be kept low until all flag bits in the IOINTFLG1 and IOINTFLG2 registers are cleared. To clear the flag you must write a "1" to the corresponding bit.

The status of the GPIO interrupt events can be monitored by reading the IOINTFLG1 and IOINTFLG2 registers. If a GPIO interrupt event has occurred the corresponding bit will be a set to a “1”. In the case of a non-interrupt the corresponding bit will be “0”.

5.2.7.5 Interrupt Multiplexing

No GPIO interrupts are multiplexed with other interrupt functions on the device.

5.3 GPIO Registers

Table 5-1 lists the memory-mapped registers for the GPIO. All register offset addresses not listed in Table 5-1 should be considered as reserved locations and the register contents should not be modified.

Table 5-1. GPIO REGISTERS

CPU Word Address	Acronym	Register Name	Section
1C06h	IODIR1	GPIO Data Direction Register 1	Section 5.3.1
1C07h	IODIR2	GPIO Data Direction Register 2	Section 5.3.2
1C08h	IOINDATA1	GPIO Data In Register 1	Section 5.3.3
1C09h	IOINDATA2	GPIO Data In Register 2	Section 5.3.4
1C0Ah	IODATAOUT1	GPIO Data Out Register 1	Section 5.3.5
1C0Bh	IODATAOUT2	GPIO Data Out Register 2	Section 5.3.6
1C0Ch	IOINTEDG1	GPIO Interrupt Edge Trigger Select 1	Section 5.3.7
1C0Dh	IOINTEDG2	GPIO Interrupt Edge Trigger Select 2	Section 5.3.8
1C0Eh	IOINTEN1	GPIO Interrupt Enable 1	Section 5.3.9

Table 5-1. GPIO REGISTERS (continued)

CPU Word Address	Acronym	Register Name	Section
1C0Fh	IOINTEN2	GPIO Interrupt Enable 2	Section 5.3.10
1C10h	IOINTFLG1	GPIO Interrupt Flag 1	Section 5.3.11
1C11h	IOINTFLG2	GPIO Interrupt Flag 2	Section 5.3.12

5.3.1 IODIR1 Register (offset = 1C06h) [reset = 0h]

IODIR1 is shown in Figure 5-1 and described in Table 5-2.

The device includes two registers for controlling whether the GPIO is set as a general-purpose Input, or Output. Use the GPIO direction register (IODIR1 and IODIR2) to set the GPIO pin as Input or Output. Each of these registers control 16 of the 32 GPIOs. IODIR1 is used for GPIO[15:0] with bit 0 corresponding to GPIO 0 through bit 15 corresponding to GPIO 15. IODIR2 is used for GPIO[31:16] with bit 0 corresponding to GPIO 16 through bit 15 corresponding to GPIO 31. Writing a "1" to these bits configures the pin as an OUTPUT and writing a "0" configures the pin as an INPUT.

Figure 5-1. IODIR1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIR[15:0]															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 5-2. IODIR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DIR[15:0]	R/W	0h	Direction of GPIO Pin 15 to 0. 0x0 = Configure pin as input. 0x1 = Configure pin as output.

5.3.2 IODIR2 Register (offset = 1C07h) [reset = 0h]

IODIR2 is shown in Figure 5-2 and described in Table 5-3.

The device includes two registers for controlling whether the GPIO is set as a general-purpose Input, or Output. Use the GPIO direction register (IODIR1 and IODIR2) to set the GPIO pin as Input or Output. Each of these registers control 16 of the 32 GPIOs. IODIR1 is used for GPIO[15:0] with bit 0 corresponding to GPIO 0 through bit 15 corresponding to GPIO 15. IODIR2 is used for GPIO[31:16] with bit 0 corresponding to GPIO 16 through bit 15 corresponding to GPIO 31. Writing a "1" to these bits configures the pin as an OUTPUT and writing a "0" configures the pin as an INPUT.

Figure 5-2. IODIR2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIR[31:16]															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 5-3. IODIR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DIR[31:16]	R/W	0h	Direction of GPIO Pin 31 to 16. 0x0 = Configure pin as input. 0x1 = Configure pin as output.

5.3.3 IOINDATA1 Register (offset = 1C08h) [reset = 0h]

IOINDATA1 is shown in Figure 5-3 and described in Table 5-4.

The device includes two registers for reading in the GPIO values when they are configured as Inputs. Use the GPIO data in registers (IOINDATA1 and IOINDATA2) to read the state of the corresponding GPIO pin. Each of these registers control 16 of the 32 GPIOs. IOINDATA1 is used for GPIO[15:0] with bit 0 corresponding to GPIO 0 through bit 15 corresponding to GPIO15. IOINDATA2 is used for GPIO[31:16] with bit 0 corresponding to GPIO 16 through bit 15 corresponding to GPIO 31.

Figure 5-3. IOINDATA1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IN[15:0]															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 5-4. IOINDATA1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	IN[15:0]	R	0h	Status of GPIO bit 15 to 0. Data bits that are used to monitor the level of the GPIO pins configured as general-purpose input pins. If DIR = 1, then X reflects the value of the output pin. If DIR = 0, then 0 equals low input and 1 equals high input. 0x0 = Input is low 0x1 = Input is High

5.3.4 IOINDATA2 Register (offset = 1C09h) [reset = 0h]

IOINDATA2 is shown in Figure 5-4 and described in Table 5-5.

The device includes two registers for reading in the GPIO values when they are configured as Inputs. Use the GPIO data in registers (IOINDATA1 and IOINDATA2) to read the state of the corresponding GPIO pin. Each of these registers control 16 of the 32 GPIOs. IOINDATA1 is used for GPIO[15:0] with bit 0 corresponding to GPIO 0 through bit 15 corresponding to GPIO15. IOINDATA2 is used for GPIO[31:16] with bit 0 corresponding to GPIO 16 through bit 15 corresponding to GPIO 31.

Figure 5-4. IOINDATA2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IN[31:16]															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 5-5. IOINDATA2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	IN[31:16]	R	0h	Status of GPIO bit 31 to 16. Data bits that are used to monitor the level of the GPIO pins configured as general-purpose input pins. If DIR = 1, then X reflects the value of the output pin. If DIR = 0, then 0 equals low input and 1 equals high input. 0x0 = Input is low 0x1 = Input is High

5.3.5 IODATAOUT1 Register (offset = 1C0Ah) [reset = 0h]

IODATAOUT1 is shown in Figure 5-5 and described in Table 5-6.

The device includes two registers for writing to the GPIO pins when they are configured as outputs. Use the GPIO data out registers (IODATAOUT1 and IODATAOUT2) to change the state of the corresponding GPIO pin. Each of these registers control 16 of the 32 GPIOs. IODATAOUT1 is used for GPIO[15:0] with bit 0 corresponding to GPIO 0 through bit 15 corresponding to GPIO15. IODATAOUT2 is used for GPIO[31:16] with bit 0 corresponding to GPIO 16 through bit 15 corresponding to GPIO 31.

Figure 5-5. IODATAOUT1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUT[15:0]															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 5-6. IODATAOUT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	OUT[15:0]	R/W	0h	Output control for GPIO 15 to 0. Data bits that are used to monitor the level of the GPIO pins configured as general-purpose input pins. If DIR = 0, then X reflects the value of the output pin. If DIR = 1, then 0 equals low input and 1 equals high input. 0x0 = Output is low 0x1 = Output is High

5.3.6 IODATAOUT2 Register (offset = 1C0Bh) [reset = 0h]

IODATAOUT2 is shown in Figure 5-6 and described in Table 5-7.

The device includes two registers for writing to the GPIO pins when they are configured as outputs. Use the GPIO data out registers (IODATAOUT1 and IODATAOUT2) to change the state of the corresponding GPIO pin. Each of these registers control 16 of the 32 GPIOs. IODATAOUT1 is used for GPIO[15:0] with bit 0 corresponding to GPIO 0 through bit 15 corresponding to GPIO15. IODATAOUT2 is used for GPIO[31:16] with bit 0 corresponding to GPIO 16 through bit 15 corresponding to GPIO 31.

Figure 5-6. IODATAOUT2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUT[31:16]															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 5-7. IODATAOUT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	OUT[31:16]	R/W	0h	Output control for GPIO 31 to 16. Data bits that are used to monitor the level of the GPIO pins configured as general-purpose input pins. If DIR = 0, then X reflects the value of the output pin. If DIR = 1, then 0 equals low input and 1 equals high input. 0x0 = Output is low 0x1 = Output is High

5.3.7 IOINTEDG1 Register (offset = 1C0Ch) [reset = 0h]

IOINTEDG1 is shown in Figure 5-7 and described in Table 5-8.

The device has two registers for configuring interrupts to trigger on the rising or falling edge of the input signal to the GPIO pins if they are configured as inputs with Interrupt enabled for the chosen GPIO. Use the GPIO interrupt edge trigger registers (IOINTEDG1 and IOINTEDG2) to enable rising or falling edge trigger for the corresponding GPIO pin. Each of these registers control 16 of the 32 GPIOs. IOINTEDG1 is used for GPIO[15:0] with bit 0 corresponding to GPIO 0 through bit 15 corresponding to GPIO15. IOINTEDG2 is used for GPIO[31:16] with bit 0 corresponding to GPIO 16 through bit 15 corresponding to GPIO 31.

Figure 5-7. IOINTEDG1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTEDG[15:0]															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 5-8. IOINTEDG1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	INTEDG[15:0]	R/W	0h	Edge select for GPIO 15 to 0. 0x0 = Rising edge triggered 0x1 = Falling edge triggered

5.3.8 IOINTEDG2 Register (offset = 1C0Dh) [reset = 0h]

IOINTEDG2 is shown in Figure 5-8 and described in Table 5-9.

The device has two registers for configuring interrupts to trigger on the rising or falling edge of the input signal to the GPIO pins if they are configured as inputs with Interrupt enabled for the chosen GPIO. Use the GPIO interrupt edge trigger registers (IOINTEDG1 and IOINTEDG2) to enable rising or falling edge trigger for the corresponding GPIO pin. Each of these registers control 16 of the 32 GPIOs. IOINTEDG1 is used for GPIO[15:0] with bit 0 corresponding to GPIO 0 through bit 15 corresponding to GPIO15. IOINTEDG2 is used for GPIO[31:16] with bit 0 corresponding to GPIO 16 through bit 15 corresponding to GPIO 31.

Figure 5-8. IOINTEDG2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTEDG[31:16]															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 5-9. IOINTEDG2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	INTEDG[31:16]	R/W	0h	Edge select for GPIO 31 to 16. 0x0 = Rising edge triggered 0x1 = Falling edge triggered

5.3.9 IOINTEN1 Register (offset = 1C0Eh) [reset = 0h]

IOINTEN1 is shown in Figure 5-9 and described in Table 5-10.

The device has two registers for enabling Interrupts on the GPIO pins if they are configured as Inputs. Use the GPIO interrupt enable registers (IOINTEN1 and IOINTEN2) to enable the interrupt of the corresponding GPIO pin. Each of these registers control 16 of the 32 GPIOs. IOINTEN1 is used for GPIO[15:0] with bit 0 corresponding to GPIO 0 through bit 15 corresponding to GPIO15. IOINTEN2 is used for GPIO[31:16] with bit 0 corresponding to GPIO 16 through bit 15 corresponding to GPIO 31.

Figure 5-9. IOINTEN1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTEN[15:0]															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 5-10. IOINTEN1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	INTEN[15:0]	R/W	0h	Interrupt enable for GPIO 15 to 0. 0x0 = Interrupt disabled 0x1 = Interrupt enabled

5.3.10 IOINTEN2 Register (offset = 1C0Fh) [reset = 0h]

IOINTEN2 is shown in Figure 5-10 and described in Table 5-11.

The device has two registers for enabling Interrupts on the GPIO pins if they are configured as Inputs. Use the GPIO interrupt enable registers (IOINTEN1 and IOINTEN2) to enable the interrupt of the corresponding GPIO pin. Each of these registers control 16 of the 32 GPIOs. IOINTEN1 is used for GPIO[15:0] with bit 0 corresponding to GPIO 0 through bit 15 corresponding to GPIO15. IOINTEN2 is used for GPIO[31:16] with bit 0 corresponding to GPIO 16 through bit 15 corresponding to GPIO 31.

Figure 5-10. IOINTEN2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTEN[31:16]															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 5-11. IOINTEN2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	INTEN[31:16]	R/W	0h	Interrupt enable for GPIO 31 to 16. 0x0 = Interrupt disabled 0x1 = Interrupt enabled

5.3.11 IOINTFLG1 Register (offset = 1C10h) [reset = 0h]

IOINTFLG1 is shown in Figure 5-11 and described in Table 5-12.

The device has two registers that latch an interrupt that occurred with the corresponding GPIO pin. Use the GPIO interrupt flag registers (IOINTFLG1 and IOINTFLG2) to determine which GPIO pin triggered the interrupt. Also, these registers are used to clear the interrupt sent to the CPU. Each of these registers control 16 of the 32 GPIOs. IOINTFLG1 is used for GPIO[15:0] with bit 0 corresponding to GPIO 0 through bit 15 corresponding to GPIO15. IOINTFLG2 is used for GPIO[31:16] with bit 0 corresponding to GPIO 16 through bit 15 corresponding to GPIO 31.

Figure 5-11. IOINTFLG1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTFLG[15:0]															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 5-12. IOINTFLG1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	INTFLG[15:0]	R/W	0h	This register latches the corresponding I/O pin(s) that triggered an interrupt. The flag is cleared by setting the corresponding bit. The interrut signal to the CPU will be kept low until all flag bits are cleared. 0x0 = No Interrupt occurred 0x1 = Interrupt occurred (if 1 is written then the flag is cleared)

5.3.12 IOINTFLG2 Register (offset = 1C11h) [reset = 0h]

IOINTFLG2 is shown in Figure 5-12 and described in Table 5-13.

The device has two registers that latch an interrupt that occurred with the corresponding GPIO pin. Use the GPIO interrupt flag registers (IOINTFLG1 and IOINTFLG2) to determine which GPIO pin triggered the interrupt. Also, these registers are used to clear the interrupt sent to the CPU. Each of these registers control 16 of the 32 GPIOs. IOINTFLG1 is used for GPIO[15:0] with bit 0 corresponding to GPIO 0 through bit 15 corresponding to GPIO15. IOINTFLG2 is used for GPIO[31:16] with bit 0 corresponding to GPIO 16 through bit 15 corresponding to GPIO 31.

Figure 5-12. IOINTFLG2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTFLG[31:16]															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 5-13. IOINTFLG2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	INTFLG[31:16]	R/W	0h	This register latches the corresponding I/O pin(s) that triggered an interrupt. The flag is cleared by setting the corresponding bit. The interrut signal to the CPU will be kept low until all flag bits are cleared. 0x0 = No Interrupt occurred 0x1 = Interrupt occurred (if 1 is written then the flag is cleared)

Inter-Integrated Circuit (I2C) Peripheral

Topic **Page**

6.1 Introduction

Described in the following sections is the operation of the inter-integrated circuit (I2C) peripheral. The scope of this document assumes that you are familiar with the Philips Semiconductors Inter-IC bus (I2C-bus) specification version 2.1.

6.1.1 Purpose of the Peripheral

The I2C peripheral provides an interface between the DSP and other devices that are compliant with the I2C-bus specification and connected by way of an I2C-bus. External components that are attached to this two-wire serial bus can transmit and receive data that is up to eight bits wide both to and from the DSP through the I2C peripheral.

6.1.2 Features

The I2C peripheral has the following features:

- Compliance with the Philips Semiconductors I2C-bus specification (version 2.1):
  - Support for byte format transfer.
  - 7-bit and 10-bit addressing modes.
  - General call.
  - START byte mode.
  - Support for multiple master-transmitters and slave-receivers mode.
  - Support for multiple slave-transmitters and master-receivers mode.
  - Combined master transmit/receive and receive/transmit mode.
  - I2C data transfer rate of from 10 kbps up to 400 kbps (Philips I2C rate).
- 2 to 8 bit format transfer.
- Free data format mode.
- One read DMA event and one write DMA event that the DMA can use.
- Seven interrupts that the CPU can use.
- Peripheral enable/disable capability.

6.1.2.1 Features Not Supported

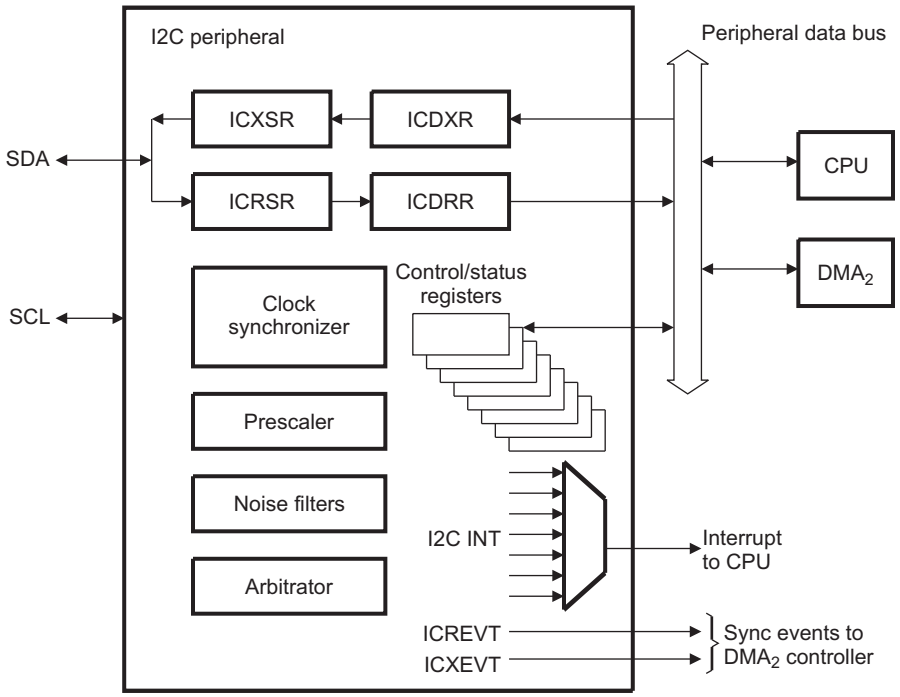
The I2C peripheral does not support the following features:

- High-speed mode.
- CBUS-compatibility mode.
- The combined format in 10-bit addressing mode (the I2C sends the slave address the second byte every time it sends the slave address the first byte).

6.1.3 Functional Block Diagram

A block diagram of the I2C peripheral is shown in Figure 6-1. Refer to Section 6.2 for detailed information about the architecture of the I2C peripheral.

Figure 6-1. I2C Peripheral Block Diagram



A Out of the four DMA controllers included in the DSP, only DMA controller 2 (DMA2) can access the I2C peripheral registers and use its DMA events.

6.1.4 Industry Standard(s) Compliance Statement

The I2C peripheral is compliant with the Philips Semiconductors Inter-IC bus (I2C-bus) specification version 2.1.

6.2 Peripheral Architecture

The I2C peripheral consists of the following primary blocks:

- A serial interface: one data pin (SDA) and one clock pin (SCL).
- Data registers to temporarily hold receive data and transmit data traveling between the SDA pin and the CPU or the DMA2 controller.
- Control and status registers.
- A peripheral data bus interface to enable the CPU and the DMA2 controller to access the I2C peripheral registers.
- A clock synchronizer to synchronize the I2C input clock (from the processor clock generator) and the clock on the SCL pin, and to synchronize data transfers with masters of different clock speeds.
- A prescaler to divide down the input clock that is driven to the I2C peripheral.
- A noise filter on each of the two pins, SDA and SCL.
- An arbitrator to handle arbitration between the I2C peripheral (when it is a master) and another master.
- Interrupt generation logic, so that an interrupt can be sent to the CPU.
- DMA event generation logic, so that activity in the DMA2 controller can be synchronized to data reception and data transmission in the I2C peripheral.

Figure 6-1 shows the four registers used for transmission and reception. The CPU or the DMA controller writes data for transmission to ICDXR and reads received data from ICDRR. When the I2C peripheral is configured as a transmitter, data written to ICDXR is copied to ICXSR and shifted out on the SDA pin one bit at a time. When the I2C peripheral is configured as a receiver, received data is shifted into ICRSR and then copied to ICDRR.

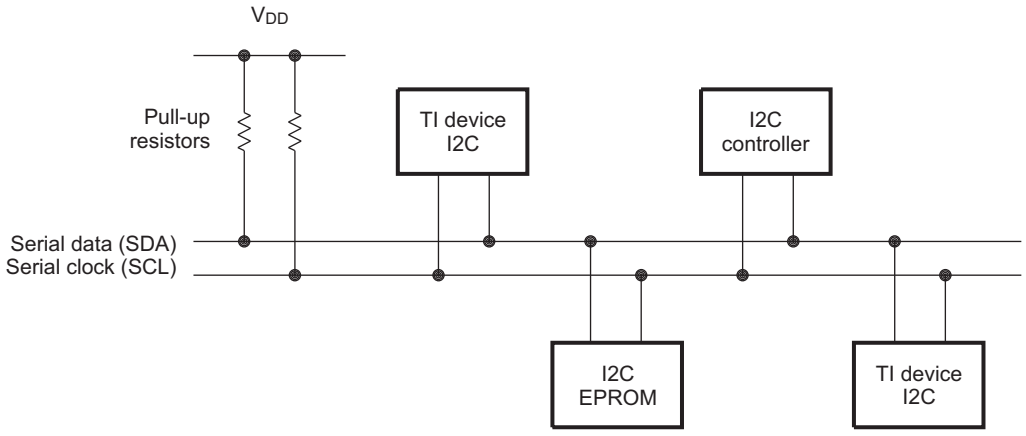
6.2.1 Bus Structure

Figure 6-1 shows how the I2C peripheral is connected to the I2C bus. The I2C bus is a multi-master bus that supports a multi-master mode. This allows more than one device capable of controlling the bus that is connected to it. A unique address recognizes each I2C device. Each I2C device can operate as either transmitter or receiver, depending on the function of the device. Devices that are connected to the I2C bus can be considered a master or slave when performing data transfers, in addition to being a transmitter or receiver.

NOTE: A master device is the device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. Any device that is addressed by this master is considered a slave during this transfer.

An example of multiple I2C modules that are connected for a two-way transfer from one device to other devices is shown in Figure 6-2.

Figure 6-2. Multiple I2C Modules Connected

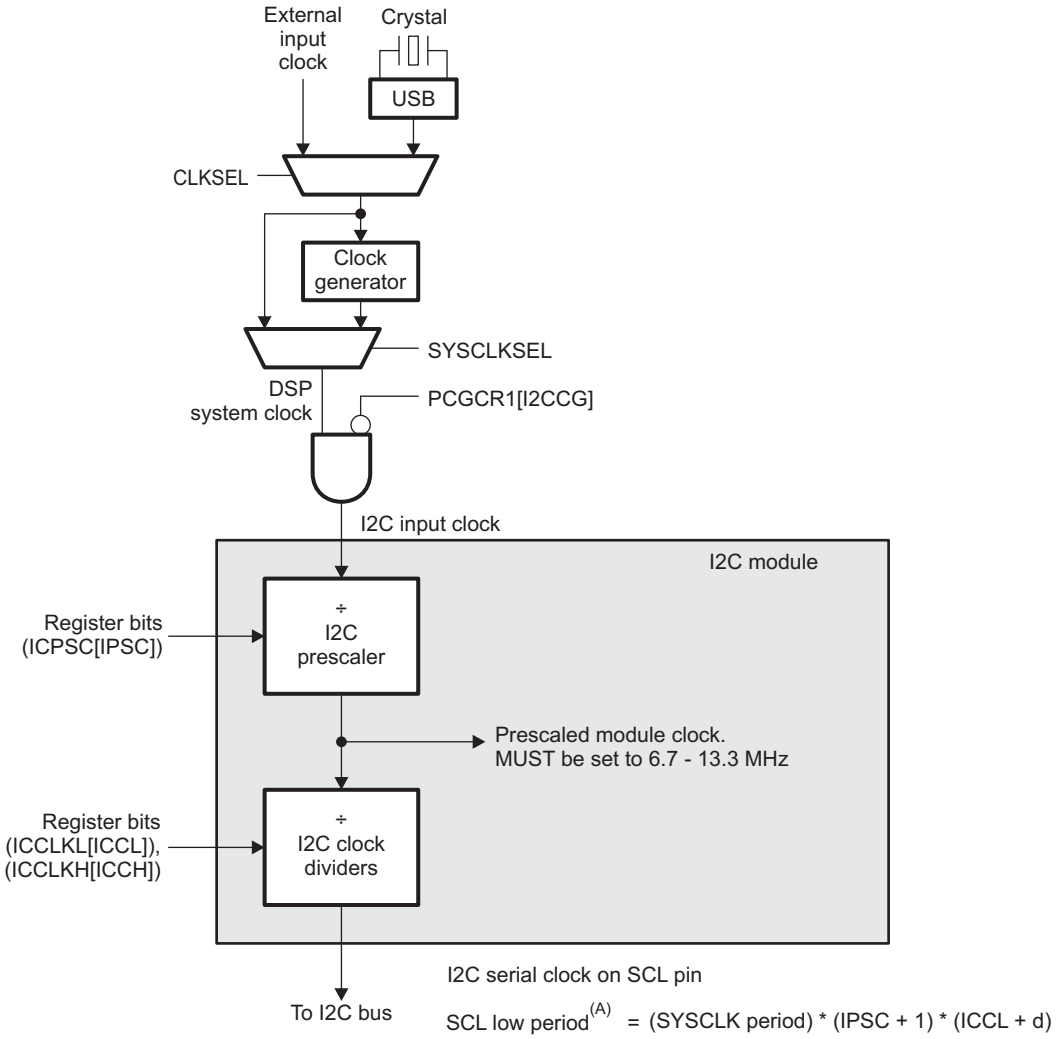


6.2.2 Clock Generation

As shown in Figure 6-3, the clock generator receives either the USB oscillator or a signal from an external clock source and produces the DSP system clock. This clock is used by the DSP CPU and peripherals. A programmable prescaler (IPSC bit in ICPSC) in the I2C module divides down the I2C input clock to produce a prescaled module clock. The prescaled module clock must be operated within the range of 6.7 to 13.3 MHz. The I2C clock dividers divide-down the high (ICCH bit in ICCLKH) and low portions (ICCL bit in ICCLKL) of the prescaled module clock signal to produce the I2C serial clock, which appears on the SCL pin when the I2C module is configured to be a master on the I2C bus.

The DSP includes logic which can be used to gate the clock to its on-chip peripherals. The I2C input clock can be enabled and disabled through the peripheral clock gating configuration register 1 (PCGCR1).

Figure 6-3. Clocking Diagram for the I2C Peripheral



Where d depends on IPSC value in I2CPSC:

IPSC value	d
0	7
1	6
2h-FFh	5

A The period defined does not include rise and fall time and latency of the synchronizer inside the peripheral. The actual transfer rate will be slower than the value calculated from the formula for the SCL period. Due to the nature of SCL synchronization, the SCL period could change if SCL synchronization occurs.

CAUTION

Prescaled Module Clock Frequency Range:

The I2C module must be operated with a prescaled module clock frequency of 6.7 to 13.3 MHz. The I2C prescaler register (ICPSC) must be configured to this frequency range.

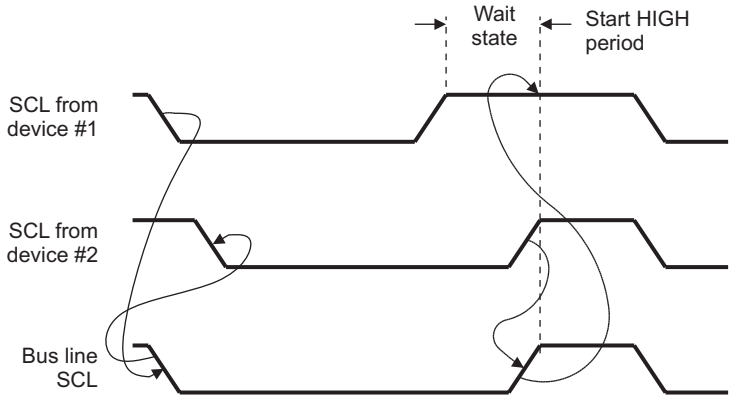
The prescaler (IPSC bit in ICPSC) must only be initialized while the I2C module is in the reset state (IRS = 0 in ICMDR). The prescaled frequency only takes effect when the IRS bit in ICMDR is changed to 1. Changing the IPSC bit in ICPSC while IRS = 1 in ICMDR has no effect. Likewise, you must configure the I2C clock dividers (ICCH bit in ICCLKH and ICCL bit in ICCLKL) while the I2C module is still in reset (IRS = 0 in ICMDR).

6.2.3 Clock Synchronization

Under normal conditions a master device generates its own clock signal on the SCL pin. However, when there are two or more masters connected to the I2C bus the clock must be synchronized such that bit-by-bit arbitration (described in Section 6.2.10) can take place. Figure 6-4 illustrates the clock synchronization. The wired-AND property of SCL means that a device that first generates a low period on SCL (device #1) overrules the other devices. At this high-to-low transition, the clock generators of the other devices are forced to start their own low period. The SCL is held low by the device with the longest low period. The other devices that finish their low periods must wait for SCL to be released before starting their high periods. Using this approach, a synchronized signal on SCL is obtained where the slowest device determines the length of the low period and the fastest device determines the length of the high period.

If a device pulls down the clock line for a longer time, the result is that all clock generators must enter the wait state. This way, a slave slows down a fast master and the slow device creates enough time to store a received data word or to prepare a data word that you are going to transmit.

Figure 6-4. Synchronization of Two I2C Clock Generators



6.2.4 Signal Descriptions

The I2C peripheral has a serial data pin (SDA) and a serial clock pin (SCL) for data communication, as shown in Figure 6-1. These two pins carry information between the device and other devices that are connected to the I2C-bus. The SDA and SCL pins both are bi-directional. They each must be connected to a positive supply voltage using a pull-up resistor. When the bus is free, both pins are high. The driver of these two pins has an open-drain configuration to perform the required wired-AND function.

See the device-specific data manual for additional timing and electrical specifications for these pins.

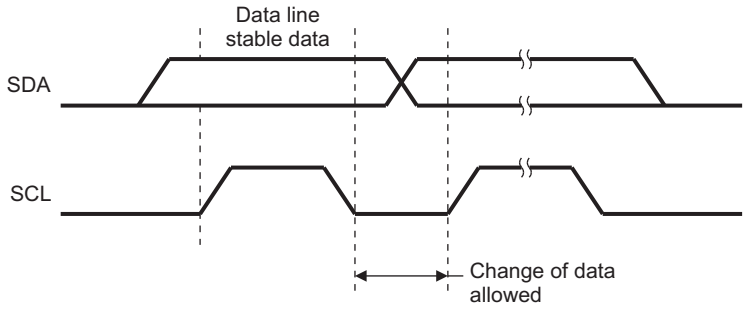
6.2.4.1 Input and Output Voltage Levels

The master device generates one clock pulse for each data bit that is transferred. Due to a variety of different technology devices that can be connected to the I2C-bus, the levels of logic 0 (low) and logic 1 (high) are not fixed and depend on the associated power supply level. See the device-specific data manual for more information.

6.2.4.2 Data Validity

The data on SDA must be stable during the high period of the clock (see Figure 6-5). The high or low state of the data line, SDA, can change only when the clock signal on SCL is low.

Figure 6-5. Bit Transfer on the I2C-Bus



6.2.5 START and STOP Conditions

The I2C peripheral can generate START and STOP conditions when the peripheral is configured to be a master on the I2C-bus, as shown in Figure 6-6:

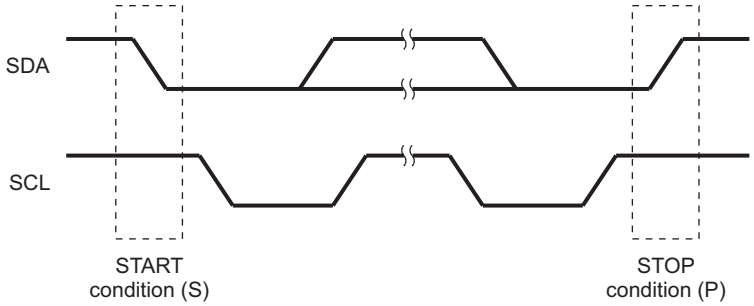
- The START condition is defined as a high-to-low transition on the SDA line while SCL is high. A master drives this condition to indicate the start of a data transfer.
- The STOP condition is defined as a low-to-high transition on the SDA line while SCL is high. A master drives this condition to indicate the end of a data transfer.

The I2C bus is considered busy after a START condition and before a subsequent STOP condition. The bus busy (BB) bit of ICSTR is 1 during the busy state. The bus is considered free between a STOP condition and the next START condition, but the current master should wait for additional tBUF time before attempting to start a new transfer. The BB bit of ICSTR is 0 during the free state.

The user must set to 1 the master mode (MST) and START condition (STT) bits in the ICMDR register for the I2C peripheral to generate a data transfer with a START condition. The STOP condition (STP) bit must be 1 for the I2C peripheral to end a data transfer with a STOP condition. A repeated START condition occurs when the BB and STT bits are set to 1.

Note: BB goes low before MST goes low. The program must wait for MST to go low before starting a new condition.

Figure 6-6. I2C Peripheral START and STOP Conditions



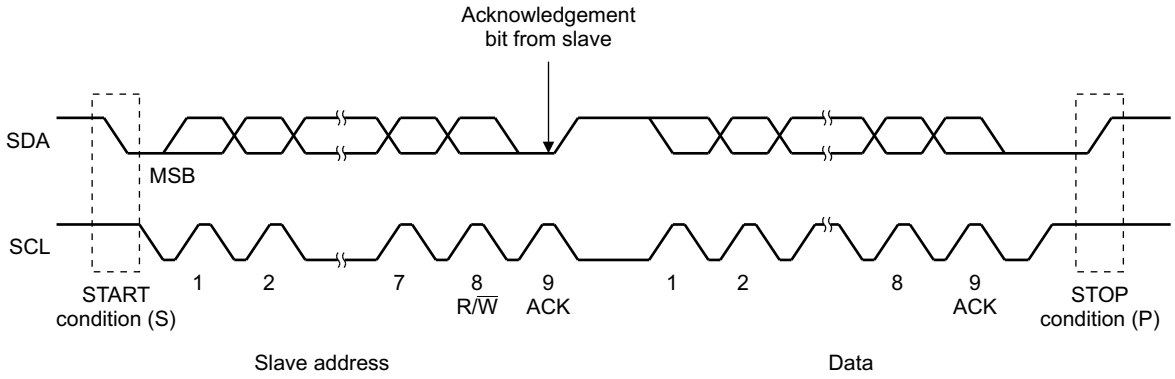
6.2.6 Serial Data Formats

Figure 6-7 shows an example of a data transfer on the I2C-bus. The I2C peripheral supports 2-bit to 8-bit data values. Figure 6-7 shows a typical I2C data transfer using (BC = 000 in ICMDR). Each bit put on the SDA line is equivalent to one pulse on the SCL line. The data is always transferred with the most-significant bit (MSB) first. The number of data values that can be transmitted or received is unrestricted; however, in most systems, the transmitter and receiver have agreed upon the number of data values to transfer before transfer begins.

The I2C peripheral supports the following data formats:

- 7-bit addressing mode.
- 10-bit addressing mode.
- Free data format mode.

Figure 6-7. I2C Peripheral Data Transfer



6.2.6.1 7-Bit Addressing Format

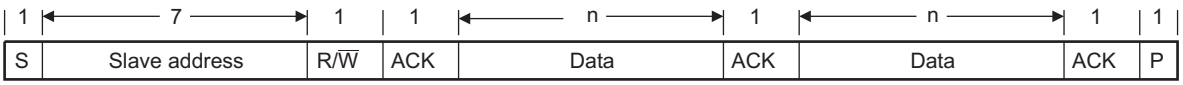
In the 7-bit addressing format (Figure 6-8), the first byte after a START condition (S) consists of a 7-bit slave address followed by a R/$\overline{W}$ bit. The R/$\overline{W}$ bit determines the direction of the data.

- R/$\overline{W}$ = 0: The master writes (transmits) data to the addressed slave.
- R/$\overline{W}$ = 1: The master reads (receives) data from the slave.

An extra clock cycle dedicated for acknowledgment (ACK) is inserted after the R/$\overline{W}$ bit. If the slave inserts the ACK bit, *n* bits of data from the transmitter (master or slave, depending on the R/$\overline{W}$ bit) follow it. *n* is a number from 2 to 8 that the bit count (BC) bits of ICMDR determine. The receiver inserts an ACK bit after the data bits have been transferred.

Write a 0 to the expanded address enable (XA) bit of ICMDR to select the 7-bit addressing format.

Figure 6-8. I2C Peripheral 7-Bit Addressing Format (FDF = 0, XA = 0 in ICMDR)



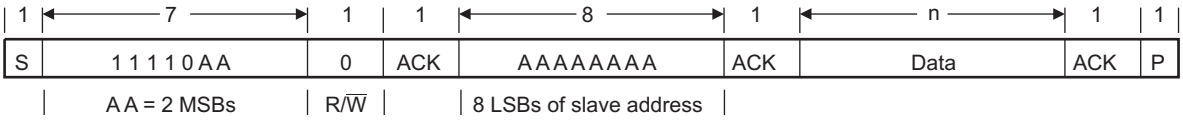
n = The number of data bits (from 2 to 8) specified by the bit count (BC) field of ICM DR.

6.2.6.2 10-Bit Addressing Format

The 10-bit addressing format (Figure 6-9) is like the 7-bit addressing format, but the master sends the slave address in two separate byte transfers. The first byte consists of 11110b, the two MSBs of the 10-bit slave address, and R/$\overline{W}$ = 0 (write). The second byte is the remaining 8 bits of the 10-bit slave address. The slave must send acknowledgment (ACK) after each of the two byte transfers. Once the master has written the second byte to the slave, the master can either write data or use a repeated START condition to change the data direction. (For more information about using 10-bit addressing, see the Philips Semiconductors I2C-bus specification.)

Write 1 to the XA bit of ICMDR to select the 10-bit addressing format.

Figure 6-9. I2C Peripheral 10-Bit Addressing Format With Master-Transmitter Writing to Slave-Receiver (FDF = 0, XA = 1 in ICMDR)



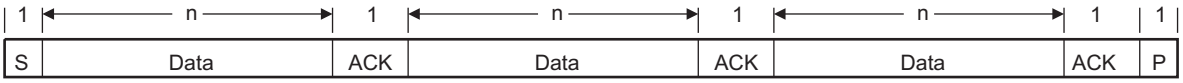
n = The number of data bits (from 2 to 8) specified by the bit count (BC) field of ICMDR.

6.2.6.3 Free Data Format

In the free data format (Figure 6-10), the first bits after a START condition (S) are a data word. An ACK bit is inserted after each data word, which can be from 2 to 8 bits, depending on the bit count (BC) bits of ICMDR. No address or data-direction bit is sent. Therefore, the transmitter and the receiver must both support the free data format, and the direction of the data must be constant throughout the transfer.

To select the free data format, write 1 to the free data format (FDF) bit of ICMDR.

Figure 6-10. I2C Peripheral Free Data Format (FDF = 1 in ICMDR)

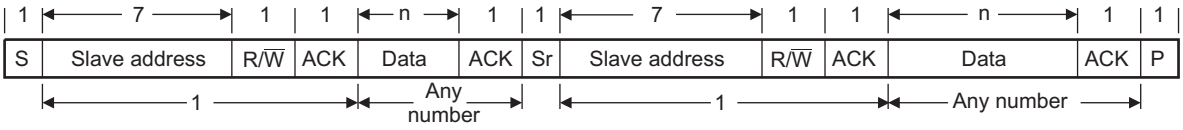


n = The number of data bits (from 2 to 8) specified by the bit count (BC) field of ICMDR.

6.2.6.4 Using a Repeated START Condition

The repeated START condition can be used with the 7-bit addressing, 10-bit addressing, and free data formats. An example of using the repeated START condition (Sr) in the 7-bit addressing format is shown in Figure 6-11. At the end of each data word, the master can drive another START condition. Using this capability, a master can transmit/receive any number of data words before driving a STOP condition. The length of a data word can be from 2 to 8 bits and is selected with the bit count (BC) bits of ICMDR.

Figure 6-11. I2C Peripheral 7-Bit Addressing Format With Repeated START Condition (FDF = 0, XA = 0 in ICMDR)



n = The number of data bits (from 2 to 8) specified by the bit count (BC) field of ICMDR.

6.2.7 Operating Modes

The I2C peripheral has four basic operating modes to support data transfers as a master and as a slave. See Table 6-1 for the names and descriptions of the modes.

If the I2C peripheral is a master, it begins as a master-transmitter and, typically, transmits an address for a particular slave. When giving data to the slave, the I2C peripheral must remain a master-transmitter. In order to receive data from a slave, the I2C peripheral must be changed to the master-receiver mode.

If the I2C peripheral is a slave, it begins as a slave-receiver and, typically, sends acknowledgment when it recognizes its slave address from a master. If the master will be sending data to the I2C peripheral, the peripheral must remain a slave-receiver. If the master has requested data from the I2C peripheral, the peripheral must be changed to the slave-transmitter mode.

Table 6-1. Operating Modes of the I2C Peripheral

Operating Mode	Description
Slave-receiver mode	The I2C peripheral is a slave and receives data from a master. All slave modules begin in this mode. In this mode, serial data bits received on SDA are shifted in with the clock pulses that are generated by the master. As a slave, the I2C peripheral does not generate the clock signal, but it can hold SCL low while the intervention of the processor is required (RSFULL = 1 in ICSTR) after data has been received.
Slave-transmitter mode	The I2C peripheral is a slave and transmits data to a master. This mode can only be entered from the slave-receiver mode; the I2C peripheral must first receive a command from the master. When you are using any of the 7-bit/10-bit addressing formats, the I2C peripheral enters its slave-transmitter mode if the slave address is the same as its own address (in ICOAR) and the master has transmitted R/$\overline{W}$ = 1. As a slave-transmitter, the I2C peripheral then shifts the serial data out on SDA with the clock pulses that are generated by the master. While a slave, the I2C peripheral does not generate the clock signal, but it can hold SCL low while the intervention of the processor is required (XSMT = 0 in ICSTR) after data has been transmitted.
Master-receiver mode	The I2C peripheral is a master and receives data from a slave. This mode can only be entered from the master-transmitter mode; the I2C peripheral must first transmit a command to the slave. When you are using any of the 7-bit/10-bit addressing formats, the I2C peripheral enters its master-receiver mode after transmitting the slave address and R/$\overline{W}$ = 1. Serial data bits on SDA are shifted into the I2C peripheral with the clock pulses generated by the I2C peripheral on SCL. The clock pulses are inhibited and SCL is held low when the intervention of the processor is required (RSFULL = 1 in ICSTR) after data has been received.
Master-transmitter mode	The I2C peripheral is a master and transmits control information and data to a slave. All master modules begin in this mode. In this mode, data assembled in any of the 7-bit/10-bit addressing formats is shifted out on SDA. The bit shifting is synchronized with the clock pulses generated by the I2C peripheral on SCL. The clock pulses are inhibited and SCL is held low when the intervention of the processor is required (XSMT = 0 in ICSTR) after data has been transmitted.

6.2.8 NACK Bit Generation

When the I2C peripheral is a receiver (master or slave), it can acknowledge or ignore bits sent by the transmitter. To ignore any new bits, the I2C peripheral must send a no-acknowledge (NACK) bit during the acknowledge cycle on the bus. Table 6-2 summarizes the various ways the I2C peripheral sends a NACK bit.

Table 6-2. Ways to Generate a NACK Bit

I2C Peripheral Condition	NACK Bit Generation	
	Basic	Optional
Slave-receiver mode	• Disable data transfers (STT = 0 in ICSTR). • Allow an overrun condition (RSFULL = 1 in ICSTR). • Reset the peripheral (IRS = 0 in ICMDR).	Set the NACKMOD bit of ICMDR before the rising edge of the last data bit you intend to receive.
Master-receiver mode AND Repeat mode (RM = 1 in ICMDR)	• Generate a STOP condition (STOP = 1 in ICMDR). • Reset the peripheral (IRS = 0 in ICMDR).	Set the NACKMOD bit of ICMDR before the rising edge of the last data bit you intend to receive.
Master-receiver mode AND Nonrepeat mode (RM = 0 in ICMDR)	• If STP = 1 in ICMDR, allow the internal data counter to count down to 0 and force a STOP condition. • If STP = 0, make STP = 1 to generate a STOP condition. • Reset the peripheral (IRS = 0 in ICMDR).	Set the NACKMOD bit of ICMDR before the rising edge of the last data bit you intend to receive.

6.2.9 NACK Response

6.2.9.1 Hardware Response to a NACK

The following steps detail the hardware response to a NACK.

1. Clear ICMDR.STP.
2. Hold SCL low.
3. Set ICSTR.NACK.

6.2.9.2 User Response to a NACK

The user must perform the following steps when responding to a NACK.

1. Set ICMDR.STP, which sends a STOP bit and releases SCL.
2. Set ICSTR.NACK=1 to clear the flag.
3. Wait for ICMDR.MST to self-clear before initiating further I2C transactions.

   When the MST bit clears, the controller has finished sending the STOP bit. Verify the MST bit clears by checking if ICMDR.MST is equal to 0 at the start of the function. This check allows the processor time to perform other tasks but not start a new transaction until ICMDR.MST has cleared.

6.2.10 Arbitration

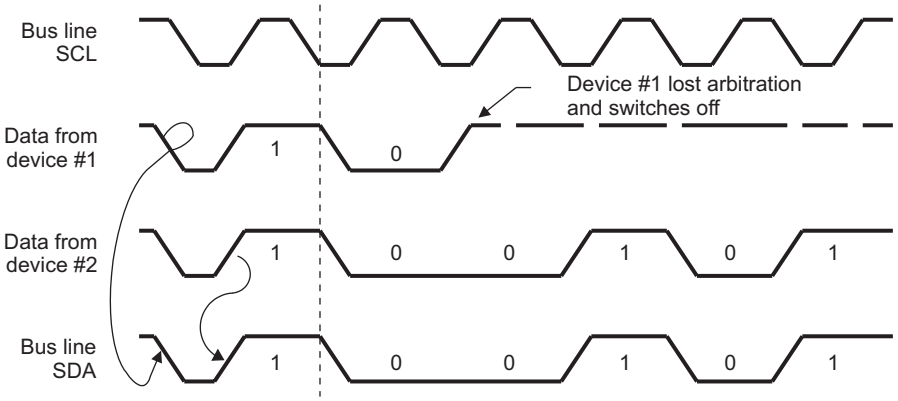
If two or more master-transmitters simultaneously start a transmission on the same bus, an arbitration procedure is invoked. The arbitration procedure uses the data presented on the serial data bus (SDA) by the competing transmitters. Figure 6-12 illustrates the arbitration procedure between two devices. The first master-transmitter, which drives SDA high, is overruled by another master-transmitter that drives SDA low. The arbitration procedure gives priority to the device that transmits the serial data stream with the lowest binary value. Should two or more devices send identical first bytes, arbitration continues on the subsequent bytes.

If the I2C peripheral is the losing master, it switches to the slave-receiver mode, sets the arbitration lost (AL) flag, and generates the arbitration-lost interrupt.

If during a serial transfer the arbitration procedure is still in progress when a repeated START condition or a STOP condition is transmitted to SDA, the master-transmitters involved must send the repeated START condition or the STOP condition at the same position in the format frame. Arbitration is not allowed between:

- A repeated START condition and a data bit.
- A STOP condition and a data bit.
- A repeated START condition and a STOP condition.

Figure 6-12. Arbitration Procedure Between Two Master-Transmitters



6.2.11 Reset Considerations

The I2C peripheral has two reset sources: software reset and hardware reset.

6.2.11.1 Software Reset Considerations

The I2C peripheral can be reset by software through the I2C reset (IRS) bit in the I2C mode register (ICMDR) or through the I2C_RST bit in the peripheral reset control register (PRCR).

When IRS is cleared to 0, all status bits in the I2C interrupt status register (ICSTR) are forced to their default values, and the I2C peripheral remains disabled until IRS is changed to 1. The SDA and SCL pins are in the high-impedance state.

NOTE: If the IRS bit is cleared to 0 during a transfer, this can cause the I2C bus to hang (SDA and SCL are in the high-impedance state).

When I2C_RST is set to 1, a hardware reset is forced on the I2C. The effects of a hardware reset are described in the next section. Please note that the I2C input clock must be enabled when using I2C_RST (see Section 6.2.2).

6.2.11.2 Hardware Reset Considerations

A hardware reset is always initiated during a full chip reset. Alternatively, software can force an I2C hardware reset through the I2C_RST bits of the peripheral reset control register (PRCR). See the device-specific data manual for more details on PRCR.

When a hardware reset occurs, all the registers of the I2C peripheral are set to their default values and the I2C peripheral remains disabled until the I2C reset (IRS) bit in the I2C mode register (ICMDR) is changed to 1.

NOTE: The IRS bit must be cleared to 0 while you configure/reconfigure the I2C peripheral. Forcing IRS to 0 can be used to save power and to clear error conditions.

6.2.12 Initialization

Proper I2C initialization is required prior to starting communication with other I2C device(s). Unless a fully fledged software driver is in place, you need to determine the required I2C configuration needed (for example, Master Receiver) and configure the I2C controller with the desired settings. Enabling the I2C input clock should be the first task. With the I2C controller still in reset, you are now ready to configure the I2C controller. Once configuration is done, you need to enable the I2C controller by releasing the controller from reset. Prior to starting communication, you need to make sure that all status bits are cleared and no pending interrupts exist. Once the bus is determined to be available (the bus is not busy), the I2C is ready to proceed with the desired communication.

6.2.12.1 Configuring the I2C in Master Receiver Mode and Servicing Receive Data via CPU

The following initialization procedure is for the I2C controller configured in Master Receiver mode. The CPU is used to move data from the I2C receive register to internal DSP memory.

1. Ensure the I2C is out of reset by setting I2C-RST=0 in the peripheral reset control register (PRCR). See the device-specific data manual for more information on PRCR.
2. Enable the I2C input clock by setting I2CCG to 1 in the peripheral clock gating configuration register (PCGCR1). See the device-specific data manual for more information on PCGCR1.
3. Place I2C in reset (clear IRS = 0 in ICMDR).
4. Configure ICMDR:
   - Configure I2C as Master (MST = 1).
   - Indicate the I2C configuration to be used; for example, Data Receiver (TRX = 0).
   - Indicate 7-bit addressing is to be used (XA = 0).

- Disable repeat mode (RM = 0).
- Disable loopback mode (DLB = 0).
- Disable free data format (FDF = 0).
- Optional: Disable start byte mode if addressing a fully fledged I2C device (STB = 0).
- Set number of bits to transfer to be 8 bits (BC = 0).
- Optional: Enable the receive interrupt (ICRINT) by setting ICRRDY = 1 in ICIMR.

5. Specify the slave address of the I2C device that will be accessed using ICSAR. For 7-bit addressing mode, only the first seven bits of ICSAR are used.
6. Configure the peripheral clock operation frequency (ICPSC). This value should be selected in such a way that the frequency is between 6.7 and 13.3 MHz.
7. Configure I2C master clock frequency:
   - Configure the low-time divider value (ICCLKL).
   - Configure the high-time divider value (ICCLKH).
8. Make sure the interrupt status register (ICSTR) and interrupt vector register (ICIVR) are cleared:
   - Read ICSTR and write back the same value (writing a 1 to the bits of ICSTR clears them).
   - Read ICIVR until it is zero.
9. Take I2C controller out of reset by setting IRS = 1 in ICMDR.
10. Wait until bus busy bit is cleared (BB = 0 in ICSTR).
11. Start the transfer by setting STT = 1 in ICMDR. The I2C controller will start the transfer using the 7-bit addressing format as described in Section 6.2.6.1.
12. Wait until data is received.
    - If using polling method, wait until ICRRDY = 1 in ICSTR.
    - If using interrupt method, wait until the I2C controller generates an interrupt, read the ICIVR register to determine if a receive interrupt (ICRINT) has been generated, then clear the interrupt flag by writing a 1 to it.
13. Read the received data from ICDRR.
14. Repeat steps 11 and 12 until last data byte has been received.
15. End transfer/release bus when transfer is done by generating a STOP event (set STP = 1 in ICMDR).

6.2.12.2 Configuring the I2C in Slave Receiver and Transmitter Mode

The following initialization procedure is for the I2C controller configured in Slave Receiver and Transmitter mode.

1. Enable I2C clock from PSC Level. Do this so that you will be able to configure the I2C registers.
2. Place I2C in Reset (Clear IRS bit).
   - ICMDR.IRS=0.
3. Assign the Address (7-bit or 10-bit address) to which the I2C Controller will be responding. This is the Address that the Master is going to broadcast when attempting to start communication with this slave device; I2C Controller.
   - If the I2C is able to respond to 7-bit Addressing: Configure ICMDR.XA=0.
   - If the I2C is able to respond to 10-bit Addressing: Configure ICMDR.XA=1.
   - Program ICOAR=Assigned Address (7-bit or 10-bit address).
4. Enable the desired interrupt you need to receive by setting the desired interrupt bit field within ICIMR to enable the particular Interrupt.
   - ICIMR.AAS=1; Expect an interrupt when Master's Address matches yours (ICOAR programmed value).
   - ICIMR.ICRRDY=1; Expect a receive interrupt when a byte worth data sent from the master is ready to be read.
   - ICIMR.ICXRDY=1; Expect to receive interrupt when the transmit register is ready to be written with

a new data that is to be sent to the master.

- ICIMR.SCD=1; Expect to receive interrupt when Stop Condition is detected.

5. Configure the I2C Controller Operating frequency; this is not the serial clock frequency. This should be between 6.7 and 13.3 MHz. Program IPSC to generate a 6.7 to 13.3 MHz operating frequency.
   - Prescaled Module Clock Frequency = PLL1 Output Frequency / (IPSC + 1).
6. Configure the I2C Serial Clock Frequency. It is advised to configure this frequency to operate at 400 kHz. This will allow the slave device to be able to attend to all Master speeds. Program ICCH and ICCL.
   - 400 kHz = I2C Operating Frequency (6.7-13.3 MHz from Step 5) / [(ICCH+5) + (ICCL+5)].
   - If ICCL==ICCH ≥ 400 kHz = Prescaled Module Clock Frequency / [2×ICCH+10].
7. Configure the Mode Register.
   - ICMDR.MST=0; Configure the I2C Controller to operate as SLAVE.
   - ICMDR.FDF=0; Free Data Format is disabled.
   - ICMDR.BC=0; Set data width to 8 bytes.
   - ICMDR.DLB=0; Disable Loopback Mode.
   - ICMDR.STB=0; I2C Controller can detect Start condition via H/W.
   - ICMDR.RM=1, STP=0, STT=1. See Table 16. (No Activity case).
   - Configure remaining bits other than ICMDR.IRS to 0.
8. Release I2C from Reset.
   - ICMDR.IRS=1; Make sure you do not over write your previous configurations.
9. Make sure Interrupt Status Register is cleared.
   - ICSTR=ICSTR; Clear Interrupt fields that require writing '1' requirements.
   - While (ICIVR != 0) Read ICIVR; Read until it is cleared to Zero.
10. Instruct I2C Controller to detect START Condition and Its Own Address.
    - ICMDR.STT=1; Make sure you do not over write your previous configurations.

    MASTER desires to perform a write transfer.
11. If Master requests a Write, i.e, I2C needs to receive data, perform the following:
    - Wait for Receive Interrupt to be received, i.e, ICSTR.ICRRDY=1.
    - Read Data.
12. Perform Step 11 until one of the two happens:
    - Master generates a STOP Condition (ICSTR.STP=1) or
    - I2C Slave desires to end receive transfer.

      If the latter, then the I2C needs to Not Acknowledge the last byte to be received from the Master. After reading the byte prior from the last byte, set NACKMOD bit so that the I2C automatically NACKs the following received data byte, which is the last data byte.
    - ICMDR.NACKMOD=1; set this field on the 2nd data prior from the last.

    Master desires to perform a read transfer.
13. If Master requests a Read, i.e, I2C needs to transmit data, perform the following.
    - Write Data.
    - Wait for Transmit Interrupt to be received, i.e, ICSTR.ICXRDY=1.
14. Perform step 13 until a STOP condition is detected, i.e. (ICSTR.STP=1).

6.2.13 Interrupt Support

The I2C is capable of interrupting the CPU. The CPU can determine which I2C events caused the interrupt by reading the I2C interrupt vector register (ICIVR). ICIVR contains a binary-coded interrupt vector type to indicate which interrupt has occurred. Reading ICIVR clears the interrupt flag; if other interrupts are pending, a new interrupt is generated. If there is more than one pending interrupt flag, reading ICIVR clears the highest-priority interrupt flag.

6.2.13.1 Interrupt Events and Requests

The I2C peripheral can generate the interrupts described in Table 6-3. Each interrupt has a flag bit in the I2C interrupt status register (ICSTR) and a mask bit in the interrupt mask register (ICIMR). When one of the specified events occurs, its flag bit is set. If the corresponding mask bit is 0, the interrupt request is blocked; if the mask bit is 1, the request is forwarded to the CPU as an I2C interrupt.

Table 6-3. Descriptions of the I2C Interrupt Events

I2C Interrupt	Initiating Event
Arbitration-lost interrupt (AL)	Generated when the I2C arbitration procedure is lost or illegal START/STOP conditions occur.
No-acknowledge interrupt (NACK)	Generated when the master I2C does not receive any acknowledge from the receiver.
Registers-ready-for-access interrupt (ARDY)	Generated by the I2C when the previously programmed address, data and command have been performed and the status bits have been updated. This interrupt is used to let the controlling processor know that the I2C registers are ready to be accessed.
Receive interrupt/status (ICRINT and ICRRDY)	Generated when the received data in the receive-shift register (ICRSR) has been copied into the ICDRR. The ICRRDY bit can also be polled by the CPU to read the received data in the ICDRR.
Transmit interrupt/status (ICXINT and ICXRDY)	Generated when the transmitted data has been copied from ICDXR to the transmit-shift register (ICXSR) and shifted out on the SDA pin. This bit can also polled by the CPU to write the next transmitted data into the ICDXR. Note that since ICXINT is generated during the copy of ICDXR to ICXSR, ICXINT will be generated even if no slave acknowledges the value being shifted out of ICXSR.
Stop-Condition-Detection interrupt (SCD)	Generated when a STOP condition has been detected.
Address-as-Slave interrupt (AAS)	Generated when the I2C has recognized its own slave address or an address of all (8) zeros.

6.2.13.2 Interrupt Multiplexing

The I2C interrupt to the CPU is not multiplexed with any other interrupt source.

6.2.14 DMA Events Generated by the I2C Peripheral

The I2C peripheral generates two DMA events. Activity in the DMA controller can be synchronized to these events. Note that out of the four DMA controllers included in the device, only DMA controller 2 (DMA2) can synchronize its activity using the I2C events.

- Receive event (ICREVT): When receive data has been copied from the receive shift register (ICRSR) to the data receive register (ICDRR), the I2C peripheral sends an REVT signal to the DMA controller. In response, the DMA controller can read the data from ICDRR.
- Transmit event (ICXEVT): When transmit data has been copied from the data transmit register (ICDXR) to the transmit shift register (ICXSR), the I2C peripheral sends an XEVT signal to the DMA controller. In response, the DMA controller can write the next transmit data value to ICDXR. Note that since ICXEVT is generated during the copy of ICDXR to ICXSR, ICXEVT will be generated even if no slave acknowledges the value being shifted out of ICXSR.

6.2.15 Power Management

There are several ways to reduce the power consumption of the I2C peripheral. First, the I2C peripheral can be clock-gated to conserve power during periods of no activity. The I2C peripheral clock can be turned off by using the peripheral clock gating configuration register (PCGCR). Second, the I2C peripheral clock and output clock (SCL) can be reduced to the minimal possible value allowed by the system. As described in Section 6.2.2, Clock Generation, these two clocks are controlled through the ICPSC, ICCLKH, ICCLKL registers. Note that the I2C peripheral clock must be maintained between 7 and 12 MHz. For detailed information on PCGCR, see the device-specific data manual.

6.2.16 Emulation Considerations

The response of the I2C events to emulation suspend events (such as halts and breakpoints) is controlled by the FREE bit in the I2C mode register (ICMDR). The I2C peripheral either stops exchanging data (FREE = 0) or continues to run (FREE = 1) when an emulation suspend event occurs. How the I2C peripheral terminates data transactions is affected by whether the I2C peripheral is acting as a master or a slave. For more information, see the description of the FREE bit in ICMDR (see).

6.3 I2C Registers

The I2C registers can be accessed by the CPU and DMA controller at the absolute 16-bit addresses specified. Note that the CPU accesses all peripheral registers through its I/O space.

Table 6-4 lists the memory-mapped registers for the I2C. All register offset addresses not listed in Table 6-4 should be considered as reserved locations and the register contents should not be modified.

Table 6-4. I2C REGISTERS

CPU Word Address	Acronym	Register Name	Section
1A00h	ICOAR	I2C Own Address Register	Section 6.3.1
1A04h	ICIMR	I2C Interrupt Mask Register	Section 6.3.2
1A08h	ICSTR	I2C Interrupt Status Register	Section 6.3.3
1A0Ch	ICCLKL	I2C Clock Low-Time Divider Register	Section 6.3.4
1A10h	ICCLKH	I2C Clock High-Time Divider Register	Section 6.3.5
1A14h	ICCNT	I2C Data Count Register	Section 6.3.6
1A18h	ICDRR	I2C Data Receive Register	Section 6.3.7
1A1Ch	ICSAR	I2C Slave Address Register	Section 6.3.8
1A20h	ICDXR	I2C Data Transmit Register	Section 6.3.9
1A24h	ICMDR	I2C Mode Register	Section 6.3.10
1A28h	ICIVR	I2C Interrupt Vector Register	Section 6.3.11
1A2Ch	ICEMDR	I2C Extended Mode Register	Section 6.3.12
1A30h	ICPSC	I2C Prescaler Register	Section 6.3.13
1A34h	ICPID1	I2C Peripheral Identification Register 1	Section 6.3.14
1A38h	ICPID2	I2C Peripheral Identification Register 2	Section 6.3.15

6.3.1 ICOAR Register (offset = 1A00h) [reset = 0h]

ICOAR is shown in Figure 6-13 and described in Table 6-5.

The I2C own address register (ICOAR) is used to specify its own slave address, which distinguishes it from other slaves connected to the I2C-bus. If the 7-bit addressing mode is selected (XA = 0 in ICMDR), only bits 6-0 are used; bits 9-7 are ignored.

Figure 6-13. ICOAR Register

15	14	13	12	11	10	9	8
Reserved						OADDR	
R-0h						RW-0h	

7	6	5	4	3	2	1	0
OADDR							
RW-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 6-5. ICOAR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	These reserved bit locations are always read as zeros. A value written to this field has no effect.
9-0	OADDR	RW	0h	Own slave address. Provides the slave address of the I2C. In 7-bit addressing mode (XA = 0 in ICMDR): bits 6-0 provide the 7-bit slave address of the I2C. Bits 9-7 are ignored. In 10-bit addressing mode (XA = 1 in ICMDR): bits 9-0 provide the 10-bit slave address of the I2C.

6.3.2 ICIMR Register (offset = 1A04h) [reset = 0h]

ICIMR is shown in Figure 6-14 and described in Table 6-6.

The I2C interrupt mask register (ICIMR) is used to individually enable or disable I2C interrupt requests.

Figure 6-14. ICIMR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved	AAS	SCD	ICXRDY	ICRRDY	ARDY	NACK	AL
R-0h	RW-0h	RW-0h	RW-0h	RW-0h	RW-0h	RW-0h	RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 6-6. ICIMR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	Reserved	R	0h	Reserved
6	AAS	RW	0h	Address-as-slave interrupt enable bit. 0x0 = Interrupt request is disabled. 0x1 = Interrupt request is enabled.
5	SCD	RW	0h	Stop condition detected interrupt enable bit. 0x0 = Interrupt request is disabled. 0x1 = Interrupt request is enabled.
4	ICXRDY	RW	0h	Transmit-data-ready interrupt enable bit. 0x0 = Interrupt request is disabled. 0x1 = Interrupt request is enabled.
3	ICRRDY	RW	0h	Receive-data-ready interrupt enable bit. 0x0 = Interrupt request is disabled. 0x1 = Interrupt request is enabled.
2	ARDY	RW	0h	Register-access-ready interrupt enable bit. 0x0 = Interrupt request is disabled. 0x1 = Interrupt request is enabled.
1	NACK	RW	0h	No-acknowledgment interrupt enable bit. 0x0 = Interrupt request is disabled. 0x1 = Interrupt request is enabled.
0	AL	RW	0h	Arbitration-lost interrupt enable bit 0x0 = Interrupt request is disabled. 0x1 = Interrupt request is enabled.

6.3.3 ICSTR Register (offset = 1A08h) [reset = 410h]

ICSTR is shown in Figure 6-15 and described in Table 6-7.

The I2C interrupt status register (ICSTR) is used to determine which interrupt has occurred and to read status information.

Figure 6-15. ICSTR Register

15	14	13	12	11	10	9	8
Reserved	SDIR	NACKSNT	BB	RSFULL	XSMT	AAS	AD0
R-0h	RW-0h	RW-0h	RW-0h	R-0h	R-1h	R-0h	R-0h
7	6	5	4	3	2	1	0
Reserved		SCD	ICXRDY	ICRRDY	ARDY	NACK	AL
R-0h		RW-0h	RW-1h	RW-0h	RW-0h	RW-0h	RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 6-7. ICSTR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	Reserved
14	SDIR	RW	0h	Slave direction bit. In digital-loopback mode (DLB), the SDIR bit must be manually cleared by writing 1. SDIR is cleared by one of the following events: - A STOP or a START condition. - SDIR is manually cleared. To clear this bit, write a 1 to it. 0x0 = I2C is acting as a master-transmitter/receiver or a slave-receiver. 0x1 = I2C is acting as a slave-transmitter.
13	NACKSNT	RW	0h	No-acknowledgment sent bit. NACKSNT bit is used when the I2C is in the receiver mode. One instance in which NACKSNT is affected is when the NACK mode is used. NACKSNT is cleared by one of the following events: - It is manually cleared. To clear this bit, write a 1 to it. - The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when 1 is written to the I2C_RST bit of PRCR). 0x0 = NACK is not sent. 0x1 = NACK is sent.
12	BB	RW	0h	Bus busy bit. BB bit indicates whether the I2C-bus is busy or is free for another data transfer. In the master mode, BB is controlled by the software. BB is cleared by one of the following events: - The I2C receives or transmits a STOP bit (bus free). - BB is manually cleared. To clear this bit, write a 1 to it. - The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when 1 is written to the I2C_RST bit of PRCR). BB is set by one of the following events: - The I2C has received or transmitted a START bit on the bus. - SCL is in a low state and the IRS bit in ICMDR is 0. 0x0 = Bus is free. 0x1 = Bus is busy.

Table 6-7. ICSTR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	RSFULL	R	0h	Receive shift register full bit. RSFULL indicates an overrun condition during reception. Overrun occurs when the receive shift register (ICRSR) is full with new data but the previous data has not been read from the data receive register (ICDRR). The new data will not be copied to ICDRR until the previous data is read. As new bits arrive from the SDA pin, they overwrite the bits in ICRSR. RSFULL is cleared by one of the following events: - ICDRR is read. - The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when 1 is written to the I2C_RST bit of PRCR). 0x0 = No overrun is detected. 0x1 = Overrun is detected.
10	XSMT	R	1h	Transmit shift register empty bit. XSMT indicates that the transmitter has experienced underflow. Underflow occurs when the transmit shift register (ICXSR) is empty but the data transmit register (ICDXR) has not been loaded since the last ICDXR-to-ICXSR transfer. The next ICDXR-to-ICXSR transfer will not occur until new data is in ICDXR. If new data is not transferred in time, the previous data may be re-transmitted on the SDA pin. XSMT is set by one of the following events: - Data is written to ICDXR. - The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when 1 is written to the I2C_RST bit of PRCR). 0x0 = Underflow is detected. 0x1 = No underflow is detected.
9	AAS	R	0h	Addressed-as-slave bit. 0x0 = The AAS bit has been cleared by a repeated START condition or by a STOP condition. 0x1 = AAS is set by one of the following events: - I2C has recognized its own slave address or an address of all zeros (general call). - The first data word has been received in the free data format (FDF = 1 in ICMDR).
8	AD0	R	0h	Address 0 bit. 0x0 = AD0 has been cleared by a START or STOP condition. 0x1 = An address of all zeros (general call) is detected.
7-6	Reserved	R	0h	These reserved bit locations are always read as zeros. A value written to this field has no effect.
5	SCD	RW	0h	Stop condition detected bit. SCD indicates when a STOP condition has been detected on the I2C bus. The STOP condition could be generated by the I2C or by another I2C device connected to the bus. SCD is cleared by one of the following events: - By reading the INTCODE bits in ICIVR as 110b. Reading ICIVR also clears the corresponding status bit in ICSTR except ARDY, ICRRDY, ICXRDY, and AAS. - SCD is manually cleared. To clear this bit, write a 1 to it. 0x0 = No STOP condition has been detected. 0x1 = A STOP condition has been detected.

Table 6-7. ICSTR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	ICXRDY	RW	1h	Transmit-data-ready interrupt flag bit. ICXRDY indicates that the data transmit register (ICDXR) is ready to accept new data because the previous data has been copied from ICDXR to the transmit shift register (ICXSR). The CPU can poll ICXRDY or use the XRDY interrupt request. ICXRDY is cleared by one of the following events: - Data is written to ICDXR. - ICXRDY is manually cleared. To clear this bit, write a 1 to it. 0x0 = ICDXR is not ready. 0x1 = ICDXR is ready. Data has been copied from ICDXR to ICXSR. ICXRDY is forced to 1 when the I2C is reset.
3	ICRRDY	RW	0h	Receive-data-ready interrupt flag bit. ICRRDY indicates that the data receive register (ICDRR) is ready to be read because data has been copied from the receive shift register (ICRSR) to ICDRR. The CPU can poll ICRRDY or use the RRDY interrupt request. ICDRR is not ready. ICRRDY is cleared by one of the following events: ICDRR is read. ICRRDY is manually cleared. To clear this bit, write a 1 to it. The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when 1 is written to the I2C_RST bit of PRCR). ICDRR is ready. Data has been copied from ICRSR to ICDRR. 0x0 = ICDRR is not ready. 0x1 = ICDRR is ready. Data has been copied from ICRSR to ICDRR.
2	ARDY	RW	0h	Register-access-ready interrupt flag bit (only applicable when the I2C is in the master mode). ARDY indicates that the I2C registers are ready to be accessed because the previously programmed address, data, and command values have been used. The CPU can poll ARDY or use the ARDY interrupt request. ARDY is cleared by one of the following events: - The I2C starts using the current register contents. - ARDY is manually cleared. To clear this bit, write a 1 to it. - The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when 1 is written to the I2C_RST bit of PRCR). This bit is set after the slave address appears on the I2C bus. - In the non-repeat mode (RM = 0 in ICMDR): If STP = 0 in ICMDR, ARDY is set when the internal data counter counts down to 0. If STP = 1, ARDY is not affected (instead, the I2C generates a STOP condition when the counter reaches 0). - In the repeat mode (RM = 1): ARDY is set at the end of each data word transmitted from ICDXR. 0x0 = The registers are not ready to be accessed. 0x1 = The registers are ready to be accessed.
1	NACK	RW	0h	No-acknowledgment interrupt flag bit. NACK applies when the I2C is a transmitter (master or slave). NACK indicates whether the I2C has detected an acknowledge bit (ACK) or a no-acknowledge bit (NACK) from the receiver. The CPU can poll NACK or use the NACK interrupt request. NACK is cleared by one of the following events: - NACK is manually cleared. To clear this bit, write a 1 to it. - The CPU reads the interrupt vector register (ICIVR) when the register contains the code for a NACK interrupt. - The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when 1 is written to the I2C_RST bit of PRCR). Note: While the I2C performs a general call transfer, NACK is 1, even if one or more slaves send acknowledgment. 0x0 = ACK received/NACK is not received. 0x1 = NACK bit is received.

Table 6-7. ICSTR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	AL	RW	0h	Arbitration-lost interrupt flag bit (only applicable when the I2C is a master-transmitter). AL primarily indicates when the I2C has lost an arbitration contest with another master-transmitter. The CPU can poll AL or use the AL interrupt request. AL is cleared by one of the following events: - AL is manually cleared. To clear this bit, write a 1 to it. - The CPU reads the interrupt vector register (ICIVR) when the register contains the code for an AL interrupt. - The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when 1 is written to the I2C_RST bit of PRCR). AL is set by one of the following events: - The I2C senses that it has lost an arbitration with two or more competing transmitters that started a transmission almost simultaneously. - The I2C attempts to start a transfer while the BB (bus busy) bit is set to 1. When AL is set to 1, the MST and STP bits of ICMDR are cleared, and the I2C becomes a slave-receiver. 0x0 = Arbitration is not lost. 0x1 = Arbitration is lost.

6.3.4 ICCLKL Register (offset = 1A0Ch) [reset = 0h]

ICCLKL is shown in Figure 6-16 and described in Table 6-8.

For each I2C serial clock cycle, ICCL determines the amount of time the signal is low. ICCLKL must be configured while the I2C is still in reset (IRS = 0 in ICMDR).

Figure 6-16. ICCLKL Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 6-8. ICCLKL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	ICCL	RW	0h	Clock low-time divide-down value of 1 to 65536. The period of the module clock is multiplied by (ICCL + d) to produce the low-time duration of the I2C serial on the SCL pin.

6.3.5 ICCLKH Register (offset = 1A10h) [reset = 0h]

ICCLKH is shown in Figure 6-17 and described in Table 6-9.

For each I2C serial clock cycle, ICCH determines the amount of time the signal is high. ICCLKH must be configured while the I2C is still in reset (IRS = 0 in ICMDR).

Figure 6-17. ICCLKH Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 6-9. ICCLKH Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	ICCH	RW	0h	Clock high-time divide-down value of 1 to 65536. The period of the module clock is multiplied by (ICCH + d) to produce the high-time duration of the I2C serial on the SCL pin.

6.3.6 ICCNT Register (offset = 1A14h) [reset = 0h]

ICCNT is shown in Figure 6-18 and described in Table 6-10.

The I2C data count register (ICCNT) is used to indicate how many data words to transfer when the I2C is configured as a master-transmitter (MST = 1 and TRX = 1 in ICMDR) and the repeat mode is off (RM = 0 in ICMDR). In the repeat mode (RM = 1), ICCNT is not used.

Figure 6-18. ICCNT Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 6-10. ICCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	ICDC	RW	0h	Data count value. When RM = 0 in ICMDR, ICDC indicates the number of data words to transfer in the non-repeat mode. When RM = 1 in ICMDR, the value in ICCNT is a don't care. If STP = 1 in ICMDR, a STOP condition is generated when the internal data counter counts down to 0. When ICDC = 0, the start value loaded to the internal data counter is 65536.

6.3.7 ICDRR Register (offset = 1A18h) [reset = 0h]

ICDRR is shown in Figure 6-19 and described in Table 6-11.

The I2C data receive register (ICDRR) is used to read the receive data.

Figure 6-19. ICDRR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							
7	6	5	4	3	2	1	0
D							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 6-11. ICDRR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	These reserved bit locations are always read as zeros. A value written to this field has no effect.
7-0	D	R	0h	Receive data.

6.3.8 ICSAR Register (offset = 1A1Ch) [reset = 3FFh]

ICSAR is shown in Figure 6-20 and described in Table 6-12.

The I2C slave address register (ICSAR) contains a 7-bit or 10-bit slave address.

Figure 6-20. ICSAR Register

15	14	13	12	11	10	9	8
Reserved						SADDR	
R-0h						RW-3FFh	

7	6	5	4	3	2	1	0
SADDR							
RW-3FFh							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 6-12. ICSAR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	These reserved bit locations are always read as zeros. A value written to this field has no effect.
9-0	SADDR	RW	3FFh	Slave address. Provides the slave address that the I2C transmits when it is in master-transmitter mode. In 7-bit addressing mode (XA = 0 in ICMDR): bits 6-0 provide the 7-bit slave address that the I2C transmits when it is in the master-transmitter mode. Bits 9-7 are ignored. In 10-bit addressing mode (XA = 1 in ICMDR): Bits 9-0 provide the 10-bit slave address that the I2C transmits when it is in the master-transmitter mode.

6.3.9 ICDXR Register (offset = 1A20h) [reset = 0h]

ICDXR is shown in Figure 6-21 and described in Table 6-13.

The CPU or DMA writes transmit data to the I2C data transmit register (ICDXR).

Figure 6-21. ICDXR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
D							
RW-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 6-13. ICDXR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	These reserved bit locations are always read as zeros. A value written to this field has no effect.
7-0	D	RW	0h	Transmit data.

6.3.10 ICMDR Register (offset = 1A24h) [reset = 0h]

ICMDR is shown in Figure 6-22 and described in Table 6-14.

The I2C mode register (ICMDR) contains the control bits of the I2C.

Figure 6-22. ICMDR Register

15	14	13	12	11	10	9	8
NACKMOD	FREE	STT	Reserved	STP	MST	TRX	XA
RW-0h	RW-0h	RW-0h	R-0h	RW-0h	RW-0h	RW-0h	RW-0h
7	6	5	4	3	2	1	0
RM	DLB	IRS	STB	FDF	BC		
RW-0h	RW-0h	RW-0h	RW-0h	RW-0h	RW-0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 6-14. ICMDR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	NACKMOD	RW	0h	No-acknowledge (NACK) mode bit (only applicable when the I2C is a receiver). 0x0 = In slave-receiver mode: The I2C sends an acknowledge (ACK) bit to the transmitter during the each acknowledge cycle on the bus. The I2C only sends a no-acknowledge (NACK) bit if you set the NACKMOD bit. In master-receiver mode: The I2C sends an ACK bit during each acknowledge cycle until the internal data counter counts down to 0. When the counter reaches 0, the I2C sends a NACK bit to the transmitter. To have a NACK bit sent earlier, you must set the NACKMOD bit. 0x1 = In either slave-receiver or master-receiver mode: The I2C sends a NACK bit to the transmitter during the next acknowledge cycle on the bus. Once the NACK bit has been sent, NACKMOD is cleared. To send a NACK bit in the next acknowledge cycle, you must set NACKMOD before the rising edge of the last data bit.
14	FREE	RW	0h	This emulation mode bit is used to determine the state of the I2C when a breakpoint is encountered in the high-level language debugger. 0x0(Read) = When I2C is master. If SCL is low when the breakpoint occurs, the I2C stops immediately and keeps driving SCL low, whether the I2C is the transmitter or the receiver. If SCL is high, the I2C waits until SCL becomes low and then stops. When I2C is slave: A breakpoint forces the I2C to stop when the current transmission/reception is complete. 0x1 = The I2C runs free; that is, it continues to operate when a breakpoint occurs.
13	STT	RW	0h	START condition bit (only applicable when the I2C is a master). The RM, STT, and STP bits determine when the I2C starts and stops data transmissions. Note that the STT and STP bits can be used to terminate the repeat mode. 0x0 = In master mode, STT is automatically cleared after the START condition has been generated. In slave mode, if STT is 0, the I2C does not monitor the bus for commands from a master. As a result, the I2C performs no data transfers. 0x1 = In master mode, setting STT to 1 causes the I2C to generate a START condition on the I2C-bus. In slave mode, if STT is 1, the I2C monitors the bus and transmits/receives data in response to commands from a master.
12	Reserved	R	0h	These reserved bit locations are always read as zeros. A value written to this field has no effect.

Table 6-14. ICMDR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	STP	RW	0h	STOP condition bit (only applicable when the I2C is a master). The RM, STT, and STP bits determine when the I2C starts and stops data transmissions. Note that the STT and STP bits can be used to terminate the repeat mode. 0x0 = STP is automatically cleared after the STOP condition has been generated. 0x1 = STP has been set to generate a STOP condition when the internal data counter of the I2C counts down to 0.
10	MST	RW	0h	Master mode bit. MST determines whether the I2C is in the slave mode or the master mode. MST is automatically changed from 1 to 0 when the I2C master generates a STOP condition. 0x0 = Slave mode. The I2C is a slave and receives the serial clock from the master. 0x1 = Master mode. The I2C is a master and generates the serial clock on the SCL pin.
9	TRX	RW	0h	Transmitter mode bit. When relevant, TRX selects whether the I2C is in the transmitter mode or the receiver mode. 0x0 = Receiver mode. The I2C is a receiver and receives data on the SDA pin. 0x1 = Transmitter mode. The I2C is a transmitter and transmits data on the SDA pin.
8	XA	RW	0h	Expanded address enable bit. 0x0 = 7-bit addressing mode (normal address mode). The I2C transmits 7-bit slave addresses (from bits 6-0 of ICSAR), and its own slave address has 7 bits (bits 6-0 of ICOAR). 0x1 = 10-bit addressing mode (expanded address mode). The I2C transmits 10-bit slave addresses (from bits 9-0 of ICSAR), and its own slave address has 10 bits (bits 9-0 of ICOAR).
7	RM	RW	0h	Repeat mode bit (only applicable when the I2C is a master-transmitter). The RM, STT, and STP bits determine when the I2C starts and stops data transmissions. If the I2C is configured in slave mode, the RM bit is don't care. 0x0 = Non-repeat mode. The value in the data count register (ICCNT) determines how many data words are received/transmitted by the I2C. 0x1 = Repeat mode. Data words are continuously received/transmitted by the I2C until the STP bit is manually set to 1.
6	DLB	RW	0h	Digital loopback mode bit (only applicable when the I2C is a master-transmitter). This bit disables or enables the digital loopback mode of the I2C. Note that DLB mode in the free data format mode (DLB = 1 and FDF = 1) is not supported. 0x0 = Digital loopback mode is disabled. 0x1 = Digital loopback mode is enabled. In this mode, the MST bit must be set to 1 and data transmitted out of ICDXR is received in ICDRR after n clock cycles by an internal path, where: n = ((I2C input clock frequency/prescaled module clock frequency) 8) The transmit clock is also the receive clock. The address transmitted on the SDA pin is the address in ICOAR.
5	IRS	RW	0h	I2C reset bit. Note that if IRS is reset during a transfer, it can cause the I2C bus to hang (SDA and SCL are in a high-impedance state). 0x0 = The I2C is in reset/disabled. When this bit is cleared to 0, all status bits (in ICSTR) are set to their default values. 0x1 = The I2C is enabled.

Table 6-14. ICMDR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	STB	RW	0h	START byte mode bit (only applicable when the I2C is a master). As described in version 2.1 of the Philips I2C-bus specification, the START byte can be used to help a slave that needs extra time to detect a START condition. When the I2C is a slave, the I2C ignores a START byte from a master, regardless of the value of the STB bit. 0x0 = The I2C is not in the START byte mode. 0x1 = The I2C is in the START byte mode. When you set the START condition bit (STT), the I2C begins the transfer with more than just a START condition. Specifically, it generates: 1. A START condition 2. A START byte (0000 0001b) 3. A dummy acknowledge clock pulse 4. A repeated START condition The I2C sends the slave address that is in ICSAR.
3	FDF	RW	0h	Free data format mode bit. Note that DLB mode in the free data format mode (DLB = 1 and FDF = 1) is not supported. 0x0 = Free data format mode is disabled. Transfers use the 7-/10-bit addressing format selected by the XA bit. 0x1 = Free data format mode is enabled.
2-0	BC	RW	0h	Bit count bits. BC defines the number of bits (2 to 8) in the next data word that is to be received or transmitted by the I2C. The number of bits selected with BC must match the data size of the other device. Note that when BC = 0, a data word has 8 bits. 0x0 = 8 bits per data word 0x1 = Reserved 0x2 = 2 bits per data word 0x3 = 3 bits per data word 0x4 = 4 bits per data word 0x5 = 5 bits per data word 0x6 = 6 bits per data word 0x7 = 7 bits per data word

6.3.11 ICIVR Register (offset = 1A28h) [reset = 0h]

ICIVR is shown in Figure 6-23 and described in Table 6-15.

The I2C interrupt vector register (ICIVR) is used by the CPU to determine which event generated the I2C interrupt. If an interrupt is pending, the corresponding interrupt flag is set in the interrupt status register (ISR). If a new interrupt occurs, reading ICIVR clears the highest priority interrupt flag in ISR. Reading ICIVR also clears the corresponding status bit in ICSTR except ARDY, ICRRDY, ICXRDY, and AAS.

Note: You must read and clear ICIVR before starting a new interrupt; otherwise, ICIVR could contain an incorrect (old interrupt flags) value. The highest priority interrupts should also check the status register to verify if a lower priority interrupt occurred and changed the status register.

Figure 6-23. ICIVR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved					INTCODE		
R-0h					R-0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 6-15. ICIVR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-3	Reserved	R	0h	These reserved bit locations are always read as zeros. A value written to this field has no effect.
2-0	INTCODE	R	0h	Interrupt code. The binary-coded-interrupt vector indicates which interrupt has occurred. Reading the ICIVR clears the interrupt flag if other interrupts are pending, a new interrupt is generated. If there are more than one interrupt flag, reading the ICIVR clears the highest priority interrupt flag. Note that users must read (clear) the IVR before doing another start otherwise the IVR could contain incorrect (old interrupt flags) value. 0x0 = No interrupt occurred 0x1 = Arbitration-lost interrupt (AL) 0x2 = No-acknowledgment interrupt (NACK) 0x3 = Register-access-ready interrupt (ARDY) 0x4 = Receive-data-ready interrupt (ICRRDY) 0x5 = Transmit-data-ready interrupt (ICXRDY) 0x6 = Stop condition detected interrupt (SCD) 0x7 = Address-as-slave interrupt (AAS)

6.3.12 ICEMDR Register (offset = 1A2Ch) [reset = 1h]

ICEMDR is shown in Figure 6-24 and described in Table 6-16.

The I2C extended mode register (ICEMDR) is used to indicate which condition generates a transmit data ready interrupt.

Figure 6-24. ICEMDR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved						IGNACK	BCM
R-0h						RW-0h	RW-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 6-16. ICEMDR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	These reserved bit locations are always read as zeros. A value written to this field has no effect.
1	IGNACK	RW	0h	Ignore NACK mode. 0x0 = Master transmitter operates normally, that is, it discontinues the data transfer and sets the ARDY and NACK bits in ICSTR when receiving a NACK from the slave. 0x1 = Master transmitter ignores a NACK from the slave.
0	BCM	RW	1h	Backward compatibility mode bit. Determines which condition generates a transmit data ready interrupt. The BCM bit only has an effect when the I2C is operating as a slave-transmitter. 0x0 = The transmit data ready interrupt is generated when the master requests more data by sending an acknowledge signal after the transmission of the last data. 0x1 = The transmit data ready interrupt is generated when the data in ICDXR is copied to ICXSR.

6.3.13 ICPSC Register (offset = 1A30h) [reset = 0h]

ICPSC is shown in Figure 6-25 and described in Table 6-17.

The I2C prescaler register (ICPSC) is used for dividing down the I2C input clock to obtain the desired prescaled module clock for the operation of the I2C.

Figure 6-25. ICPSC Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
IPSC							
RW-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 6-17. ICPSC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	These reserved bit locations are always read as zeros. A value written to this field has no effect.
7-0	IPSC	RW	0h	I2C prescaler divide-down value. IPSC determines how much the I2C input clock is divided to create the I2C prescaled module clock: I2C clock frequency = I2C input clock frequency/(IPSC + 1) Note: IPSC must be initialized while the I2C is in reset (IRS = 0 in ICMDR).

6.3.14 ICPID1 Register (offset = 1A34h) [reset = 106h]

ICPID1 is shown in Figure 6-26 and described in Table 6-18.

The I2C peripheral identification registers (ICPID1) contain identification data (class, revision, and type) for the peripheral.

Figure 6-26. ICPID1 Register

15	14	13	12	11	10	9	8
CLASS							
R-1h							
7	6	5	4	3	2	1	0
REVISION							
R-6h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 6-18. ICPID1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	CLASS	R	1h	Identifies class of peripheral.
7-0	REVISION	R	6h	Identifies revision of peripheral.

6.3.15 ICPID2 Register (offset = 1A38h) [reset = 5h]

ICPID2 is shown in Figure 6-27 and described in Table 6-19.

The I2C peripheral identification register (ICPID2).

Figure 6-27. ICPID2 Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							
7	6	5	4	3	2	1	0
TYPE							
R-5h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 6-19. ICPID2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	These reserved bit locations are always read as zeros. A value written to this field has no effect.
7-0	TYPE	R	5h	Identifies type of peripheral.

Inter-IC Sound (I2S) Bus

This chapter describes the features and operation of Inter-IC Sound (I2S) Bus.

Topic **Page**

7.1 Introduction

The following sections describe the features and operation of Inter-IC Sound (I2S) Bus. This peripheral allows serial transfer of full duplex streaming data, usually streaming audio, between DSP and an external I2S peripheral such as an audio codec.

7.1.1 Purpose of the Peripheral

The I2S bus is used as an interface for full-duplex serial ports such as those found in audio or voice-band analog to digital converters (ADC) to acquire audio signals or digital-to analog converters (DAC) to drive speakers and headphones.

7.1.2 Features

The I2S bus supports the following features:

- Full-duplex (transmit and receive) communication.
- Double buffered data registers that allow for continuous data stream.
- Most significant bit (MSB) - first data transfers.
- I2S/Left-justified and DSP serial data communication formats with a data delay of 1 or 2 bits.
- Data word-lengths of 8, 10, 12, 14, 16, 18, 20, 24, or 32 bits.
- Ability to sign-extend received data samples for easy use in signal processing algorithms.
- Ability to pack multiple data words into CPU or DMA accessible data registers to reduce interrupts for more efficient operation.
- Programmable polarity for both frame synchronization and bit-serial clocks.
- Digital loopback of data from transmit to receive data register(s) for application code debug.
- Stereo (in I2S/Left-justified or DSP data formats) or mono (in DSP data format) mode.
- Programmable divider for serial data clock (bit-clock) generation when I2S bus is used as a master device.
- Programmable divider for frame sync clock generation when I2S bus is used as the master device.
- Detection of over-run, under-run, and frame-synchronization error conditions.

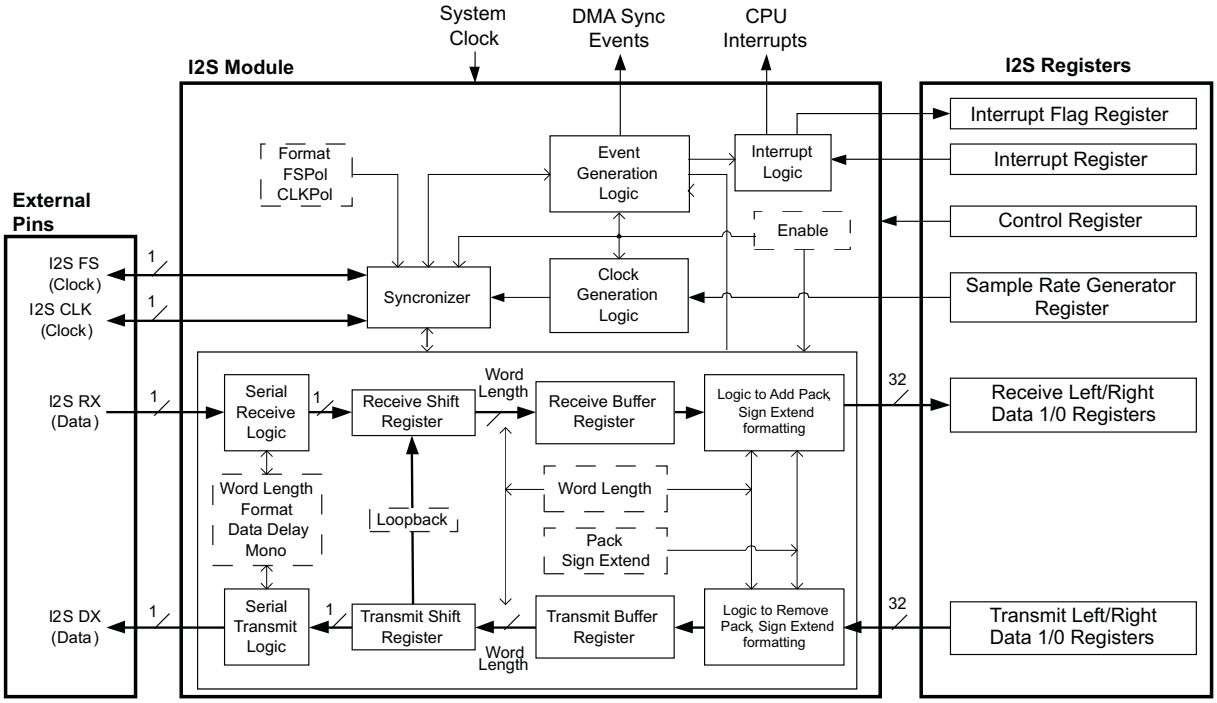
The DSP includes four independent I2S modules.

7.1.3 Functional Block Diagram

Figure 7-1 is a functional block diagram of the I2S bus illustrating the different control, data transfer, clock generation and event management blocks and their interactions. The I2S peripheral has a set of control and data registers which the CPU can access through its I/O space. The DMA can also make 32-bit accesses to receive and transmit data registers for efficient data transfer.

The bus is configured by writing to the I2S*n* Serializer Control Register (I2SSCTRL) bit fields. The bit fields in this register determine the communication protocol over the I2S bus and the arrangement of data in the data registers.

Figure 7-1. Functional Block Diagram



Data on the I2S*n*_RX pin is shifted serially into the Receive Shift Register and then copied into the Receive Buffer Register. The data is then copied to I2S*n* Receive Left/Right Data *n* Registers. For each channel (left and right), these registers can be accessed as two 16-bit registers by the CPU or as a 32-bit register by the DMA. Similarly, the I2S*n* Transmit Left/Right Data *n* Registers store the data to be transmitted out of the I2S peripheral. The CPU or DMA writes the transmit data to the I2S*n* Transmit Left/Right Data *n* Registers which is then copied to the Transmit Shift Register via the Transmit Buffer Register and shifted serially out to I2S*n*_DX pin. This structure allows internal data movement and external data communications simultaneously. Data handling and movement is discussed in further detail in later sections.

The control block consists of internal clock generation, frame synchronization signal generation, and their control. The I2S*n* Sample Rate Generator Register (I2SSRATE) contains fields to configure the frame-synchronization and bit-clock dividers to drive the I2S*n*_FS and I2S*n*_CLK clocks when the I2S peripheral is configured as a master device. When configured as a slave device, the internal clock generation logic is disabled and frame synchronization is performed on the clocks generated by the external master I2S device (see Section 7.2.2). The polarities of the bit-clock and the frame-synchronization can be set by the CLKPOL bit and FSPOL bit respectively in the I2SSCTRL register. The I2S supports a data delay of 1 bit or 2 bits as configured by the DATADLY bit in the I2SSCTRL register.

The I2S peripheral can be configured to interrupt the CPU by writing to the I2Sn Interrupt Mask Register (I2SINTMASK). When interrupts are enabled, the event-generation block posts a transmit interrupt when transmit data registers are empty and a receive interrupt when receive data registers are full. The corresponding flag is set in the I2Sn Interrupt Flag Register (I2SINTFL). In addition to data transaction interrupts, error events are also flagged in this register. Error events are not connected to interrupts on the CPU. The I2S also sends synchronization events corresponding to the transmit and receive events to the DMA controller associated with the I2S module. The I2SINTMASK register has no effect on DMA sync signal generation events. (See Section 7.2.10, Section 7.2.11, and)

7.1.4 Industry Standard(s) Compliance

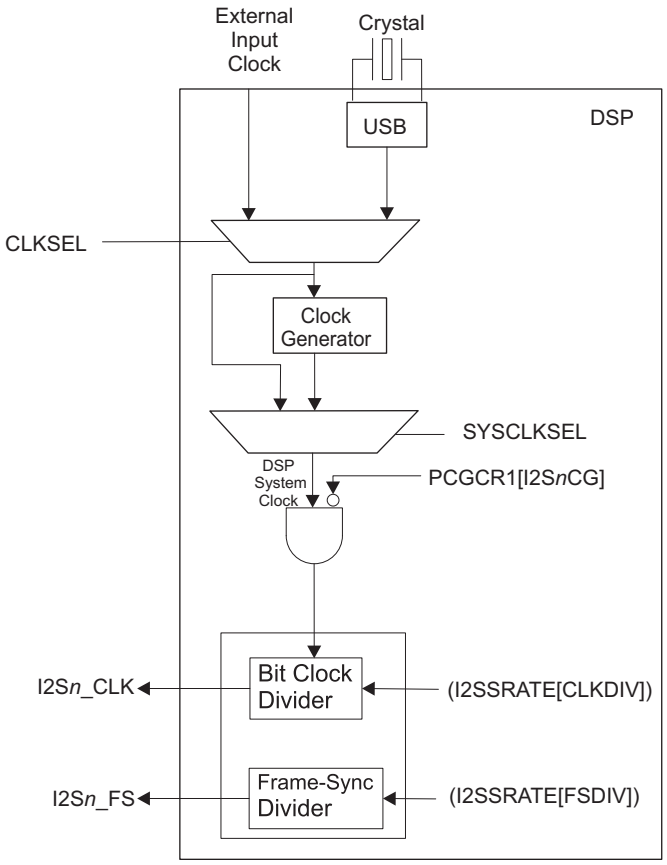
This Inter-IC Sound (I2S) Bus is compliant to industry I2S Bus Standard.

7.2 Architecture

7.2.1 Clock Control

As shown in Figure 7-2, the I2S bus is driven by the system clock. Unused I2S modules can be independently idled (clock-gated) via the peripheral clock gating configuration register 1 (PCGCR1) for power dissipation savings. Each I2S bus should be brought out of idle before being programmed. For more details, see Section 7.2.12.

Figure 7-2. Inter-IC Sound Clock Control Diagram



If the I2S bus is configured as the master device, the DSP clock generator may need to be programmed to achieve an appropriate system clock so that the I2S clock dividers can generate the required clock rates. For more information on the DSP clock generation options, see Chapter 1, *System Control*.

7.2.2 I2S Clock Generator

The I2S*n* Sample Rate Generator Register (I2SSRATE) controls the clock generation logic in the I2S bus. In slave mode (MODE = 0 in I2SSCTRL), the required clock signals (I2Sn_CLK and I2Sn_FS) are generated by the external master I2S device and the internal I2S clock generator is not used. Configuring the I2SSRATE register has no effect in this mode. However, when configured as a master device (MODE = 1), the I2S module generates these clocks by dividing the system clock by a value calculated from the CLKDIV and FSDIV fields programmed in the I2SSRATE register. The clocks can be calculated as shown below:

I2S*n*_CLK = SystemClock / ($2^{CLKDIV+1}$)

I2S*n*_FS = I2S*n*_CLK / ($2^{FSDIV+3}$)

I2S*n*_CLK is the bit-clock that determines the rate at which data bits are transferred on the serial I2S bus. I2S*n*_FS is referred to as the frame-sync clock or word clock and is the rate at which a data word is transferred to and/or from the I2S module. This can also be seen as the frequency at which data (audio from a microphone for example) is sampled by an analog-to-digital converter (ADC) or an audio codec.

The clock divide-by value, $2^{FSDIV+3}$, that derives the frame-sync clock from the bit-clock (as shown above), gives the number of data bits (bit-clocks) in one cycle of the frame-sync clock. Since one cycle of the frame-sync clock should accommodate two data words (one left and one right channel data word) for stereo operation and one data word (one left channel only) for mono operation, the following restrictions apply while choosing an appropriate setting for FSDIV:

$2^{FSDIV+3} \geq 2 * WDLNGTH$ (for stereo mode)

$2^{FSDIV+3} \geq WDLNGTH$ (for mono mode)

For example, to achieve a particular sampling rate of I2S*n*_FS = 48000 Hz with stereo operation of data length of 16 bits, the value of the FSDIV bit in the I2SSRATE register should be first chosen such that:

$2^{FSDIV+3} \geq 2*16 = 32.$

If we choose

FSDIV = 2 (010 binary)

the resultant I2S*n*_CLK can be calculated as:

I2S*n*_CLK = I2S*n*_FS * ($2^{FSDIV+3}$) = 48000 * 32 = 1.536 MHz.

Based on application requirements, if the DSP needs to be run at a minimum system clock or DSP clock of 45 MHz, the CLKDIV bit in the I2SSRATE register can be chosen such that,

SystemClock ≥ I2S*n*_CLK * $2^{CLKDIV+1}$ ≥ 45 MHz

Hence we should choose:

CLKDIV = 4 (100 binary)

Which will give us,

SystemClock = 1.536 MHz * 2^{4+1} = 49.15 MHz

As a result, the DSP clock generator should be configured to generate the required clock of 49.15 MHz. Due to limitations/restrictions of the DSP clock generator, it may not be possible to generate the exact system clock required, in which case I2S*n*_FS will deviate from the expected value. While it is sufficient to choose a setting for FSDIV such that 2FSDIV+3 is equal to the required number of data bits per frame-sync clock (as shown in example above), it may sometimes be necessary to choose a higher setting so that a faster bit-clock can be achieved. For a given system clock, this expedites transfer of data bits of the programmed word length on the I2S bus. As a result, the interrupts/events occur earlier in the frame-sync cycle, providing more time for the CPU/DMA to service the interrupt/event before the next interrupt/event. This is a particularly useful technique when the I2S uses CPU to handle data transfers to/from its data registers.

NOTE: I2S peripheral clock generator should only be configured if the I2S is configured as a master device. When the I2S is configured as the slave, the external master device supplies the required clocks.

7.2.3 Signal and Pin Descriptions

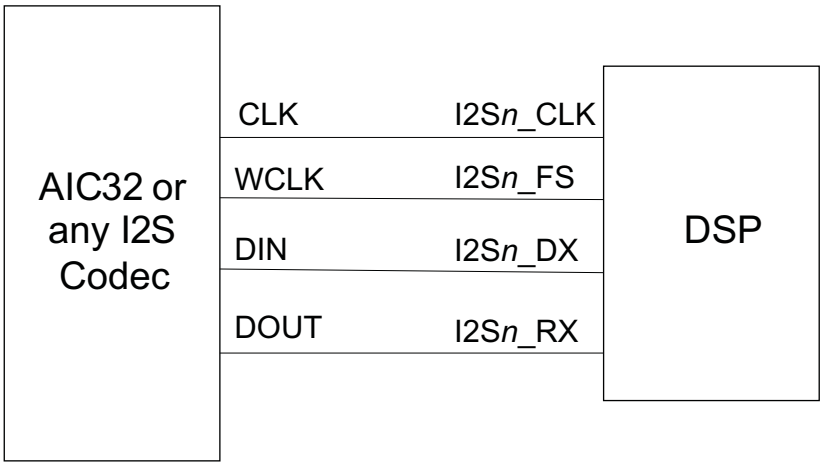
The I2S bus is a four-wire interface with two clock pins, bit-serial clock (I2S*n*_CLK) and frame-synchronization or word clock (I2S*n*_FS), and two data pins, serial data transmit (I2S*n*_DX) and serial data receive (I2S*n*_RX), for data communication as shown in Figure 7-1. The I2S*n*_CLK and I2S*n*_FS pins are bi-directional based on whether the I2S peripheral is configured as a master or slave device.

Table 7-1. I2S Signal Descriptions

Name	Signal	Description
I2S*n*_CLK	INPUT /OUTPUT	I2S Clock
I2S*n*_FS	INPUT /OUTPUT	I2S Frame Sync Clock
I2S*n*_DX	OUTPUT	I2S Data Transmit
I2S*n*_RX	INPUT	I2S Data Receive

The diagram below is a typical connection between I2S interface to an audio or voice-band Codec.

Figure 7-3. Block Diagram of I2S Interface to Audio/Voice Band Codec



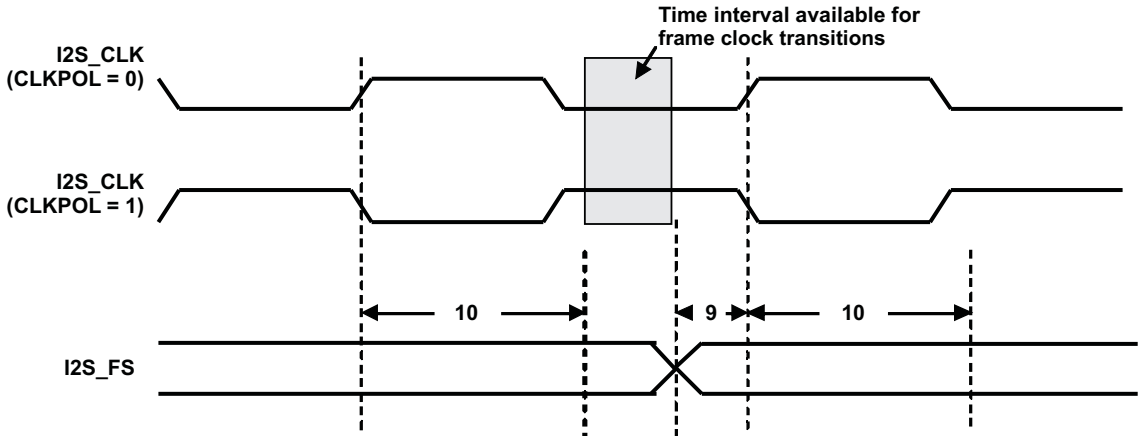
7.2.3.1 Pin Multiplexing

Depending on the I2S bus being used, the DSP should be configured to route those I2S signals to the multiplexed Serial Port 0, Serial Port 1, or Parallel Port pins by writing to the External Bus Selection Register (EBSR). For more information on pin multiplexing, see Chapter 1, *System Control*.

7.2.4 Frame Clock Timing Requirement in Slave Mode

When configured as the slave, frame clock (I2S_FS) is required to be latched on both edges of the bit clock (I2S_CLK), which are generated by the external master device. This imposes an additional constraint on the timing of I2S_FS as illustrated in Figure 7-4. The generated frame clock should meet the specified setup and hold requirements with respect to the sampling edge of the generated bit clock. For actual timing requirements, see the I2S section of the datasheet. These constraints imply that the frame clock transitions should occur in the time window as indicated by the shaded region in the figure.

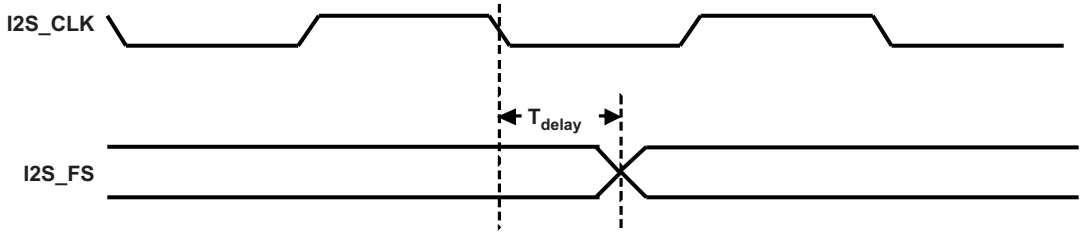
Figure 7-4. I2S Frame Clock Timing Constraint in Slave Mode



No.	Parameter	Description
9	$t_{su(FSV-CLKH)}$	Minimum setup time, I2S_FS valid before I2S_CLK high (CLKPOL = 0)
	$t_{su(FSV-CLKL)}$	Minimum setup time, I2S_FS valid before I2S_CLK low (CLKPOL = 1)
10	$t_{h(CLKH-FSV)}$	Minimum hold time, I2S_CLK high to I2S_FS (CLKPOL = 0)
	$t_{h(CLKL-FSV)}$	Minimum hold time, I2S_CLK low to I2S_FS (CLKPOL = 1)

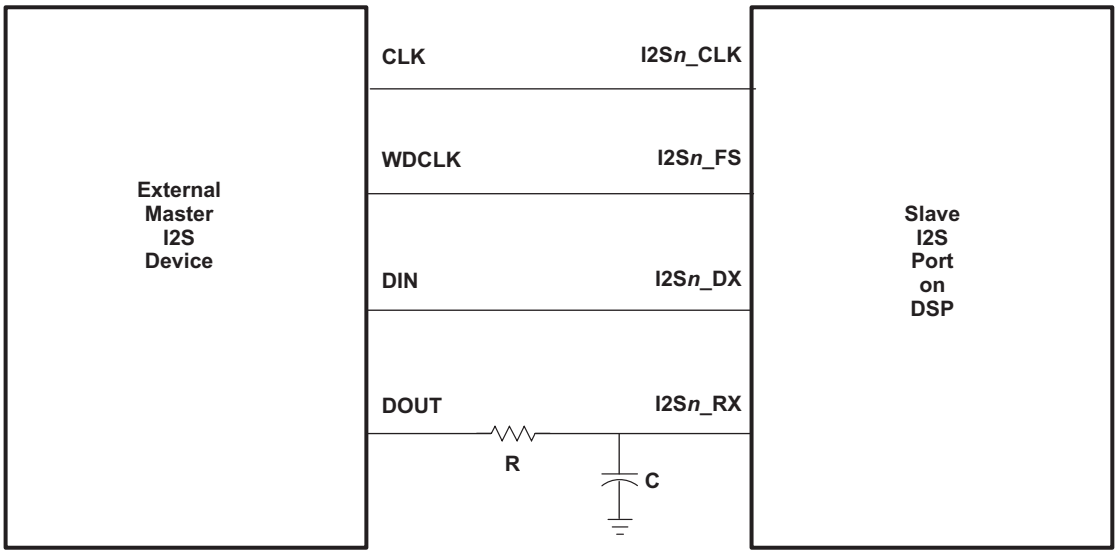
Devices (ADCs, DACS and audio/voice-band codecs) that interface to the I2S module usually only specify a maximum delay for the frame clock transition from the falling edge of the bit clock in master mode as indicated by parameter T_{delay} in Figure 7-5.

Figure 7-5. Typical Frame Clock Timing Specification



Synchronization issues may occur if the frame clock transitions close to the falling edge of the bit clock violating the previously described hold requirement resulting in incorrect data transfer. In these circumstances, the frame clock should be delayed with respect to the bit clock by introducing a time delay in its signal path as shown in Figure 7-6. The RC circuit delays the frame clock by a value given by the relation Trc = RC.

Figure 7-6. Delaying I2S Frame Clock to Overcome Synchronization Problems



NOTE: Signal should be probed as close to the device pins as possible for better results.

7.2.5 Protocol Description

The I2S bus communicates with a corresponding external I2S peripheral in a series of 1's and 0's. This series has a hierarchical organization that can be described in terms of bits, words, and frames.

A bit is the smallest entity in the serial stream. A "1" is represented by logic high on the data pin for the entire duration of a single bit clock. A "0" is represented by a logic low for the entire duration of a single bit clock.

A word is a group of bits that make up the data being transmitted or received. The length of the word is programmed by the user in the WDLNGTH field in the I2S*n* Serializer Control Register (I2SSCTRL).

A frame is a group of words (usually one – mono or two – stereo) that make up the data being transmitted or received. The number of bit clocks per frame and the frame rate (sampling frequency) is programmed by the user in the I2S*n* Sample Rate Generator Register (I2SSRATE).

I2S supports two serial data communication formats with external I2S devices: I2S/Left-justified format and DSP format. The I2S format is a specialized case of the more general left-justified data format. In DSP mode, the frame is marked between two consecutive pulses of the frame sync signal. On I2S, the frame is marked by a whole clock cycle of the frame sync signal with 50% duty cycle.

7.2.5.1 I2S/Left-Justified Format

In the left-justified format, the frame-synchronization or word clock has a 50% duty cycle indicating dual channel data fields with left channel data transferred during one half of the cycle and right channel data transferred during the other half. The MSB-first data is transferred serially, left justified in its own field with appropriate bit delays.

As shown in Figure 7-7, the typical I2S format utilizes left-justified format with a data delay of one bit and low frame synchronization pulse for left channel data and high pulse for right channel data. Serial data sent by the transmitter may be synchronized with either the trailing or the leading edge of serial clock I2S*n*_CLK. However, the serial data must be latched by the receiver on the leading edge of I2S*n*_CLK. In this format, the MSB of the left channel is valid on the second leading edge of the bit-clock, I2S*n*_CLK after the trailing edge of the frame-synchronization clock, I2S*n*_FS. Similarly the MSB of the right channel is valid on the second leading edge of I2S*n*_CLK after the leading edge of I2S*n*_FS.

Figure 7-7. Timing Diagram for Left-Justified Mode with Inverse Frame-Sync Polarity and One-Bit Delay

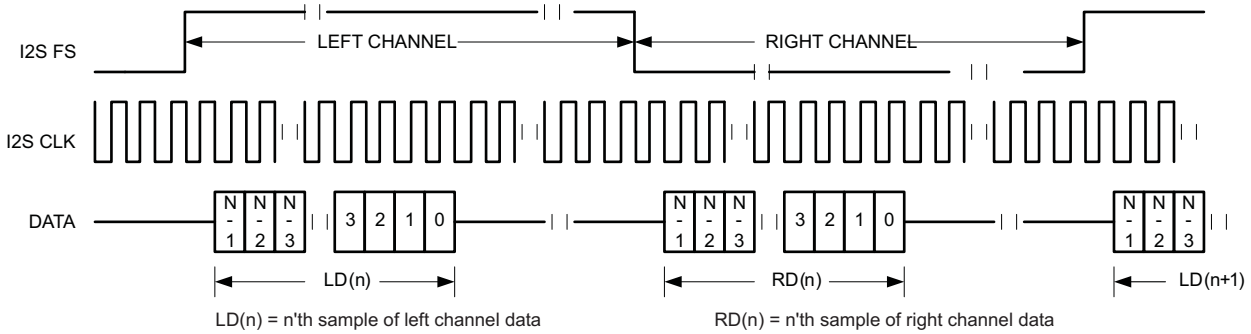


Figure 7-8. Timing Diagram for I2S Mode

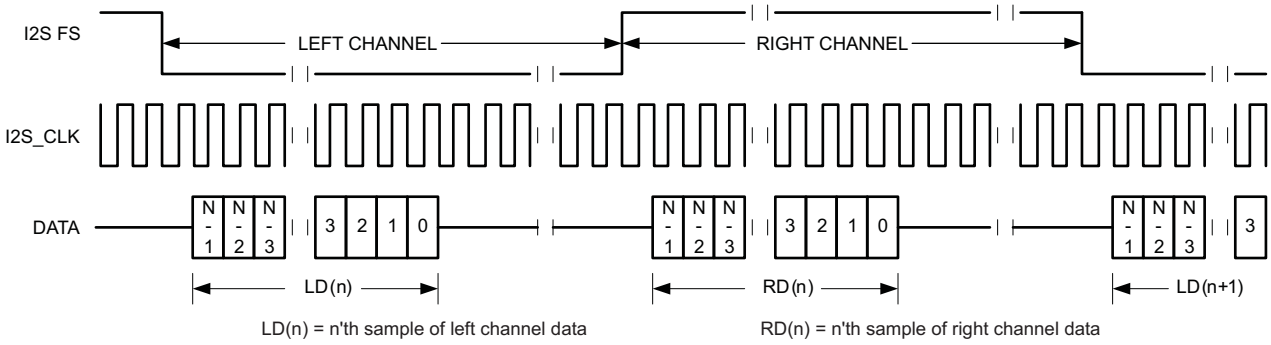
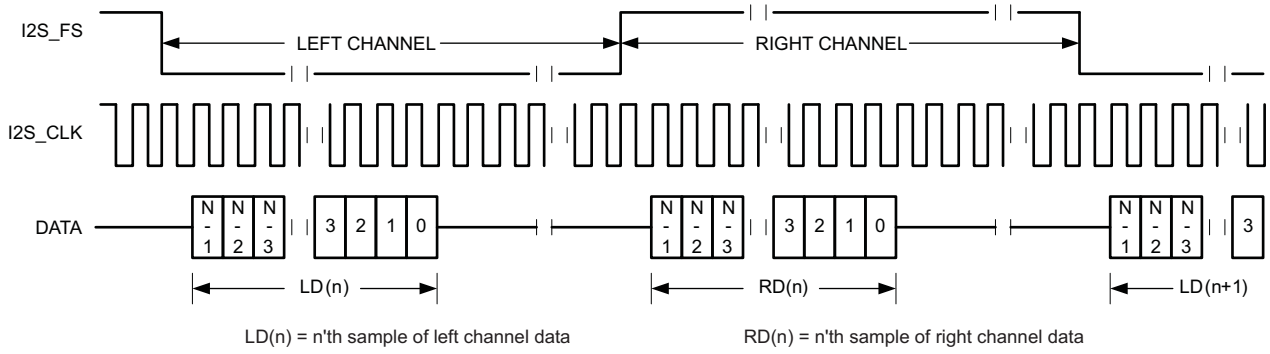


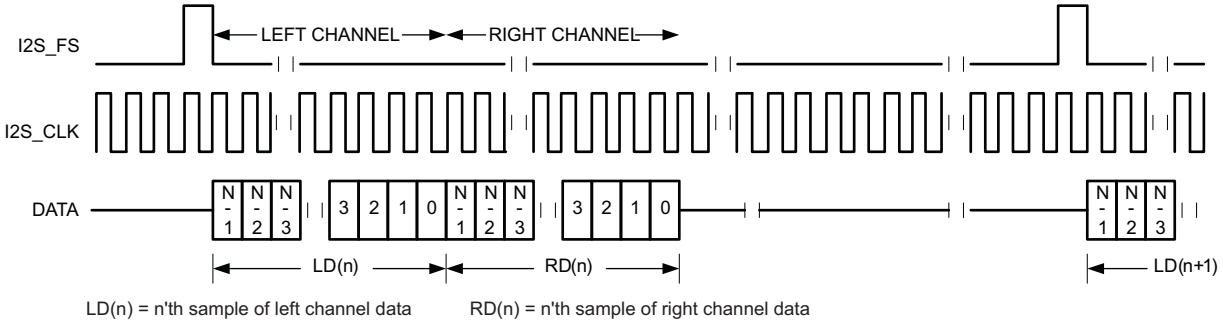
Figure 7-9. Timing Diagram for I2S Mode with Inverse Bit-Clock Polarity



7.2.5.2 DSP Format

In DSP format, the trailing edge of the frame-synchronization pulse, I2S*n*_FS, starts the data transfer with the left channel data first and immediately followed by the right channel data. Each data bit is valid on the trailing edge of the bit-clock, I2S*n*_CLK. The first data sample can be delayed by 1 bit or 2 bits after the trailing edge of I2S*n*_FS. With one bit delay, the MSB coincides with the trailing edge of I2S*n*_FS. With two bit delay, the MSB follows the trailing edge of I2S*n*_FS after one I2S*n*_CLK. Figure 7-10 illustrates DSP format operation with a one-bit data delay.

Figure 7-10. Timing Diagram for DSP Mode With One-Bit Delay



NOTE:

- The I2S*n*_DX and I2S*n*_RX pins are tri-stated during unused bit clocks in a frame.
- In I2S/Left-justified format:
  - Mono operation is not supported due to the 50% duty cycle restriction on the frame-synchronization clock.
  - The number of I2S*n*_CLKs should be greater than or equal to twice the configured data word-length.
- In DSP format:
  - The number of I2S*n*_CLKs should be greater than or equal to twice the configured data word-length for stereo operation.
  - The number of I2S*n*_CLKs should be greater than or equal to the configured data word-length for mono operation.

7.2.6 I2S Data Transfer and Control Behavior

When the I2S module is enabled, MSB-first data transfer starts when the appropriate level is detected on the frame-sync clock. Data in the Transmit Shift Register is shifted out serially to the I2Sn_DX while data bits are shifted in serially from the I2Sn_RX pin to the Receive Shift Register on the falling or leading edge of the bit-clock as programmed in the CLKPOL field of the I2SSCTRL. Data for the left channel is transferred first followed by the right channel data.

The module generates the transmit interrupt/event to indicate that the Transmit Left/Right Data1/0 registers should be filled with valid data for the next set of transfers. The transmit and receive interrupts should be enabled in the I2SINTMASK register if CPU transfers are desired; if DMA is used to transfer data these interrupts should be disabled. See Section 7.2.10, *Interrupt Support*, and Section 7.2.11, *DMA Event Support*.

The CPU or DMA servicing the transmit interrupt/event writes the next valid data to the above mentioned registers. Failure to do so before the next frame-sync cycle will result in the OUERROR being flagged in the I2SINTFL register (assuming that error detection has been enabled in the I2SINTMASK register). At the next frame-sync cycle, data from the Transmit Left/Right Data 1/0 registers is copied into the Transmit Buffer register and then to the Transmit Shift register. See Figure 7-1, *Functional Block Diagram*.

NOTE: Data should be written into the Transmit Left/Right Data1/0 registers only on a transmit interrupt. Data can not be preloaded into these registers before enabling the I2S module or before receiving the first transmit interrupt.

When the required number of data words is received in the Receive Shift register (one for mono or two for stereo), the data is moved into the Receive Buffer register and ultimately into the Receive Left/Right Data1/0 registers. A receive interrupt/event is generated at this point. The CPU or DMA servicing this interrupt reads the register into memory. Failure to do so before the next frame-sync cycle will result in the OUERROR being flagged in the I2SINTFL register (assuming that error detection has been enabled in the I2SINTMASK register).

Transmit and receive interrupts/events are continuously generated after every transfer from the transmit data registers to the transmit buffer register and from the receive buffer register to the receive data registers respectively. If packed mode is enabled (PACK = 1 in I2SSCTRL), interrupts/events are generated after all the required number of data words have been transmitted/received (see Section 7.2.8).

7.2.7 I2S Data Transfer Latency

Due to the buffered nature of the I2S module, there exists some latency in the transmit and receive paths (from the Transmit Data registers to the I2S_DX pin and I2S_RX pin to Receive Data registers) which is dependent on the desired configuration of the module. The latency may not be of consequence when the I2S module in its intended scope of a streaming audio peripheral. However, it is documented here in the interests of completeness. It will also help explain observed loopback data (LOOPBACK=1 in I2SSCTRL) as the latency is also present in loopback mode.

7.2.7.1 Transmit Path Latency

After the I2S is enabled, the first valid data sample on the I2Sn_DX pin will appear after:

- Five frame-sync clocks if PACK mode is used (PACK=1 in I2SSCTRL) or,
- Three frame-sync clocks if PACK mode is not used (PACK=0 in I2SSCTRL)

Hence there is a latency of three or five samples (for each channel) before the first data sample that is written to the Transmit Left/Right Data 1/0 registers after the first transmit interrupt/event. During this time, the I2S transmits zeros that should be discarded or ignored.

7.2.7.2 Receive Path Latency

After the I2S is enabled, the receive path starts receiving data after:

- 1 or 2 frame-sync clocks for 8-, 18-, 20-, 24-, or 32-bit data depending on other configuration
- 1, 2 or 3 frame-sync clocks for 10-, 12-, 14-, or 16-bit data depending on other configuration

7.2.7.3 Loopback Path Latency

The internal loopback mode (LOOPBACK=1 in I2SSCTRL) can be used as a debug tool to verify the user's program to service I2S interrupts/events. This is different from an external loopback which would require the I2Sn_DX pin to be connected to the I2Sn_RX pin. In the internal loopback mode, data from the Transmit Shift register is directly routed to the Receive Shift register, changing the data latency as given below:

- If pack mode is used (PACK=1 in I2SSCTRL)
  - Ignore the first 6 samples received for 8-bit data and FSDIV=000 in I2SSRATE
  - Ignore the first 5 samples received for 8-bit data and FSDIV > 000 in I2SSRATE
  - Ignore the first 6 samples received for 10-, 12-, 14- or 16-bit data
- If pack mode is not used (PACK=0 in I2SSCTRL), ignore the first 2 samples received.

7.2.8 Data Packing and Sign Extension Options

The I2S bus supports the use of packed (multiple) and/or sign extended data words in its data registers to reduce software overheads in servicing interrupts for every data sample and for ease of data handling in software algorithms.

NOTE: Using the Pack or Sign Extend options does not affect transmission of data samples over the serial I2S bus; it only affects how data words are arranged in the Receive and Transmit Data Registers.

7.2.8.1 Data Pack Mode

Setting the PACK bit field in the I2SSCTRL register enables data packing in the 32-bit I2S data registers for word-lengths of 8, 10, 12, 14 and 16 bits. This mode can be used in the following scenario:

- Using DMA to transfer data samples to make better use of data buffers.
- Using CPU to transfer data samples to reduce interrupt overheads.

During the receive operation, the I2S bus puts successive data samples into the I2S*n* Receive Left/Right Data *n* Registers (Data 1 register first, Data 0 register next) before generating the interrupt/event. The transmit data is expected in a similar format in the I2S*n* Transmit Left/Right Data *n* Registers. Four 8-bit data samples or two 10, 12, 14 or 16-bit data samples can be packed in the data registers, as shown in Section 7.2.8.3.

The advantages of using the PACK mode can be seen as given below:

- Reduces the number of I2S interrupts/events, which results in reducing interrupt overheads and the better use of bus bandwidth.
- Efficient use of internal/on-chip memory if DMA is used for transferring data between I2S and main memory. Since the DMA transfers a double-word (all 32-bits of the I2S*n* Transfer Left/Right Data *n* Registers) during each transaction, using packed I2S*n* Receive/Transmit Left/Right Data *n* Registers results in efficient transfer of data samples. Hence, for a given number of samples, size of data buffers is reduced by a factor of two for 10-, 12-, 14- or 16-bit word length or by a factor of four for 8-bit word length.

Figure 7-11 and Figure 7-12 illustrate packed and unpacked data receive behavior for mono transmission of four 12-bit data samples (sign extension is not enabled). When pack mode is not used, the I2S module stores 12-bit data left-justified in MSW (with trailing zeros) and generates DMA event resulting in zeros stored in alternate memory locations. When pack mode is used, the module packs 12-bit data left-justified in MSW first and then LSW (with trailing zeros) and generates DMA event once every two transfers resulting in better utilization of memory.

Figure 7-11. Example of Unpacked 12-Bit Data Receive

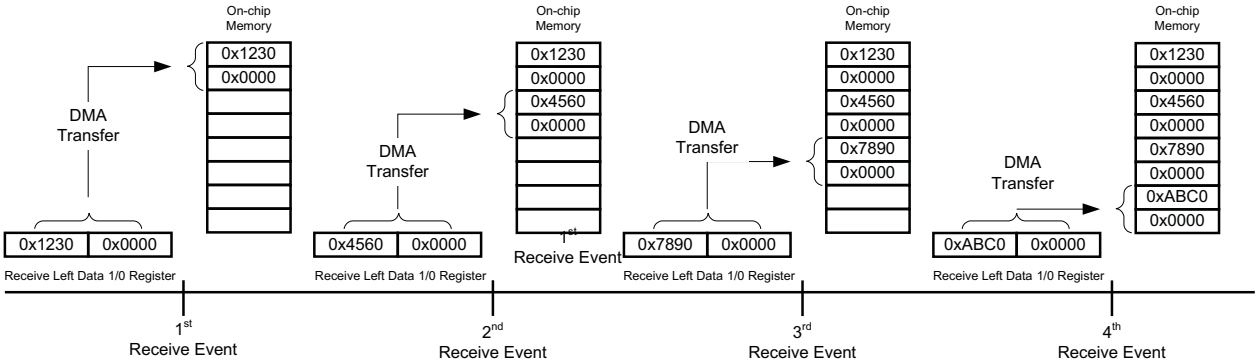
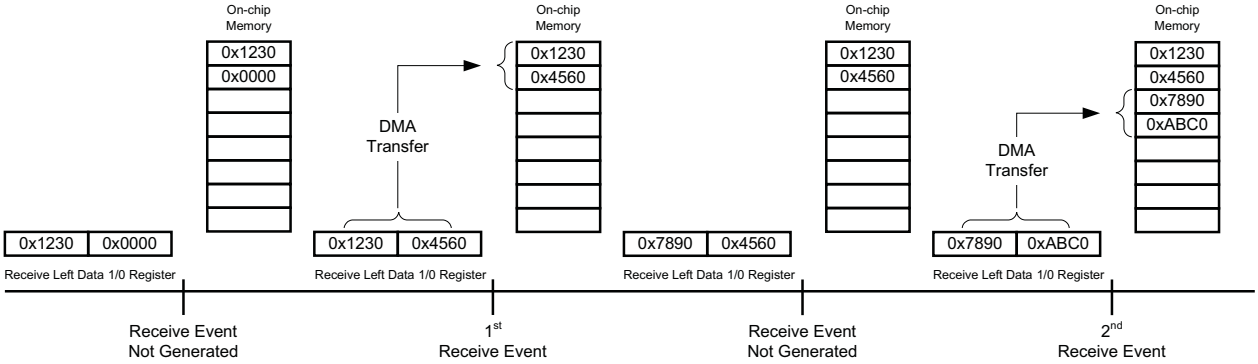


Figure 7-12. Example of Packed 12-Bit Data Receive



7.2.8.2 Data Sign Extend Mode

The I2S peripherals can be configured to work with sign extended data samples for ease of use in commonly used S16 or S32 data representations in signal processing algorithms. Data words of lengths 8-, 10-, 12- or 14-bits are sign-extended (right-justified) to 16 bits (8-bit data cannot be sign extended in packed mode as four samples are packed into the 32-bit data registers). Data of word lengths 18-, 20- or 24-bits are sign extended (right-justified) to 32 bits.

The choice of PACK, SIGN_EXT and WDLNGTH bit options in the I2SSCTRL register affects the arrangement of received data in the I2S*n* Receive Left/Right Data *n* Registers. Similarly, the I2S peripheral expects data to be arranged in a similar fashion in the I2S*n* Transmit Left/Right Data *n* Registers for the correct data value to be shifted out to the I2S*n*_DX pin.

Table 7-2 illustrates sign extension behavior with an example.

Table 7-2. Example of Sign Extension Behavior

		Data Register With SIGN_EXT=0		Data Register With SIGN_EXT=1	
Word Length	Data on I2S Bus	Data 1 Register	Data 0 Register	Data 1 Register	Data 0 Register
8	0xA1	0xA100	0x0000	0xFFA1	0x0000
10	0x2B7	0xAD80	0x0000	0xFEB7	0x0000
12	0x6AA	0x6AA0	0x0000	0x06AA	0x0000
14	0x3999	0xE666	0x0000	0xF999	0x0000
16	0x29CC	0x29CC	0x0000	0x29CC	0x0000
18	0x19EFA	0x67BE	0x8000	0x0001	0x9EFA
20	0x7ADE1	0x7ADE	0x1000	0x0007	0xADE1
24	0x98A311	0x98A3	0x1100	0xFF98	0xA311
32	0xD16AEE09	0xD16A	0xEE09	0xD16A	0xEE09

7.2.8.3 PACK and Sign Extend Data Arrangement for Various Word Lengths

The tables in this section describe sign extended and packed data arrangement for each configurable word length. Data samples are indicated by x[0], x[1], x[2] and x[3] in increasing temporal order (x[3] is the most recent sample).

7.2.8.3.1 8-Bit Word Length

Table 7-3. PACK and Sign Extend Data Arrangement for 8-Bit Word Length

	I2S Receive/Transmit Left or Right Data 1 Register			I2S Receive/Transmit Left or Right Data 0 Register		
PACK = 0 SIGN_EXT = 0	15 8	7 0		15 0		
	x(0)	0		0		
PACK = 0 SIGN_EXT = 1	15 8	7 0		15 0		
	Sign Bits (Bit #7)	x(0)		0		
PACK = 1 SIGN_EXT = 0	15 8	7 0		15 8	7 0	
	x(0)	x(1)		x(2)	x(3)	
PACK = 1 SIGN_EXT = 1	15 8	7 0		15 8	7	
	x(0)	x(1)		x(2)	x(3)	

7.2.8.3.2 10-Bit Word Length

Table 7-4. PACK and Sign Extend Data Arrangement for 10-Bit Word Length

	I2S Receive/Transmit Left or Right Data 1 Register			I2S Receive/Transmit Left or Right Data 0 Register		
PACK = 0 SIGN_EXT = 0	15 6	5 0		15		0
	x(0)	0		0		
PACK = 0 SIGN_EXT = 1	15 10	9 0		15		0
	Sign Bits (Bit #9)	x(0)		0		
PACK = 1 SIGN_EXT = 0	15 6	5 0		15 6	5	0
	x(0)	0		x(1)	0	
PACK = 1 SIGN_EXT = 1	15 10	9 0		15 10	9	
	Sign Bits (Bit #9)	x(0)		Sign Bits (Bit #9)	x(1)	

7.2.8.3.3 12-Bit Word Length

Table 7-5. PACK and Sign Extend Data Arrangement for 12-Bit Word Length

	I2S Receive/Transmit Left or Right Data 1 Register			I2S Receive/Transmit Left or Right Data 0 Register		
PACK = 0 SIGN_EXT = 0	15 4	3 0		15		0
	x(0)	0		0		
PACK = 0 SIGN_EXT = 1	15 12	11 0		15		0
	Sign Bits (Bit #11)	x(0)		0		
PACK = 1 SIGN_EXT = 0	15 4	3 0		15 4	3	0
	x(0)	0		x(1)	0	
PACK = 1 SIGN_EXT = 1	15 12	11 0		15 12	11	
	Sign Bits (Bit #11)	x(0)		Sign Bits (Bit #11)	x(1)	

7.2.8.3.4 14-Bit Word Length

Table 7-6. PACK and Sign Extend Data Arrangement for 14-Bit Word Length

	I2S Receive/Transmit Left or Right Data 1 Register			I2S Receive/Transmit Left or Right Data 0 Register		
PACK = 0 SIGN_EXT = 0	15 2	1 0		15		0
	x(0)	0		0		
PACK = 0 SIGN_EXT = 1	15 14	13 0		15		0
	Sign Bits (Bit #13)	x(0)		0		
PACK = 1 SIGN_EXT = 0	15 2	1 0		15 2	1	0
	x(0)	0		x(1)	0	
PACK = 1 SIGN_EXT = 1	15 14	13 0		15 14	13	
	Sign Bits (Bit #13)	x(0)		Sign Bits (Bit #13)	x(1)	

7.2.8.3.5 16-Bit Word Length

Table 7-7. PACK and Sign Extend Data Arrangement for 16-Bit Word Length

	I2S Receive/Transmit Left or Right Data 1 Register		I2S Receive/Transmit Left or Right Data 0 Register	
PACK = 0 SIGN_EXT = 0	15	0	15	0
	x(0)		0	
PACK = 0 SIGN_EXT = 1	15	0	15	0
	x(0)		0	
PACK = 1 SIGN_EXT = 0	15	0	15	0
	x(0)		x(1)	
PACK = 1 SIGN_EXT = 1	15	0	15	
	x(0)		x(1)	

7.2.8.3.6 18-Bit Word Length

Table 7-8. PACK and Sign Extend Data Arrangement for 18-Bit Word Length

	I2S Receive/Transmit Left or Right Data 1 Register			I2S Receive/Transmit Left or Right Data 0 Register		
PACK = 0 SIGN_EXT = 0	15		0	15	14 13	0
	x(0) Bits[17-2]			x(0) Bits[1-0]		0
PACK = 0 SIGN_EXT = 1	15	2 1	0	15		0
	Sign Bits(Bit #17)	x(0) Bits[17-16]		x(0) Bits[15-0]		
PACK = 1 SIGN_EXT = 0	Not Supported			Not Supported		
PACK = 1 SIGN_EXT = 1	Not Supported			Not Supported		

7.2.8.3.7 20-Bit Word Length

Table 7-9. PACK and Sign Extend Data Arrangement for 20-Bit Word Length

	I2S Receive/Transmit Left or Right Data 1 Register			I2S Receive/Transmit Left or Right Data 0 Register		
PACK = 0 SIGN_EXT = 0	15		0	15	12 11	0
	x(0) Bits[19-4]			x(0) Bits[3-0]		0
PACK = 0 SIGN_EXT = 1	15	4 3	0	15		0
	Sign Bits(Bit #19)	x(0) Bits[19-16]		x(0) Bits[15-0]		
PACK = 1 SIGN_EXT = 0	Not Supported			Not Supported		
PACK = 1 SIGN_EXT = 1	Not Supported			Not Supported		

7.2.8.3.8 24-Bit Word Length

Table 7-10. PACK and Sign Extend Data Arrangement for 24-Bit Word Length

	I2S Receive/Transmit Left or Right Data 1 Register			I2S Receive/Transmit Left or Right Data 0 Register		
PACK = 0 SIGN_EXT = 0	15		0	15	8 7	0
	x(0) Bits[23-8]			x(0) Bits[7-0]		0
PACK = 0 SIGN_EXT = 1	15	8 7	0	15		0
	Sign Bits(Bit #23)	x(0) Bits[23-16]		x(0) Bits[15-0]		
PACK = 1 SIGN_EXT = 0	Not Supported			Not Supported		
PACK = 1 SIGN_EXT = 1	Not Supported			Not Supported		

7.2.8.3.9 32-Bit Word Length

Table 7-11. PACK and Sign Extend Data Arrangement for 32-Bit Word Length

	I2S Receive/Transmit Left or Right Data 1 Register		I2S Receive/Transmit Left or Right Data 0 Register	
PACK = 0 SIGN_EXT = 0	15	0	15	0
	x(0) Bits[32-16]		x(0) Bits[15-0]	
PACK = 0 SIGN_EXT = 1	15	0	15	0
	x(0) Bits[32-16]		x(0) Bits[15-0]	
PACK = 1 SIGN_EXT = 0	Not Supported		Not Supported	
PACK = 1 SIGN_EXT = 1	Not Supported		Not Supported	

7.2.9 Reset Considerations

The I2S bus has two reset sources: software reset and hardware reset.

7.2.9.1 Software Reset Considerations

The I2S bus can be reset by software through the Peripheral Reset Control Register (PRCR). The software reset is very similar to hardware reset with the only exception being that the data in shift registers are not reset. For more details on PRCR, see the device-specific DSP system guide.

7.2.9.2 Hardware Reset Considerations

A hardware reset is always initiated during a full chip reset. When a hardware reset occurs, all the registers of the I2S bus are set to their default values.

7.2.10 Interrupt Support

Every I2S bus supports transmit and receive data-transfer interrupts/events to CPU or DMA and flags two error conditions. These are enabled or disabled by writing to the I2SINTMASK register. There are separate data-transfer interrupt mask bits for stereo or mono operating modes. The interrupts are also flagged in the I2SINTFL.

For Stereo operating mode (MONO=0 in I2SSCTRL) and with the corresponding STEREO interrupts enabled, transmit/receive interrupts are generated after the right channel data has been transferred (since right channel data is always transmitted/received last). In Mono mode (MONO=1 in I2SSCTRL) , only left channel data is transferred, hence the interrupts are generated after the transmit/receive of left channel data. Mono mode can only be used with DSP format (FRMT = 1 in I2SSCRTL).

NOTE: I2S bus behavior is not defined if stereo and mono interrupts are simultaneously enabled. The I2S module flags an OUERROR (if enabled in I2SINTMASK) incorrectly when the module is enabled for the first time. The user program should ignore this error.

7.2.10.1 Interrupt Multiplexing

The interrupts to the CPU from I2S0 are multiplexed with interrupts for McBSP, MMC/SD0, and MMC/SD1. Configuring the Serial Port 0 field in the External Bus Selection register to route I2S signals to the external pins also routes the corresponding I2S interrupts to the CPU. For details, see Chapter 1, *System Control*.

7.2.11 DMA Event Support

Each I2S bus has access to one of the four DMA peripherals on the DSP as shown in Table 7-12. To enable seamless transfers using the DMA, the I2S bus generates data-transfer events to DMA irrespective of the value configured in the I2SINTMASK register. Hence, it is recommended to disable CPU data-transfer interrupts in the I2SINTMASK register when using the DMA for data transfers. For Stereo operating mode (MONO=0 in I2SSCTRL), transmit/receive events are generated after the right channel data has been transferred (since right channel data is always transmitted/received last). In Mono mode (MONO=1 in I2SSCTRL) , only left channel data is transferred, hence the events are generated after the transmit/receive of left channel data. Mono mode can only be used with DSP format (FRMT = 1 in I2SSCRTL). For details on DMA configurations options, see Chapter 1, *System Control*.

Table 7-12. DMA Access to I2S

DMA Engine	I2S Bus
DMA0	I2S0
DMA1	I2S2
DMA2	I2S3

NOTE: The DMA can be used to transfer data samples in single double-word bursts from the I2S to:

- On-chip memory (DARAM/SARAM)
- NAND/NOR memory via EMIF

7.2.12 Power Management

The I2S peripherals can be put into idle condition to save power consumption. This is achieved by gating the system clock to the I2S peripherals via individual fields in the Peripheral Clock Gating Configuration registers described in

7.2.13 Emulation Considerations

An emulation halt will not stop I2S operation and an over-run/under-run error condition will be flagged (if enabled) in the I2SINTFL register if the CPU or DMA is unable to service I2S interrupts/events as a result of the emulation halt.

Refreshing CPU registers or memory contents in Code Composer Studio when the I2S is running could result in the DSP missing real-time operation due to JTAG communication overheads over the internal data buses. This would result in over-run/under-run errors being flagged (if enabled) in the I2SINTFL register.

7.2.14 After Frame Sync

7.2.14.1 Transmit

Once XSR is empty, the internal wakeup signal will be asserted (high), which turns on the clock for McBSP and DMA domains and de-asserts the MBPCLKSTPREQ bit in the CLKSTOP1 register. After McBSP and DMA wake up, the DXR to XSR copy triggers a new data write to DXR by DMA. Complete DMA write to DXR will de-assert the internal wakeup signal, and MBPCLKSTPREQ will go high again. Then, chip level will turn off the clock of McBSP module and DMA module. McBSP get into Idle again.

7.2.14.2 Receive

Once RBR is full, internal wakeup signal will be asserted (High), which will be used to turn on the clock for McBSP and DMA domains and de-assert the MBPCLKSTPREQ. After McBSP and DMA wake up, RBR to DRR copy will trigger a new data read from DRR by DMA. Complete DRR read by DMA will de-assert the internal wakeup signal, and MBPCLKSTPREQ will go high again. Then, chip level will turn off the clock of McBSP module and DMA module. McBSP get into Idle again.

7.2.15 Steps for I2S Configuration and I2S Interrupt Service Routine (ISR)

A sequence of steps for configuring an I2S bus and servicing interrupts in an interrupt service routine (ISR) are given below:

7.2.15.1 Initialization and Configuration Steps

- Bring I2S out of idle.
- If using DMA, reset DMA and MPORT idle bit-fields and run idle instruction to force MPORT out of idle.
- Disable DSP global interrupts.
- Clear DSP interrupt flag registers and enable appropriate I2S/DMA interrupts.
- If using DMA, configure DMA and sync DMA channel(s) to I2S sync event(s).
- Configure external I2S-compatible device.
- Enable DSP global interrupts.
- I2S Configuration:
  - Route I2S signals to external pins.
  - If I2S is the master device, configure the I2SSRATE register.
  - If using CPU interrupts, configure I2SINTMASK register to enable stereo receive/transmit interrupts for stereo mode operation or mono receive/transmit interrupts for mono mode operation (Peripheral behavior is not defined if both stereo and mono interrupts are enabled).
    - If the software/application is designed to respond to the detection of the error condition, enable error interrupts (disregard first OUERROR that is generated).
    - Do not enable stereo/mono TX or RX interrupts if DMA is used for data transfers.
  - Write desired configuration value to I2S*n* Serializer Control Register (I2SSCTRL) (If writing to individual bit fields, enable the I2S by setting the ENABLE bit in I2SSCTRL last.
    - The I2S bus now starts data conversion.
    - If configured as master device, I2S bus will generate interrupts/events even if external I2S-compatible device is not configured correctly and hence not executing data transfers.
    - If configured as a slave device, interrupt/event generation depends on proper operation of the external master device.
- If CPU is not performing other operations, the CPU can now be idled for power savings (if DMA is used for data transfers and software/application requires detection of error conditions, CPU may read the I2S*n* Interrupt Flag Register (I2SINTFL) at regular intervals to check the error flag).
- An I2S (or DMA) interrupt will indicate completion of data transfer(s) and CPU is automatically brought of idle and the interrupt is taken.

7.2.15.2 ISR Steps (for CPU transfers)

Transmit and receive interrupts should have distinct interrupt service routines. A common framework is given below:

- Read the I2SINTFL register to reset the flags and if error detection is required, check for error flags.
- Read or write the I2S*n* Receive/Transmit Left/Right Data *n* Registers for receive and transmit interrupts respectively based on the I2S configuration (Mono, PACK, Sign Extend, Word Length options).
- Return from interrupt.

7.3 I2S0 Registers

Table 7-13 lists the memory-mapped registers for the I2S0. All register offset addresses not listed in Table 7-13 should be considered as reserved locations and the register contents should not be modified.

Table 7-13. I2S0 REGISTERS

CPU Word Address	Acronym	Register Name	Section
2800h	I2S0SCTRL	I2S0 Serializer Control Register	Section 7.3.1
2804h	I2S0SRATE	I2S0 Sample Rate Generator Register	Section 7.3.2
2808h	I2S0TXLT1	I2S0 Transmit Left Data Register 1	Section 7.3.3
2809h	I2S0TXLT2	I2S0 Transmit Left Data Register 2	Section 7.3.4
280Ch	I2S0TXRT1	I2S0 Transmit Right Data Register 1	Section 7.3.5
280Dh	I2S0TXRT2	I2S0 Transmit Right Data Register 2	Section 7.3.6
2810h	I2S0INTFL	I2S0 Interrupt Flag Register	Section 7.3.7
2814h	I2S0INTMASK	I2S0 Interrupt Mask Register	Section 7.3.8
2828h	I2S0RXLT1	I2S0 Receive Left Data Register 1	Section 7.3.9
2829h	I2S0RXLT2	I2S0 Receive Left Data Register 2	Section 7.3.10
282Ch	I2S0RXRT1	I2S0 Receive Right Data Register 1	Section 7.3.11
282Dh	I2S0RXRT2	I2S0 Receive Right Data Register 2	Section 7.3.12

7.3.1 I2S0SCTRL Register (offset = 2800h) [reset = 0h]

I2S0SCTRL is shown in Figure 7-13 and described in Table 7-14.

Figure 7-13. I2S0SCTRL Register

15	14	13	12	11	10	9	8
ENABLE	Reserved		MONO	LOOPBACK	FSPOL	CLKPOL	DATADLY
R/W-0h	R-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

7	6	5	4	3	2	1	0
PACK	SIGN_EXT	WDLNGTH				MODE	FRMT
R/W-0h	R/W-0h	R/W-0h				R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-14. I2S0SCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	ENABLE	R/W	0h	Resets or enables the serializer transmission or reception. 0x0(Read) = The I2S Peripheral (Data transfer, clock generation, and event generate logic) is disabled and held in reset state. 0x1 = I2S enabled
14-13	Reserved	R	0h	
12	MONO	R/W	0h	Sets I2S into mono or Stereo mode. 0x0(Read) = Stereo mode. 0x1 = Mono mode.
11	LOOPBACK	R/W	0h	Routes data from transmit shift register back to receive shift register for internal digital loopback. 0x0(Read) = Normal operation, no loopback. 0x1 = Digital loopback mode enabled.
10	FSPOL	R/W	0h	Inverts I2S frame-synchronization polarity. 0x0(Read) = 2S/LJ: Frame-sync pulse is low for left word and high for right word. DSP: Frame-sync is pulsed high 0x1 = I2S/LJ: Frame-sync pulse is high for left word and low for right word. DSP: Frame-sync is pulsed low
9	CLKPOL	R/W	0h	Controls I2S clock polarity. 0x0(Read) = Receive data is sampled on the rising edge and transmit data shifted on the falling edge of the bit clock. 0x1 = Receive data is sampled on the falling edge and transmit data shifted on the rising edge of the bit clock
8	DATADLY	R/W	0h	Sets the I2S receive/transmit data delay. 0x0(Read) = 1 bit data delay 0x1 = 2 bit data delay
7	PACK	R/W	0h	Divides down the generation of interrupts so that data is packed into the 32-bit receive/transmit word registers for each channel (left/right) 0x0(Read) = Data packing mode disabled 0x1 = Data packing mode enabled
6	SIGN_EXT	R/W	0h	Enable/disable sign extension of words 0x0(Read) = no sign extension 0x1 = received data is sign extended. Transmit data is expected to be sign extended
5-2	WDLNGTH	R/W	0h	Choose serializer word length. 0x0 = 8 bit data word 0x1 = 10 bit data 0x8 = 32 bit data word

Table 7-14. I2S0SCTRL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	MODE	R/W	0h	Set the serializer in master or slave mode. 0x0(Read) = serializer is configured as slave. I2S_BCLK and I2S_FS pins are configured as inputs. The bit-clock and frame-sync signals are derived from an external source and are provided directly to the I2S synchronizer without being further divided. 0x1 = Serializer is configured as master. I2S_BCLK and FS pins are configured as outputs and driven by the clock generators. The bit clock and frame-sync signals are derived from the internal CPU clock.
0	FRMT	R/W	0h	Sets the serializer data format. 0x0(Read) = I2S/left-justified format 0x1 = DSP format

7.3.2 I2S0SRATE Register (offset = 2804h) [reset = 0h]

I2S0SRATE is shown in Figure 7-14 and described in Table 7-15.

Figure 7-14. I2S0SRATE Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		FSDIV			CLKDIV		
R-0h		R/W-0h			R/W-0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-15. I2S0SRATE Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	
5-3	FSDIV	R/W	0h	Divider to generate I2S frame sync clock. The I2S_BCLK is divided down by the configured value to generate the frame sync clock. No effect when configured as slave. 0x0 = DIV8: Divide by 8 0x1 = DIV16: Divide by 16 0x2 = DIV32: Divide by 32 0x3 = DIV64: Divide by 64 0x4 = DIV128: Divide by 128 0x5 = DIV256: Divide by 256 0x6 = RES1: reserved 0x7 = RES2: reserved
2-0	CLKDIV	R/W	0h	Divider to generate I2S bit clock. The system clock to the I2S is divided down by the configured value to generate the bit clock. No effect when configured as slave. 0x0 = DIV2: Divide by 2 0x1 = DIV4: Divide by 4 0x2 = DIV8: Divide by 8 0x3 = DIV16: Divide by 16 0x4 = DIV32: Divide by 32 0x5 = DIV64: Divide by 64 0x6 = DIV128: Divide by 128 0x7 = DIV256: Divide by 256

7.3.3 I2S0TXLT1 Register (offset = 2808h) [reset = 0h]

I2S0TXLT1 is shown in Figure 7-15 and described in Table 7-16.

Figure 7-15. I2S0TXLT1 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-16. I2S0TXLT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Transmit Left Data Lower 16 bits

7.3.4 I2S0TXLT2 Register (offset = 2809h) [reset = 0h]

I2S0TXLT2 is shown in Figure 7-16 and described in Table 7-17.

Figure 7-16. I2S0TXLT2 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-17. I2S0TXLT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Transmit Left Data Upper 16 bits

7.3.5 I2S0TXRT1 Register (offset = 280Ch) [reset = 0h]

I2S0TXRT1 is shown in Figure 7-17 and described in Table 7-18.

Figure 7-17. I2S0TXRT1 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-18. I2S0TXRT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Transmit Right Data Lower 16 bits

7.3.6 I2S0TXRT2 Register (offset = 280Dh) [reset = 0h]

I2S0TXRT2 is shown in Figure 7-18 and described in Table 7-19.

Figure 7-18. I2S0TXRT2 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-19. I2S0TXRT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Transmit Right Data Upper 16 bits

7.3.7 I2S0INTFL Register (offset = 2810h) [reset = 0h]

I2S0INTFL is shown in Figure 7-19 and described in Table 7-20.

Figure 7-19. I2S0INTFL Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		XMITSTFL	XMITMONFL	RCVSTFL	RCVMONFL	FERRFL	OUERRFL
R-0h		R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-20. I2S0INTFL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	
5	XMITSTFL	R	0h	Stereo data transmit flag 0x0(Read) = no pending stereo transmit interrupt 0x1 = stereo transmit interrupt pending. Write new data value to I2S Transmit Left and Right data 0 and 1 registers
4	XMITMONFL	R	0h	Mono Data Transmit flag 0x0(Read) = No pending mono transmit interrupt 0x1 = Mono transmit interrupt pending. Write new data value to Transmit Left Data 0 and 1 registers
3	RCVSTFL	R	0h	Stereo Data Receive flag 0x0(Read) = No pending Stereo receive interrupt 0x1 = stereo receive interrupt pending. Read Receive Left and Right data 0 and 1 registers
2	RCVMONFL	R	0h	Mono Data Receive flag 0x0(Read) = No pending mono receive interrupt 0x1 = Mono receive interrupt pending. Read Receive Left data 0 and 1 registers
1	FERRFL	R	0h	Frame Sync Error flag 0x0(Read) = No Frame-sync errors 0x1 = Frame-sync error occurred
0	OUERRFL	R	0h	Overrun or Underrun condition 0x0(Read) = No overrun/underrun errors 0x1 = The data registers were not read from or written to before the receive/transmit buffer was overwritten.

7.3.8 I2S0INTMASK Register (offset = 2814h) [reset = 0h]

I2S0INTMASK is shown in Figure 7-20 and described in Table 7-21.

Figure 7-20. I2S0INTMASK Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		XMITST	XMITMON	RCVST	RCVMON	FERR	OUERR
R-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-21. I2S0INTMASK Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	
5	XMITST	R/W	0h	Enable Stereo Left/right Transmit Data Interrupt 0x0 = Disable stereo TX data interrupt 0x1 = Enable stereo TX data interrupt
4	XMITMON	R/W	0h	Enable Mono Left Transmit Data Interrupt 0x0 = Disable mono TX data interrupt 0x1 = Enable mono TX data Interrupt
3	RCVST	R/W	0h	Enable Stereo Left/Right Receive Data Interrupt 0x0 = Disable stereo RX data interrupt 0x1 = Enable stereo RX data interrupt
2	RCVMON	R/W	0h	Enable Mono Left Receive Data Interrupt 0x0 = Disable mono RX data interrupt 0x1 = Enable mono RX data Interrupt
1	FERR	R/W	0h	Enable Frame Sync error 0x0 = Disable frame-Sync error interrupt 0x1 = Enable frame-Sync error interrupt
0	OUERR	R/W	0h	Enable Overrun or Underrun condition 0x0 = Disable overrun or underrun error interrupt 0x1 = Enable overrun or underrun error interrupt

7.3.9 I2S0RXLT1 Register (offset = 2828h) [reset = 0h]

I2S0RXLT1 is shown in Figure 7-21 and described in Table 7-22.

Figure 7-21. I2S0RXLT1 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-22. I2S0RXLT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Receive Left Data Lower 16 bits

7.3.10 I2S0RXLT2 Register (offset = 2829h) [reset = 0h]

I2S0RXLT2 is shown in Figure 7-22 and described in Table 7-23.

Figure 7-22. I2S0RXLT2 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-23. I2S0RXLT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Receive Left Data Upper 16 bits

7.3.11 I2S0RXRT1 Register (offset = 282Ch) [reset = 0h]

I2S0RXRT1 is shown in Figure 7-23 and described in Table 7-24.

Figure 7-23. I2S0RXRT1 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-24. I2S0RXRT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Receive Right Data Lower 16 bits

7.3.12 I2S0RXRT2 Register (offset = 282Dh) [reset = 0h]

I2S0RXRT2 is shown in Figure 7-24 and described in Table 7-25.

Figure 7-24. I2S0RXRT2 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-25. I2S0RXRT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Receive Right Data Upper 16 bits

7.4 I2S2 Registers

Table 7-26 lists the memory-mapped registers for the I2S2. All register offset addresses not listed in Table 7-26 should be considered as reserved locations and the register contents should not be modified.

Table 7-26. I2S2 REGISTERS

CPU Word Address	Acronym	Register Name	Section
2A00h	I2S2SCTRL	I2S2 Serializer Control Register	Section 7.4.1
2A04h	I2S2SRATE	I2S2 Sample Rate Generator Register	Section 7.4.2
2A08h	I2S2TXLT1	I2S2 Transmit Left Data Register 1	Section 7.4.3
2A09h	I2S2TXLT2	I2S2 Transmit Left Data Register 2	Section 7.4.4
2A0Ch	I2S2TXRT1	I2S2 Transmit Right Data Register 1	Section 7.4.5
2A0Dh	I2S2TXRT2	I2S2 Transmit Right Data Register 2	Section 7.4.6
2A10h	I2S2INTFL	I2S2 Interrupt Flag Register	Section 7.4.7
2A14h	I2S2INTMASK	I2S2 Interrupt Mask Register	Section 7.4.8
2A28h	I2S2RXLT1	I2S2 Receive Left Data Register 1	Section 7.4.9
2A29h	I2S2RXLT2	I2S2 Receive Left Data Register 2	Section 7.4.10
2A2Ch	I2S2RXRT1	I2S2 Receive Right Data Register 1	Section 7.4.11
2A2Dh	I2S2RXRT2	I2S2 Receive Right Data Register 2	Section 7.4.12

7.4.1 I2S2SCTRL Register (offset = 2A00h) [reset = 0h]

I2S2SCTRL is shown in Figure 7-25 and described in Table 7-27.

Figure 7-25. I2S2SCTRL Register

15	14	13	12	11	10	9	8
ENABLE	Reserved		MONO	LOOPBACK	FSPOL	CLKPOL	DATADLY
R/W-0h	R-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
PACK	SIGN_EXT	WDLNGTH				MODE	FRMT
R/W-0h	R/W-0h	R/W-0h				R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-27. I2S2SCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	ENABLE	R/W	0h	Resets or enables the serializer transmission or reception. 0x0(Read) = The I2S Peripheral (Data transfer, clock generation, and event generate logic) is disabled and held in reset state. 0x1 = I2S enabled
14-13	Reserved	R	0h	
12	MONO	R/W	0h	Sets I2S into mono or Stereo mode. 0x0(Read) = Stereo mode. 0x1 = Mono mode.
11	LOOPBACK	R/W	0h	Routes data from transmit shift register back to receive shift register for internal digital loopback. 0x0(Read) = Normal operation, no loopback. 0x1 = Digital loopback mode enabled.
10	FSPOL	R/W	0h	Inverts I2S frame-synchronization polarity. 0x0(Read) = 2S/LJ: Frame-sync pulse is low for left word and high for right word. DSP: Frame-sync is pulsed high 0x1 = I2S/LJ: Frame-sync pulse is high for left word and low for right word. DSP: Frame-sync is pulsed low
9	CLKPOL	R/W	0h	Controls I2S clock polarity. 0x0(Read) = Receive data is sampled on the rising edge and transmit data shifted on the falling edge of the bit clock. 0x1 = Receive data is sampled on the falling edge and transmit data shifted on the rising edge of the bit clock
8	DATADLY	R/W	0h	Sets the I2S receive/transmit data delay. 0x0(Read) = 1 bit data delay 0x1 = 2 bit data delay
7	PACK	R/W	0h	Divides down the generation of interrupts so that data is packed into the 32-bit receive/transmit word registers for each channel (left/right) 0x0(Read) = Data packing mode disabled 0x1 = Data packing mode enabled
6	SIGN_EXT	R/W	0h	Enable/disable sign extension of words 0x0(Read) = no sign extension 0x1 = received data is sign extended. Transmit data is expected to be sign extended
5-2	WDLNGTH	R/W	0h	Choose serializer word length. 0x0 = 8 bit data word 0x1 = 10 bit data 0x8 = 32 bit data word

Table 7-27. I2S2SCTRL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	MODE	R/W	0h	Set the serializer in master or slave mode. 0x0(Read) = serializer is configured as slave. I2S_BCLK and I2S_FS pins are configured as inputs. The bit-clock and frame-sync signals are derived from an external source and are provided directly to the I2S synchronizer without being further divided. 0x1 = Serializer is configured as master. I2S_BCLK and FS pins are configured as outputs and driven by the clock generators. The bit clock and frame-sync signals are derived from the internal CPU clock.
0	FRMT	R/W	0h	Sets the serializer data format. 0x0(Read) = I2S/left-justified format 0x1 = DSP format

7.4.2 I2S2SRATE Register (offset = 2A04h) [reset = 0h]

I2S2SRATE is shown in Figure 7-26 and described in Table 7-28.

Figure 7-26. I2S2SRATE Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		FSDIV			CLKDIV		
R-0h		R/W-0h			R/W-0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-28. I2S2SRATE Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	
5-3	FSDIV	R/W	0h	Divider to generate I2S frame sync clock. The I2S_BCLK is divided down by the configured value to generate the frame sync clock. No effect when configured as slave. 0x0 = DIV8: Divide by 8 0x1 = DIV16: Divide by 16 0x2 = DIV32: Divide by 32 0x3 = DIV64: Divide by 64 0x4 = DIV128: Divide by 128 0x5 = DIV256: Divide by 256 0x6 = RES1: reserved 0x7 = RES2: reserved
2-0	CLKDIV	R/W	0h	Divider to generate I2S bit clock. The system clock to the I2S is divided down by the configured value to generate the bit clock. No effect when configured as slave. 0x0 = DIV2: Divide by 2 0x1 = DIV4: Divide by 4 0x2 = DIV8: Divide by 8 0x3 = DIV16: Divide by 16 0x4 = DIV32: Divide by 32 0x5 = DIV64: Divide by 64 0x6 = DIV128: Divide by 128 0x7 = DIV256: Divide by 256

7.4.3 I2S2TXLT1 Register (offset = 2A08h) [reset = 0h]

I2S2TXLT1 is shown in Figure 7-27 and described in Table 7-29.

Figure 7-27. I2S2TXLT1 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-29. I2S2TXLT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Transmit Left Data Lower 16 bits

7.4.4 I2S2TXLT2 Register (offset = 2A09h) [reset = 0h]

I2S2TXLT2 is shown in Figure 7-28 and described in Table 7-30.

Figure 7-28. I2S2TXLT2 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-30. I2S2TXLT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Transmit Left Data Upper 16 bits

7.4.5 I2S2TXRT1 Register (offset = 2A0Ch) [reset = 0h]

I2S2TXRT1 is shown in Figure 7-29 and described in Table 7-31.

Figure 7-29. I2S2TXRT1 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-31. I2S2TXRT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Transmit Right Data Lower 16 bits

7.4.6 I2S2TXRT2 Register (offset = 2A0Dh) [reset = 0h]

I2S2TXRT2 is shown in Figure 7-30 and described in Table 7-32.

Figure 7-30. I2S2TXRT2 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-32. I2S2TXRT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Transmit Right Data Upper 16 bits

7.4.7 I2S2INTFL Register (offset = 2A10h) [reset = 0h]

I2S2INTFL is shown in Figure 7-31 and described in Table 7-33.

Figure 7-31. I2S2INTFL Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		XMITSTFL	XMITMONFL	RCVSTFL	RCVMONFL	FERRFL	OUERRFL
R-0h		R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-33. I2S2INTFL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	
5	XMITSTFL	R	0h	Stereo data transmit flag 0x0(Read) = no pending stereo transmit interrupt 0x1 = stereo transmit interrupt pending. Write new data value to I2S Transmit Left and Right data 0 and 1 registers
4	XMITMONFL	R	0h	Mono Data Transmit flag 0x0(Read) = No pending mono transmit interrupt 0x1 = Mono transmit interrupt pending. Write new data value to Transmit Left Data 0 and 1 registers
3	RCVSTFL	R	0h	Stereo Data Receive flag 0x0(Read) = No pending Stereo receive interrupt 0x1 = stereo receive interrupt pending. Read Receive Left and Right data 0 and 1 registers
2	RCVMONFL	R	0h	Mono Data Receive flag 0x0(Read) = No pending mono receive interrupt 0x1 = Mono receive interrupt pending. Read Receive Left data 0 and 1 registers
1	FERRFL	R	0h	Frame Sync Error flag 0x0(Read) = No Frame-sync errors 0x1 = Frame-sync error occurred
0	OUERRFL	R	0h	Overrun or Underrun condition 0x0(Read) = No overrun/underrun errors 0x1 = The data registers were not read from or written to before the receive/transmit buffer was overwritten.

7.4.8 I2S2INTMASK Register (offset = 2A14h) [reset = 0h]

I2S2INTMASK is shown in Figure 7-32 and described in Table 7-34.

Figure 7-32. I2S2INTMASK Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		XMITST	XMITMON	RCVST	RCVMON	FERR	OUERR
R-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-34. I2S2INTMASK Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	
5	XMITST	R/W	0h	Enable Stereo Left/right Transmit Data Interrupt 0x0 = Disable stereo TX data interrupt 0x1 = Enable stereo TX data interrupt
4	XMITMON	R/W	0h	Enable Mono Left Transmit Data Interrupt 0x0 = Disable mono TX data interrupt 0x1 = Enable mono TX data Interrupt
3	RCVST	R/W	0h	Enable Stereo Left/Right Receive Data Interrupt 0x0 = Disable stereo RX data interrupt 0x1 = Enable stereo RX data interrupt
2	RCVMON	R/W	0h	Enable Mono Left Receive Data Interrupt 0x0 = Disable mono RX data interrupt 0x1 = Enable mono RX data Interrupt
1	FERR	R/W	0h	Enable Frame Sync error 0x0 = Disable frame-Sync error interrupt 0x1 = Enable frame-Sync error interrupt
0	OUERR	R/W	0h	Enable Overrun or Underrun condition 0x0 = Disable overrun or underrun error interrupt 0x1 = Enable overrun or underrun error interrupt

7.4.9 I2S2RXLT1 Register (offset = 2A28h) [reset = 0h]

I2S2RXLT1 is shown in Figure 7-33 and described in Table 7-35.

Figure 7-33. I2S2RXLT1 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-35. I2S2RXLT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Receive Left Data Lower 16 bits

7.4.10 I2S2RXLT2 Register (offset = 2A29h) [reset = 0h]

I2S2RXLT2 is shown in Figure 7-34 and described in Table 7-36.

Figure 7-34. I2S2RXLT2 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-36. I2S2RXLT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Receive Left Data Upper 16 bits

7.4.11 I2S2RXRT1 Register (offset = 2A2Ch) [reset = 0h]

I2S2RXRT1 is shown in Figure 7-35 and described in Table 7-37.

Figure 7-35. I2S2RXRT1 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-37. I2S2RXRT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Receive Right Data Lower 16 bits

7.4.12 I2S2RXRT2 Register (offset = 2A2Dh) [reset = 0h]

I2S2RXRT2 is shown in Figure 7-36 and described in Table 7-38.

Figure 7-36. I2S2RXRT2 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-38. I2S2RXRT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Receive Right Data Upper 16 bits

7.5 I2S3 Registers

Table 7-39 lists the memory-mapped registers for the I2S3. All register offset addresses not listed in Table 7-39 should be considered as reserved locations and the register contents should not be modified.

Table 7-39. I2S3 REGISTERS

CPU Word Address	Acronym	Register Name	Section
2B00h	I2S3SCTRL	I2S3 Serializer Control Register	Section 7.5.1
2B04h	I2S3SRATE	I2S3 Sample Rate Generator Register	Section 7.5.2
2B08h	I2S3TXLT1	I2S3 Transmit Left Data Register 1	Section 7.5.3
2B09h	I2S3TXLT2	I2S3 Transmit Left Data Register 2	Section 7.5.4
2B0Ch	I2S3TXRT1	I2S3 Transmit Right Data Register 1	Section 7.5.5
2B0Dh	I2S3TXRT2	I2S3 Transmit Right Data Register 2	Section 7.5.6
2B10h	I2S3INTFL	I2S3 Interrupt Flag Register	Section 7.5.7
2B14h	I2S3INTMASK	I2S3 Interrupt Mask Register	Section 7.5.8
2B28h	I2S3RXLT1	I2S3 Receive Left Data Register 1	Section 7.5.9
2B29h	I2S3RXLT2	I2S3 Receive Left Data Register 2	Section 7.5.10
2B2Ch	I2S3RXRT1	I2S3 Receive Right Data Register 1	Section 7.5.11
2B2Dh	I2S3RXRT2	I2S3 Receive Right Data Register 2	Section 7.5.12

7.5.1 I2S3SCTRL Register (offset = 2B00h) [reset = 0h]

I2S3SCTRL is shown in Figure 7-37 and described in Table 7-40.

Figure 7-37. I2S3SCTRL Register

15	14	13	12	11	10	9	8
ENABLE	Reserved		MONO	LOOPBACK	FSPOL	CLKPOL	DATADLY
R/W-0h	R-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
PACK	SIGN_EXT	WDLNGTH				MODE	FRMT
R/W-0h	R/W-0h	R/W-0h				R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-40. I2S3SCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	ENABLE	R/W	0h	Resets or enables the serializer transmission or reception. 0x0(Read) = The I2S Peripheral (Data transfer, clock generation, and event generate logic) is disabled and held in reset state. 0x1 = I2S enabled
14-13	Reserved	R	0h	
12	MONO	R/W	0h	Sets I2S into mono or Stereo mode. 0x0(Read) = Stereo mode. 0x1 = Mono mode.
11	LOOPBACK	R/W	0h	Routes data from transmit shift register back to receive shift register for internal digital loopback. 0x0(Read) = Normal operation, no loopback. 0x1 = Digital loopback mode enabled.
10	FSPOL	R/W	0h	Inverts I2S frame-synchronization polarity. 0x0(Read) = 2S/LJ: Frame-sync pulse is low for left word and high for right word. DSP: Frame-sync is pulsed high 0x1 = I2S/LJ: Frame-sync pulse is high for left word and low for right word. DSP: Frame-sync is pulsed low
9	CLKPOL	R/W	0h	Controls I2S clock polarity. 0x0(Read) = Receive data is sampled on the rising edge and transmit data shifted on the falling edge of the bit clock. 0x1 = Receive data is sampled on the falling edge and transmit data shifted on the rising edge of the bit clock
8	DATADLY	R/W	0h	Sets the I2S receive/transmit data delay. 0x0(Read) = 1 bit data delay 0x1 = 2 bit data delay
7	PACK	R/W	0h	Divides down the generation of interrupts so that data is packed into the 32-bit receive/transmit word registers for each channel (left/right) 0x0(Read) = Data packing mode disabled 0x1 = Data packing mode enabled
6	SIGN_EXT	R/W	0h	Enable/disable sign extension of words 0x0(Read) = no sign extension 0x1 = received data is sign extended. Transmit data is expected to be sign extended
5-2	WDLNGTH	R/W	0h	Choose serializer word length. 0x0 = 8 bit data word 0x1 = 10 bit data 0x8 = 32 bit data word

Table 7-40. I2S3SCTRL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	MODE	R/W	0h	Set the serializer in master or slave mode. 0x0(Read) = serializer is configured as slave. I2S_BCLK and I2S_FS pins are configured as inputs. The bit-clock and frame-sync signals are derived from an external source and are provided directly to the I2S synchronizer without being further divided. 0x1 = Serializer is configured as master. I2S_BCLK and FS pins are configured as outputs and driven by the clock generators. The bit clock and frame-sync signals are derived from the internal CPU clock.
0	FRMT	R/W	0h	Sets the serializer data format. 0x0(Read) = I2S/left-justified format 0x1 = DSP format

7.5.2 I2S3SRATE Register (offset = 2B04h) [reset = 0h]

I2S3SRATE is shown in Figure 7-38 and described in Table 7-41.

Figure 7-38. I2S3SRATE Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		FSDIV			CLKDIV		
R-0h		R/W-0h			R/W-0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-41. I2S3SRATE Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	
5-3	FSDIV	R/W	0h	Divider to generate I2S frame sync clock. The I2S_BCLK is divided down by the configured value to generate the frame sync clock. No effect when configured as slave. 0x0 = DIV8: Divide by 8 0x1 = DIV16: Divide by 16 0x2 = DIV32: Divide by 32 0x3 = DIV64: Divide by 64 0x4 = DIV128: Divide by 128 0x5 = DIV256: Divide by 256 0x6 = RES1: reserved 0x7 = RES2: reserved
2-0	CLKDIV	R/W	0h	Divider to generate I2S bit clock. The system clock to the I2S is divided down by the configured value to generate the bit clock. No effect when configured as slave. 0x0 = DIV2: Divide by 2 0x1 = DIV4: Divide by 4 0x2 = DIV8: Divide by 8 0x3 = DIV16: Divide by 16 0x4 = DIV32: Divide by 32 0x5 = DIV64: Divide by 64 0x6 = DIV128: Divide by 128 0x7 = DIV256: Divide by 256

7.5.3 I2S3TXLT1 Register (offset = 2B08h) [reset = 0h]

I2S3TXLT1 is shown in Figure 7-39 and described in Table 7-42.

Figure 7-39. I2S3TXLT1 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-42. I2S3TXLT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Transmit Left Data Lower 16 bits

7.5.4 I2S3TXLT2 Register (offset = 2B09h) [reset = 0h]

I2S3TXLT2 is shown in Figure 7-40 and described in Table 7-43.

Figure 7-40. I2S3TXLT2 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-43. I2S3TXLT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Transmit Left Data Upper 16 bits

7.5.5 I2S3TXRT1 Register (offset = 2B0Ch) [reset = 0h]

I2S3TXRT1 is shown in Figure 7-41 and described in Table 7-44.

Figure 7-41. I2S3TXRT1 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-44. I2S3TXRT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Transmit Right Data Lower 16 bits

7.5.6 I2S3TXRT2 Register (offset = 2B0Dh) [reset = 0h]

I2S3TXRT2 is shown in Figure 7-42 and described in Table 7-45.

Figure 7-42. I2S3TXRT2 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-45. I2S3TXRT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Transmit Right Data Upper 16 bits

7.5.7 I2S3INTFL Register (offset = 2B10h) [reset = 0h]

I2S3INTFL is shown in Figure 7-43 and described in Table 7-46.

Figure 7-43. I2S3INTFL Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		XMITSTFL	XMITMONFL	RCVSTFL	RCVMONFL	FERRFL	OUERRFL
R-0h		R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-46. I2S3INTFL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	
5	XMITSTFL	R	0h	Stereo data transmit flag 0x0(Read) = no pending stereo transmit interrupt 0x1 = stereo transmit interrupt pending. Write new data value to I2S Transmit Left and Right data 0 and 1 registers
4	XMITMONFL	R	0h	Mono Data Transmit flag 0x0(Read) = No pending mono transmit interrupt 0x1 = Mono transmit interrupt pending. Write new data value to Transmit Left Data 0 and 1 registers
3	RCVSTFL	R	0h	Stereo Data Receive flag 0x0(Read) = No pending Stereo receive interrupt 0x1 = stereo receive interrupt pending. Read Receive Left and Right data 0 and 1 registers
2	RCVMONFL	R	0h	Mono Data Receive flag 0x0(Read) = No pending mono receive interrupt 0x1 = Mono receive interrupt pending. Read Receive Left data 0 and 1 registers
1	FERRFL	R	0h	Frame Sync Error flag 0x0(Read) = No Frame-sync errors 0x1 = Frame-sync error occurred
0	OUERRFL	R	0h	Overrun or Underrun condition 0x0(Read) = No overrun/underrun errors 0x1 = The data registers were not read from or written to before the receive/transmit buffer was overwritten.

7.5.8 I2S3INTMASK Register (offset = 2B14h) [reset = 0h]

I2S3INTMASK is shown in Figure 7-44 and described in Table 7-47.

Figure 7-44. I2S3INTMASK Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		XMITST	XMITMON	RCVST	RCVMON	FERR	OUERR
R-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-47. I2S3INTMASK Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	
5	XMITST	R/W	0h	Enable Stereo Left/right Transmit Data Interrupt 0x0 = Disable stereo TX data interrupt 0x1 = Enable stereo TX data interrupt
4	XMITMON	R/W	0h	Enable Mono Left Transmit Data Interrupt 0x0 = Disable mono TX data interrupt 0x1 = Enable mono TX data Interrupt
3	RCVST	R/W	0h	Enable Stereo Left/Right Receive Data Interrupt 0x0 = Disable stereo RX data interrupt 0x1 = Enable stereo RX data interrupt
2	RCVMON	R/W	0h	Enable Mono Left Receive Data Interrupt 0x0 = Disable mono RX data interrupt 0x1 = Enable mono RX data Interrupt
1	FERR	R/W	0h	Enable Frame Sync error 0x0 = Disable frame-Sync error interrupt 0x1 = Enable frame-Sync error interrupt
0	OUERR	R/W	0h	Enable Overrun or Underrun condition 0x0 = Disable overrun or underrun error interrupt 0x1 = Enable overrun or underrun error interrupt

7.5.9 I2S3RXLT1 Register (offset = 2B28h) [reset = 0h]

I2S3RXLT1 is shown in Figure 7-45 and described in Table 7-48.

Figure 7-45. I2S3RXLT1 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-48. I2S3RXLT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Receive Left Data Lower 16 bits

7.5.10 I2S3RXLT2 Register (offset = 2B29h) [reset = 0h]

I2S3RXLT2 is shown in Figure 7-46 and described in Table 7-49.

Figure 7-46. I2S3RXLT2 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-49. I2S3RXLT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Receive Left Data Upper 16 bits

7.5.11 I2S3RXRT1 Register (offset = 2B2Ch) [reset = 0h]

I2S3RXRT1 is shown in Figure 7-47 and described in Table 7-50.

Figure 7-47. I2S3RXRT1 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-50. I2S3RXRT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Receive Right Data Lower 16 bits

7.5.12 I2S3RXRT2 Register (offset = 2B2Dh) [reset = 0h]

I2S3RXRT2 is shown in Figure 7-48 and described in Table 7-51.

Figure 7-48. I2S3RXRT2 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 7-51. I2S3RXRT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	R/W	0h	Receive Right Data Upper 16 bits

Multichannel Buffered Serial Port (McBSP) Interface

Topic **Page**

8.1 Introduction

This document describes the operation of the multichannel buffered serial port (McBSP) interface. The McBSP consists of a data path and a control path that connect to external devices. Separate pins for transmission and reception communicate data to these external devices.

8.1.1 Purpose of the Peripheral

The primary use for the multichannel buffered serial port (McBSP) is for audio interface purposes. The McBSP is a specialized version of the McBSP peripheral used on other TI DSPs. The primary audio modes that are supported are the AC97 and IIS modes. In addition to the primary audio modes, the McBSP can be programmed to support other serial formats but is not intended to be used as a high-speed interface.

8.1.2 Features

The McBSP provides the following functions:

- Full-duplex communication
- Double-buffered data registers, which allow a continuous data stream
- Independent framing and clocking for receive and transmit
- Direct interface to industry-standard codecs, analog interface chips (AICs), and other serially connected analog-to-digital (A/D) and digital-to-analog (D/A) devices
- External shift clock or an internal, programmable frequency shift clock for data transfer

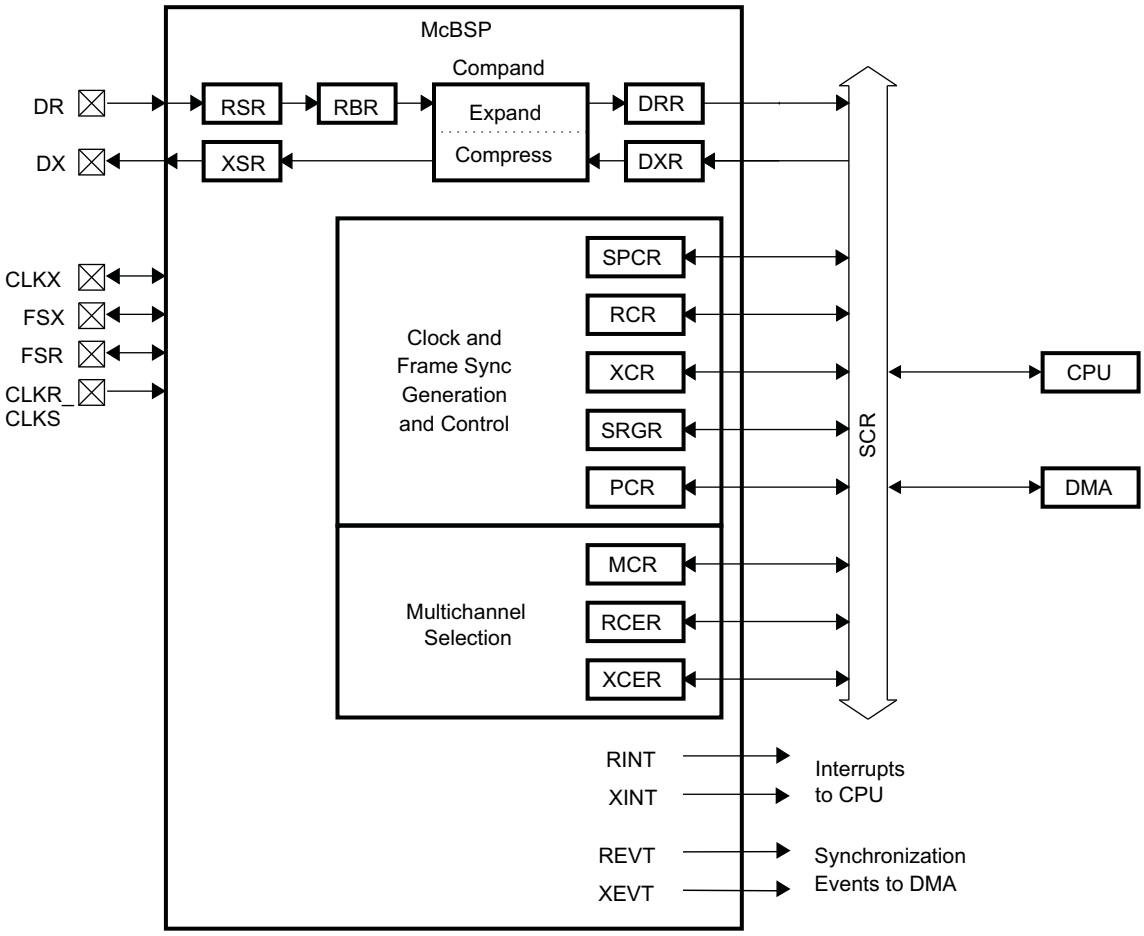
In addition, the McBSP has the following capabilities:

- Direct interface to:
  - T1/E1 framers
  - MVIP switching compatible and ST-BUS compliant devices including:
    - MVIP framers
    - H.100 framers
    - SCSA framers
  - IOM-2 compliant devices
  - AC97 compliant devices (the necessary multiphase frame synchronization capability is provided)
  - IIS compliant devices
- Multichannel transmit and receive of up to 128 channels
- A wide selection of data sizes, including 8, 12, 16, 20, 24, and 32 bits
- μ-Law and A-Law companding
- 8-bit data transfers with the option of LSB or MSB first
- Programmable polarity for both frame synchronization and data clocks
- Highly programmable internal clock and frame generation

8.1.3 Functional Block Diagram

The McBSP consists of a data path and control path, as shown in Figure 8-1

Figure 8-1. McBSP Block Diagram



8.1.4 Industry Standard Compliance Statement

The McBSP supports the following industry standard interfaces:

AC97— The AC97 standard specifies a 5-wire digital serial link between an audio codec device and its digital controller.

IIS— IIS is a protocol for transmitting two channels of digital audio data over a single serial connection. The IIS bus is an industry standard three-wire interface for streaming stereo audio between devices, typically between a CPU/DSP and a DAC/ADC.

8.2 Architecture

This section describes the architecture of the McBSP.

8.2.1 Clock Control

The McBSP can use an internal or external clock source. Either clock source can be divided-down inside the McBSP to generate the actual interface bit clock frequency. Detailed information about how the interface clock and frame synchronization signals are generated is provided in Section 8.2.5.

8.2.2 Signal Descriptions

The signals used on the McBSP interface are listed in Table 8-1.

Table 8-1. McBSP Interface Signals

Pin	I/O/Z	Description
CLKR_CLKS	I/O/Z	Receive clock - supplies or receives a reference clock for the receiver; or supplies a reference clock to the sample rate generator (Bit 15=0 of the EBSR register (1C00h) determines this pin to be CLKR)
CLKR_CLKS	I	Supplies the input clock of the sample rate generator (Bit 15=1 of the EBSR register determines this pin to be CLKS)
CLKX	I/O/Z	Transmit clock - supplies or receives a reference clock for the transmitter; or supplies a reference clock to the sample rate generator
DR	I	Received serial data
DX	O/Z	Transmitted serial data
FSR	I/O/Z	Receive frame synchronization - control signal to synchronize the start of received data
FSX	I/O/Z	Transmit frame synchronization - control signal to synchronize the start of transmitted data

8.2.3 Pin Multiplexing

Extensive pin multiplexing is used to accommodate the largest number of peripheral functions in the smallest possible package. Pin multiplexing is controlled using a combination of hardware configuration at device reset and software programmable register settings. In order to use the McBSP pins, in the EBSR register (1C00h), set to 3 the bits in Serial Port 0 Mode Control (SP0MODE).

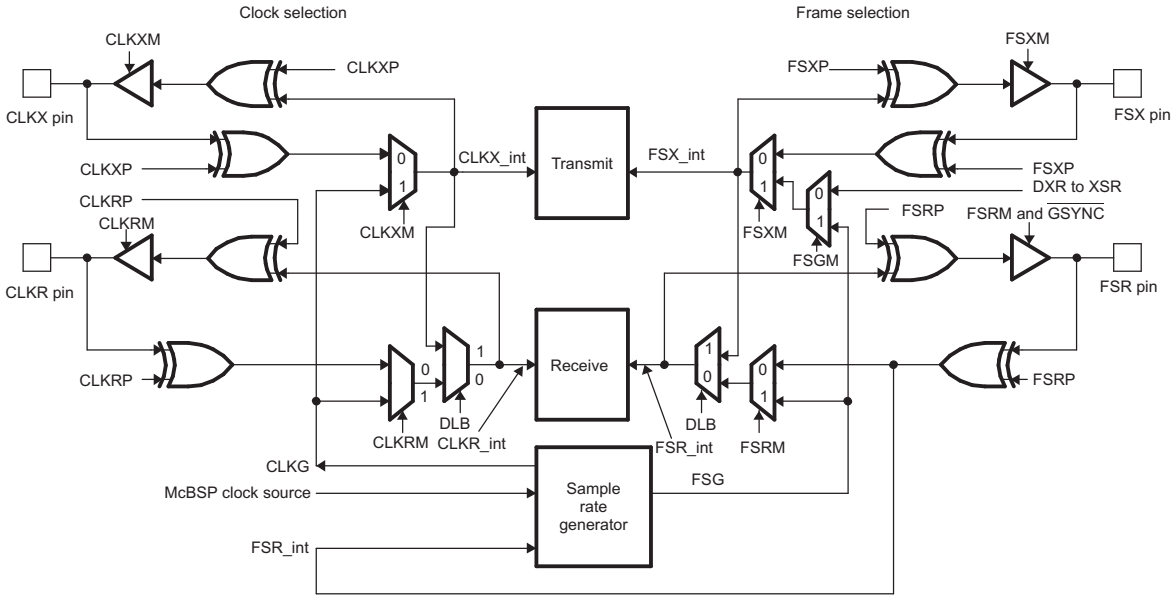
8.2.4 Endianness Considerations

There are no endianness considerations for the McBSP.

8.2.5 Clock, Frames, and Data

The McBSP has several ways of selecting clocking and framing for both the receiver and transmitter. Clocking and framing can be sent to both portions by the sample rate generator. Each portion can select external clocking and/or framing independently. Figure 8-2 is a block diagram of the clock and frame selection circuitry.

Figure 8-2. Clock and Frame Generation



For the McBSP clock source, see Figure 8-5.

8.2.5.1 Frame and Clock Operation

Receive and transmit frame sync pulses (FSR/X), and clocks (CLKR/X), can either be generated internally by the sample rate generator (see Section 8.2.5.2) or be driven by an external source. The source of frame sync and clock is selected by programming the mode bits, FS(R/X)M and CLK(R/X)M respectively, in the pin control register (PCR). FSR is also affected by the GSYNC bit in the sample rate generator register (SRGR), see Section 8.2.5.4.2 for details.

When FSR and FSX are inputs (FSXM = FSRM = 0), the McBSP detects them on the internal falling edge of clock, CLKR_int and CLKX_int, respectively (see Figure 8-2). The receive data arriving at the DR pin is also sampled on the falling edge of CLKR_int. These internal clock signals are either derived from an external source via the CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of the internal clock, CLK(R/X)_int. Similarly, data on DX is output on the rising edge of CLKX_int.

The FSRP, FSXP, CLKRP, and CLKXP bits in PCR configure the polarities of FSR, FSX, CLKR, and CLKX. All frame sync signals (FSR_int and FSX_int) internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to the McBSP) and FSRP = FSXP = 1, the external active (low) frame sync signals are inverted before being sent to the receiver signal (FSR_int) and transmitter signal (FSX_int). Similarly, if internal synchronization is selected (FSR/FSX are outputs and GSYNC = 0), the internal active (high) sync signals are inverted if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.Figure 8-2 shows this inversion using XOR gates.

On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of CLKX_int (see Figure 8-3). If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge-triggered input clock on CLKX is inverted to a rising-edge-triggered clock before being sent to the transmitter. If CLKXP = 1 and internal clocking is selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge-triggered) clock, CLKX_int, is inverted before being sent out on the CLKX pin.

Similarly, the receiver can reliably sample data that is clocked (by the transmitter) with a rising-edge clock. The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of CLKR_int (see Figure 8-4). Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and CLKR is an input pin), the external rising-edge triggered input clock on CLKR is inverted to a falling-edge clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling-edge-triggered clock is inverted to a rising edge before being sent out on the CLKR pin.

In a system where the same clock (internal or external) is used to clock the receiver and transmitter, CLKRP = CLKXP. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold times of data around this edge. Figure 8-4 shows how data clocked by an external serial device using a rising-edge clock can be sampled by the McBSP receiver with the falling edge of the same clock.

Figure 8-3. Transmit Data Clocking

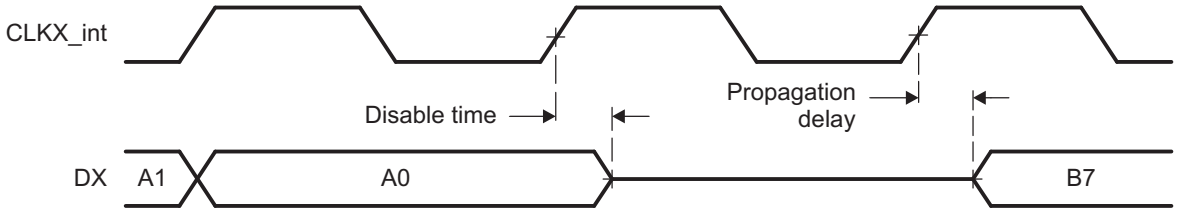
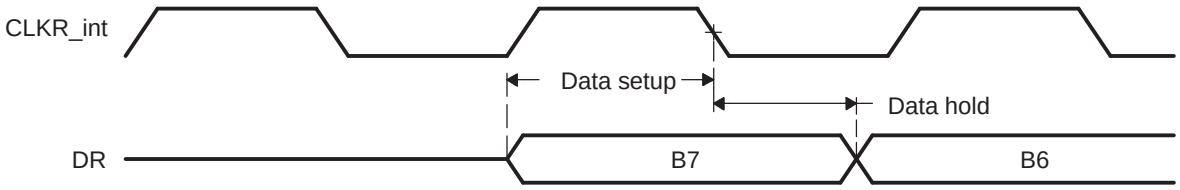


Figure 8-4. Receive Data Clocking



8.2.5.2 Sample Rate Generator Clocking and Framing

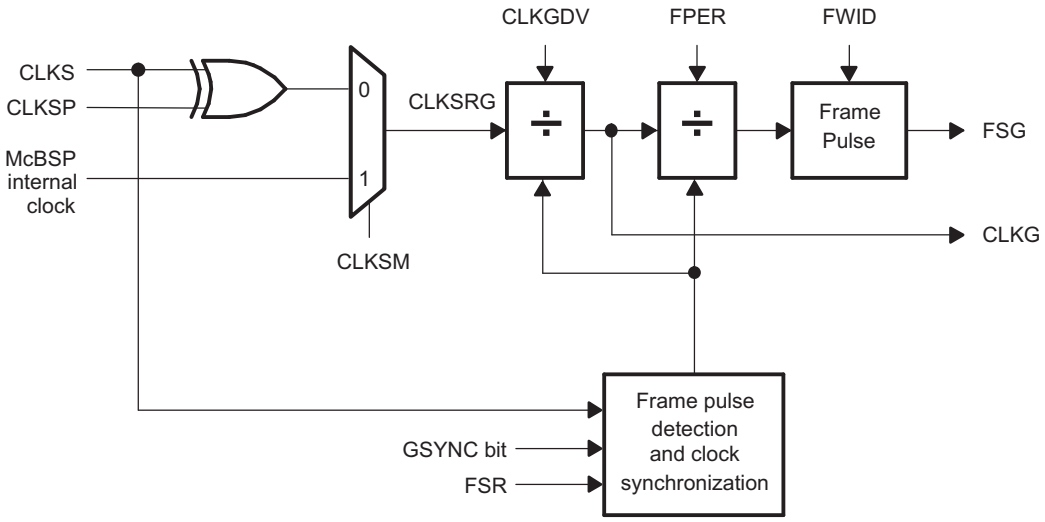
The sample rate generator is composed of a 3-stage clock divider that provides a programmable data clock (CLKG) and framing signal (FSG), as shown in Figure 8-5. CLKG and FSG are McBSP internal signals that can be programmed to drive receive and/or transmit clocking, CLK(R/X), and framing, FS(R/X). The sample rate generator can be programmed to be driven by an internal clock source or an internal clock derived from an external clock source.

The sample rate generator is not used when CLKX, FSX, CLKR, and FSR are driven by an external source. Therefore, the GRST bit in the serial port control register (SPCR) does not need to be enabled (GRST = 1) for this setup. The three stages of the sample rate generator circuit compute:

- Clock divide-down (CLKGDV): The number of input clocks per data bit clock
- Frame period (FPER): The frame period in data bit clocks
- Frame width (FWID): The width of an active frame pulse in data bit clocks

In addition, a frame pulse detection and clock synchronization module allows synchronization of the clock divide-down with an incoming frame pulse. The operation of the sample rate generator during device reset is described in Section 8.2.10.

Figure 8-5. Sample Rate Generator Block Diagram



On this DSP, the McBSP internal input clock frequency is the same as the CPU frequency.

8.2.5.3 Data Clock Generation

When the receive/transmit clock mode is set to 1 (CLK(R/X)M = 1 in the pin control register (PCR)), the data clocks (CLK(R/X)) are driven by the internal sample rate generator output clock, CLKG. You can select for the receiver and transmitter from a variety of data bit clocks including:

- The input clock to the sample rate generator, which can be either the internal clock source or a dedicated external clock source via the CLKX, CLKR, or CLKS pins. See Section 8.2.5.3.1 for details on the source of the McBSP internal clock.
- The input clock source (internal clock source or external clock CLKS) to the sample rate generator can be divided-down by a programmable value (CLKGDV bit in the sample rate generator register (SRGR)) to drive CLKG.

Regardless of the source to the sample rate generator, the rising edge of CLKSRG (see Figure 8-5) generates CLKG and FSG.

8.2.5.3.1 Input Clock Source Mode: CLKSM and SCLKME

The sample rate generator input clock signal can be driven from one of four sources selectable with the SCLKME bit in the pin control register (PCR) and the CLKSM bit in the sample rate generator register (SRGR), see Table 8-2.

Table 8-2. Choosing an Input Clock for the Sample Rate Generator With the SCLKME and CLKSM Bits

SCLKME Bit in PCR	CLKSM Bit in SRGR	Input Clock for Sample Rate Generator
0	0	Signal on CLKS pin
0	1	McBSP internal input clock
1	0	Signal on CLKR pin
1	1	Signal on CLKX pin

8.2.5.3.2 Sample Rate Generator Data Bit Clock Rate: CLKGDV

The first divider stage generates the serial data bit clock from the input clock. This divider stage uses a counter that is preloaded by the CLKGDV bit in the sample rate generator register (SRGR) and that contains the divide ratio value. The output of this stage is the data bit clock that is output on the sample rate generator output, CLKG, and that serves as the input for the second and third divider stages.

CLKG has a frequency equal to 1/(CLKGDV + 1) of the sample rate generator input clock. Thus, the sample rate generator input clock frequency is divided by a value between 1 to 256. The CLKGDV value chosen must result in a clock that meets the timing requirements/limitations specified in the device-specific data manual.

When CLKGDV is an odd value or equal to 0, the CLKG duty cycle is 50%. Note that an odd CLKGDV value means an even divide down of the source clock and an even CLKGDV value means an odd divide down of the source clock. When CLKGDV is an even value (2p), the high state duration is p + 1 cycles and the low state duration is p cycles. This is illustrated in Example 8-1, Example 8-2, and Example 8-3.

In the following examples:

S_{IN} = sample generator input clock period

f_{IN} = sample generator input clock frequency

S_G = CLKG period

f_G = CLKG frequency

The following equation is given above: $f_G = f_{IN}/(CLKGDV + 1)$; therefore, $S_G = (CLKGDV + 1) \times S_{IN}$.

Example 8-1. CLKGDV = 0

CLKGDV = 0

$S_G = (CLKGDV + 1) \times S_{IN} = (0 + 1) \times S_{IN} = S_{IN}$

Pulse width high = $S_{IN} \times (CLKGDV + 1)/2 = S_{IN} \times (0+ 1)/2 = 0.5 \times S_{IN}$

Pulse width low = $S_{IN} \times (CLKGDV + 1)/2 = S_{IN} \times (0+ 1)/2 = 0.5 \times S_{IN}$

Example 8-2. CLKGDV = 1

CLKGDV = 1

$S_G = (CLKGDV + 1) \times S_{IN} = (1 + 1) \times S_{IN} = 2 \times S_{IN}$

Pulse width high = $S_{IN} \times (CLKGDV + 1)/2 = S_{IN} \times (1 +1)/2 = S_{IN}$

Pulse width low = $S_{IN} \times (CLKGDV + 1)/2 = S_{IN} \times (1+ 1)/2 = S_{IN}$

Example 8-3. CLKGDV = 2

CLKGDV = 2

$S_G = (CLKGDV + 1) \times S_{IN} = (2 + 1) \times S_{IN} = 3 \times S_{IN}$

Pulse width high = $S_{IN} \times (CLKGDV/2 + 1) = S_{IN} \times (2/2 + 1) = 2 \times S_{IN}$

Pulse width low = $S_{IN} \times CLKGDV/2 = S_{IN} \times 2/2 = 1 \times S_{IN}$

8.2.5.3.3 Bit Clock Polarity: CLKSP

The external clock (CLKS) is selected to drive the sample rate generator clock divider by selecting CLKSM = 0 in the sample rate generator register (SRGR) and SCLKME = 0 in the pin control register (PCR). In this case, the CLKSP bit in SRGR selects the edge of CLKS on which sample rate generator data bit clock (CLKG) and frame sync signal (FSG) are generated. Since the rising edge of CLKSRG generates CLKG and FSG, the rising edge of CLKS when CLKSP = 0 or the falling edge of CLKS when CLKSP = 1 causes the transition on CLKG and FSG.

8.2.5.3.4 Bit Clock and Frame Synchronization

When the external clock (CLKS) is selected to drive the sample rate generator (CLKSM = 0 in SRGR and SCLKME = 0 in PCR), the GSYNC bit in SRGR can be used to configure the timing of CLKG relative to CLKS. GSYNC = 1 ensures that the McBSP and the external device to which it is communicating are dividing down the CLKS with the same phase relationship. If GSYNC = 0, this feature is disabled and CLKG runs freely and is not resynchronized. If GSYNC = 1, an inactive-to-active transition on FSR triggers a resynchronization of CLKG and the generation of FSG. CLKG always begins at a high state after synchronization. Also, FSR is always detected at the same edge of CLKS that generates CLKG, regardless of the length the FSR pulse. Although an external FSR is provided, FSG can still drive internal receive frame synchronization when GSYNC = 1. When GSYNC = 1, FPER does not matter, because the frame period is determined by the arrival of the external frame sync pulse.

Figure 8-6 and Figure 8-7 show this operation with various polarities of CLKS and FSR. These figures assume that FWID is 0, for a FSG = 1 CLKG wide.

These figures show what happens to CLKG when it is initially in sync and GSYNC = 1, as well as when it is not in sync with the frame synchronization and GSYNC = 1.

Figure 8-6. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 1

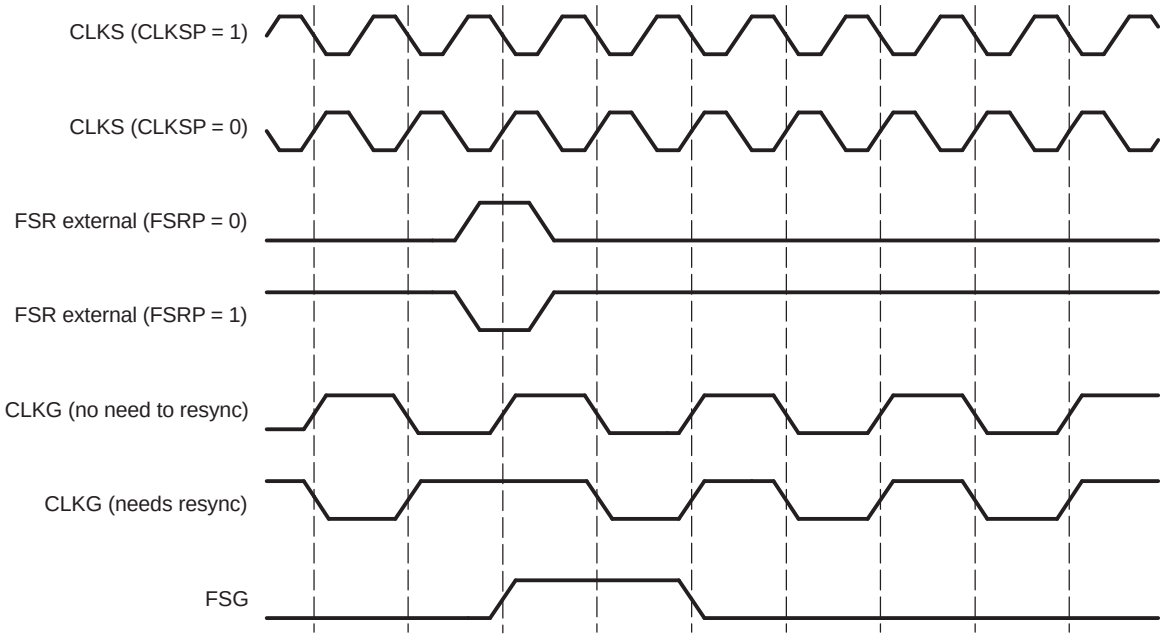
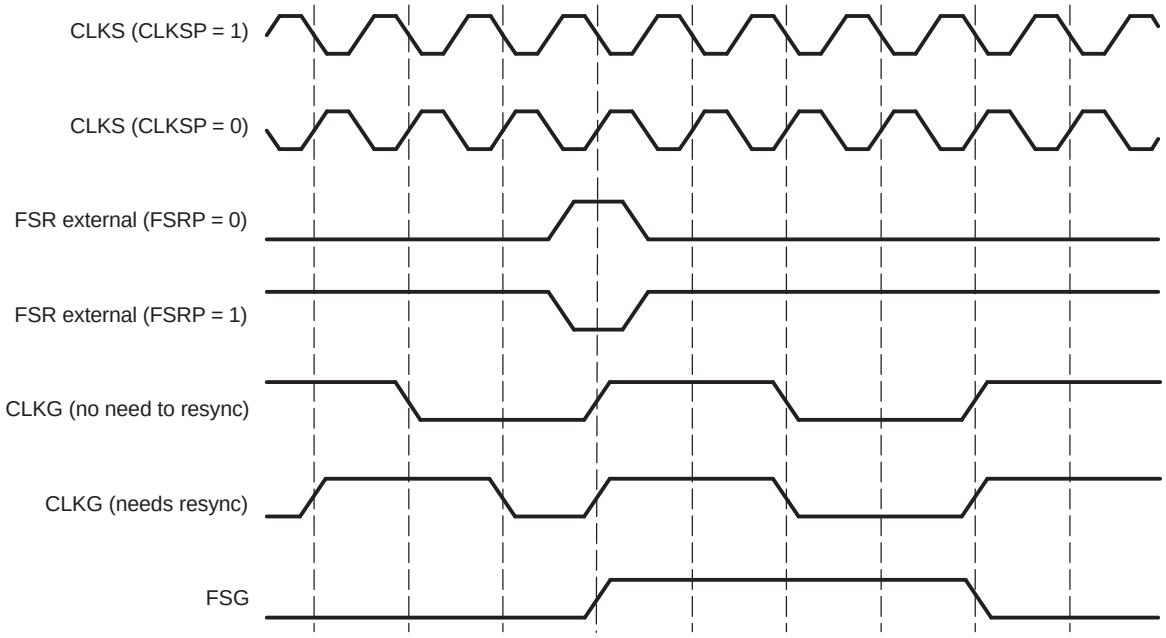


Figure 8-7. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 3



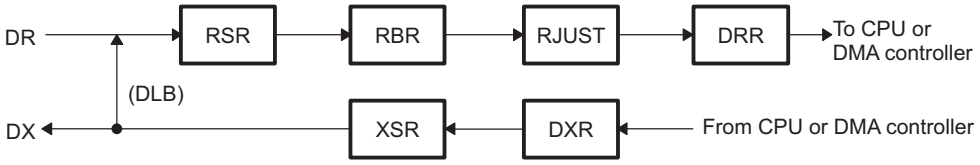
When GSYNC = 1, the transmitter can operate synchronously with the receiver, provided that the following conditions are met:

- FSX is programmed to be driven by the sample rate generator frame sync, FSG (FSGM = 1 in SRGR and FSXM = 1 in PCR).
- The sample-rate generator clock should drive the transmit and receive bit clock (CLK(R/X)M = 1 in SPCR). Therefore, the CLK(R/X) pin should not be driven by any other source.

8.2.5.3.5 Digital Loopback Mode: DLB

Setting DLB = 1 in SPCR enables digital loopback mode. In DLB mode, DR, FSR, and CLKR are internally connected through multiplexers to DX, FSX, and CLKX, respectively, as shown in Figure 8-2 and Figure 8-8. DLB mode allows testing of serial port code without using the external interface of the McBSP. CLKX and FSX must be enabled as outputs (CLKXM = FSXM = 1) in DLB mode.

Figure 8-8. Digital Loopback Mode



8.2.5.3.6 Receive Clock Selection: DLB, CLKRM

Table 8-3 shows how the digital loopback bit (DLB) in the serial port control register (SPCR) and the CLKRM bit in the pin control register (PCR) select the receiver clock. In digital loopback mode (DLB = 1), the transmitter clock drives the receiver. CLKRM determines whether the CLKR pin is an input or an output.

Table 8-3. Receive Clock Selection

DLB Bit in SPCR	CLKRM Bit in PCR	Source of Receive Clock	CLKR Function
0	0	CLKR acts as an input driven by the external clock and inverted as determined by CLKRP before being used.	Input.
0	1	The sample rate generator clock (CLKG) drives CLKR.	Output. CLKG inverted as determined by CLKRP before being driven out on CLKR.
1	0	CLKX_int drives the receive clock CLKR_int as selected and is inverted.	High impedance.
1	1	CLKX_int drives CLKR_int as selected and is inverted.	Output. CLKR (same as CLKX) is inverted as determined by CLKRP before being driven out.

8.2.5.3.7 Transmit Clock Selection: CLKXM

Table 8-4 shows how the CLKXM bit in the pin control register (PCR) selects the transmit clock and whether the CLKX pin is an input or output.

Table 8-4. Transmit Clock Selection

CLKXM Bit in PCR	Source of Transmit Clock	CLKX Function
0	The external clock drives the CLKX input pin. CLKX is inverted as determined by CLKXP before being used.	Input.
1	The sample rate generator clock (CLKG) drives the transmit clock.	Output. CLKG is inverted as determined by CLKXP before being driven out on CLKX.

8.2.5.3.8 Stopping Clocks

Two methods can be used to stop serial clocks between data transfers:

- The SPI™ CLKSTP mode where clocks are stopped between single-element transfers. This mode is not supported on this device.
- The clocks are inputs to the McBSP (CLKXM or CLKRM = 0 in the pin control register (PCR)) and the McBSP operates in non-SPI mode (the clocks can be stopped between data transfers). If the external device stops the serial clock between data transfers, the McBSP interprets it as a slowed-down serial clock. Ensure that there are no glitches on the CLK(R/X) lines as the McBSP may interpret them as clock-edge transitions. Restarting the serial clock is equivalent to a normal clock transition after a slow CLK(R/X) cycle. Note that just as in normal operations, transmit under flow (XEMPTY) may occur if the DXR is not properly serviced at least three CLKX cycles before the next frame sync. Therefore, if the serial clock is stopped before DXR is properly serviced, the external device needs to restart the clock at least three CLKX cycles before the next frame sync to allow the DXR write to be properly synchronized. See Figure 8-29 for a graphical explanation on when DXR needs to be written to avoid underflow.

8.2.5.4 Frame Sync Generation

Data frame synchronization is independently programmable for the receiver and the transmitter for all data delay values. When the FRST bit in the serial port control register (SPCR) is set to 1 the frame generation logic is activated to generate frame sync signals, provided that FSGM = 1 in the sample rate generator register (SRGR). The frame sync programming options are:

- A frame pulse with a programmable period between sync pulses and a programmable active width specified in SRGR.
- The transmitter can trigger its own frame sync signal that is generated by a DXR-to-XSR copy. This causes a frame sync to occur on every DXR-to-XSR copy. The data delays can be programmed as required. However, maximum packet frequency cannot be achieved in this method for data delays of 1 and 2.
- Both the receiver and transmitter can independently select an external frame synchronization on the FSR and FSX pins, respectively.

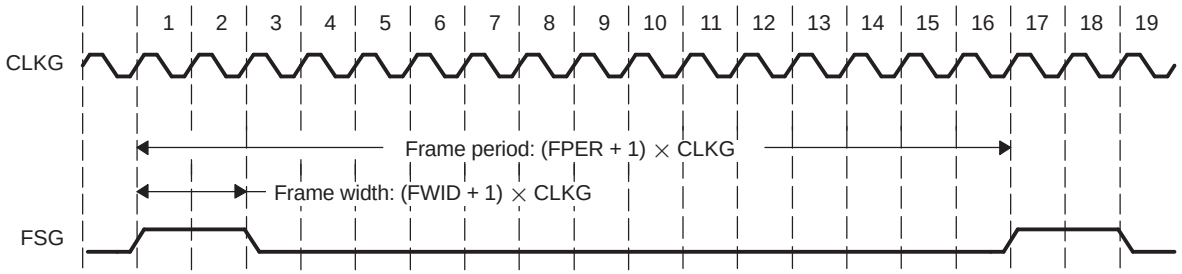
8.2.5.4.1 Frame Period (FPER) and Frame Width (FWID)

The FPER bits in the sample rate generator register (SRGR) are a 12-bit down-counter that can count down the generated data clocks from 4095 to 0. FPER controls the period of active frame sync pulses. The FWID bits in SRGR are an 8-bit down-counter. FWID controls the active width of the frame sync pulse.

When the sample rate generator comes out of reset, FSG is in an inactive (low) state. After this, when FRST = 1 in SPCR and FSGM = 1 in SRGR, frame sync signals are generated. The frame width value (FWID + 1) is counted down on every CLKG cycle until it reaches 0 when FSG goes low. Thus, the value of FWID + 1 determines an active frame pulse width ranging from 1 to 256 data bit clocks. At the same time, the frame period value (FPER + 1) is also counting down, and when this value reaches 0, FSG goes high again, indicating a new frame is beginning. Thus, the value of FPER + 1 determines a frame length from 1 to 4096 data bits. When GSYNC = 1 in SRGR, the value of FPER does not matter. Figure 8-9 shows a frame of 16 CLKG periods (FPER = 15 or 0000 1111b).

It is recommended that FWID be programmed to a value less than (R/X)WDLEN1/2.

Figure 8-9. Programmable Frame Period and Width



8.2.5.4.2 Receive Frame Synchronization Selection: DLB and FSRM

Table 8-5 shows how you can select various sources to provide the receive frame synchronization signal. Note that in digital loopback mode (DLB = 1 in the serial port control register (SPCR)), the transmit frame sync signal is used as the receive frame sync signal and that DR is internally connected to DX.

NOTE: FSR_int and FSX_int are shown in Figure 8-2.

Table 8-5. Receive Frame Synchronization Selection

DLB Bit in SPCR	FSRM Bit in PCR	GSYNC Bit in SRGR	Source of Receive Frame Synchronization	FSR Pin Function
0	0	X	External frame sync signal drives the FSR input pin, whose signal is then inverted as determined by FSRP before being used as FSR_int.	Input.
0	1	0	Sample rate generator frame sync signal (FSG) drives FSR_int, FRST = 1.	Output. FSG is inverted as determined by FSRP before being driven out on the FSR pin.
0	1	1	Sample rate generator frame sync signal (FSG) drives FSR_int, FRST = 1.	Input. The external frame sync input on FSR is used to synchronize CLKG and generate FSG.
1	0	0	FSX_int drives FSR_int. FSX is selected as shown in Table 8-6.	High impedance.
1	X	1	FSX_int drives FSR_int and is selected as shown in Table 8-6.	Input. External FSR is not used for frame synchronization but is used to synchronize CLKG and generate FSG since GSYNC = 1.
1	1	0	FSX_int drives FSR_int and is selected as shown in Table 8-6.	Output. Receive (same as transmit) frame synchronization is inverted as determined by FSRP before being driven out.

8.2.5.4.3 Transmit Frame Synchronization Selection: FSXM and FSGM

Table 8-6 shows how you can select the source of the transmit frame synchronization signal. The three choices are:

- External frame sync input
- The sample rate generator frame sync signal, FSG
- A signal that indicates a DXR-to-XSR copy has been made

NOTE: FSR_int and FSX_int are shown in Figure 8-2.

Table 8-6. Transmit Frame Synchronization Selection

FSXM Bit in PCR	FSGM Bit in SRGR	Source of Transmit Frame Synchronization	FSX Pin Function
0	X	External frame sync input on the FSX pin. This is inverted by FSXP before being used as FSX_int.	Input.
1	1	Sample rate generator frame sync signal (FSG) drives FSX_int. FRST = 1.	Output. FSG is inverted by FSXP before being driven out on FSX.
1	0	A DXR-to-XSR copy activates transmit frame sync signal.	Output. 1-bit-clock-wide signal inverted as determined by FSXP before being driven out on FSX.

8.2.5.4.4 Frame Detection

To facilitate detection of frame synchronization, the receive and transmit CPU interrupts (RINT and XINT) can be programmed to detect frame synchronization by setting the RINTM and XINTM bits in the serial port control register (SPCR) to 10b. The associated portion (receiver/transmitter) of the McBSP must be out of reset.

8.2.5.5 Data and Frames

8.2.5.5.1 Frame Synchronization Phases

Frame synchronization indicates the beginning of a transfer on the McBSP. The data stream following frame synchronization can have up to two phases, phase 1 and phase 2. The number of phases can be selected by the phase bit, (R/X)PHASE, in RCR and XCR. The number of elements per frame and bits per element can be independently selected for each phase via (R/X)FRLEN1/2 and (R/X)WDLEN1/2, respectively. Figure 8-10 shows a frame in which the first phase consists of two elements of 12 bits, each followed by a second phase of three elements of 8 bits each. The entire bit stream in the frame is contiguous; no gaps exist either between elements or phases. Table 8-7 shows the fields in the receive/transmit control registers (RCR/XCR) that control the frame length and element length for each phase for both the receiver and the transmitter. The maximum number of elements per frame is 128 for a single-phase frame and 256 elements in a dual-phase frame. The number of bits per element can be 8, 12, 16, 20, 24, or 32.

NOTE: For a dual-phase frame with internally generated frame synchronization, the maximum number of elements per phase depends on the word length. This is because the frame period, FPER, is only 12-bits wide and, therefore, provides 4096 bits per frame. Hence, the maximum number of 256 elements per dual- phase frame applies only when the WDLEN is 16 bits. However, any combination of element numbers and element size (defined by the FRLEN and WDLEN bits, respectively) is valid as long as their product is less than or equal to 4096 bits. This limitation does not apply for dual-phase with external frame sync.

Figure 8-10. Dual-Phase Frame Example

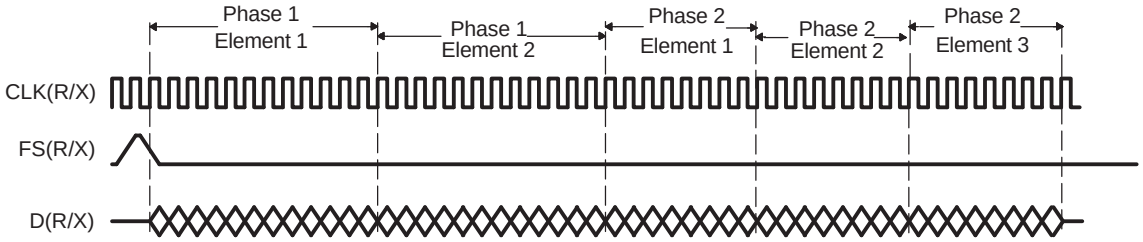


Table 8-7. RCR/XCR Fields Controlling Elements per Frame and Bits per Element

Serial Port	Frame Phase	RCR/XCR Field Control	
		Elements per Frame	Bits per Element
Receive	1	RFRLEN1	RWDLEN1
Receive	2	RFRLEN2	RWDLEN2
Transmit	1	XFRLEN1	XWDLEN1
Transmit	2	XFRLEN2	XWDLEN2

8.2.5.5.2 Frame Length: RFRLEN1/2 and XFRLEN1/2

Frame length specifies the maximum number of serial elements or logical time slots or channels that are available for transfer per frame synchronization signal. In multichannel selection mode, the frame length value is independent of, and perhaps different from, the actual number of channels that the device is programmed to receive or transmit per frame via the MCR, RCERE*n*, and XCERE*n* registers. See Section 8.2.8 for details on multichannel selection mode operation. The 7-bit (R/X)FRLEN1/2 bits in (R/X)CR support up to 128 elements per phase in a frame, as shown in Table 8-8. (R/X)PHASE = 0 selects a single-phase data frame, and (R/X)PHASE = 1 selects a dual-phase frame for the data stream. For a single-phase frame, the value of (R/X)FRLEN2 does not matter. Program the frame length fields with (*w* minus 1), where *w* represents the number of elements per frame. For Figure 8-10, (R/X)FRLEN1 = 1 or 000 0001b and (R/X)FRLEN2 = 2 or 000 0010b.

Table 8-8. Receive/Transmit Frame Length Configuration

(R/X)PHASE	(R/X)FRLEN1	(R/X)FRLEN2	Frame Length
0	0 ≤ n ≤ 127	x	Single-phase frame; (n + 1) elements per frame
1	0 ≤ n ≤ 127	0 ≤ m ≤ 127	Dual-phase frame; (n + 1) plus (m + 1) elements per frame

8.2.5.5.3 Element Length: RWDLEN1/2 and XWDLEN1/2

The (R/X)WDLEN1/2 fields in the receive/transmit control register (RCR and XCR) determine the element length in bits per element for the receiver and the transmitter for each phase of the frame, as indicated in Table 8-7. Table 8-9 shows how the value of these fields selects particular element lengths in bits. For the example in Figure 8-10, (R/X)WDLEN1 = 001b and (R/X)WDLEN2 = 000b. If (R/X)PHASE = 0, indicating a single-phase frame, then (R/X)WDLEN2 is not used by the McBSP and its value does not matter.

Table 8-9. Receive/Transmit Element Length Configuration

(R/X)WDLEN1/2	Element Length (Bits)
000	8
001	12
010	16
011	20
100	24
101	32
110	Reserved
111	Reserved

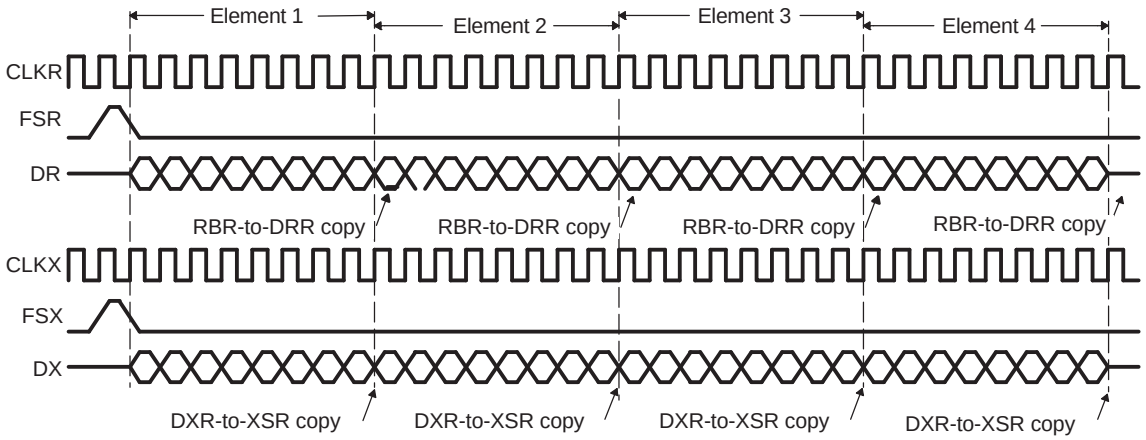
8.2.5.5.4 Data Packing Using Frame Length and Element Length

The frame length and element length can be manipulated to effectively pack data. For example, consider a situation in which four 8-bit elements are transferred in a single-phase frame, as shown in Figure 8-11. In this case:

- (R/X)PHASE = 0, indicating a single-phase frame
- (R/X)FRLEN1 = 000 0011b, indicating a 4-element frame
- (R/X)WDLEN1 = 000b, indicating 8-bit elements

In this example, four 8-bit data elements are transferred to and from the McBSP by the CPU or the DMA controller. Four reads of DRR and four writes of DXR are necessary for each frame.

Figure 8-11. Single-Phase Frame of Four 8-Bit Elements

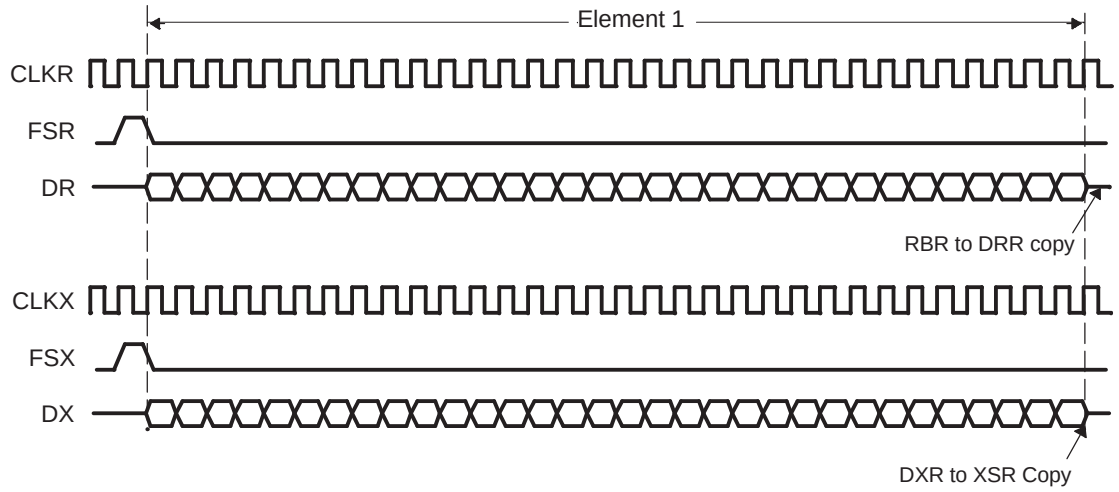


The example in Figure 8-11 can also be viewed as a data stream of a single-phase frame of one 32-bit data element, as shown in Figure 8-12. In this case:

- (R/X)PHASE = 0, indicating a single phase frame
- (R/X)FRLEN1 = 0, indicating a 1-element frame
- (R/X)WDLEN1 = 101b, indicating 32-bit elements

In this example, one 32-bit data element is transferred to and from the McBSP by the CPU or the DMA controller. Thus, one read of DRR and one write of DXR is necessary for each frame. As a result, the number of transfers is one-fourth that of the previous example (Figure 8-11). This manipulation reduces the percentage of bus time required for serial port data movement.

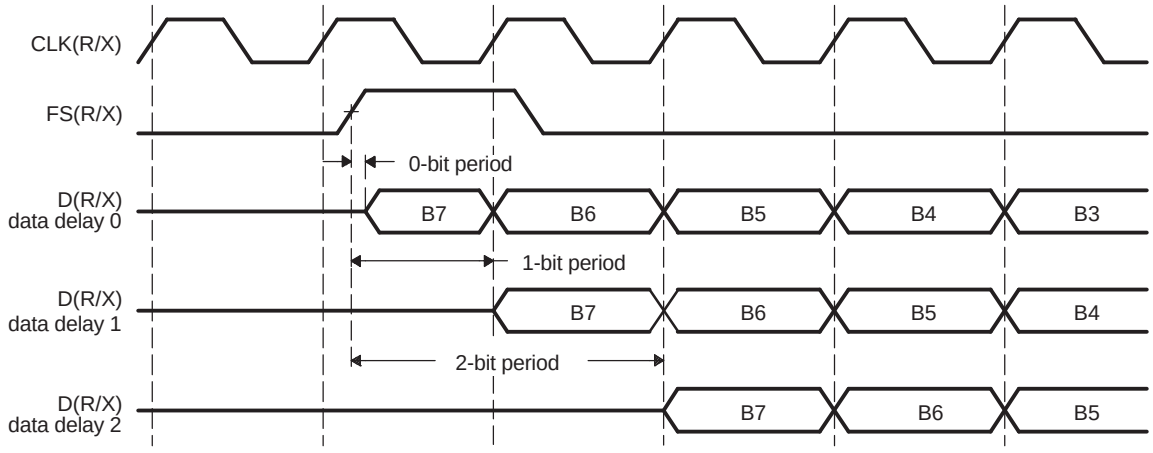
Figure 8-12. Single-Phase Frame of One 32-Bit Element



8.2.5.5.5 Data Delay: RDATDLY and XDATDLY

The start of a frame is defined by the first clock cycle in which frame synchronization is active. The beginning of actual data reception or transmission with respect to the start of the frame can be delayed if required. This delay is called data delay. RDATDLY (in RCR) and XDATDLY (in XCR) specify the data delay for reception and transmission, respectively. The range of programmable data delay is zero to two bit clocks ((R/X)DATDLY = 00b to10b), as shown in Figure 8-13. Typically, a 1-bit delay is selected because data often follows a 1-cycle active frame sync pulse.

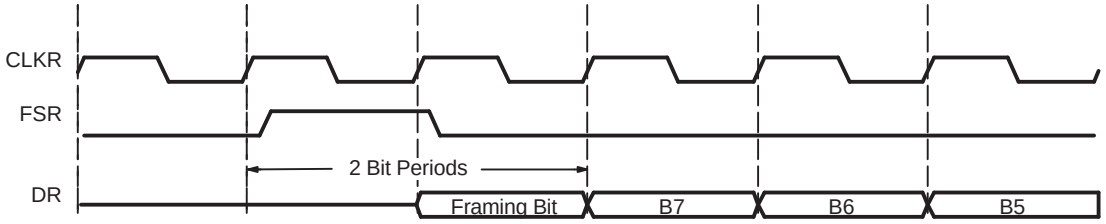
Figure 8-13. Data Delay



Normally, a frame sync pulse is detected or sampled with respect to an edge of serial clock CLK(R/X). Thus, on a subsequent cycle (depending on data delay value), data can be received or transmitted. However, in the case of a 0-bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle. For reception, this problem is solved by receive data being sampled on the first falling edge of CLKR when an active (high) FSR is detected. However, data transmission must begin on the rising edge of CLKX that generated the frame synchronization. Therefore, the first data bit is assumed to be in the XSR and DX. The transmitter then asynchronously detects the frame synchronization, FSX goes active, and it immediately starts driving the first bit to be transmitted on the DX pin.

Another common operation uses a data delay of 2. This configuration allows the serial port to interface to different types of T1 framing devices in which the data stream is preceded by a framing bit. During the reception of such a stream with a data delay of two bits, the framing bit appears after a 1-bit delay and data appears after a 2-bit delay). The serial port essentially discards the framing bit from the data stream, as shown in Figure 8-14. In transmission, by delaying the first transfer bit, the serial port essentially inserts a blank period (high-impedance period) in place of the framing bit. Here, it is expected that the framing device inserts its own framing bit or that the framing bit is generated by another device. Alternatively, you may pull up or pull down DX to achieve the desired value.

Figure 8-14. 2-Bit Data Delay Used to Discard Framing Bit



8.2.5.5.6 Receive Data Justification and Sign Extension: RJUST

The RJUST bit in the serial port control register (SPCR) selects whether data in RBR is right- or left-justified (with respect to the MSB) in the DRR. If right justification is selected, RJUST further selects whether the data is sign-extended or zero-filled. Table 8-10 and Table 8-11 summarize the effect that various values of RJUST have on example receive data.

Table 8-10. Effect of RJUST Bit Values With 12-Bit Example Data ABCh

RJUST Bit in SPCR	Justification	Extension	Value in DRR
00	Right	Zero-fill MSBs	0000 0ABCh
01	Right	Sign-extend MSBs	FFFF FABCh
10	Left	Zero-fill LSBs	ABC0 0000h
11	Reserved	Reserved	Reserved

Table 8-11. Effect of RJUST Bit Values With 20-Bit Example Data ABCDEh

RJUST Bit in SPCR	Justification	Extension	Value in DRR
00	Right	Zero-fill MSBs	000A BCDEh
01	Right	Sign-extend MSBs	FFFA BCDEh
10	Left	Zero-fill LSBs	ABCD E000h
11	Reserved	Reserved	Reserved

8.2.5.5.7 32-Bit Bit Reversal: RWDREVRS, XWDREVRS

Normally all transfers are sent and received with the MSB first; however, you can reverse the receive/transmit bit ordering of a 32-bit element (LSB first) using the 32-bit reversal feature of the McBSP by setting all of the following:

- (R/X)WDREVRS = 1 in the receive/transmit control register (RCR/XCR).
- (R/X)COMPAND = 01b in RCR/XCR.
- (R/X)WDLEN(1/2) = 101b in RCR/XCR to indicate 32-bit elements.

When you set the register fields as above, the bit ordering of the 32-bit element is reversed before being received by or sent from the serial port. If the (R/W)WDREVRS and (R/X)COMPAND fields are set as above, but the element size is not set to 32-bit, operation is undefined.

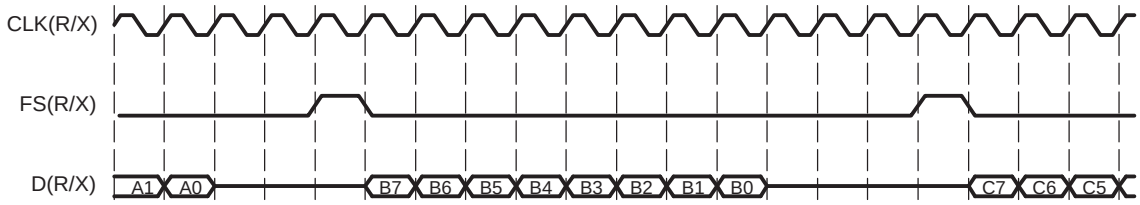
8.2.6 McBSP Standard Operation

During a serial transfer, there are typically periods of serial port inactivity between packets or transfers. The receive and transmit frame synchronization pulse occurs for every serial transfer. When the McBSP is not in the reset state and has been configured for the desired operation, a serial transfer can be initiated by programming (R/X)PHASE = 0 for a single-phase frame with the required number of elements programmed in (R/X)FRLEN1. The number of elements can range from 1 to 128 ((R/X)FRLEN1 = 00h to 7Fh). The required serial element length is set in the (R/X)WDLEN1 field in the (R/X)CR. If a dual-phase frame is required for the transfer, RPHASE = 1 and each (R/X)FRLEN1/2 can be set to any value between 0h and 7Fh.

Figure 8-15 shows a single-phase data frame of one 8-bit element. Since the transfer is configured for a 1-bit data delay, the data on the DX and DR pins are available one bit clock after FS(R/X) goes active. This figure, as well as all others in this section, use the following assumptions:

- (R/X)PHASE = 0, specifying a single-phase frame
- (R/X)FRLEN1 = 0b, specifying one element per frame
- (R/X)WDLEN1 = 000b, specifying eight bits per element
- (R/X)FRLEN2 = (R/X)WDLEN2 = Value is ignored
- CLK(R/X)P = 0, specifying that the receive data is clocked on the falling edge and that transmit data is clocked on the rising edge
- FS(R/X)P = 0, specifying that active (high) frame sync signals are used
- (R/X)DATDLY = 01b, specifying a 1-bit data delay

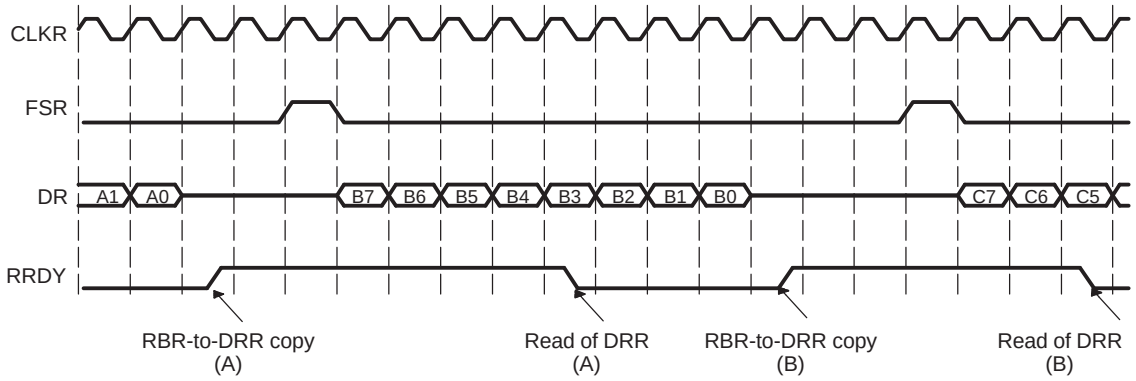
Figure 8-15. McBSP Standard Operation



8.2.6.1 Receive Operation

Figure 8-16 shows serial reception. Once the receive frame synchronization signal (FSR) transitions to its active state, it is detected on the first falling edge of the receivers CLKR. The data on the DR pin is then shifted into the receive shift register (RSR) after the appropriate data delay as set by the RDATDLY bit in the receive control register (RCR). The contents of RSR is copied to RBR at the end of every element on the rising edge of the clock, provided RBR is not full with the previous data. Then, an RBR-to-DRR copy activates the RRDY status bit in the serial port control register (SPCR) to 1 on the following falling edge of CLKR. This indicates that the receive data register (DRR) is ready with the data to be read by the CPU or the DMA controller. RRDY is deactivated when the DRR is read by the CPU or the DMA controller. See also Section 8.2.12.1.2 and Section 8.2.13.1.

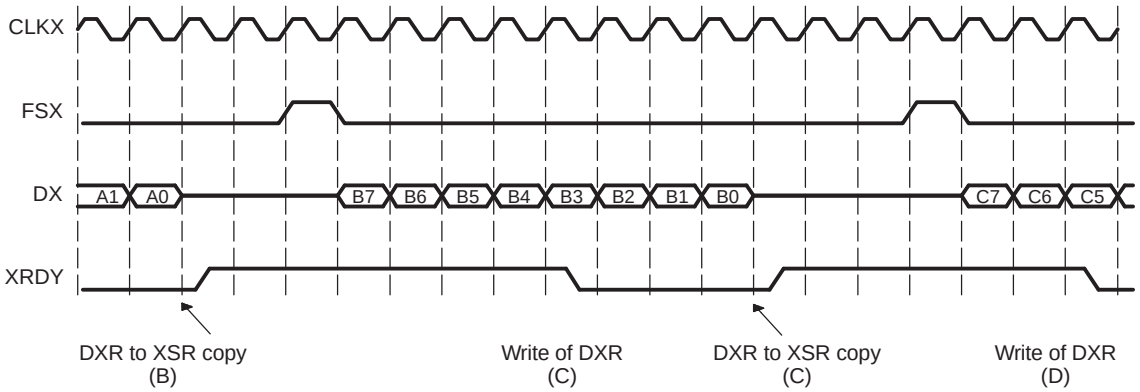
Figure 8-16. Receive Operation



8.2.6.2 Transmit Operation

Once transmit frame synchronization occurs, the value in the transmit shift register (XSR) is shifted out and driven on the DX pin after the appropriate data delay as set by the XDATDLY bit in the transmit control register (XCR). The XRDY bit in the serial port control register (SPCR) is activated after every DXR-to-XSR copy on the following falling edge of CLKX, indicating that the data transmit register (DXR) can be written with the next data to be transmitted. XRDY is deactivated when the DXR is written by the CPU or the DMA controller. Figure 8-17 illustrates serial transmission. See Section 8.2.6.5.4 for information on transmit operation when the transmitter is pulled out of reset (XRST = 1). See also Section 8.2.12.1.3 and Section 8.2.13.2.

Figure 8-17. Transmit Operation



8.2.6.3 Maximum Frame Frequency

The frame frequency is determined by the following equation, which calculates the period between frame synchronization signals:

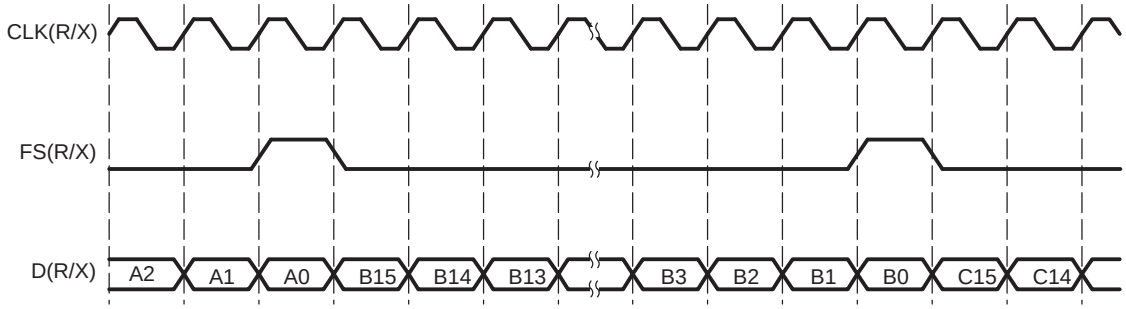
$$\text{Frame frequency} = \frac{\text{Bit-clock frequency}}{\text{Number of bit clocks between frame sync signals}}$$

The frame frequency may be increased by decreasing the time between frame synchronization signals in bit clocks (which is limited only by the number of bits per frame). As the frame transmit frequency is increased, the inactivity period between the data frames for adjacent transfers decreases to 0. The minimum time between frame synchronization pulses is the number of bits transferred per frame. This time also defines the maximum frame frequency, which is calculated by the following equation:

$$\text{Maximum frame frequency} = \frac{\text{Bit-clock frequency}}{\text{Number of bits per frame}}$$

Figure 8-18 shows the McBSP operating at maximum frame frequency. The data bits in consecutive frames are transmitted continuously with no inactivity between bits. If there is a 1-bit data delay, as shown, the frame synchronization pulse overlaps the last bit transmitted in the previous frame.

Figure 8-18. Maximum Frame Frequency for Transmit and Receive



NOTE: For (R/X)DATDLY = 0, the first bit of data transmitted is asynchronous to CLKX, as shown in Figure 8-13.

Maximum frame frequency may not be possible when the word length is 8-bits, depending on the clock divide value CLKGDV. The CPU or DMA may not be able to service serial port requests that occur as frequently as every 8-bit clocks. This situation can be resolved by allowing additional space between words or choosing a slower bit clock (larger CLKGDV value).

8.2.6.4 Frame Synchronization Ignore

The McBSP can be configured to ignore transmit and receive frame synchronization pulses. The (R/X)FIG bit in (R/X)CR can be cleared to 0 to recognize frame sync pulses, or it can be set to 1 to ignore frame sync pulses. In this way, you can use (R/X)FIG either to pack data, if operating at maximum frame frequency, or to ignore unexpected frame sync pulses.

8.2.6.4.1 Frame Sync Ignore and Unexpected Frame Sync Pulses

RFIG and XFIG are used to ignore unexpected internal or external frame sync pulses. Any frame sync pulse is considered unexpected if it occurs one or more bit clocks earlier than the programmed data delay from the end of the previous frame specified by ((R/X)DATDLY). Setting the frame ignore bits to 1 causes the serial port to ignore these unexpected frame sync signals.

In reception, if not ignored (RFIG = 0), an unexpected FSR pulse discards the contents of RSR in favor of the incoming data. Therefore, if RFIG = 0, an unexpected frame synchronization pulse aborts the current data transfer, sets RSYNCERR in SPCR to 1, and begins the reception of a new data element. When RFIG = 1, the unexpected frame sync pulses are ignored.

In transmission, if not ignored (XFIG = 0), an unexpected FSX pulse aborts the ongoing transmission, sets the XSYNCERR bit in SPCR to 1, and reinitiates transmission of the current element that was aborted. When XFIG = 1, unexpected frame sync signals are ignored.

Figure 8-19 shows that element B is interrupted by an unexpected frame sync pulse when (R/X)FIG = 0. The reception of B is aborted (B is lost), and a new data element (C) is received after the appropriate data delay. This condition causes a receive synchronization error and thus sets the RSYNCERR bit. However, for transmission, the transmission of B is aborted and the same data (B) is retransmitted after the appropriate data delay. This condition is a transmit synchronization error and thus sets the XSYNCERR bit. Synchronization errors are discussed in Section 8.2.6.5.2 and Section 8.2.6.5.5.

Figure 8-19. Unexpected Frame Synchronization With (R/X)FIG = 0

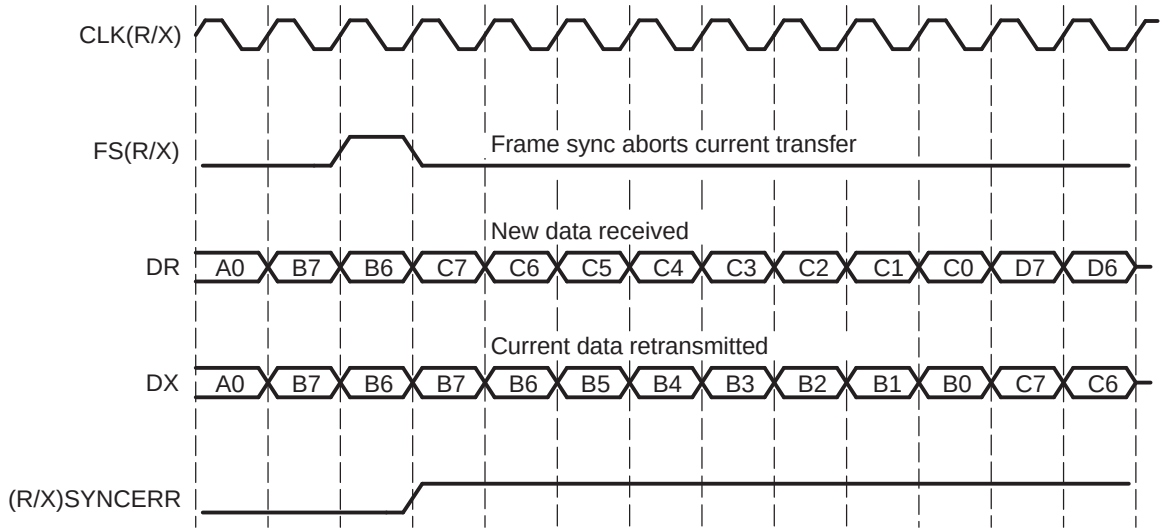
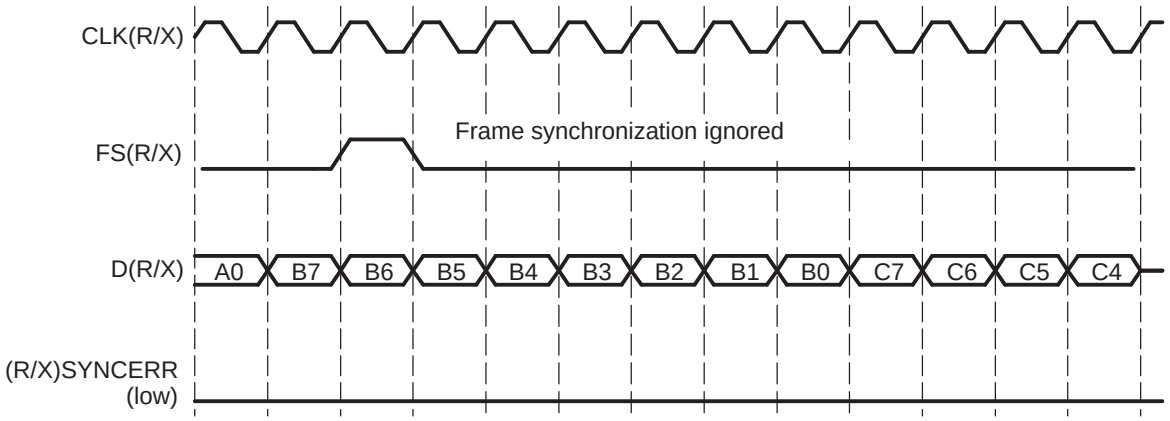


Figure 8-20 shows McBSP operation when unexpected internal or external frame synchronization signals are ignored by setting (R/X)FIG = 1. Here, the transfer of element B is not affected by an unexpected frame synchronization.

Figure 8-20. Unexpected Frame Synchronization With (R/X)FIG = 1



8.2.6.4.2 Data Packing using Frame Sync Ignore Bits

Section 8.2.5.5.4 describes one method of changing the element length and frame length to simulate 32-bit serial element transfers, thus requiring much less bus bandwidth than four 8-bit transfers require. This example works when there are multiple elements per frame.

Now consider the case of the McBSP operating at maximum packet frequency, as shown in Figure 8-21. Here, each frame has only a single 8-bit element. This stream takes one read transfer and one write transfer for each 8-bit element. Figure 8-22 shows the McBSP configured to treat this stream as a continuous stream of 32-bit elements. In this example, (R/X)FIG is set to 1 to ignore unexpected subsequent frames. Only one read transfer and one write transfer is needed every 32 bits. This configuration effectively reduces the required bus bandwidth to one-fourth of the bandwidth needed to transfer four 8-bit blocks.

Figure 8-21. Maximum Frame Frequency Operation With 8-Bit Data

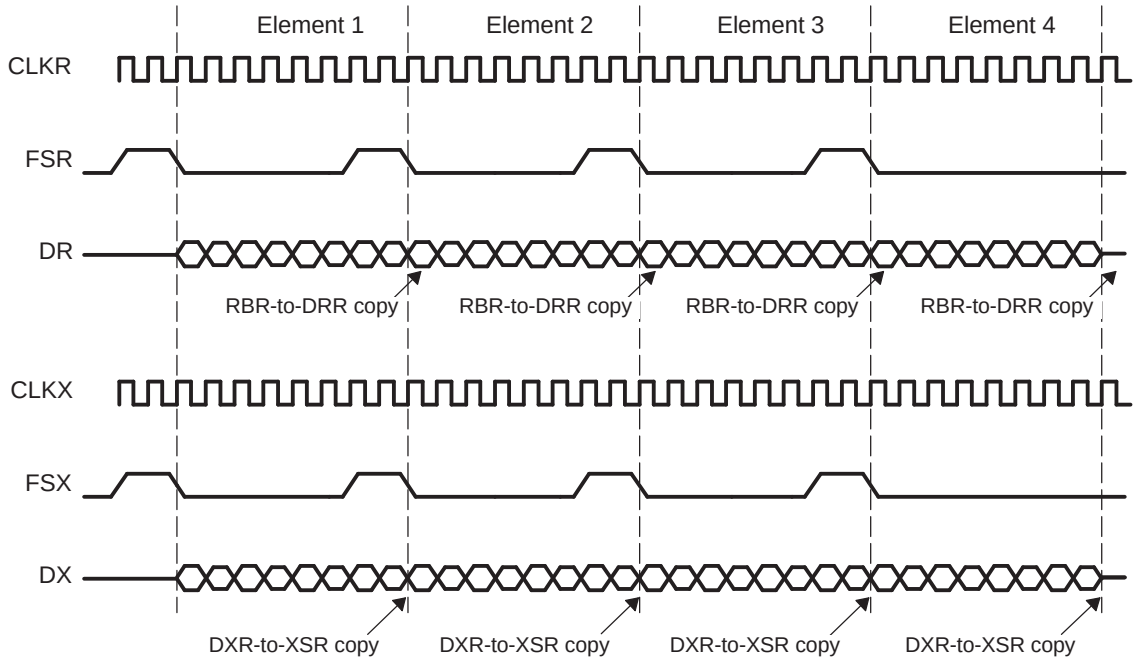
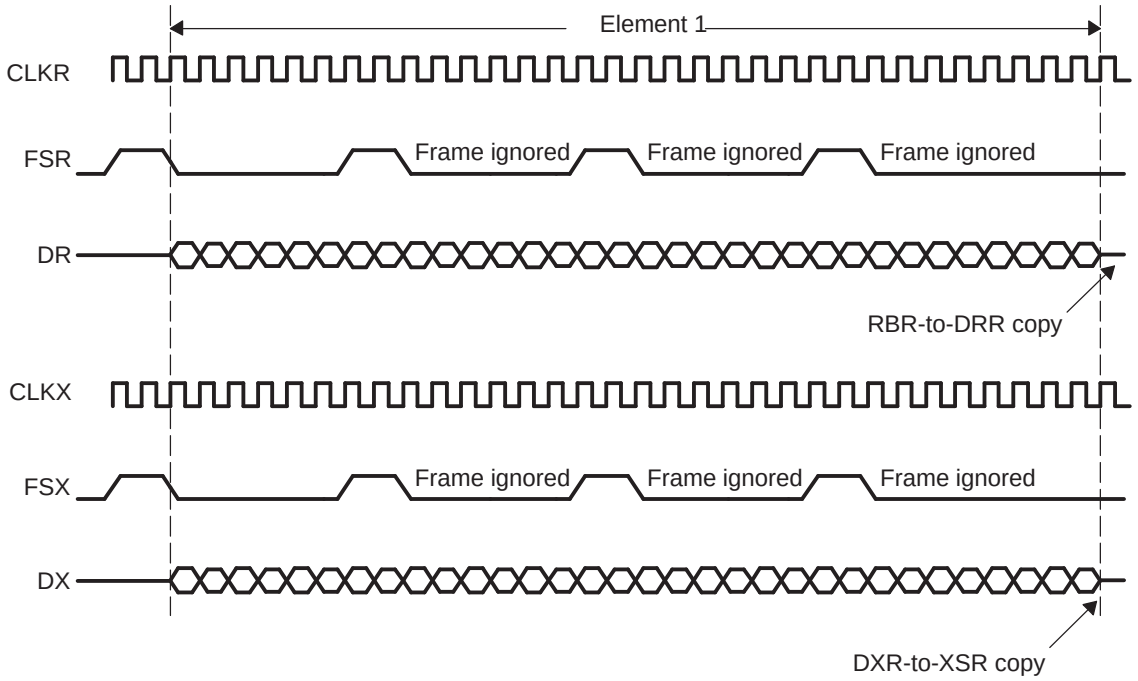


Figure 8-22. Data Packing at Maximum Frame Frequency With (R/X)FIG = 1



8.2.6.5 Serial Port Exception Conditions

There are five serial port events that can constitute a system error:

- Receive overrun (RFULL = 1 in SPCR)
- Unexpected receive frame synchronization (RSYNCERR = 1 in SPCR)
- Transmit data overwrite
- Transmit empty (XEMPTY = 0 in SPCR)
- Unexpected transmit frame synchronization (XSYNCERR = 1 in SPCR)

8.2.6.5.1 Receive Overrun: RFULL

RFULL = 1 in the serial port control register (SPCR) indicates that the receiver has experienced overrun and is in an error condition. RFULL is set when the following conditions are met:

- DRR has not been read since the last RBR-to-DRR transfer.
- RBR is full and an RBR-to-DRR copy has not occurred.
- RSR is full and an RSR-to-RBR transfer has not occurred.

The data arriving on DR is continuously shifted into RSR (Figure 8-23). Once a complete element is shifted into RSR, an RSR-to-RBR transfer can occur only if an RBR-to-DRR copy is complete. Therefore, if DRR has not been read by the CPU or the DMA controller since the last RBR-to-DRR transfer (RRDY = 1), an RBR-to-DRR copy does not take place until RRDY = 0. This prevents an RSR-to-RBR copy. New data arriving on the DR pin is shifted into RSR, and the previous contents of RSR are lost. After the receiver starts running from reset, a minimum of three elements must be received before RFULL can be set, because there was no last RBR-to-DRR transfer before the first element.

This data loss can be avoided if DRR is read no later than two and a half CLKR cycles before the end of the third element (data C) in RSR, as shown in Figure 8-24.

Either of the following events clears the RFULL bit to 0 and allows subsequent transfers to be read properly:

- Reading DRR
- Resetting the receiver (RRST = 0) or the device

Another frame synchronization is required to restart the receiver.

Figure 8-23 shows the receive overrun condition. Because element A is not read before the reception of element B is complete, B is not transferred to DRR yet. Another element, C, arrives and fills RSR. DRR is finally read, but not earlier than two and one half cycles before the end of element C. New data D overwrites the previous element C in RSR. If RFULL is still set after the DRR is read, the next element can overwrite D if DRR is not read in time.

Figure 8-24 shows the case in which RFULL is set but the overrun condition is averted by reading the contents of DRR at least two and a half cycles before the next element, C, is completely shifted into RSR. This ensures that a RBR-to-DRR copy of data B occurs before the next element is transferred from RSR to RBR.

Figure 8-23. Serial Port Receive Overrun

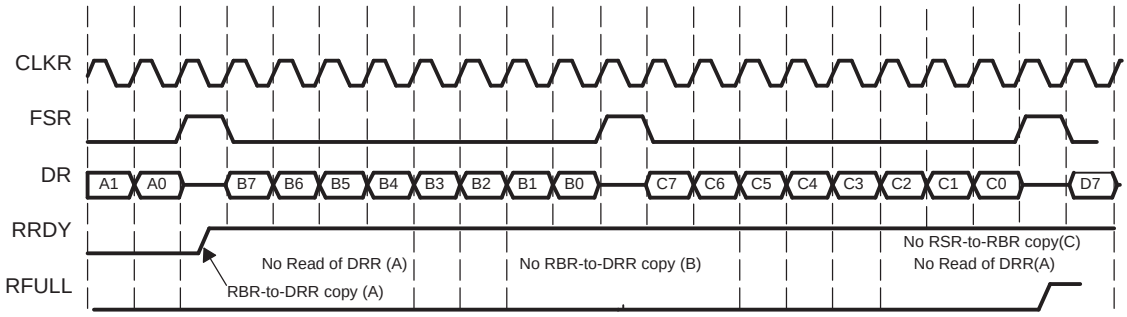
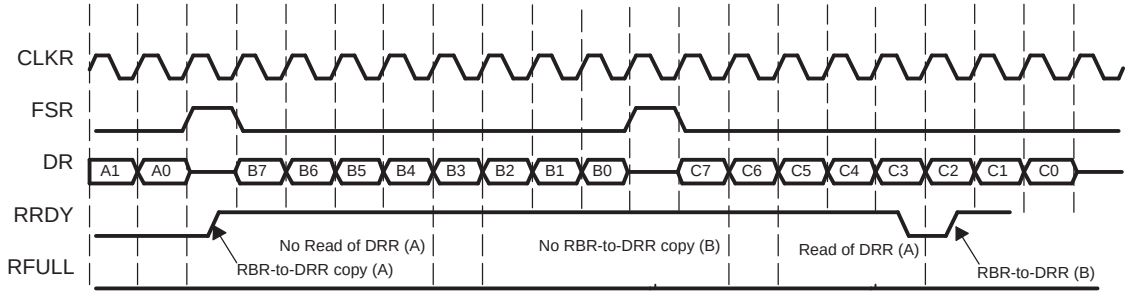


Figure 8-24. Serial Port Receive Overrun Avoided



8.2.6.5.2 Unexpected Receive Frame Synchronization: RSYNCERR

Figure 8-25 shows the decision tree that the receiver uses to handle all incoming frame synchronization pulses. The diagram assumes that the receiver has been activated (RRST = 1). Unexpected frame sync pulses can originate from an external source or from the internal sample rate generator. An unexpected frame sync pulse is defined as a sync pulse which occurs RDATDLY bit clocks earlier than the last transmitted bit of the previous frame. Any one of three cases can occur:

- Case 1: Unexpected FSR pulses with RFIG = 1. This case is discussed in Section 8.2.6.4.1 and shown in Figure 8-20. Here, receive frame sync pulses are ignored and the reception continues.
- Case 2: Normal serial port reception. There are three reasons for a receive not to be in progress:
  - This FSR is the first after RRST = 1.
  - This FSR is the first after DRR has been read clearing an RFULL condition.
  - The serial port is in the inter-packet intervals. The programmed data delay (RDATDLY) for reception may start during these inter-packet intervals for the first bit of the next element to be received. Thus, at maximum frame frequency, frame synchronization can still be received RDATDLY bit clocks before the first bit of the associated element.

    For this case, reception continues normally, because these are not unexpected frame sync pulses.
- Case 3: Unexpected receive frame synchronization with RFIG = 0 (unexpected frame not ignored). This case was shown in Figure 8-19 for maximum packet frequency. Figure 8-26 shows this case during normal operation of the serial port with time intervals between packets. Unexpected frame sync pulses are detected when they occur the value in RDATDLY bit clocks before the last bit of the previous element is received on DR. In both cases, RSYNCERR in SPCR is set. RSYNCERR can be cleared only by receiver reset or by writing a 0 to this bit in SPCR. If RINTM = 11b in SPCR, RSYNCERR drives the receive interrupt (RINT) to the CPU.

NOTE: Note that the RSYNCERR bit in SPCR is a read/write bit, so writing a 1 to it sets the error condition. Typically, writing a 0 is expected.

Figure 8-25. Decision Tree Response to Receive Frame Synchronization Pulse

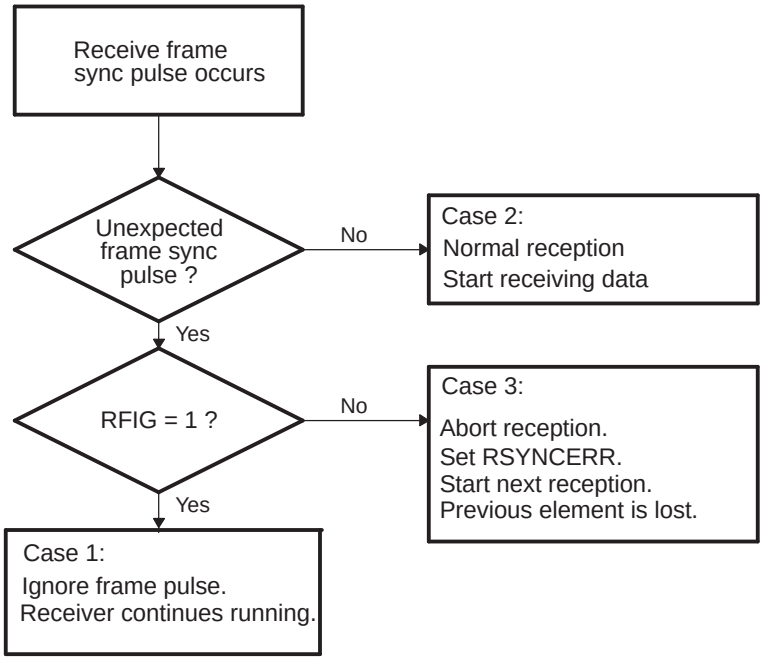
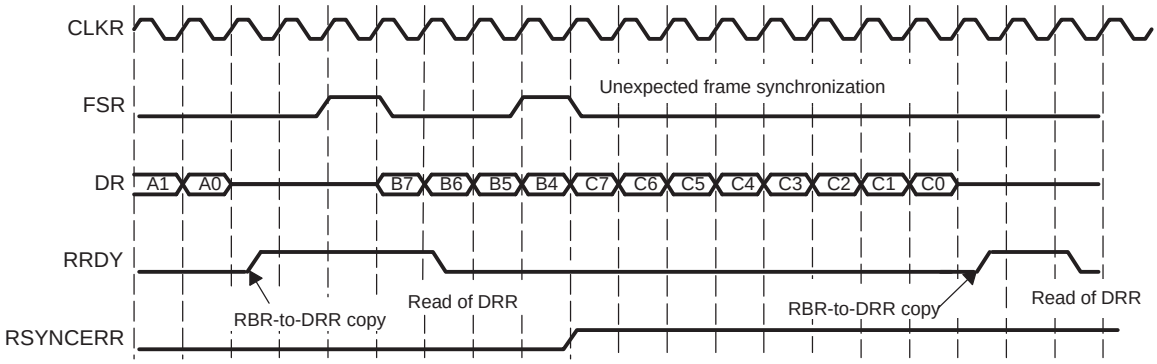


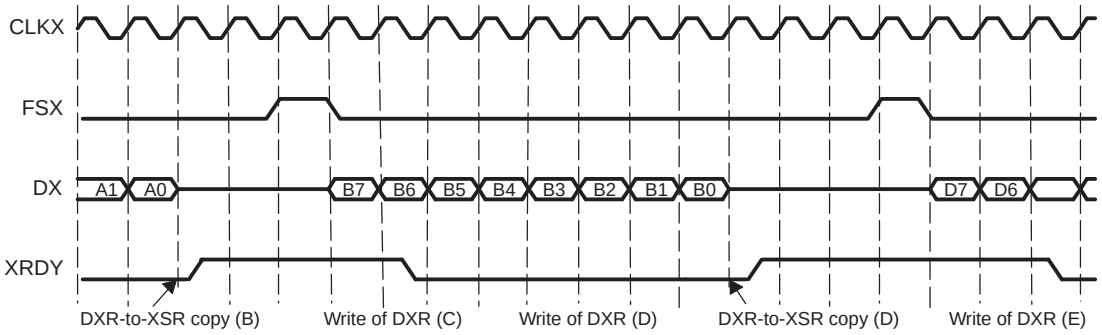
Figure 8-26. Unexpected Receive Frame Synchronization Pulse



8.2.6.5.3 Transmit With Data Overwrite

Figure 8-27 shows what happens if the data in DXR is overwritten before it is transmitted. Suppose you load the DXR with data C. A subsequent write to the DXR overwrites C with D before C is copied to the XSR. Thus, C is never transmitted on DX. The CPU can avoid overwriting data by polling XRDY before writing to DXR or by waiting for a programmed XINT to be triggered by XRDY (XINTM = 00b). The DMA controller can avoid overwriting by synchronizing data writes with XEVT. See also Section 8.2.12.1.3.

Figure 8-27. Transmit With Data Overwrite



8.2.6.5.4 Transmit Empty: XEMPTY

XEMPTY indicates whether the transmitter has experienced underflow. Either of the following conditions causes XEMPTY to become active (XEMPTY = 0):

- During transmission, DXR has not been loaded since the last DXR-to-XSR copy, and all bits of the data element in XSR have been shifted out on DX.
- The transmitter is reset (XRST = 0 or the device is reset), and then restarted.

During an underflow condition, the transmitter continues to transmit the old data in DXR for every new frame sync signal FSX (generated by an external device, or by the internal sample rate generator) until a new element is loaded into DXR by the CPU or the DMA controller. XEMPTY is deactivated (XEMPTY = 1) when this new element in DXR is transferred to XSR. In the case when the FSX is generated by a DXR-to-XSR copy (FSXM = 1 in PCR and FSGM = 0 in SRGR), the McBSP does not generate any new frame sync until new data is written to the DXR and a DXR-to-XSR copy occurs.

When the transmitter is taken out of reset (XRST = 1), it is in a transmit ready (XRDY = 1) and transmit empty (XEMPTY = 0) condition. If DXR is loaded by the CPU or the DMA controller before FSX goes active, a valid DXR-to-XSR transfer occurs. This allows for the first element of the first frame to be valid even before the transmit frame sync pulse is generated or detected. Alternatively, if a transmit frame sync is detected before DXR is loaded, 0s are output on DX.

Figure 8-28 shows a transmit underflow condition. After B is transmitted, B is retransmitted on DX if you fail to reload the DXR before the subsequent frame synchronization. Figure 8-29 shows the case of writing to DXR just before a transmit underflow condition that would otherwise occur. After B is transmitted, C is written to DXR before the next transmit frame sync pulse occurs.

Figure 8-28. Transmit Empty

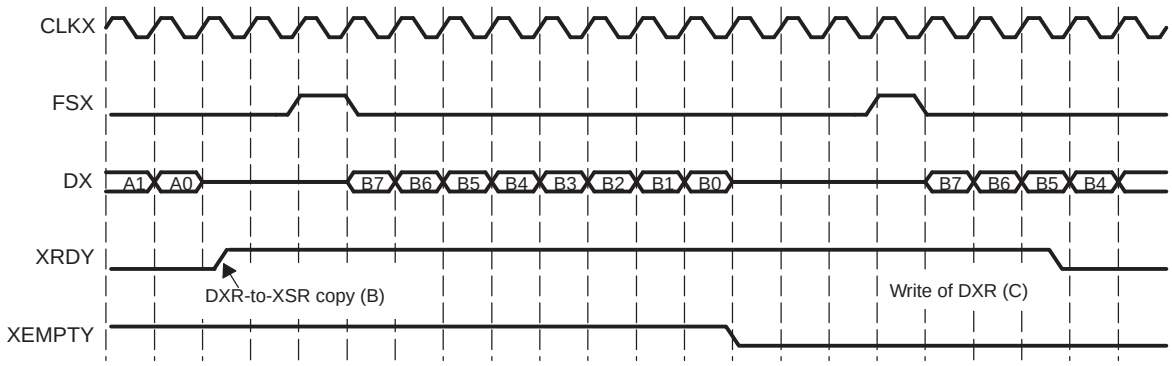
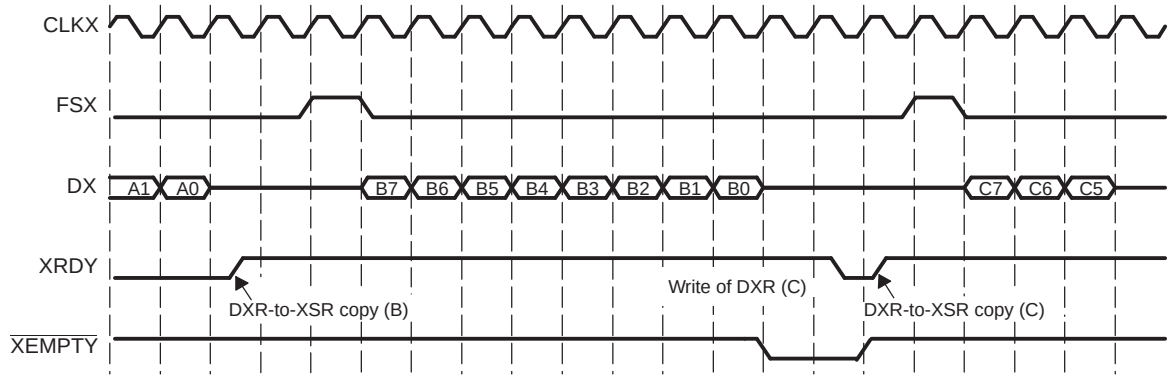


Figure 8-29. Transmit Empty Avoided



8.2.6.5.5 Unexpected Transmit Frame Synchronization: XSYNCERR

A transmit frame sync error (XSYNCERR) may occur the first time the transmitter is enabled (XRST = 1) after a device reset. To avoid this, after enabling the transmitter for the first time, the following procedure must be followed:

1. Wait for two CLKG cycles. The unexpected frame sync error (XSYNCERR), if any, occurs within this time period.
2. Disable the transmitter (XRST = 0). This clears any XSYNCERR.
3. Re-enable the transmitter (XRST = 1).

See also Section 8.2.11 for details on initialization procedure.

Figure 8-30 shows the decision tree that the transmitter uses to handle all incoming frame synchronization signals. The diagram assumes that the transmitter has been started (XRST = 1). An unexpected transmit frame sync pulse is defined as a sync pulse that occurs XDATDLY bit clocks earlier than the last transmitted bit of the previous frame. Any one of three cases can occur:

- Case 1: Unexpected FSX pulses with XFIG = 1. This case is discussed in Section 8.2.6.4.1 and shown in Figure 8-20. In this case, unexpected FSX pulses are ignored, and the transmission continues.
- Case 2: FSX pulses with normal serial port transmission. This situation is discussed in Section 8.2.6.2. There are two possible reasons for a transmit not to be in progress:
  - This FSX pulse is the first one to occur after XRST = 1.
  - The serial port is in the interpacket intervals. The programmed data delay (XDATDLY) may start during these interpacket intervals before the first bit of the next element is transmitted. Thus, if operating at maximum packet frequency, frame synchronization can still be received XDATDLY bit clocks before the first bit of the associated element.
- Case 3: Unexpected transmit frame synchronization with XFIG = 0. The case was shown in Figure 8-19 for frame synchronization with XFIG = 0 at maximum packet frequency. Figure 8-31 shows the case for normal operation of the serial port with interpacket intervals. In both cases, XSYNCERR in SPCR is set. XSYNCERR can be cleared only by transmitter reset or by writing a 0 to this bit in SPCR. If XINTM = 11b in SPCR, XSYNCERR drives the receive interrupt (XINT) to the CPU.

NOTE: The XSYNCERR bit in SPCR is a read/write bit, so writing a 1 to it sets the error condition. Typically, writing a 0 is expected.

Figure 8-30. Decision Tree Response to Transmit Frame Synchronization Pulse

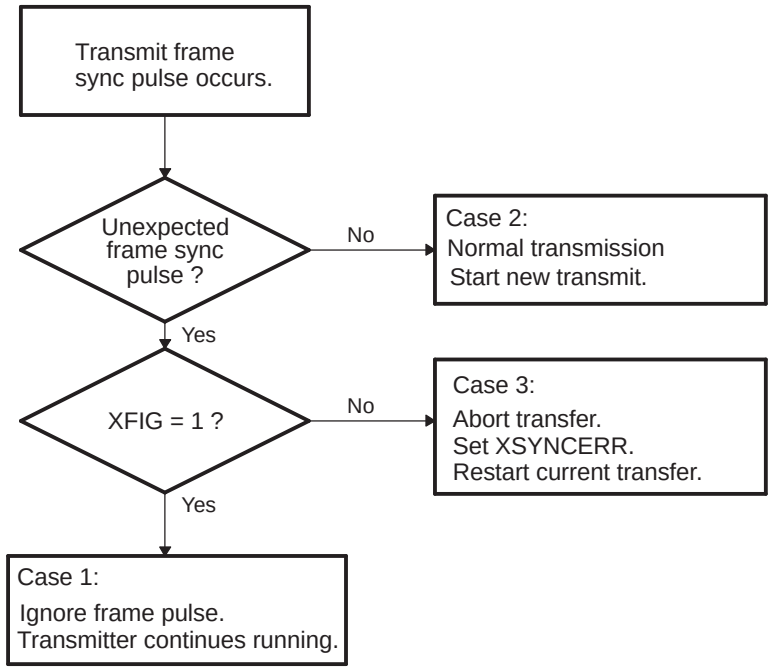
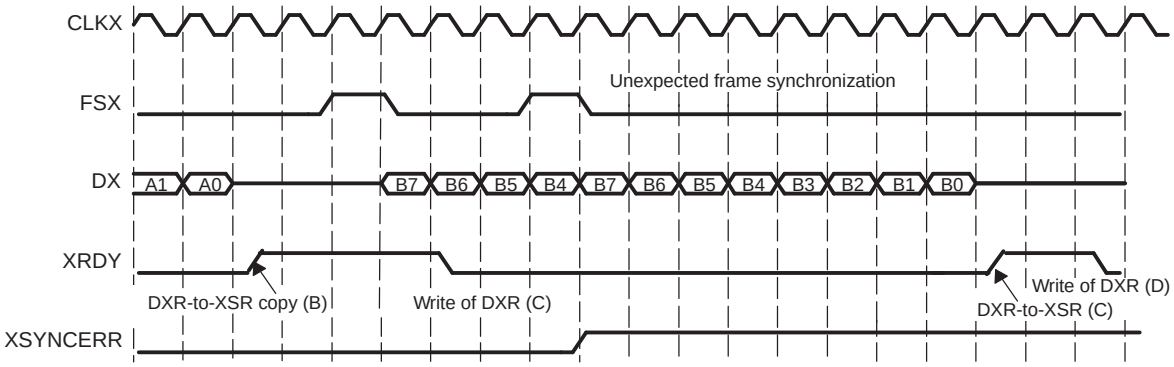


Figure 8-31. Unexpected Transmit Frame Synchronization Pulse



8.2.7 μ-Law/A-Law Companding Hardware Operation

Companding (compressing and expanding) hardware allows compression and expansion of data in either μ-law or A-law format. The specification for μ-law and A-law log PCM is part of the CCITT G.711 recommendation. The companding standard employed in the United States and Japan is μ-law and allows 14 bits of dynamic range. The European companding standard is A-law and allows 13 bits of dynamic range. Any values outside these ranges are set to the most positive or most negative value. Thus, for companding to work best here, the data transferred to and from the McBSP via the CPU or the DMA controller must be at least 16 bits wide.

The μ-law and A-law formats encode data into 8-bit code elements. Companded data is always 8 bits wide, so the appropriate (R/X)WDLEN1/2 must be cleared to 0, indicating an 8-bit serial data stream. If companding is enabled and either phase of the frame does not have an 8-bit element length, companding continues as if the element length is eight bits.

When companding is used, transmit data is encoded according to the specified companding law, and receive data is decoded to 2s-complement format. Companding is enabled and the desired format is selected by appropriately setting (R/X)COMPAND in the (R/X)CR. Compression occurs during the process of copying data from DXR to XSR and expansion occurs from RBR to DRR, as shown in Figure 8-32.

For transmit data to be compressed, it should be 16-bit, left-justified data, such as LAW16, as shown in Figure 8-33. The value can be either 13 or 14 bits wide, depending on the companding law. This 16-bit data is aligned in DXR, as shown in Figure 8-34.

For reception, the 8-bit compressed data in RBR is expanded to a left-justified 16-bit data, LAW16. This can be further justified to 32-bit data by programming the RJUST bits in the serial port control register (SPCR), as shown in Table 8-12.

Figure 8-32. Companding Flow

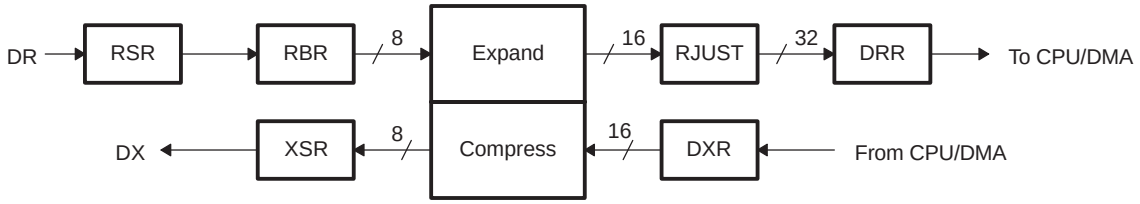


Figure 8-33. Companding Data Formats

LAW16	15	2	1	0
μ-Law	Value		0	0

LAW16	15	3	2	1	0
A-Law	Value		0	0	0

Figure 8-34. Transmit Data Companding Format in DXR

31	16	15	0
Don't care		LAW16	

Table 8-12. Justification of Expanded Data in DRR

RJUST Bit in SPCR	DRR Bits			
	31	16	15	0
00	0			LAW16
01	sign			LAW16
10	LAW16			0
11	Reserved			

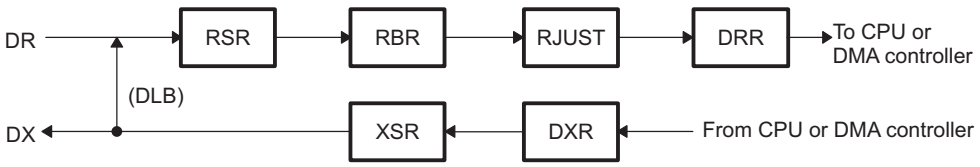
8.2.7.1 Companding Internal Data

If the McBSP is unused, the companding hardware can compand internal data. This hardware can be used to:

- Convert linear data to the appropriate μ-law or A-law format.
- Convert μ-law or A-law data to the linear format.
- Observe the quantization effects in companding by transmitting linear data and compressing and re-expanding this data. This is useful only if both XCOMPAND and RCOMPAND enable the same companding format.

Figure 8-35 shows the method by which the McBSP can compand internal data. The data path is indicated by the (DLB) arrow. The McBSP is enabled in digital loopback (DLB) mode with companding appropriately enabled by the RCOMPAND and XCOMPAND bits. Receive and transmit interrupts (RINT when RINTM = 0 and XINT when XINTM = 0) or synchronization events (REVT and XEVT) allow synchronization of the CPU or the DMA controller to these conversions, respectively. The time for this companding depends on the serial bit rate selected.

Figure 8-35. Companding of Internal Data



8.2.7.2 Bit Ordering

Normally, all transfers on the McBSP are sent and received with the MSB first. However, certain 8-bit data protocols (that do not use companded data) require the LSB to be transferred first. By setting the (R/X)COMPAND = 01b in (R/X)CR, the bit ordering of 8-bit elements is reversed (LSB first) before being sent to the serial port. Like the companding feature, this feature is enabled only if the appropriate (R/X)WDLEN1/2 bit is cleared to 0, indicating that 8-bit elements are to be serially transferred. A 32-bit bit reversal feature is also available, as shown in Section 8.2.5.5.7

8.2.8 Multichannel Selection Modes

This section defines and provides the functions and all related information concerning the multichannel selection modes.

8.2.8.1 Channels, Blocks, and Partitions

A McBSP channel is a time slot for shifting in/out the bits of one serial word. Each McBSP supports up to 128 channels for reception and 128 channels for transmission. In the receiver and in the transmitter, the 128 available channels are divided into eight blocks that each contain 16 contiguous channels:

Block 0: Channels 0 through 15	Block 4: Channels 64 through 79
Block 1: Channels 16 through 31	Block 5: Channels 80 through 95
Block 2: Channels 32 through 47	Block 6: Channels 96 through 111
Block 3: Channels 48 through 63	Block 7: Channels 112 through 127

The blocks are assigned to partitions according to the selected partition mode. In the 2-partition mode, you assign one even-numbered block (0, 2, 4, or 6) to partition A and one odd-numbered block (1, 3, 5, or 7) to partition B. In the 8-partition mode, blocks 0 through 7 are automatically assigned to partitions, A through H, respectively.

The number of partitions for reception and the number of partitions for transmission are independent. For example, it is possible to use 2 receive partitions (A and B) and 8 transmit partitions (A through H).

8.2.8.2 Multichannel Selection

When McBSP uses a time-division multiplexed (TDM) data stream while communicating with other McBSPs or serial devices, the McBSP may need to receive and/or transmit on only a few channels. To save memory and bus bandwidth, you can use a multichannel selection mode to prevent data flow in some of the channels. The McBSP has one receive multichannel selection mode and three transmit multichannel selection modes.

Each channel partition has a dedicated channel enable register. If the appropriate multichannel selection mode is on, each bit in the register controls whether data flow is allowed or prevented in one of the channels that is assigned to that partition.

8.2.8.3 Configuring a Frame for Multichannel Selection

Before you enable a multichannel selection mode, make sure you properly configure the data frame:

- Select a single-phase frame (RPHASE/XPHASE = 0). Each frame represents a TDM data stream.
- Set a frame length (RFRLEN1/XFRLEN1) that includes the highest-numbered channel that is to be used. For example, if you plan to use channels 0, 15, and 39 for reception, the receive frame length must be at least 40 (RFRLEN1 = 39). If XFRLEN1 = 39 in this case, the receiver creates 40 time slots per frame but only receives data during time slots 0, 15, and 39 of each frame.

NOTE: The frame-sync pulse can be generated internally by the sample rate generator or it can be supplied externally by another source. In a multichannel mode configuration with external frame-sync generation, the McBSP transmitter will ignore the first frame-sync pulse after it is taken out of reset. The transmitter will transmit only on the second frame-sync pulse. The receiver will shift in data on the first frame-sync pulse, regardless of whether it is generated internally or externally.

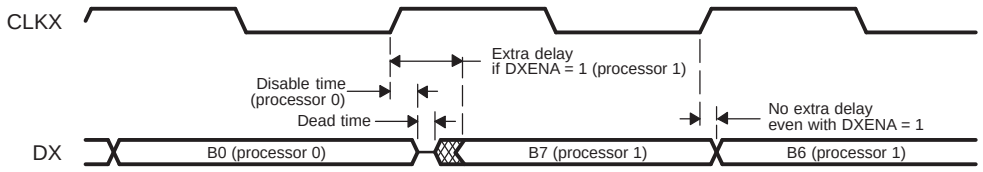
8.2.8.4 DX Enabler: DXENA

The DXENA bit in the serial port control register (SPCR) controls the high-impedance enable on the DX pin. When DXENA = 1, the McBSP enables extra delay for the DX pin turn-on time. This feature is useful for McBSP multichannel operations, such as in a time-division multiplexed (TDM) system. The McBSP supports up to 128 channels in a multichannel operation. These channels can be driven by different devices in a TDM data communication line, such as the T1/E1 line. In any multichannel operation where multiple devices transmit over the same DX line, you need to ensure that no two devices transmit data simultaneously, which results in bus contention. Enough dead time should exist between the transmission of the first data bit of the current device and the transmission of the last data bit of the previous device. In other words, the last data bit of the previous device needs to be disabled to a high-impedance state before the next device begins transmitting data to the same data line, as shown in Figure 8-36.

When two McBSPs are used to transmit data over the same TDM line, bus contention occurs if DXENA = 0. The first McBSP turns off the transmission of the last data bit (changes DX from valid to a high-impedance state) after a disable time specified in the data manual. As shown in Figure 8-36, this disable time is measured from the CLKX active clock edge. The next McBSP turns on its DX pin (changes from a high-impedance state to valid) after a delay time. Again, this delay time is measured from the CLKX active clock edge. Bus contention occurs because the dead time between the two devices is not enough. You need to apply alternative software or hardware methods to ensure proper multichannel operation in this case.

If you set DXENA = 1 in the second McBSP, the second McBSP turns on its DX pin after some extra delay time. This ensures that the previous McBSP on the same DX line is disabled before the second McBSP starts driving out data. The DX enabler controls only the high-impedance enable on the DX pin, not the data itself. Data is shifted out to the DX pin at the same time as in the case when DXENA = 0. The only difference is that with DXENA = 1, the DX pin is masked to a high-impedance state for some extra CPU cycles before the data is seen on the TDM data line. Therefore, only the first bit of data is delayed. Refer to the specific device datasheet for the exact amount of delay.

Figure 8-36. DX Timing for Multichannel Operation



8.2.8.5 Using Two Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use two partitions or eight partitions. If you choose the 2-partition mode (RMCME = 0 for reception, XMCME = 0 for transmission), McBSP channels are activated using an alternating scheme. In response to a frame-sync pulse, the receiver or transmitter begins with the channels in partition A and then alternates between partitions B and A until the complete frame has been transferred. When the next frame-sync pulse occurs, the next frame is transferred, beginning with the channels in partition A.

8.2.8.5.1 Assigning Blocks to Partitions A and B

For reception, any two of the eight receive-channel blocks can be assigned to receive partitions A and B (see Table 8-13), which means up to 32 receive channels can be enabled at any given point in time. Similarly, any two of the eight transmit-channel blocks (up 32 enabled transmit channels) can be assigned to transmit partitions A and B (see Table 8-14). You can dynamically change which blocks of channels are assigned to the partitions, see Section 8.2.8.5.2.

For reception:

- Assign an even-numbered channel block (0, 2, 4, or 6) to receive partition A by writing to the RPABLK bit in the multichannel control register (MCR). In the receive multichannel selection mode, the channels in this partition are controlled by the enhanced receive channel enable register partition A (RCEREA).
- Assign an odd-numbered block (1, 3, 5, or 7) to receive partition B with the RPBBLK bit in MCR. In the receive multichannel selection mode, the channels in this partition are controlled by the enhanced receive channel enable register partition B (RCEREB).

For transmission:

- Assign an even-numbered channel block (0, 2, 4, or 6) to transmit partition A by writing to the XPABLK bit in the multichannel control register (MCR). In one of the transmit multichannel selection modes, the channels in this partition are controlled by the enhanced transmit channel enable register partition A (XCEREA).
- Assign an odd-numbered block (1, 3, 5, or 7) to transmit partition B with the XPBBLK bit in MCR. In one of the transmit multichannel selection modes, the channels in this partition are controlled by the enhanced transmit channel enable register partition B (XCEREB).

Table 8-13. Receive Channel Assignment and Control When Two Receive Partitions are Used

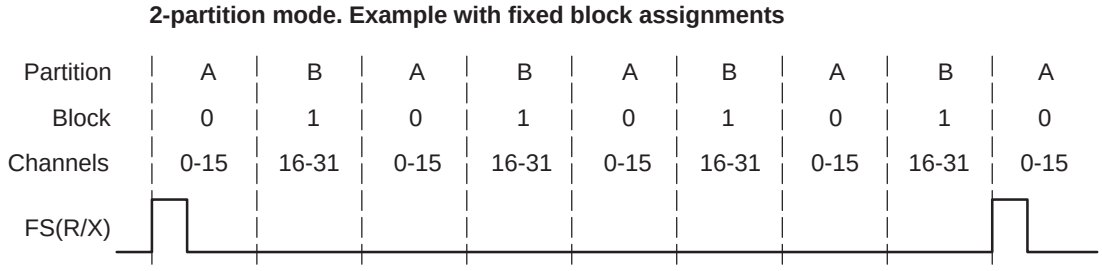
Receive Partition	Assigned Block of Receive Channels	RPABLK Bit in MCR	RPBBLK Bit in MCR	RCEREA/B Bits
A	Block 0: channels 0 through 15	0	–	RCEA0-RCEA15
A	Block 2: channels 32 through 47	1h	–	RCEA0-RCEA15
A	Block 4: channels 64 through 79	2h	–	RCEA0-RCEA15
A	Block 6: channels 96 through 111	3h	–	RCEA0-RCEA15
B	Block 1: channels 16 through 31	–	0	RCEB0-RCEB15
B	Block 3: channels 48 through 63	–	1h	RCEB0-RCEB15
B	Block 5: channels 80 through 95	–	2h	RCEB0-RCEB15
B	Block 7: channels 112 through 127	–	3h	RCEB0-RCEB15

Table 8-14. Transmit Channel Assignment and Control When Two Transmit Partitions are Used

Transmit Partition	Assigned Block of Transmit Channels	XPABLK Bit in MCR	XPBBLK Bit in MCR	XCEREA/B Bits
A	Block 0: channels 0 through 15	0	–	XCEA0-XCEA15
A	Block 2: channels 32 through 47	1h	–	XCEA0-XCEA15
A	Block 4: channels 64 through 79	2h	–	XCEA0-XCEA15
A	Block 6: channels 96 through 111	3h	–	XCEA0-XCEA15
B	Block 1: channels 16 through 31	–	0	XCEB0-XCEB15
B	Block 3: channels 48 through 63	–	1h	XCEB0-XCEB15
B	Block 5: channels 80 through 95	–	2h	XCEB0-XCEB15
B	Block 7: channels 112 through 127	–	3h	XCEB0-XCEB15

Figure 8-37 shows an example of alternating between the channels of partition A and the channels of partition B. Channels 0-15 have been assigned to partition A, and channels 16-31 have been assigned to partition B. In response to a frame-sync pulse, the McBSP begins a frame transfer with partition A and then alternates between partitions B and A until the complete frame is transferred.

Figure 8-37. Alternating Between the Channels of Partition A and the Channels of Partition B



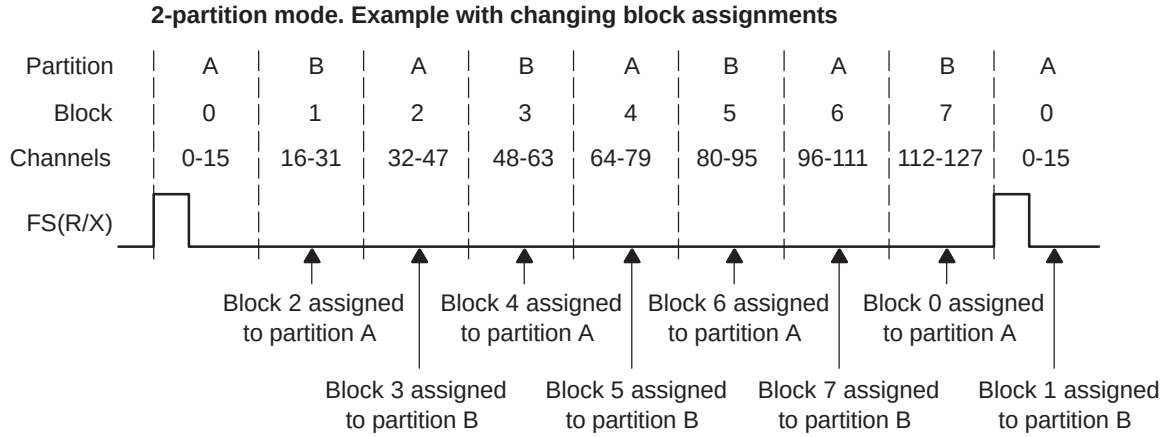
8.2.8.5.2 Reassigning Blocks During Reception/Transmission

If you want to use more than 32 channels, you can change which channel blocks are assigned to partitions A and B during the course of a data transfer. However, these changes must be carefully timed. While a partition is being transferred, it's the associated block assignment bits cannot be modified, and its associated channel enable register cannot be modified. For example, if block 3 is being transferred and block 3 is assigned to partition A, you cannot modify (R/X)PABLK to assign different channels to partition A, and you cannot modify (R/X)CEREA to change the channel configuration for partition A. Several features of the McBSP helps to time the reassignment:

- The block of channels currently involved in reception/transmission (the current block) is reflected in the RCBLK/XCBLK bits. Your program can poll these bits to determine which partition is active. When a partition is not active, it is safe to change its block assignment and channel configuration.
- At the end of every block (at the boundary of two partitions), an interrupt can be sent to the CPU. In response to the interrupt, the CPU can then check the RCBLK/XCBLK bits and update the inactive partition.

Figure 8-38 shows an example of reassigning channels throughout a data transfer. In response to a frame-sync pulse, the McBSP alternates between partitions A and B. Whenever partition B is active, the CPU changes the block assignment for partition A. Whenever partition A is active, the CPU changes the block assignment for partition B.

Figure 8-38. Reassigning Channel Blocks Throughout a McBSP Data Transfer



8.2.8.6 Using Eight Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use eight partitions or two partitions. If you choose the 8-partition mode (RMCME = 1 for reception, XMCME = 1 for transmission), McBSP partitions are activated in the following order: A, B, C, D, E, F, G, H. In response to a frame-sync pulse, the receiver or transmitter begins with the channels in partition A and then continues with the other partitions in order until the complete frame has been transferred. When the next frame-sync pulse occurs, the next frame is transferred, beginning with the channels in partition A. In the 8-partition mode, the (R/X)PABLK and (R/X)PBBLK bits are ignored and the 16-channel blocks are assigned to the partitions as shown in Table 8-15 and Table 8-16. These assignments cannot be changed. Table 8-15 and Table 8-16 also show the registers used to control the channels in the partitions.

Table 8-15. Receive Channel Assignment and Control When Eight Receive Partitions are Used

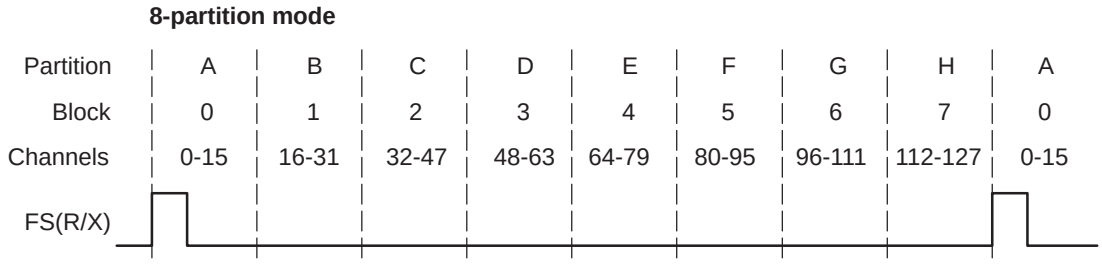
Receive Partition	Assigned Block of Receive Channels	Register Used For Channel Control
A	Block 0: channels 0 through 15	RCEREA
B	Block 1: channels 16 through 31	RCEREB
C	Block 2: channels 32 through 47	RCEREC
D	Block 3: channels 48 through 63	RCERED
E	Block 4: channels 64 through 79	RCEREE
F	Block 5: channels 80 through 95	RCEREF
G	Block 6: channels 96 through 111	RCEREG
H	Block 7: channels 112 through 127	RCEREH

Table 8-16. Transmit Channel Assignment and Control When Eight Transmit Partitions are Used

Transmit Partition	Assigned Block of Transmit Channels	Register Used For Channel Control
A	Block 0: channels 0 through 15	XCEREA
B	Block 1: channels 16 through 31	XCEREB
C	Block 2: channels 32 through 47	XCEREC
D	Block 3: channels 48 through 63	XCERED
E	Block 4: channels 64 through 79	XCEREE
F	Block 5: channels 80 through 95	XCEREF
G	Block 6: channels 96 through 111	XCEREG
H	Block 7: channels 112 through 127	XCEREH

Figure 8-39 shows an example of the McBSP using the 8-partition mode. In response to a frame-sync pulse, the McBSP begins a frame transfer with partition A and then activates B, C, D, E, F, G, and H to complete a 128-word frame.

Figure 8-39. McBSP Data Transfer in the 8-Partition Mode



8.2.8.7 Receive Multichannel Selection Mode

The RMCM bit in the multichannel control register (MCR) determines whether all channels or only selected channels are enabled for reception. When RMCM = 0, all 128 receive channels are enabled and cannot be disabled. When RMCM = 1, the receive multichannel selection mode is enabled. In this mode:

- Channels can be individually enabled or disabled. The only channels enabled are those selected in the appropriate enhanced receive channel enable register (RCERE*n*). The way channels are assigned to the RCERE*n* depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit in MCR.
- If a receive channel is disabled, any bits received in that channel are passed only as far as the receive buffer register (RBR). The receiver does not copy the content of the RBR to the DRR, and as a result, does not set the receiver ready bit (RRDY). Therefore, no DMA synchronization event (REVT) is generated, and if the receiver interrupt mode depends on RRDY (RINTM = 0), no interrupt is generated.

As an example of how the McBSP behaves in the receive multichannel selection mode, suppose you enable only channels 0, 15, and 39 and that the frame length is 40. The McBSP:

1. Accepts bits shifted in from the DR pin in channel 0.
2. Ignores bits received in channels 1-14.
3. Accepts bits shifted in from the DR pin in channel 15.
4. Ignores bits received in channels 16-38.
5. Accepts bits shifted in from the DR pin in channel 39.

8.2.8.8 Transmit Multichannel Selection Mode

The XMCM bit in the multichannel control register (MCR) determines whether all channels or only selected channels are enabled and unmasked for transmission. The McBSP has three transmit multichannel selection modes (XMCM = 1, XMCM = 2h, and XMCM = 3h), which are described in Table 8-17.

Table 8-17. Selecting a Transmit Multichannel Selection Mode With the XMCM Bits

XMCM Bit in MCR	Transmit Multichannel Selection Mode
0	No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked.
1h	All channels are disabled unless they are selected in the appropriate enhanced transmit channel enable register (XCERE*n*). If enabled, a channel in this mode is also unmasked.
	The XMCME bit in MCR determines whether 32 channels or 128 channels are selectable in XCERE*n*.
2h	All channels are enabled, but they are masked unless they are selected in the appropriate enhanced transmit channel enable register (XCERE*n*).
	The XMCME bit in MCR determines whether 32 channels or 128 channels are selectable in XCERE*n*.
3h	This mode is used for symmetric transmission and reception. All channels are disabled for transmission unless they are enabled for reception in the appropriate enhanced receive channel enable register (RCERE*n*). Once enabled, they are masked unless they are also selected in the appropriate enhanced transmit channel enable register (XCERE*n*).
	The XMCME bit in MCR determines whether 32 channels or 128 channels are selectable in RCERE*n* and XCERE*n*.

As an example of how the McBSP behaves in a transmit multichannel selection mode, suppose that XMCM = 1 (all channels disabled unless individually enabled) and that you have enabled only channels 0, 15, and 39. Suppose also that the frame length is 40. TheMcBSP:

1. Shifts data to the DX pin in channel 0.
2. Places the DX pin in the high-impedance state in channels 1–14.
3. Shifts data to the DX pin in channel 15.
4. Places the DX pin in the high-impedance state in channels 16–38.
5. Shifts data to the DX pin in channel 39.

8.2.8.8.1 Disabling/Enabling Versus Masking/Unmasking

For transmission, a channel can be:

- Enabled and unmasked (transmission can begin and can be completed)
- Enabled but masked (transmission can begin but cannot be completed)
- Disabled (transmission cannot occur)

The following definitions explain the channel control options:

Enabled channel	A channel that can begin transmission by passing data from the data transmit register (DXR) to the transmit shift register (XSR).
Masked channel	A channel that cannot complete transmission. The DX pin is held in the high-impedance state; data cannot be shifted out on the DX pin. In systems where symmetric transmit and receive provides software benefits, this feature allows transmit channels to be disabled on a shared serial bus. A similar feature is not needed for reception because multiple receptions cannot cause serial bus contention.
Disabled channel	A channel that is not enabled. A disabled channel is also masked. Because no DXR-to-XSR copy occurs, the XRDY bit in SPCR is not set. Therefore, no DMA synchronization event (XEVT) is generated, and if the transmit interrupt mode depends on XRDY (XINTM = 0 in SPCR), no interrupt is generated. The XEMPTY bit in SPCR is not affected.
Unmasked channel	A channel that is not masked. Data in the XSR is shifted out on the DX pin.

8.2.8.8.2 Activity on McBSP Pins for Different Values of XMCM

Figure 8-40 shows the activity on the McBSP pins for the various XMCM values. In all cases, the transmit frame is configured as follows:

- In transmit control register (XCR):
  - XPHASE = 0: Single-phase frame (required for multichannel selection modes)
  - XFRLEN1 = 3h: 4 words per frame
  - XWDLEN1 = 0: 8 bits per word
- In multichannel control register (MCR):
  - XMCME = 0: 2-partition mode (only partitions A and B used)

In the case where XMCM = 3h, transmission and reception are symmetric, which means the corresponding bits for the receiver (RPHASE, RFRLEN1, RWDLEN1, and RMCME) must have the same values as XPHASE, XFRLEN1, and XWDLEN1, respectively.

In Figure 8-40, the arrows showing where the various events occur are only sample indications. Wherever possible, there is a time window in which these events can occur.

8.2.8.9 Using Interrupts Between Block Transfers

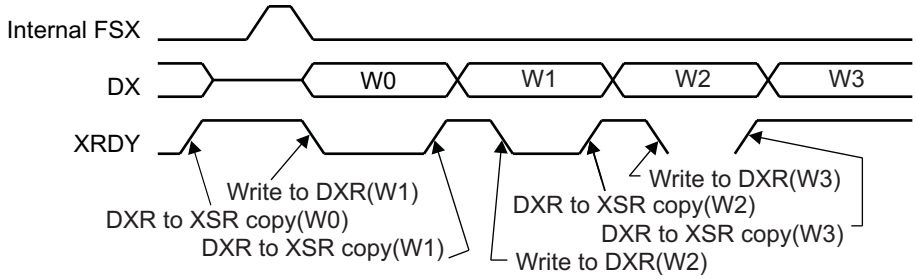
When a multichannel selection mode is used, an interrupt request can be sent to the CPU at the end of every 16-channel block (at the boundary between partitions and at the end of the frame). In the receive multichannel selection mode, a receive interrupt (RINT) request is generated at the end of each block transfer if the RINTM bit in the serial port control register (SPCR) is set to 1. In any of the transmit multichannel selection modes, a transmit interrupt (XINT) request is generated at the end of each block transfer if the XINTM bit in SPCR is set to 1. When RINTM/XINTM = 1, no interrupt is generated unless a multichannel selection mode is on.

These interrupt pulses are active high and last for two McBSP internal input clock cycles.

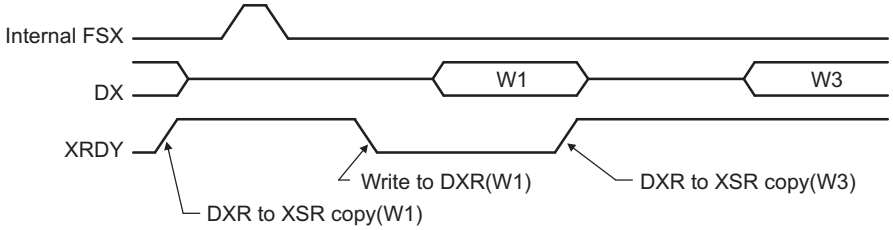
This type of interrupt is especially helpful if you are using the two-partition mode and you want to know when you can assign a different block of channels to partition A or B.

Figure 8-40. Activity on McBSP Pins for the Possible Values of XMCM

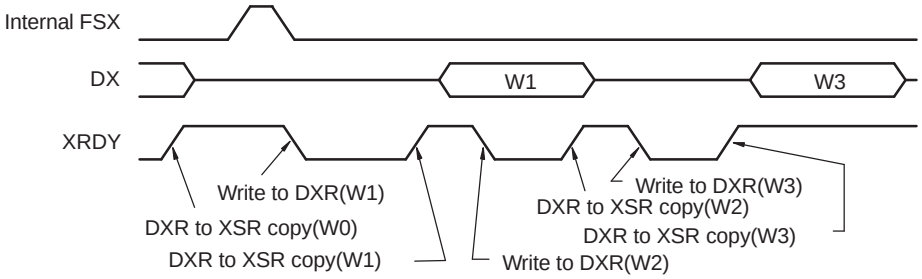
(a) XMCM = 0: All channels enabled and unmasked



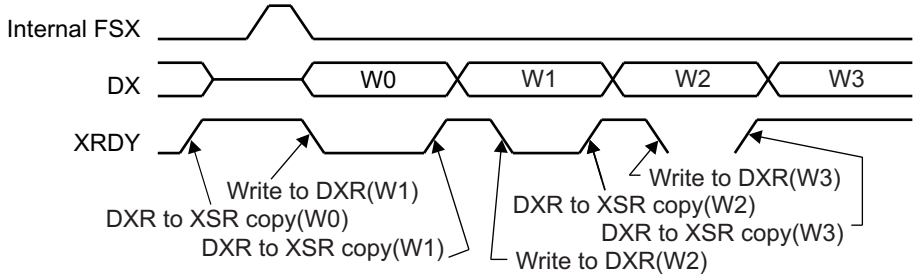
(b) XMCM = 1h, XPABLK = 0, XCEREA = 000Ah: Only channels 1 and 3 enabled and unmasked



(c) XMCM = 2h, XPABLK = 0, XCEREA = 000Ah: All channels enabled, only 1 and 3 unmasked



(d) XMCM = 3h, RPABLK = 0, XPABLK = x, RCEREA = 0008h, XCEREA = 000Ah: Receive channels: 1 and 3 enabled; transmit channels: 1 and 3 enabled, but only 3 unmasked



8.2.9 SPI Operation Using the Clock Stop Mode

The McBSP on this device does not support the SPI protocol.

8.2.10 Resetting the Serial Port: RRST, XRST, GRST, and RESET

Device reset or McBSP reset: When the McBSP is reset by device reset or McBSP reset, the state machine is reset to its initial state. All counters and status bits are reset. This includes the receive status bits RFULL, RRDY, and RSYNCERR, and the transmit status bits XEMPTY, XRDY, and XSYNCERR in the serial port control register (SPCR).

The serial port can be reset in the following two ways:

- Device reset (RESET pin is low) places the receiver, the transmitter, and the sample rate generator in reset. When the device reset is removed (RESET = 1), FRST = GRST = RRST = XRST = 0 in SPCR, keeping the entire serial port in the reset state.
- The serial port transmitter and receiver can be independently reset by the XRST and RRST bits in SPCR. The sample rate generator is reset by the GRST bit in SPCR.

Table 8-18 shows the state of the McBSP pins when the serial port is reset by these methods.

Table 8-18. Reset State of McBSP Pins

Pin	Direction	Device Reset ($\overline{\text{RESET}}$ = 0)	McBSP Reset
			Receiver Reset (RRST = 0 and GRST = 1)
DR	I	Input	Input
CLKR	I/O/Z	Input	Known state if input; CLKR if output
FSR	I/O/Z	Input	Known state if input; FSRP(inactive state) if output
CLKS	I	Input	Input
			Transmitter Reset (XRST = 0 and GRST = 1)
DX	O/Z	High impedance	High impedance
CLKX	I/O/Z	Input	Known state if input; CLKX if output
FSX	I/O/Z	Input	Known state if input; FSXP(inactive state) if output
CLKS	I	Input	Input

8.2.10.1 Software Reset Considerations

McBSP reset: When the receiver and transmitter reset bits, RRST and XRST in SPCR, are written with 0, the respective portions of the McBSP are reset and activity in the corresponding section stops. All input-only pins, such as DR, and all other pins that are configured as inputs are in a known state. FS(R/X) is driven to its inactive state (same as its polarity bit, FS(R/X)P) if it is an output. If CLK(R/X) are programmed as outputs, they are driven by CLKG, provided that GRST = 1. The DX pin is in the high-impedance state when the transmitter is reset. During normal operation, the sample rate generator can be reset by writing a 0 to GRST. The sample rate generator should only be reset when not being used by the transmitter or the receiver. In this case, the internal sample rate generator clock CLKG, and its frame sync signal (FSG) is driven inactive (low). When the sample rate generator is not in the reset state (GRST = 1), FSR and FSX are in an inactive state when RRST = 0 and XRST = 0, respectively, even if they are outputs driven by FSG. This ensures that when only one portion of the McBSP is in reset, the other portion can continue operation when FRST = 1 and frame sync is driven by FSG.

Sample-rate generator reset: As mentioned previously, the sample rate generator is reset when the device is reset or when its reset bit, GRST in SPCR, is written with 0.

Emulator software reset: In the event of an emulator software reset initiated from the DSP, the McBSP register values are reset to their default values.

8.2.10.2 Hardware Reset Considerations

When the McBSP is reset due to device reset, the entire serial port (including the transmitter, receiver, and the sample rate generator) is reset. All input-only pins and 3-state pins should be in a known state. The output-only pin, DX, is in the high-impedance state. When the device is pulled out of reset, the serial port remains in the reset condition (RRST = XRST = FRST = GRST = 0 in SPCR).

8.2.11 McBSP Initialization Procedure

The McBSP initialization procedure varies depending on the specific system setup. Section 8.2.11.1 provides a general initialization sequence.

The transmitter and the receiver of the McBSP can operate independently from each other. Therefore, they can be placed in or taken out of reset individually by modifying only the desired bit in the registers without disrupting the other portion. The steps in the following sections discuss the initialization procedure for taking both the transmitter and the receiver out of reset. To initialize only one portion, configure only the portion desired.

The McBSP internal sample rate generator and internal frame sync generator are shared between the transmitter and the receiver. Table 8-19 and Table 8-20 describe their usage base upon the clock and frame sync configurations of the receiver and transmitter, respectively.

Table 8-19. Receiver Clock and Frame Configurations

CLKR Source	FSR Source	Comment on Configuration
Internal	Internal	The McBSP internal sample rate generator and internal frame sync generator are used by the receiver.
External	Internal	The McBSP internal sample rate generator is not used, but the internal frame sync generator is used by the receiver.
Internal	External	The McBSP internal sample rate generator is used but the internal frame sync generator is not used by the receiver.
External	External	The McBSP internal sample rate generator and internal frame sync generator are not used by the receiver.

Table 8-20. Transmitter Clock and Frame Configurations

CLKX Source	FSX Source	Comment on Configuration
Internal	Internal	The McBSP internal sample rate generator is used by the transmitter. The transmitter can generate frame sync FSX in one of two ways. First, it can generate FSX by using the internal frame sync generator (FSGM = 1). Alternatively, it can generate FSX upon each DXR-to-XSR copy (FSGM = 0). In this case, the internal frame sync generator can be kept in reset (FRST = 0) if it is not used by the receiver. You can follow the general initialization sequence in Section 8.2.11.1.
External	Internal	The McBSP internal sample rate generator and internal frame sync generator are not used by the transmitter. This configuration is only valid with FSGM = 0 where the McBSP transmitter generates FSX upon each DXR-to-XSR copy. You can follow the general initialization sequence in Section 8.2.11.1.
Internal	External	The McBSP internal sample rate generator is used by the transmitter but the internal frame sync generator is not.
External	External	The McBSP internal sample rate generator and internal frame sync generator are not used by the transmitter.

8.2.11.1 General Initialization Procedure

This section provides the general initialization procedure.

1. With the McBSP still disabled (Peripheral Clock Gating Control Register 1 (PCGCR1) in the disabled state):
   (a) Program the PCGCR1 register in the System Module to put the McBSP in the enable state (see your device-specific *System Reference Guide*).
   (b) Perform the necessary device pin multiplexing setup (see your device-specific data manual).
2. Ensure that no portion of the McBSP is using the internal sample rate generator signal CLKG and the internal frame sync generator signal FSG (GRST = FRST = 0 in SPCR). The respective portion of the McBSP needs to be in reset (XRST = 0 and/or RRST = 0 in SPCR).
3. Program the control registers as required. Ensure the internal sample rate generator and the internal frame sync generator are still in reset (GRST = FRST = 0). Also ensure the respective portion of the McBSP is still in reset in this step (XRST = 0 and/or RRST = 0).
4. Wait for proper internal synchronization. If the external device provides the bit clock, wait for two CLKR or CLKX cycles. If the McBSP generates the bit clock as a clock master, wait for two CLKSRG cycles. In this case, the clock source to the sample rate generator (CLKSRG) is selected by the CLKSM bit in SRGR and the SCLKME bit in PCR.
5. Skip this step if the bit clock is provided by the external device. This step only applies if the McBSP is the bit clock master and the internal sample rate generator is used.
   (a) Start the sample rate generator by setting the GRST bit to 1. Wait two CLKG bit clocks for synchronization. CLKG is the output of the sample rate generator.
   (b) On the next rising edge of CLKSRG, CLKG transitions to 1 and starts clocking with a frequency equal to 1/(CLKGDV + 1) of the sample rate generator source clock CLKSRG.
6. Skip this step if the transmitter is not used. If the transmitter is used, a transmit sync error (XSYNCERR) may occur when it is enabled for the first time after device reset. The purpose of this step is to clear any potential XSYNCERR that occurs on the transmitter at this time:
   (a) Set the XRST bit to 1 to enable the transmitter.
   (b) Wait for any unexpected frame sync error to occur. If the external device provides the bit clock, wait for two CLKR or CLKX cycles. If the McBSP generates the bit clock as a clock master, wait for two CLKG cycles. The unexpected frame sync error (XSYNCERR), if any, occurs within this time period.
   (c) Disable the transmitter (XRST = 0). This clears any outstanding XSYNCERR.
7. Setup data acquisition as required:
   (a) If the DMA is used to service the McBSP, setup data acquisition as desired and start the DMA in this step, before the McBSP is taken out of reset.
   (b) If CPU interrupt is used to service the McBSP, enable the transmit and/or receive interrupt as required.
   (c) If CPU polling is used to service the McBSP, no action is required in this step.
8. Set the XRST bit and/or the RRST bit to 1 to enable the corresponding section of the McBSP The McBSP is now ready to transmit and/or receive.
   (a) If the DMA is used to service the McBSP, it services the McBSP automatically upon receiving the XEVT and/or REVT.
   (b) If CPU interrupt is used to service the McBSP, the interrupt service routine is automatically entered upon receiving the XINT and/or RINT.
   (c) If CPU polling is used to service the McBSP, it can do so now by polling the XRDY and/or RRDY bit.
9. If the internal frame sync generator is used (FSGM = 1), proceed to the additional steps to turn on the internal frame sync generator. Initialization is complete if any one of the following is true:
   (a) The external device generates frame sync FSX and/or FSR. The McBSP is now ready to transmit and/or receive upon receiving external frame sync.
   (b) The McBSP generates transmit frame sync FSX upon each DXR-to-XSR copy. The internal frame sync generator is not used (FSGM = 0).

The following additional steps to turn on the internal frame sync generator apply only if FSGM = 1:

10. Skip this step if the transmitter is not used. If the transmitter is used, ensure that DXR is serviced before you start the internal frame sync generator. You can do so by checking XEMPTY = 1 (XSR is not empty) in SPCR.
11. Set the FRST bit to 1 to start the internal frame sync generator. The internal frame sync signal FSG is generated on a CLKG active edge after 7 to 8 CLKG clocks have elapsed.

8.2.11.2 Special Case: External Device is the Transmit Frame Master

Care must be taken if the transmitter expects a frame sync from an external device. After the transmitter comes out of reset (XRST = 1), it waits for a frame sync from the external device. If the first frame sync arrives very shortly after the transmitter is enabled, the CPU or DMA controller may not have a chance to service the data transmit register (DXR). In this case, the transmitter shifts out the default data in the transmit shift register (XSR) instead of the desired value, which has not yet arrived in DXR. This causes problems in some applications, as the first data element in the frame is invalid. The data stream appears element-shifted (the first data word may appear in the second channel instead of the first).

To ensure proper operation when the external device is the frame master, you must assure that DXR is already serviced with the first word when a frame sync occurs. To do so, you can keep the transmitter in reset until the first frame sync is detected. Software is setup such that it will only take the transmitter out of reset (XRST = 1) promptly after detecting the first frame sync. This assures that the transmitter does not begin data transfers at the data pin during the first frame sync period. This also provides almost an entire frame period for the CPU to service DXR with the first word before the second frame sync occurs. The transmitter only begins data transfers upon receiving the second frame sync. At this point, DXR is already serviced with the first word.

8.2.11.2.1 How to Detect First Frame Sync

Although the McBSP is capable of generating an interrupt to the CPU upon the detection of frame synchronization (XINTM = 2h and/or RINTM = 2h in the serial port control register (SPCR)), the McBSP requires the associated portion (receiver/transmitter) of the McBSP to be out of reset in order for the interrupt to be generated. Therefore, instead of directly using the McBSP interrupt to detect the first frame sync, you can use the GPIO peripheral. This can be achieved by connecting the frame sync signal to a GPIO pin. Software can either poll the GPIO pin to detect the first frame sync or program the GPIO peripheral to generate an interrupt to the CPU upon detecting the first frame sync edge. For more information on the GPIO peripheral, see the GPIO chapter.

The following are some recommended GPIO pin(s) on the device that you can use to detect the first McBSP external frame sync:

- **GPIO pin located near the McBSP pins.** Connect the external frame sync to both the McBSP FSX/FSR pin(s) and the dedicated GPIO pin.
- **GPIO pin multiplexed with the McBSP FSX signal.** Note that on the device, the GPIO pins (of the GPIO peripheral) are multiplexed with the McBSP pins. Software can program the device's pin multiplexing register (PINMUX) to default these pins to the GPIO function, and only switch them to the McBSP function upon detecting the first frame sync. This method is only recommended if the external device is both the frame sync and clock master; that is, the external device drives both the FSX and CLKX signals. This method is not recommended if the McBSP is the clock master (driving CLKX and/or CLKR), as the “on-the-fly” pin multiplexed switching can cause a glitch on the CLKX/CLKR pin. For more details on pin multiplexing, see the device-specific data manual.

8.2.11.2.2 Initialization Procedure When External Device is Frame Sync Master

The initialization procedure assumes the following:

- Using a GPIO pin multiplexed with the McBSP FSX signal. If a dedicated GPIO pin is used instead, skip step 1 and step 8b.
- Software polls the GPIO pin to detect the first frame sync. If the GPIO interrupt is used instead to detect the first frame sync, step 8 can be performed within an interrupt service routine (ISR).

1. The GPIO and McBSP signals are multiplexed together on the device. Start by programming the external bus selection register (EBSR) to select the GPIO function on the GPIO/McBSP multliplexed pins. Program the GPIO peripheral so that these pins function as GPIO inputs.
2. Ensure that no portion of the McBSP is using the internal sample rate generator signal CLKG and the internal frame sync generator signal FSG (GRST = FRST = 0 in SPCR). The respective portion of the McBSP needs to be in reset (XRST = 0 and/or RRST = 0 in SPCR).
3. Program the sample rate generator register (SRGR) and other control registers as required. Ensure the internal sample rate generator and the internal frame sync generator are still in reset (GRST = FRST = 0 in SPCR). Also ensure the respective portion of the McBSP is still in reset in this step (XRST = 0 and/or RRST = 0 in SPCR).
4. Wait for proper McBSP internal synchronization:
   (a) If the external device provides the bit clock, wait for two CLKR or CLKX cycles. Skip step 5.
   (b) If the McBSP generates the bit clock as a clock master, wait for two CLKSRG cycles. In this case, the clock source to the sample rate generator (CLKSRG) is selected by the CLKSM bit in SRGR.
5. Skip this step if the bit clock is provided by the external device. This step only applies if the McBSP is the bit clock master and the internal sample rate generator is used.
   (a) Start the sample rate generator by setting the GRST bit in SPCR to 1. Wait two CLKG bit clocks for synchronization. CLKG is the output of the sample rate generator.
   (b) On the next rising edge of CLKSRG, CLKG transitions to 1 and starts clocking with a frequency equal to 1/(CLKGDV + 1) of the sample rate generator source clock CLKSRG.
6. A transmit sync error (XSYNCERR) may occur when it is enabled for the first time after device reset. The purpose of this step is to clear any potential XSYNCERR that occurs on the transmitter at this time:
   (a) Set the XRST bit in SPCR to 1 to enable the transmitter.
   (b) Wait for any unexpected frame sync error to occur. If the external device provides the bit clock, wait for two CLKR or CLKX cycles. If the McBSP generates the bit clock as a clock master, wait for two CLKG cycles. The unexpected frame sync error (XSYNCERR), if any, occurs within this time period.
   (c) Disable the transmitter (XRST = 0). This clears any outstanding XSYNCERR.
7. Setup data acquisition as required:
   (a) If the DMA controller is used to service the McBSP, setup data acquisition as desired and start the DMA controller in this step, before the McBSP is taken out of reset.
   (b) If the CPU interrupt is used to service the McBSP, no action is required in this step.
   (c) If CPU polling is used to service the McBSP, no action is required in this step.
8. Poll the GPIO pin (through reading the appropriate registers in the GPIO peripheral) to detect the first transmit frame sync from the external device. Upon detection of the first frame sync, perform the following in this order:
   (a) Set the XRST bit and/or the RRST bit to 1 to enable the respective portion of the McBSP. The McBSP is now ready to transmit and/or receive.
   (b) Program PINMUX to switch the GPIO/McBSP multiplexed pins to the McBSP function.
9. Service the McBSP:
   (a) If CPU polling is used to service the McBSP in normal operations, it can do so upon exit from the ISR.
   (b) If the CPU interrupt is used to service the McBSP in normal operations, upon XRDY interrupt service routine is entered. The ISR should be setup to verify that XRDY = 1 and service the McBSP accordingly.
   (c) If the DMA controller is used to service the McBSP in normal operations, it services the McBSP automatically upon receiving the XEVT and/or REVT.
10. Upon detection of the second frame sync, DXR is already serviced and the transmitter is ready to transmit the valid data. The receiver is also serviced properly by the CPU.

8.2.12 Interrupt Support

The McBSP can send both receive and transmit interrupts to the DSP controller.

8.2.12.1 Interrupt Events and Requests

The RRDY and XRDY bits in the serial port control register (SPCR) indicate the ready state of the McBSP receiver and transmitter, respectively. Writes and reads from the serial port can be synchronized by any of the following methods:

- Polling RRDY and XRDY bits in SPCR
- Using the events sent to the DMA controller (REVT and XEVT)
- Using the interrupts to the CPU (RINT and XINT) that the events generate

Reading DRR and writing to DXR affects RRDY and XRDY, respectively.

8.2.12.1.1 Interrupt Events: RINT and XINT

The receive interrupt (RINT) and transmit interrupt (XINT) signals inform the CPU of changes to the serial port status. Three options exist for configuring these interrupts. These options are set by the receive/transmit interrupt mode bits (RINTM and XINTM) in SPCR. The possible values of the mode, and the configurations they represent, are:

- (R/X)INTM = 00b: Interrupt on every serial element by tracking the (R/X)RDY bits in SPCR.
- (R/X)INTM = 01b: Interrupt at the end of a subframe (16 elements or less) within a frame. See Section 8.2.8.9 for more details.
- (R/X)INTM = 10b: Interrupt on detection of frame synchronization pulses. The associated portion (receiver/transmitter) of the McBSP must be out of reset.
- (R/X)INTM = 11b: Interrupt on frame synchronization error. Note that if any of the other interrupt modes are selected, (R/X)SYNCERR may be read when servicing the interrupts to detect this condition. See Section 8.2.6.5.2 and Section 8.2.6.5.5 for more details on synchronization error.

8.2.12.1.2 Receive Ready Status: RINT and RRDY

RRDY = 1 indicates that the RBR contents have been copied to DRR and that the data can now be read by either the CPU or the DMA controller. Once that data has been read by either the CPU, RRDY is cleared to 0. Also, at device reset or serial port receiver reset (RRST = 0), the RRDY bit is cleared to 0 to indicate that no data has been received and loaded into DRR. RRDY directly drives the McBSP receive interrupt (RINT) to the CPU if RINTM = 00b (default value) in SPCR.

8.2.12.1.3 Transmit Ready Status: XINT and XRDY

XRDY = 1 indicates that the DXR contents have been copied to XSR and that DXR is ready to be loaded with a new data word. When the transmitter transitions from reset to non-reset (XRST transitions from 0 to 1), XRDY also transitions from 0 to 1 indicating that DXR is ready for new data. Once new data is loaded by the CPU, the XRDY bit is cleared to 0. However, once this data is copied from DXR to XSR, the XRDY bit transitions again from 0 to 1. The CPU can write to DXR although XSR has not yet been shifted out on DX. XRDY directly drives the McBSP transmit interrupt (XINT) to the CPU if XINTM = 00b (default value) in SPCR.

NOTE: If the polling method is used to service the transmitter, the CPU should wait for one McBSP bit clock (CLKX) before polling again to write the next element in DXR. This is because XRDY transitions occur based on bit clock and not CPU clock. The CPU clock is much faster and can cause false XRDY status, leading to data errors due to overwrites.

8.2.12.2 Interrupt Multiplexing

The RINT and XINT interrupts generated by the McBSP peripheral to the CPU are multiplexed with other interrupt sources. For more detailed information, see Section 1.6, *Interrupts*.

8.2.13 DMA Event Support

8.2.13.1 Receive Ready Status: REVT and RRDY

RRDY = 1 in the serial port control register (SPCR) indicates that the RBR contents have been copied to DRR and that the data can now be read by the DMA controller. Once that data has been read by the DMA controller, RRDY is cleared to 0. Also, at device reset or serial port receiver reset (RRST = 0 in SPCR), the RRDY bit is cleared to 0 to indicate that no data has been received and loaded into DRR. RRDY directly drives the McBSP receive event to the DMA controller (via REVT).

For detailed information on using the DMA to read or write to the McBSP, see Chapter 3, *DMA Controller*.

8.2.13.2 Transmit Ready Status: XEVT and XRDY

XRDY = 1 in the serial port control register (SPCR) indicates that the DXR contents have been copied to XSR and that DXR is ready to be loaded with a new data word. When the transmitter transitions from reset to non-reset (XRST transitions from 0 to 1 in SPCR), XRDY also transitions from 0 to 1 indicating that DXR is ready for new data. Once new data is loaded by the DMA controller, the XRDY bit is cleared to 0. However, once this data is copied from DXR to XSR, the XRDY bit transitions again from 0 to 1. The DMA controller can write to DXR although XSR has not yet been shifted out on DX. XRDY directly drives the transmit synchronization event to the DMA controller (via XEVT).

For detailed information on using the DMA to read or write to the McBSP, see Chapter 3, *DMA Controller*.

NOTE: If the polling method is used to service the transmitter, the CPU should wait for one McBSP bit clock (CLKX) before polling again to write the next element in DXR. This is because XRDY transitions occur based on bit clock and not CPU clock. The CPU clock is much faster and can cause false XRDY status, leading to data errors due to overwrites.

8.2.14 Power Management

The McBSP can be placed in reduced power modes to conserve power during periods of low activity. The power management of the peripheral is controlled by the Peripheral Clock Gating Controller (PCGCR1). The PCGCR1 acts as a master controller for power management for all of the peripherals on the device.

In order for the McBSP to be placed in power-down mode by the PCGCR1, ensure that the XRDY and RRDY flags in the serial port control register (SPCR) are cleared by performing the following steps:

1. Place the McBSP in reset by clearing the XRST, RRST, FRST, and GRST bits to 0 in SPCR.

   If DMA is being used to service the transmitter and/or the receiver, disable the associated DMA channels.

   For detailed information on using the DMA to read or write to the McBSP, see Chapter 3, *DMA Controller*.
2. Switch the McBSP clocks and frames to internal clock source:

   (a) Set the CLKSM and FSGM bits to 1 in the sample rate generator register (SRGR).

   (b) Set the CLKXM, CLKRM, FSXM, and FSRM bits to 1 in the pin control register (PCR).

   (c) Clear the SCLKME bit to 0 in PCR.
3. Bring the McBSP out of reset by setting the XRST, RRST, and GRST bits to 1 in SPCR.
4. Wait for two CLKSRG cycles for proper internal synchronization.
5. Write a dummy data value to the data transmit register (DXR) in order to clear the first XRDY flag.
6. Wait for at least one McBSP bit clock, since once the first dummy data value is internally copied from DXR to XSR, the XRDY flag transitions again from 0 to 1.
7. Write a second dummy data value to DXR in order to clear the second XRDY flag.
8. Check the RRDY flag in SPCR and if set to 1, read the data receive register (DRR) and discard the data to clear the RRDY flag.
9. Place the McBSP in power-down mode by issuing the proper PCGCR1 commands. For detailed information on power management procedures using the PCGCR1, see Chapter 1, *System Control*.

NOTE: After waking up the McBSP from a power-down mode using the proper PCGCR1 commands, remember to reconfigure the SPCR, SRGR, and PCR registers to the clock and frame combination that they were in before entering the power-down sequence and discard the two dummy data values that were used to clear the XRDY flags. If DMA is used, re-enable the corresponding DMA channels.

8.2.15 IDLE Logic Module

IDLE Activity Logic Module is used to decide when to send out the IDLE ready (CLKSTOP_ACK) signal. Once the IDLE request (CLKSTOP_REQ) is asserted, it will assert CLKSTOP_ACK based on the McBSP activities. Active McBSP internal activity (DMA or CPU transfers) can delay the assertion of CLKSTOP_ACK. This is done to prevent any active McBSP access from terminating due to the MCBSP being in IDLE.

8.2.16 IDLE Control Signals

8.2.16.1 CLKSTOP_REQ and CLKSTOP_ACK

When IDLE request (CLKSTOP_REQ) is coming to McBSP, McBSP asserts an IDLE ready signal (CLKSTOP_ACK) to the on-chip clock gating module in the following conditions.

- If McBSP has no activity (Vbus transfer, MMR access), immediately disable the internal clock and return an IDLE ready signal (CLKSTOP_ACK). When CLKSTOP_REQ (IDLE request) of McBSP is asserted, McBSP should return CLKSTOP_ACK (IDLE ready) immediately if McBSP is in software reset state.
- If McBSP has activity (Vbus transfer, Frame Transfer, MMR access), return an IDLE ready signal (CLKSTOP_ACK) after the current activity is finished.

8.2.16.2 McBSP Idle Mode

The McBSP can be put into IDLE mode by IDLE_IN signal.

In the McBSP idle mode:

- If the McBSP is configured to operate with internally generated clocking and frame synchronization, it stop completely after the current transfer activity is complete and CLKSTOP_ACK is sent out.
- If the McBSP is configured to operate with externally generated clocking and frame synchronization (either directly or through the sample rate generator), the external interface portion of the McBSP continues to function during external clock activity periods. Once the current transfer activities are complete and full frame is serviced, McBSP will send out the CLKSTOP_ACK.

8.2.16.3 McBSP Idle Wakeup Process

8.2.16.3.1 Internal Wakeup

The CPU reactivates the peripheral domain and DMA domain. Then the CPU de-asserts CLKSTOP_REQ. Chip level logic will unblock the module interface clock to wake up the McBSP module.

8.2.16.3.2 External Wakeup

The McBSP is configured to operate with externally generated clocking and frame synchronization. The frame synchronization can come externally or through the sample rate generator. The external interface portion of the McBSP will continue to function during external clock activity periods.

8.2.16.3.2.1 After Frame Synchronization

1. **Transmit.** Once XSR is empty, an internal wakeup signal is asserted (high), which turns on the clock for the McBSP and DMA domains and de-asserts the MBPCLKSTPREQ bit in the CLKSTOP1 register. After McBSP and DMA wake up, the DXR to XSR copy triggers a new data write to DXR by DMA. A complete DMA write to DXR de-asserts the internal wakeup signal and MBPCLKSTPREQ goes high again. Then, the chip level turns off the clocks of the McBSP and DMA modules and McBSP goes idle again.
2. **Receive.** Once RBR is full, an internal wakeup signal is asserted (high), which turns on the clock for the McBSP and DMA domains and de-asserts the MBPCLKSTPREQ bit in the CLKSTOP1 register. After McBSP and DMA wake up, the RBR to DRR copy triggers a new data read from DRR by DMA. A complete DRR read by DMA de-asserts the internal wakeup signal and MBPCLKSTPREQ goes high again. Then, the chip level turns off the clocks of the McBSP and DMA modules and McBSP goes idle again.

8.2.17 Emulation Considerations

The FREE and SOFT bits are special emulation bits in the serial port control register (SPCR) that determine the state of the McBSP when an emulation suspend event occurs in the emulator. An emulation suspend event corresponds to any type of emulator access to the CPU, such as a hardware or software breakpoint, a probe point, or a printf instruction.

Table 8-21 shows the effects of the FREE and SOFT bits on the response of the McBSP to emulation suspend events.

Table 8-21. McBSP Emulation Modes Selectable With the FREE and SOFT Bits of SPCR

FREE Bit in SPCR	SOFT Bit in SPCR	McBSP Emulation Mode
0	0	Immediate stop mode (reset condition). The transmitter and receiver stop immediately in response to an emulation suspend event.
0	1	Soft stop mode. When an emulation suspend event occurs, the transmitter stops after completion of the current word. The receiver is not affected.
1	0 or 1	Free run mode. The transmitter and receiver continue to run when an emulation suspend event occurs.

8.3 MCBSP Registers

Table 8-22 lists the memory-mapped registers for the MCBSP. All register offset addresses not listed in Table 8-22 should be considered as reserved locations and the register contents should not be modified.

Table 8-22. MCBSP REGISTERS

CPU Word Address	Acronym	Register Name	Section
4000h	DRRL	Data Receive Register Lower	Section 8.3.1
4001h	DRRU	Data Receive Register Upper	Section 8.3.2
4004h	DXRL	Data Transmit Register Lower	Section 8.3.3
4005h	DXRU	Data Transmit Register Upper	Section 8.3.4
4008h	SPCRL	Serial Port Control Register Lower	Section 8.3.5
4009h	SPCRU	Serial Port Control Register Upper	Section 8.3.6
400Ch	RCRL	Receive Control Register Lower	Section 8.3.7
400Dh	RCRU	Receive Control Register Upper	Section 8.3.8
4010h	XCRL	Transmit Control Register Lower	Section 8.3.9
4011h	XCRU	Transmit Control Register Upper	Section 8.3.10
4014h	SRGRL	Sample Rate Generator Register Lower	Section 8.3.11
4015h	SRGRU	Sample Rate Generator Register Upper	Section 8.3.12
4018h	MCRL	Multichannel Control Register Lower	Section 8.3.13
4019h	MCRU	Multichannel Control Register Upper	Section 8.3.14
401Ch	RCERA	Enhanced Receive Channel Enable Register Partition A	Section 8.3.15
401Dh	RCERB	Enhanced Receive Channel Enable Register Partition B	Section 8.3.16
4020h	XCERA	Enhanced Transmit Channel Enable Register Partition A	Section 8.3.17
4021h	XCERB	Enhanced Transmit Channel Enable Register Partition B	Section 8.3.18
4024h	PCRL	Pin Control Register Lower	Section 8.3.19
4025h	PCRU	Pin Control Register Upper	Section 8.3.20
4028h	RCERC	Enhanced Receive Channel Enable Register Partition C	Section 8.3.21
4029h	RCERD	Enhanced Receive Channel Enable Register Partition D	Section 8.3.22
402Ch	XCERC	Enhanced Transmit Channel Enable Register Partition C	Section 8.3.23
402Dh	XCERD	Enhanced Transmit Channel Enable Register Partition D	Section 8.3.24
4030h	RCERE	Enhanced Receive Channel Enable Register Partition E	Section 8.3.25
4031h	RCERF	Enhanced Receive Channel Enable Register Partition F	Section 8.3.26
4034h	XCERE	Enhanced Transmit Channel Enable Register Partition E	Section 8.3.27
4035h	XCERF	Enhanced Transmit Channel Enable Register Partition F	Section 8.3.28
4038h	RCERG	Enhanced Receive Channel Enable Register Partition G	Section 8.3.29
4039h	RCERH	Enhanced Receive Channel Enable Register Partition H	Section 8.3.30
403Ch	XCERG	Enhanced Transmit Channel Enable Register Partition G	Section 8.3.31
403Dh	XCERH	Enhanced Transmit Channel Enable Register Partition H	Section 8.3.32

8.3.1 DRRL Register (offset = 4000h) [reset = 0h]

DRRL is shown in Figure 8-41 and described in Table 8-23.

Contains the Lower value received on the data bus.

Figure 8-41. DRRL Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-23. DRRL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DR	R	0h	Data receiver register value loaded from the RBR. Value is 0000 to FFFFh.

8.3.2 DRRU Register (offset = 4001h) [reset = 0h]

DRRU is shown in Figure 8-42 and described in Table 8-24.

Contains the upper value received on the data bus.

Figure 8-42. DRRU Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-24. DRRU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DR	R	0h	Data receiver register value loaded from the RBR. Value is 0000 to FFFFh.

8.3.3 DXRL Register (offset = 4004h) [reset = 0h]

DXRL is shown in Figure 8-43 and described in Table 8-25.

Contains the lower value to be loaded into the data transmit shift register.

Figure 8-43. DXRL Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-25. DXRL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DX	R/W	0h	Data transmit register value to be loaded into the data transmit shift register XSR. Value is 0000 to FFFFh.

8.3.4 DXRU Register (offset = 4005h) [reset = 0h]

DXRU is shown in Figure 8-44 and described in Table 8-26.

contains the upper value to be loaded into the data transmit shift register.

Figure 8-44. DXRU Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-26. DXRU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DX	R/W	0h	Data transmit register value to be loaded into the data transmit shift register XSR. Value is 0000 to FFFFh.

8.3.5 SPCRL Register (offset = 4008h) [reset = 0h]

SPCRL is shown in Figure 8-45 and described in Table 8-27.

This register contains configuration bits for McBSP desired operation.

Figure 8-45. SPCRL Register

15	14	13	12	11	10	9	8
DLB	RJUST		Reserved				
R/W-0h	R/W-0h		R-0h				

7	6	5	4	3	2	1	0
DXENA	Reserved	RINTM		RSYNCERR	RFULL	RRDY	RRST
R/W-0h	R-0h	R/W-0h		R/W-0h	R-0h	R-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-27. SPCRL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	DLB	R/W	0h	Digital Loop Back Mode enable bit. 0x0 = Digital loop back disabled 0x1 = Digital loop back enabled
14-13	RJUST	R/W	0h	Receive Sign-Extension and Justification Mode bit. 0x0 = right-justify and zero-fill MSBs in DRR 0x1 = right-justify and sign-extend MSBs in DRR 0x2 = left-justify and zero-fill LSBs in DRR 0x3 = Reserved
12-11	CLKSTP	R/W	0h	Clock stop mode select bit. 0x0 = Clock stop mode disabled. Even with combination 0'b01 it will be disabled. 0x2 = Clock starts with rising/falling edge without delay. 0x3 = Clock starts with rising/falling edge with delay.
10-8	Reserved	R	0h	
7	DXENA	R/W	0h	DX Enabler bit. 0x0 = DX enabler is off 0x1 = DX enabler is on
6	ABIS	R	0h	ABIS mode select bit. 0x0 = Disabled. 0x1 = Enabled.
5-4	RINTM	R/W	0h	Receive Interrupt mode selection bits. 0x0 = Receive interrupt generated by RRDY 0x1 = Receive interrupt generated by EOF. 0x2 = Receive interrupt generated by new frame synchronization (NEWFS) 0x3 = Receive interrupt generated by SYNCERR
3	RSYNCERR	R/W	0h	Receive Synchronization Error bit. 0x0 = NOSYNCERR 0x1 = SYNCERR
2	RFULL	R	0h	Receive Shift Register (RSR) Full status. 0x0 = RBR is not in overrun condition 0x1 = RBR is in overrun condition
1	RRDY	R	0h	Receiver Ready status bit. 0x0 = Receiver is busy 0x1 = Receiver is ready

Table 8-27. SPCRL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	RRST	R/W	0h	Receiver reset bit. 0x0 = Receiver in reset 0x1 = Receiver enabled

8.3.6 SPCRU Register (offset = 4009h) [reset = 0h]

SPCRU is shown in Figure 8-46 and described in Table 8-28.

This register contains configuration bits for McBSP desired operation.

Figure 8-46. SPCRU Register

15	14	13	12	11	10	9	8
Reserved						FREE	SOFT
R-0h						R/W-0h	R/W-0h

7	6	5	4	3	2	1	0
FRST	GRST	XINTM		XSYNCERR	XEMPTY	XRDY	XRST
R/W-0h	R/W-0h	R/W-0h		R/W-0h	R-0h	R-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-28. SPCRU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	
9	FREE	R/W	0h	Free running mode enable bit. 0x0 = Free running mode disabled 0x1 = Free running mode enabled
8	SOFT	R/W	0h	Soft mode select bit. 0x0 = SOFT mode is disabled 0x1 = SOFT mode is enabled
7	FRST	R/W	0h	Frame Sync Generator Reset bit. 0x0 = Frame Synchronization logic is reset 0x1 = Frame Synchronization logic is active
6	GRST	R/W	0h	Sample Rate Generator Reset bit. 0x0 = Sample rate generator in reset 0x1 = Sample rate generator out of reset
5-4	XINTM	R/W	0h	Transmit Interrupt mode selection bits. 0x0 = Transmit interrupt generated by XRDY 0x1 = Transmit interrupt generated by EOF. 0x2 = Transmit interrupt generated by new frame synchronization (NEWFS) 0x3 = Transmit interrupt generated by SYNCERR
3	XSYNCERR	R/W	0h	Transmit Synchronization Error bit. 0x0 = No sync error detected by McBSP 0x1 = SYNCERR
2	XEMPTY	R	0h	Transmit Shift Register (XSR) Empty status bit. 0x0 = XSR is empty 0x1 = XSR is not empty
1	XRDY	R	0h	Transmitter Ready status bit. 0x0 = Transmitter is busy 0x1 = Transmitter is ready
0	XRST	R/W	0h	Transmitter reset bit. 0x0 = Transmitter in reset 0x1 = Transmitter enabled

8.3.7 RCRL Register (offset = 400Ch) [reset = 0h]

RCRL is shown in Figure 8-47 and described in Table 8-29.

Configure various parameters of receive operation.

Figure 8-47. RCRL Register

15	14	13	12	11	10	9	8
Reserved	RFRLEN1						
R-0h	R/W-0h						

7	6	5	4	3	2	1	0
RWDLEN1			RWDREVRS	Reserved			
R/W-0h			R/W-0h	R-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-29. RCRL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	
14-8	RFRLEN1	R/W	0h	Specifies the receive frame length (number of words) in phase1.
7-5	RWDLEN1	R/W	0h	Specifies the receive word length (number of bits) in phase 1. 0x0 = Receive word length 8-bits 0x1 = Receive word length 12-bits 0x2 = Receive word length 16-bits 0x3 = Receive word length 20-bits 0x4 = Receive word length 24-bits 0x5 = Receive word length 32-bits 0x6 = Reserved 0x7 = Reserved
4	RWDREVRS	R/W	0h	Receive 32-bit reversal feature select bit. 0x0 = 32-bit reversal disabled 0x1 = 32-bit reversal enabled
3-0	Reserved	R	0h	

8.3.8 RCRU Register (offset = 400Dh) [reset = 0h]

RCRU is shown in Figure 8-48 and described in Table 8-30.

Configure various parameters of receive operation.

Figure 8-48. RCRU Register

15	14	13	12	11	10	9	8
RPHASE	RFRLEN2						
R/W-0h	R/W-0h						

7	6	5	4	3	2	1	0
RWDLEN2			RCOMPAND		RFIG	RDATDLY	
R/W-0h			R/W-0h		R/W-0h	R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-30. RCRU Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RPHASE	R/W	0h	Receive Phase selection bit. 0x0 = Single phase frame selected 0x1 = Dual phase frame selected
14-8	RFRLEN2	R/W	0h	Specifies the receive frame length (number of words) in phase 2.
7-5	RWDLEN2	R/W	0h	Specifies the receive word length (number of bits) in phase 2. 0x0 = Receive word length 8bits 0x1 = Receive word length 12bits 0x2 = Receive word length 16bits 0x3 = Receive word length 20bits 0x4 = Receive word length 24bits 0x5 = Receive word length 32bits 0x6 = Reserved 0x7 = Reserved
4-3	RCOMPAND	R/W	0h	Receive Companding Mode bits. 0x0 = Receiver no companding, data transfer starts with MSB first 0x1 = Receiver no companding, 8-bit data, transfer starts with LSB first 0x2 = Receiver compand using u-law for receive data 0x3 = Receiver compand using A-law for receive data
2	RFIG	R/W	0h	Receive Frame Ignore bit. 0x0 = Receive Frame ignore disabled 0x1 = Receive Frame ignore enabled
1-0	RDATDLY	R/W	0h	Receive data delay bits. 0x0 = 0-bit data delay 0x1 = 1-bit data delay 0x2 = 2-bit data delay 0x3 = Reserved

8.3.9 XCRL Register (offset = 4010h) [reset = 0h]

XCRL is shown in Figure 8-49 and described in Table 8-31.

Configure various parameters of transmit operation.

Figure 8-49. XCRL Register

15	14	13	12	11	10	9	8
Reserved	XFRLEN1						
R-0h	R/W-0h						

7	6	5	4	3	2	1	0
XWDLEN1			XWDREVRS	Reserved			
R/W-0h			R/W-0h	R-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-31. XCRL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	
14-8	XFRLEN1	R/W	0h	Transmit Frame Length 1.
7-5	XWDLEN1	R/W	0h	Transmit Word Length 1 select bits. 0x0 = Transmit word length 8bits 0x1 = Transmit word length 12bits 0x2 = Transmit word length 16bits 0x3 = Transmit word length 20bits 0x4 = Transmit word length 24bits 0x5 = Transmit word length 32bits 0x6 = Reserved 0x7 = Reserved
4	XWDREVRS	R/W	0h	Transmit 32-bit reversal feature select bit. 0x0 = 32-bit reversal disabled 0x1 = 32-bit reversal enabled
3-0	Reserved	R	0h	

8.3.10 XCRU Register (offset = 4011h) [reset = 0h]

XCRU is shown in Figure 8-50 and described in Table 8-32.

Configure various parameters of transmit operation.

Figure 8-50. XCRU Register

15	14	13	12	11	10	9	8
XPHASE	XFRLEN2						
R/W-0h	R/W-0h						

7	6	5	4	3	2	1	0
XWDLEN2			XCOMPAND		XFIG	XDATDLY	
R/W-0h			R/W-0h		R/W-0h	R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-32. XCRU Register Field Descriptions

Bit	Field	Type	Reset	Description
15	XPHASE	R/W	0h	Transmit Phase selection bit. 0x0 = Single phase frame selected 0x1 = Dual phase frame selected
14-8	XFRLEN2	R/W	0h	Specifies the transmit frame length (number of words) in phase 2.
7-5	XWDLEN2	R/W	0h	Specifies the transmit word length (number of bits) in phase 2. 0x0 = Transmit word length 8bits 0x1 = Transmit word length 12bits 0x2 = Transmit word length 16bits 0x3 = Transmit word length 20bits 0x4 = Transmit word length 24bits 0x5 = Transmit word length 32bits 0x6 = Reserved 0x7 = Reserved
4-3	XCOMPAND	R/W	0h	Transmit Companding Mode bits. Modes other than 0 are only enabled when XWDLEN1/2 bit is 000b (Indicating 8-bit data) 0x0 = Transmit no companding, data transfer starts with MSB first 0x1 = Transmit no companding, 8-bit data, transfer starts with LSB first 0x2 = Transmit compand using u-law for transmit data 0x3 = Transmit compand using A-law for transmit data
2	XFIG	R/W	0h	Transmit Frame Ignore bit. 0x0 = Transmit Frame ignore disabled 0x1 = Transmit Frame ignore enabled
1-0	XDATDLY	R/W	0h	Transmit data delay bits. 0x0 = 0-bit data delay 0x1 = 1-bit data delay 0x2 = 2-bit data delay 0x3 = Reserved

8.3.11 SRGRL Register (offset = 4014h) [reset = 1h]

SRGRL is shown in Figure 8-51 and described in Table 8-33.

Controls the operation of various features of the sample rate generator.

Figure 8-51. SRGRL Register

15	14	13	12	11	10	9	8
FWID							
R/W-0h							
7	6	5	4	3	2	1	0
CLKGDV							
R/W-1h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-33. SRGRL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	FWID	R/W	0h	Frame width value plus 1 specifies the width of the frame-sync pulse (FSG) during its active period.
7-0	CLKGDV	R/W	1h	Sample-rate generator clock divider value is used as the divide-down number to generate the required sample rate generator clock frequency.

8.3.12 SRGRU Register (offset = 4015h) [reset = 2000h]

SRGRU is shown in Figure 8-52 and described in Table 8-34.

Controls the operation of various features of the sample rate generator.

Figure 8-52. SRGRU Register

15	14	13	12	11	10	9	8
GSYNC	CLKSP	CLKSM	FSGM	FPER			
R/W-0h	R/W-0h	R/W-1h	R/W-0h	R/W-0h			

7	6	5	4	3	2	1	0
FPER							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-34. SRGRU Register Field Descriptions

Bit	Field	Type	Reset	Description
15	GSYNC	R/W	0h	Sample-rate generator clock synchronization bit 0x0 = the sample rate generator clock (CLKG) is free running. 0x1 = The sample rate generator clock (CLKG) is running
14	CLKSP	R/W	0h	CLKS polarity clock edge select bit 0x0 = rising edge of CLKS generates CLKG and FSG 0x1 = falling edge of CLKS generates CLKG and FSG.
13	CLKSM	R/W	1h	Sample rate generator input clock mode select bits. 0x0 = Sample rate generator clock derived from the CLKS pin depending on SCLKME 0x1 = Sample rate generator clock derived from CPU clock depending on SCLKME
12	FSGM	R/W	0h	Sample-rate generator transmit frame-synchronization mode selection bits 0x0 = Transmit frame sync signal (FSX) due to DXR-to-XSR copy 0x1 = Transmit frame sync signal driven by the sample rate generator frame sync signal, FSG
11-0	FPER	R/W	0h	Frame period value plus 1 specifies when the next frame-sync signal becomes active. Range is 1 to 4096 sample-rate generator clock (CLKG) periods.

8.3.13 MCRL Register (offset = 4018h) [reset = 0h]

MCRL is shown in Figure 8-53 and described in Table 8-35.

Control and status bits for multichannel operation.

Figure 8-53. MCRL Register

15	14	13	12	11	10	9	8
Reserved					REMODE	RMCME	RPBBLK
R-0h					R/W-0h	R/W-0h	R/W-0h

7	6	5	4	3	2	1	0
RPBBLK	RPABLK		RCBLK			Reserved	RMCM
R/W-0h	R/W-0h		R-0h			R-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-35. MCRL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	
10	REMODE	R/W	0h	Receive multi-channel mode selection bit. 0x0 = Traditional/32/128 channel operation 0x1 = 512 channel mode with PDMA
9	RMCME	R/W	0h	Enhanced receive multichannel selection enable bits 0x0 = Maximum 32 channels can be enabled at one time 0x1 = Maximum 128 channels can be enabled at one time
8-7	RPBBLK	R/W	0h	Receive Partition B Block bits 0x0 = Block 1-Channel 16 to channel 31 0x1 = Block 3-Channel 48 to channel 63 0x2 = Block 5-Channel 80 to channel 95 0x3 = Block 7-Channel 112 to channel 127
6-5	RPABLK	R/W	0h	Receive Partition A Block bits. 0x0 = Block 0-Channel 0 to channel 15 0x1 = Block 2-Channel 32 to channel 47 0x2 = Block 4-Channel 64 to channel 79 0x3 = Block 6-Channel 96 to channel 111
4-2	RCBLK	R	0h	Receive Current Block status bits 0x0 = Block 0. Channel 0 to channel 15 0x1 = Block 1. Channel 16 to channel 31 0x2 = Block 2. Channel 32 to channel 47 0x3 = Block 3. Channel 48 to channel 63 0x4 = Block 4. Channel 64 to channel 79 0x5 = Block 5. Channel 80 to channel 95 0x6 = Block 6. Channel 96 to channel 111 0x7 = Block 7. Channel 112 to channel 127
1	Reserved	R	0h	
0	RMCM	R/W	0h	Receive Multi-channel Selection Enable bits 0x0 = all 128 channels enabled 0x1 = all channels disabled by default. Required channels are selected by enabling RP(A/B)BLK and RCER(A/B) appropriately

8.3.14 MCRU Register (offset = 4019h) [reset = 0h]

MCRU is shown in Figure 8-54 and described in Table 8-36.

Control and status bits for multichannel operation.

Figure 8-54. MCRU Register

15	14	13	12	11	10	9	8
Reserved			DX		XEMODE	XMCME	XPBBLK
R-0h			R/W-0h		R/W-0h	R/W-0h	R/W-0h

7	6	5	4	3	2	1	0
XPBBLK	XPABLK		XCBLK			XMCM	
R/W-0h	R/W-0h		R-0h			R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 8-36. MCRU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	
12-11	DX	R/W	0h	Transmit pin mode configuration. This is used to configure the DX pin state while the time slot is disabled. 0x0 = Hi-Z 0x1 = Driven as 1. 0x2 = Driven as 0.
10	XEMODE	R/W	0h	Transmit multi-channel mode selection bit. 0x0 = Traditional/32/128 channel operation. 0x1 = 512 channel mode with PDMA.
9	XMCME	R/W	0h	Enhanced transmit multichannel selection enable bits 0x0 = Maximum 32 channels can be enabled at one time 0x1 = Maximum 128 channels can be enabled at one time
8-7	XPBBLK	R/W	0h	Transmit Partition B Block bits 0x0 = Block 1-Channel 16 to channel 31 0x1 = Block 3-Channel 48 to channel 63 0x2 = Block 5-Channel 80 to channel 95 0x3 = Block 7-Channel 112 to channel 127
6-5	XPABLK	R/W	0h	Transmit Partition A Block bits. 0x0 = Block 0-Channel 0 to channel 15 0x1 = Block 2-Channel 32 to channel 47 0x2 = Block 4-Channel 64 to channel 79 0x3 = Block 6-Channel 96 to channel 111
4-2	XCBLK	R	0h	Transmit Current Block indicator bits 0x0 = Block 0. Channel 0 to channel 15 0x1 = Block 1. Channel 16 to channel 31 0x2 = Block 2. Channel 32 to channel 47 0x3 = Block 3. Channel 48 to channel 63 0x4 = Block 4. Channel 64 to channel 79 0x5 = Block 5. Channel 80 to channel 95 0x6 = Block 6. Channel 96 to channel 111 0x7 = Block 7. Channel 112 to channel 127

Table 8-36. MCRU Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1-0	XMCM	R/W	0h	Transmit Multi-channel Selection mode bit. XMCM determines whether all channel or only selected channel are enabled and unmasked for transmission 0x0 = Transmit multichannel selection is off. All channels are enabled and unmasked. No channels can be disabled or masked. 0x1 = All channels are disabled unless they are selected in the appropriate enhanced transmit channel enable register(XCEREn). If enabled a channel in this mode is also unmasked. The XMCME bit determines whether 32 channels or 128 channels are selectable in XCEREn, 0x2 = All channels are enabled, but they are masked unless they are selected in the appropriate enhanced transmit channel enable register (XCEREn). The XMCME bit determines whether 32 channels or 128 channels are selectable in XCEREn, 0x3 = This mode is used for symmetric transmission and reception. All channels are disabled for transmission unless they are enabled for reception in the appropriate enhanced receive channel enable register (RCEREn). Once enabled, they are masked unless they are also selected in the appropriate enhanced transmit channel enable register (XCEREn). The XMCME bit determines whether 32 channels or 128 channels are selectable in RCEREn and XCEREn.

8.3.15 RCERA Register (offset = 401Ch) [reset = 0h]

RCERA is shown in Figure 8-55 and described in Table 8-37.

Enhanced Receive Channel Enable Partition A

Figure 8-55. RCERA Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-37. RCERA Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RCEA	R/W	0h	0x0 = Reception of channel 15 to 0 in an even-numbered block in partition A is disabled 0x1 = Reception of channel 15 to 0 in an even-numbered block in partition A is enabled

8.3.16 RCERB Register (offset = 401Dh) [reset = 0h]

RCERB is shown in Figure 8-56 and described in Table 8-38.

Enhanced Receive Channel Enable Partition B

Figure 8-56. RCERB Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-38. RCERB Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RCEB	R/W	0h	0x0 = Reception of channel 15 to 0 in an odd-numbered block in partition B is disabled 0x1 = Reception of channel 15 to 0 in an odd-numbered block in partition B is enabled

8.3.17 XCERA Register (offset = 4020h) [reset = 0h]

XCERA is shown in Figure 8-57 and described in Table 8-39.

Enhanced Receive Channel Enable Register Partition A

Figure 8-57. XCERA Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-39. XCERA Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	XCEA	R/W	0h	0x0 = Transmission of channel 15 to 0 in an even-numbered block in partition A is disabled 0x1 = Transmission of channel 15 to 0 in an even-numbered block in partition A is enabled

8.3.18 XCERB Register (offset = 4021h) [reset = 0h]

XCERB is shown in Figure 8-58 and described in Table 8-40.

Enhanced Receive Channel Enable Register Partition B

Figure 8-58. XCERB Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-40. XCERB Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	XCEB	R/W	0h	0x0 = Transmission of channel 15 to 0 in an odd-numbered block in partition B is disabled 0x1 = Transmission of channel 15 to 0 in an odd-numbered block in partition B is enabled

8.3.19 PCRL Register (offset = 4024h) [reset = 0h]

PCRL is shown in Figure 8-59 and described in Table 8-41.

Configures serial port pin as general-purpose input or output.

Figure 8-59. PCRL Register

15	14	13	12	11	10	9	8
Reserved	IDLE_EN	XIOEN	RIOEN	FSXM	FSRM	CLKXM	CLKRM
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
SCLKME	CLKS_STAT	DX_STAT	DR_STAT	FSXP	FSRP	CLKXP	CLKRP
R/W-0h	R-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-41. PCRL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	
14	IDLE_EN	R/W	0h	Idle control bit for McBSP. 0x0 = McBSP is running. 0x1 = McBSP is idle.
13	XIOEN	R/W	0h	Transmit general-purpose I/O mode-only select bit. 0x0 = Not GPIO. DR,CLKS are not general purpose inputs, DX is not general purpose output and FS(R/X), CLK(R/X) are not general purpose I/Os. 0x1 = GPIO. DR,CLKS are general purpose inputs, DX is general purpose output and FS(R/X), CLK(R/X) are general purpose I/Os.
12	RIOEN	R/W	0h	Receive general-purpose I/O mode-only select bit. 0x0 = Not GPIO. DR,CLKS are not general purpose inputs, DX is not general purpose output and FS(R/X), CLK(R/X) are not general purpose I/Os. 0x1 = GPIO. DR,CLKS are general purpose inputs, DX is general purpose output and FS(R/X), CLK(R/X) are general purpose I/Os.
11	FSXM	R/W	0h	Transmit Frame Synchronization Mode 0x0 = Frame synchronization signal derived from an external source 0x1 = Frame synchronization is determined by the Sample Rate Generator frame synchronization mode bit FSGM in the SRGR
10	FSRM	R/W	0h	Receive Frame Synchronization Mode 0x0 = Frame synchronization pulses generated by an external device. FSR is an input pin 0x1 = Frame synchronization generated internally by sample rate generator. FSR is an output pin.
9	CLKXM	R/W	0h	Transmitter Clock Mode bit. 0x0 = CLKX is an input pin and is driven by an external clock 0x1 = CLKX is an output pin and is driven by the internal sample-rate generator
8	CLKRM	R/W	0h	Receiver Clock Mode bit. 0x0 = CLKR is an input pin and is driven by an external clock / Receive clock (not the CLKR pin) is driven by transmit clock (CLKX). 0x1 = CLKR is an output pin and is driven by the internal sample-rate generator or CLKR is an output pin and is driven by the transmit clock.
7	SCLKME	R/W	0h	Enhanced sample clock mode selection bit. 0x0 = Signal on CLKS or internal clock 0x1 = Signal on CLKR/CLKX pin

Table 8-41. PCRL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	CLKS_STAT	R	0h	CLKS pin status bit. 0x0 = CLKS pin is at logic low. 0x1 = CLKS pin is at logic high.
5	DX_STAT	R/W	0h	DX pin status bit. 0x0 = DX pin is at logic low. 0x1 = DX pin is at logic high.
4	DR_STAT	R	0h	DR pin status bit. 0x0 = DR pin is at logic low. 0x1 = DR pin is at logic high.
3	FSXP	R/W	0h	Transmit Frame-Synchronization Polarity bit 0x0 = Transmit frame synchronization pulse FSX is active high 0x1 = Transmit frame synchronization pulse FSX is active low
2	FSRP	R/W	0h	Receive Frame-Synchronization Polarity bit 0x0 = Receive frame synchronization pulse FSR is active high 0x1 = Receive frame synchronization pulse FSR is active low
1	CLKXP	R/W	0h	Transmit Clock Polarity bit 0x0 = Transmit data sampled on rising edge of CLKX 0x1 = Transmit data sampled on falling edge of CLKX
0	CLKRP	R/W	0h	Receive Clock Polarity bit 0x0 = Receive data sampled on falling edge of CLKR 0x1 = Receive data sampled on rising edge of CLKR

8.3.20 PCRU Register (offset = 4025h) [reset = 0h]

PCRU is shown in Figure 8-60 and described in Table 8-42.

Configures serial port pin as general-purpose input or output.

Figure 8-60. PCRU Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-42. PCRU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	Reserved	R	0h	

8.3.21 RCERC Register (offset = 4028h) [reset = 0h]

RCERC is shown in Figure 8-61 and described in Table 8-43.

Enhanced Receive Channel Enable Register

Figure 8-61. RCERC Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-43. RCERC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RCEC	R/W	0h	0x0 = Reception of channel 15 to 0 in an even-numbered block in partition C is disabled 0x1 = Reception of channel 15 to 0 in an even-numbered block in partition C is enabled

8.3.22 RCERD Register (offset = 4029h) [reset = 0h]

RCERD is shown in Figure 8-62 and described in Table 8-44.

Enhanced Receive Channel Enable Register

Figure 8-62. RCERD Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-44. RCERD Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RCED	R/W	0h	0x0 = Reception of channel 15 to 0 in an odd-numbered block in partition D is disabled 0x1 = Reception of channel 15 to 0 in an odd-numbered block in partition D is enabled

8.3.23 XCERC Register (offset = 402Ch) [reset = 0h]

XCERC is shown in Figure 8-63 and described in Table 8-45.

Enhanced Transmit Channel Enable Register

Figure 8-63. XCERC Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-45. XCERC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	XCEC	R/W	0h	0x0 = Transmission of channel 15 to 0 in an even-numbered block in partition C is disabled 0x1 = Transmission of channel 15 to 0 in an even-numbered block in partition C is enabled

8.3.24 XCERD Register (offset = 402Dh) [reset = 0h]

XCERD is shown in Figure 8-64 and described in Table 8-46.

Enhanced Transmit Channel Enable Register

Figure 8-64. XCERD Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-46. XCERD Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	XCED	R/W	0h	0x0 = Transmission of channel 15 to 0 in an odd-numbered block in partition D is disabled 0x1 = Transmission of channel 15 to 0 in an odd-numbered block in partition D is enabled

8.3.25 RCERE Register (offset = 4030h) [reset = 0h]

RCERE is shown in Figure 8-65 and described in Table 8-47.

Enhanced Receive Channel Enable Register Partition E

Figure 8-65. RCERE Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-47. RCERE Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RCEE	R/W	0h	0x0 = Reception of channel 15 to 0 in an even-numbered block in partition E is disabled 0x1 = Reception of channel 15 to 0 in an even-numbered block in partition E is enabled

8.3.26 RCERF Register (offset = 4031h) [reset = 0h]

RCERF is shown in Figure 8-66 and described in Table 8-48.

Enhanced Receive Channel Enable Register Partition F

Figure 8-66. RCERF Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-48. RCERF Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RCEF	R/W	0h	0x0 = Reception of channel 15 to 0 in an odd-numbered block in partition F is disabled 0x1 = Reception of channel 15 to 0 in an odd-numbered block in partition F is enabled

8.3.27 XCERE Register (offset = 4034h) [reset = 0h]

XCERE is shown in Figure 8-67 and described in Table 8-49.

Enhanced Transmit Channel Enable Register Partition E

Figure 8-67. XCERE Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-49. XCERE Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	XCEE	R/W	0h	0x0 = Transmission of channel 15 to 0 in an even-numbered block in partition E is disabled 0x1 = Transmission of channel 15 to 0 in an even-numbered block in partition E is enabled

8.3.28 XCERF Register (offset = 4035h) [reset = 0h]

XCERF is shown in Figure 8-68 and described in Table 8-50.

Enhanced Transmit Channel Enable Register Partition F

Figure 8-68. XCERF Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-50. XCERF Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	XCEF	R/W	0h	0x0 = Transmission of channel 15 to 0 in an odd-numbered block in partition F is disabled 0x1 = Transmission of channel 15 to 0 in an odd-numbered block in partition F is enabled

8.3.29 RCERG Register (offset = 4038h) [reset = 0h]

RCERG is shown in Figure 8-69 and described in Table 8-51.

Figure 8-69. RCERG Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-51. RCERG Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RCEG	R/W	0h	0x0 = Reception of channel 15 to 0 in an even-numbered block in partition G is disabled 0x1 = Reception of channel 15 to 0 in an even-numbered block in partition G is enabled

8.3.30 RCERH Register (offset = 4039h) [reset = 0h]

RCERH is shown in Figure 8-70 and described in Table 8-52.

Figure 8-70. RCERH Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-52. RCERH Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RCEH	R/W	0h	0x0 = Reception of channel 15 to 0 in an odd-numbered block in partition H is disabled 0x1 = Reception of channel 15 to 0 in an odd-numbered block in partition H is enabled

8.3.31 XCERG Register (offset = 403Ch) [reset = 0h]

XCERG is shown in Figure 8-71 and described in Table 8-53.

Enhanced Transmit Channel Enable Register Partition G

Figure 8-71. XCERG Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-53. XCERG Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	XCEG	R/W	0h	0x0 = Transmission of channel 15 to 0 in an even-numbered block in partition G is disabled 0x1 = Transmission of channel 15 to 0 in an even-numbered block in partition G is enabled

8.3.32 XCERH Register (offset = 403Dh) [reset = 0h]

XCERH is shown in Figure 8-72 and described in Table 8-54.

Enhanced Transmit Channel Enable Register Partition H

Figure 8-72. XCERH Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 8-54. XCERH Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	XCEH	R/W	0h	0x0 = Transmission of channel 15 to 0 in an odd-numbered block in partition H is disabled 0x1 = Transmission of channel 15 to 0 in an odd-numbered block in partition H is enabled

Multichannel Serial Port Interface (McSPI)

Topic **Page**

9.1 Introduction

The multichannel serial port interface (McSPI) is a master/slave synchronous serial bus. The McSPI module supports up to three peripherals.

McSPI includes the following main features:

- Serial clock with programmable frequency, polarity, and phase for each channel
- Wide selection of McSPI word lengths ranging from 4–32 bits
- Up to three master channels or single channel in slave mode
- Master multichannel mode:
  - Full duplex/half duplex
  - Transmit-only/receive-only/transmit-and-receive modes
  - Flexible I/O port controls per channel
  - Two direct memory access (DMA) requests (read/write) per channel
- Single interrupt line for multiple interrupt source events
- Power management through wake-up capabilities
- Enable the addition of a programmable start-bit for McSPI transfer per channel (start-bit mode)
- Support start-bit write command
- Support start-bit pause and break sequence
- 128 bytes built-in FIFO available for a single channel
- Force CS mode for continuous transfers

NOTE: In this chapter, *x*, such as in CH*x*CONF, represents the channel in signal and register naming.

x = 0, 1, or 2

9.2 McSPI Environment

9.2.1 McSPI Interface in Master Mode

Figure 9-1 shows a case in master mode (full-duplex) where the McSPI module is connected with two slave devices.

Figure 9-1. McSPI Master Mode (Full-Duplex)

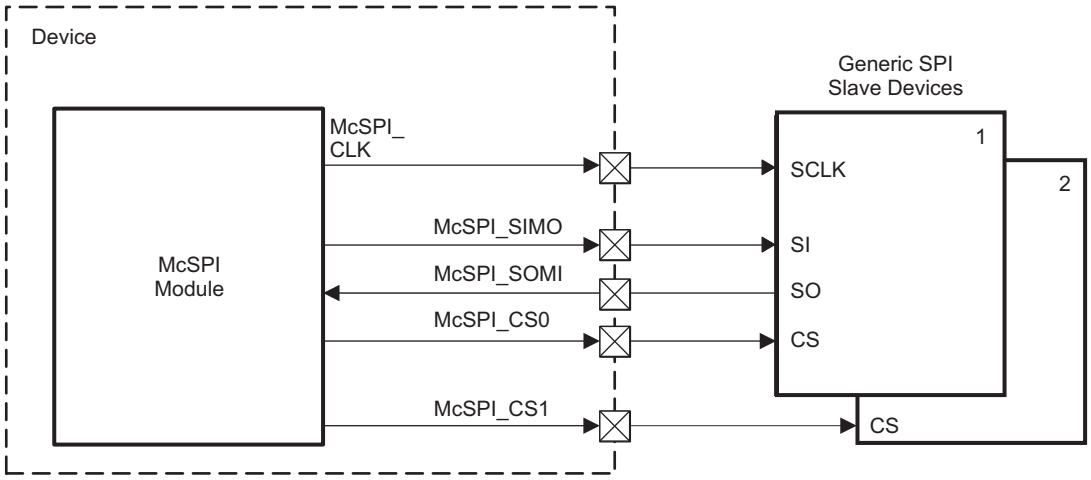
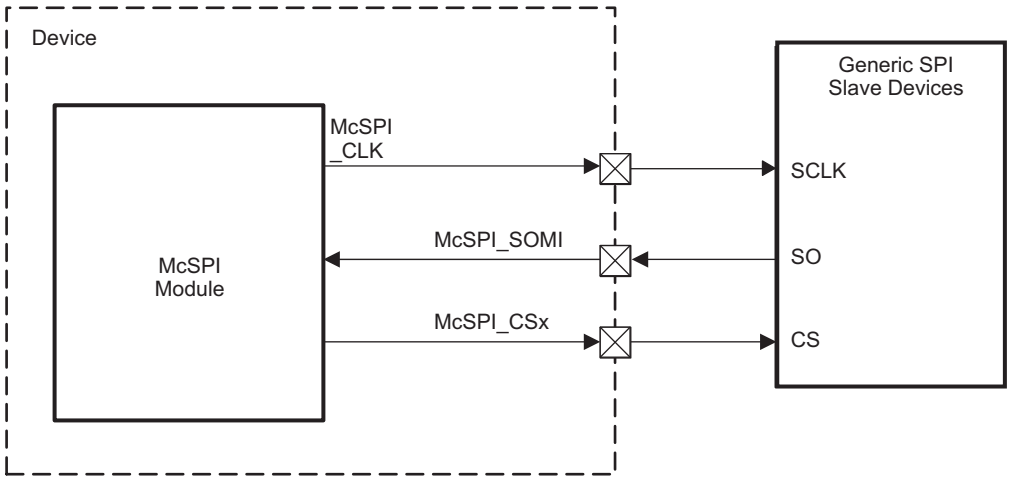


Figure 9-2 shows the master single mode, which can also be configured in receive-only mode.

Figure 9-2. McSPI Master Single Mode (Receive-Only)



9.2.2 McSPI Interface in Slave Mode

Figure 9-3 shows a case in slave mode (full-duplex).

NOTE: Only Channel 0 can be configured as slave, and only McSPI_CS0 can be used as a chip-enable in slave mode.

Figure 9-3. McSPI Slave Mode (Full Duplex)

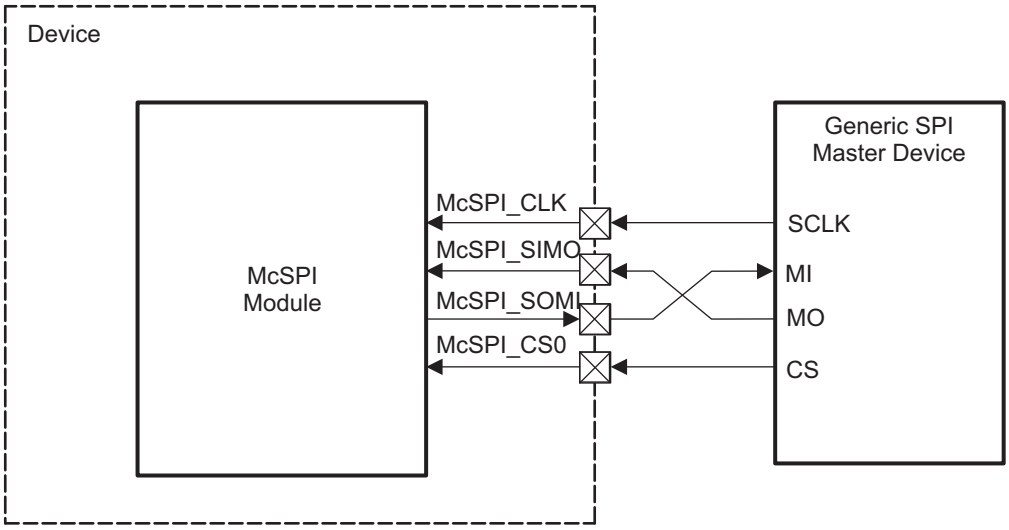
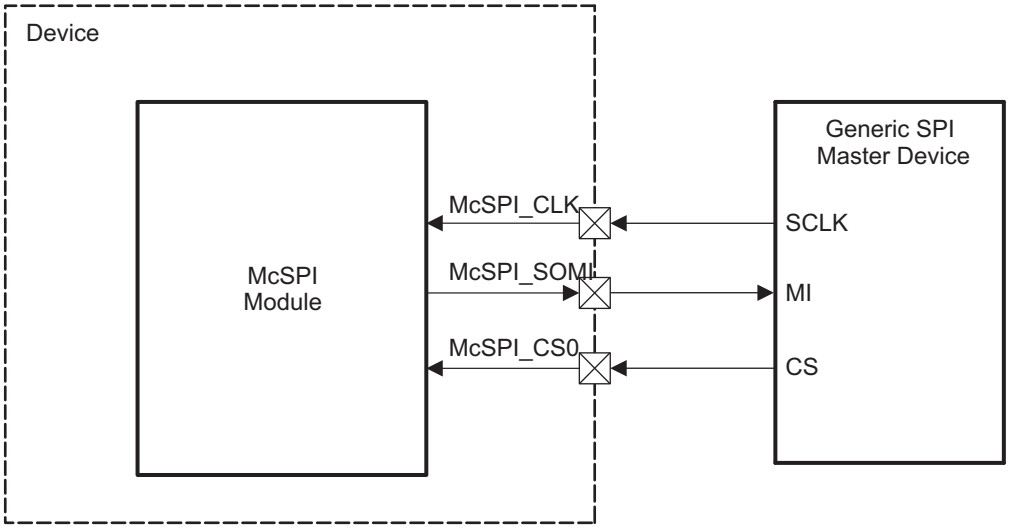


Figure 9-4 shows the slave single mode, which can also be configured in transmit-only mode.

Figure 9-4. McSPI Slave Single Mode (Transmit Only)



9.3 McSPI Functional Interface

9.3.1 Basic McSPI Pins for Master Mode

Figure 9-5 shows all of the McSPI interface signals in master mode.

Figure 9-5. McSPI Interface Signals in Master Mode

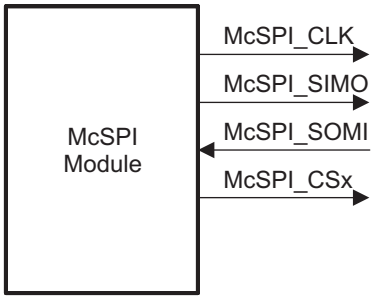


Table 9-1 describes the McSPI I/O in master mode.

Table 9-1. McSPI I/O Description (Master Mode)

Signal Name	I/O	Description	Reset [1]
McSPI_CLK	O	McSPI module serial clock	Low
McSPI_SIMO	O	McSPI module serial data master out (slave input, master output)	Hi-Z
McSPI_SOMI	I	McSPI module serial data master input (slave output, master input)	Hi-Z
McSPI_CSx	O	McSPI module chip-select x output	Hi-Z

(1) After reset, the McSPI module signals are not routed to I/O pins. See the External Bus Selection Register description in the device's data manual for details.

9.3.2 Basic McSPI Pins for Slave Mode

Figure 9-6 shows all of the McSPI interface signals in slave mode.

Figure 9-6. McSPI Interface Signals in Slave Mode

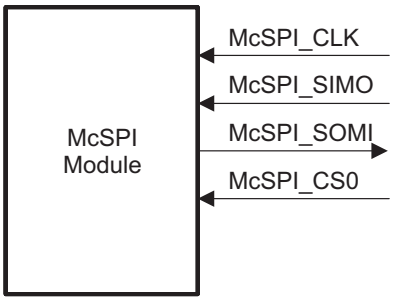


Table 9-2 describes the McSPI I/O in slave mode.

Table 9-2. McSPI I/O Description (Slave Mode)

Signal Name	I/O	Description	Reset [1]
McSPI_CLK	I	McSPI module serial clock	Low
McSPI_SIMO	I	McSPI module serial data master out (slave input, master output)	Hi-Z
McSPI_SOMI	O	McSPI module serial data master input (slave output, master input)	Hi-Z
McSPI_CS0	I	McSPI module chip-select 0 input	Hi-Z

(1) After reset, the SPI modules are in slave mode by default. This paragraph implies that the McSPI module is configured in slave mode. (See the MCSPI_MODULCTRL[2] MS bit in Module Control Register (MCSPI_MODULCTRL).)

9.3.3 Multichannel SPI Protocol and Data Format

The synchronous McSPI protocol allows a master device to initiate serial data transfers to a slave device. A slave select line (McSPI_CSx) allows selection of an individual slave SPI device. Slave devices that are not selected do not interfere with SPI bus activities.

McSPI offers the flexibility to modify the following parameters to adapt to the device features:

- Word length

  McSPI supports any SPI word ranging from 4 bits to 32 bits long (CHxCONFL[11:7] WL field).

  McSPI word length can be changed between transmissions to allow the master device to communicate with peripheral slaves that have different requirements.

- McSPI enable (McSPI_CSx, for channel x)

  The polarity of the McSPI enable signals is programmable (CHxCONFL[6] EPOL bit). McSPI_CSx signals can be active high or low.

  The assertion of the McSPI_CSx signals is programmable and can be manually or automatically asserted. The manual assertion mode is available in single master mode only. McSPI_CSx can be kept active between words with the CHxCONFU[4] FORCE bit.

  Two consecutive words for two different slave devices can go along with active McSPI_CSx signals with different polarity (see the example in Section 9.5, *McSPI Basic Programming Model*).

- Programmable start-bit

  In start-bit mode a start-bit is added before the SPI word length to indicate how the next SPI word must be handled. The start-bit is enabled by setting CHxCONFU[7] SBE bit to 1. The CHxCONFU[8] SBPOL bit defines the polarity of the start-bit.

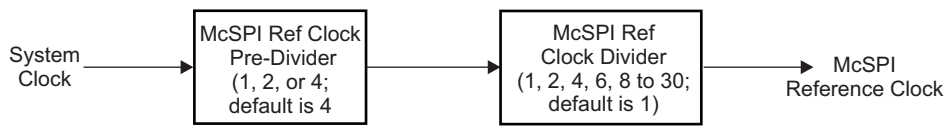
- Programmable McSPI clock
  - Bit rate

    In master mode, the baud rate of the SPI serial clock is programmable using the internal McSPI module functional clock, McSPI Reference Clock. The McSPI Reference Clock is divided down from the system clock by two dividers programmable through the McSPI Functional Clock Divide Register, 0x1C3C, (see Section 1.7.8):

    1. McSPI Reference Clock Pre-divider that divides by 1, 2, or 4
    2. McSPI Reference Clock Divider that divides from 1, 2, 4, 6, 8, to 30.

    The McSPI Reference Clock operates up to 50 MHz.

Figure 9-7. McSPI Reference Clock Diagram



The McSPI Reference Clock is further programmable for individual channel, 0, 1 and 2, using the CLKG, EXTCLK and CLKD bits in Peripheral Control Registers. See Section 9.4.1.8.1, *Clock Ratio Granularity*, for details.

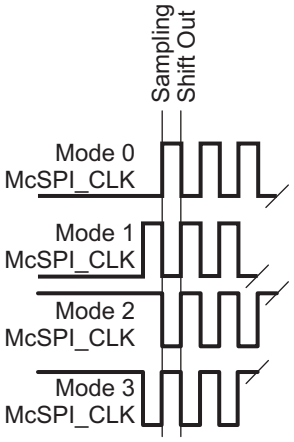
  - Polarity and phase

The polarity (the CHxCONFL[1] POL bit) and the phase (the CHxCONFL[0] PHA bit) of the McSPI serial clock (McSPI_CLK) are configurable to offer four combinations. Software selects the right combination, depending on the device. See Table 9-3 and Figure 9-8.

Table 9-3. Phase and Polarity Combinations

Polarity (POL)	Phase (PHA)	SPI Mode	Comments
0	0	Mode 0	McSPI_CLK is active high and sampling occurs on the rising edge.
0	1	Mode 1	McSPI_CLK is active high and sampling occurs on the falling edge.
1	0	Mode 2	McSPI_CLK is active low and sampling occurs on the falling edge.
1	1	Mode 3	McSPI_CLK is active low and sampling occurs on the rising edge.

Figure 9-8. Phase and Polarity Combinations



9.3.3.1 Transfer Format

In both master and slave modes, McSPI drives the data lines when McSPI_CSx is asserted.

Each word is transmitted starting with the most-significant bit (MSB).

This section explains the two cases of data transmission determined by the clock phase (PHA) and the type of data transmission using a start-bit (SBE) called the start-bit mode:

- Transmission in mode 0 and mode 2 (PHA = 0)

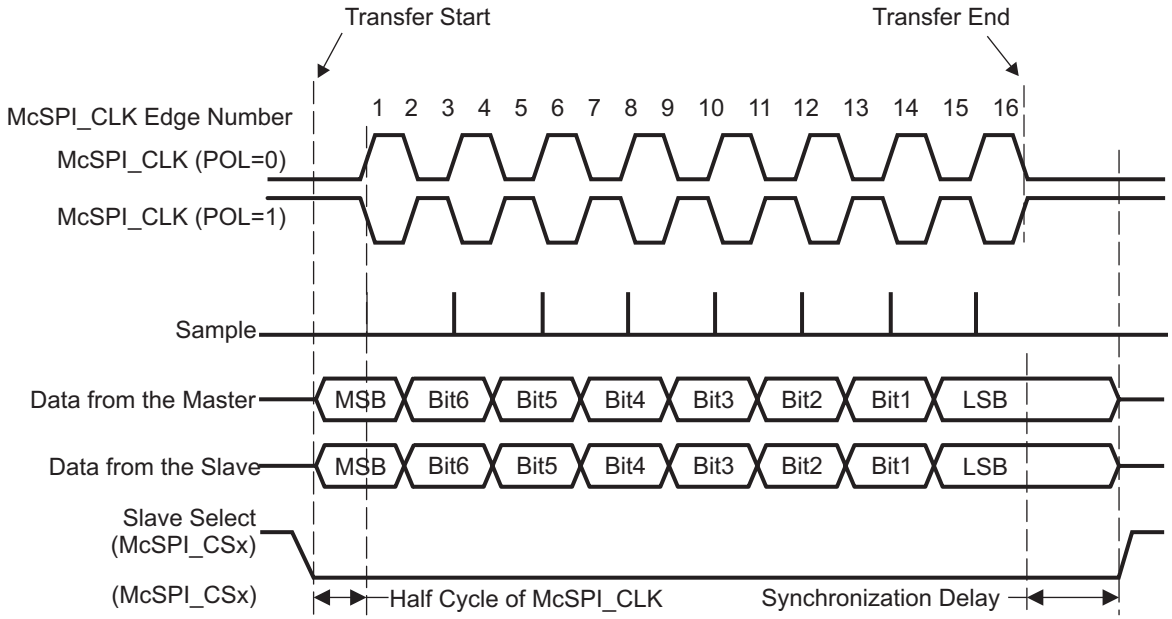
  When PHA = 0, the first bit of the McSPI word to transmit (on the master or the slave data output pin) is valid one half cycle of McSPI_CLK after the McSPI_CSx assertion.

  Therefore, the first edge of the McSPI_CLK line is used by the master to sample the first data bit sent by the slave. On the same edge, the first data bit sent by the master is sampled by the slave.

  On the next McSPI_CLK edge, the received data bit is shifted into the receive shift register and a new data bit is transmitted on the serial data line.

This process continues for a number of pulses on the McSPI_CLK line defined by the McSPI word length programmed in the master device, with data being latched on odd-numbered edges and shifted on even-numbered edges. See Figure 9-9.

Figure 9-9. Full-Duplex Transfer Format With PHA = 0



- Transmission in mode 1 and mode 3 (PHA = 1)

  When PHA = 1, the first bit of the McSPI word to transmit (on the master or the slave data output pin) is valid on the following McSPI_CLK edge (one half cycle later). This is the sampling edge for the master and slave. A synchronization delay is added between the McSPI_CSx activation and the first McSPI_CLK edge.

  The received data bit is shifted into the shift register on the third McSPI_CLK edge.

  This process continues for a number of pulses on the McSPI_CLK line defined by the McSPI word length programmed in the master device, with data being latched on even-numbered edges and shifted on odd-numbered edges.

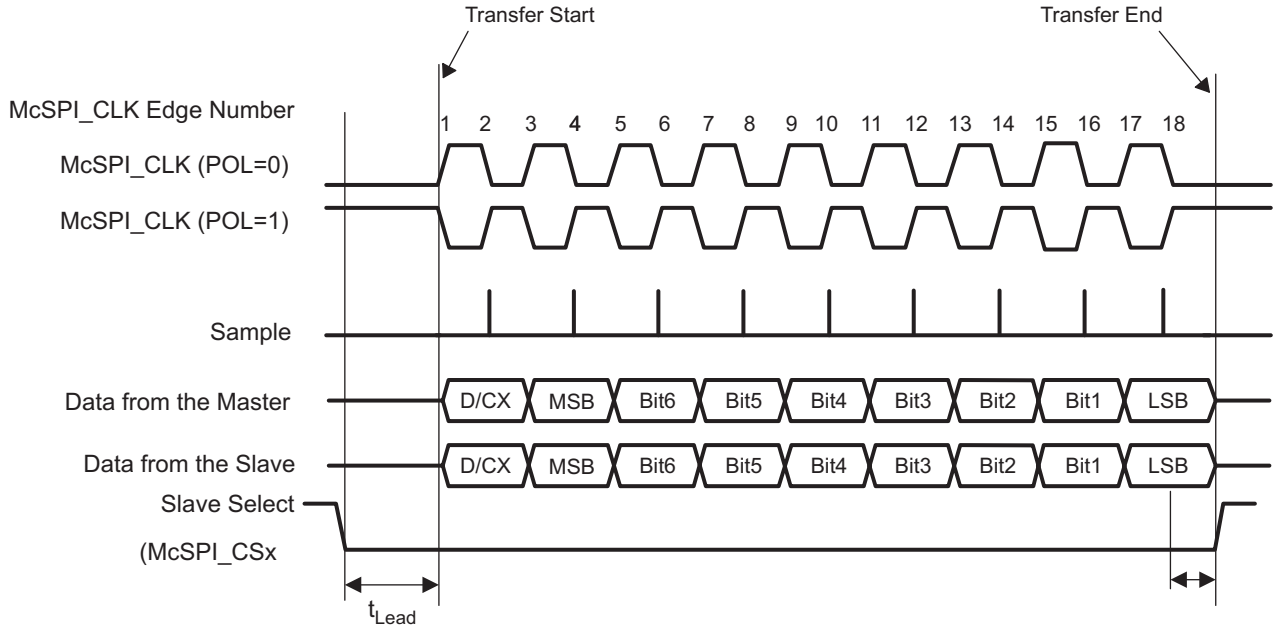
  NOTE: The minimum synchronization delay is one cycle of McSPI_CLK, if the McSPI_CLK frequency equals the McSPI Reference Clock frequency in master mode. The minimum synchronization delay is one-half cycle of McSPI_CLK, if the McSPI_CLK frequency is lower than the McSPI Reference Clock frequency in both master and slave modes.

- Transmission with a start-bit (SBE = 1)

  When the CHxCONFU[7] SBE bit = 1, a start-bit is added before the MSB to indicate whether the next McSPI word must be handled as a command or as data.

  Figure 9-10 shows an example of a data transfer with an extra start-bit.

Figure 9-10. Extended SPI Transfer With a Start-Bit (SBE = 1)



9.4 McSPI Functional Description

9.4.1 Master Mode

9.4.1.1 Master Mode Features

The McSPI master mode supports multichannel communication with up to three independent SPI communication channel contexts. The McSPI initiates a data transfer on the data lines (McSPI_SIMO and McSPI_SOMI) and generates clock (McSPI_CLK) and control signals (McSPI_CSx).

Connected to multiple external devices, the McSPI exchanges data with one SPI device at a time through two main modes:

- Two-data-pins interface mode (transmit-and-receive mode for full-duplex transmission)
- Single-data-pin interface mode (recommended for half-duplex transmission)

Two DMA request events (read and write) allow synchronized accesses of the DMA controller with the activity of McSPI.

Three interrupt events can be used for data transmission and reception in master mode (for more information on interrupts, see Section 9.4.4.1, *Interrupt Events in Master Mode*).

9.4.1.2 Master Transmit-and-Receive Mode (Full Duplex)

In full-duplex transmission, data is transmitted (shifted out serially on McSPI_SIMO) and received (shifted in serially on McSPI_SOMI) simultaneously on separate data lines.

The master transmit-and-receive mode is programmable per channel (the CHxCONFL[13:12] TRM field).

Channel access to the shift registers for transmission/reception is based on the CHxTX transmitter register state, the CHxRX receiver register state, and round robin arbitration.

NOTE: Transmitter register CHxTX refers also to CHxTXL and CHxTXU.

Receiver register CHxRX refers also to CHxRXL and CHxRXU.

Channels that meet the following rules are included in the round robin list of active channels scheduled for transmission and/or reception. The arbiter skips channels that do not meet the rules and searches in the rotation for the next enabled channel.

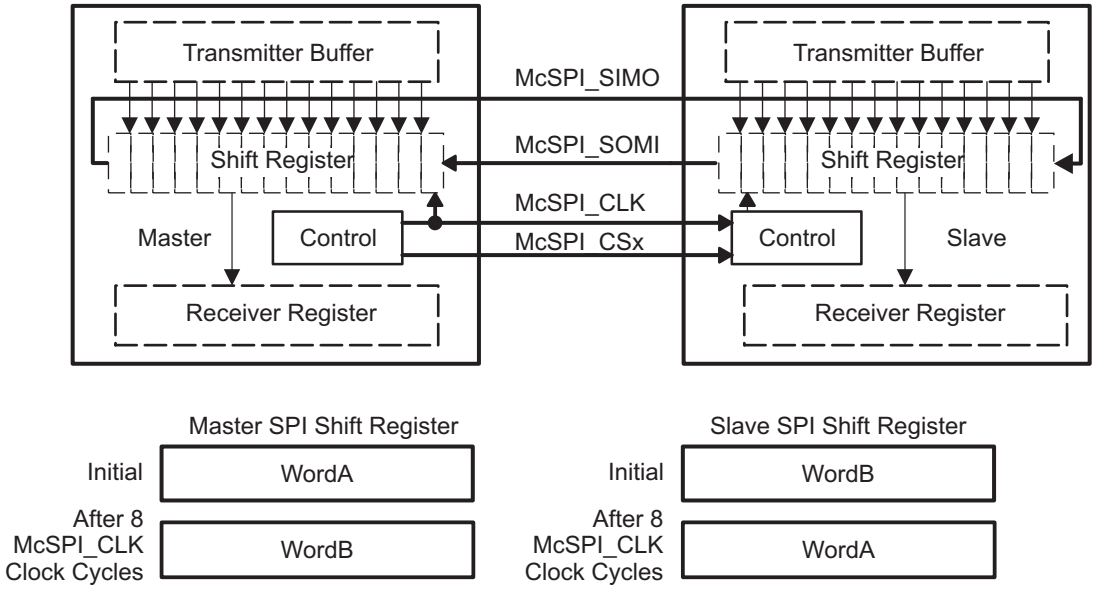
- Rule 1: Only enabled channels (the CHxCTRLL[0] EN bit) can be scheduled for transmission and/or reception.
- Rule 2: If its CHxTX transmitter register is not empty (the CHxSTATL[1] TXS bit), an enabled channel can be scheduled when the shift register is assigned. If the transmitter register is empty when the shift register is assigned, the TXx_UNDERFLOW event is activated, and the next enabled channel with new data to transmit is scheduled (see also the transmit-only mode).
- Rule 3: An enabled channel can be scheduled if its receive register is not full (the CHxSTATL[0] RXS bit) when the shift register is assigned (see also the receive-only mode). Therefore, the receiver register cannot be overwritten. Note that the IRQSTATUSL[3] RX0_OVERFLOW bit is never set to this mode.

When McSPI word transfer completes (the CHxSTATL[2:1] EOT and TXS bits are set), the updated transmitter register of the next scheduled channel is loaded into the shift register. The serialization (transmit-and-receive) starts depending on the channel communication configuration. When serialization completes, the received data transfers to the channel receive register.

The serial clock (McSPI_CLK) synchronizes shifting and sampling of the information on the two serial data lines (McSPI_SIMO and McSPI_SOMI). Each time a bit transfers out from the master, 1 bit transfers in from the slave.

Figure 9-11 shows an example of a full-duplex system with a master device on the left and a slave device on the right. After eight cycles of the serial clock McSPI_CLK, WordA transfers from the master to the slave. At the same time, WordB transfers from the slave to the master.

Figure 9-11. SPI Full-Duplex Transmission (Example)



9.4.1.3 Master Transmit-Only Mode (Half Duplex)

The master transmit-only mode prevents the core processor from reading the CHxRX register (minimizing data movement) when only transmission is meaningful.

The master transmit-only mode is programmable per channel (the CHxCONFL[13:12] TRM field). Transmission starts only after data is loaded into the CHxTX register.

Rule 1 and Rule 2, defined in Section 9.4.1.2, are applicable in this mode.

Rule 3, defined in Section 9.4.1.2, is not applicable.

In master transmit-only mode, the CHxRX register state FULL does not prevent transmission and the CHxRX register is always overwritten with the new McSPI word. This event is not significant when only transmission is meaningful. Thus, the RX0_OVERFLOW bit in the IRQSTATUSL register is never set in this mode.

The hardware automatically disables the RX_FULL interrupt and the DMA read requests.

The transfer status is given by the CHxSTATL[2:1] EOT and TXS bits.

9.4.1.4 Master Receive-Only Mode (Half Duplex)

The master receive mode prevents the core processor from refilling the CHxTX register (minimizing data movement) when only reception is meaningful.

The master receive mode is programmable per channel (the CHxCONFL[13:12] TRM field).

The master receive-only mode enables channel scheduling only on the empty state of the CHxRX register.

Rule 1 and Rule 3, defined in Section 9.4.1.2, are applicable in this mode.

Rule 2, defined in Section 9.4.1.2, is not applicable.

In the master receive-only mode, software must write dummy data to the CHxTX transmitter register and only after the TX buffer is empty (check the CHxSTAT[2:1] bits in the software). Only one dummy write is enough to receive any number of words from the slave. Software should ensure that the transmitter register is always full (the TXx_EMPTY bits of IRQSTATUSL) when receiving. The content of the transmitter register is always loaded into the shift register when the shift register is assigned. After writing the dummy data to the transmitter register, the TXx_EMPTY and TXx_UNDERFLOW bits in the IRQSTATUS register are never set in receive-only mode.

The CHxSTATL[2] EOT bit gives the status of serialization. The RXx_FULL bits of the IRQSTATUSL register are set when received data is loaded from the shift register to the corresponding transmitter register. The IRQSTATUSL[3] RX0_OVERFLOW bit is never set in this mode.

9.4.1.5 Single-Channel Master Mode

When the McSPI is configured as a master device with a single enabled channel, the assertion of the McSPI_CSx signal can be controlled in two different ways:

- If the MODULCTRLL[0] SINGLE bit is set to 0, McSPI_CSx assertion/deassertion after each SPI word is automatically controlled by the McSPI module (see subsections of Section 9.4.1.1, *Master Mode Features*.
- If the MODULCTRLL[0] SINGLE bit and the CHxCONFU[4] FORCE bit are set to 1: McSPI_CSx assertion/deassertion is controlled by software (see Section 9.4.1.5.1, *Programming Tips When Switching to Another Channel*.

9.4.1.5.1 Programming Tips When Switching to Another Channel

When a single channel is enabled and data transfer is ongoing:

- Wait for completion of the SPI word transfer (wait until the CHxSTATL[2] EOT bit is set to 1) before disabling the current channel and enabling a different channel.
- Disable the current channel first, and then enable the other channel.

9.4.1.5.2 Force McSPI_CSx Mode

Continuous transfers are manually allowed by keeping the McSPI_CSx signal active for successive McSPI words transfer. Several sequences (configuration/enable/disable of the channel) can be run without deactivating the McSPI_CSx line. This mode is supported by all channels and any master sequence can be used (transmit-receive, transmit-only, receive-only).

Keeping the McSPI_CSx active mode is supported when:

- A single channel is used (with the MODULCTRLL[0] SINGLE bit set to 1).
- A single channel is enabled (the CHxCTRLL[0] EN bit is set to 1).
- Transfer parameters are loaded in the configuration register of the appropriate channel (CHxCONFL and CHxCONFU).
- Writing 1 to the CHxCONFU[4] FORCE bit enables the force McSPI CS mode.

When the channel is enabled, the McSPI_CSx signal activates with the programmed polarity. As in the multichannel master mode, the transfer start depends on the status of the CHxTX register transmitter (the CHxSTATL[1] TXS bit), the status of the CHxRX receiver register (the CHxSTATL[0] RXS bit), and the defined mode (the CHxCONFL[13:12] TRM field) of the channel enabled.

The CHxSTATL[2] EOT bit gives the transfer status of each SPI word. The RXx_FULL bit in the IRQSTATUSL register is set when received data is loaded from the shift register to the receiver register.

A change in the configuration parameters is propagated directly on the SPI interface. If the McSPI_CSx signal is activated, ensure that the configuration is changed only between SPI words to avoid corrupting the current transfer.

NOTE: To avoid data corruption, McSPI_CSx polarity and McSPI_CLK phase and McSPI_CLK polarity must not be modified when the McSPI_CSx signal is activated.

A delay between McSPI words that requires the connected SPI slave device to switch from one configuration (transmit-only for instance) to another (receive-only for instance) must be handled by software.

At the end of the last SPI word, the channel must be deactivated (the CHxCTRLL[0] EN bit set to 0) and McSPI_CSx can be forced to its inactive state using the CHxCONFU[4] FORCE bit.

Figure 9-12 and Figure 9-13 show successive transfers with McSPI_CSx maintained active low with a different configuration for each McSPI word in half- and full-duplex modes, respectively.

Figure 9-12. Continuous Transfers With McSPI_CSx Maintained Active (Half Duplex Mode)

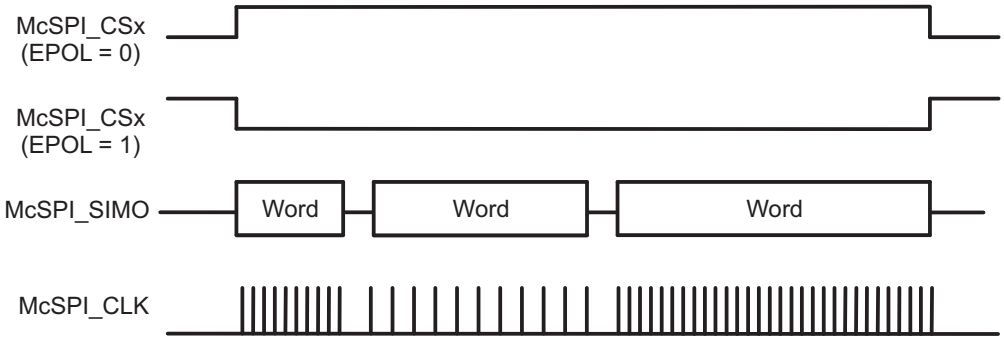
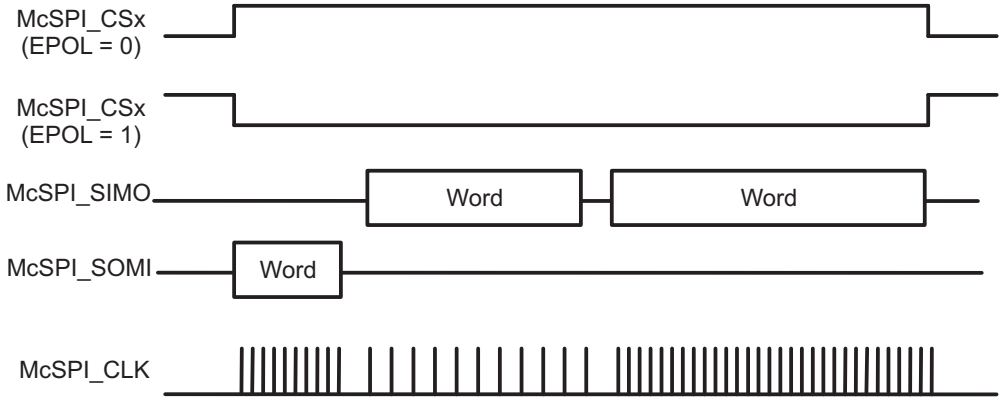


Figure 9-13. Continuous Transfers With McSPI_CSx Maintained Active (Full Duplex Mode)



NOTE: The turbo mode described in Section 9.4.1.5.3, *Turbo Mode*, maintains McSPI_CSx in active mode when the following conditions are met:

- A single channel is explicitly used (the MODULCTRLL[0] SINGLE bit is set to 1).
- Turbo mode is enabled in the configuration of the channel. (The CHxCONFU[3] TURBO bit is set to 1.)

9.4.1.5.3 Turbo Mode

The turbo mode improves the throughput of the SPI interface when a single channel is enabled by allowing transfers until the shift register and the CHxRX receiver register are full. The turbo mode is useful (time savings) when a transfer exceeds two words. This mode is programmable per channel (via the CHxCONFU[3] TURBO bit).

When several channels are enabled, the TURBO bit has no effect and the channel access to the shift registers remains as previously described.

In turbo mode, Rule 1 and Rule 2 applies, but Rule 3 does not (see Section 9.4.1.2). An enabled channel can be scheduled if its receive register is full (the CHxSTATL[0]) RXS bit) at the time of the shift-register assignment until the shift register is full.

The receiver register cannot be overwritten in turbo mode. Consequently, the IRQSTATUSL[3] RX0_OVERFLOW bit is never set in this mode.

9.4.1.6 Start Bit Mode

In start bit mode, an extended bit is added before the SPI word in order to indicate whether the next SPI word must be handled as a command or as data. This feature is only available in master mode. The start bit mode cannot be used at the same time as turbo mode and/or force McSPI_CSx mode. In this case, only one channel can be used; round-robin arbitration is not possible.

This mode is programmable per channel by setting the CHxCONFU[7] SBE bit to 1. The polarity of the extended bit is programmable per channel. When the CHxCONFU[8] SBPOL bit is set to 0, the SPI word must be handled as a command. When the CHxCONFU[8] SBPOL bit is set to 1, the SPI word must be handled as data. Moreover, start-bit polarity can be changed dynamically during start bit transfer without disabling the channel for reconfiguration; in this case, users must configure the CHxCONFU[8] SBPOL bit before writing the SPI word to be transmitted to the TX register.

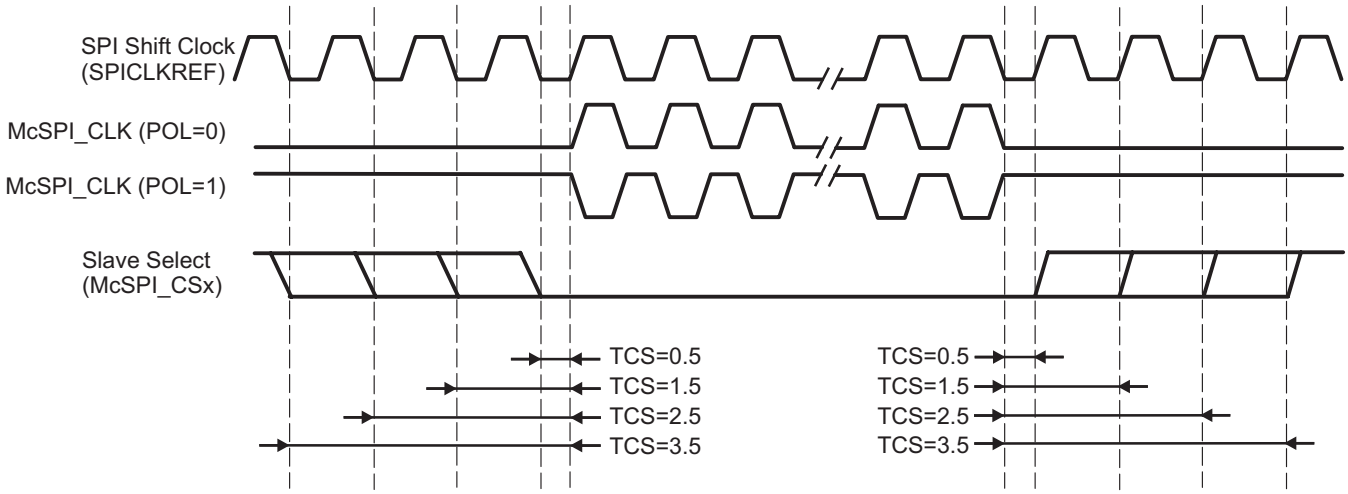
9.4.1.7 Chip-Select Timing Control

The chip select timing control is only available in master mode with automatic chip select generation (MODULCTRLL[0] SINGLE bit field set to 0), to add a programmable delay between chip select assertion and first clock edge, or chip select removal and last clock edge.

This mode is programmable per channel (the TCS bit of the CHxCONFU register).

Figure 9-14 shows the chip-select McSPI_CSx timing control.

Figure 9-14. Chip-Select McSPI_CSx Timing Controls



NOTE: Because of the design implementation for transfers using a clock divider ratio set to 1 (clock bypassed), a half cycle must be added to the value between chip-select assertion and the first clock edge with PHA = 1 or between chip-select removal and the last clock edge with PHA = 0.

9.4.1.8 Programmable McSPI Clock (McSPI_CLK)

In master mode, the baud rate of the SPI serial clock is programmable.

The internal McSPI Reference Clock is used as input of a programmable divider (the CHxCONFL[5:2] CLKD field) to generate the bit rate of the serial output clock McSPI_CLK. The divider value doubles each iteration until 0xF: 1, 2, 4, 8, 18 to 32768.

9.4.1.8.1 Clock Ratio Granularity

To further refine the granularity capability to program the McSPI_CLK, each individual channel can use the CLKG, EXTCLK and CLKD bits in the Peripheral Control Registers to generate an Fratio used to divide the McSPI Module Reference Clock to become the McSPI_CLK for that channel.

By default the clock division ratio is defined by the CHxCONFL[5:2] CLKD bit field with power of two granularity leading to a clock division in range 1 to 4096. With the CHxCONFU[13] CLKG bit, clock division granularity can be changed to one clock cycle, in that case the CHxCTRLL[15:8] EXTCLK bit field is concatenated with CHxCONFL[5:2] CLKD bit field to give a 12-bit width division ratio in range 1 to 4096.

If CLKG = 0: Fratio = 2 to the power of CLKD.

If CLKG = 1: Fratio = EXTCLK concatenated with CLKD + 1 as shown below:

Fratio ± 1											
11	10	9	8	7	6	5	4	3	2	1	0
CHxCTRLL[15:8] EXTCLK bit field								CHxCONFL[5:2] CLKD bit field			
15	14	13	12	11	10	9	8	5	4	3	2

Table 9-4 shows clock granularity examples with a clock source frequency of 50 MHz.

Table 9-4. Clock Granularity Examples

CLKG	EXTCLK	CLKD	Equation	Fratio	SPI Module (MHz)	McSPI_CLK (MHz)
0	X	0	2^0	1	50	50
0	X	1	2^1	2	50	25
0	X	2	2^2	4	50	12.5
0	X	3	2^3	8	50	6.25
0	X	4	2^4	16	50	3.15
0	X	5	2^5	32	50	1.58
1	0	0	0x00 + 1	1	50	50
1	0	1	0x01 + 1	2	50	33.33
1	0	2	0x02 + 1	3	50	25
1	0	3	0x03 + 1	4	50	20
1	0	4	0x04 + 1	5	50	10
1	1	0	0x10 + 1	17	50	2.94
1	1	1	0x11 + 1	18	50	2.78
1	1	2	0x12 + 1	19	50	2.63
1	5	0	0x50 + 1	81	50	0.62
1	5	7	0x57 + 1	88	50	0.57

9.4.2 Slave Mode

To select the McSPI slave mode, set the MODULCTRLL[2] MS bit.

In slave mode, the McSPI initiates data transfer on the data lines (McSPI_SIMO and McSPI_SOMI) when it is selected by an active control signal (McSPI_CS0) and receives an SPI clock (McSPI_CLK) from the external SPI master device. Only channel 0 can be configured as a slave. In slave mode, the McSPI uses the edge of McSPI_CS0 to detect word length. For this reason, McSPI_CS0 must become inactive between each word.

The McSPI does not support McSPI_CS0 active between SPI words. It uses the edge to detect word length.

9.4.2.1 Dedicated Resources

In slave mode, only channel 0 can be enabled; enabling any other channel will have no effect.

The channel 0 in slave mode has the following resources:

- Its own channel enable, programmable with the CHxCTRLL[0] EN bit. This channel must be enabled before transmission and reception.
- For this mode, the slave-select signal can be detected only on McSPI_CS0.
- Its own transmitter register, CH0TX, on top of the common transmit shift register. If the transmitter register is empty, the CH0STAT[1] TXS bit is set. If McSPI is selected by an external master (McSPI_CS0 port assigned to channel 0), the transmitter register content of channel 0 is always loaded into the shift register, whether its content is updated or not. The transmitter register must be loaded before McSPI is selected by a master.
- Its own receiver register, CH0RX, on top of the common receive shift register. If the receiver register is full, the CH0STATL[0] RXS bit is set.

NOTE: The CHxTX register and CHxRX registers of the other channels are not used. Reading from or writing to a channel register other than channel 0 has no effect.

- Its own communication configuration with the following parameters through the CH0CONFL register:
  - Transmit and receive modes, programmable with the TRM field
  - Interface mode (two data pins or single data pin) and data pins assignment, both programmable with the IS and DPE bits. (The SPI module is in slave mode after reset and must be properly configured for the modules to act in master mode.)
  - SPI word length, programmable with the WL bit
  - McSPI_CSx polarity, programmable with the EPOL bit
  - McSPI_CLK polarity, programmable with the POL bit
  - McSPI_CLK phase, programmable with the PHA bit

  The McSPI_CLK frequency of a transfer is controlled by the external SPI master connected to the McSPI slave device. The CH0CONFL[5:2] CLKD field is not used in slave mode.

NOTE: The configuration of the channel can be loaded in the CH0CONFL and CH0CONFU register when the channel is disabled.

- Two DMA request events, read and write, synchronize read/write accesses of the DMA controller with the activity of McSPI. DMA requests are asserted using the CH0CONFL[15] DMAR bit for reading and the CH0CONFL[14] DMAW bit for writing.
- Four interrupt events (see Section 9.4.4.2, *Interrupt Events in Slave Mode*)

9.4.2.2 Slave Transmit-and-Receive Mode

Only channel 0 can be configured as a slave, and only McSPI_CS0 can be used as a chip-enable in slave mode.

The slave receive mode is programmable (set the CH0CONFL[13:12] TRM field to 0x0).

In slave transmit-and-receive mode, the CH0TX transmitter register must be loaded before McSPI is selected by an external SPI master device.

After a channel is enabled, transmission and reception proceed with interrupt and DMA request events.

The transmitter register content is always loaded in the shift register whether it is updated or not. The event TX0_UNDERFLOW is activated accordingly and does not prevent transmission.

When a McSPI word transfer completes (the CH0STATL[2:1] EOT and TXS bits set to 1), the received data is transferred to the channel receive register.

9.4.2.3 Slave Transmit-Only Mode

The slave transmit-only mode is programmable (set the CH0CONFL[13:12] TRM field to 0x2) and avoids the requirement for the local host to read the CH0RX receiver register (minimizing data movement) only when transmission is meaningful.

NOTE: Transmitter register CHxTX refers also to CHxTXL and CHxTXU.

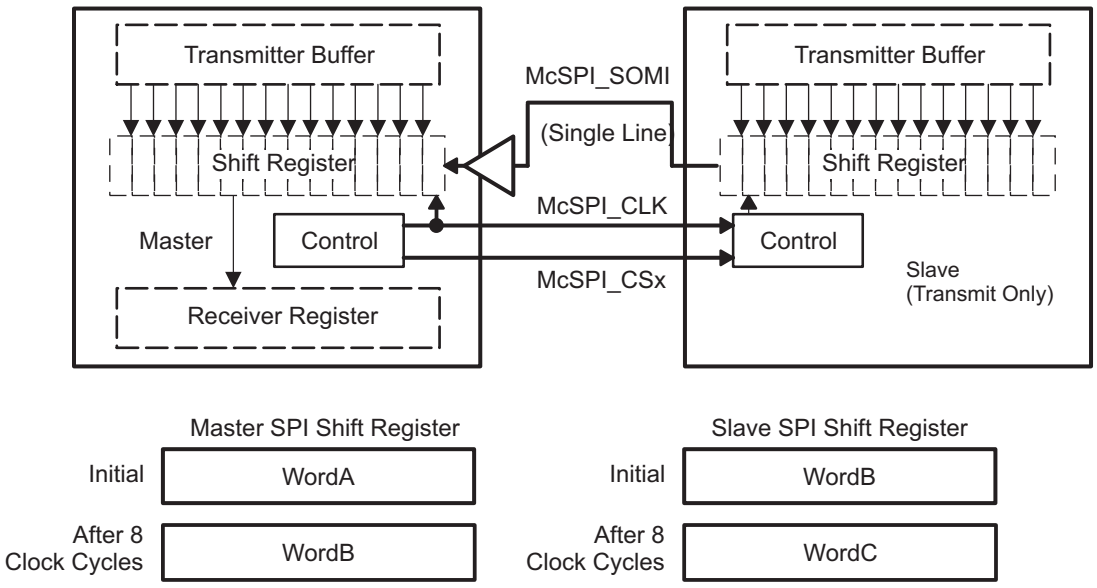
Receiver register CHxRX refers also to CHxRXL and CHxRXU.

To use the McSPI as a slave transmit-only device, the RX0_FULL and RX0_OVERFLOW interrupts and DMA read requests must be disabled because of the receiver register state.

When the McSPI word transfer completes, the CH0STATL[2:1] EOT and TXS bits are set.

Figure 9-15 shows a half-duplex system with a master device on the left and a transmit-only slave device on the right. Each time a bit transfers out from the slave device, 1 bit transfers in the master. After eight cycles of the serial clock McSPI_CLK, WordB transfers from the slave to the master.

Figure 9-15. McSPI Half-Duplex Transmission (Transmit-Only Slave)



9.4.2.4 Slave Receive-Only Mode

The slave receive mode is programmable (set the CH0CONFL[13:12] TRM field to 0x1).

In receive-only mode, the CH0TX transmitter register must be loaded before the McSPI is selected by an external SPI master device. The transmitter register content is always loaded into the shift register whether it is updated or not. The TX0_UNDERFLOW event is activated accordingly and does not prevent transmission.

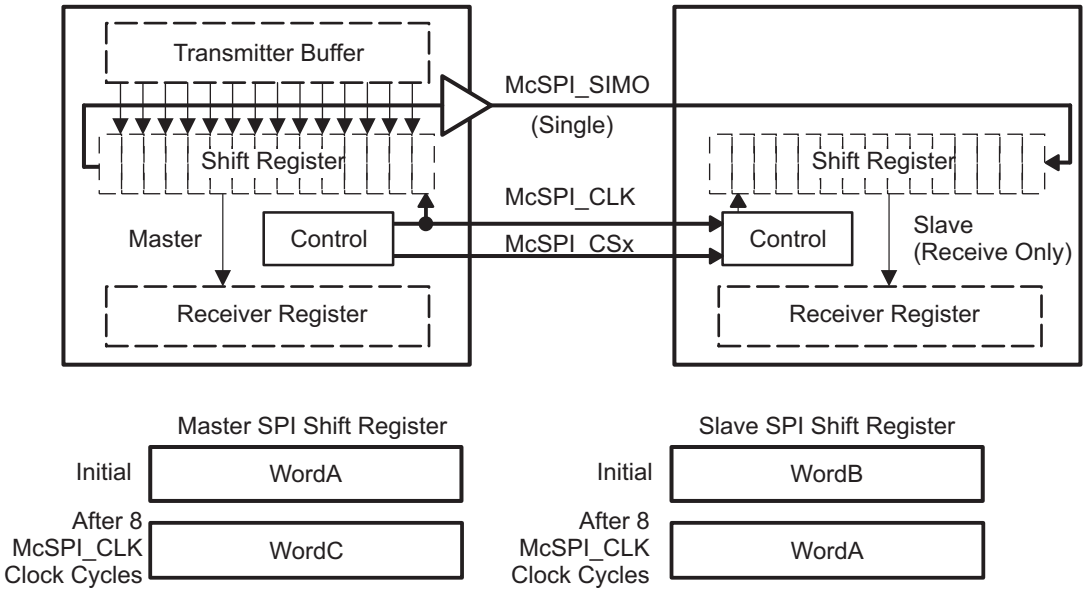
When an SPI word transfer completes (the CH0STATL[2] EOT bit is set to 1), the received data is transferred to the channel receive register.

To use the McSPI as a slave receive-only device, the TX0_EMPTY and TX0_UNDERFLOW interrupts and the DMA write requests must be disabled due to the transmitter register state.

For a full-duplex transmission, the serial clock (McSPI_CLK) synchronizes shifting and sampling of the information on the single serial data line. For full duplex, two data lines are required. If McSPI_CLK synchronizes on a single serial data line, the data line should be half-duplex.

Figure 9-16 shows an example of a half-duplex system with a master device on the left and a receive-only slave device on the right. Each time a bit transfers out from the master, 1 bit transfers in from the slave. After eight cycles of the serial clock McSPI_CLK, Word A transfers from the master to the slave.

Figure 9-16. McSPI Half-Duplex Transmission (Receive-Only Slave)



9.4.3 FIFO Buffer Management

The McSPI controller has a built-in, 128-byte buffer to unload DMA or interrupt handler and improve data throughput.

This buffer can be used by only one channel at once and is selected by setting the CHxCONFU[12] FFER bit or CHxCONFU[11] FFEW bit to 1. If several channel are selected and several FIFO enable bit fields set to 1, the controller forces buffer not to be used, it is the responsibility of the driver to set only one FIFO enable bit field.

The buffer can be used in the modes defined below:

- Master or Slave mode.
- Transmit only, Receive only or Transmit/Receive mode.
- Single channel or turbo mode, or in normal round robin mode. In round robin mode the buffer is used by only one channel.
- Every word length CHxCONFL[11:7] WL are supported.

In transmit/receive mode, the buffer can be used in transmit (see Figure 9-17) or receive (see Figure 9-18) directions, or in both directions. If only one direction is chosen in transmit/receive mode, the full buffer is used for this direction. In both directions, the buffer is split into two 64-byte buffers, one for each direction. See Figure 9-19.

Figure 9-17. Buffer Use in Transmit Direction Only

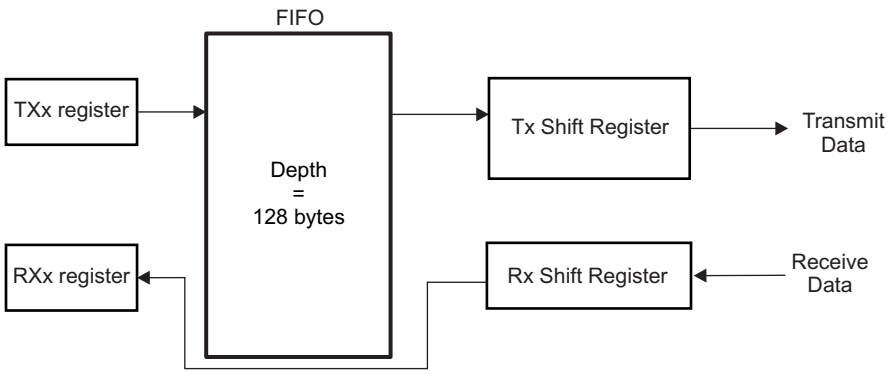


Figure 9-18. Buffer Use in Receive Direction Only

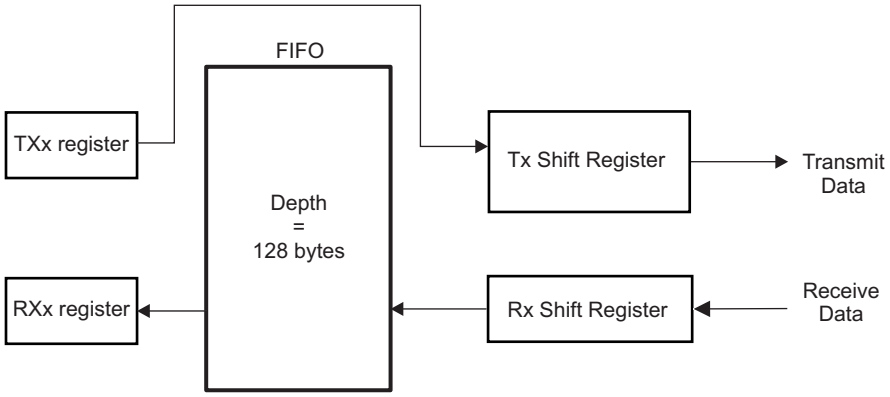
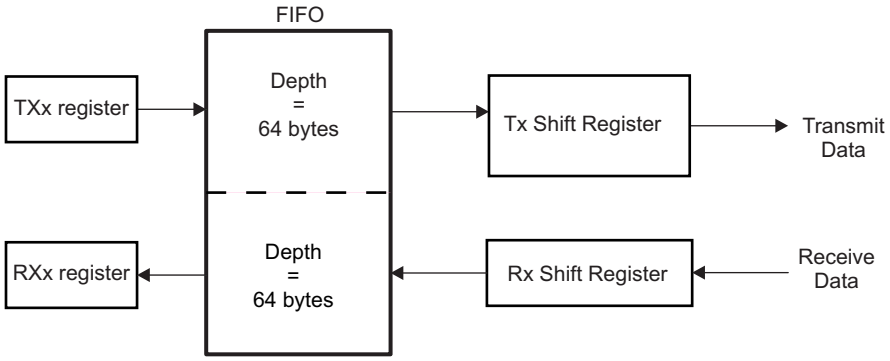


Figure 9-19. Buffer Used For Both Transmit/Receive Directions



Two levels, XFERLEVELU[15:8] AFL and XFERLEVELL[7:0] AEL, rule the buffer management. The granularity of these levels is one byte, then it is not aligned with SPI word length. It is the responsibility of the driver to set these values as a multiple of SPI word length defined in WL. Table 9-5 shows the number of byte written in the FIFO depending on the word length.

Table 9-5. FIFO Writes, Word Length Relationship

SPI Word Length WL	3 ≤ WL ≤ 7	8 ≤ WL ≤ 15	16 ≤ WL ≤ 31
Number of byte written in the FIFO	1 byte	2 bytes	4 bytes

The FIFO buffer pointers are reset when the corresponding channel is enabled or FIFO configuration changes.

9.4.3.1 Buffer Almost Full

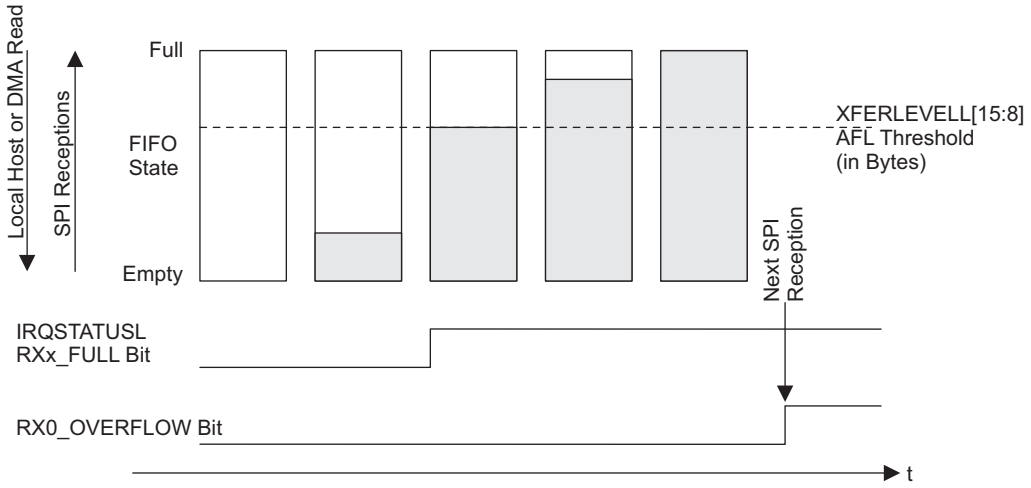
The XFERLEVELL[15:8] AFL bit field is needed when the buffer is used to receive SPI word from a slave (CHxCONFU[12] FFER bit must be set to 1). It defines the Almost Full buffer status. See Figure 9-20.

When FIFO pointer reaches this level an interrupt or a DMA request is sent to the local host to enable system to read AFL+1 bytes from Receive register. Be careful AFL+1 must correspond to a multiple value of CHxCONFL[11:7] WL bit field.

When DMA is used, the request is de-asserted after the first Receive register read.

No new request will be asserted again as long as system has not performed the right number of read accesses.

Figure 9-20. Buffer Almost Full Level (AFL)



NOTE: IRQSTATUSL and IRQSTATSU register bits are not available in DMA mode. In DMA mode, the DMA request is asserted on the same conditions than the IRQSTATUSL RXx_FULL flag.

9.4.3.2 Buffer Almost Empty

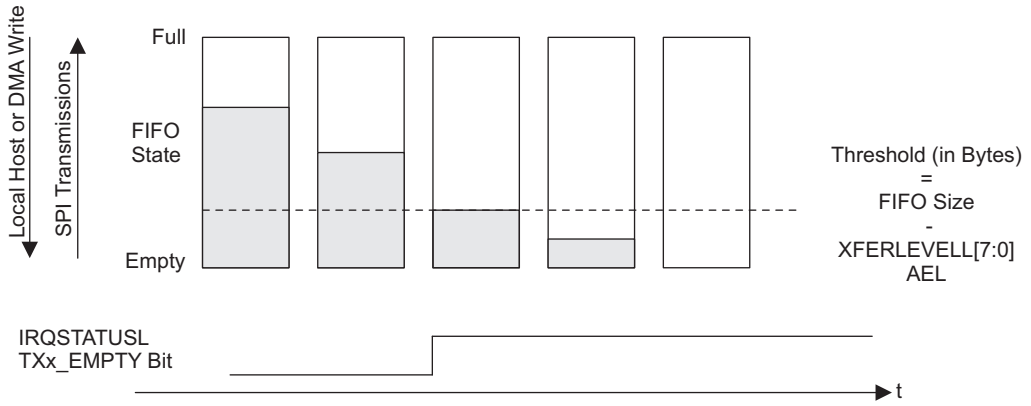
The XFERLEVELL[7:0] AEL bit field is needed when the buffer is used to transmit SPI word to a slave (CHxCONFU[11] FFEW bit must be set to 1). It defines the Almost Empty buffer status. See Figure 9-21.

When FIFO pointer has reached this level an interrupt or a DMA request is sent to the core processor to enable system to write AEL+1 bytes to Transmit register. Be careful AEL+1 must correspond to a multiple value of CHxCONFL[11:7] WL bit field.

When DMA is used, the request is de-asserted after the first Transmit register write.

No new request will be asserted again as long as system has not performed the right number of write accesses.

Figure 9-21. Buffer Almost Empty Level (AEL)



NOTE: IRQSTATUSL and IRQSTATSU register bits are not available in DMA mode. In DMA mode, the DMA request is asserted on the same conditions than the IRQSTATUSL TXx_EMPTY flag.

9.4.3.3 End of Transfer Management

When the FIFO buffer is enabled for a channel, the user shall previously configure in the XFERLEVELL register the AEL and AFL levels and especially the XFERLEVELU[15:0] WCNT bit field to define the number of SPI words to be transferred using the FIFO before enabling the channel.

This counter allows the controller to stop the transfer correctly after a defined number of SPI word transfers. If WNCT is set to 0x0000, the counter is not used and the user must stop the transfer manually by disabling the channel, in this case the user does not know how many SPI transfers have been done. For received words, software shall poll the CHxSTAT[5] RXFFE bit and read the receiver register to empty the FIFO buffer.

When the End Of Word count interrupt is generated (IRQSTATUSU[11] EOW bit set), the user can disable the channel and poll the CHxSTATL[5] RXFFE bit to know it lasts SPI words in the FIFO buffer and read them.

No new request will be asserted again as long as system has not performed the right number of write accesses.

NOTE: The RXFFE status bit shows only the FIFO status. The data received are stored not only in the FIFO, but also in the shift register and the MCSPI_RX register. Therefore, the FIFO can be empty even after receiving two words.

9.4.4 Interrupts

Each channel can issue interrupt events.

Each interrupt event has status bits in the IRQSTATUSL register (RXx_FULL, TXx_UNDERFLOW, TXx_EMPTY, ...) with x = [0,2] that indicate if service is required. Each status bit has an interrupt enable bit (a mask) in the IRQENABLEL register (RXx_FULL_ENABLE, TXx_UNDERFLOW_ENABLE, TXx_EMPTY_ENABLE, ...).

When an interrupt occurs and a mask is later applied on it, the interrupt line is not asserted again, even if the interrupt source is not serviced.

The McSPI supports interrupt-driven and polling operations.

9.4.4.1 Interrupt Events in Master Mode

In master mode, the interrupt events related to the CHxTX register state are TXx_EMPTY and TXx_UNDERFLOW. The interrupt event related to the CHxRX register state is RXx_FULL.

9.4.4.1.1 TXx_EMPTY

The TXx_EMPTY event is activated when a channel is enabled and its CHxTX register is empty (transient event). Enabling a channel automatically triggers this event, except in master receive-only mode (see Section 9.4.1.4, *Master Receive-Only Mode*). When the FIFO buffer is enabled (CHxCONFU[11] FFEW bit set to 1), the IRQSTATUSL TXx_EMPTY bit is set as soon as there is enough space in buffer to write a number of bytes defined by the XFERLEVELL[7:0] AEL bit field.

The transmitter register must be loaded with data to remove the source of the interrupt; the IRQSTATUSL TXx_EMPTY interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

When FIFO is enabled, no new TXx_EMPTY event will be asserted as soon as the local host has not performed the number of write into the transmitter register defined by XFERLEVELL[7:0] AEL bit field. It is the responsibility of the local host to perform the right number of writes.

9.4.4.1.2 TXx_UNDERFLOW

The event TXx_UNDERFLOW is activated when the channel is enabled and if the transmitter register or if the FIFO is empty (not updated with new data) when an external master device starts a data transfer with the McSPI (transmit and receive).

The TXx_UNDERFLOW is a harmless warning in master mode.

To avoid having TXx_UNDERFLOW event at the beginning of a transmission, the event TXx_UNDERFLOW is not activated when no data has been loaded into the transmitter register since the channel has been enabled. To avoid having a TXx_UNDERFLOW event, the transmitter register must be seldom loaded.

The IRQSTATUSL TXx_UNDERFLOW interrupt status bit must be cleared for interrupt line de-assertion (if event enable as interrupt source).

9.4.4.1.3 RXx_ FULL

The RXx_FULL event is activated when channel is enabled and CHxRX register becomes filled (transient event). When FIFO buffer is enabled (CHxCONFU[12] FFER bit set to 1), the RXx_FULL is asserted as soon as the number of bytes holds in the FIFO to be read reaches the XFERLEVELL[15:8] AFL threshold.

The receiver register must be read to remove the source of the interrupt; the IRQSTATUSL RXx_FULL interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

When FIFO is enabled, no new RXx_FULL event will be asserted as soon as the local host has not performed AFL+1 reads into the receiver registers. It is the responsibility of local host to perform the right number of reads.

9.4.4.1.4 End Of Word Count

The IRQSTATUSU[1] EOW event (End Of Word count) is activated when channel is enabled and configured to use the built-in FIFO. This interrupt is raised when the controller had performed the number of transfer defined in the XFERLEVELU[15:0] WCNT bit field. If WCNT is set to 0x0000, the counter is not enable and this interrupt is not generated.

The End of Word count interrupt also indicates that the SPI transfer is halt on channel using the FIFO buffer as soon as XFERLEVELU[15:0] WCNT is not reloaded and the channel is not re-enabled.

The IRQSTATUSU[1] EOW interrupt status bit must be cleared for interrupt line de-assertion (if event enable as interrupt source).

9.4.4.2 Interrupt Events in Slave Mode

In slave mode, the interrupt events related to the transmitter register state are TXx_EMPTY and TXx_UNDERFLOW. The interrupt events related to the receiver register state are RXx_FULL and RX0_OVERFLOW (Channels 1 and 2 do not have a receiver overflow status bit). See the IRQSTATUSL and IRQSTATUSU register.

9.4.4.2.1 TXx_EMPTY

The TXx_EMPTY event is activated when a channel is enabled and its transmitter register is empty. Enabling the channel automatically raises this event. If the FIFO buffer is enabled (CHxCONFU[11] FFEW bit set to 1), the TXx_EMPTY event is asserted as soon as there is enough space in buffer to write a number of byte defined by the XFERLEVELL[7:0] AEL bit field.

The transmitter register must be loaded with data to remove the source of the interrupt; the IRQSTATUSL TXx_EMPTY interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

When FIFO is enabled, no new TXx_EMPTY event will be asserted as soon as the local host has not performed the number of write into the CHxTX transmitter register defined by XFERLEVELL[7:0] AEL bit field. It is the responsibility of the local host to perform the right number of writes.

9.4.4.2.2 TXx_UNDERFLOW

The TXx_UNDERFLOW event is activated when a channel is enabled and if the transmitter register is empty (not updated with new data) when an external master device starts a data transfer with the McSPI (transmit and receive).

When FIFO is enabled, the data emitted while underflow event is raised is not the last data written in the FIFO but the next data in the FIFO (an old transmitted value or a dummy data is the FIFO has been reset).

TXx_UNDERFLOW indicates an error (data loss) in slave mode.

To avoid having a TXx_UNDERFLOW event at the beginning of a transmission, the TXx_UNDERFLOW event is not activated when data is not loaded into the transmitter register because the channel is enabled.

The IRQSTATUSL TXx_UNDERFLOW interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

9.4.4.2.3 RXx_FULL

The RXx_FULL event is activated when a channel is enabled and the receiver register is being filled (transient event). When FIFO buffer is enabled (CHxCONFU[12] FFER bit set to 1), RXx_FULL is asserted as soon as there is a number of bytes holds in buffer to read defined by the XFERLEVELL[15:8] AFL bit field.

The receiver register must be read to remove the source of the interrupt; the IRQSTATUSL RXx_FULL interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

When FIFO is enabled, no new RXx_FULL event will be asserted as soon as the local host has not performed AFL+1 reads into the receiver registers. It is the responsibility of local host to perform the right number of reads.

9.4.4.2.4 RX0_OVERFLOW

The RX0_OVERFLOW event is activated in slave mode in either transmit-and-receive or receive-only mode, when a channel is enabled and the receiver register or FIFO is full when a new SPI word is received. The receiver register is always overwritten with the new SPI word. If the FIFO is enabled data within the FIFO are overwritten, it must be considered as corrupted. The RX0_OVERFLOW event should not appear in slave mode using the FIFO.

The RX0_OVERFLOW indicates an error (data loss) in slave mode.

The IRQSTATUSL[3] RX0_OVERFLOW interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

9.4.4.2.5 End Of Word Count

The IRQSTATUSU[1] EOW (End of Word count) event is activated when channel is enabled and configured to use the built-in FIFO. This interrupt is raised when the controller had performed the number of transfer defined in the XFERLEVELU[15:0] WCNT bit field. If WCNT is set to 0x0000, the counter is not enabled and this interrupt is not generated.

The End of Word count interrupt also indicates that the SPI transfer is halt on channel using the FIFO buffer as soon as WCNT is not reloaded and channel re-enabled.

The IRQSTATUSU[1] EOW interrupt status bit must be cleared for interrupt line de-assertion (if event enable as interrupt source).

9.4.4.3 Interrupt-Driven Operation

An interrupt enable bit, in IRQENABLEL and IRQENABLEU register, can be set to enable each event to generate interrupt requests when the corresponding event occurs. Status bits are automatically set by hardware logic conditions.

When an event occurs (the single interrupt line is asserted), the local host must:

- Read the IRQSTATUSL and IRQSTATUSU registers to identify which event occurred.
- Read the receiver register that corresponds to the event to remove the source of an RXx_FULL event or write into the transmitter register that corresponds to the event to remove the source of a TXx_EMPTY event. No action is required to remove the source of the TXx_ UNDERFLOW and

RX0_OVERFLOW events.

- Write 1 into the corresponding bit of the IRQSTATUSL and IRQSTATUSU register to clear an interrupt status and then release the interrupt line.

The interrupt status bit must always be reset after channel enabling and before events are enabled as interrupt sources.

9.4.4.4 Polling

When the interrupt capability of an event is disabled in the IRQENABLEL and IRQENABLEU register, the interrupt line is not asserted, but the status bits in the IRQSTATUSL and IRQSTATUS register can be polled by software to detect when the corresponding event occurs.

Once the expected event occurs:

- RXx_FULL: To remove the source of the event, the local host must read the corresponding receiver register.
- TXx_EMPTY: To remove the source of the event, the local host must write into the corresponding transmitter register.
- TXx_UNDERFLOW and RX0_OVERFLOW: No action is required to remove the source of the event.

To clear an interrupt, set the corresponding status bit of the IRQSTATUS registers to 1. This does not affect the interrupt line state.

9.4.5 DMA Requests

The DMA controller module manages DMA accesses. The DMA controller advantage is to discharge the local host for data transfers.

Each McSPI channel can issue DMA requests if they are enabled. There are two DMA request lines per McSPI channel (one for read and one for write).

The DMA read request line is asserted when the McSPI channel is enabled and new data is available in the receiver register of the McSPI channel. A DMA read request can be individually masked with the CHxCONFL[15] DMAR bit. The DMA read request line is deasserted on read completion of the receiver register of the McSPI channel.

The DMA write request line is asserted when the McSPI channel is enabled and the transmitter register of the McSPI channel is empty. A DMA write request can be individually masked with the CHxCONFL[14] DMAW bit. The DMA write request line is deasserted on load completion of the transmitter register of the channel.

9.4.6 Power Saving Management

Power consumption can be optimized by switching off internal clocks (McSPI bridge and functional clock) when there is no activity. The McSPI supports two modes of operations from a power management perspective: normal mode and idle mode.

9.4.6.1 Normal Mode

In normal mode, the internal McSPI bridge clocks are automatically switched off (autogating) when there is no activity in slave or master mode.

Autogating of the McSPI bridge clock and functional clock occurs when the following conditions are met:

- The SYSCONFIGL[0] AUTOIDLE bit is set (default setting).
- In master mode, there is no data to transmit or receive in all channels.
- In slave mode, the McSPI is not selected by the external master and there are no register accesses.

Autogating of the McSPI bridge clock and functional clock stops when the following conditions are met:

- In master mode, an internal access occurs.
- In slave mode, an internal access occurs or the McSPI is selected by the external master.

9.4.6.2 Idle Mode

The McSPI bridge clock and functional clock may be switched off on a system clock stop request and switched back on a module request. McSPI works closely with the system Peripheral Clock Stop Request/Acknowledge Register according to the programmable mode in the SYSCONFIGL[4:3] SIDLEMODE bits to support: No idle mode, force idle mode, and smart idle mode.

- When programmed for no idle mode (the SIDLEMODE is set to “01”), the module ignores the Peripheral Clock Stop Request and behaves normally, as if the request was not asserted.
- When programmed for force idle mode (the SIDLEMO is set to “00”), the module acknowledges the Peripheral Clock Stop Request unconditionally.
- When programmed for smart idle mode (the SIDLEMODE is set to “10”), the module acknowledges the Peripheral Clock Stop Request according to its internal state.

The McSPI bridge clock will be switched off, during the smart idle mode period, if the McSPI Bridge Clock Stop Request bit of the system Peripheral Clock Stop Request/Acknowledge Register is set and the SYSCONFIGL [9:8] CLOCKACTIVITY bits is enabled.

The McSPI clock will be switched off, during the smart idle mode period, if the McSPI Clock Stop Request bit of the Peripheral Clock Stop Request/Acknowledge Register is set and the SYSCONFIGL [9:8] CLOCKACTIVITY bits is enabled.

9.4.6.2.1 Force-Idle Mode

Force-idle mode is enabled and exited as follows:

- Force-idle mode is enabled when the SYSCONFIGL[4:3] SIDLEMODE field is set to 0x0.

  In force-idle mode, McSPI responds unconditionally to the idle request by deasserting unconditionally the interrupt and DMA request lines, if asserted.

  The transition from normal mode to idle mode does not affect the interrupt event bits of the IRQSTATUSL and IRQSTATUSU registers.

  In force-idle mode, the module must be disabled so the interrupt and DMA request lines are likely deasserted. The McSPI bridge clock and SPI clock provided to the McSPI can be switched off.

  An idle request during an SPI data transfer can lead to an unexpected and unpredictable result. The software must avoid such a request.
- The module exits force-idle mode through the idle and wake-up hardware handshake protocol.

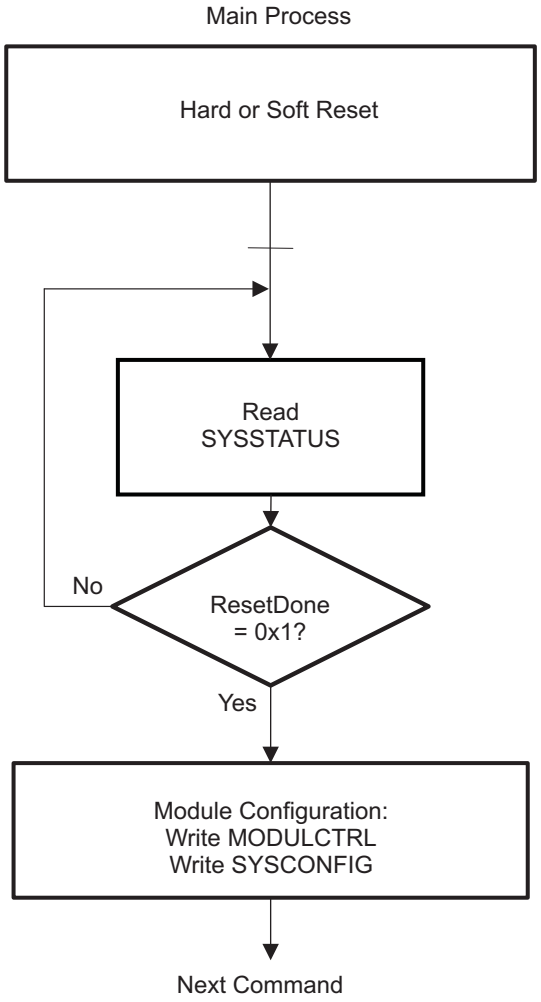
  The module is fully operational. The interrupt and DMA request lines are optionally asserted one clock cycle later.

9.5 McSPI Basic Programming Model

9.5.1 Initialization of Modules

Figure 9-22 shows the overview of the module initialization flow process. Section 9.5.2 and Section 9.5.3 show the steps required to configure McSPI modes.

Figure 9-22. Module Initialization Flow



NOTE: Before the SYSSTATUSL[0] RESETDONE bit is set, the CLK and CLKSPIREF clocks must be provided to the module.

To avoid unpredictable behavior, reset the module before changing from master mode to slave mode, or vice versa.

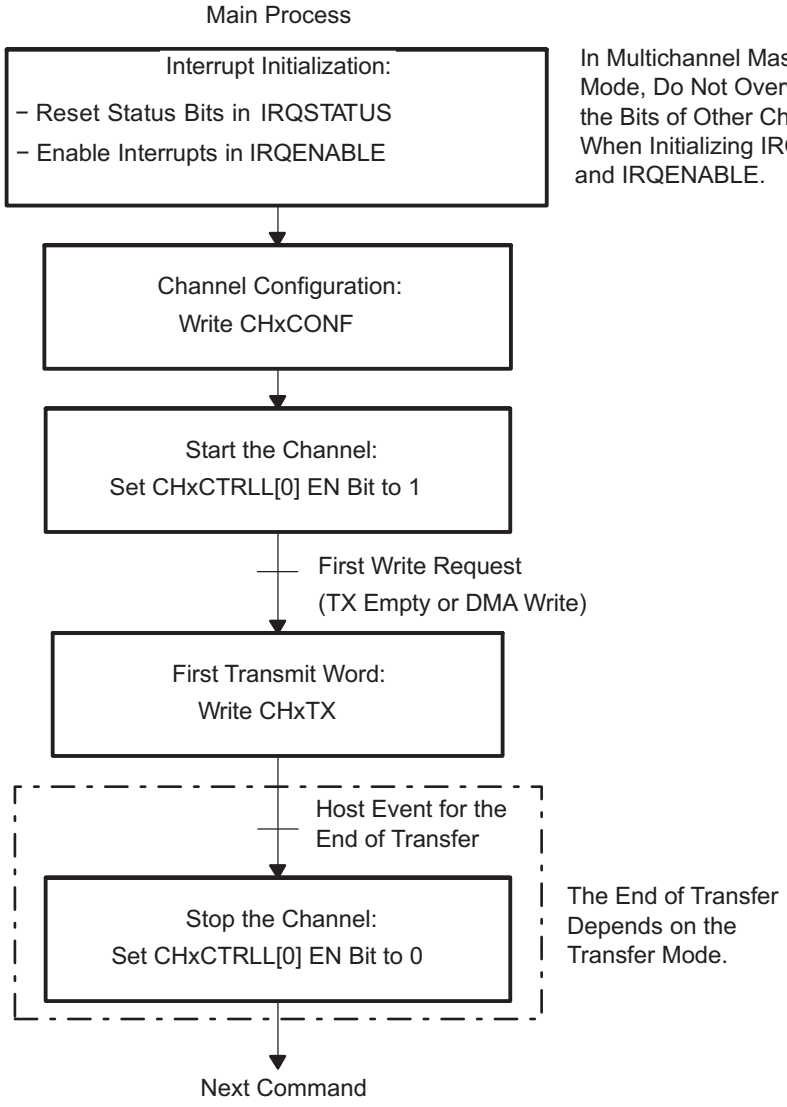
9.5.2 Transfer Procedures without FIFO

In the following subsections, the transfer procedures are described without FIFO using (CHxCONFU[12:11] FFER and FFEW = 0).

9.5.2.1 Common Transfer Procedure

Figure 9-23 shows the main sequence common to all transfers. In multichannel master mode, the flows of different channels can be run simultaneously.

Figure 9-23. Common Transfer Sequence: Main Process



9.5.2.2 End-of-Transfer Procedure

For transfers carried out with DMA or interrupt mode, the end of transfer must be achieved by following the steps shown in the flowcharts corresponding to Table 9-6, which summarizes the end-of-transfer types per transfer mode and provides cross-references for further information.

Table 9-6. End-of-Transfer Sequences

		Transmit Receive		Transmit Only		Receive Only	
		Interrupt	DMA	Interrupt	DMA	Interrupt	DMA
Master normal	End of transfer sequence	See Section 9.5.2.3		See Section 9.5.2.4	See Section 9.5.2.4	See Section 9.5.2.5.1	See Section 9.5.2.5.1
	Minimum number of words	1	1	1	1	1	1
	DMA transfer size[(1)]		w		w		w – 1

(1) w = number of words to transfer

Table 9-6. End-of-Transfer Sequences (continued)

		Transmit Receive		Transmit Only		Receive Only	
		Interrupt	DMA	Interrupt	DMA	Interrupt	DMA
Master turbo	End of transfer sequence	See Section 9.5.2.3		See Section 9.5.2.4	See Section 9.5.2.4	See Section 9.5.2.5.2	See Section 9.5.2.5.2
	Minimum number of words	1	1	1	1	2	3
	DMA transfer size[1]		w		w		w – 2
Slave	End of transfer sequence	See Section 9.5.2.3		See Section 9.5.2.4	See Section 9.5.2.4	See Section 9.5.2.5.3	
	Minimum number of words	1	1	1	1	1	1
	DMA transfer size[1]		w		w	w	w

The different sequences can be merged in one process to manage transfers of several types. The end-of-transfer sequences are described from the start of the channel.

In these sequences and in later sections of this chapter, some software variables are used:

- WRITE_COUNT (= 0 at initialization): Contains the number of words to transfer
- READ_COUNT (= 0 at initialization): Contains the number of words to receive
- CHANNEL_ENABLE (= false at initialization)
- LAST_TRANSFER (= false at initialization): Indicates that the last word is in transmission
- LAST_REQUEST (= false at initialization): Indicates that the last request is in progress

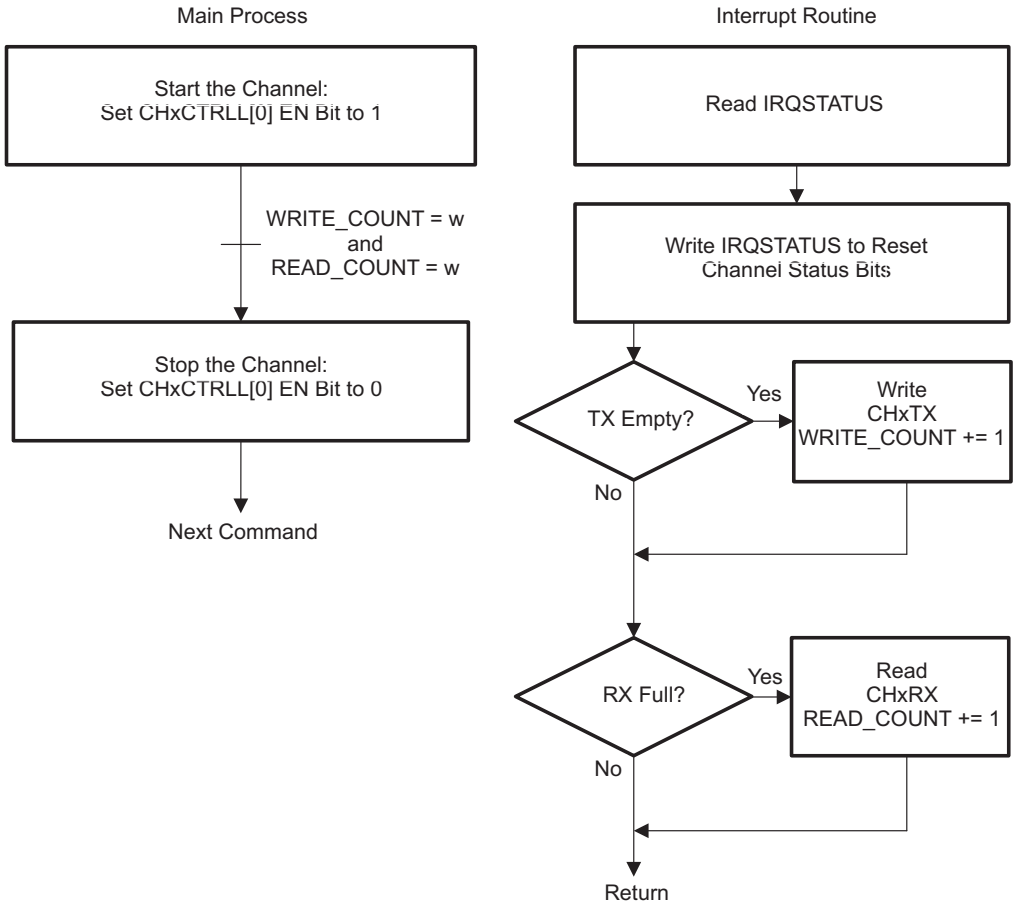
All variables are initialized before starting the channel.

9.5.2.3 Transmit and Receive Procedure

Figure 9-24 shows the handling procedure for words received and transmitted by interrupt in master and slave modes. The main process flow shows how the end of the transfer must be done after all words are received for this mode.

If the requests are configured in DMA, WRITE_COUNT and READ_COUNT are assigned with the value w when the DMA handler completes w interface accesses.

Figure 9-24. Transmit and Receive (Master and Slave)

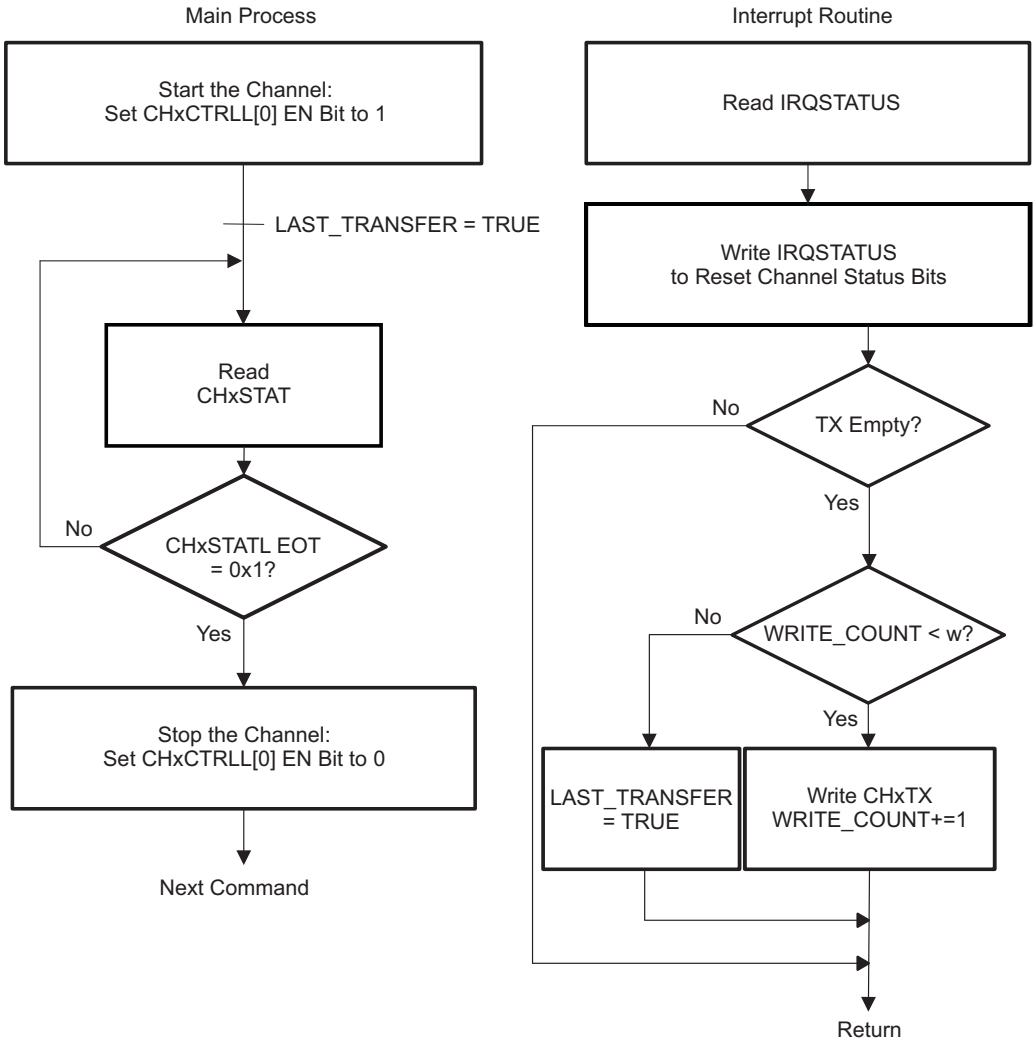


9.5.2.4 Transmit-Only Procedure

9.5.2.4.1 Based on Interrupt Requests

Figure 9-25 shows the handling procedure for words transmitted by interrupt in transmit-only mode. The main process flow shows how the end-of-transfer must be done after all words are received for this mode.

Figure 9-25. Transmit-Only With Interrupts (Master and Slave)

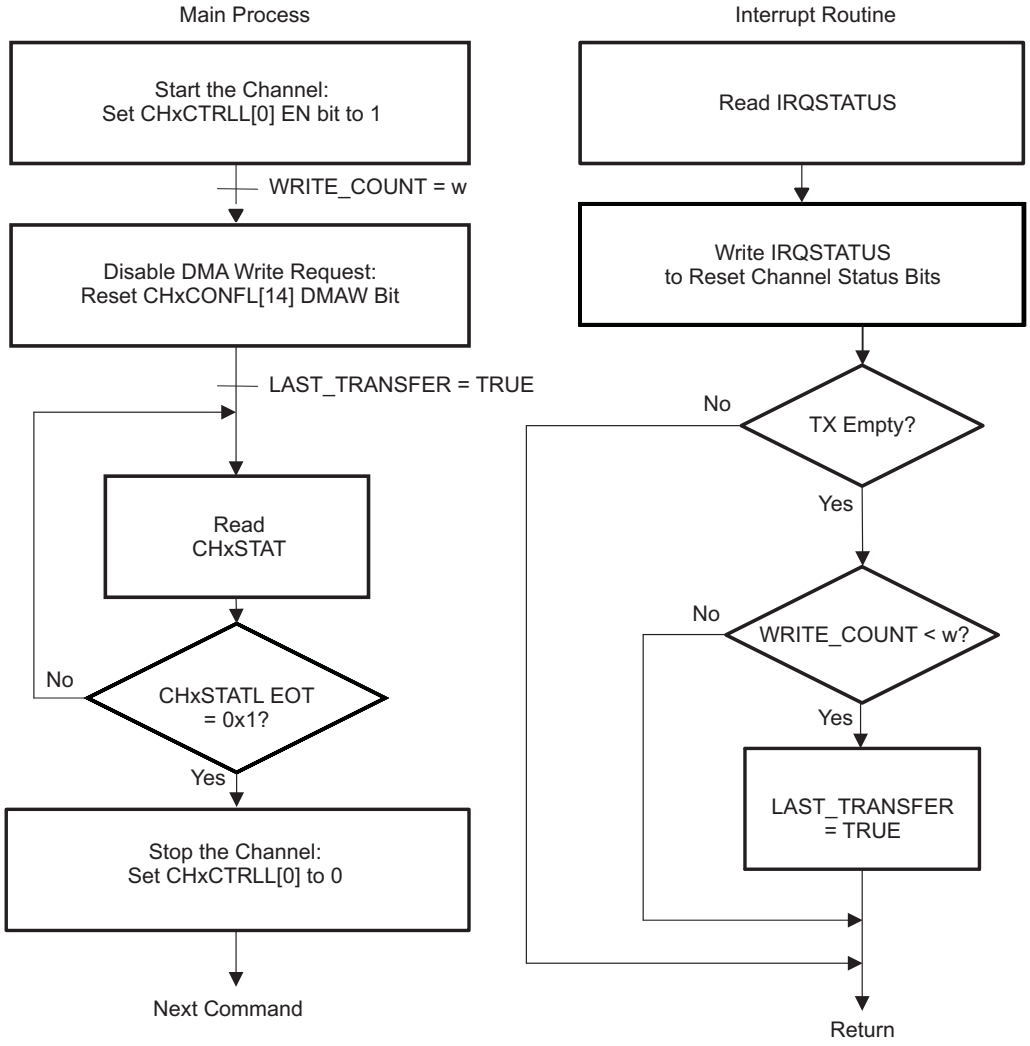


9.5.2.4.2 Transmit-Only Based on DMA Write Requests

In Figure 9-26, the main process shows completion of the transfer of words in transmit-only mode with DMA write requests.

When the DMA handler completes w interface accesses, WRITE_COUNT is assigned the value *w*.

Figure 9-26. Transmit-Only With DMA (Master and Slave)



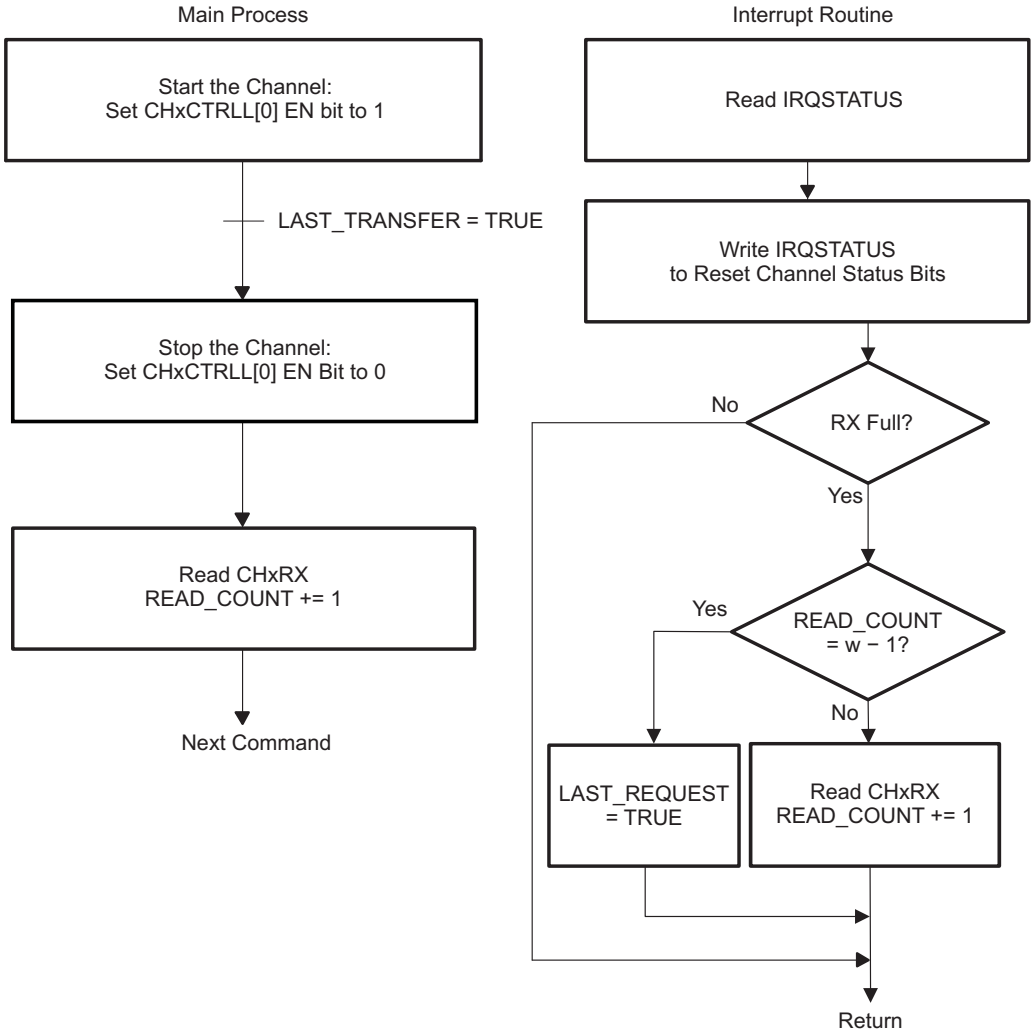
9.5.2.5 Receive-Only Procedure

9.5.2.5.1 Master Normal Receive-Only Procedure

9.5.2.5.1.1 Based on Interrupt Requests

Figure 9-27 shows the handling procedure for words received by interrupt in master normal receive-only mode. The main process flow shows how the end-of-transfer must be done after all words are received for this mode.

Figure 9-27. Receive-Only With Interrupt (Master Normal)

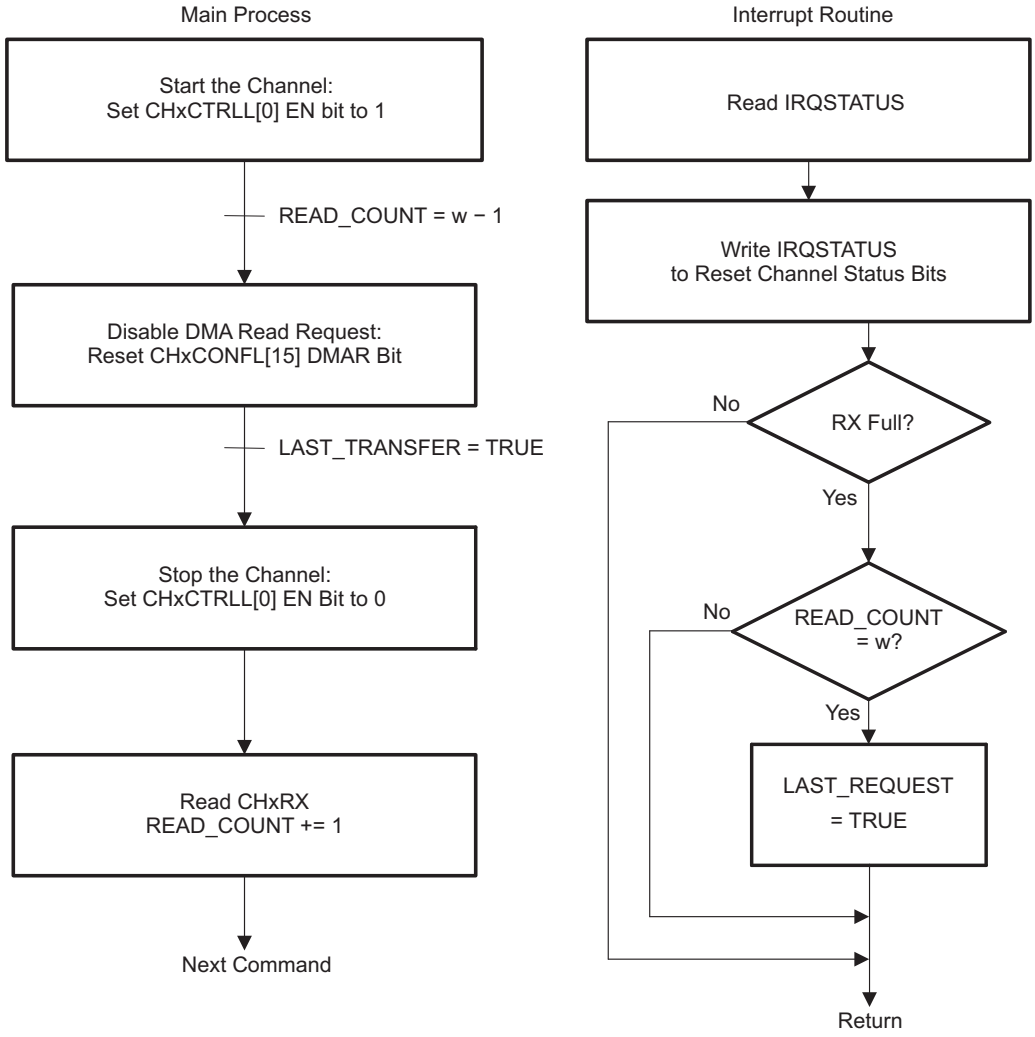


9.5.2.5.1.2 Receive-Only Based on DMA Read Requests

In Figure 9-28, the main process shows the completion of a word transfer in receive-only mode with DMA read requests.

When the DMA handler completes *w* - 1 interface accesses, READ_COUNT is assigned the value *w* - 1.

Figure 9-28. Receive-Only With DMA (Master Normal)

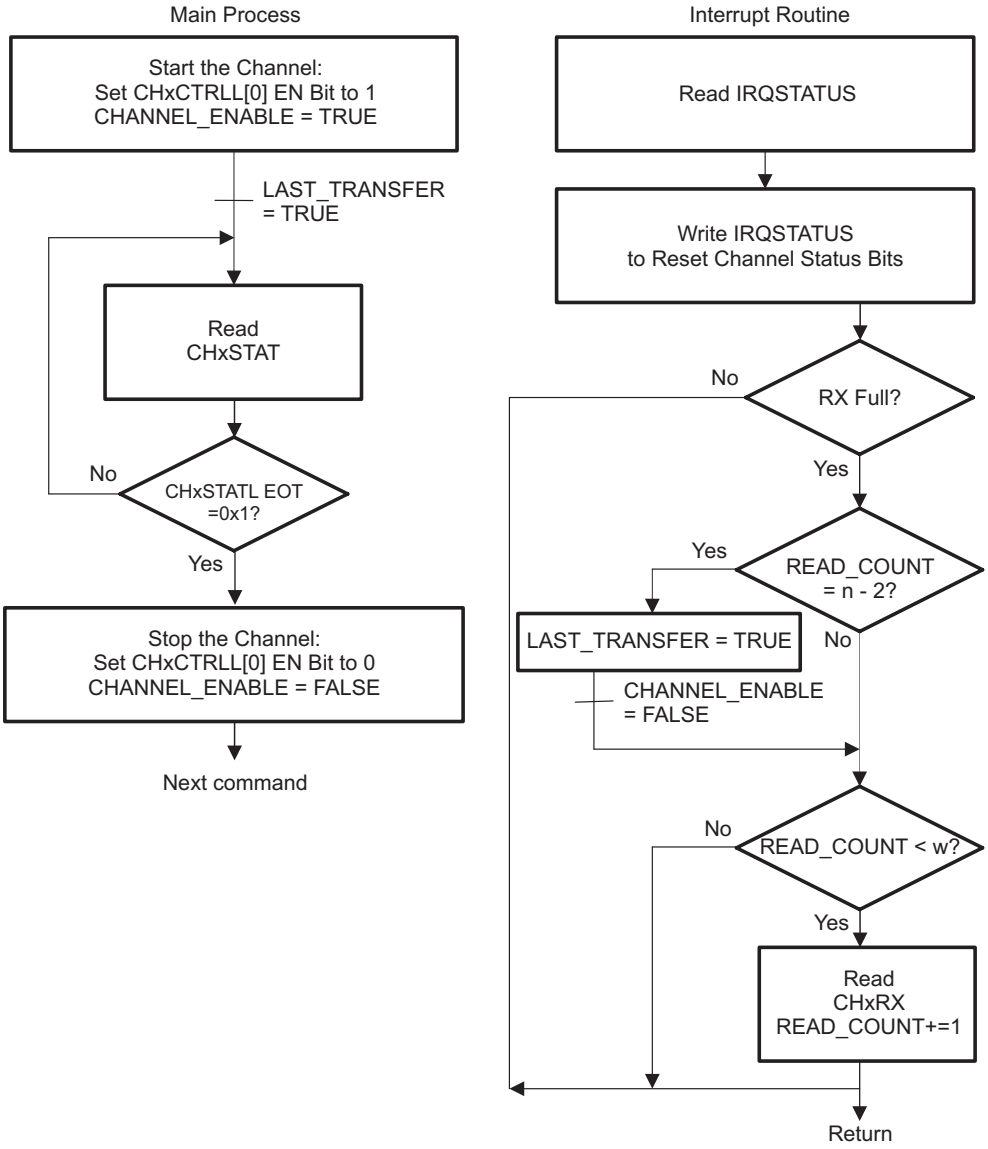


9.5.2.5.2 Master Turbo Receive-Only Procedure

9.5.2.5.2.1 Based on Interrupt Requests

Figure 9-29 shows the handling procedure for words received by interrupt in master turbo receive-only mode. The main process shows how the end-of-transfer must be done after all words are received for this mode.

Figure 9-29. Receive-Only With Interrupt (Master Turbo)

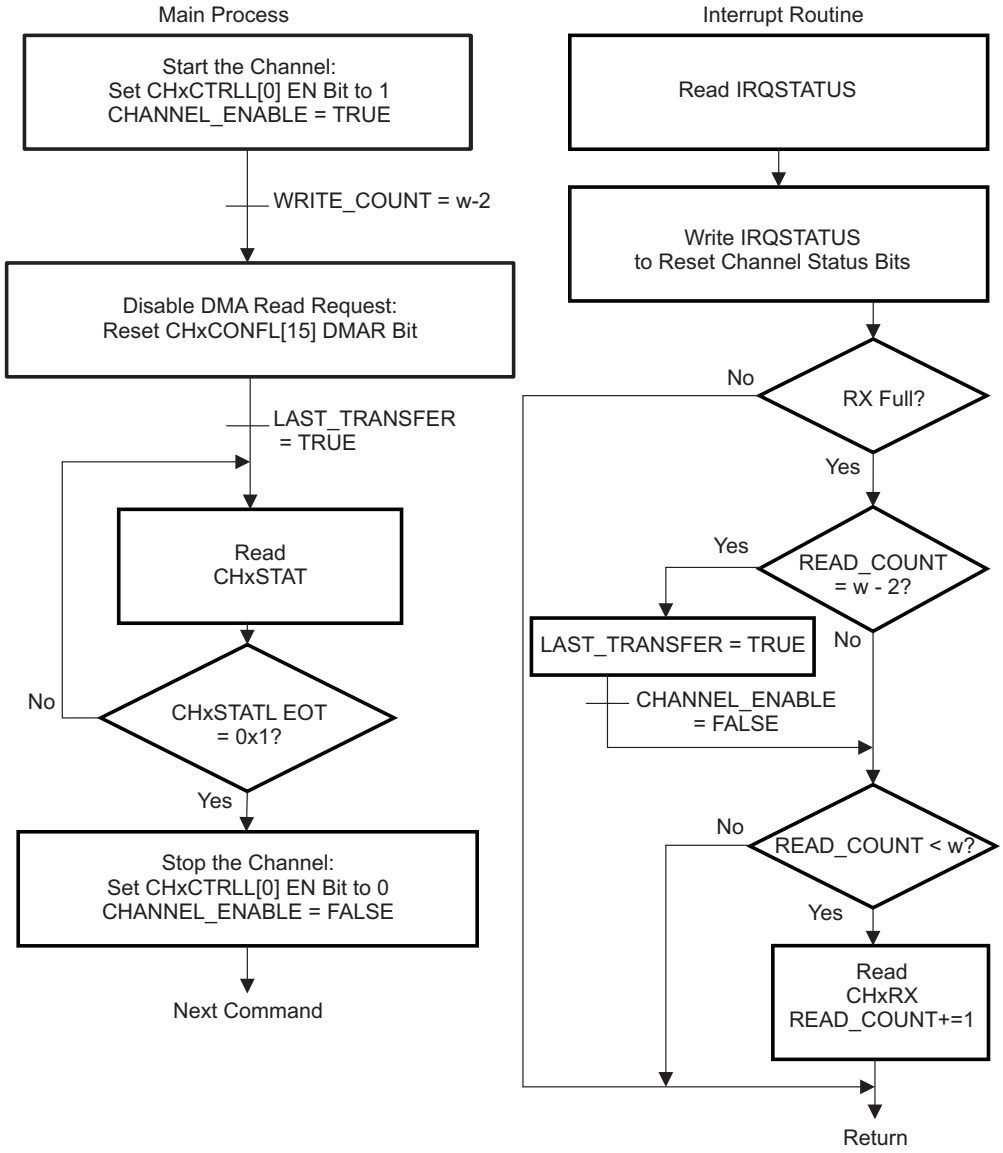


9.5.2.5.2.2 Based on DMA Read Requests

In Figure 9-30, the main process shows the completion of a word reception in master turbo receive-only mode with DMA write requests.

When the DMA handler completes *w* - 2 interface accesses, READ_COUNT is assigned the value *w* - 2.

Figure 9-30. Receive-Only With DMA (Master Turbo)

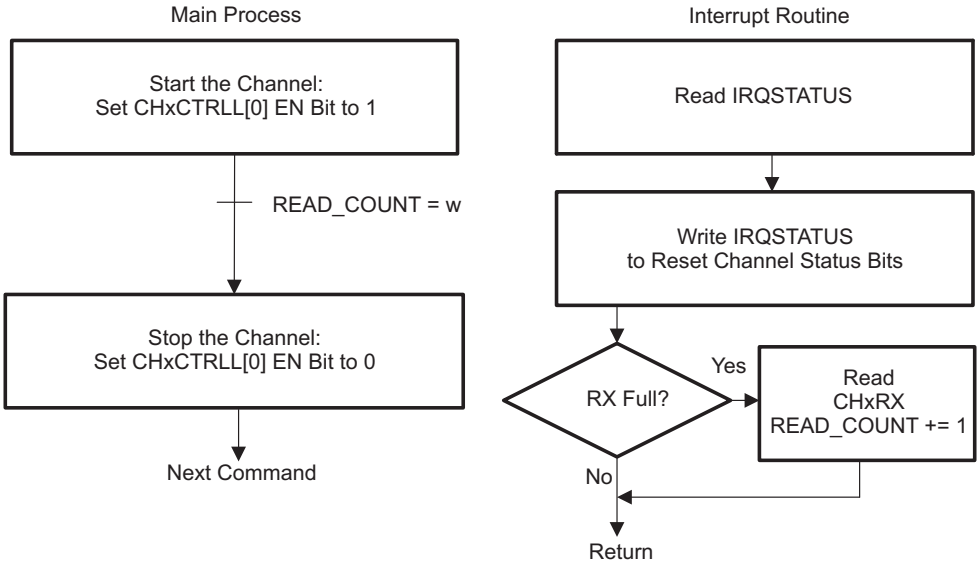


9.5.2.5.3 Slave Receive-Only Procedure

Figure 9-31 shows the handling procedure for words received by interrupt in slave receive-only mode. The main process shows how the end-of-transfer must be done after all words are received for this mode.

If the requests are configured in DMA, READ_COUNT is assigned the value *w* when the DMA handler completes *w* interface processes.

Figure 9-31. Receive-Only (Slave)



9.5.2.6 McSPI Configuration and Operations Example

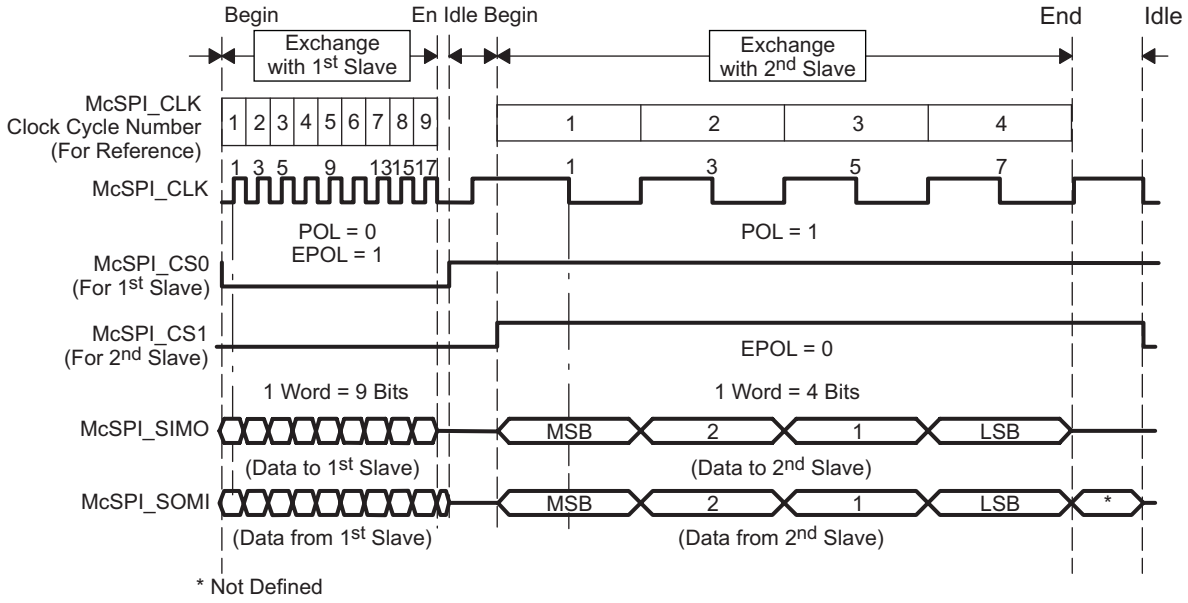
This section details an example of a typical configuration.

Figure 9-32 shows an implementation of successive transfers between two devices (slave) and the McSPI (master) in transmit-and-receive mode.

McSPI is the master, and McSPI_SIMO shifts out the data to the external slaves. McSPI_SOMI is connected to the data output port of two slave devices. The McSPI controls the first device with the McSPI_CS0 signal (active low) for the exchange of 9-bit words synchronized with an active-high SPI clock.

The second device is controlled by the McSPI_CS1 control signal (active high) for the exchange of 4-bit words synchronized with a SPI clock (active low) with a slower frequency than used with the first slave device.

Figure 9-32. Two SPI Transfers With PHA = 0 (Flexibility of McSPI)



This section details the steps required for this type of transmission.

9.5.2.6.1 McSPI Initialization Sequence

As shown in Figure 9-22, the McSPI module must first reset all registers and all state-machines.

For a software reset:

1. Set the SYSCONFIGL[1] SOFTRESET bit to 1.
2. Read the SYSSTATUSL[0] RESETDONE bit and check that it is set to 1.

9.5.2.6.2 Operations for the First Slave (On Channel 0)

9.5.2.6.2.1 Programming in Polling Mode

####### 9.5.2.6.2.1.1 Mode Selection

The CH0CONF register allows configuration of the operating mode:

1. Set the CH0CONFU[2] IS bit to 0 for the McSPI_SOMI pin in receive mode.
2. Set the CH0CONFU[1] DPE1 bit to 0 and the CH0CONFU[0] DPE0 bit to 1 for the McSPI_SIMO pin in transmit mode.
3. Set the CH0CONFL[13:12] TRM field to 0x0 for transmit and receive mode.

4. Write 0x8 in the CH0CONFL[11:7] WL field for 9-bit word length.
5. Set the CH0CONFL[6] EPOL bit to 1 for McSPI_CS0 activated low during active state.
6. Set the CH0CONFL[1] POL bit to 0 for McSPI_CLK held high during active state.
7. Set the CH0CONFL[0] PHA bit to 0 for data latched on odd-numbered edges of the SPI clock.

Clock Initialization and McSPI_CS0 Enable

In master mode, the SPI must provide the clock and enable the channel:

8. Set the MODULCTRLL[2] MS bit to 0 to provide the clock.
9. Set the CH0CTRLL[0] EN bit to 1 to enable channel 0.

9.5.2.6.2.1.2 Write Operation

1. Write 1 to the IRQSTATUSL[0] TX0_EMPTY bit to reset the status.
2. Write the command/address or data value in the transmitter register to transmit the value.
3. If the IRQSTATUSL[0] TX0_EMPTY bit is set to 1, write 1 to it and return to Step 2 (polling method).

9.5.2.6.2.1.3 Read Operation

1. Read the IRQSTATUSL[2] RX0_FULL bit and if it is set to 1, go to Step 2.
2. If the IRQSTATUSL[2] RX0_FULL bit is set to 1, write 1 to it and return to Step 1 (polling method).

NOTE: Write and read operations can be performed simultaneously.

9.5.2.6.3 Programming in Interrupt Mode

This section follows the flow of Figure 9-22.

1. Initialize software variables: WRITE_COUNT = 0 and READ_COUNT = 0.
2. Initialize interrupts: Write 0x7 in the IRQSTATUSL[3:0] field and set the IRQENABLEL[3:0] field to 0x7.
3. Follow the steps described in Section 9.5.2.6.2.1.1, *Mode Selection*.
4. If WRITE_COUNT = w and READ_COUNT = w, write CHxCTRLL[0] EN to 0 to stop the channel.

This interrupt routine follows the flow of Table 9-6 and Figure 9-24.

1. Read the IRQSTATUSL[3:0] field.
2. If the IRQSTATUSL[0] TX0_EMPTY bit is set to 1:
   (a) Write the command/address or data value in the transmitter register.
   (b) WRITE_COUNT + = 1
   (c) Write IRQSTATUSL[0] = 0x1.
3. If the IRQSTATUSL[2] RX0_FULL bit is set to 1:
   (a) Read the receiver registers
   (b) READ_COUNT += 1
   (c) Write IRQSTATUSL[2] = 0x1

9.5.2.6.4 Operations for the Second Slave (on Channel 1) in Polling Mode

9.5.2.6.4.1 Mode Selection

The CH1CONF register allows configuration of the operating mode:

1. Set the CH1CONFU[2] IS bit to 0 for the McSPI_SOMI pin in receive mode.
2. Set the CH1CONFU[1] DPE1 bit to 0 and the CH1CONF[0] DPE0 bit to 1 for the McSPI_SOMI pin in transmit mode.
3. Set the CH1CONFL[13:12] TRM field to 0x0 for transmit-and-receive mode.
4. Write 0x3 in the CH1CONFL[11:7] WL field.

5. Set the CH1CONFL[6] EPOL bit to 0 for McSPI_CS1 activated high during the active state.
6. Set the CH1CONFL[1] POL bit to 1 for McSPI_CLK held low during the active state.
7. Set the CH1CONFL[0] PHA bit to 1 for data latched on even numbered edges of McSPI_CLK.

   Clock Initialization and McSPI_CS1 Enable

   The MODULCTRLL[2] MS bit was set to 0 (providing the clock).

   In master mode, the SPI must provide the clock and enable the channel:
8. Set the CH0CTRL[0] EN bit to 0 to disable Channel 0, and set the CH1CTRL[0] EN bit to 1 to enable Channel 1.

NOTE: Read and write operations for the second slave are identical to those for the first slave.

9.5.2.6.4.2 Write Operation

1. Write 1 to the IRQSTATUSL[4] TX1_EMPTY bit to reset the status.
2. Write the command/address or data value in the transmitter register to transmit the value.
3. If the IRQSTATUSL[4] TX1_EMPTY bit is set to 1, write 1 to it and return to Step 2 (polling method).

9.5.2.6.4.3 Read Operation

1. Read the IRQSTATUSL[6] RX1_FULL bit and if it is set to 1, go to Step 2.
2. If the IRQSTATUSL[6] RX1_FULL bit is set to 1, write 1 to it and return to Step 1 (polling method).

NOTE: Write and read operations can be performed simultaneously.

9.5.3 Transfer Procedures with FIFO

In the subsections below, the transfer procedures are described with FIFO using (CHxCONFU[12:11] FFER and/or FFEW = 1).

The MCSPI module allows the transfer of one or more words, according to different modes:

- Master normal, master turbo, slave
- Transmit-and-receive, transmit-only, receive-only
- Write and read requests: Interrupts, DMA

For these flows, the host process contains the main process and the interrupt routine, which is called on the IRQ signals or by an internal call if the module is used in polling mode.

9.5.3.1 Common Transfer Procedure

The common transfer sequence is the host sequence for a transfer of any type defined above.

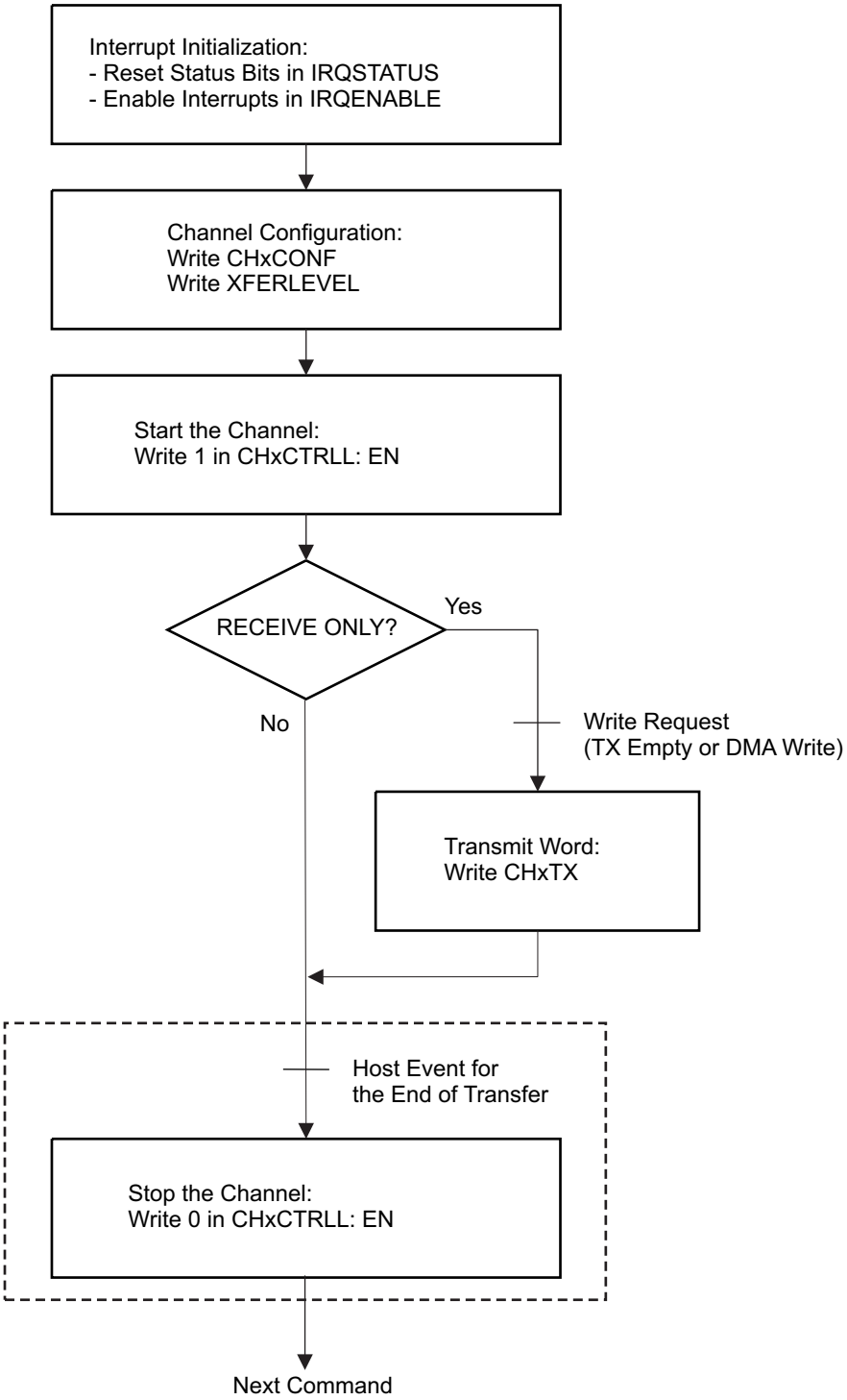
In multichannel, only one channel can use the FIFO. Before enabling the FIFO for a channel (FFEW and FFER bits in the CHx_CONFU register), the host must ensure that the FIFO is not enabled for another channel, even if these channels are not used.

In transmit/receive mode, the FIFO can be enabled for write or read request only, without FIFO for the other request.

In slave mode, only Channel 0 only can be activated by writing a 0 to the CH0CONFU[6:5] SPICSSLV bits. Since the default value is 0 already, it is not required to write a 0 to activate.

The MCSPI module can start the transfer only when the first write request is released by writing the transmitter register, even in receive-only mode (only one write request occurs in this case). Figure 9-33 shows the main process of the common transfer sequence.

Figure 9-33. FIFO Mode Common Transfer Sequence/Main Process (Master)



The MCSPI module can start the transfer only after the first write request is released by writing the transmitter register, even in receive-only mode (only one write request occurs in this case).

This first write request can be managed by the IRQ routine or DMA handler, as are as other requests. It appears in Figure 9-33 to show this point.

The end of the transfer (dotted line in Figure 9-33) is more complex and depends on the transfer type. Table 9-7 describes the different types of end-of-transfer and the transfer types to which they are applicable.

Table 9-7. End-of-Transfer Types

Word Count	Transmit/Receive	Transmit Only	Receive Only
Yes	See Section 9.5.3.2, *Transmit-Receive With Word Count*.	See Section 9.5.3.4, *Transmit-Only*.	See Section 9.5.3.5 , *Receive-Only With Word Count*.
No	See Section 9.5.3.3, *Transmit-Receive Without Word Count*.	See Section 9.5.3.4, *Transmit-Only*.	See Section 9.5.3.6, *Receive-Only Without Word Count*.

The sequence differs depending on whether word count is used (XFERLEVELU:WCNT is set or not). The AEL and/or AFL values can be different, but they must be multiples of the word size in the FIFO: 1, 2, or 4 bytes, according to word length.

In these sequences, the transfer to execute has a size of N words.

When accessing the FIFO, only one word is written or read for each internal module access.

In these sequences, the number of words written or read for each write or read FIFO request is:

- write_request_size
- read_request_size

If they are not submultiples of N, the last request sizes are:

- last_write_request_size (< write_request_size)
- last_read_request_size. (< read_request_size)

The different sequences can be merged in one process to manage transfers of several types.

The end-of-transfer sequences are described from the start of the channel.

In these sequences, some soft variables are used:

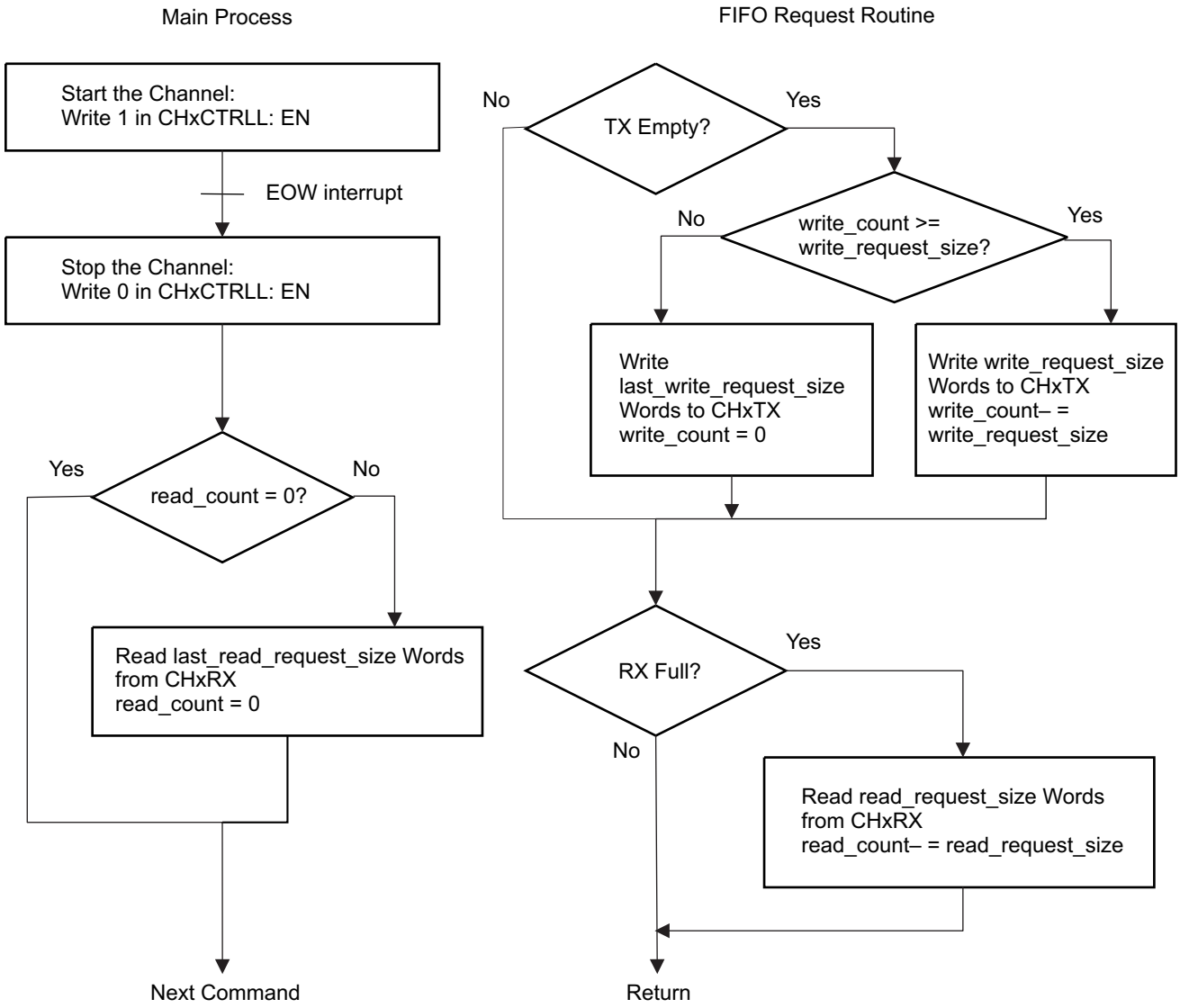
- write_count = N
- read_count = N
- last_request = FALSE

They are initialized before starting the channel.

9.5.3.2 Transmit-Receive Procedure With Word Count (WCNT≠0)

Figure 9-34 shows the flow of a transfer in transmit-receive mode, with word count.

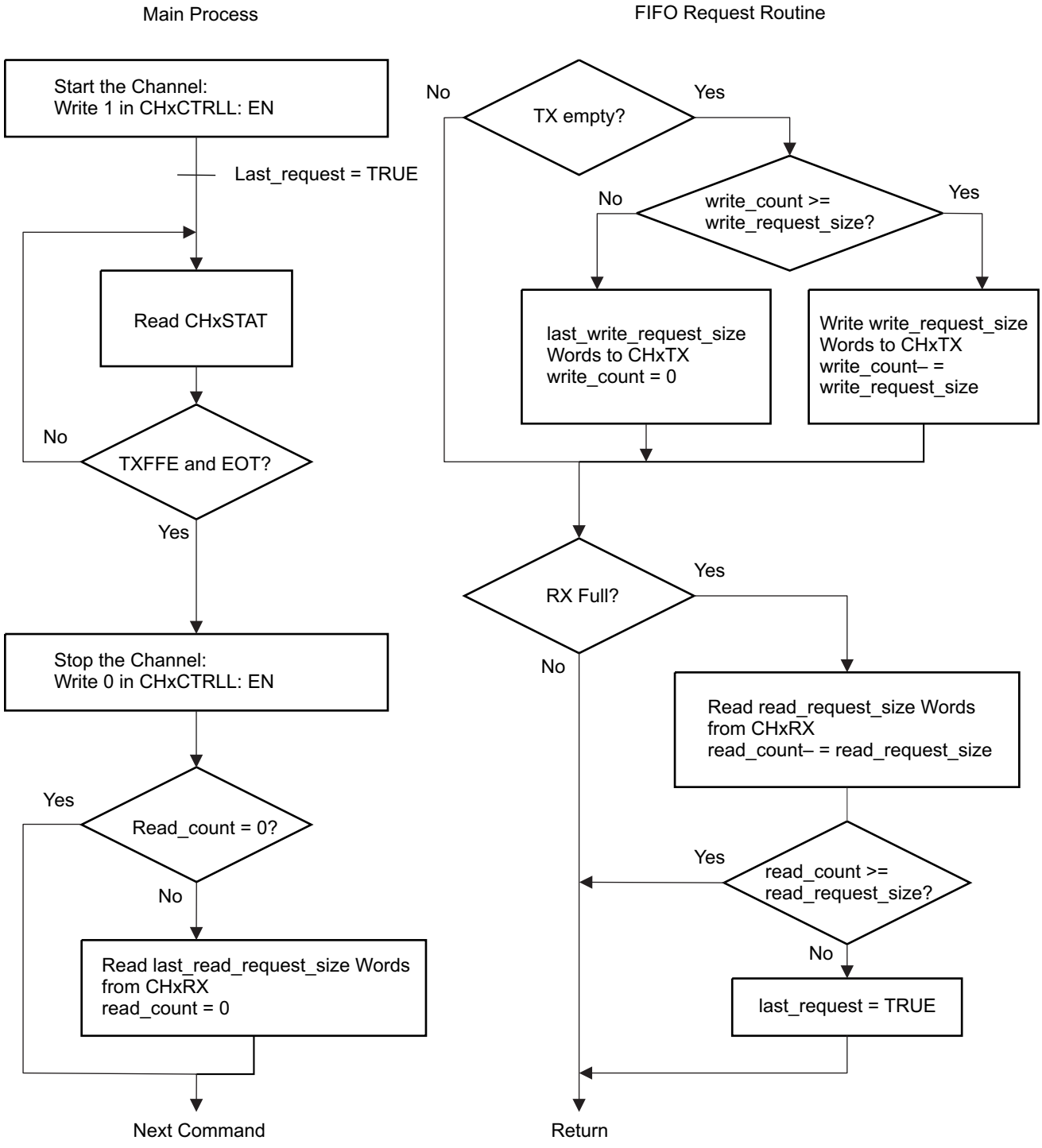
Figure 9-34. FIFO Mode Transmit-Receive With Word Count (Master)



9.5.3.3 Transmit-Receive Procedure Without Word Count (WCNT=0)

Figure 9-35 shows the flow of a transfer in transmit-receive mode, without word count.

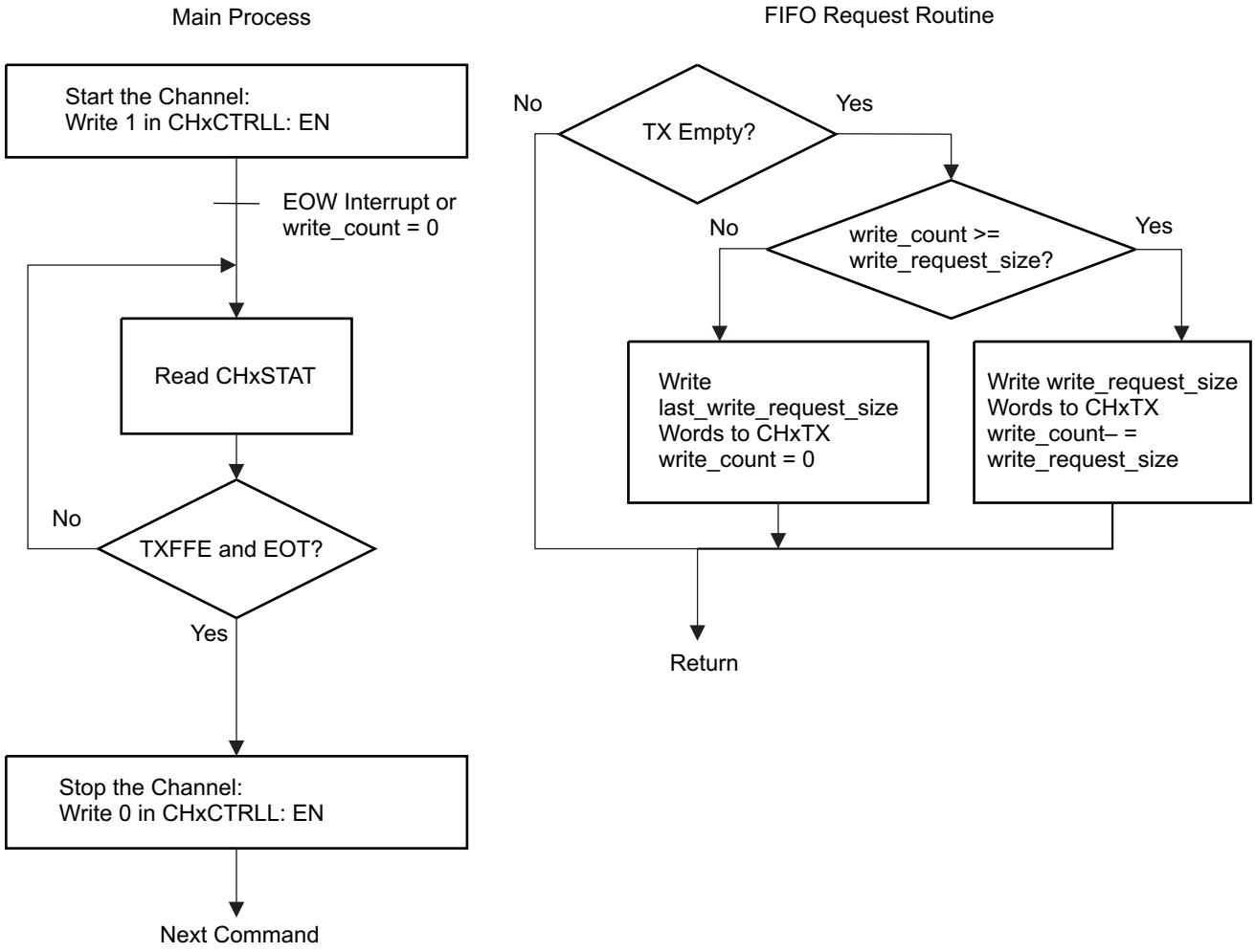
Figure 9-35. FIFO Mode Transmit-Receive Without Word Count (Master)



9.5.3.4 Transmit-Only Procedure

Figure 9-36 shows the flow of a transfer in transmit only mode, with our without word count.

Figure 9-36. FIFO Mode Transmit-Only (Master)



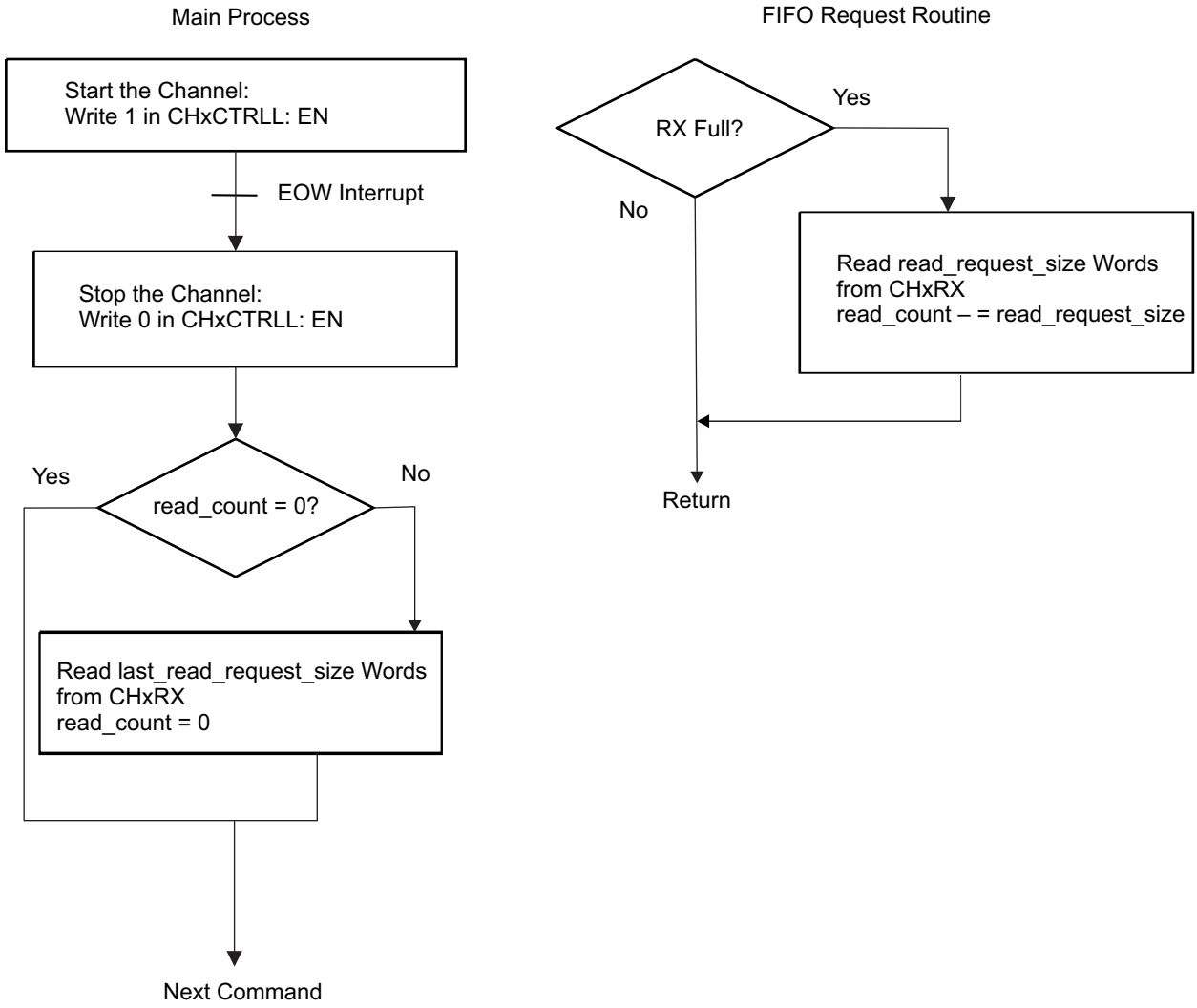
The difference between word count enabled or not is the condition after starting the channel:

- Word count enable: Wait for EOW interrupt.
- Word count disable: Wait for write_count = 0.

9.5.3.5 Receive-Only Procedure With Word Count (WCNT≠0)

Figure 9-37 shows the flow of a transfer in receive-only mode, with word count.

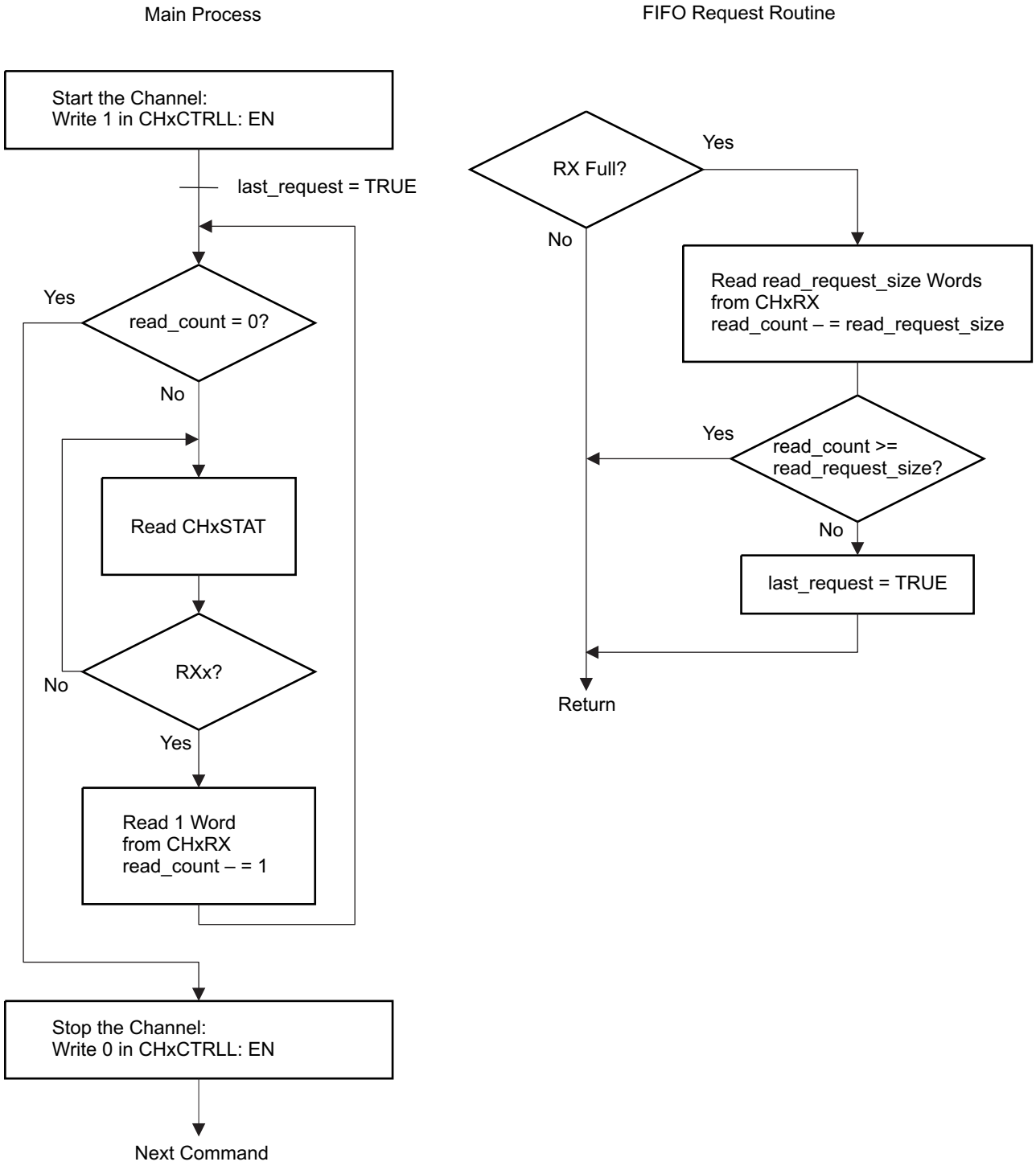
Figure 9-37. FIFO Mode Receive-Only With Word Count (Master)



9.5.3.6 Receive-Only Procedure Without Word Count (WCNT=0)

Figure 9-38 shows the flow of a transfer in receive-only mode, without word count.

Figure 9-38. FIFO Mode Receive-Only Without Word Count (Master)



9.6 McSPI Registers

Table 9-8 lists the memory-mapped registers for the MCSPI. All register offset addresses not listed in Table 9-8 should be considered as reserved locations and the register contents should not be modified.

Table 9-8. McSPI REGISTERS

CPU Word Address	Acronym	Register Name	Section
3500h	REVISIONL	Revision Register Lower	Section 9.6.1
3510h	SYSCONFIGL	System Configuration Register Lower	Section 9.6.2
3514h	SYSSTATUSL	System Status Register Lower	Section 9.6.3
3518h	IRQSTATUSL	Interrupt Status Register Lower	Section 9.6.4
3519h	IRQSTATUSU	Interrupt Status Register Upper	Section 9.6.5
351Ch	IRQENABLEL	Interrupt Enable Register Lower	Section 9.6.6
351Dh	IRQENABLEU	Interrupt Enable Register Upper	Section 9.6.7
3520h	WAKEUPENABLEL	Wakeup Enable Register Lower	Section 9.6.8
3528h	MODULCTRLL	Module Control Register Lower	Section 9.6.9
352Ch	CH0CONFL	Channel 0 Configuration Register Lower	Section 9.6.10
352Dh	CH0CONFU	Channel 0 Configuration Register Upper	Section 9.6.11
3530h	CH0STATL	Channel 0 Status Register Lower	Section 9.6.12
3534h	CH0CTRLL	Channel 0 Control Register Lower	Section 9.6.13
3538h	CH0TXL	Channel 0 Transmitter Register Lower	Section 9.6.14
3539h	CH0TXU	Channel 0 Transmitter Register Upper	Section 9.6.15
353Ch	CH0RXL	Channel 0 Receiver Register Lower	Section 9.6.16
353Dh	CH0RXU	Channel 0 Receiver Register Lower	Section 9.6.17
3540h	CH1CONFL	Channel 1 Configuration Register Lower	Section 9.6.18
3541h	CH1CONFU	Channel 1 Configuration Register Upper	Section 9.6.19
3544h	CH1STATL	Channel 1 Status Register Lower	Section 9.6.20
3548h	CH1CTRLL	Channel 1 Control Register Lower	Section 9.6.21
354Ch	CH1TXL	Channel 1 Transmitter Register Lower	Section 9.6.22
354Dh	CH1TXU	Channel 1 Transmitter Register Lower	Section 9.6.23
3550h	CH1RXL	Channel 1 Receiver Register Lower	Section 9.6.24
3551h	CH1RXU	Channel 1 Receiver Register Lower	Section 9.6.25
3554h	CH2CONFL	Channel 2 Configuration Register Lower	Section 9.6.26
3555h	CH2CONFU	Channel 2 Configuration Register Upper	Section 9.6.27
3558h	CH2STATL	Channel 2 Status Register Lower	Section 9.6.28
355Ch	CH2CTRLL	Channel 2 Control Register Lower	Section 9.6.29
3560h	CH2TXL	Channel 2 Transmitter Register Lower	Section 9.6.30
3561h	CH2TXU	Channel 2 Transmitter Register Upper	Section 9.6.31
3564h	CH2RXL	Channel 2 Receiver Register Lower	Section 9.6.32
3565h	CH2RXU	Channel 2 Receiver Register Upper	Section 9.6.33
357Ch	XFERLEVELL	Transfer Levels Register Lower	Section 9.6.34
357Dh	XFERLEVELU	Transfer Levels Register Upper	Section 9.6.35
3580h	DAFTXL	DMA Address Aligned FIFO Transmitter Register Lower	Section 9.6.36
3581h	DAFTXU	DMA Address Aligned FIFO Transmitter Register Upper	Section 9.6.37
35A0h	DAFRXL	DMA Address Aligned FIFO Receiver Register Lower	Section 9.6.38
35A1h	DAFRXU	DMA Address Aligned FIFO Receiver Register Upper	Section 9.6.39

9.6.1 REVISIONL Register (offset = 3500h) [reset = 2Bh]

REVISIONL is shown in Figure 9-39 and described in Table 9-9.

This register contains the hard coded RTL revision number

Figure 9-39. REVISIONL Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
REV							
R-2Bh							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-9. REVISIONL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved. Reads return 0
7-0	REV	R	2Bh	IP revision [7:4] is Major revision [3:0] is Minor revision.

9.6.2 SYSCONFIGL Register (offset = 3510h) [reset = 15h]

SYSCONFIGL is shown in Figure 9-40 and described in Table 9-10.

This register allows controlling various parameters of the McSPI bridge. It is not sensitive to soft reset.

Figure 9-40. SYSCONFIGL Register

15	14	13	12	11	10	9	8
Reserved						CLOCKACTIVITY	
R-0h						RW-0h	

7	6	5	4	3	2	1	0
Reserved			SIDLEMODE		ENAWAKEUP	SOFTRESET	AUTOIDLE
R-0h			RW-2h		RW-1h	RW-0h	RW-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-10. SYSCONFIGL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved. Reads return 0
9-8	CLOCKACTIVITY	RW	0h	Clocks activity during wake up mode period 0x0 = McSPI bridge and Functional clocks may be switched off 0x1 = McSPI bridge and Functional clocks may be switched off 0x2 = McSPI bridge and Functional clocks may be switched off 0x3 = McSPI bridge and Functional clocks are maintained
7-5	Reserved	R	0h	Reserved. Reads return 0
4-3	SIDLEMODE	RW	2h	Power management 0x0 = If an idle request is detected, the McSPI acknowledges it unconditionally and goes in Inactive mode. Interrupt, DMA requests and wake up lines are unconditionally de-asserted and the module wakeup capability is deactivated even if the bit MCSPI_SYSCONFIG[EnaWakeUp] is set 0x1 = If an idle request is detected, the request is ignored and the module does not switch to wake up mode, and keeps on behaving normally 0x2 = Smart-idle mode: local target's idle state eventually follows (acknowledges) the system's idle requests, depending on the IP module's internal requirements. IP module shall not generate (IRQ- or DMA-request-related) wakeup events 0x3 = Reserved
2	ENAWAKEUP	RW	1h	WakeUp feature control 0x0 = WakeUp capability is disabled 0x1 = WakeUp capability is enabled
1	SOFTRESET	RW	0h	Software reset. During reads it always returns 0. 0x0 = Normal mode 0x1 = Set this bit to 1 to trigger a module reset.The bit is automatically reset by the hardware
0	AUTOIDLE	RW	1h	Internal McSPI bridge Clock gating strategy 0x0 = McSPI bridge clock is free-running 0x1 = Automatic McSPI bridge clock gating strategy is applied, based on the McSPI bridge activity

9.6.3 SYSSTATUSL Register (offset = 3514h) [reset = 0h]

SYSSTATUSL is shown in Figure 9-41 and described in Table 9-11.

This register provides status information about the module excluding the interrupt status information

Figure 9-41. SYSSTATUSL Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved							RESETDONE
R-0h							R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-11. SYSSTATUSL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	Reserved	R	0h	Reserved. Reads return 0
0	RESETDONE	R	0h	Internal Reset Monitoring 0x0 = Internal module reset is on-going 0x1 = Reset completed

9.6.4 IRQSTATUSL Register (offset = 3518h) [reset = 0h]

IRQSTATUSL is shown in Figure 9-42 and described in Table 9-12.

The interrupt status regroups all the status of the module internal events that can generate an interrupt

Figure 9-42. IRQSTATUSL Register

15	14	13	12	11	10	9	8
Reserved	RX3_FULL	TX3_UNDERFLOW	TX3_EMPTY	Reserved	RX2_FULL	TX2_UNDERFLOW	TX2_EMPTY
R-0h	RW-0h	RW-0h	RW-0h	R-0h	RW-0h	RW-0h	RW-0h
7	6	5	4	3	2	1	0
Reserved	RX1_FULL	TX1_UNDERFLOW	TX1_EMPTY	RX0_OVERFLOW	RX0_FULL	TX0_UNDERFLOW	TX0_EMPTY
R-0h	RW-0h	RW-0h	RW-0h	RW-0h	RW-0h	RW-0h	RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-12. IRQSTATUSL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	Reserved. Reads return 0
14	RX3_FULL	RW	0h	Receiver register full or almost full. Channel 3 0x0(Read) = False event 0x0(Write) = Event status bit unchanged 0x1(Read) = Event is pending 0x1(Write) = Event status bit is reset
13	TX3_UNDERFLOW	RW	0h	Transmitter register underflow. Channel 3 0x0(Read) = False event 0x0(Write) = Event status bit unchanged 0x1(Read) = Event is pending 0x1(Write) = Event status bit is reset
12	TX3_EMPTY	RW	0h	Transmitter register empty or almost empty. Channel 3 0x0(Read) = False event 0x0(Write) = Event status bit unchanged 0x1(Read) = Event is pending 0x1(Write) = Event status bit is reset
11	Reserved	R	0h	Reserved. Reads return 0
10	RX2_FULL	RW	0h	Receiver register full or almost full. Channel 2 0x0(Read) = False event 0x0(Write) = Event status bit unchanged 0x1(Read) = Event is pending 0x1(Write) = Event status bit is reset
9	TX2_UNDERFLOW	RW	0h	Transmitter register underflow. Channel 2 0x0(Read) = False event 0x0(Write) = Event status bit unchanged 0x1(Read) = Event is pending 0x1(Write) = Event status bit is reset

Table 9-12. IRQSTATUSL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8	TX2_EMPTY	RW	0h	Transmitter register empty or almost empty. Channel 2 0x0(Read) = False event 0x0(Write) = Event status bit unchanged 0x1(Read) = Event is pending 0x1(Write) = Event status bit is reset
7	Reserved	R	0h	Reserved. Reads return 0
6	RX1_FULL	RW	0h	Receiver register full or almost full. Channel 1 0x0(Read) = False event 0x0(Write) = Event status bit unchanged 0x1(Read) = Event is pending 0x1(Write) = Event status bit is reset
5	TX1_UNDERFLOW	RW	0h	Transmitter register underflow. Channel 1 0x0(Read) = False event 0x0(Write) = Event status bit unchanged 0x1(Read) = Event is pending 0x1(Write) = Event status bit is reset
4	TX1_EMPTY	RW	0h	Transmitter register empty or almost empty. Channel 1 0x0(Read) = False event 0x0(Write) = Event status bit unchanged 0x1(Read) = Event is pending 0x1(Write) = Event status bit is reset
3	RX0_OVERFLOW	RW	0h	Receiver register overflow (slave mode only). Channel 0 0x0(Read) = False event 0x0(Write) = Event status bit unchanged 0x1(Read) = Event is pending 0x1(Write) = Event status bit is reset
2	RX0_FULL	RW	0h	Receiver register full or almost full. Channel 0 0x0(Read) = False event 0x0(Write) = Event status bit unchanged 0x1(Read) = Event is pending 0x1(Write) = Event status bit is reset
1	TX0_UNDERFLOW	RW	0h	Transmitter register underflow. Channel 0 0x0(Read) = False event 0x0(Write) = Event status bit unchanged 0x1(Read) = Event is pending 0x1(Write) = Event status bit is reset
0	TX0_EMPTY	RW	0h	Transmitter register empty or almost empty. Channel 0 0x0(Read) = False event 0x0(Write) = Event status bit unchanged 0x1(Read) = Event is pending 0x1(Write) = Event status bit is reset

9.6.5 IRQSTATUSU Register (offset = 3519h) [reset = 0h]

IRQSTATUSU is shown in Figure 9-43 and described in Table 9-13.

The interrupt status regroups all the status of the module internal events that can generate an interrupt

Figure 9-43. IRQSTATUSU Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved						EOW	Reserved
R-0h						RW-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-13. IRQSTATUSU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved. Reads return 0
1	EOW	RW	0h	End of word count event when a channel is enabled using the FIFO buffer and the channel had sent the number of SPI word defined by MCSPI_XFERLEVEL[WCNT]. 0x0(Write) = Event status bit unchanged 0x1(Write) = Event status bit is reset
0	Reserved	R	0h	Reserved. Reads return 0

9.6.6 IRQENABLEL Register (offset = 351Ch) [reset = 0h]

IRQENABLEL is shown in Figure 9-44 and described in Table 9-14.

This register allows to enable/disable the module internal sources of interrupt, on an event-by-event basis.

Figure 9-44. IRQENABLEL Register

15	14	13	12	11	10	9	8
Reserved	RX3_FULL_ENABLE	TX3_UNDERFLOW_ENABLE	TX3_EMPTY_ENABLE	Reserved	RX2_FULL_ENABLE	TX2_UNDERFLOW_ENABLE	TX2_EMPTY_ENABLE
R-0h	RW-0h	RW-0h	RW-0h	R-0h	RW-0h	RW-0h	RW-0h
7	6	5	4	3	2	1	0
Reserved	RX1_FULL_ENABLE	TX1_UNDERFLOW_ENABLE	TX1_EMPTY_ENABLE	RX0_OVERFLOW_ENABLE	RX0_FULL_ENABLE	TX0_UNDERFLOW_ENABLE	TX0_EMPTY_ENABLE
R-0h	RW-0h	RW-0h	RW-0h	RW-0h	RW-0h	RW-0h	RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-14. IRQENABLEL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	Reserved. Reads return 0
14	RX3_FULL_ENABLE	RW	0h	Receiver register Full or almost full Interrupt Enable. Ch 3 0x0 = Interrupt disabled 0x1 = Interrupt enabled
13	TX3_UNDERFLOW_ENABLE	RW	0h	Transmitter register Underflow Interrupt Enable. Ch 3 0x0 = Interrupt disabled 0x1 = Interrupt enabled
12	TX3_EMPTY_ENABLE	RW	0h	Transmitter register Empty or almost empty Interrupt Enable. Ch 3 0x0 = Interrupt disabled 0x1 = Interrupt enabled
11	Reserved	R	0h	Reserved. Reads return 0
10	RX2_FULL_ENABLE	RW	0h	Receiver register Full or almost full Interrupt Enable. Ch 2 0x0 = Interrupt disabled 0x1 = Interrupt enabled
9	TX2_UNDERFLOW_ENABLE	RW	0h	Transmitter register Underflow Interrupt Enable. Ch 2 0x0 = Interrupt disabled 0x1 = Interrupt enabled
8	TX2_EMPTY_ENABLE	RW	0h	Transmitter register Empty or almost empty Interrupt Enable. Ch 2 0x0 = Interrupt disabled 0x1 = Interrupt enabled
7	Reserved	R	0h	Reserved. Reads return 0
6	RX1_FULL_ENABLE	RW	0h	Receiver register Full or almost full Interrupt Enable. Ch 1 0x0 = Interrupt disabled 0x1 = Interrupt enabled
5	TX1_UNDERFLOW_ENABLE	RW	0h	Transmitter register Underflow Interrupt Enable. Ch 1 0x0 = Interrupt disabled 0x1 = Interrupt enabled

Table 9-14. IRQENABLEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	TX1_EMPTY_ENABLE	RW	0h	Transmitter register Empty or almost empty Interrupt Enable. Ch 1 0x0 = Interrupt disabled 0x1 = Interrupt enabled
3	RX0_OVERFLOW_ENABLE	RW	0h	Receiver register Overflow Interrupt Enable. Ch 0 0x0 = Interrupt disabled 0x1 = Interrupt enabled
2	RX0_FULL_ENABLE	RW	0h	Receiver register Full or almost full Interrupt Enable. Ch 0 0x0 = Interrupt disabled 0x1 = Interrupt enabled
1	TX0_UNDERFLOW_ENABLE	RW	0h	Transmitter register Underflow Interrupt Enable. Ch 0 0x0 = Interrupt disabled 0x1 = Interrupt enabled
0	TX0_EMPTY_ENABLE	RW	0h	Transmitter register Empty or almost empty Interrupt Enable. Ch 0 0x0 = Interrupt disabled 0x1 = Interrupt enabled

9.6.7 IRQENABLEU Register (offset = 351Dh) [reset = 0h]

IRQENABLEU is shown in Figure 9-45 and described in Table 9-15.

This register allows to enable/disable the module internal sources of interrupt, on an event-by-event basis.

Figure 9-45. IRQENABLEU Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved						EOWE	Reserved
R-0h						RW-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-15. IRQENABLEU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved. Reads return 0
1	EOWE	RW	0h	End of Word count Interrupt Enable 0x0 = Interrupt disabled 0x1 = Interrupt enabled
0	Reserved	R	0h	Reserved. Reads return 0

9.6.8 WAKEUPENABLEL Register (offset = 3520h) [reset = 0h]

WAKEUPENABLEL is shown in Figure 9-46 and described in Table 9-16.

The wakeup enable register allows to enable/disable the module internal sources of wakeup on event-by-event basis.

Figure 9-46. WAKEUPENABLEL Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-16. WAKEUPENABLEL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	Reserved	R	0h	Reserved. Reads return 0

9.6.9 MODULCTRLL Register (offset = 3528h) [reset = 4h]

MODULCTRLL is shown in Figure 9-47 and described in Table 9-17.

This register is dedicated to the configuration of the serial port interface.

Figure 9-47. MODULCTRLL Register

15	14	13	12	11	10	9	8
Reserved							FDAA
R-0h							RW-0h

7	6	5	4	3	2	1	0
MOA	INITDLY			Reserved	MS	Reserved	SINGLE
RW-0h	RW-0h			R-0h	RW-1h	R-0h	RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-17. MODULCTRLL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-9	Reserved	R	0h	Reserved. Reads return 0
8	FDAA	RW	0h	FIFO DMA Address 256-bit aligned 0x0 = FIFO data managed by CHxTX and CHxRX registers 0x1 = FIFO data managed by DAFTX and DAFRX registers
7	MOA	RW	0h	Multiple word internal McSPI bridge access 0x0 = Multiple word access disabled 0x1 = Multiple word access enabled with FIFO
6-4	INITDLY	RW	0h	Initial spi delay for first transfer 0x0 = No delay for first spi transfer. 0x1 = The controller wait 4 spi bus clock 0x2 = The controller wait 8 spi bus clock 0x3 = The controller wait 16 spi bus clock 0x4 = The controller wait 32 spi bus clock
3	Reserved	R	0h	Reserved. This bit must be set as 0x0 for proper operation.
2	MS	RW	1h	Master/ Slave 0x0 = Master - The module generates the MCSPI_CLK and MCSPI_CS0 0x1 = Slave - The module receives the MCSPI_CLK and MCSPI_CS0
1	Reserved	R	0h	Reserved. This bit must be set as 0x0 for proper operation.
0	SINGLE	RW	0h	Single channel / Multi Channel (master mode only) 0x0 = More than one channel will be used in master mode 0x1 = Only one channel will be used in master mode. This bit must be set in Force McSPI_CSx mode

9.6.10 CH0CONFL Register (offset = 352Ch) [reset = 0h]

CH0CONFL is shown in Figure 9-48 and described in Table 9-18.

This register is dedicated to the configuration of the channel 0

Figure 9-48. CH0CONFL Register

15	14	13	12	11	10	9	8
DMAR	DMAW	TRM		WL			
RW-0h	RW-0h	RW-0h		RW-0h			

7	6	5	4	3	2	1	0
WL	EPOL	CLKD				POL	PHA
RW-0h	RW-0h	RW-0h				RW-0h	RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-18. CH0CONFL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	DMAR	RW	0h	DMA Read request 0x0 = DMA Read Request disabled 0x1 = DMA Read Request enabled
14	DMAW	RW	0h	DMA Write request 0x0 = DMA Write Request disabled 0x1 = DMA Write Request enabled
13-12	TRM	RW	0h	Transmit/Receive modes 0x0 = Transmit and Receive mode 0x1 = Receive only mode 0x2 = Transmit only mode

Table 9-18. CH0CONFL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11-7	WL	RW	0h	SPI word length 0x3 = 4BIT 0x4 = 5BIT 0x5 = 6BIT 0x6 = 7BIT 0x7 = 8BIT 0x8 = 9BIT 0x9 = 10BIT 0xa = 11BIT 0xb = 12BIT 0xc = 13BIT 0xd = 14BIT 0xe = 15BIT 0xf = 16BIT 0x10 = 17BIT 0x11 = 18BIT 0x12 = 19BIT 0x13 = 20BIT 0x14 = 21BIT 0x15 = 22BIT 0x16 = 23BIT 0x17 = 24BIT 0x18 = 25BIT 0x19 = 26BIT 0x1a = 27BIT 0x1b = 28BIT 0x1c = 29BIT 0x1d = 30BIT 0x1e = 31BIT 0x1f = 32BIT
6	EPOL	RW	0h	CS0 polarity 0x0 = MCSPI_CS0 is held high during the active state 0x1 = MCSPI_CS0 is held low during the active state
5-2	CLKD	RW	0h	Frequency divider for MCSPI_CLK 0x0 = DIV1 0x1 = DIV2 0x2 = DIV4 0x3 = DIV8 0x4 = DIV16 0x5 = DIV32 0x6 = DIV64 0x7 = DIV128 0x8 = DIV256 0x9 = DIV512 0xa = DIV1024 0xb = DIV2048 0xc = DIV4096 0xd = DIV8192 0xe = DIV16384 0xf = DIV32768
1	POL	RW	0h	MCSPI_CLK polarity 0x0 = MCSPI_CLK is held high during the active state 0x1 = MCSPI_CLK is held low during the active state

Table 9-18. CH0CONFL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	PHA	RW	0h	MCSPI_CLK phase 0x0 = Data are latched on odd numbered edges of MCSPI_CLK. 0x1 = Data are latched on even numbered edges of MCSPI_CLK.

9.6.11 CH0CONFU Register (offset = 352Dh) [reset = 6h]

CH0CONFU is shown in Figure 9-49 and described in Table 9-19.

This register is dedicated to the configuration of the channel 0

Figure 9-49. CH0CONFU Register

15	14	13	12	11	10	9	8
Reserved		CLKG	FFER	FFEW	TCS		SBPOL
R-0h		RW-0h	RW-0h	RW-0h	RW-0h		RW-0h

7	6	5	4	3	2	1	0
SBE	SPICSSLV		FORCE	TURBO	IS	DPE1	DPE0
RW-0h	RW-0h		RW-0h	RW-0h	RW-1h	RW-1h	RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-19. CH0CONFU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	Reserved	R	0h	Reserved. Reads return 0
13	CLKG	RW	0h	Clock divider granularity 0x0 = Clock granularity of power of two 0x1 = One clock cycle granularity
12	FFER	RW	0h	FIFO enabled for receive 0x0 = The buffer is not used to receive data. 0x1 = The buffer is used to receive data
11	FFEW	RW	0h	FIFO enabled for Transmit 0x0 = The buffer is not used to transmit data 0x1 = The buffer is used to transmit data.
10-9	TCS	RW	0h	Chip Select Time Control 0x0 = 0.5 clock cycle 0x1 = 1.5 clock cycle 0x2 = 2.5 clock cycle 0x3 = 3.5 clock cycle
8	SBPOL	RW	0h	Start bit polarity. 0x0 = Start bit polarity is held to 0 during SPI transfer. 0x1 = Start bit polarity is held to 1 during SPI transfer
7	SBE	RW	0h	Start bit enable for SPI transfer 0x0 = Default SPI transfer length as specified by WL bit field 0x1 = Start bit D/CX added before SPI transfer, polarity is defined by MCSPI_CH(i)CONF[SBPOL]
6-5	SPICSSLV	RW	0h	Channel 0 only and slave mode only: SPI slave select signal detection 0x0 = Detection enabled only on McSPI_CS0 0x1 = Reserved 0x2 = Reserved 0x3 = Reserved
4	FORCE	RW	0h	Manual McSPI_CS0 assertion to keep McSPI_CS0 active between SPI words 0x0 = Writing 0 into this bit drives low the McSPI_CS0 line when MCSPI_CH(i)CONF[EPOL]=0, and drives it high when MCSPI_CH(i)CONF[EPOL]=1 0x1 = Writing 1 into this bit drives high the McSPI_CS0 line when MCSPI_CH(i)CONF[EPOL]=0, and drives it low when MCSPI_CH(i)CONF[EPOL]=1

Table 9-19. CH0CONFU Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	TURBO	RW	0h	Turbo mode 0x0 = Turbo is deactivated (recommended for single SPI word transfer) 0x1 = Turbo is activated to maximize the throughput for multi SPI words transfer
2	IS	RW	1h	Input Select. Note: DPE1 and DPE0 are bidirectional data lines which can be mapped independently and programmed as transmit or receive mode. 0x0 = Data Line0 (MCSPI_SIMO) selected for reception 0x1 = Data Line1 (MCSPI_SOMI) selected for reception
1	DPE1	RW	1h	Transmission Enable for data line 1. Note: DPE1 and DPE0 are bidirectional data lines which can be mapped independently and programmed as transmit or receive mode. 0x0 = Data Line1 (MCSPI_SOMI) selected for transmission 0x1 = No transmission on Data Line1 (MCSPI_SOMI)
0	DPE0	RW	0h	Transmission Enable for data line 0. Note: DPE1 and DPE0 are bidirectional data lines which can be mapped independently and programmed as transmit or receive mode. 0x0 = Data Line0 (MCSPI_SIMO) selected for transmission 0x1 = No transmission on Data Line0 (MCSPI_SIMO)

9.6.12 CH0STATL Register (offset = 3530h) [reset = 0h]

CH0STATL is shown in Figure 9-50 and described in Table 9-20.

This register provides status information about transmitter and receiver registers of channel 0

Figure 9-50. CH0STATL Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved	RXFFF	RXFFE	TXFFF	TXFFE	EOT	TXS	RXS
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-20. CH0STATL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	Reserved	R	0h	Reserved. Reads return 0
6	RXFFF	R	0h	FIFO Receive Buffer Full Status 0x0 = FIFO Receive Buffer is not full 0x1 = FIFO Receive Buffer is full
5	RXFFE	R	0h	FIFO Receive Buffer Empty Status 0x0 = FIFO Receive Buffer is not empty 0x1 = FIFO Receive Buffer is empty
4	TXFFF	R	0h	FIFO Transmit Buffer Full Status 0x0 = FIFO Transmit Buffer is not full 0x1 = FIFO Transmit Buffer is full
3	TXFFE	R	0h	FIFO Transmit Buffer Empty Status 0x0 = FIFO Transmit Buffer is not empty 0x1 = FIFO Transmit Buffer is empty
2	EOT	R	0h	End of transfer Status 0x0 = Shift register is loaded with the data from the transmitter register 0x1 = End of an SPI transfer.
1	TXS	R	0h	Transmitter Register Status 0x0 = Register is full 0x1 = Register is empty
0	RXS	R	0h	Receiver Register Status 0x0 = Register is empty 0x1 = Register is full

9.6.13 CH0CTRLL Register (offset = 3534h) [reset = 0h]

CH0CTRLL is shown in Figure 9-51 and described in Table 9-21.

This register is dedicated to enable the channel and defines the extended clock ratio with one clock cycle granularity

Figure 9-51. CH0CTRLL Register

15	14	13	12	11	10	9	8
EXTCLK							
RW-0h							

7	6	5	4	3	2	1	0
Reserved							EN
R-0h							RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-21. CH0CTRLL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	EXTCLK	RW	0h	Clock ratio extension 0x0 = Clock ratio is CLKD + 1 0x1 = Clock ratio is CLKD + 1 + 16 0x2 = Clock ratio is CLKD + 1 + 32 0xff = Clock ratio is CLKD + 1 + 4080
7-1	Reserved	R	0h	Reserved. Reads return 0
0	EN	RW	0h	Channel Enable 0x0 = Channel 0 disable 0x1 = Channel 0 enable

9.6.14 CH0TXL Register (offset = 3538h) [reset = 0h]

CH0TXL is shown in Figure 9-52 and described in Table 9-22.

This register contains a single SPI word to transmit on the serial link, what ever SPI word length is.

Figure 9-52. CH0TXL Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-22. CH0TXL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TDATA	RW	0h	Data to transmit

9.6.15 CH0TXU Register (offset = 3539h) [reset = 0h]

CH0TXU is shown in Figure 9-53 and described in Table 9-23.

This register contains a single SPI word to transmit on the serial link, what ever SPI word length is.

Figure 9-53. CH0TXU Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-23. CH0TXU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TDATA	RW	0h	Data to transmit

9.6.16 CH0RXL Register (offset = 353Ch) [reset = 0h]

CH0RXL is shown in Figure 9-54 and described in Table 9-24.

This register contains a single SPI word received through the serial link, what ever SPI word length is.

Figure 9-54. CH0RXL Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-24. CH0RXL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RDATA	R	0h	Received Data

9.6.17 CH0RXU Register (offset = 353Dh) [reset = 0h]

CH0RXU is shown in Figure 9-55 and described in Table 9-25.

This register contains a single SPI word received through the serial link, what ever SPI word length is.

Figure 9-55. CH0RXU Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-25. CH0RXU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RDATA	R	0h	Received Data

9.6.18 CH1CONFL Register (offset = 3540h) [reset = 0h]

CH1CONFL is shown in Figure 9-56 and described in Table 9-26.

This register is dedicated to the configuration of the channel 1

Figure 9-56. CH1CONFL Register

15	14	13	12	11	10	9	8
DMAR	DMAW	TRM		WL			
RW-0h	RW-0h	RW-0h		RW-0h			

7	6	5	4	3	2	1	0
WL	EPOL	CLKD				POL	PHA
RW-0h	RW-0h	RW-0h				RW-0h	RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-26. CH1CONFL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	DMAR	RW	0h	DMA Read request 0x0 = DMA Read Request disabled 0x1 = DMA Read Request enabled
14	DMAW	RW	0h	DMA Write request 0x0 = DMA Write Request disabled 0x1 = DMA Write Request enabled
13-12	TRM	RW	0h	Transmit/Receive modes 0x0 = Transmit and Receive mode 0x1 = Receive only mode 0x2 = Transmit only mode

Table 9-26. CH1CONFL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11-7	WL	RW	0h	SPI word length 0x3 = 4BIT 0x4 = 5BIT 0x5 = 6BIT 0x6 = 7BIT 0x7 = 8BIT 0x8 = 9BIT 0x9 = 10BIT 0xa = 11BIT 0xb = 12BIT 0xc = 13BIT 0xd = 14BIT 0xe = 15BIT 0xf = 16BIT 0x10 = 17BIT 0x11 = 18BIT 0x12 = 19BIT 0x13 = 20BIT 0x14 = 21BIT 0x15 = 22BIT 0x16 = 23BIT 0x17 = 24BIT 0x18 = 25BIT 0x19 = 26BIT 0x1a = 27BIT 0x1b = 28BIT 0x1c = 29BIT 0x1d = 30BIT 0x1e = 31BIT 0x1f = 32BIT
6	EPOL	RW	0h	MCSPI_CS1 polarity 0x0 = MCSPI_CS1 is held high during the active state 0x1 = MCSPI_CS1 is held low during the active state
5-2	CLKD	RW	0h	Frequency divider for SPICLK 0x0 = DIV1 0x1 = DIV2 0x2 = DIV4 0x3 = DIV8 0x4 = DIV16 0x5 = DIV32 0x6 = DIV64 0x7 = DIV128 0x8 = DIV256 0x9 = DIV512 0xa = DIV1024 0xb = DIV2048 0xc = DIV4096 0xd = DIV8192 0xe = DIV16384 0xf = DIV32768
1	POL	RW	0h	MCSPI_CLK polarity 0x0 = MCSPI_CLK is held high during the active state 0x1 = MCSPI_CLK is held low during the active state

Table 9-26. CH1CONFL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	PHA	RW	0h	MCSPI_CLK phase 0x0 = Data are latched on odd numbered edges of MCSPI_CLK. 0x1 = Data are latched on even numbered edges of MCSPI_CLK.

9.6.19 CH1CONFU Register (offset = 3541h) [reset = 6h]

CH1CONFU is shown in Figure 9-57 and described in Table 9-27.

This register is dedicated to the configuration of the channel 1

Figure 9-57. CH1CONFU Register

15	14	13	12	11	10	9	8
Reserved		CLKG	FFER	FFEW	TCS		SBPOL
R-0h		RW-0h	RW-0h	RW-0h	RW-0h		RW-0h

7	6	5	4	3	2	1	0
SBE	SPICSSLV		FORCE	TURBO	IS	DPE1	DPE0
RW-0h	RW-0h		RW-0h	RW-0h	RW-1h	RW-1h	RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-27. CH1CONFU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	Reserved	R	0h	Reserved. Reads return 0
13	CLKG	RW	0h	Clock divider granularity 0x0 = Clock granularity of power of two 0x1 = One clock cycle granularity
12	FFER	RW	0h	FIFO enabled for receive 0x0 = The buffer is not used to receive data. 0x1 = The buffer is used to receive data
11	FFEW	RW	0h	FIFO enabled for Transmit 0x0 = The buffer is not used to transmit data 0x1 = The buffer is used to transmit data.
10-9	TCS	RW	0h	Chip Select Time Control 0x0 = 0.5 clock cycle 0x1 = 1.5 clock cycle 0x2 = 2.5 clock cycle 0x3 = 3.5 clock cycle
8	SBPOL	RW	0h	Start bit polarity. 0x0 = Start bit polarity is held to 0 during SPI transfer. 0x1 = Start bit polarity is held to 1 during SPI transfer
7	SBE	RW	0h	Start bit enable for SPI transfer 0x0 = Default SPI transfer length as specified by WL bit field 0x1 = Start bit D/CX added before SPI transfer, polarity is defined by MCSPI_CH(i)CONF[SBPOL]
6-5	SPICSSLV	RW	0h	Channel 0 only and slave mode only: SPI slave select signal detection 0x0 = Reserved 0x1 = Reserved 0x2 = Reserved 0x3 = Reserved
4	FORCE	RW	0h	Manual MCSPI_CS1 assertion to keep MCSPI_CS1 active between SPI words 0x0 = Writing 0 into this bit drives low the McSPI_CS1 line when MCSPI_CH(i)CONF[EPOL]=0, and drives it high when MCSPI_CH(i)CONF[EPOL]=1 0x1 = Writing 1 into this bit drives high the McSPI_CS1 line when MCSPI_CH(i)CONF[EPOL]=0, and drives it low when MCSPI_CH(i)CONF[EPOL]=1

Table 9-27. CH1CONFU Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	TURBO	RW	0h	Turbo mode 0x0 = Turbo is deactivated (recommended for single SPI word transfer) 0x1 = Turbo is activated to maximize the throughput for multi SPI words transfer
2	IS	RW	1h	Input Select. Note: DPE1 and DPE0 are bidirectional data lines which can be mapped independently and programmed as transmit or receive mode. 0x0 = Data Line0 (MCSPI_SIMO) selected for reception 0x1 = Data Line1 (MCSPI_SOMI) selected for reception
1	DPE1	RW	1h	Transmission Enable for data line 1. Note: DPE1 and DPE0 are bidirectional data lines which can be mapped independently and programmed as transmit or receive mode. 0x0 = Data Line1 (MCSPI_SOMI) selected for transmission 0x1 = No transmission on Data Line1 (MCSPI_SOMI)
0	DPE0	RW	0h	Transmission Enable for data line 0. Note: DPE1 and DPE0 are bidirectional data lines which can be mapped independently and programmed as transmit or receive mode. 0x0 = Data Line0 (MCSPI_SIMO) selected for transmission 0x1 = No transmission on Data Line0 (MCSPI_SIMO)

9.6.20 CH1STATL Register (offset = 3544h) [reset = 0h]

CH1STATL is shown in Figure 9-58 and described in Table 9-28.

This register provides status information about transmitter and receiver registers of channel 1

Figure 9-58. CH1STATL Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved	RXFFF	RXFFE	TXFFF	TXFFE	EOT	TXS	RXS
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-28. CH1STATL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	Reserved	R	0h	Reserved. Reads return 0
6	RXFFF	R	0h	FIFO Receive Buffer Full Status 0x0 = FIFO Receive Buffer is not full 0x1 = FIFO Receive Buffer is full
5	RXFFE	R	0h	FIFO Receive Buffer Empty Status 0x0 = FIFO Receive Buffer is not empty 0x1 = FIFO Receive Buffer is empty
4	TXFFF	R	0h	FIFO Transmit Buffer Full Status 0x0 = FIFO Transmit Buffer is not full 0x1 = FIFO Transmit Buffer is full
3	TXFFE	R	0h	FIFO Transmit Buffer Empty Status 0x0 = FIFO Transmit Buffer is not empty 0x1 = FIFO Transmit Buffer is empty
2	EOT	R	0h	End of transfer Status 0x0 = Shift register is loaded with the data from the transmitter register 0x1 = End of an SPI transfer.
1	TXS	R	0h	Transmitter Register Status 0x0 = Register is full 0x1 = Register is empty
0	RXS	R	0h	Receiver Register Status 0x0 = Register is empty 0x1 = Register is full

9.6.21 CH1CTRLL Register (offset = 3548h) [reset = 0h]

CH1CTRLL is shown in Figure 9-59 and described in Table 9-29.

This register is dedicated to enable the channel and defines the extended clock ratio with one clock cycle granularity.

Figure 9-59. CH1CTRLL Register

15	14	13	12	11	10	9	8
EXTCLK							
RW-0h							

7	6	5	4	3	2	1	0
Reserved							EN
R-0h							RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-29. CH1CTRLL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	EXTCLK	RW	0h	Clock ratio extension 0x0 = Clock ratio is CLKD + 1 0x1 = Clock ratio is CLKD + 1 + 16 0x2 = Clock ratio is CLKD + 1 + 32 0xff = Clock ratio is CLKD + 1 + 4080
7-1	Reserved	R	0h	Reserved. Reads return 0
0	EN	RW	0h	Channel Enable 0x0 = Channel 1 disable 0x1 = Channel 1 enable

9.6.22 CH1TXL Register (offset = 354Ch) [reset = 0h]

CH1TXL is shown in Figure 9-60 and described in Table 9-30.

This register contains a single SPI word to transmit on the serial link, what ever SPI word length is.

Figure 9-60. CH1TXL Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-30. CH1TXL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TDATA	RW	0h	Data to transmit

9.6.23 CH1TXU Register (offset = 354Dh) [reset = 0h]

CH1TXU is shown in Figure 9-61 and described in Table 9-31.

This register contains a single SPI word to transmit on the serial link, what ever SPI word length is.

Figure 9-61. CH1TXU Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-31. CH1TXU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TDATA	RW	0h	Data to transmit

9.6.24 CH1RXL Register (offset = 3550h) [reset = 0h]

CH1RXL is shown in Figure 9-62 and described in Table 9-32.

This register contains a single SPI word received through the serial link, what ever SPI word length is.

Figure 9-62. CH1RXL Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-32. CH1RXL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RDATA	R	0h	Received Data

9.6.25 CH1RXU Register (offset = 3551h) [reset = 0h]

CH1RXU is shown in Figure 9-63 and described in Table 9-33.

This register contains a single SPI word received through the serial link, what ever SPI word length is.

Figure 9-63. CH1RXU Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-33. CH1RXU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RDATA	R	0h	Received Data

9.6.26 CH2CONFL Register (offset = 3554h) [reset = 0h]

CH2CONFL is shown in Figure 9-64 and described in Table 9-34.

This register is dedicated to the configuration of the channel 2

Figure 9-64. CH2CONFL Register

15	14	13	12	11	10	9	8
DMAR	DMAW	TRM		WL			
RW-0h	RW-0h	RW-0h		RW-0h			

7	6	5	4	3	2	1	0
WL	EPOL	CLKD				POL	PHA
RW-0h	RW-0h	RW-0h				RW-0h	RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-34. CH2CONFL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	DMAR	RW	0h	DMA Read request 0x0 = DMA Read Request disabled 0x1 = DMA Read Request enabled
14	DMAW	RW	0h	DMA Write request 0x0 = DMA Write Request disabled 0x1 = DMA Write Request enabled
13-12	TRM	RW	0h	Transmit/Receive modes 0x0 = Transmit and Receive mode 0x1 = Receive only mode 0x2 = Transmit only mode

Table 9-34. CH2CONFL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11-7	WL	RW	0h	SPI word length 0x3 = 4BIT 0x4 = 5BIT 0x5 = 6BIT 0x6 = 7BIT 0x7 = 8BIT 0x8 = 9BIT 0x9 = 10BIT 0xa = 11BIT 0xb = 12BIT 0xc = 13BIT 0xd = 14BIT 0xe = 15BIT 0xf = 16BIT 0x10 = 17BIT 0x11 = 18BIT 0x12 = 19BIT 0x13 = 20BIT 0x14 = 21BIT 0x15 = 22BIT 0x16 = 23BIT 0x17 = 24BIT 0x18 = 25BIT 0x19 = 26BIT 0x1a = 27BIT 0x1b = 28BIT 0x1c = 29BIT 0x1d = 30BIT 0x1e = 31BIT 0x1f = 32BIT
6	EPOL	RW	0h	MCSPI_CS2 polarity 0x0 = MCSPI_CS2 is held high during the active state 0x1 = MCSPI_CS2 is held low during the active state
5-2	CLKD	RW	0h	Frequency divider for MCSPI_CLK 0x0 = DIV1 0x1 = DIV2 0x2 = DIV4 0x3 = DIV8 0x4 = DIV16 0x5 = DIV32 0x6 = DIV64 0x7 = DIV128 0x8 = DIV256 0x9 = DIV512 0xa = DIV1024 0xb = DIV2048 0xc = DIV4096 0xd = DIV8192 0xe = DIV16384 0xf = DIV32768
1	POL	RW	0h	MCSPI_CLK polarity 0x0 = MCSPI_CLK is held high during the active state 0x1 = MCSPI_CLK is held low during the active state

Table 9-34. CH2CONFL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	PHA	RW	0h	MCSPI_CLK phase 0x0 = Data are latched on odd numbered edges of MCSPI_CLK. 0x1 = Data are latched on even numbered edges of MCSPI_CLK.

9.6.27 CH2CONFU Register (offset = 3555h) [reset = 6h]

CH2CONFU is shown in Figure 9-65 and described in Table 9-35.

This register is dedicated to the configuration of the channel 2

Figure 9-65. CH2CONFU Register

15	14	13	12	11	10	9	8
Reserved		CLKG	FFER	FFEW	TCS		SBPOL
R-0h		RW-0h	RW-0h	RW-0h	RW-0h		RW-0h

7	6	5	4	3	2	1	0
SBE	SPICSSLV		FORCE	TURBO	IS	DPE1	DPE0
RW-0h	RW-0h		RW-0h	RW-0h	RW-1h	RW-1h	RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-35. CH2CONFU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	Reserved	R	0h	Reserved. Reads return 0
13	CLKG	RW	0h	Clock divider granularity 0x0 = Clock granularity of power of two 0x1 = One clock cycle granularity
12	FFER	RW	0h	FIFO enabled for receive 0x0 = The buffer is not used to receive data. 0x1 = The buffer is used to receive data
11	FFEW	RW	0h	FIFO enabled for transmit. 0x0 = The buffer is not used to transmit data 0x1 = The buffer is used to transmit data.
10-9	TCS	RW	0h	Chip Select Time Control 0x0 = 0.5 clock cycle 0x1 = 1.5 clock cycle 0x2 = 2.5 clock cycle 0x3 = 3.5 clock cycle
8	SBPOL	RW	0h	Start bit polarity. 0x0 = Start bit polarity is held to 0 during SPI transfer. 0x1 = Start bit polarity is held to 1 during SPI transfer
7	SBE	RW	0h	Start bit enable for SPI transfer 0x0 = Default SPI transfer length as specified by WL bit field 0x1 = Start bit D/CX added before SPI transfer, polarity is defined by MCSPI_CH(i)CONF[SBPOL]
6-5	SPICSSLV	RW	0h	Channel 0 only and slave mode only: SPI slave select signal detection 0x0 = Reserved 0x1 = Reserved 0x2 = Reserved 0x3 = Reserved
4	FORCE	RW	0h	Manual MCSPI_CS2 assertion to keep MCSPI_CS2 active between SPI words 0x0 = Writing 0 into this bit drives low the McSPI_CS2 line when MCSPI_CH(i)CONF[EPOL]=0, and drives it high when MCSPI_CH(i)CONF[EPOL]=1 0x1 = Writing 1 into this bit drives high the McSPI_CS2 line when MCSPI_CH(i)CONF[EPOL]=0, and drives it low when MCSPI_CH(i)CONF[EPOL]=1

Table 9-35. CH2CONFU Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	TURBO	RW	0h	Turbo mode 0x0 = Turbo is deactivated (recommended for single SPI word transfer) 0x1 = Turbo is activated to maximize the throughput for multi SPI words transfer
2	IS	RW	1h	Input Select. Note: DPE1 and DPE0 are bidirectional data lines which can be mapped independently and programmed as transmit or receive mode. 0x0 = Data Line0 (MCSPI_SIMO) selected for reception 0x1 = Data Line1 (MCSPI_SOMI) selected for reception
1	DPE1	RW	1h	Transmission Enable for data line 1. Note: DPE1 and DPE0 are bidirectional data lines which can be mapped independently and programmed as transmit or receive mode. 0x0 = Data Line1 (MCSPI_SOMI) selected for transmission 0x1 = No transmission on Data Line1 (MCSPI_SOMI)
0	DPE0	RW	0h	Transmission Enable for data line 0. Note: DPE1 and DPE0 are bidirectional data lines which can be mapped independently and programmed as transmit or receive mode. 0x0 = Data Line0 (MCSPI_SIMO) selected for transmission 0x1 = No transmission on Data Line0 (MCSPI_SIMO)

9.6.28 CH2STATL Register (offset = 3558h) [reset = 0h]

CH2STATL is shown in Figure 9-66 and described in Table 9-36.

This register provides status information about transmitter and receiver registers of channel 2

Figure 9-66. CH2STATL Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved	RXFFF	RXFFE	TXFFF	TXFFE	EOT	TXS	RXS
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-36. CH2STATL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	Reserved	R	0h	Reserved. Reads return 0
6	RXFFF	R	0h	FIFO Receive Buffer Full Status 0x0 = FIFO Receive Buffer is not full 0x1 = FIFO Receive Buffer is full
5	RXFFE	R	0h	FIFO Receive Buffer Empty Status 0x0 = FIFO Receive Buffer is not empty 0x1 = FIFO Receive Buffer is empty
4	TXFFF	R	0h	FIFO Transmit Buffer Full Status 0x0 = FIFO Transmit Buffer is not full 0x1 = FIFO Transmit Buffer is full
3	TXFFE	R	0h	FIFO Transmit Buffer Empty Status 0x0 = FIFO Transmit Buffer is not empty 0x1 = FIFO Transmit Buffer is empty
2	EOT	R	0h	End of transfer Status 0x0 = Shift register is loaded with the data from the transmitter register 0x1 = End of an SPI transfer.
1	TXS	R	0h	Transmitter Register Status 0x0 = Register is full 0x1 = Register is empty
0	RXS	R	0h	Receiver Register Status 0x0 = Register is empty 0x1 = Register is full

9.6.29 CH2CTRLL Register (offset = 355Ch) [reset = 0h]

CH2CTRLL is shown in Figure 9-67 and described in Table 9-37.

This register is dedicated to enable the channel and defines the extended clock ratio with one clock cycle granularity.

Figure 9-67. CH2CTRLL Register

15	14	13	12	11	10	9	8
EXTCLK							
RW-0h							

7	6	5	4	3	2	1	0
Reserved							EN
R-0h							RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-37. CH2CTRLL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	EXTCLK	RW	0h	Clock ratio extension 0x0 = Clock ratio is CLKD + 1 0x1 = Clock ratio is CLKD + 1 + 16 0x2 = Clock ratio is CLKD + 1 + 32 0xff = Clock ratio is CLKD + 1 + 4080
7-1	Reserved	R	0h	Reserved. Reads return 0
0	EN	RW	0h	Channel Enable 0x0 = Channel 2 disable 0x1 = Channel 2 enable

9.6.30 CH2TXL Register (offset = 3560h) [reset = 0h]

CH2TXL is shown in Figure 9-68 and described in Table 9-38.

This register contains a single SPI word to transmit on the serial link, what ever SPI word length is.

Figure 9-68. CH2TXL Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-38. CH2TXL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TDATA	RW	0h	Channel 2 Data to transmit

9.6.31 CH2TXU Register (offset = 3561h) [reset = 0h]

CH2TXU is shown in Figure 9-69 and described in Table 9-39.

This register contains a single SPI word to transmit on the serial link, what ever SPI word length is.

Figure 9-69. CH2TXU Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-39. CH2TXU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TDATA	RW	0h	Channel 2 Data to transmit

9.6.32 CH2RXL Register (offset = 3564h) [reset = 0h]

CH2RXL is shown in Figure 9-70 and described in Table 9-40.

This register contains a single SPI word received through the serial link, what ever SPI word length is.

Figure 9-70. CH2RXL Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-40. CH2RXL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RDATA	R	0h	Channel 2 Received Data

9.6.33 CH2RXU Register (offset = 3565h) [reset = 0h]

CH2RXU is shown in Figure 9-71 and described in Table 9-41.

This register contains a single SPI word received through the serial link, what ever SPI word length is.

Figure 9-71. CH2RXU Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-41. CH2RXU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RDATA	R	0h	Channel 2 Received Data

9.6.34 XFERLEVELL Register (offset = 357Ch) [reset = 0h]

XFERLEVELL is shown in Figure 9-72 and described in Table 9-42.

This register provides transfer levels needed while using FIFO buffer during transfer

Figure 9-72. XFERLEVELL Register

15	14	13	12	11	10	9	8
AFL							
RW-0h							
7	6	5	4	3	2	1	0
AEL							
RW-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-42. XFERLEVELL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	AFL	RW	0h	Buffer Almost Full
7-0	AEL	RW	0h	Buffer Almost Empty

9.6.35 XFERLEVELU Register (offset = 357Dh) [reset = 0h]

XFERLEVELU is shown in Figure 9-73 and described in Table 9-43.

This register provides transfer levels needed while using FIFO buffer during transfer

Figure 9-73. XFERLEVELU Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-43. XFERLEVELU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	WCNT	RW	0h	SPI word counter

9.6.36 DAFTXL Register (offset = 3580h) [reset = 0h]

DAFTXL is shown in Figure 9-74 and described in Table 9-44.

This register contains the SPI words to transmit on the serial link when FIFO used and DMA address is aligned on 256 bit, 3580h.

Figure 9-74. DAFTXL Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-44. DAFTXL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DAFTDATA	RW	0h	FIFO Data to transmit with DMA 256 bit aligned address

9.6.37 DAFTXU Register (offset = 3581h) [reset = 0h]

DAFTXU is shown in Figure 9-75 and described in Table 9-45.

This register contains the SPI words to transmit on the serial link when FIFO used and DMA address is aligned on 256 bit

Figure 9-75. DAFTXU Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-45. DAFTXU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DAFTDATA	RW	0h	FIFO Data to transmit with DMA 256 bit aligned address

9.6.38 DAFRXL Register (offset = 35A0h) [reset = 0h]

DAFRXL is shown in Figure 9-76 and described in Table 9-46.

This register contains the SPI words to received on the serial link when FIFO used and DMA address is aligned on 256 bit, 35A0h.

Figure 9-76. DAFRXL Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-46. DAFRXL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DAFRDATA	R	0h	FIFO Received Data with DMA 256 bit aligned address

9.6.39 DAFRXU Register (offset = 35A1h) [reset = 0h]

DAFRXU is shown in Figure 9-77 and described in Table 9-47.

This register contains the SPI words to received on the serial link when FIFO used and DMA address is aligned on 256 bit.

Figure 9-77. DAFRXU Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 9-47. DAFRXU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DAFRDATA	R	0h	FIFO Received Data with DMA 256 bit aligned address

Multimedia Card (MMC)/Secure Digital (SD) Card Controller

Topic **Page**

10.1 Introduction

This document describes the Multimedia Card (MMC)/Secure Digital (SD) Card Controller in the digital signal processor (DSP).

10.1.1 Purpose of the Peripheral

The Multimedia Card (MMC)/secure digital (SD) card is used in a number of applications to provide removable data storage. The MMC/SD card controller provides an interface to MMC, SD and SDHC external cards.

10.1.2 Features

The MMC/SD card controller has the following features:

- Supports interface to Multimedia Cards (MMC).
- Supports interface to secure digital (SD) memory cards.
- Supports the use of both MMC/SD and SDIO protocols.
- Programmable clock frequency to control the timing of transfers between the MMC/SD controller and memory card.
- 256-bit read/write FIFO to lower system overhead.
- Signaling to support direct memory access (DMA) transfers (slave).

The device includes two independent MMC/SD card controllers.

10.1.3 Functional Block Diagram

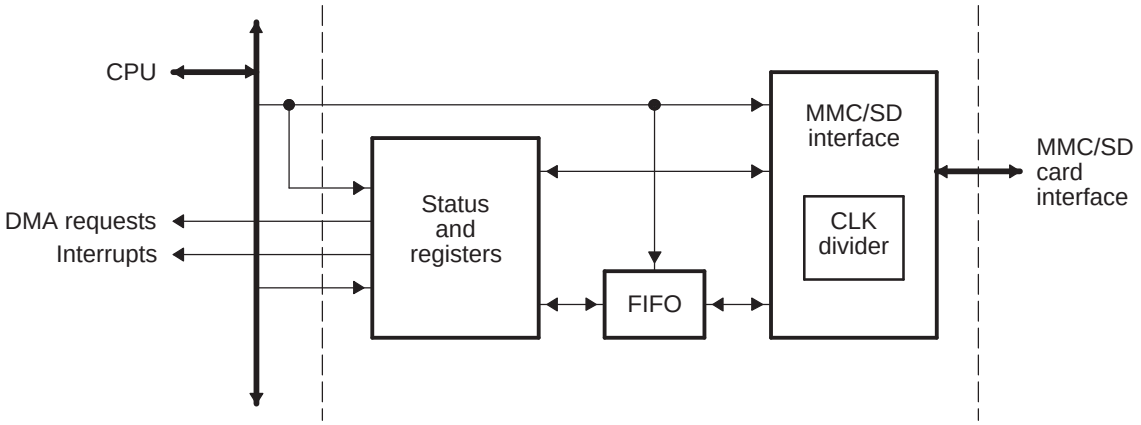
The MMC/SD card controller transfers data between the CPU, DMA, and MMC/SD as shown in Figure 10-1. The CPU and DMA controller can read/write the data in the card by accessing the registers in the MMC/SD controller.

10.1.4 Supported Use Case Statement

The MMC/SD card controller supports the following user cases:

- MMC/SD card identification
- MMC/SD single-block read using CPU
- MMC/SD single-block read using DMA
- MMC/SD single-block write using CPU
- MMC/SD single-block write using DMA
- MMC/SD multiple-block read using CPU
- MMC/SD multiple-block read using DMA
- MMC/SD multiple-block write using CPU
- MMC/SD multiple-block write using DMA
- SDIO function

Figure 10-1. MMC/SD Card Controller Block Diagram



10.1.5 Industry Standard(s) Compliance Statement

The MMC/SD card controller will support the following industry standards (with the exception noted below):

- Multimedia Card (MMC) Specification V3.31.
- Secure Digital (SD) Physical Layer Specification V2.0.
- Secure Digital Input Output (SDIO) Specification V2.0.

The information in this document assumes the reader is familiar with these standards.

The MMC/SD controller does not support the SPI mode of operation.

10.2 Peripheral Architecture

The MMC/SD controller uses the MMC/SD protocol to communicate with the MMC/SD cards. The MMC/SD controller can be configured to work as an MMC or SD controller, based on the type of card with which the controller is communicating. Figure 10-2 summarizes the MMC/SD mode interface. Figure 10-3 illustrates how the controller is interfaced to the cards in MMC/SD mode.

In the MMC/SD mode, the controller supports one or more MMC/SD cards. When multiple cards are connected, the MMC/SD controller selects one by using identification broadcast on the data line. The following MMC/SD controller pins are used:

- CMD: This pin is used for two-way communication between the connected card and the MMC/SD controller. The MMC/SD controller transmits commands to the card and the memory card drives responses to the commands on this pin.
- DAT0 or DAT0-3: MMC cards use only one data line (DAT0) and SD cards use one or four data lines. The number of DAT pins (the data bus width) is set by the WIDTH bit in the MMC Control Register (MMCCTL), see).
- CLK: This pin provides the clock to the memory card from the MMC/SD controller.

Figure 10-2. MMC/SD Controller Interface Diagram

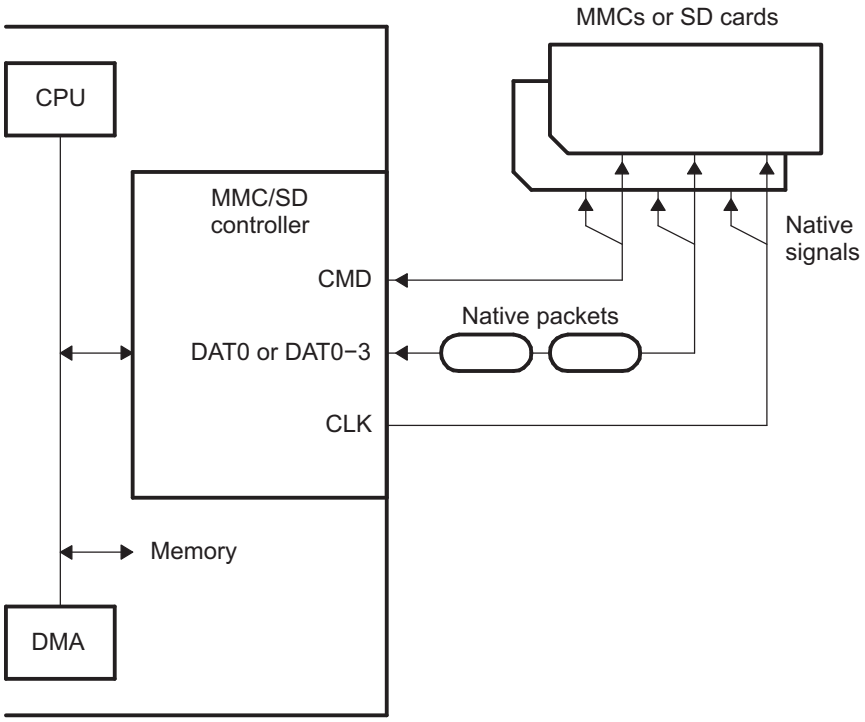
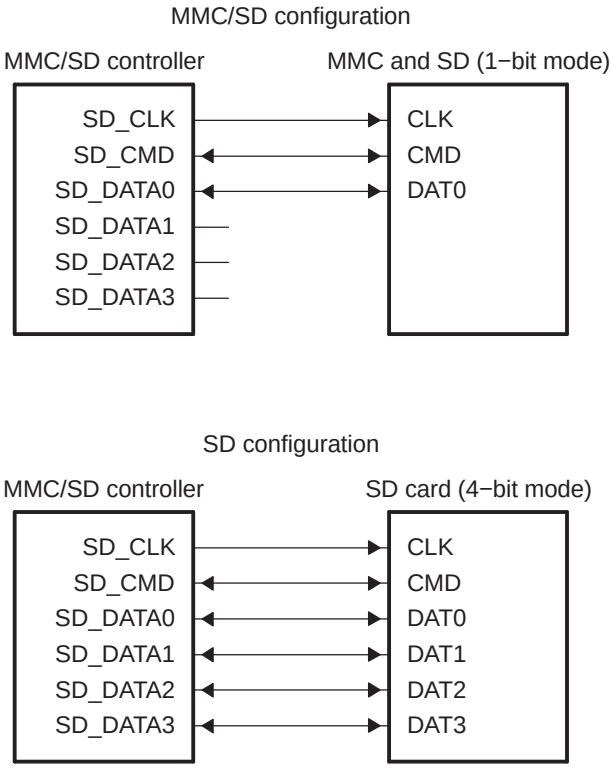


Figure 10-3. MMC Configuration and SD Configuration Diagram



10.2.1 Clock Control

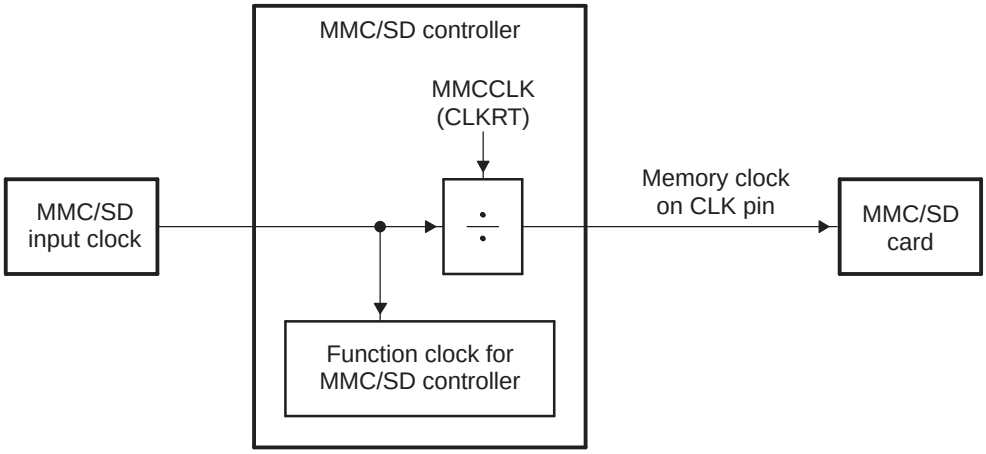
The MMC/SD controller has two clocks: the function clock and the memory clock (see Figure 10-4).

The function clock determines the operational frequency of the MMC/SD controller and is the input clock to the MMC/SD controller on the device. The functional clock of MMCSD controller is capable of operating up to system clock frequency.

The memory clock appears on the SD_CLK pin of the MMC/SD controller interface. The memory clock controls the timing of communication between the MMC/SD controller and the connected memory card. The memory clock is generated by dividing the function clock in the MMC/SD controller. The divide-down value is set by CLKRT bits in the MMC memory clock control register (MMCCLK) and is determined by the equation:

memory clock frequency = function clock frequency/(2 * (CLKRT + 1))

Figure 10-4. MMC/SD Controller Clocking Diagram



(1) Maximum memory clock frequency for MMC card can be 20 MHz.

(2) Maximum memory clock frequency for SD card can be 50 MHz.

10.2.2 Signal Descriptions

Table 10-1 shows the MMC/SD controller pins used in each mode. The MMC/SD protocol uses the clock, command (two-way communication between the MMC controller and memory card), and data (DAT0 for MMC card, DAT0-3 for SD card) pins.

Table 10-1. MMC/SD Controller Pins Used in Each Mode

		Function	
Pin	**Type[1]**	**MMC Communications**	**SD Communications**
CLK	O	Clock line	Clock line
CMD	I/O	Command line	Command line
DAT0	I/O	Data line 0	Data line 0
DAT1	I/O	(Not used)	Data line 1
DAT2	I/O	(Not used)	Data line 2
DAT3	I/O	(Not used)	Data line 3

(1) I = input to the MMC/SD controller; O = output from the MMC/SD controller.

10.2.3 Pin Multiplexing

The MMC/SD card controller pins are multiplexed with other peripherals on the DSP. Before using the controller, the DSP should be configured to route the MMC/SD card controller signals to the multiplexed Serial Port 0 or Serial Port 1 pins by writing to the External Bus Selection Register (EBSR). For more information on pin multiplexing, see Section 1.1, *System Control*.

NOTE: Configuring the EBSR to route the MMC/SD0 or MMC/SD1 signals to Serial Port0 or Serial Port1, respectively, also routes those MMC/SD interrupts to the CPU.

10.2.4 Protocol Descriptions

The MMC/SD controller follows the MMC/SD protocol for completing any kind of transaction with the Multimedia Card and secure digital cards. For detailed information, refer to the supported MMC and SD specifications in Section 10.1.5.

10.2.4.1 MMC/SD Mode Write Sequence

Figure 10-5 and Table 10-2 show the signal activity when the MMC/SD controller is in the MMC/SD mode and is writing data to a memory card. Before initiating a write transfer, ensure that the block length definition in the MMC/SD controller and the memory card are identical.

- The MMC/SD controller sends a write command to the card.
- The card receives the command and sends responses to the command.
- The MMC/SD controller sends a block of data to the card.
- The card sends the CRC status to the MMC/SD controller.
- The card sends a low BUSY bit until all the data has been programmed into the flash memory inside the card.

Figure 10-5. MMC/SD Mode Write Sequence Timing Diagram

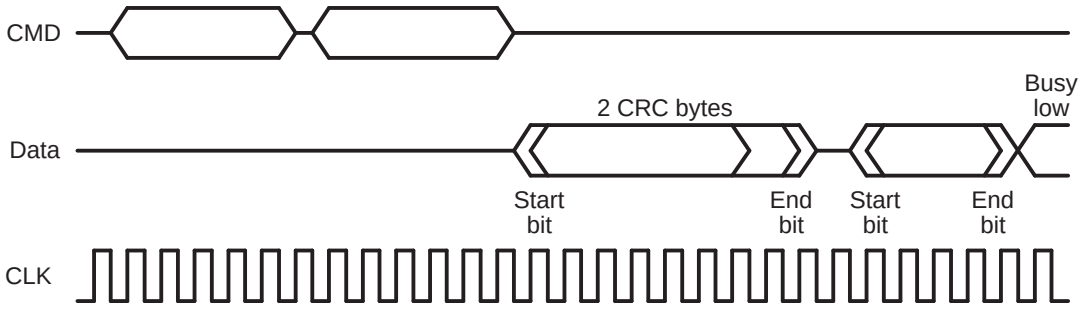


Table 10-2. MMC/SD Mode Write Sequence

Portion of the Sequence	Description
WR CMD	Write command: A 6-byte WRITE_BLOCK command token is sent from the CPU to the card.
CMD RSP	Command response: The card sends a 6-byte response of type R1 to acknowledge the WRITE_BLOCK to the CPU.
DAT BLK	Data block: The CPU writes a block of data to the card. The data content is preceded by one start bit and is followed by two CRC bytes and one end bit.
CRC STAT	CRC status: The card sends a one byte CRC status information, which indicates to the CPU whether the data has been accepted by the card or rejected due to a CRC error. The CRC status information is preceded by one start bit and is followed by one end bit.
BUSY	BUSY bit: The CRC status information is followed by a continuous stream of low busy bits until all of the data has been programmed into the flash memory on the card.

10.2.4.2 MMC/SD Mode Read Sequence

Figure 10-6 and Table 10-3 show the signal activity when the MMC controller is in the MMC/SD mode and is reading data from a memory card. Before initiating a read transfer, ensure that the block length definition in the MMC/SD controller and the memory card are identical. In a successful read sequence, the following steps occur:

- The MMC/SD controller sends a read command to the card.
- The card drives responses to the command.
- The card sends a block of data to the CPU.

Figure 10-6. MMC/SD Mode Read Sequence Timing Diagram

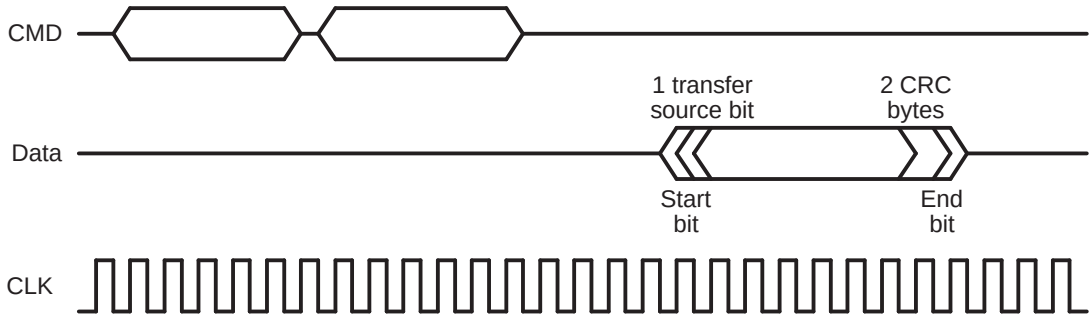


Table 10-3. MMC/SD Mode Read Sequence

Portion of the Sequence	Description
RD CMD	Read command: A 6-byte READ_SINGLE_BLOCK command token is sent from the CPU to the card.
CMD RSP	Command response: The card sends a response of type R1 to acknowledge the READ_SINGLE_BLOCK command to the CPU.
DAT BLK	Data block: The card sends a block of data to the CPU. The data content is preceded by a start bit and is followed by two CRC byte and an end bit.

10.2.5 Data Flow in the Input/Output FIFO

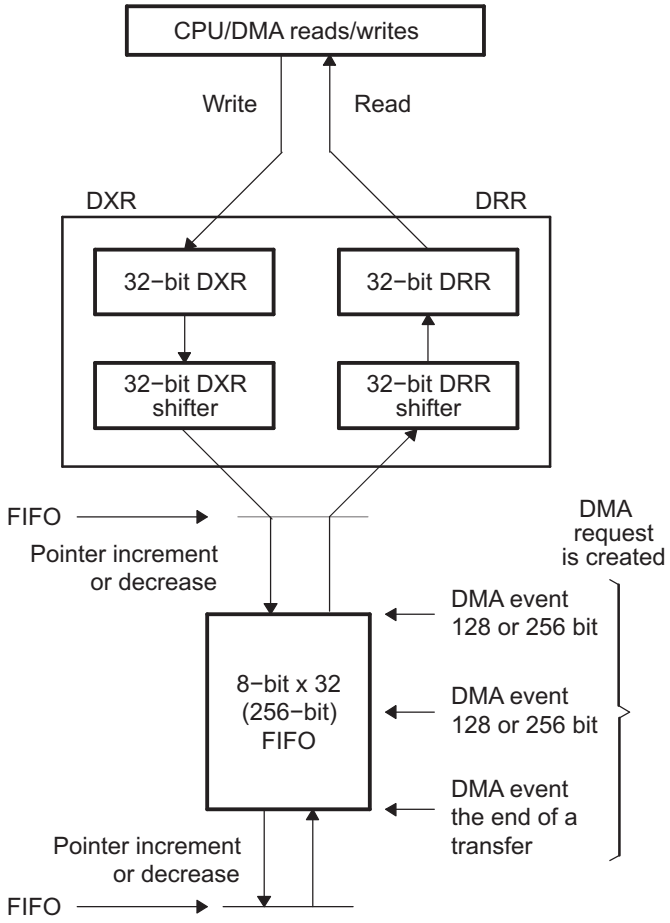
The MMC/SD controller contains a single 256-bit FIFO that is used for both reading data from the memory card and writing data to the memory card (see Figure 10-7). The FIFO is organized as 32 eight-bit entries. The conversion from the 32-bit bus to the byte format of the FIFO follows the little-endian convention (details are provided in later sections). The FIFO includes logic to generate DMA events and interrupts based on the amount of data available in the FIFO. FIFO depth (threshold) is a programmable number that describes how many bytes can be received/transmitted at a time. There are also flags set when the FIFO is full or empty. A high-level operational description is:

- Data is written to the FIFO through the MMC Data Transmit Registers (MMCDXR1 and 2). Data is read from the FIFO through the MMC Data Receive Registers (MMCDRR1 and 2).
- The ACCWD bits in the MMC FIFO Control Register (MMCFIFOCTL) determines the behavior of the FIFO full (FIFOFUL) and FIFO empty (FIFOEMP) status flags in the MMC Status Register 1 (MMCST1):
  - If ACCWD = 00b:
    - FIFO full is active when the write pointer + 4 > read pointer
    - FIFO empty is active when the write pointer - 4 < read pointer
  - If ACCWD = 01b:
    - FIFO full is active when the write pointer + 3 > read pointer
    - FIFO empty is active when the write pointer - 3 < read pointer
  - If ACCWD = 10b:
    - FIFO full is active when the write pointer + 2 > read pointer

- FIFO empty is active when the write pointer - 2 < read pointer

– If ACCWD = 11b:
  - FIFO full is active when the write pointer + 1 > read pointer
  - FIFO empty is active when the write pointer - 1 < read pointer

Figure 10-7. FIFO Operation Diagram



Transmission of data

Step 1: Reset FIFO
Step 2: Set FIFO direction
Step 3: DMA driven transaction
Step 4: DMA sends xmit data
Step 5: If DXR ready is active, 32−bit DXR -> FIFO
Step 6: CPU driven transaction: Fill the FIFO by writing to MMCDXR (only first time) or every 128 or 256−bits transmitted and DXRDYINT interrupt is generated

Reception of data

Step 1: Reset FIFO
Step 2: Set FIFO direction
Step 3: DMA driven transaction
Step 4: FIFO-> 32-bit DRR
Step 5: DRRDYINT interrupt occur when FIFO every 128 or 256−bits of data received by FIFO
Step 6: DMA reads reception data

10.2.6 Data Flow in the Data Registers (MMCDRR and MMCDXR)

The CPU can read 16 bits and the DMA controller can read 32 bits at a time from the FIFO by reading the MMC data receive registers (MMCDRR1 and 2) and, similarly, write to the FIFO by writing to the MMC data transmit registers (MMCDXR1 and 2). However, since the memory card is an 8-bit device, it transmits or receives one byte at a time. Figure 10-8 and Figure 10-9 shows how the data-size difference is handled by the data registers.

Figure 10-8. Little-Endian Access to MMCDXR/MMCDRR1 and 2 From the CPU or the DMA

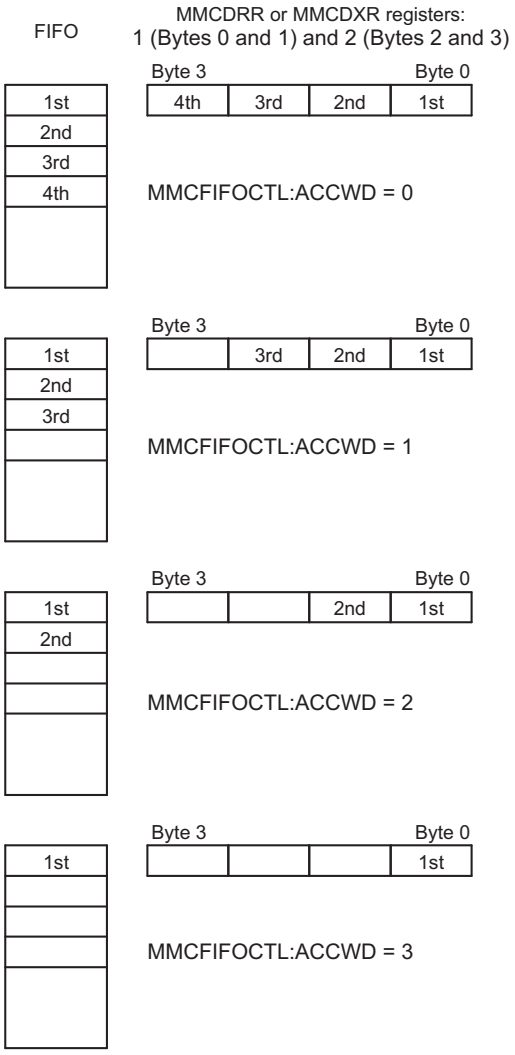
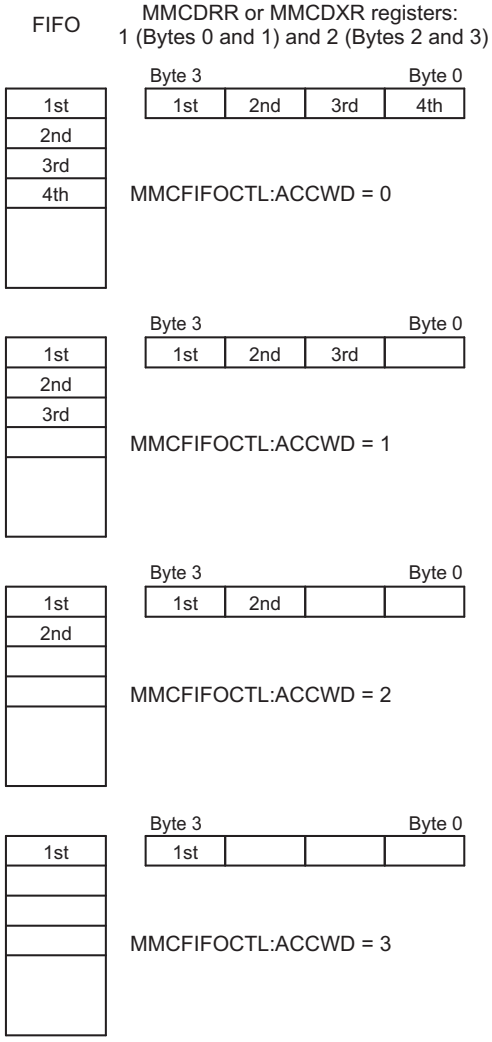


Figure 10-9. Big-Endian Access to MMCDXR/MMCDRR1 and 2 From the CPU or the DMA



10.2.7 FIFO Operation During Card Read Operation

The MMC/SD controller supports 1-, 2-, 3-, or 4-byte reads as shown in Figure 10-8 and Figure 10-9. The CPU makes 16-bit and the DMA makes 32-bit accesses to the MMCDRR registers.

10.2.7.1 DMA Reads

The FIFO controller manages the activities of reading the data in from the card and issuing DMA read events. Each time a DMA read event is issued, a DMA read request interrupt is also generated.

Figure 10-10 provides details of the FIFO controllers operation. As data is received from the card, it is read into the FIFO. When the number of bytes of data received is equal to the level set by the FIFOLEV bits in MMCFIFOCTL, a DMA read event is issued and new DMA events are disabled. Data is read from the FIFO by way of MMCDRR registers (MMCDRR1 should be used as the destination address in the DMA configuration). The FIFO controller continues to read in data from the card while checking for the DMA event to occur or the FIFO to become full. Once the DMA event finishes, new DMA events are enabled. If the FIFO fills up, the FIFO controller stops the MMC/SD controller from reading anymore data until the FIFO is no longer full.

A DMA read event is also generated when the last data arrives as determined by the MMC block length register (MMCBLEN) and the MMC number of blocks register (MMCNBLK) settings. This DMA event enables the FIFO to be flushed of all the data that was read from the card.

Each time a DMA read event is generated, an interrupt (DRRDYINT) is also generated (if enabled in the MMC Interrupt Mask Register (MMCIM) register) and the DRRDY bit in the MMC status register 0 (MMCST0) is also set.

10.2.7.2 CPU Reads

The system CPU can also directly read the card data by reading the MMC data receive register (MMCDRR 1 and/or 2) based on the ACCWD field in the MMCFIFOCTL. Data is ready to be read when the DRRDYINT interrupt is posted or when the DRRDY bit in the MMCST0 register is set.

10.2.8 FIFO Operation During Card Write Operation

The MMC/SD controller supports 1-, 2-, 3-, or 4-byte writes as shown in Figure 10-8 and Figure 10-9. The CPU makes 16-bit and the DMA makes 32-bit accesses to the MMCDXR registers.

10.2.8.1 DMA Writes

The FIFO controller manages the activities of accepting data from the CPU or DMA and passing the data to the MMC/SD controller. The FIFO controller issues DMA write events as appropriate. However, the first DMA event has to be manually generated by setting the DMATRIG bit in the MMC Command Register (MMCSD2) after the desired write command is written to the MMCSD1 register.

Figure 10-11 provides details of the FIFO controller's operation. Data is written by the DMA to the FIFO by the way of MMCDXR registers (MMCDXR1 should be used as the destination address in the DMA configuration). The FIFO then passes the data to the MMC/SD controller which manages to write the data to the card. When the number of bytes of data in the FIFO is less than the level set by the FIFOLEV bits in MMCFIFOCTL, a DMA write event is issued and new DMA events are disabled. The FIFO controller continues to transfer data to the MMC/SD controller while checking for the DMA event to finish or for the FIFO to become empty. Once the DMA event completes, new DMA events are enabled. If the FIFO becomes empty, the FIFO controller informs the MMC/SD controller.

Each time a DMA write event is generated, an interrupt (DXRDYINT) is also generated (if enabled in the MMC Interrupt Mask Register (MMCIM) register) and the DXRDY bit in the MMC status register 0 (MMCST0) is also set.

10.2.8.2 CPU Writes

The system CPU can also directly write the card data by writing to the MMC data transmit register (MMCDXR 1 and/or 2) based on the ACCWD field in the MMCFIFOCTL. Data is ready to be written when the DXRDYINT interrupt is posted or when the DXRDY bit in the MMCST0 register is set.

Figure 10-10. FIFO Operation During Card Read Diagram

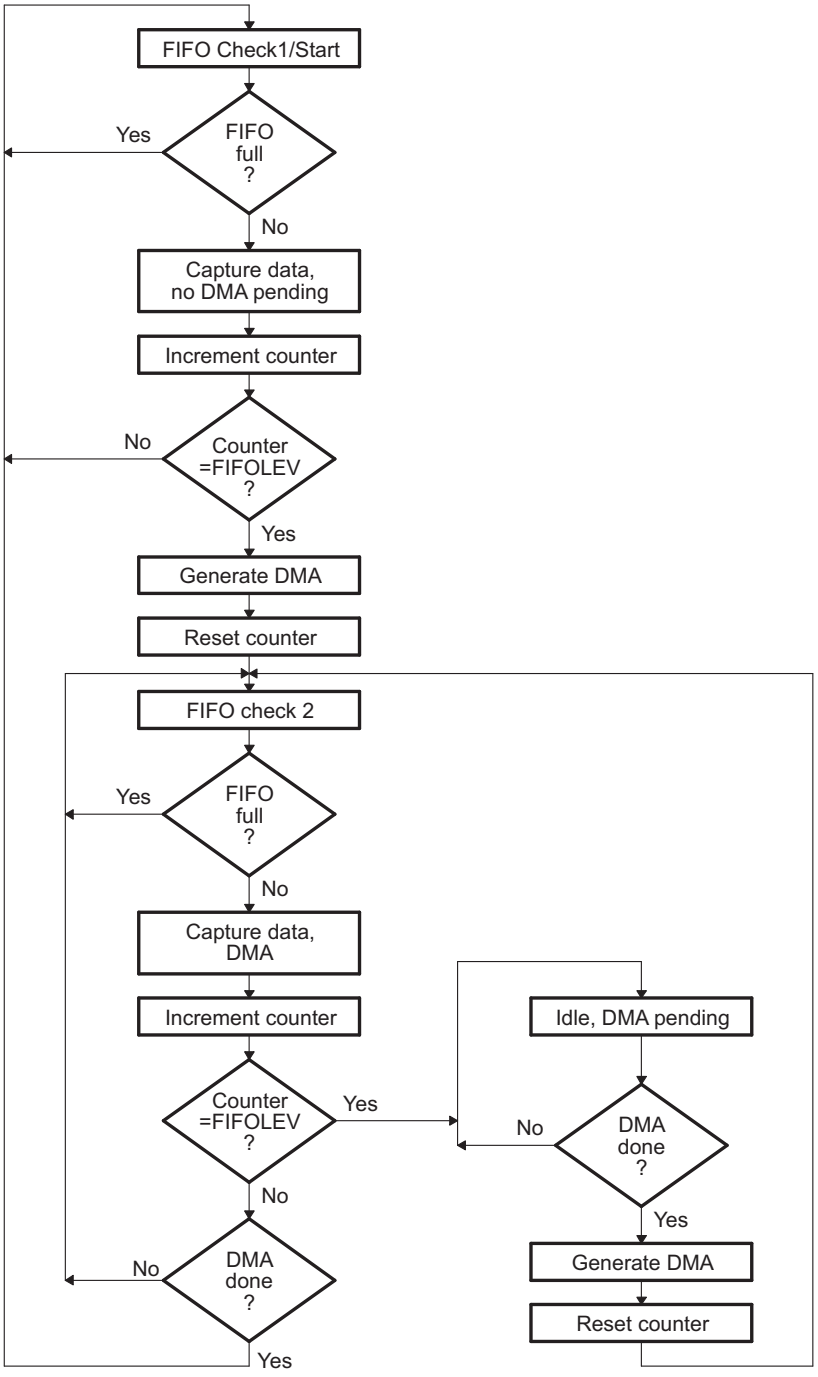
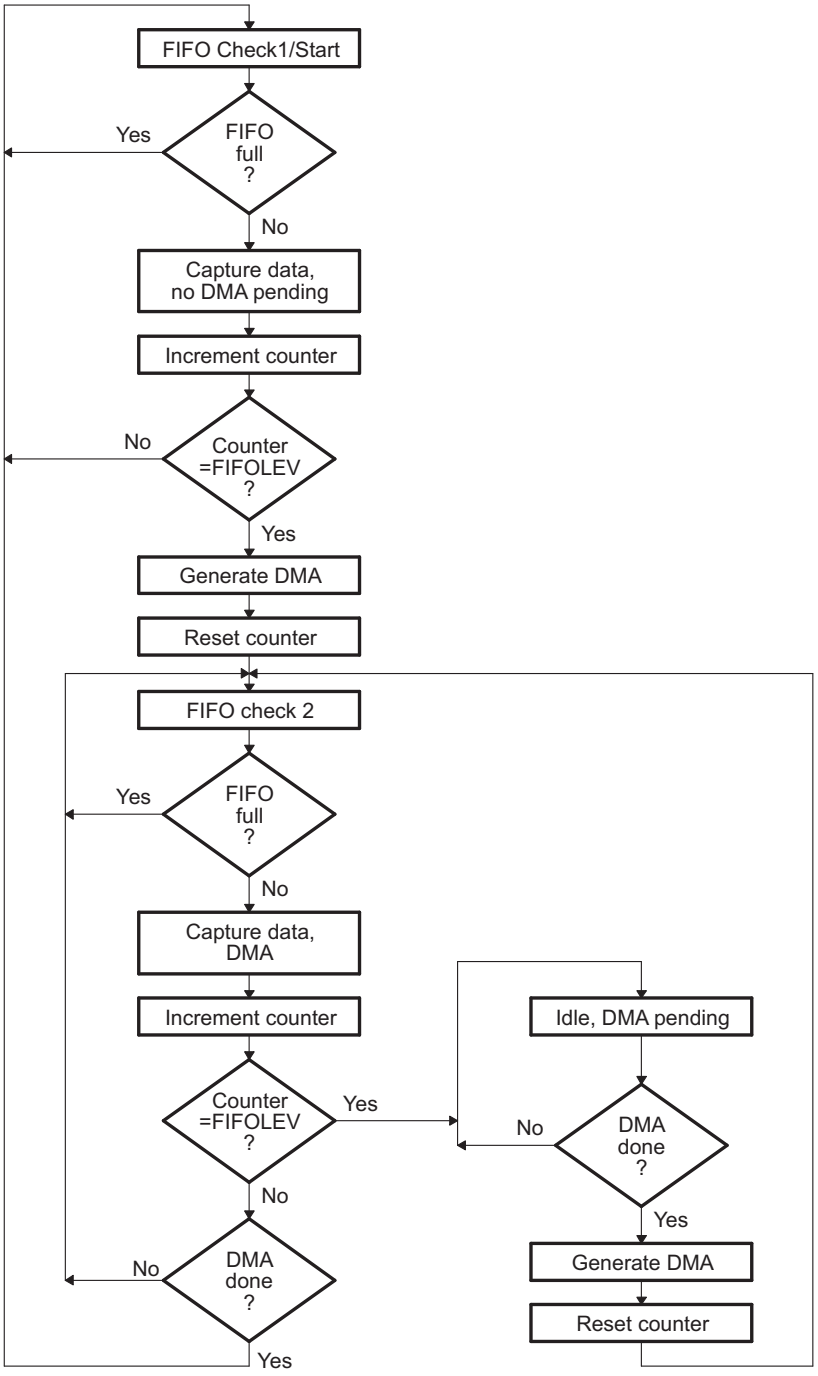


Figure 10-11. FIFO Operation During Card Write Diagram



10.2.9 Reset Considerations

The MMC/SD controller has two reset sources: hardware reset and software reset.

10.2.9.1 Software Reset Considerations

A software reset (such as a reset generated by the emulator) will not cause the MMC/SD controller registers to be altered. After a software reset, the MMC/SD controller continues to operate as it was configured prior to the reset.

10.2.9.2 Hardware Reset Considerations

A hardware reset of the processor will cause the MMC/SD controller registers to return to their default values after reset.

10.2.10 Programming and Using the MMCSD Controller

10.2.10.1 MMC/SD Mode Initialization

The general procedure for initializing the MMC/SD controller is given in the following steps. Details about the registers or register bit fields to be configured in the MMC/SD mode are in the subsequent subsections.

1. Place the MMC/SD controller in its reset state by setting the CMDRST bit and DATRST bit in the MMCCTL. After the reset, other bits in MMCCTL can be set.
2. Write the required values to MMC/SD controller registers to complete the MMC/SD controller configuration.
3. Clear the CMDRST bit and DATRST bit in MMCCTL to release the MMC/SD controller from its reset state. It is recommended not to rewrite the values written to the other bits of MMCCTL in step 1.
4. Enable the SD_CLK pin so that the memory clock is sent to the memory card by setting the CLKEN bit in the MMC memory clock control register (MMCCLK).

NOTE: The External Bus Selection Register must be configured to enable MMC/SD signals at the pins as described in Section 10.2.3 before the controller can communicate with the MMC/SD card.

10.2.10.2 Initializing the MMC Control Register (MMCCTL)

When operating the MMC/SD controller in the MMC/SD mode, the bits in the MMC control register (MMCCTL) affect the operation of the MMC/SD controller. The subsections that follow help you decide how to initialize each of the control register bits.

The DATEG bits in MMCCTL enable or disable edge detection on the SD_DATA3 pin. If edge detection is enabled and an edge is detected, the DATEG flag bit in the MMC status register 0 (MMCST0) is set. In addition, if the EDATED bit in the MMC interrupt mask register (MMCIM) is set, an interrupt request is generated.

In the MMC/SD mode, the MMC/SD controller must know how wide the data bus must be for the memory card that is connected. If an MMC card is connected, specify a 1-bit data bus (WIDTH = 0 in MMCCTL); if an SD card is connected, specify a 4-bit data bus (WIDTH = 1 in MMCCTL).

To place the MMC/SD controller in its reset state and disable it, set the CMDRST bit and DATRST bit in MMCCTL. The first step of the MMC/SD controller initialization process is to disable both sets of logic. When initialization is complete but before you enable the SD_CLK pin, enable the MMC/SD controller by clearing the CMDRST bit and DATRST bit in MMCCTL.

10.2.10.3 Initializing the Clock Controller Registers (MMCCLK)

A clock divider in the MMC/SD controller divides-down the function clock to produce the memory clock. Load the divide-down value into the CLKRT bits in the MMC memory clock control register (MMCCLK). The divide-down value is determined by the equation:

memory clock frequency = function clock frequency/(2 * (CLKRT + 1))

The CLKEN bit in MMCCLK determines whether the memory clock appears on the SD_CLK pin. If CLKEN is cleared to 0, the memory clock is not provided except when required.

10.2.10.4 Initialize the Interrupt Mask Register (MMCIM)

The bits in the MMC interrupt mask register (MMCIM) individually enable or disable the interrupt requests. To enable the associated interrupt request, set the corresponding bit in MMCIM. To disable the associated interrupt request, clear the corresponding bit. Load zeros into the bits not used in the MMC/SD mode.

10.2.10.5 Initialize the Time-Out Registers (MMCTOR and MMCTOD)

Specify the time-out period for responses using the MMC response time-out register (MMCTOR) and the time-out period for reading data using the MMC data read time-out register (MMCTOD).

When the MMC/SD controller sends a command to a memory card, it often must wait for a response. The MMC/SD controller can wait indefinitely or up to 255 memory clock cycles. If you load 0 into MMCTOR, the MMC/SD controller waits indefinitely for a response. If you load a nonzero value into MMCTOR, the MMC/SD controller stops waiting after the specified number of memory clock cycles and then sets a response time-out flag (TOUTRS) in the MMC status register 0 (MMCST0). If the associated interrupt request is enabled, the MMC/SD controller also sends an interrupt request to the CPU.

When the MMC/SD controller requests data from a memory card, it can wait indefinitely for that data or it can stop waiting after a programmable number of cycles. If you load 0 into MMCTOD, the MMC/SD controller waits indefinitely. If you load a nonzero value into MMCTOD, the MMC/SD controller waits the specified number of memory clock cycles and then sets a read data time-out flag (TOUTRD) in MMCST0. If the associated interrupt request is enabled, the MMC/SD controller also sends an interrupt request to the CPU.

10.2.10.6 Initialize the Data Block Registers (MMCBLEN and MMCNBLK)

Specify the number of bytes in a data block in the MMC block length register (MMCBLEN) and the number of blocks in a multiple-block transfer in the MMC number of blocks register (MMCNBLK).

In MMCBLEN, you must define the size for each block of data transferred between the MMC/SD controller and a memory card. The valid size depends on the type of read/write operations. A length of 0 bytes is prohibited.

For multiple-block transfers, you must specify how many blocks of data are to be transferred between the MMC/SD controller and a memory card. You can specify an infinite number of blocks by loading 0 into MMCNBLK. When MMCNBLK = 0, the MMC/SD controller continues to transfer data blocks until the transferring is stopped with a STOP_TRANSMISSION command. To transfer a specific number of blocks, load MMCNBLK with a value from 1 to 65535.

For high capacity cards (2 GB and larger), by default the read/write block length of the card is 1024 bytes. Note that CMD16 (SET_BLOCK_LEN) can only set the block length up to 512 bytes even for high capacity cards. Therefore, if you want to change the block length of a high capacity card, you are limited to 512 bytes. In this case, you should discard the block length read from the CSD register in the card and set the block length up to 512 bytes.

10.2.10.7 Using the Command Registers (MMCSD1 and MMCSD2)

The MMCSD1 register can be programmed to choose the type command to be sent to the MMC/SD card and the expected outcome of the transaction. Any writes to this register triggers the controller to send a command to the MMC/SD card as programmed in the CMD field. This behavior makes it necessary to program the whole register in a single write.

The DMATRIG field is a write-only field in the MMCSD2 register. It is only used for write operations involving the DMA. Setting this bit field triggers the associated DMA channel to transfer data to the controller FIFO while the MMC/SD card prepares itself for the write operation. Subsequent DMA triggers are automatically generated when the controller writes the data in the FIFO out to the MMC/SD card and do not need the DMATRIG bit to be set. This field should only be used for write operations involving the DMA. Data read operations do not require this intervention.

NOTE: You must only write to the DMATRIG bit in MMCSD2 after the desired write command has been written to the MMCSD1 register. If this bit is written to at any other time, the controller will resend the last command (configured in the MMCSD1 register) to the card.

10.2.10.8 Monitoring Activity in the MMC/SD Mode

This section describes registers and specific register bits that you use to obtain the status of the MMC/SD controller in the MMC/SD mode. The status of the MMC/SD controller is determined by reading the bits in the MMC status register 0 (MMCST0) and MMC status register 1 (MMCST1).

10.2.10.8.1 Detecting Edges on the DAT3 Pin

The MMC/SD controller sets the DATED bit in MMCST0 if SD_DATA3 edge detection is enabled (DATEG bits are nonzero in MMCCTL) and the specified edge is detected. The CPU is also notified of the SD_DATA3 by an interrupt if the interrupt request is enabled (EDATED = 1 in MMCIM).

10.2.10.8.2 Detecting Level Changes on the DAT3 Pin

The DAT3ST bit in MMCST1 monitors the signal level on the SD_DATA3 pin.

10.2.10.8.3 Determining Whether New Data is Available in MMCDRR Registers

The MMC/SD controller sets the DRRDY bit in MMCST0 when data in the FIFO is greater than the threshold set in MMCFIFOCTL. The CPU is also notified of the event by an interrupt if the interrupt request is enabled (EDRRDY = 1 in MMCIM). The DRRDY flag is cleared by a read of MMCDDR register.

10.2.10.8.4 Verifying that MMCDXR is Ready to Accept New Data

The MMC/SD controller sets the DXRDY bit in MMCST0 when the amount of data in the FIFO is less than the threshold set in MMCFIFOCTL. The CPU is also notified of the event by an interrupt if the interrupt request is enabled (EDXRDY = 1 in MMCIM).

10.2.10.8.5 Checking for CRC Errors

The MMC/SD controller sets the CRCRS, CRCRD, and CRCWR bits in MMCST0 in response to the corresponding CRC errors of command response, data read, and data write. The CPU is also notified of the CRC error by an interrupt if the interrupt request is enabled (ECRCRS/ECRCRD/ECRCWR = 1 in MMCIM).

10.2.10.8.6 Checking for Time-Out Events

The MMC/SD controller sets the TOUTRS and TOUTRD bits in MMCST0 in response to the corresponding command response or data read time-out event. The CPU is also notified of the event by an interrupt if the interrupt request is enabled (ETOUTRS/ETOUTRD = 1 in MMCIM).

10.2.10.8.7 Determining When a Response/Command is Done

The MMC/SD controller sets the RSPDNE bit in MMCST0 when the response is done or, in the case of commands that do not require a response, when the command is done. The CPU is also notified of the done condition by an interrupt if the interrupt request is enabled. (ERSPDNE = 1 in MMCIM).

10.2.10.8.8 Determining Whether the Memory Card is Busy

The card sends a busy signal either when waiting for an R1b-type response or when programming the last write data into its flash memory. The MMC/SD controller has two flags to notify you whether the memory card is sending a busy signal. The two flags are complements of each other:

- BSYDNE flag in MMCST0 is set if the card did not send or is not sending a busy signal when the MMC/SD controller is expecting a busy signal (BSYEXP = 1 in MMCSD). The interrupt by this bit is enabled by a corresponding interrupt enable bit (EBSYDNE = 1 in MMCIM).
- BUSY flag in MMCST1 is set when a busy signal is received from the card.

10.2.10.8.9 Determining Whether a Data Transfer is Done

The MMC/SD controller sets the DATDNE bit in MMCST0 when all the bytes of a data transfer have been transmitted/received. The DATDNE bit is polled to determine when to stop writing to the data transmit register (for a write operation) or when to stop reading from the data receive register (for a read operation). The CPU is also notified of the time-out event by an interrupt if the interrupt request is enabled (EDATDNE = 1 in MMCIM).

10.2.10.8.10 Determining When Last Data has Been Written to Card (SanDisk SD cards)

Some SanDisk brand SD™ cards exhibit a behavior that requires a multiple-block write command to be terminated with a STOP (CMD12) command before the data write sequence is completed. To enable support of this function, the transfer done interrupt (TRNDNE) is provided. The TRNDNE interrupt is enabled by setting the ETRNDNE bit in MMCIM. This interrupt is issued when the last byte of data (as defined by MMCNBLK and MMCBLEN) is transferred from the FIFO to the output shift register. The CPU should respond to this interrupt by sending a STOP command to the card. This interrupt differs from DATDNE by timing. DATDNE does not occur until after the CRC and memory programming are completed.

10.2.10.8.11 Checking For a Data Transmit Empty Condition

During transmission, a data value is passed from the MMC data transmit registers (MMCDXR1 and 2) to the data transmit shift register. The data is then passed from the shift register to the memory card one bit at a time. The DXEMP bit in MMCST1 indicates when the shift register is empty.

Typically, the DXEMP bit is not used to control data transfers; rather, it is checked during recovery from an error condition. There is no interrupt associated with the DXEMP bit.

10.2.10.8.12 Checking for a Data Receive Full Condition

During reception, the data receive shift register accepts a data value one bit at a time. The entire value is then passed from the shift register to the MMC data receive registers (MMCDRR1 and 2). The DRFUL bit in MMCST1 indicates when the shift register is full; no new bits can be shifted in from the memory card.

Typically, the DRFUL bit is not used to control data transfers; rather, it is checked during recovery from an error condition. There is no interrupt associated with the DRFUL bit.

10.2.10.8.13 Checking the Status of the SD_CLK Pin

Read the CLKSTP bit in MMCST1 to determine whether the memory clock has been stopped on the SD_CLK pin.

10.2.10.8.14 Checking the Remaining Block Count During a Multiple-Block Transfer

During a transfer of multiple data blocks, the MMC number of blocks counter register (MMCNBLC) indicates how many blocks are remaining to be transferred. MMCNBLC is a read-only register.

10.2.11 Interrupt Support

10.2.11.1 Interrupt Events and Requests

The MMC/SD controller generates the interrupt requests described in Table 10-4. When an interrupt event occurs, its flag bit is set in the MMC status register 0 (MMCST0). If the enable bits corresponding to each flag are set in the MMC interrupt mask register (MMCIM), an interrupt request is generated. All such requests are multiplexed to a single MMC/SD interrupt request from the MMC/SD controller to the CPU.

The MMC/SD interrupts can be masked into the CPU by means of the 4 programmable interrupt sources. This is accomplished through the External Bus Selection Register. Selecting Serial Port 0 or 1 Mode will route the appropriate I2S or MMC/SD interrupt to the CPU.

The interrupt service routine (ISR) for the SDIO0 interrupt can determine the event that caused the interrupt by checking the bits in MMCST0. When MMCST0 is read (either by CPU or emulation), all of the register bits are automatically cleared.

Table 10-4. Description of MMC/SD Interrupt Requests

Interrupt Request	Interrupt Event
TRNDNEINT	**For read operations:** The MMC/SD controller has received the last byte of data (before CRC check).
	For write operations: The MMC/SD controller has transferred the last word of data to the output shift register.
DATEDINT	An edge was detected on the DAT3 pin.
DRRDYINT	MMCDRR is ready to be read (data in FIFO is above threshold).
DXRDYINT	MMCDXR is ready to transmit new data (data in FIFO is less than threshold).
CRCRSINT	A CRC error was detected in a response from the memory card.
CRCRDINT	A CRC error was detected in the data read from the memory card.
CRCWRINT	A CRC error was detected in the data written to the memory card.
TOUTRSINT	A time-out occurred while the MMC controller was waiting for a response to a command.
TOUTRDINT	A time-out occurred while the MMC controller was waiting for the data from the memory card.
RSPDNEINT	**For a command that requires a response:** The MMC controller has received the response without a CRC error.
	For a command that does not require a response: The MMC controller has finished sending the command.
BSYDNEINT	The memory card stops or is no longer sending a busy signal when the MMC controller is expecting a busy signal.
DATDNEINT	**For read operations:** The MMC controller has received data without a CRC error.
	For write operations: The MMC controller has finished sending data.

10.2.12 DMA Event Support

The MMC/SD controller is capable of generating DMA events for both read and write operations in order to request service from a DMA controller. Based on the FIFO threshold setting, the DMA event signals would be generated every time 128-bit or 256-bit data is transferred from the FIFO.

10.2.13 Emulation Considerations

The MMC/SD controller is not affected by emulation halt events (such as breakpoints).

10.3 Procedures for Common Operations

10.3.1 Card Identification Operation

Before the MMC/SD controller starts data transfers to or from memory cards in the MMC/SD native mode, it has to first identify how many cards are present on the bus and configure them. For each card that responds to the ALL_SEND_CID broadcast command, the controller reads that card's unique card identification address (CID) and then assigns it a relative address (RCA). This address is much shorter than the CID and is used by the MMC/SD controller to identify the card in all future commands that involve the card.

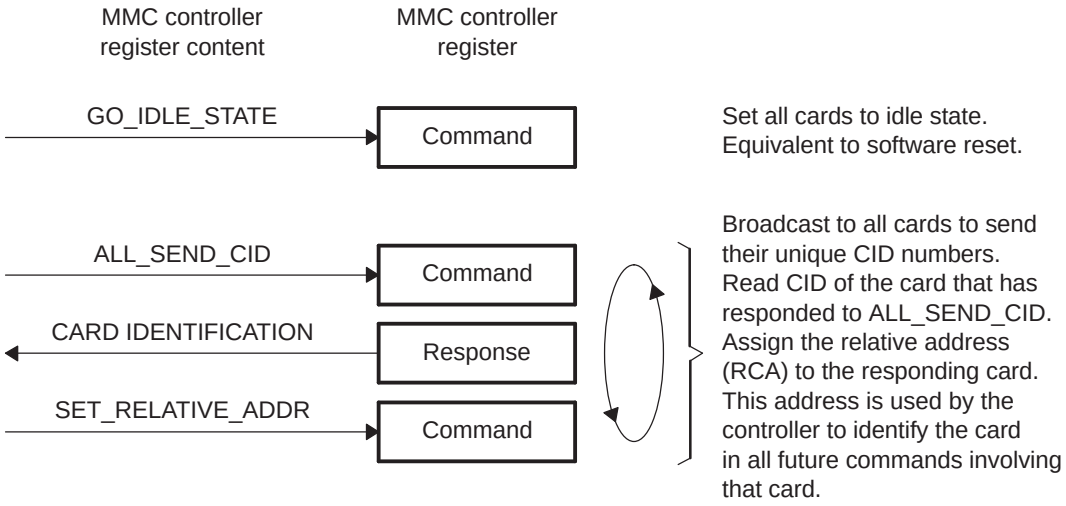
Only one card completes the response to ALL_SEND_CID at any one time. The absence of any response to ALL_SEND_CID indicates that all cards have been identified and configured.

The procedure for a card identification operation is:

1. Use MMCSD1 to send the GO_IDLE_STATE command to the cards. This puts all cards in the idle state. The SEND_IF_COND command should be used next to check for SD card version, followed by the SD_SEND_OP_COND command for host capacity support operating condition information exchange to see if card is standard or high capacity (if card has been identified as ver2.0).
2. Use MMCSD1 to send the ALL_SEND_CID command to the cards. This notifies all cards to identify themselves.
3. Wait for a card to respond. If a card responds, go to step 4; otherwise, stop.
4. Read the CID from the MMC response registers (MMCRSP0–7) and assign a relative address to the card by sending the SET_RELATIVE_ADDR command.

The sequence of events in this operation is shown in Figure 10-12.

Figure 10-12. Card Identification (Native MMC/SD Mode)



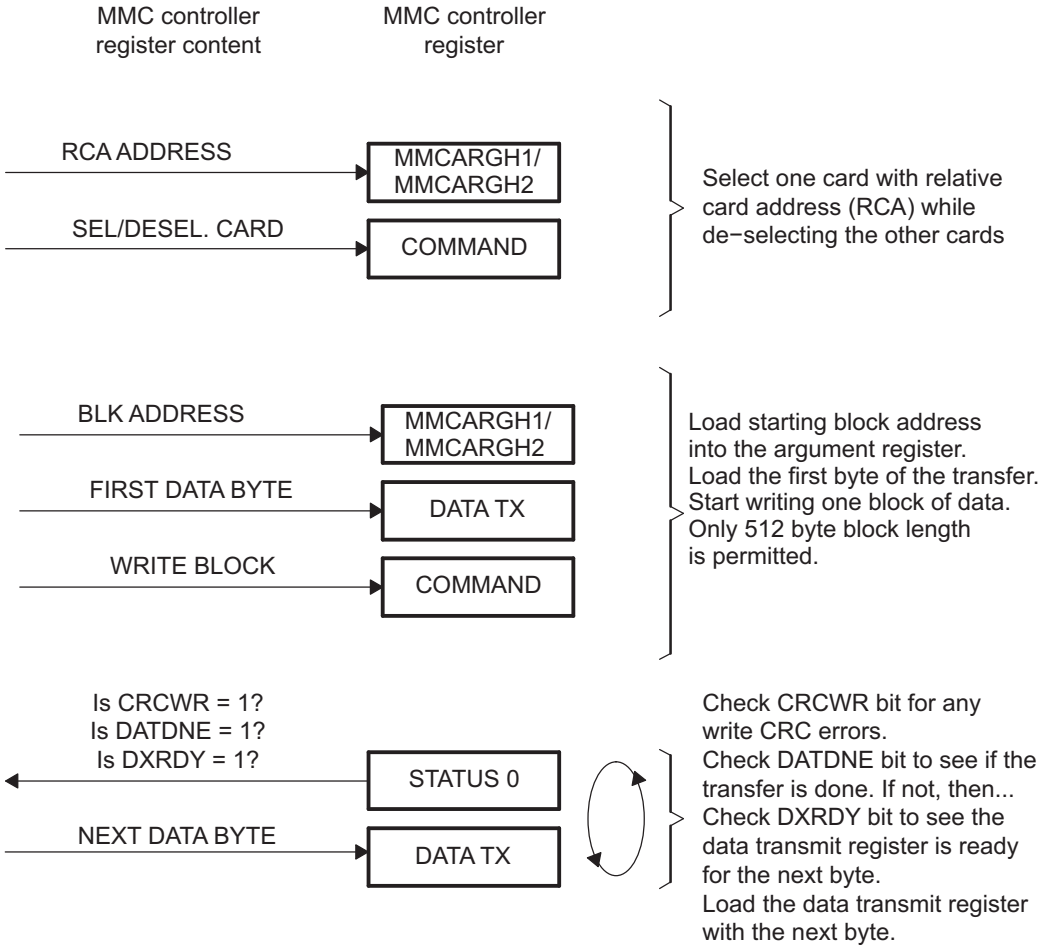
10.3.2 MMC/SD Mode Single-Block Write Operation Using CPU

To perform a single-block write, the block length must be 512 bytes and the same length needs to be set in both the MMC/SD controller and the memory card. The procedure for this operation is:

1. Write the card's relative address to the MMC argument registers (MMCARG1/MMCARG2).
2. Use the MMCSD1 to send the SELECT/DESELECT_CARD broadcast command. This selects the addressed card and deselects the others.
3. Write the destination start address to the MMC argument registers.
4. Read card CSD to determine the card's maximum block length.
5. Use MMCSD1 to send the SET_BLOCKLEN command (if the block length is different than the length used in the previous operation). The block length must be a multiple of 512 bytes and less then the maximum block length specified in the CSD.
6. Reset the FIFO by setting the FIFORST bit in MMCFIFOCTL.
7. Bring the FIFO out of reset by clearing the FIFORST bit and set the FIFO direction to transmit (FIFODIR bit in MMCFIFOCTL).
8. Set the access width (ACCWD bits in MMCFIFOCTL).
9. Enable the MMC interrupt.
10. Enable DXRDYINT interrupt.
11. Write the first 32 bits of the data block to the data transmit register (MMCDXR).
12. Use MMCSD1 to send the WRITE_BLOCK command to the card.
13. Wait for the MMC interrupt
14. Use the MMC status register 0 (MMCST0) to check for errors and the status of the FIFO. If all of the data has not been written and if the FIFO is not full, go to step 15. If all of the data has been written, stop.
15. Write the next *n* bytes (depends on setting of FIFOLEV in MMCFIFOCTL:0 = 16 bytes , 1 = 32 bytes) of the data block to the MMC data transmit register (MMCDXR) and go to step 13.

The sequence of events in this operation is shown in Figure 10-13.

Figure 10-13. MMC/SD Mode Single-Block Write Operation



10.3.3 MMC/SD Mode Single-Block Write Operation Using DMA

To perform a single-block write, the block length must be 512 bytes and the same length needs to be set in both the MMC/SD controller and the card. The procedure for this operation is:

1. Write the card's relative address to the MMC argument registers (MMCARG1/MMCARG2).
2. Read card CSD to determine the card's maximum block length.
3. Use MMCSD1 to send the SET_BLOCKLEN command (if the block length is different than the length used in the previous operation). The block length must be a multiple of 512 bytes and less then the maximum block length specified in the CSD.
4. Reset the FIFO (FIFORST bit in MMCFIFOCTL).
5. Set the FIFO direction to transmit (FIFODIR bit in MMCFIFOCTL).
6. Set the access width (ACCWD bits in MMCFIFOCTL).
7. Set the FIFO threshold (FIFOLEV bit in MMCFIFOCTL).
8. Set up DMA (DMA size needs to be greater than or equal to FIFOLEV setting).
9. Use MMCSD1 to send the WRITE _BLOCK command to the card.
10. Use MMCSD2 to trigger first DMA transfer to FIFO by setting the DMATRIG bit.
11. Wait for DMA sequence to complete or the DATADNE flag in the MMC status register 0 (MMCST0) is set.
12. Use MMCST0 to check for errors.

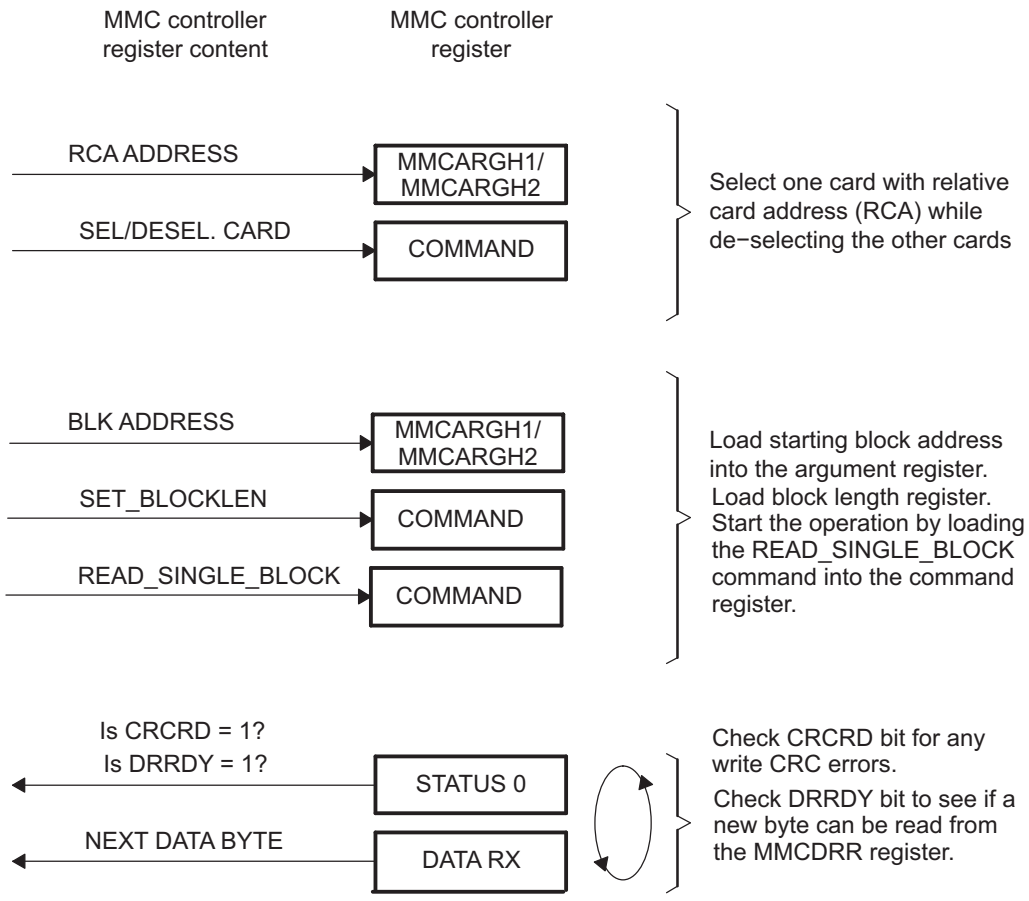
10.3.4 MMC/SD Mode Single-Block Read Operation Using CPU

To perform a single-block read, the same block length needs to be set in both the MMC/SD controller and the card. The procedure for this operation is:

1. Write the card's relative address to the MMC argument registers (MMCARG1/MMCARG2).
2. Use MMCSD1 to send the SELECT/DESELECT_CARD broadcast command. This selects the addressed card and deselects the others.
3. Write the source start address to the MMC argument registers.
4. Read card CSD to determine the card's maximum block length.
5. Use MMCSD1 to send the SET_BLOCKLEN command (if the block length is different than the length used in the previous operation). The block length must be a multiple of 512 bytes and less then the maximum block length specified in the CSD.
6. Reset the FIFO by setting the FIFORST bit in MMCFIFOCTL.
7. Bring the FIFO out of reset by clearing the FIFORST bit and set the FIFO direction to receive (FIFODIR bit in MMCFIFOCTL).
8. Set the access width (ACCWD bits in MMCFIFOCTL).
9. Set the FIFO threshold (FIFOLEV bit in MMCFIFOCTL).
10. Enable the MMC interrupt.
11. Enable DRRDYINT interrupt.
12. Use MMCSD1 to send the READ_SINGLE_BLOCK command.
13. Wait for MMC interrupt.
14. Use the MMC status register 0 (MMCST0) to check for errors and the status of the FIFO. If the FIFO is not empty, go to step 14. If the all of the data has been read, stop.
15. Read the next *n* bytes of data (depends on setting of FIFOLEV in MMCFIFOCTL: 0 = 16 bytes, 1 = 32 bytes) from the MMC data receive register (MMCDRR) and go to step 13.

The sequence of events in this operation is shown in Figure 10-14.

Figure 10-14. eMMC/SD Mode Single-Block Read Operation



10.3.5 MMC/SD Mode Single-Block Read Operation Using DMA

To perform a single-block read, the same block length needs to be set in both the MMC/SD controller and the card. The procedure for this operation is:

1. Write the card's relative address to the MMC argument registers (MMCARG1/MMCARG2).
2. Read card CSD to determine the card's maximum block length.
3. Use the MMCSD1 to send the SET_BLOCKLEN command (if the block length is different than the length used in the previous operation). The block length must be a multiple of 512 bytes and less then the maximum block length specified in the CSD.
4. Reset the FIFO by setting the FIFORST bit in MMCFIFOCTL.
5. Bring the FIFO out of reset by clearing the FIFORST bit and set the FIFO direction to receive (FIFODIR bit in MMCFIFOCTL).
6. Set the access width (ACCWD bits in MMCFIFOCTL).
7. Set the FIFO threshold (FIFOLEV bit in MMCFIFOCTL).
8. Set up DMA (DMA size needs to be greater than or equal to FIFOLEV setting).
9. Use MMCSD1 to send the READ _BLOCK command to the card.
10. Wait for DMA sequence to complete.
11. Use the MMC status register 0 (MMCST0) to check for errors.

10.3.6 MMC/SD Mode Multiple-Block Write Operation Using CPU

To perform a multiple-block write, the same block length needs to be set in both the MMC/SD controller and the card.

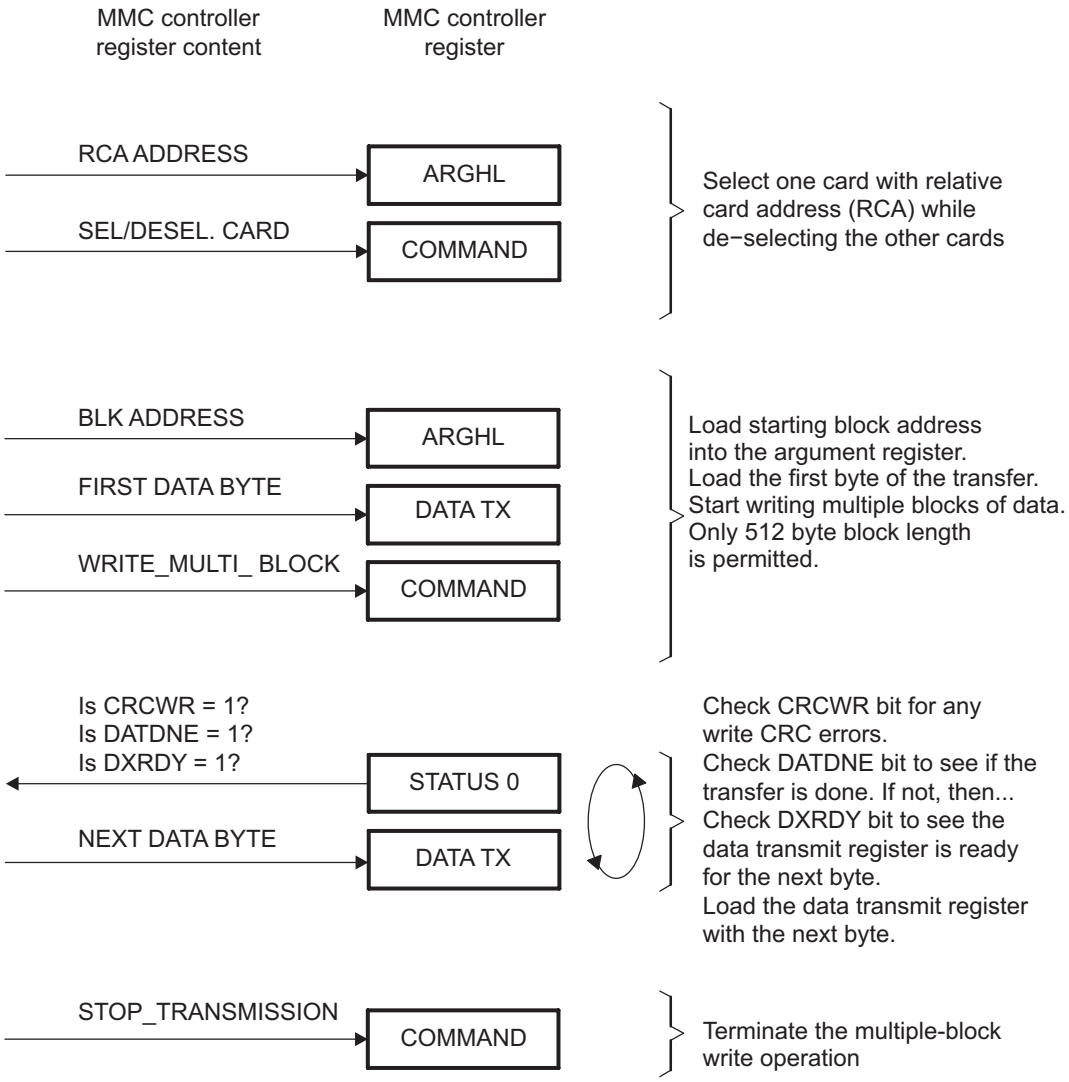
NOTE: The procedure in this section uses a STOP_TRANSMISSION command to end the block transfer. This assumes that the value in the MMC number of blocks counter register (MMCNBLK) is 0. A multiple-block operation terminates itself if you load MMCNBLK with the exact number of blocks you want transferred.

The procedure for this operation is:

1. Write the card's relative address to the MMC argument registers (MMCARG1/MMCARG2).
2. Read card CSD to determine the card's maximum block length.
3. Use MMCSD1 to send the SET_BLOCKLEN command (if the block length is different than the length used in the previous operation). The block length must be a multiple of 512 bytes and less then the maximum block length specified in the CSD.
4. Reset the FIFO by setting the FIFORST bit in MMCFIFOCTL.
5. Bring the FIFO out of reset by clearing the FIFORST bit and set the FIFO direction to transmit (FIFODIR bit in MMCFIFOCTL).
6. Set the access width (ACCWD bits in MMCFIFOCTL).
7. Set the FIFO threshold (FIFOLEV bit in MMCFIFOCTL).
8. Enable the MMC interrupt.
9. Enable DXRDYINT interrupt.
10. Write the first 32 bits of the data block to the MMC data transmit register (MMCDXR).
11. Use MMCSD1 to send the WRITE_MULTI_BLOCK command to the card.
12. Wait for MMC interrupt.
13. Use the MMC status register 1 (MMCST1) to check for errors and to determine the status of the FIFO. If more bytes are to be written and the FIFO is not full, go to step 14. If the all of the data has been written, go to step 15.
14. Write the next *n* bytes (depends on setting of FIFOLEV in MMCFIFOCTL:0 = 16 bytes , 1 = 32 bytes) of the data block to MMCDXR, and go to step 12.
15. Use MMCSD1 to send the STOP_TRANSMISSION command.

The sequence of events in this operation is shown in Figure 10-15.

Figure 10-15. MMC/SD Multiple-Block Write Operation



10.3.7 MMC/SD Mode Multiple-Block Write Operation Using DMA

To perform a multiple-block write, the same block length needs to be set in both the MMC/SD controller and the card. The procedure for this operation is:

1. Write the card's relative address to the MMC argument registers (MMCARG1/MMCARG2).
2. Read card CSD to determine the card's maximum block length.
3. Use MMCSD1 to send the SET_BLOCKLEN command (if the block length is different than the length used in the previous operation). The block length must be a multiple of 512 bytes and less then the maximum block length specified in the CSD.
4. Reset the FIFO by setting the FIFORST bit in MMCFIFOCTL.
5. Bring the FIFO out of reset by clearing the FIFORST bit and set the FIFO direction to transmit (FIFODIR bit in MMCFIFOCTL).
6. Set the FIFO threshold (FIFOLEV bit in MMCFIFOCTL).
7. Set the access width (ACCWD bits in MMCFIFOCTL).
8. Set up DMA (DMA size needs to be greater than or equal to FIFOLEV setting).
9. Use MMCSD1 to send the WRITE_MULTI_BLOCK command to the card.

10. Use MMCSD2 to trigger first DMA transfer to FIFO by setting the DMATRIG bit.
11. Wait for DMA sequence to complete or the DATADNE flag in the MMC status register 0 (MMCST0) is set.
12. Use MMCST0 to check for errors.
13. Use MMCSD1 to send the STOP_TRANSMISSION command.

10.3.8 MMC/SD Mode Multiple-Block Read Operation Using CPU

To perform a multiple-block read, the same block length needs to be set in both the MMC/SD controller and the card.

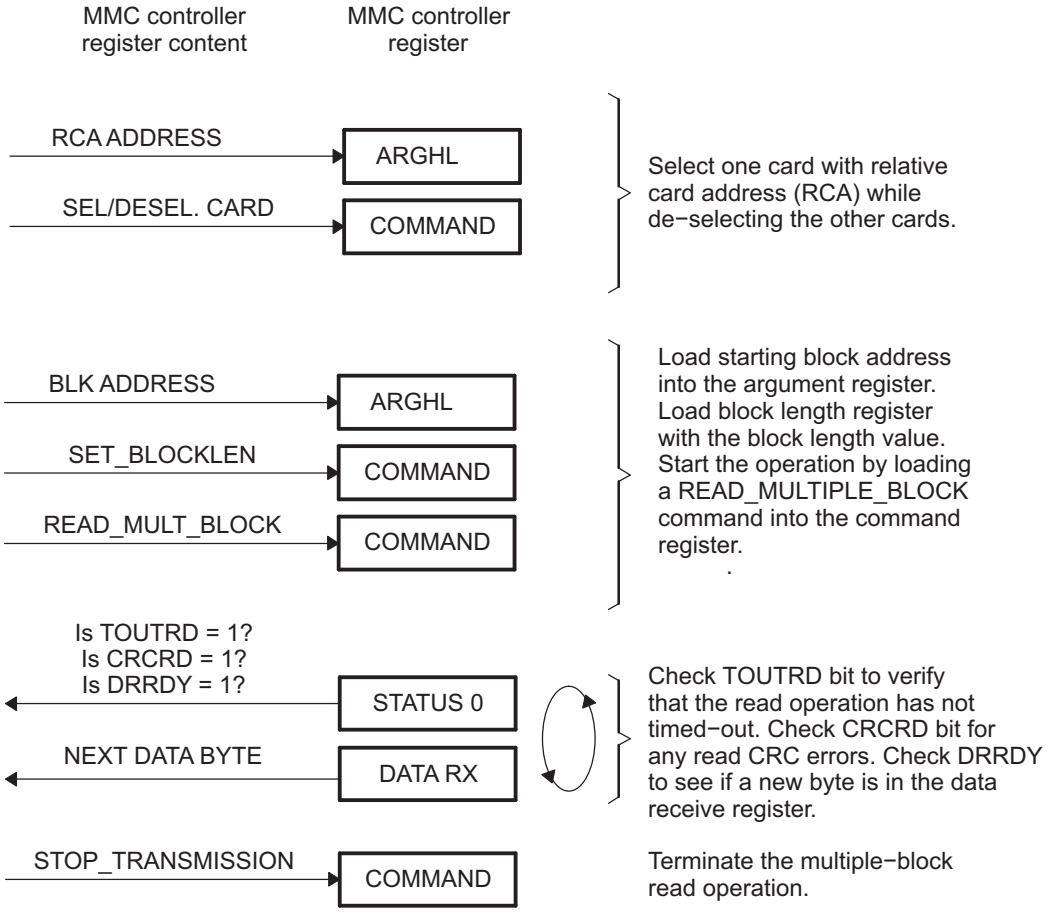
NOTE: The procedure in this section uses a STOP_TRANSMISSION command to end the block transfer. This assumes that the value in the MMC number of blocks counter register (MMCNBLK) is 0. A multiple-block operation terminates itself if you load MMCNBLK with the exact number of blocks you want transferred.

The procedure for this operation is:

1. Write the card's relative address to the MMC argument registers (MMCARG1/MMCARG2).
2. Read card CSD to determine the card's maximum block length.
3. Use MMCSD1 to send the SET_BLOCKLEN command (if the block length is different than the length used in the previous operation). The block length must be a multiple of 512 bytes and less then the maximum block length specified in the CSD.
4. Reset the FIFO by setting the FIFORST bit in MMCFIFOCTL.
5. Bring the FIFO out of reset by clearing the FIFORST bit and set the FIFO direction to receive (FIFODIR bit in MMCFIFOCTL).
6. Set FIFO threshold (FIFOLEV bit in MMCFIFOCTL).
7. Set the access width (ACCWD bits in MMCFIFOCTL).
8. Enable the MMC interrupt.
9. Enable DRRDYINT interrupt.
10. Use MMCSD1 to send the READ_MULT_BLOCKS command.
11. Wait for MMC interrupt.
12. Use the MMC status register 1 (MMCST1) to check for errors and to determine the status of the FIFO. If FIFO is not empty and more bytes are to be read, go to step 13. If the all of the data has been read, go to step 14.
13. Read *n* bytes (depends on setting of FIFOLEV in MMCFIFOCTL:0 = 16 bytes , 1 = 32 bytes) of data from the MMC data receive register (MMCDRR) and go to step 10.
14. Use MMCSD1 to send the STOP_TRANSMISSION command.

The sequence of events in this operation is shown in Figure 10-16.

Figure 10-16. MMC/SD Mode Multiple-Block Read Operation



10.3.9 MMC/SD Mode Multiple-Block Read Operation Using DMA

To perform a multiple-block read, the same block length needs to be set in both the MMC/SD controller and the card. The procedure for this operation is:

1. Write the card's relative address to the MMC argument registers (MMCARG1/MMCARG2).
2. Read card CSD to determine the card's maximum block length.
3. Use MMCSD1 to send the SET_BLOCKLEN command (if the block length is different than the length used in the previous operation). The block length must be a multiple of 512 bytes and less then the maximum block length specified in the CSD.
4. Reset the FIFO by setting the FIFORST bit in MMCFIFOCTL.
5. Bring the FIFO out of reset by clearing the FIFORST bit and set the FIFO direction to receive (FIFODIR bit in MMCFIFOCTL).
6. Set the FIFO threshold (FIFOLEV bit in MMCFIFOCTL).
7. Set the access width (ACCWD bits in MMCFIFOCTL).
8. Set up DMA (DMA size needs to be greater than or equal to FIFOLEV setting).
9. Use MMCSD1 to send the READ_MULTI_BLOCK command to the card.
10. Wait for DMA sequence to complete.
11. Use the MMC status register 0 (MMCST0) to check for errors.
12. Use MMCSD1 to send the STOP_TRANSMISSION command.

10.3.10 SD High Speed Mode

To perform the high-speed mode operation the card need to be placed in high-speed mode. The procedure for this operation is:

1. Follow the normal card identification procedure, since all the high-speed cards are by default initially normal SD cards. Once card is successfully identified, it needs to be switched into high-speed mode.
2. Send CMD16 (SET_BLOCK_LEN) with argument as 8 (8 bytes) to set the block length in the card.
3. Set the block length as 8 bytes and number of blocks as 1 in MMC controller registers MMCBLEN and MMNBLK, respectively.
4. Read the SCR register by sending ACMD51.
5. Parse the 64-bit response received from the card to check whether the card has support of SD spec Ver1.10. The high-speed support is available only cards those are supporting SD VER 1.10.
6. Send CMD16 (SET_BLOCK_LEN) with argument as 64 (64 bytes) to set the block length in the card.
7. Set the block length as 64 bytes and number of blocks as 1 in MMC controller registers MMCBLEN and MMNBLK, respectively.
8. Send CMD6 with Mode 0.
9. Parse the 512-bit response received from the card to check whether the card has high-speed function support. If yes, check that the Maximum current consumption for this function is within the limit which is specified in CSD register (CSD register bits 61:50, response for the CMD9, SEND_CSD).
10. Send CMD6 with Mode 1 to enable the high-speed function.
11. Parse the 512-bit response received from the card to check whether the card has successfully been placed in high-speed mode.
12. Increase the MMC clock rate up to 50 MHz.
13. Follow normal read/write operation.

10.3.11 SDIO Card Function

To support the SDIO card, the following features are available in the MMC/SD controller:

- Read wait operation request.

When in 1-bit mode and the transfer clock (memory clock) is off, this peripheral cannot recognize an SDIO interrupt from SD_DATA1 line. Two options are available to deal with this situation:

1. Do not turn off the memory clock in 1-bit mode. The clock is enabled by the CLKEN bit in the MMC memory clock control register (MMCCLK).
2. If the memory clock needs to be turned off, physically connect a GPIO signal and SD_DATA1, and use the GPIO as an external interrupt input. When the memory clock is enabled, disable the GPIO interrupt and enable the SDIO interrupt. When the memory clock is disabled, enable the GPIO interrupt and disable the SDIO interrupt by software.

10.3.11.1 SDIO Control Register (SDIOCTL)

The SDIO card control register (SDIOCTL) is used to configure the read wait operation using the SD_DATA2 line.

10.3.11.2 SDIO Status Register 0 (SDIOST0)

The SDIO card status register 0 (SDIOST0) is used to check the status of the SD_DATA1 signal, check the status of being in an interrupt period, or check the status of being in a read wait operation.

10.3.11.3 SDIO Interrupt Control Registers (SDIOIEN, SDIOIST)

The SDIO card controller issues an interrupt to the CPU when the read wait operation starts or when an SDIO interrupt is detected on the SD_DATA1 line.

Interrupt flags of each case are checked with the SDIO interrupt status register (SDIOIST). To issue an actual interrupt to CPU, enabling each interrupt in the SDIO interrupt enable register (SDIOIEN) is required.

When both interrupts are enabled, they are both reported to the CPU as a single interrupt (whether one or both occurred). The interrupt(s) that occurred are determined by reading SDIOIST.

10.4 MMC-SD0 CONTROLLER Registers

Table 10-5 lists the memory-mapped registers for the MMC-SD0 CONTROLLER. All register offset addresses not listed in Table 10-5 should be considered as reserved locations and the register contents should not be modified.

Table 10-5. MMC-SD0 CONTROLLER REGISTERS

CPU Word Address	Acronym	Register Name	Section
3A00h	MMCCTL	MMC Control Register	Section 10.4.1
3A04h	MMCCLK	MMC Memory Clock Control Register	Section 10.4.2
3A08h	MMCST0	MMC Status Register 0	Section 10.4.3
3A0Ch	MMCST1	MMC Status Register 1	Section 10.4.4
3A10h	MMCIM	MMC Interrupt Mask Register	Section 10.4.5
3A14h	MMCTOR	MMC Response Time-Out Register	Section 10.4.6
3A18h	MMCTOD	MMC Data Read Time-Out Register	Section 10.4.7
3A1Ch	MMCBLEN	MMC Block Length Register	Section 10.4.8
3A20h	MMCNBLK	MMC Number of Blocks Register	Section 10.4.9
3A24h	MMCNBLC	MMC Number of Blocks Counter Register	Section 10.4.10
3A28h	MMCDRR1	MMC Data Receive Register 1	Section 10.4.11
3A29h	MMCDRR2	MMC Data Receive Register 2	Section 10.4.12
3A2Ch	MMCDXR1	MMC Data Transmit Register 1	Section 10.4.13
3A2Dh	MMCDXR2	MMC Data Transmit Register 2	Section 10.4.14
3A30h	MMCCMD1	MMC Command Register 1	Section 10.4.15
3A31h	MMCCMD2	MMC Command Register 2	Section 10.4.16
3A34h	MMCARG1	MMC Argument Register 1	Section 10.4.17
3A35h	MMCARG2	MMC Argument Register 2	Section 10.4.18
3A38h	MMCRSP0	MMC Response Register 0	Section 10.4.19
3A39h	MMCRSP1	MMC Response Register 1	Section 10.4.20
3A3Ch	MMCRSP2	MMC Response Register 2	Section 10.4.21
3A3Dh	MMCRSP3	MMC Response Register 3	Section 10.4.22
3A40h	MMCRSP4	MMC Response Register 4	Section 10.4.23
3A41h	MMCRSP5	MMC Response Register 5	Section 10.4.24
3A44h	MMCRSP6	MMC Response Register 6	Section 10.4.25
3A45h	MMCRSP7	MMC Response Register 7	Section 10.4.26
3A48h	MMCDRSP	MMC Data Response Register	Section 10.4.27
3A50h	MMCCIDX	MMC Command Index Register	Section 10.4.28
3A64h	SDIOCTL	SDIO Control Register	Section 10.4.29
3A68h	SDIOST0	SDIO Status Register 0	Section 10.4.30
3A6Ch	SDIOIEN	SDIO Interrupt Enable Register	Section 10.4.31
3A70h	SDIOIST	SDIO Interrupt Status Register	Section 10.4.32
3A74h	MMCFIFOCTL	MMC FIFO Control Register	Section 10.4.33

10.4.1 MMCCTL Register (offset = 3A00h) [reset = 0h]

MMCCTL is shown in Figure 10-17 and described in Table 10-6.

The MMC control register (MMCCTL) is used to enable or configure various modes of the MMC controller. Set or clear the DATRST and CMDRST bits at the same time to reset or enable the MMC controller.

Figure 10-17. MMCCTL Register

15	14	13	12	11	10	9	8
Reserved					PERMDX	PERMDR	Reserved
R-0h					R/W-0h	R/W-0h	R-0h

7	6	5	4	3	2	1	0
DATEG		Reserved			WIDTH	CMDRST	DATRST
R/W-0h		R-0h			R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-6. MMCCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved
10	PERMDX	R/W	0h	Endian select enable when writing. 0x0 = Little Endian is selected. 0x1 = Big Endian is selected.
9	PERMDR	R/W	0h	Endian select enable when reading. 0x0 = Little Endian is selected. 0x1 = Big Endian is selected.
8	Reserved	R	0h	Reserved
7-6	DATEG	R/W	0h	DAT3 Edge Detection Select 0x0 = DAT3 edge detection is disabled. 0x1 = DAT3 rising-edge detection is enabled. 0x2 = DAT3 falling-edge detection is enabled. 0x3 = DAT3 rising-edge and falling-edge detections are enabled.
5-3	Reserved	R	0h	Reserved
2	WIDTH	R/W	0h	Data Bus Width (MMC mode) 0x0 = Data bus has 1 bit (only DAT0 is used). 0x1 = Data bus has 4 bits (all DAT0 to 3 are used).
1	CMDRST	R/W	0h	CMD Logic Reset 0x0 = The CMD line portion is enabled. 0x1 = The CMD line portion is disabled and in reset state.
0	DATRST	R/W	0h	DAT Logic Reset 0x0 = The DAT line portion is enabled. 0x1 = The DAT line portion is disabled and in reset state.

10.4.2 MMCCLK Register (offset = 3A04h) [reset = FFh]

MMCCLK is shown in Figure 10-18 and described in Table 10-7.

The MMC memory clock control register (MMCCLK) is used to: (a) Select whether the CLK pin is enabled or disabled (CLKEN bit). (b) Select how much the function clock is divided-down to produce the memory clock (CLKRT bits). When the CLK pin is enabled, the MMC controller drives the memory clock on this pin to control the timing of communications with attached memory cards.

Figure 10-18. MMCCLK Register

15	14	13	12	11	10	9	8
Reserved							CLKEN
R-0h							R/W-0h
7	6	5	4	3	2	1	0
CLKRT							
R/W-FFh							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-7. MMCCLK Register Field Descriptions

Bit	Field	Type	Reset	Description
15-9	Reserved	R	0h	Reserved
8	CLKEN	R/W	0h	CLK Enable 0x0 = CLK pin is disabled and fixed low. 0x1 = The CLK pin is enabled; it shows the memory clock signal.
7-0	CLKRT	R/W	FFh	Clock Pin Rate Clock rate. Use this field to set the divide-down value for the memory clock. The function clock is divided down as follows to produce the memory clock. memory clock frequency = function clock frequency/(2 * (CLKRT + 1)

10.4.3 MMCST0 Register (offset = 3A08h) [reset = 200h]

MMCST0 is shown in Figure 10-19 and described in Table 10-8.

The MMC status register 0 (MMCST0) records specific events or errors. The transition from 0 to 1 on each bit in MMCST0 can cause an interrupt signal to be sent to the CPU. If an interrupt is desired, set the corresponding interrupt enable bit in the MMC interrupt mask register (MMCIM). When a status bit is read (by CPU or emulation) it is cleared. Additionally DRRDY bit and the DXRDY bit are also cleared in response to the functional events. As the command portion and the data portion of the MMC/SD controller are independent, any command such as CMD0 (GO_IDLE_STATE) or CMD12 (STOP_TRANSMISSION) can be sent to the card, even if during block transfer. In this situation, the data portion will detect this and wait, releasing the busy state only when the command sent was R1b (to be specific, command with BSYEXP bit), otherwise it will keep transferring data.

Figure 10-19. MMCST0 Register

15	14	13	12	11	10	9	8
Reserved			TRNDNE	DATED	DRRDY	DXRDY	Reserved
R-0h			R-0h	R-0h	R-0h	R-1h	R-0h
7	6	5	4	3	2	1	0
CRCRS	CRCRD	CRCWR	TOUTRS	TOUTRD	RSPDNE	BSYDNE	DATDNE
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-8. MMCST0 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved
12	TRNDNE	R	0h	Transfer Done. TRNDNE indicates that the last byte of a transfer has been completed. 0x0 = No data transfer is done. 0x1 = Data transfer of specified length is done.
11	DATED	R	0h	DAT3 edge detected. DATED is cleared when read by CPU. 0x0 = No DAT3 edge is detected. 0x1 = DAT3 edge has been detected.
10	DRRDY	R	0h	Data receive ready. DRRDY is cleared to 0 when the DAT logic is reset (DATRST = 1 in MMCCTL), when a command is sent with data receive/transmit clear (DCLR = 1 in MMCCMD), or when data is read from the MMC data receive registers (MMCDRR1 and MMCDRR2). 0x0 = MMCDRR is not ready. 0x1 = MMCDRR is ready. New data has arrived and can be read by the CPU or by the DMA controller.
9	DXRDY	R	1h	Data transmit ready. DXRDY is set to 1 when the DAT logic is reset (DATRST = 1 in MMCCTL), when a command is sent with data receive/transmit clear (DCLR = 1 in MMCCMD), or when data is written to the MMC data transmit register (MMCDXR). 0x0 = MMCDXR is not ready. 0x1 = MMCDXR is ready. The data in MMCDXR has been transmitted; MMCDXR can accept new data from the CPU or from the DMA controller.
8	Reserved	R	0h	Reserved
7	CRCRS	R	0h	Response CRC error. 0x0 = A response CRC error has not been detected. 0x1 = A response CRC error has been detected.

Table 10-8. MMCST0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	CRCRD	R	0h	Read-data CRC error. 0x0 = A read-data CRC error has not been detected. 0x1 = A read-data CRC error has been detected.
5	CRCWR	R	0h	Write-data CRC error. 0x0 = A write-data CRC error has not been detected. 0x1 = A write-data CRC error has been detected.
4	TOUTRS	R	0h	Response time-out event. 0x0 = A response time-out event has not occurred. 0x1 = A time-out event has occurred while the MMC controller was waiting for a response to a command.
3	TOUTRD	R	0h	Read-data time-out event. 0x0 = A read-data time-out event has not occurred. 0x1 = A time-out event has occurred while the MMC controller was waiting for data.
2	RSPDNE	R	0h	Command / Response Done 0x0 = No receiving response done 0x1 = Command has been sent without response or response has been received for the command sent.
1	BSYDNE	R	0h	Busy Done 0x0 = Busy state is not released. 0x1 = Released from busy state or expected busy not detected.
0	DATDNE	R	0h	Data Done. DATDNE occurs at end of a transfer but not until the CRC check and programming has been completed. 0x0 = The data has not been fully transmitted. 0x1 = The data has been fully transmitted.

10.4.4 MMCST1 Register (offset = 3A0Ch) [reset = 2h]

MMCST1 is shown in Figure 10-20 and described in Table 10-9.

The MMC status register 1 (MMCST1) records specific events or errors. There are no interrupts associated with these events or errors.

Figure 10-20. MMCST1 Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved	FIFOFUL	FIFOEMP	DAT3ST	DRFUL	DXEMP	CLKSTP	BUSY
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-1h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-9. MMCST1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	Reserved	R	0h	Reserved
6	FIFOFUL	R	0h	FIFO is full. 0x0 = FIFO is not full. 0x1 = FIFO is full.
5	FIFOEMP	R	0h	FIFO is empty. 0x0 = FIFO is not empty. 0x1 = FIFO is empty.
4	DAT3ST	R	0h	DAT3 Status 0x0 = The status of DAT3 is L level 0x1 = The status of DAT3 is H level
3	DRFUL	R	0h	Data receive register (MMCDRR) is full. 0x0 = A data receive register full condition is not detected. 0x1 = A data receive register full condition is detected.
2	DXEMP	R	0h	Data transmit register (MMCDXR) is empty. 0x0 = A data transmit register empty condition is not detected. The data transmit shift register is not empty. 0x1 = A data transmit register empty condition is detected. The data transmit shift register is empty. No bits are available to be shifted out to the memory card.
1	CLKSTP	R	1h	Clock Stop Status 0x0 = CLK is active. The memory clock signal is being driven on the pin. 0x1 = CLK is held low because of a manual stop (CLKEN = 0 in MMCCLK), receive shift register is full, or transmit shift register is empty.
0	BUSY	R	0h	Busy 0x0 = No busy signal is detected 0x1 = A busy signal is detected (the memory card is busy).

10.4.5 MMCIM Register (offset = 3A10h) [reset = 0h]

MMCIM is shown in Figure 10-21 and described in Table 10-10.

The MMC interrupt mask register (MMCIM) is used to enable (bit = 1) or disable (bit = 0) status interrupts. If an interrupt is enabled, the transition from 0 to 1 of the corresponding interrupt bit in the MMC status register 0 (MMCST0) can cause an interrupt signal to be sent to the CPU.

Figure 10-21. MMCIM Register

15	14	13	12	11	10	9	8
Reserved			ETRNDNE	EDATED	EDRRDY	EDXRDY	Reserved
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h

7	6	5	4	3	2	1	0
ECRCRS	ECRCRD	ECRCWR	ETOUTRS	ETOUTRD	ERSPDNE	EBSYDNE	EDATDNE
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-10. MMCIM Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved
12	ETRNDNE	R/W	0h	Transfer done (TRNDNE) interrupt enable. 0x0 = Transfer done interrupt is disabled. 0x1 = Transfer done interrupt is enabled.
11	EDATED	R/W	0h	DAT3 edge detect (DATED) interrupt enable. 0x0 = DAT3 edge detect interrupt is disabled. 0x1 = DAT3 edge detect interrupt is enabled.
10	EDRRDY	R/W	0h	Data receive register ready (DRRDY) interrupt enable. 0x0 = Data receive register ready interrupt is disabled. 0x1 = Data receive register ready interrupt is enabled.
9	EDXRDY	R/W	0h	Data transmit register (MMCDXR) ready interrupt enable. 0x0 = Data transmit register ready interrupt is disabled. 0x1 = Data transmit register ready interrupt is enabled.
8	Reserved	R	0h	Reserved
7	ECRCRS	R/W	0h	Response CRC error (CRCRS) interrupt enable. 0x0 = Response CRC error interrupt is disabled. 0x1 = Response CRC error interrupt is enabled.
6	ECRCRD	R/W	0h	Read-data CRC error (CRCRD) interrupt enable. 0x0 = Read-data CRC error interrupt is disabled. 0x1 = Read-data CRC error interrupt is enabled.
5	ECRCWR	R/W	0h	Write-data CRC error (CRCWR) interrupt enable. 0x0 = Write-data CRC error interrupt is disabled. 0x1 = Write-data CRC error interrupt is disabled.
4	ETOUTRS	R/W	0h	Response time-out event (TOUTRS) interrupt enable. 0x0 = Response time-out event interrupt is disabled. 0x1 = Response time-out event interrupt is enabled.
3	ETOUTRD	R/W	0h	Read-data time-out event (TOUTRD) interrupt enable. 0x0 = Read-data time-out event interrupt is disabled. 0x1 = Read-data time-out event interrupt is enabled.
2	ERSPDNE	R/W	0h	Command/response done (RSPDNE) interrupt enable. 0x0 = Command/response done interrupt is disabled. 0x1 = Command/response done interrupt is enabled.

Table 10-10. MMCIM Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	EBSYDNE	R/W	0h	Busy done (BSYDNE) interrupt enable. 0x0 = Busy done interrupt is disabled. 0x1 = Busy done interrupt is enabled.
0	EDATDNE	R/W	0h	Data done (DATDNE) interrupt enable. 0x0 = Data done interrupt is disabled. 0x1 = Data done interrupt is enabled.

10.4.6 MMCTOR Register (offset = 3A14h) [reset = 0h]

MMCTOR is shown in Figure 10-22 and described in Table 10-11.

The MMC response time-out register (MMCTOR) defines how long the MMC controller waits for a response from a memory card before recording a time-out condition in the TOUTRS bit of the MMC status register 0 (MMCST0). If the corresponding ETOUTRS bit in the MMC interrupt mask register (MMCIM) is set, an interrupt is generated when the TOUTRS bit is set in MMCST0. If a memory card should require a longer time-out period than MMCTOR can provide, a software time-out mechanism can be implemented.

Figure 10-22. MMCTOR Register

15	14	13	12	11	10	9	8
TOD_23_16							
R/W-0h							
7	6	5	4	3	2	1	0
TOR							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-11. MMCTOR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	TOD_23_16	R/W	0h	Upper 5 bits of Data Time-Out Register (in MMC and SPI Modes). Time-Out Count For Data Read (in MMC mode): 00000: no time-out 00001: 1 CLK clock 00002: 2 CLK clocks ... 1FFFF: 2097151 CLK clocks. Time-Out Count For Data Read (in SPI mode): 00000: no time-out 00001: 1 x 8 CLK clocks 00002: 2 x 8 CLK clocks ... 1FFFFF: 2097151 x 8 CLK clocks.
7-0	TOR	R/W	0h	Time-Out Count For Response. In MMC mode: 0000: no time-out 0001: 1 CLK clock 0010: 2 CLK clocks ... 1111: 255 CLK clocks. In SPI mode: 0000: no time-out 0001: 1 x 8 CLK clock 0010: 2 x 8 CLK clocks ... 1111: 255 x 8 CLK clocks.

10.4.7 MMCTOD Register (offset = 3A18h) [reset = 0h]

MMCTOD is shown in Figure 10-23 and described in Table 10-12.

The MMC data read time-out register (MMCTOD) defines how long the MMC controller waits for the data from a memory card before recording a time-out condition in the TOUTRD bit of the MMC status register 0 (MMCST0). If the corresponding ETOUTRD bit in the MMC interrupt mask register (MMCIM) is set, an interrupt is generated when the TOUTRD bit is set in MMCST0. If a memory card should require a longer time-out period than MMCTOD can provide, a software time-out mechanism can be implemented.

Figure 10-23. MMCTOD Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOD_15_0															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-12. MMCTOD Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TOD_15_0	R/W	0h	Lower 16 bits of Data TimeOut Register. Time-Out Count For Data Read (in MMC mode): 00000: No time-out 00001: 1 CLK clock 00002: 2 CLK clocks ... 1FFFF: 2097151 CLK clocks. Time-Out Count For Data Read (in SPI mode): 00000: No time-out 00001: 1 x 8 CLK clocks 00002: 2 x 8 CLK clocks ... FFFFFF: 2097151 x 8 CLK clocks

10.4.8 MMCBLEN Register (offset = 3A1Ch) [reset = 200h]

MMCBLEN is shown in Figure 10-24 and described in Table 10-13.

The MMC block length register (MMCBLEN) specifies the data block length in bytes. This value must match the block length setting in the memory card.

Figure 10-24. MMCBLEN Register

15	14	13	12	11	10	9	8
Reserved				BLEN			
R-0h				R/W-200h			

7	6	5	4	3	2	1	0
BLEN							
R/W-200h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-13. MMCBLEN Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	Reserved	R	0h	Reserved
11-0	BLEN	R/W	200h	Block length, value 1h to FFFh. This field is used to set the block length, which is the byte count of a data block. The value 0 is prohibited. The BLEN bits value must be the same as the CSD register settings in the MMC/SD card. To be precise, it should match the value of the READ_BL_LEN field for read, or WRITE_BL_LEN field for write.

10.4.9 MMCNBLK Register (offset = 3A20h) [reset = 0h]

MMCNBLK is shown in Figure 10-25 and described in Table 10-14.

The MMC number of blocks register (MMCNBLK) specifies the number of blocks for a multiple-block transfer.

Figure 10-25. MMCNBLK Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-14. MMCNBLK Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	NBLK	R/W	0h	Number of Blocks. Specifies the total number of blocks to be transferred. The value 0x0000 means infinite blocks. 0x0 = Infinite number of blocks. The MMC controller reads/writes blocks of data until a STOP_TRANSMISSION command is written to the MMC command registers (MMCCMD1 and MMCCMD2). 0x1 = Bit value ranges from 1h to FFFFh blocks. The MMC controller reads/writes only n blocks of data, even if the STOP_TRANSMISSION command has not been written to the MMC command registers (MMCCMD1 and MMCCMD2).

10.4.10 MMCNBLC Register (offset = 3A24h) [reset = FFFFh]

MMCNBLC is shown in Figure 10-26 and described in Table 10-15.

The MMC number of blocks counter register (MMCNBLC) is a down-counter for tracking the number of blocks remaining to be transferred during a multiple-block transfer.

Figure 10-26. MMCNBLC Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-15. MMCNBLC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	NBLC	R	FFFFh	MMCNBLC is used in order to test the counter which operates with the value of MMCNBLK in a factory test or a chip evaluation. Reading is possible in order to check the number of blocks to be transferred.

10.4.11 MMCDRR1 Register (offset = 3A28h) [reset = 0h]

MMCDRR1 is shown in Figure 10-27 and described in Table 10-16.

The MMC data receive registers (MMCDRR1 and MMCDRR2) are used for storing the data received from the MMC card. The CPU or the DMA controller can read data from this register. MMCDRR1 and MMCDRR2 expects the data in little-endian format.

Figure 10-27. MMCDRR1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DRR1															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-16. MMCDRR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DRR1	R/W	0h	This register is used for receiving data from the card when a block is read. Register expects little endian formatting.

10.4.12 MMCDRR2 Register (offset = 3A29h) [reset = 0h]

MMCDRR2 is shown in Figure 10-28 and described in Table 10-17.

The MMC data receive registers (MMCDRR1 and MMCDRR2) are used for storing the data received from the MMC card. The CPU or the DMA controller can read data from this register. MMCDRR1 and MMCDRR2 expects the data in little-endian format.

Figure 10-28. MMCDRR2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DRR2															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-17. MMCDRR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DRR2	R/W	0h	This register is used for receiving data from the card when a block is read. Register expects little endian formatting

10.4.13 MMCDXR1 Register (offset = 3A2Ch) [reset = 0h]

MMCDXR1 is shown in Figure 10-29 and described in Table 10-18.

The MMC data transmit registers (MMCDXR1 and MMCDXR2) are used for storing the data to be transmitted from the MMC controller to the memory card. The CPU or the DMA controller can write data to this register to be transmitted. MMCDXR1 and MMCDXR2 data is based on the endian setting in the MMCCTL register.

Figure 10-29. MMCDXR1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DXR1															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-18. MMCDXR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DXR1	R/W	0h	This register is used for transmitting data to the card when a block is written. Register expects little endian formatting

10.4.14 MMCDXR2 Register (offset = 3A2Dh) [reset = 0h]

MMCDXR2 is shown in Figure 10-30 and described in Table 10-19.

The MMC data transmit registers (MMCDXR1 and MMCDXR2) are used for storing the data to be transmitted from the MMC controller to the memory card. The CPU or the DMA controller can write data to this register to be transmitted. MMCDXR1 and MMCDXR2 data is based on the endian setting in the MMCCTL register.

Figure 10-30. MMCDXR2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DXR2															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-19. MMCDXR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DXR2	R/W	0h	This register is used for transmitting data to the card when a block is written. Register expects little endian formatting

10.4.15 MMCCMD1 Register (offset = 3A30h) [reset = 0h]

MMCCMD1 is shown in Figure 10-31 and described in Table 10-20.

Writing to the MMC command registers (MMCCMD1 and MMCCMD2) causes the MMC controller to send the programmed command. Therefore, the MMC argument registers (MMCARG1/MMCARG2) must be loaded properly before a write to MMCCMD. The MMC command registers (MMCCMD1 and 2) specify the type of command to be sent and defines the operation (command, response, additional activity) for the MMC controller. The content of MMCCMD is kept after the transfer to the transmit shift register. When the CPU writes to MMCCMD1 and 2, the MMC controller sends the programmed commands, including any arguments in the MMCARG1/MMCARG2 registers.

Figure 10-31. MMCCMD1 Register

15	14	13	12	11	10	9	8
DCLR	INITCK	WDATX	STRMTP	DTRW	RSPFMT		BSYEXP
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h		R/W-0h
7	6	5	4	3	2	1	0
PPLEN	Reserved	CMD					
R/W-0h	R-0h	R/W-0h					

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-20. MMCCMD1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	DCLR	R/W	0h	Data receive/transmit clear. Use this bit to clear the data receive ready (DRRDY) bit and the data transmit ready (DXRDY) bit in the MMC status register 0 (MMCST0) before a new read or write sequence. This clears any previous status. 0x0 = Do not clear DRRDY and DXRDY bits in MMCST0. 0x1 = Clear DRRDY and DXRDY bits in MMCST0.
14	INITCK	R/W	0h	Initialization clock cycles. 0x0 = Do not insert initialization clock cycles. 0x1 = Insert initialization clock cycles; insert 80 CLK cycles before sending the command specified in the CMD bits. These dummy clock cycles are required for resetting a card after power on.
13	WDATX	R/W	0h	Data transfer indicator. 0x0 = There is no data transfer associated with the command being sent. 0x1 = There is data transfer associated with the command being sent.
12	STRMTP	R/W	0h	Stream transfer enable. 0x0 = If STRMTP = 0, the data transfer is a block transfer. The data transfer stops after the movement of the programmed number of bytes (defined by the programmed block size and the programmed number of blocks). 0x1 = If STRMTP = 1, the data transfer is a stream transfer. Once the data transfer is started, the data transfer does not stop until the MMC controller issues a stop command to the memory card.
11	DTRW	R/W	0h	Data transfer write enable. 0x0 = If WDATX = 0, the data transfer is a read operation. 0x1 = If WDATX = 1, the data transfer is a write operation.
10-9	RSPFMT	R/W	0h	Response format (expected type of response to the command). 0x0 = No response. 0x1 = R1, R4, R5, or R6 response. 48 bits with CRC. 0x2 = R2 response. 136 bits with CRC. 0x3 = R3 response. 48 bits with no CRC.

Table 10-20. MMCCMD1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8	BSYEXP	R/W	0h	Busy Expected 0x0 = No busy signal expected. 0x1 = A busy signal is expected.
7	PPLEN	R/W	0h	Push Pull Enable 0x0 = Push Pull Driver of CMD line is disabled (open drain). 0x1 = Push Pull Driver of CMD line is enabled.
6	Reserved	R	0h	Reserved
5-0	CMD	R/W	0h	Command index. This field contains the command index for the command to be sent to the memory card.

10.4.16 MMCCMD2 Register (offset = 3A31h) [reset = 0h]

MMCCMD2 is shown in Figure 10-32 and described in Table 10-21.

Writing to the MMC command registers (MMCCMD1 and MMCCMD2) causes the MMC controller to send the programmed command. Therefore, the MMC argument registers (MMCARG1/MMCARG2) must be loaded properly before a write to MMCCMD. The MMC command registers (MMCCMD1 and 2) specify the type of command to be sent and defines the operation (command, response, additional activity) for the MMC controller. The content of MMCCMD is kept after the transfer to the transmit shift register. When the CPU writes to MMCCMD1 and 2, the MMC controller sends the programmed commands, including any arguments in the MMCARG1/MMCARG2 registers.

Figure 10-32. MMCCMD2 Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved							DMATRIG
R-0h							R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-21. MMCCMD2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	Reserved	R	0h	Reserved
0	DMATRIG	R/W	0h	Generate a DMA event once to trigger the first DMA transfer for data write operations. Subsequent DMA events are automatically generated. 0x0 = DMA transfer event generation is disabled. 0x1 = Trigger a DMA transfer event for the first data transfer to the FIFO.

10.4.17 MMCARG1 Register (offset = 3A34h) [reset = 0h]

MMCARG1 is shown in Figure 10-33 and described in Table 10-22.

Do not modify the MMC argument registers (MMCARG1 and MMCARG2) while they are being used for an operation. MMCARG1 and MMCARG2 specify the arguments to be sent with the command specified in the MMC command register (MMCCMD). Writing to MMCCMD causes the MMC controller to send a command; therefore, MMCARG1 and MMCARG2 must be configured before writing to MMCCMD. The content of MMCARG1 and MMCARG2 are kept after the transfer to the shift register; however, modification to MMCARG1 and MMCARG2 are not allowed during a sending operation.

Figure 10-33. MMCARG1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARG1															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-22. MMCARG1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	ARG1	R/W	0h	LSB of Argument Register

10.4.18 MMCARG2 Register (offset = 3A35h) [reset = 0h]

MMCARG2 is shown in Figure 10-34 and described in Table 10-23.

Do not modify the MMC argument registers (MMCARG1 and MMCARG2) while they are being used for an operation. MMCARG1 and MMCARG2 specify the arguments to be sent with the command specified in the MMC command register (MMCCMD). Writing to MMCCMD causes the MMC controller to send a command; therefore, MMCARG1 and MMCARG2 must be configured before writing to MMCCMD. The content of MMCARG1 and MMCARG2 are kept after the transfer to the shift register; however, modification to MMCARG1 and MMCARG2 are not allowed during a sending operation.

Figure 10-34. MMCARG2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARG2															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-23. MMCARG2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	ARG2	R/W	0h	MSB of Argument Register. MMCARGHL is used for specifying the arguments to be sent with the command specified by MMCCMD. Writing to MMCCMD cause start event of sending operation, that is, MMCARG has to be set before writing to MMCCMD. The content of this register will be kept after transfer to the shift register. However, modification to MMCARG is not allowed during sending operation

10.4.19 MMCRSP0 Register (offset = 3A38h) [reset = 0h]

MMCRSP0 is shown in Figure 10-35 and described in Table 10-24.

Each command has a preset response type. When the MMC controller receives a response, it is stored in some or all of the MMC response registers (MMCRSP0-MMCRSP7). The response registers are updated as the responses arrive, even if the CPU has not read the previous contents. Each of the MMC response registers holds up to 16 bits. The first byte of the response is a command index byte and is stored in the MMC command index register (MMCCIDX).

Figure 10-35. MMCRSP0 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MMCRSP0															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-24. MMCRSP0 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	MMCRSP0	R/W	0h	MMC Response Register 0

10.4.20 MMCRSP1 Register (offset = 3A39h) [reset = 0h]

MMCRSP1 is shown in Figure 10-36 and described in Table 10-25.

Each command has a preset response type. When the MMC controller receives a response, it is stored in some or all of the MMC response registers (MMCRSP0-MMCRSP7). The response registers are updated as the responses arrive, even if the CPU has not read the previous contents. Each of the MMC response registers holds up to 16 bits. The first byte of the response is a command index byte and is stored in the MMC command index register (MMCCIDX).

Figure 10-36. MMCRSP1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MMCRSP1															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-25. MMCRSP1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	MMCRSP1	R/W	0h	MMC Response Register 1

10.4.21 MMCRSP2 Register (offset = 3A3Ch) [reset = 0h]

MMCRSP2 is shown in Figure 10-37 and described in Table 10-26.

Each command has a preset response type. When the MMC controller receives a response, it is stored in some or all of the MMC response registers (MMCRSP0-MMCRSP7). The response registers are updated as the responses arrive, even if the CPU has not read the previous contents. Each of the MMC response registers holds up to 16 bits. The first byte of the response is a command index byte and is stored in the MMC command index register (MMCCIDX).

Figure 10-37. MMCRSP2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MMCRSP2															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-26. MMCRSP2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	MMCRSP2	R/W	0h	MMC Response Register 2

10.4.22 MMCRSP3 Register (offset = 3A3Dh) [reset = 0h]

MMCRSP3 is shown in Figure 10-38 and described in Table 10-27.

Each command has a preset response type. When the MMC controller receives a response, it is stored in some or all of the MMC response registers (MMCRSP0-MMCRSP7). The response registers are updated as the responses arrive, even if the CPU has not read the previous contents. Each of the MMC response registers holds up to 16 bits. The first byte of the response is a command index byte and is stored in the MMC command index register (MMCCIDX).

Figure 10-38. MMCRSP3 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MMCRSP3															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-27. MMCRSP3 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	MMCRSP3	R/W	0h	MMC Response Register 3

10.4.23 MMCRSP4 Register (offset = 3A40h) [reset = 0h]

MMCRSP4 is shown in Figure 10-39 and described in Table 10-28.

Each command has a preset response type. When the MMC controller receives a response, it is stored in some or all of the MMC response registers (MMCRSP0-MMCRSP7). The response registers are updated as the responses arrive, even if the CPU has not read the previous contents. Each of the MMC response registers holds up to 16 bits. The first byte of the response is a command index byte and is stored in the MMC command index register (MMCCIDX).

Figure 10-39. MMCRSP4 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MMCRSP4															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-28. MMCRSP4 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	MMCRSP4	R/W	0h	Response Register 4

10.4.24 MMCRSP5 Register (offset = 3A41h) [reset = 0h]

MMCRSP5 is shown in Figure 10-40 and described in Table 10-29.

Each command has a preset response type. When the MMC controller receives a response, it is stored in some or all of the MMC response registers (MMCRSP0-MMCRSP7). The response registers are updated as the responses arrive, even if the CPU has not read the previous contents. Each of the MMC response registers holds up to 16 bits. The first byte of the response is a command index byte and is stored in the MMC command index register (MMCCIDX).

Figure 10-40. MMCRSP5 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MMCRSP5															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-29. MMCRSP5 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	MMCRSP5	R/W	0h	Response Register 5

10.4.25 MMCRSP6 Register (offset = 3A44h) [reset = 0h]

MMCRSP6 is shown in Figure 10-41 and described in Table 10-30.

Each command has a preset response type. When the MMC controller receives a response, it is stored in some or all of the MMC response registers (MMCRSP0-MMCRSP7). The response registers are updated as the responses arrive, even if the CPU has not read the previous contents. Each of the MMC response registers holds up to 16 bits. The first byte of the response is a command index byte and is stored in the MMC command index register (MMCCIDX).

Figure 10-41. MMCRSP6 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MMCRSP6															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-30. MMCRSP6 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	MMCRSP6	R/W	0h	Response Register 6

10.4.26 MMCRSP7 Register (offset = 3A45h) [reset = 0h]

MMCRSP7 is shown in Figure 10-42 and described in Table 10-31.

Each command has a preset response type. When the MMC controller receives a response, it is stored in some or all of the MMC response registers (MMCRSP0-MMCRSP7). The response registers are updated as the responses arrive, even if the CPU has not read the previous contents. Each of the MMC response registers holds up to 16 bits. The first byte of the response is a command index byte and is stored in the MMC command index register (MMCCIDX).

Figure 10-42. MMCRSP7 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MMCRSP7															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-31. MMCRSP7 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	MMCRSP7	R/W	0h	Response Register 7

10.4.27 MMCDRSP Register (offset = 3A48h) [reset = 0h]

MMCDRSP is shown in Figure 10-43 and described in Table 10-32.

After the MMC controller sends a data block to a memory card, the CRC status from the memory card is stored in the CRC status register.

Figure 10-43. MMCDRSP Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
DRSP							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-32. MMCDRSP Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	DRSP	R/W	0h	Data Response During a write operation, the CRC status token is stored in DRSP.

10.4.28 MMCCIDX Register (offset = 3A50h) [reset = 0h]

MMCCIDX is shown in Figure 10-44 and described in Table 10-33.

The MMC command index register (MMCCIDX) stores the first byte of a response from a memory card.

Figure 10-44. MMCCIDX Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
STRT	XMIT	CIDX					
R/W-0h	R/W-0h	R/W-0h					

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-33. MMCCIDX Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7	STRT	R/W	0h	Start bit. When the MMC controller receives a response, the start bit is stored in STRT.
6	XMIT	R/W	0h	Transmission bit. When the MMC controller receives a response, the transmission bit is stored in XMIT.
5-0	CIDX	R/W	0h	Command index. When the MMC controller receives a response, the command index is stored in CIDX.

10.4.29 SDIOCTL Register

SDIOCTL is shown in Figure 10-45 and described in Table 10-34.

SDIO Control Register

Figure 10-45. SDIOCTL Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved						RDWTCR	RDWTRQ
R-0h						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-34. SDIOCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	RDWTCR	R/W	0h	Read wait enable for CRC error. If this bit is set, read wait operation automatically starts in case a CRC error is detected during multiple block read access, and while it is not the last block to be transferred. In this case, RDWTRQ (read wait request) is automatically set. To end the read wait operation, write 0 to RDWTRQ. (No need to clear RDWTCR) 0x0 = Read wait is disabled 0x1 = Automatically start read wait on CRC error detection during multiple block read access and not the last block to be transferred. RDWTRQ is automatically set to 1.
0	RDWTRQ	R/W	0h	Read wait request 0x0 = End read wait operation and release DAT[2]. 0x1 = Set a read wait request. If this bit is set, read wait operation starts 2 clocks after the end of the read data block. MMCIF asserts low level on DAT[2] until this bit is cleared.

10.4.30 SDIOST0 Register (offset = 3A68h) [reset = 3h]

SDIOST0 is shown in Figure 10-46 and described in Table 10-35.

SDIO Status Register 0

Figure 10-46. SDIOST0 Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved					RDWTST	INTPRD	DAT1
R-0h					R-0h	R-1h	R-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-35. SDIOST0 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-3	Reserved	R	0h	Reserved
2	RDWTST	R	0h	Read wait status. RDWTST becomes 1 during read wait operation 0x0 = Read wait operation not in progress 0x1 = Read wait operation in progress
1	INTPRD	R	1h	Interrupt period. INTPRD becomes 1 during Interrupt period 0x0 = Interrupt not in progress 0x1 = Interrupt in progress
0	DAT1	R	1h	DAT1 reflects the external state of the SD_DATA1 pin. 0x0 = Logic-low level on the SD_DATA1 pin 0x1 = Logic-high level on the SD_DATA1 pin

10.4.31 SDIOIEN Register (offset = 3A6Ch) [reset = 0h]

SDIOIEN is shown in Figure 10-47 and described in Table 10-36.

SDIO Interrupt Enable Register

Figure 10-47. SDIOIEN Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved						RWSEN	IOINTEN
R-0h						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-36. SDIOIEN Register Field Descriptions

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	RWSEN	R/W	0h	Read wait interrupt enable 0x0 = Read wait interrupt is disabled 0x1 = Read wait interrupt is enabled
0	IOINTEN	R/W	0h	SDIO card interrupt enable 0x0 = SDIO interrupt is disabled 0x1 = SDIO interrupt is enabled

10.4.32 SDIOIST Register (offset = 3A70h) [reset = 0h]

SDIOIST is shown in Figure 10-48 and described in Table 10-37.

SDIO Interrupt Status Register

Figure 10-48. SDIOIST Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved						RWS	IOINT
R-0h						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-37. SDIOIST Register Field Descriptions

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	RWS	R/W	0h	Read wait interrupt status. This bit is set to 1 when read wait operation starts and read wait interrupt is enabled. To clear this bit, write 1 0x0 = No read wait interrupt occurred 0x1 = Read wait interrupt occurred. Read wait operation starts and read wait interrupt is enabled (RWSEN = 1 in SDIOIEN).
0	IOINT	R/W	0h	SDIO card interrupt status. This bit is set to 1 if the SDIO card interrupt is detected and card interrupt is enabled. To clear this bit, write 1 after the interrupt on the DAT[1] line is removed. 0x0 = No SDIO card interrupt occurred 0x1 = SDIO card interrupt occurred. SDIO card interrupt is detected and SDIO card interrupt is enabled (IOINTEN = 1 in SDIOIEN).

10.4.33 MMCFIFOCTL Register (offset = 3A74h) [reset = 0h]

MMCFIFOCTL is shown in Figure 10-49 and described in Table 10-38.

FIFO Control Register

Figure 10-49. MMCFIFOCTL Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved			ACCWD		FIFOLEV	FIFODIR	FIFORST
R-0h			R/W-0h		R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-38. MMCFIFOCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0h	Reserved
4-3	ACCWD	R/W	0h	Access Width (Used by FIFO control to determine full/empty flag) 0x0 = CPU/DMA access width of 4 bytes 0x1 = CPU/DMA access width of 3 bytes 0x2 = CPU/DMA access width of 2 bytes 0x3 = CPU/DMA access width of 1 byte
2	FIFOLEV	R/W	0h	Sets the threshold level which is used to determine when DMA request and FIFO threshold interrupt are triggered 0x0 = DMA Request every 128 bit sent/received 0x1 = DMA Request every 256 bit sent/received
1	FIFODIR	R/W	0h	Determines if the FIFO is being written to or read from 0x0 = Read from FIFO 0x1 = Write to FIFO
0	FIFORST	R/W	0h	Reset the internal state of the FIFO 0x0 = FIFO reset is disabled. 0x1 = FIFO reset is enabled.

10.5 MMC-SD1 CONTROLLER Registers

Table 10-39 lists the memory-mapped registers for the MMC-SD1 CONTROLLER. All register offset addresses not listed in Table 10-39 should be considered as reserved locations and the register contents should not be modified.

Table 10-39. MMC-SD1 CONTROLLER REGISTERS

CPU Word Address	Acronym	Register Name	Section
3B00h	MMCCTL	MMC Control Register	Section 10.5.1
3B04h	MMCCLK	MMC Memory Clock Control Register	Section 10.5.2
3B08h	MMCST0	MMC Status Register 0	Section 10.5.3
3B0Ch	MMCST1	MMC Status Register 1	Section 10.5.4
3B10h	MMCIM	MMC Interrupt Mask Register	Section 10.5.5
3B14h	MMCTOR	MMC Response Time-Out Register	Section 10.5.6
3B18h	MMCTOD	MMC Data Read Time-Out Register	Section 10.5.7
3B1Ch	MMCBLEN	MMC Block Length Register	Section 10.5.8
3B20h	MMCNBLK	MMC Number of Blocks Register	Section 10.5.9
3B24h	MMCNBLC	MMC Number of Blocks Counter Register	Section 10.5.10

Table 10-39. MMC-SD1 CONTROLLER REGISTERS (continued)

CPU Word Address	Acronym	Register Name	Section
3B28h	MMCDRR1	MMC Data Receive Register 1	Section 10.5.11
3B29h	MMCDRR2	MMC Data Receive Register 2	Section 10.5.12
3B2Ch	MMCDXR1	MMC Data Transmit Register 1	Section 10.5.13
3B2Dh	MMCDXR2	MMC Data Transmit Register 2	Section 10.5.14
3B30h	MMCCMD1	MMC Command Register 1	Section 10.5.15
3B31h	MMCCMD2	MMC Command Register 2	Section 10.5.16
3B34h	MMCARG1	MMC Argument Register 1	Section 10.5.17
3B35h	MMCARG2	MMC Argument Register 2	Section 10.5.18
3B38h	MMCRSP0	MMC Response Register 0	Section 10.5.19
3B39h	MMCRSP1	MMC Response Register 1	Section 10.5.20
3B3Ch	MMCRSP2	MMC Response Register 2	Section 10.5.21
3B3Dh	MMCRSP3	MMC Response Register 3	Section 10.5.22
3B40h	MMCRSP4	MMC Response Register 4	Section 10.5.23
3B41h	MMCRSP5	MMC Response Register 5	Section 10.5.24
3B44h	MMCRSP6	MMC Response Register 6	Section 10.5.25
3B45h	MMCRSP7	MMC Response Register 7	Section 10.5.26
3B48h	MMCDRSP	MMC Data Response Register	Section 10.5.27
3B50h	MMCCIDX	MMC Command Index Register	Section 10.5.28
3B64h	SDIOCTL	SDIO Control Register	Section 10.5.29
3B68h	SDIOST0	SDIO Status Register 0	Section 10.5.30
3B6Ch	SDIOIEN	SDIO Interrupt Enable Register	Section 10.5.31
3B70h	SDIOIST	SDIO Interrupt Status Register	Section 10.5.32
3B74h	MMCFIFOCTL	MMC FIFO Control Register	Section 10.5.33

10.5.1 MMCCTL Register (offset = 3B00h) [reset = 0h]

MMCCTL is shown in Figure 10-50 and described in Table 10-40.

The MMC control register (MMCCTL) is used to enable or configure various modes of the MMC controller. Set or clear the DATRST and CMDRST bits at the same time to reset or enable the MMC controller.

Figure 10-50. MMCCTL Register

15	14	13	12	11	10	9	8
Reserved					PERMDX	PERMDR	Reserved
R-0h					R/W-0h	R/W-0h	R-0h

7	6	5	4	3	2	1	0
DATEG		Reserved			WIDTH	CMDRST	DATRST
R/W-0h		R-0h			R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-40. MMCCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved
10	PERMDX	R/W	0h	Endian select enable when writing. 0x0 = Little Endian is selected. 0x1 = Big Endian is selected.
9	PERMDR	R/W	0h	Endian select enable when reading. 0x0 = Little Endian is selected. 0x1 = Big Endian is selected.
8	Reserved	R	0h	Reserved
7-6	DATEG	R/W	0h	DAT3 Edge Detection Select 0x0 = DAT3 edge detection is disabled. 0x1 = DAT3 rising-edge detection is enabled. 0x2 = DAT3 falling-edge detection is enabled. 0x3 = DAT3 rising-edge and falling-edge detections are enabled.
5-3	Reserved	R	0h	Reserved
2	WIDTH	R/W	0h	Data Bus Width (MMC mode) 0x0 = Data bus has 1 bit (only DAT0 is used). 0x1 = Data bus has 4 bits (all DAT0 to 3 are used).
1	CMDRST	R/W	0h	CMD Logic Reset 0x0 = The CMD line portion is enabled. 0x1 = The CMD line portion is disabled and in reset state.
0	DATRST	R/W	0h	DAT Logic Reset 0x0 = The DAT line portion is enabled. 0x1 = The DAT line portion is disabled and in reset state.

10.5.2 MMCCLK Register (offset = 3B04h) [reset = FFh]

MMCCLK is shown in Figure 10-51 and described in Table 10-41.

The MMC memory clock control register (MMCCLK) is used to: (a) Select whether the CLK pin is enabled or disabled (CLKEN bit). (b) Select how much the function clock is divided-down to produce the memory clock (CLKRT bits). When the CLK pin is enabled, the MMC controller drives the memory clock on this pin to control the timing of communications with attached memory cards.

Figure 10-51. MMCCLK Register

15	14	13	12	11	10	9	8
Reserved							CLKEN
R-0h							R/W-0h
7	6	5	4	3	2	1	0
CLKRT							
R/W-FFh							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-41. MMCCLK Register Field Descriptions

Bit	Field	Type	Reset	Description
15-9	Reserved	R	0h	Reserved
8	CLKEN	R/W	0h	CLK Enable 0x0 = CLK pin is disabled and fixed low. 0x1 = The CLK pin is enabled; it shows the memory clock signal.
7-0	CLKRT	R/W	FFh	Clock Pin Rate Clock rate. Use this field to set the divide-down value for the memory clock. The function clock is divided down as follows to produce the memory clock. memory clock frequency = function clock frequency/(2 * (CLKRT + 1)

10.5.3 MMCST0 Register (offset = 3B08h) [reset = 200h]

MMCST0 is shown in Figure 10-52 and described in Table 10-42.

The MMC status register 0 (MMCST0) records specific events or errors. The transition from 0 to 1 on each bit in MMCST0 can cause an interrupt signal to be sent to the CPU. If an interrupt is desired, set the corresponding interrupt enable bit in the MMC interrupt mask register (MMCIM). When a status bit is read (by CPU or emulation) it is cleared. Additionally DRRDY bit and the DXRDY bit are also cleared in response to the functional events. As the command portion and the data portion of the MMC/SD controller are independent, any command such as CMD0 (GO_IDLE_STATE) or CMD12 (STOP_TRANSMISSION) can be sent to the card, even if during block transfer. In this situation, the data portion will detect this and wait, releasing the busy state only when the command sent was R1b (to be specific, command with BSYEXP bit), otherwise it will keep transferring data.

Figure 10-52. MMCST0 Register

15	14	13	12	11	10	9	8
Reserved			TRNDNE	DATED	DRRDY	DXRDY	Reserved
R-0h			R-0h	R-0h	R-0h	R-1h	R-0h
7	6	5	4	3	2	1	0
CRCRS	CRCRD	CRCWR	TOUTRS	TOUTRD	RSPDNE	BSYDNE	DATDNE
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-42. MMCST0 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved
12	TRNDNE	R	0h	Transfer Done. TRNDNE indicates that the last byte of a transfer has been completed. 0x0 = No data transfer is done. 0x1 = Data transfer of specified length is done.
11	DATED	R	0h	DAT3 edge detected. DATED is cleared when read by CPU. 0x0 = No DAT3 edge is detected. 0x1 = DAT3 edge has been detected.
10	DRRDY	R	0h	Data receive ready. DRRDY is cleared to 0 when the DAT logic is reset (DATRST = 1 in MMCCTL), when a command is sent with data receive/transmit clear (DCLR = 1 in MMCCMD), or when data is read from the MMC data receive registers (MMCDRR1 and MMCDRR2). 0x0 = MMCDRR is not ready. 0x1 = MMCDRR is ready. New data has arrived and can be read by the CPU or by the DMA controller.
9	DXRDY	R	1h	Data transmit ready. DXRDY is set to 1 when the DAT logic is reset (DATRST = 1 in MMCCTL), when a command is sent with data receive/transmit clear (DCLR = 1 in MMCCMD), or when data is written to the MMC data transmit register (MMCDXR). 0x0 = MMCDXR is not ready. 0x1 = MMCDXR is ready. The data in MMCDXR has been transmitted; MMCDXR can accept new data from the CPU or from the DMA controller.
8	Reserved	R	0h	Reserved
7	CRCRS	R	0h	Response CRC error. 0x0 = A response CRC error has not been detected. 0x1 = A response CRC error has been detected.

Table 10-42. MMCST0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	CRCRD	R	0h	Read-data CRC error. 0x0 = A read-data CRC error has not been detected. 0x1 = A read-data CRC error has been detected.
5	CRCWR	R	0h	Write-data CRC error. 0x0 = A write-data CRC error has not been detected. 0x1 = A write-data CRC error has been detected.
4	TOUTRS	R	0h	Response time-out event. 0x0 = A response time-out event has not occurred. 0x1 = A time-out event has occurred while the MMC controller was waiting for a response to a command.
3	TOUTRD	R	0h	Read-data time-out event. 0x0 = A read-data time-out event has not occurred. 0x1 = A time-out event has occurred while the MMC controller was waiting for data.
2	RSPDNE	R	0h	Command / Response Done 0x0 = No receiving response done 0x1 = Command has been sent without response or response has been received for the command sent.
1	BSYDNE	R	0h	Busy Done 0x0 = Busy state is not released. 0x1 = Released from busy state or expected busy not detected.
0	DATDNE	R	0h	Data Done. DATDNE occurs at end of a transfer but not until the CRC check and programming has been completed. 0x0 = The data has not been fully transmitted. 0x1 = The data has been fully transmitted.

10.5.4 MMCST1 Register (offset = 3B0Ch) [reset = 2h]

MMCST1 is shown in Figure 10-53 and described in Table 10-43.

The MMC status register 1 (MMCST1) records specific events or errors. There are no interrupts associated with these events or errors.

Figure 10-53. MMCST1 Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved	FIFOFUL	FIFOEMP	DAT3ST	DRFUL	DXEMP	CLKSTP	BUSY
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-1h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-43. MMCST1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	Reserved	R	0h	Reserved
6	FIFOFUL	R	0h	FIFO is full. 0x0 = FIFO is not full. 0x1 = FIFO is full.
5	FIFOEMP	R	0h	FIFO is empty. 0x0 = FIFO is not empty. 0x1 = FIFO is empty.
4	DAT3ST	R	0h	DAT3 Status 0x0 = The status of DAT3 is L level 0x1 = The status of DAT3 is H level
3	DRFUL	R	0h	Data receive register (MMCDRR) is full. 0x0 = A data receive register full condition is not detected. 0x1 = A data receive register full condition is detected.
2	DXEMP	R	0h	Data transmit register (MMCDXR) is empty. 0x0 = A data transmit register empty condition is not detected. The data transmit shift register is not empty. 0x1 = A data transmit register empty condition is detected. The data transmit shift register is empty. No bits are available to be shifted out to the memory card.
1	CLKSTP	R	1h	Clock Stop Status 0x0 = CLK is active. The memory clock signal is being driven on the pin. 0x1 = CLK is held low because of a manual stop (CLKEN = 0 in MMCCLK), receive shift register is full, or transmit shift register is empty.
0	BUSY	R	0h	Busy 0x0 = No busy signal is detected 0x1 = A busy signal is detected (the memory card is busy).

10.5.5 MMCIM Register (offset = 3B10h) [reset = 0h]

MMCIM is shown in Figure 10-54 and described in Table 10-44.

The MMC interrupt mask register (MMCIM) is used to enable (bit = 1) or disable (bit = 0) status interrupts. If an interrupt is enabled, the transition from 0 to 1 of the corresponding interrupt bit in the MMC status register 0 (MMCST0) can cause an interrupt signal to be sent to the CPU.

Figure 10-54. MMCIM Register

15	14	13	12	11	10	9	8
Reserved			ETRNDNE	EDATED	EDRRDY	EDXRDY	Reserved
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h

7	6	5	4	3	2	1	0
ECRCRS	ECRCRD	ECRCWR	ETOUTRS	ETOUTRD	ERSPDNE	EBSYDNE	EDATDNE
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-44. MMCIM Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved
12	ETRNDNE	R/W	0h	Transfer done (TRNDNE) interrupt enable. 0x0 = Transfer done interrupt is disabled. 0x1 = Transfer done interrupt is enabled.
11	EDATED	R/W	0h	DAT3 edge detect (DATED) interrupt enable. 0x0 = DAT3 edge detect interrupt is disabled. 0x1 = DAT3 edge detect interrupt is enabled.
10	EDRRDY	R/W	0h	Data receive register ready (DRRDY) interrupt enable. 0x0 = Data receive register ready interrupt is disabled. 0x1 = Data receive register ready interrupt is enabled.
9	EDXRDY	R/W	0h	Data transmit register (MMCDXR) ready interrupt enable. 0x0 = Data transmit register ready interrupt is disabled. 0x1 = Data transmit register ready interrupt is enabled.
8	Reserved	R	0h	Reserved
7	ECRCRS	R/W	0h	Response CRC error (CRCRS) interrupt enable. 0x0 = Response CRC error interrupt is disabled. 0x1 = Response CRC error interrupt is enabled.
6	ECRCRD	R/W	0h	Read-data CRC error (CRCRD) interrupt enable. 0x0 = Read-data CRC error interrupt is disabled. 0x1 = Read-data CRC error interrupt is enabled.
5	ECRCWR	R/W	0h	Write-data CRC error (CRCWR) interrupt enable. 0x0 = Write-data CRC error interrupt is disabled. 0x1 = Write-data CRC error interrupt is disabled.
4	ETOUTRS	R/W	0h	Response time-out event (TOUTRS) interrupt enable. 0x0 = Response time-out event interrupt is disabled. 0x1 = Response time-out event interrupt is enabled.
3	ETOUTRD	R/W	0h	Read-data time-out event (TOUTRD) interrupt enable. 0x0 = Read-data time-out event interrupt is disabled. 0x1 = Read-data time-out event interrupt is enabled.
2	ERSPDNE	R/W	0h	Command/response done (RSPDNE) interrupt enable. 0x0 = Command/response done interrupt is disabled. 0x1 = Command/response done interrupt is enabled.

Table 10-44. MMCIM Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	EBSYDNE	R/W	0h	Busy done (BSYDNE) interrupt enable. 0x0 = Busy done interrupt is disabled. 0x1 = Busy done interrupt is enabled.
0	EDATDNE	R/W	0h	Data done (DATDNE) interrupt enable. 0x0 = Data done interrupt is disabled. 0x1 = Data done interrupt is enabled.

10.5.6 MMCTOR Register (offset = 3B14h) [reset = 0h]

MMCTOR is shown in Figure 10-55 and described in Table 10-45.

The MMC response time-out register (MMCTOR) defines how long the MMC controller waits for a response from a memory card before recording a time-out condition in the TOUTRS bit of the MMC status register 0 (MMCST0). If the corresponding ETOUTRS bit in the MMC interrupt mask register (MMCIM) is set, an interrupt is generated when the TOUTRS bit is set in MMCST0. If a memory card should require a longer time-out period than MMCTOR can provide, a software time-out mechanism can be implemented.

Figure 10-55. MMCTOR Register

15	14	13	12	11	10	9	8
TOD_23_16							
R/W-0h							
7	6	5	4	3	2	1	0
TOR							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-45. MMCTOR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	TOD_23_16	R/W	0h	Upper 5 bits of Data Time-Out Register (in MMC and SPI Modes). Time-Out Count For Data Read (in MMC mode): 00000: no time-out 00001: 1 CLK clock 00002: 2 CLK clocks ... 1FFFF: 2097151 CLK clocks. Time-Out Count For Data Read (in SPI mode): 00000: no time-out 00001: 1 x 8 CLK clocks 00002: 2 x 8 CLK clocks ... 1FFFFF: 2097151 x 8 CLK clocks.
7-0	TOR	R/W	0h	Time-Out Count For Response. In MMC mode: 0000: no time-out 0001: 1 CLK clock 0010: 2 CLK clocks ... 1111: 255 CLK clocks. In SPI mode: 0000: no time-out 0001: 1 x 8 CLK clock 0010: 2 x 8 CLK clocks ... 1111: 255 x 8 CLK clocks.

10.5.7 MMCTOD Register (offset = 3B18h) [reset = 0h]

MMCTOD is shown in Figure 10-56 and described in Table 10-46.

The MMC data read time-out register (MMCTOD) defines how long the MMC controller waits for the data from a memory card before recording a time-out condition in the TOUTRD bit of the MMC status register 0 (MMCST0). If the corresponding ETOUTRD bit in the MMC interrupt mask register (MMCIM) is set, an interrupt is generated when the TOUTRD bit is set in MMCST0. If a memory card should require a longer time-out period than MMCTOD can provide, a software time-out mechanism can be implemented.

Figure 10-56. MMCTOD Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOD_15_0															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-46. MMCTOD Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TOD_15_0	R/W	0h	Lower 16 bits of Data TimeOut Register. Time-Out Count For Data Read (in MMC mode): 00000: No time-out 00001: 1 CLK clock 00002: 2 CLK clocks ... 1FFFF: 2097151 CLK clocks. Time-Out Count For Data Read (in SPI mode): 00000: No time-out 00001: 1 x 8 CLK clocks 00002: 2 x 8 CLK clocks ... FFFFFF: 2097151 x 8 CLK clocks

10.5.8 MMCBLEN Register (offset = 3B1Ch) [reset = 200h]

MMCBLEN is shown in Figure 10-57 and described in Table 10-47.

The MMC block length register (MMCBLEN) specifies the data block length in bytes. This value must match the block length setting in the memory card.

Figure 10-57. MMCBLEN Register

15	14	13	12	11	10	9	8
Reserved				BLEN			
R-0h				R/W-200h			

7	6	5	4	3	2	1	0
BLEN							
R/W-200h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-47. MMCBLEN Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	Reserved	R	0h	Reserved
11-0	BLEN	R/W	200h	Block length, value 1h to FFFh. This field is used to set the block length, which is the byte count of a data block. The value 0 is prohibited. The BLEN bits value must be the same as the CSD register settings in the MMC/SD card. To be precise, it should match the value of the READ_BL_LEN field for read, or WRITE_BL_LEN field for write.

10.5.9 MMCNBLK Register (offset = 3B20h) [reset = 0h]

MMCNBLK is shown in Figure 10-58 and described in Table 10-48.

The MMC number of blocks register (MMCNBLK) specifies the number of blocks for a multiple-block transfer.

Figure 10-58. MMCNBLK Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-48. MMCNBLK Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	NBLK	R/W	0h	Number of Blocks. Specifies the total number of blocks to be transferred. The value 0x0000 means infinite blocks. 0x0 = Infinite number of blocks. The MMC controller reads/writes blocks of data until a STOP_TRANSMISSION command is written to the MMC command registers (MMCCMD1 and MMCCMD2). 0x1 = Bit value ranges from 1h to FFFFh blocks. The MMC controller reads/writes only n blocks of data, even if the STOP_TRANSMISSION command has not been written to the MMC command registers (MMCCMD1 and MMCCMD2).

10.5.10 MMCNBLC Register (offset = 3B24h) [reset = FFFFh]

MMCNBLC is shown in Figure 10-59 and described in Table 10-49.

The MMC number of blocks counter register (MMCNBLC) is a down-counter for tracking the number of blocks remaining to be transferred during a multiple-block transfer.

Figure 10-59. MMCNBLC Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-49. MMCNBLC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	NBLC	R	FFFFh	MMCNBLC is used in order to test the counter which operates with the value of MMCNBLK in a factory test or a chip evaluation. Reading is possible in order to check the number of blocks to be transferred.

10.5.11 MMCDRR1 Register (offset = 3B28h) [reset = 0h]

MMCDRR1 is shown in Figure 10-60 and described in Table 10-50.

The MMC data receive registers (MMCDRR1 and MMCDRR2) are used for storing the data received from the MMC card. The CPU or the DMA controller can read data from this register. MMCDRR1 and MMCDRR2 expects the data in little-endian format.

Figure 10-60. MMCDRR1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DRR1															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-50. MMCDRR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DRR1	R/W	0h	This register is used for receiving data from the card when a block is read. Register expects little endian formatting.

10.5.12 MMCDRR2 Register (offset = 3B29h) [reset = 0h]

MMCDRR2 is shown in Figure 10-61 and described in Table 10-51.

The MMC data receive registers (MMCDRR1 and MMCDRR2) are used for storing the data received from the MMC card. The CPU or the DMA controller can read data from this register. MMCDRR1 and MMCDRR2 expects the data in little-endian format.

Figure 10-61. MMCDRR2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DRR2															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-51. MMCDRR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DRR2	R/W	0h	This register is used for receiving data from the card when a block is read. Register expects little endian formatting

10.5.13 MMCDXR1 Register (offset = 3B2Ch) [reset = 0h]

MMCDXR1 is shown in Figure 10-62 and described in Table 10-52.

The MMC data transmit registers (MMCDXR1 and MMCDXR2) are used for storing the data to be transmitted from the MMC controller to the memory card. The CPU or the DMA controller can write data to this register to be transmitted. MMCDXR1 and MMCDXR2 data is based on the endian setting in the MMCCTL register.

Figure 10-62. MMCDXR1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DXR1															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-52. MMCDXR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DXR1	R/W	0h	This register is used for transmitting data to the card when a block is written. Register expects little endian formatting

10.5.14 MMCDXR2 Register (offset = 3B2Dh) [reset = 0h]

MMCDXR2 is shown in Figure 10-63 and described in Table 10-53.

The MMC data transmit registers (MMCDXR1 and MMCDXR2) are used for storing the data to be transmitted from the MMC controller to the memory card. The CPU or the DMA controller can write data to this register to be transmitted. MMCDXR1 and MMCDXR2 data is based on the endian setting in the MMCCTL register.

Figure 10-63. MMCDXR2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DXR2															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-53. MMCDXR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DXR2	R/W	0h	This register is used for transmitting data to the card when a block is written. Register expects little endian formatting

10.5.15 MMCCMD1 Register (offset = 3B30h) [reset = 0h]

MMCCMD1 is shown in Figure 10-64 and described in Table 10-54.

Writing to the MMC command registers (MMCCMD1 and MMCCMD2) causes the MMC controller to send the programmed command. Therefore, the MMC argument registers (MMCARG1/MMCARG2) must be loaded properly before a write to MMCCMD. The MMC command registers (MMCCMD1 and 2) specify the type of command to be sent and defines the operation (command, response, additional activity) for the MMC controller. The content of MMCCMD is kept after the transfer to the transmit shift register. When the CPU writes to MMCCMD1 and 2, the MMC controller sends the programmed commands, including any arguments in the MMCARG1/MMCARG2 registers.

Figure 10-64. MMCCMD1 Register

15	14	13	12	11	10	9	8
DCLR	INITCK	WDATX	STRMTP	DTRW	RSPFMT		BSYEXP
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h		R/W-0h
7	6	5	4	3	2	1	0
PPLEN	Reserved	CMD					
R/W-0h	R-0h	R/W-0h					

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-54. MMCCMD1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	DCLR	R/W	0h	Data receive/transmit clear. Use this bit to clear the data receive ready (DRRDY) bit and the data transmit ready (DXRDY) bit in the MMC status register 0 (MMCST0) before a new read or write sequence. This clears any previous status. 0x0 = Do not clear DRRDY and DXRDY bits in MMCST0. 0x1 = Clear DRRDY and DXRDY bits in MMCST0.
14	INITCK	R/W	0h	Initialization clock cycles. 0x0 = Do not insert initialization clock cycles. 0x1 = Insert initialization clock cycles; insert 80 CLK cycles before sending the command specified in the CMD bits. These dummy clock cycles are required for resetting a card after power on.
13	WDATX	R/W	0h	Data transfer indicator. 0x0 = There is no data transfer associated with the command being sent. 0x1 = There is data transfer associated with the command being sent.
12	STRMTP	R/W	0h	Stream transfer enable. 0x0 = If STRMTP = 0, the data transfer is a block transfer. The data transfer stops after the movement of the programmed number of bytes (defined by the programmed block size and the programmed number of blocks). 0x1 = If STRMTP = 1, the data transfer is a stream transfer. Once the data transfer is started, the data transfer does not stop until the MMC controller issues a stop command to the memory card.
11	DTRW	R/W	0h	Data transfer write enable. 0x0 = If WDATX = 0, the data transfer is a read operation. 0x1 = If WDATX = 1, the data transfer is a write operation.
10-9	RSPFMT	R/W	0h	Response format (expected type of response to the command). 0x0 = No response. 0x1 = R1, R4, R5, or R6 response. 48 bits with CRC. 0x2 = R2 response. 136 bits with CRC. 0x3 = R3 response. 48 bits with no CRC.

Table 10-54. MMCCMD1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8	BSYEXP	R/W	0h	Busy Expected 0x0 = No busy signal expected. 0x1 = A busy signal is expected.
7	PPLEN	R/W	0h	Push Pull Enable 0x0 = Push Pull Driver of CMD line is disabled (open drain). 0x1 = Push Pull Driver of CMD line is enabled.
6	Reserved	R	0h	Reserved
5-0	CMD	R/W	0h	Command index. This field contains the command index for the command to be sent to the memory card.

10.5.16 MMCCMD2 Register (offset = 3B31h) [reset = 0h]

MMCCMD2 is shown in Figure 10-65 and described in Table 10-55.

Writing to the MMC command registers (MMCCMD1 and MMCCMD2) causes the MMC controller to send the programmed command. Therefore, the MMC argument registers (MMCARG1/MMCARG2) must be loaded properly before a write to MMCCMD. The MMC command registers (MMCCMD1 and 2) specify the type of command to be sent and defines the operation (command, response, additional activity) for the MMC controller. The content of MMCCMD is kept after the transfer to the transmit shift register. When the CPU writes to MMCCMD1 and 2, the MMC controller sends the programmed commands, including any arguments in the MMCARG1/MMCARG2 registers.

Figure 10-65. MMCCMD2 Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved							DMATRIG
R-0h							R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-55. MMCCMD2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	Reserved	R	0h	Reserved
0	DMATRIG	R/W	0h	Generate a DMA event once to trigger the first DMA transfer for data write operations. Subsequent DMA events are automatically generated. 0x0 = DMA transfer event generation is disabled. 0x1 = Trigger a DMA transfer event for the first data transfer to the FIFO.

10.5.17 MMCARG1 Register (offset = 3B34h) [reset = 0h]

MMCARG1 is shown in Figure 10-66 and described in Table 10-56.

Do not modify the MMC argument registers (MMCARG1 and MMCARG2) while they are being used for an operation. MMCARG1 and MMCARG2 specify the arguments to be sent with the command specified in the MMC command register (MMCCMD). Writing to MMCCMD causes the MMC controller to send a command; therefore, MMCARG1 and MMCARG2 must be configured before writing to MMCCMD. The content of MMCARG1 and MMCARG2 are kept after the transfer to the shift register; however, modification to MMCARG1 and MMCARG2 are not allowed during a sending operation.

Figure 10-66. MMCARG1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARG1															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-56. MMCARG1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	ARG1	R/W	0h	LSB of Argument Register

10.5.18 MMCARG2 Register (offset = 3B35h) [reset = 0h]

MMCARG2 is shown in Figure 10-67 and described in Table 10-57.

Do not modify the MMC argument registers (MMCARG1 and MMCARG2) while they are being used for an operation. MMCARG1 and MMCARG2 specify the arguments to be sent with the command specified in the MMC command register (MMCCMD). Writing to MMCCMD causes the MMC controller to send a command; therefore, MMCARG1 and MMCARG2 must be configured before writing to MMCCMD. The content of MMCARG1 and MMCARG2 are kept after the transfer to the shift register; however, modification to MMCARG1 and MMCARG2 are not allowed during a sending operation.

Figure 10-67. MMCARG2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARG2															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-57. MMCARG2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	ARG2	R/W	0h	MSB of Argument Register. MMCARGHL is used for specifying the arguments to be sent with the command specified by MMCCMD. Writing to MMCCMD cause start event of sending operation, that is, MMCARG has to be set before writing to MMCCMD. The content of this register will be kept after transfer to the shift register. However, modification to MMCARG is not allowed during sending operation

10.5.19 MMCRSP0 Register (offset = 3B38h) [reset = 0h]

MMCRSP0 is shown in Figure 10-68 and described in Table 10-58.

Each command has a preset response type. When the MMC controller receives a response, it is stored in some or all of the MMC response registers (MMCRSP0-MMCRSP7). The response registers are updated as the responses arrive, even if the CPU has not read the previous contents. Each of the MMC response registers holds up to 16 bits. The first byte of the response is a command index byte and is stored in the MMC command index register (MMCCIDX).

Figure 10-68. MMCRSP0 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MMCRSP0															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-58. MMCRSP0 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	MMCRSP0	R/W	0h	MMC Response Register 0

10.5.20 MMCRSP1 Register (offset = 3B39h) [reset = 0h]

MMCRSP1 is shown in Figure 10-69 and described in Table 10-59.

Each command has a preset response type. When the MMC controller receives a response, it is stored in some or all of the MMC response registers (MMCRSP0-MMCRSP7). The response registers are updated as the responses arrive, even if the CPU has not read the previous contents. Each of the MMC response registers holds up to 16 bits. The first byte of the response is a command index byte and is stored in the MMC command index register (MMCCIDX).

Figure 10-69. MMCRSP1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MMCRSP1															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-59. MMCRSP1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	MMCRSP1	R/W	0h	MMC Response Register 1

10.5.21 MMCRSP2 Register (offset = 3B3Ch) [reset = 0h]

MMCRSP2 is shown in Figure 10-70 and described in Table 10-60.

Each command has a preset response type. When the MMC controller receives a response, it is stored in some or all of the MMC response registers (MMCRSP0-MMCRSP7). The response registers are updated as the responses arrive, even if the CPU has not read the previous contents. Each of the MMC response registers holds up to 16 bits. The first byte of the response is a command index byte and is stored in the MMC command index register (MMCCIDX).

Figure 10-70. MMCRSP2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MMCRSP2															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-60. MMCRSP2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	MMCRSP2	R/W	0h	MMC Response Register 2

10.5.22 MMCRSP3 Register (offset = 3B3Dh) [reset = 0h]

MMCRSP3 is shown in Figure 10-71 and described in Table 10-61.

Each command has a preset response type. When the MMC controller receives a response, it is stored in some or all of the MMC response registers (MMCRSP0-MMCRSP7). The response registers are updated as the responses arrive, even if the CPU has not read the previous contents. Each of the MMC response registers holds up to 16 bits. The first byte of the response is a command index byte and is stored in the MMC command index register (MMCCIDX).

Figure 10-71. MMCRSP3 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MMCRSP3															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-61. MMCRSP3 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	MMCRSP3	R/W	0h	MMC Response Register 3

10.5.23 MMCRSP4 Register (offset = 3B40h) [reset = 0h]

MMCRSP4 is shown in Figure 10-72 and described in Table 10-62.

Each command has a preset response type. When the MMC controller receives a response, it is stored in some or all of the MMC response registers (MMCRSP0-MMCRSP7). The response registers are updated as the responses arrive, even if the CPU has not read the previous contents. Each of the MMC response registers holds up to 16 bits. The first byte of the response is a command index byte and is stored in the MMC command index register (MMCCIDX).

Figure 10-72. MMCRSP4 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MMCRSP4															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-62. MMCRSP4 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	MMCRSP4	R/W	0h	Response Register 4

10.5.24 MMCRSP5 Register (offset = 3B41h) [reset = 0h]

MMCRSP5 is shown in Figure 10-73 and described in Table 10-63.

Each command has a preset response type. When the MMC controller receives a response, it is stored in some or all of the MMC response registers (MMCRSP0-MMCRSP7). The response registers are updated as the responses arrive, even if the CPU has not read the previous contents. Each of the MMC response registers holds up to 16 bits. The first byte of the response is a command index byte and is stored in the MMC command index register (MMCCIDX).

Figure 10-73. MMCRSP5 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MMCRSP5															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-63. MMCRSP5 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	MMCRSP5	R/W	0h	Response Register 5

10.5.25 MMCRSP6 Register (offset = 3B44h) [reset = 0h]

MMCRSP6 is shown in Figure 10-74 and described in Table 10-64.

Each command has a preset response type. When the MMC controller receives a response, it is stored in some or all of the MMC response registers (MMCRSP0-MMCRSP7). The response registers are updated as the responses arrive, even if the CPU has not read the previous contents. Each of the MMC response registers holds up to 16 bits. The first byte of the response is a command index byte and is stored in the MMC command index register (MMCCIDX).

Figure 10-74. MMCRSP6 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MMCRSP6															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-64. MMCRSP6 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	MMCRSP6	R/W	0h	Response Register 6

10.5.26 MMCRSP7 Register (offset = 3B45h) [reset = 0h]

MMCRSP7 is shown in Figure 10-75 and described in Table 10-65.

Each command has a preset response type. When the MMC controller receives a response, it is stored in some or all of the MMC response registers (MMCRSP0-MMCRSP7). The response registers are updated as the responses arrive, even if the CPU has not read the previous contents. Each of the MMC response registers holds up to 16 bits. The first byte of the response is a command index byte and is stored in the MMC command index register (MMCCIDX).

Figure 10-75. MMCRSP7 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MMCRSP7															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 10-65. MMCRSP7 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	MMCRSP7	R/W	0h	Response Register 7

10.5.27 MMCDRSP Register (offset = 3B48h) [reset = 0h]

MMCDRSP is shown in Figure 10-76 and described in Table 10-66.

After the MMC controller sends a data block to a memory card, the CRC status from the memory card is stored in the CRC status register.

Figure 10-76. MMCDRSP Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
DRSP							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-66. MMCDRSP Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	DRSP	R/W	0h	Data Response During a write operation, the CRC status token is stored in DRSP.

10.5.28 MMCCIDX Register (offset = 3B50h) [reset = 0h]

MMCCIDX is shown in Figure 10-77 and described in Table 10-67.

The MMC command index register (MMCCIDX) stores the first byte of a response from a memory card.

Figure 10-77. MMCCIDX Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
STRT	XMIT	CIDX					
R/W-0h	R/W-0h	R/W-0h					

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-67. MMCCIDX Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7	STRT	R/W	0h	Start bit. When the MMC controller receives a response, the start bit is stored in STRT.
6	XMIT	R/W	0h	Transmission bit. When the MMC controller receives a response, the transmission bit is stored in XMIT.
5-0	CIDX	R/W	0h	Command index. When the MMC controller receives a response, the command index is stored in CIDX.

10.5.29 SDIOCTL Register

SDIOCTL is shown in Figure 10-78 and described in Table 10-68.

SDIO Control Register

Figure 10-78. SDIOCTL Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved						RDWTCR	RDWTRQ
R-0h						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-68. SDIOCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	RDWTCR	R/W	0h	Read wait enable for CRC error. If this bit is set, read wait operation automatically starts in case a CRC error is detected during multiple block read access, and while it is not the last block to be transferred. In this case, RDWTRQ (read wait request) is automatically set. To end the read wait operation, write 0 to RDWTRQ. (No need to clear RDWTCR) 0x0 = Read wait is disabled 0x1 = Automatically start read wait on CRC error detection during multiple block read access and not the last block to be transferred. RDWTRQ is automatically set to 1.
0	RDWTRQ	R/W	0h	Read wait request 0x0 = End read wait operation and release DAT[2]. 0x1 = Set a read wait request. If this bit is set, read wait operation starts 2 clocks after the end of the read data block. MMCIF asserts low level on DAT[2] until this bit is cleared.

10.5.30 SDIOST0 Register (offset = 3B68h) [reset = 3h]

SDIOST0 is shown in Figure 10-79 and described in Table 10-69.

SDIO Status Register 0

Figure 10-79. SDIOST0 Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved					RDWTST	INTPRD	DAT1
R-0h					R-0h	R-1h	R-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-69. SDIOST0 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-3	Reserved	R	0h	Reserved
2	RDWTST	R	0h	Read wait status. RDWTST becomes 1 during read wait operation 0x0 = Read wait operation not in progress 0x1 = Read wait operation in progress
1	INTPRD	R	1h	Interrupt period. INTPRD becomes 1 during Interrupt period 0x0 = Interrupt not in progress 0x1 = Interrupt in progress
0	DAT1	R	1h	DAT1 reflects the external state of the SD_DATA1 pin. 0x0 = Logic-low level on the SD_DATA1 pin 0x1 = Logic-high level on the SD_DATA1 pin

10.5.31 SDIOIEN Register (offset = 3B6Ch) [reset = 0h]

SDIOIEN is shown in Figure 10-80 and described in Table 10-70.

SDIO Interrupt Enable Register

Figure 10-80. SDIOIEN Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved						RWSEN	IOINTEN
R-0h						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-70. SDIOIEN Register Field Descriptions

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	RWSEN	R/W	0h	Read wait interrupt enable 0x0 = Read wait interrupt is disabled 0x1 = Read wait interrupt is enabled
0	IOINTEN	R/W	0h	SDIO card interrupt enable 0x0 = SDIO interrupt is disabled 0x1 = SDIO interrupt is enabled

10.5.32 SDIOIST Register (offset = 3B70h) [reset = 0h]

SDIOIST is shown in Figure 10-81 and described in Table 10-71.

SDIO Interrupt Status Register

Figure 10-81. SDIOIST Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved						RWS	IOINT
R-0h						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-71. SDIOIST Register Field Descriptions

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	RWS	R/W	0h	Read wait interrupt status. This bit is set to 1 when read wait operation starts and read wait interrupt is enabled. To clear this bit, write 1 0x0 = No read wait interrupt occurred 0x1 = Read wait interrupt occurred. Read wait operation starts and read wait interrupt is enabled (RWSEN = 1 in SDIOIEN).
0	IOINT	R/W	0h	SDIO card interrupt status. This bit is set to 1 if the SDIO card interrupt is detected and card interrupt is enabled. To clear this bit, write 1 after the interrupt on the DAT[1] line is removed. 0x0 = No SDIO card interrupt occurred 0x1 = SDIO card interrupt occurred. SDIO card interrupt is detected and SDIO card interrupt is enabled (IOINTEN = 1 in SDIOIEN).

10.5.33 MMCFIFOCTL Register (offset = 3B74h) [reset = 0h]

MMCFIFOCTL is shown in Figure 10-82 and described in Table 10-72.

FIFO Control Register

Figure 10-82. MMCFIFOCTL Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved			ACCWD		FIFOLEV	FIFODIR	FIFORST
R-0h			R/W-0h		R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 10-72. MMCFIFOCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0h	Reserved
4-3	ACCWD	R/W	0h	Access Width (Used by FIFO control to determine full/empty flag) 0x0 = CPU/DMA access width of 4 bytes 0x1 = CPU/DMA access width of 3 bytes 0x2 = CPU/DMA access width of 2 bytes 0x3 = CPU/DMA access width of 1 byte
2	FIFOLEV	R/W	0h	Sets the threshold level which is used to determine when DMA request and FIFO threshold interrupt are triggered 0x0 = DMA Request every 128 bit sent/received 0x1 = DMA Request every 256 bit sent/received
1	FIFODIR	R/W	0h	Determines if the FIFO is being written to or read from 0x0 = Read from FIFO 0x1 = Write to FIFO
0	FIFORST	R/W	0h	Reset the internal state of the FIFO 0x0 = FIFO reset is disabled. 0x1 = FIFO reset is enabled.

Real-Time Clock (RTC)

Topic **Page**

11.1 Introduction

The following sections describe the features and operation of the real-time clock (RTC) on the digital signal processor (DSP).

11.1.1 Purpose of the Peripheral

The device includes a real time clock (RTC) that provides a time reference to an application executing on the DSP.

The RTC has its own power supply and isolation circuits. The RTC has the capability to wake up the rest of the device through an alarm interrupt, periodic interrupt, or external WAKEUP signal.

To prevent unintentional access to the RTC registers, gate-keeper registers must be programmed before changing the RTC registers (see Section 11.2.3, *Configuring the RTC Registers*).

NOTE: The RTC Core (CV_{DDRTC}) must be powered using an external power source if the RTC is not used.

11.1.2 Features

The real-time clock (RTC) provides the following features:

- 100-year calendar up to year 2099
- Counts milliseconds, seconds, minutes, hours, and date (including day, month, and year with leap year compensation)
- Millisecond time correction
- Binary-coded-decimal (BCD) representation of time, calendar, and alarm
- 24-hour clock mode
- Alarm interrupt for precise time of day, including second, minute, hour, or day
- Periodic interrupt: every millisecond, second, minute, hour, or day
- Single interrupt to the DSP CPU
- 32.768kHz oscillator with frequency calibration
- Bidirectional I/O pin that can be set up as:
  - Input for an external device to wake up the DSP
  - Output to wake up an external device

The current date and time is tracked in a set of counter registers that update once per millisecond. The time is represented in 24-hour mode. For information on how to set the time and date, see Section 11.2.4.

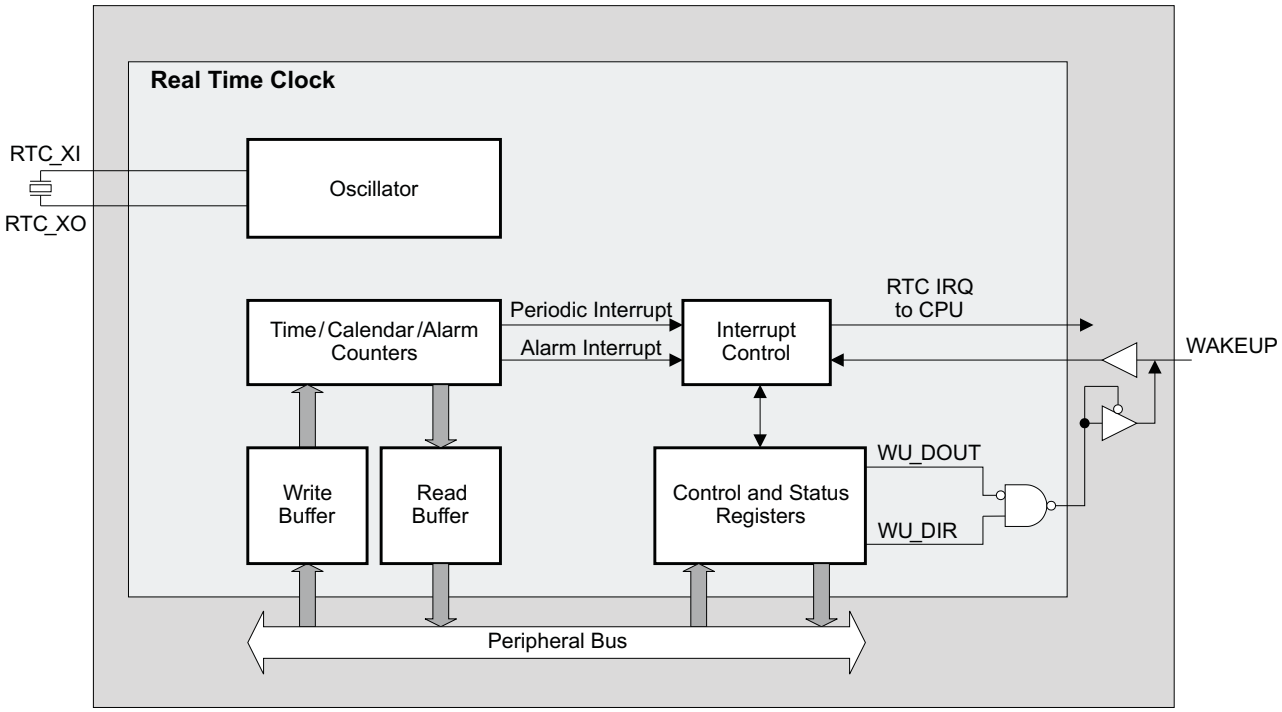
Alarms can be set to interrupt the DSP CPU at a particular time, or at periodic time intervals, such as once per minute or once per day. For information on how to set and use alarms, see Section 11.2.5.

The clock reference for the RTC is an external 32.768kHz crystal (connected between signals RTC_XI and RTC_XO). The RTC also has separate core and I/O power supplies that are isolated from the rest of the DSP.

11.1.3 Functional Block Diagram

Figure 11-1 shows a block diagram of the RTC.

Figure 11-1. Block Diagram



11.2 Peripheral Architecture

This section describes the Real-Time Clock (RTC) peripheral.

NOTE: If the WAKEUP pin is used to wake up the device, DV_{DDRTC} and CV_{DDRTC} must be powered. If the RTC is not used DV_{DDRTC} can be tied to ground (V_{SS}), but CV_{DDRTC} must be powered by an external power source. In addition, the RTC_XI pin must be tied to CV_{DDRTC} and the RTC_XO pin must be tied to V_{SS} or left unconnected. If RTC_XI is tied off, then the RTC registers are inaccessible.

11.2.1 Clock Control

The RTC is driven by an external 32.768 KHz crystal connected between RTC_XI and RTC_XO.

11.2.2 Signal Descriptions

As shown in Figure 11-1, the WAKEUP pin is a bidirectional pin that can be used as an input to wake up the DSP clock domains or it can be used as an open-drain output to wake up an external device. At power-up the WAKEUP pin is configured as an input. This signal can be used to trigger the RTC interrupt to the CPU and to wake-up gated clocks regardless of whether the clocks were gated by the master clock gate or whether they were gated by the DSP's idle instruction.

A high pulse for a minimum of one RTC clock period (30.5 µs) to the WAKEUP pin will trigger the RTC interrupt when:

- The WAKEUP pin is configured as an input (RTCPMGT:WU_DIR = 0)
- The RTC interrupt is enabled (RTCINTEN:RTCINTEN = 1)
- The External Event Interrupt in RTCINTREG is enabled (RTCINTREG:EXTINTEN = 1)

In addition, when WAKEUP is high, the Master Clock Gater for the whole digital core is forced into the un-gated state (clocks on). Note that the interrupt generation is edge sensitive while the clocks on condition is level sensitive. Please see Section 11.2.6 for Interrupt support.

11.2.3 Configuring the RTC Registers

In order to write to the RTC registers, the following steps must be taken:

1. Ensure the RTC is not isolated by writing a value of 1 to the RSCR register.
2. Unlock access to the RTC registers by writing the fixed keys 0x95A4 and 0xF1E0 to the RGKR_LSW and RGKR_MSW gate-keeper registers, respectively.

After writing to the RTC registers:

1. Lock the RTC registers to prevent unintended access by clearing the gate-keeper registers.
2. Enable the RTC isolation by clearing the RSCR register.

The RTC system control and gate-keeper registers are part of a double isolation control for the RTC to protect it from corruption after power-up in case of power glitches. These registers should be cleared at all times except when the RTC registers need to be set.

For more information, see Section 1.5.4, *Static Power Management*.

NOTE: Do not isolate the RTC if using the RTC alarm or periodic interrupts to the CPU. Use the WAKEUP pin to interrupt the CPU when the RTC is isolated. For more information, see the *TMS320C5517 Fixed-Point DSP Silicon Errata* [literature number SPRZ383].

11.2.4 Using the Real-Time Clock Time and Calendar Registers

The current time and date are maintained in the RTC time and calendar registers. Information about how to use these registers is in the sections that follow.

11.2.4.1 Time/Calendar Data Format

The time and calendar data in the RTC is stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. Although most of the time/calendar registers have 4 bits assigned to each BCD digit, some of the register fields are shorter since the range of valid numbers may be limited. For example, only 3 bits are required to represent the first digit (most significant digit) of the “seconds” because only 0 through 5 are required.

The summary of the time/calendar registers is shown in Table 11-1. The alarm registers are interleaved with the time/calendar registers and are not shown in this table. The alarm registers are shown in Table 11-2. A complete description all RTC registers is available in Section 11.3.

Table 11-1. Time/Calendar Registers

Address (Hex)	Name	Function	Decimal Range	BCD Format
1904h	RTCMIL	Milliseconds	0-1023	0000-1023
1908h	RTCSEC	Seconds	0-59	00-59
190Ch	RTCMIN	Minutes	0-59	00-59
1910h	RTCHOUR	Hours (24)	0-23	00-23
1914h	RTCDAY	Days	1-31	01-31
1918h	RTCMONTH	Months (January = 01)	1-12	01-12
191Ch	RTCYEAR	Years	0-99	00-99

- The RTC Milliseconds Register (RTCMIL) stores the milliseconds value of the current time. After the milliseconds count reaches 1023 then the seconds register is updated by one. The reason for the rollover occurring at 1024, rather than 1000, is due to the crystal's oscillation frequency being a power of two, 32.768kHz. 32768 / 1024 = 32 clocks per 'millisecond'. Thus, calling this register a 'milliseconds' register is a bit of a misnomer. The milliseconds digit 3 is 1 bit and the milliseconds digits 2:0 are 4 bits; digits 3:0 are encoded BCD values 0000 (0b 0000b 0000b 0000b) through 1023 (1b 0000b 0010b 0011b).
- The RTC Seconds Register (RTCSEC) stores the seconds value of the current time. The seconds digit 1 is 3 bits and the seconds digit 0 is 4 bits; digits 1 and 0 are encoded BCD values 00 (000b 0000b) through 59 (101b 1001b).
- The RTC Minutes Register (RTCMIN) stores the minutes value of the current time. The minutes digit 1 is 3 bits and the minutes digit 0 is 4 bits; digits 1 and 0 are encoded BCD values 00 (000b 0000b) through 59 (101b 1001b).
- The RTC Hours Register (RTCHOUR) stores the hours value of the current time. The hours digit 1 is 2 bits and the hours digit 0 is 4 bits; digits 1 and 0 are encoded BCD values 01 (00b 0001b) through 23 (10b 0011b).
- The RTC Days Register (RTCDAY) stores the day of the month for the current date. The days digit 1 is 2 bits and the days digit 0 is 4 bits; digits 1 and 0 are encoded BCD 01 (00b 0001b) through 31 (11b 0001b).
- The RTC Months Register (RTCMONTH) stores the month for the current date. The months digit 1 is 1 bit and the months digit 0 is 4 bits; digits 1 and 0 are encoded BCD values 01 (0b 0000b) through 12 (1b 0010b).
- The RTC Years Register (RTCYEAR) stores the year for the current date. The years digit 1 is 4 bits and the years digit 0 is 4 bits; digits 1 and 0 are encoded BCD values 00 (0000b 0000b) through 99 (1001b 1001b).

11.2.4.2 Setting the Time/Calendar Register

The time/calendar registers are set or initialized by writing to the appropriate register bytes. To set date and time, unlock the RTC registers and write all the time and date registers (see Section 11.2.3). When written to, the data is stored to a buffer. Next, set the TIMEUPDT bit in the RTC Update Register (RTCUPDATE). Setting this bit causes the time/calendar values in the buffer to be loaded into the RTC simultaneously. All values should be encoded as BCD values.

11.2.4.3 Reading the Time/Calendar Registers

The time/calendar registers are updated every millisecond as the time changes. To get the most accurate time reading you should start with reading the Millisecond register (RTCMIL) and then the Second register (RTCSEC) followed by the remaining time/calendar register values (RTCMIN, RTCHOUR, RTCDAY, RTCMONTH, and RTCYEAR). Read the RTCMIL again and compare to the previous value. If both values are the same, an RTC update did not occur while the other registers were being read and all the values read represent the current time. If the Milliseconds have changed, this indicates that and RTC update occurred while the registers were being read and the process should be repeated. Results are unpredictable if values are written out of the register's normal range.

11.2.5 Using the Real-Time Clock Time and Calendar Alarms

Alarms can be configured to interrupt the CPU at a specific time, that is, at specific values for the following:

- Milliseconds
- Seconds
- Minutes
- Hours
- Days of the month
- Months
- On specific Years

The time/calendar alarm registers control the setting of alarms. Information about how to use these registers can be found in the following sections. The alarms can be configured to generate an interrupt to the CPU or to wake-up the clocks. The operation of the alarm interrupt is described in Section 11.2.6.4.

11.2.5.1 Time/Calendar Alarm Data Format

The time and calendar alarm data in the RTC is stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. Although most of the time/calendar alarm registers have 4 bits assigned to each BCD digit, some of the register field lengths may differ to accommodate the desired function.

The summary of the time/calendar alarm registers is shown in Table 11-2. The time/calendar registers are interleaved with the alarm registers and are not shown in this table. The time/calendar registers are shown in Table 11-1. A complete description of all RTC registers is available in Section 11.3.

Table 11-2. Time and Calendar Alarm Data

Address (Hex)	Name	Function	Decimal Range	BCD Format
1905h	RTCMILA	Milliseconds alarm	0-1023	0000-1023
1909h	RTCSECA	Seconds alarm	0-59	00-59
190Dh	RTCMINA	Minutes alarm	0-59	00-59
1911h	RTCHOURA	Hours (24) alarm	0-23	00-23
1915h	RTCDAYA	Days alarm	1-31	01-31
1919h	RTCMONTHA	Months (January = 01) alarm	1-12	01-12
191Dh	RTCYEARA	Years alarm	0-99	00-99

The RTC Milliseconds Alarm Register (RTCMILA) stores the milliseconds value of the desired alarm. The milliseconds alarm digit 3 is 1 bit and the milliseconds alarm digits 2:0 are 4; digits 3:0 are encoded BCD values 0000 (0b 0000b 0000b 0000b) through 1023 (1b 0000b 0010b 0011b). Values outside of the decimal range of 0 – 1023 will cause the alarm to never occur.

The RTC Seconds Alarm Register (RTCSECA) stores the seconds value of the desired alarm. The seconds alarm digit 1 is 3 bits and the seconds alarm digit 0 is 4 bits; digits 1 and 0 are encoded BCD values 00 (000b 0000b) through 59 (101b 1001b). Values outside of the decimal range of 0 - 59 will cause the alarm to never occur.

The RTC Minutes Alarm Register (RTCMINA) stores the minute value of the desired alarm. The minutes alarm digit 1 is 3 bits and the minutes alarm digit 0 is 4 bits; digits 1 and 0 are encoded BCD values 00 (000b 0000b) through 59 (101b 1001b). Values outside of the decimal range of 0 - 59 will cause the alarm to never occur.

The RTC Hours Alarm Register (RTCHOURA) stores the hour value of the desired alarm. The hours alarm digit 1 is 2 bits and the hours alarm digit 0 is 4 bits; digits 1 and 0 are encoded BCD values 01 (00b 0001b) through 23 (10b 0011b). Values outside of the decimal range of 1 - 23 will cause the alarm to never occur.

The RTC Days Alarm Register (RTCDAYA) stores the day of the month value of the desired alarm. The days alarm digit 1 is 2 bits and the days alarm digit 0 is 4 bits; digits 1 and 0 are encoded BCD 01 (00b 0001b) through 31 (11b 0001b). Values outside of the decimal range of 1 - 31 will cause the alarm to never occur.

The RTC Months Alarm Register (RTCMONTHA) stores the month of the year value of the desired alarm. The months alarm digit 1 is 1 bit and the months alarm digit 0 is 4 bits; digits 1 and 0 are encoded BCD values 01 (0b 0000b) through 12 (1b 0010b). Values outsides the range 1 - 12 will cause the alarm to never occur.

The RTC Years Alarm Register (RTCYEARA) stores the year value of the desired alarm. The years alarm digit 1 is 4 bits and the years alarm digit 0 is 4 bits; digits 1 and 0 are encoded BCD values 00 (0000b 0000b) through 99 (1001b 1001b). Values outsides the range 0 - 99 will cause the alarm to never occur.

11.2.5.2 Setting and Reading the Time/Calendar Alarm Registers

The time/calendar alarm registers are set or initialized by writing to the appropriate register bytes. To set date and time, unlock the RTC registers and write all the time and date registers (see Section 11.2.3). Then set the ALARMUPDT bit in the RTC Update Register (RTCUPDATE). This will simultaneously copy all the alarm register settings in one RTC cycle.

Time/calendar alarm registers can be read at any time and are not updated by the RTC.

11.2.5.3 Examples of Time/Calendar Alarm Settings

Some examples of various alarm settings are shown in Table 11-3. A complete description of the RTC registers and their functions is provided in Section 11.3.

Table 11-3. Time/Calendar Alarm Settings

Alarm Occurs...	RTCYEARA	RTCMONTHA	RTCDAYA	RTCHOURA	RTCMINA	RTCSECA	RTCMILA
May 7, 2010 @ 3:19:46 AM	10h	5h	7h	3h	19h	46h	0h
Dec 22, 2099 @ 5:50:15 and 300ms PM	99h	12h	22h	17h	50h	15h	300h

11.2.6 Real-Time Clock Interrupt Requests

The RTC provides the ability to interrupt the CPU based on three events: a periodic interrupt, an alarm interrupt, or an external "Wakeup" interrupt. Although three interrupt sources are available, the RTC makes a single interrupt request to the CPU. Specific information about using each of the interrupt types is in the sections that follow.

NOTE: Do not isolate the RTC if using RTC alarm or periodic interrupts to the CPU. Use the WAKEUP pin to interrupt the CPU when the RTC is isolated. For more information, see the *TMS320C5517 Fixed-Point DSP Silicon Errata* [literature number SPRZ383].

11.2.6.1 Interrupt Enable

The RTC has two registers for enabling interrupts. The RTC Interrupt Enable (RTCINTEN) enables the RTC interrupt to the CPU. This bit allows any interrupt that is triggered in the RTC to be sent to the CPU. The second register is the RTC Interrupt Register (RTSINTREG) is used to enable the different interrupt events that can be passed to the CPU. These include the following:

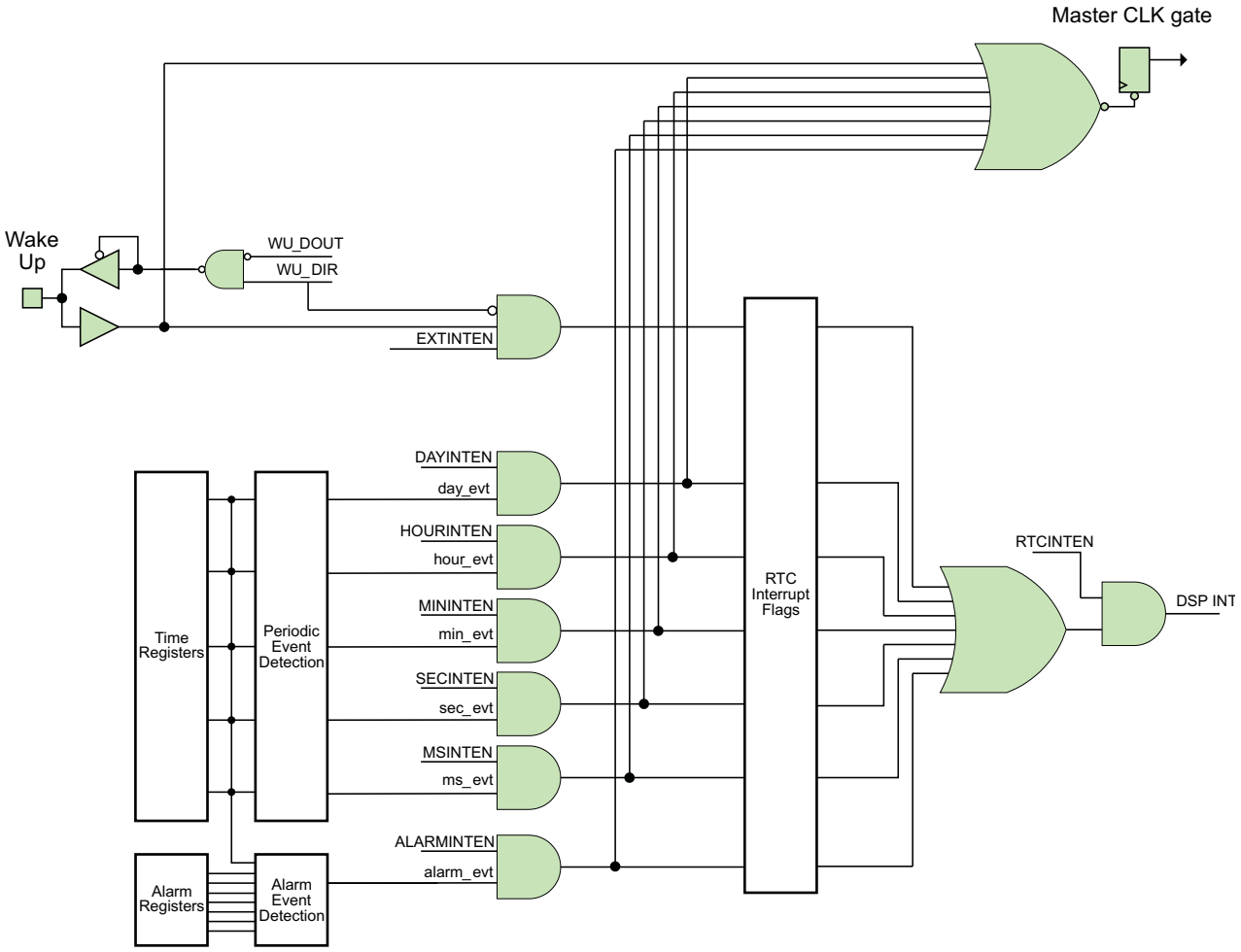
- Alarm Interrupt
- External “wakeup” Interrupt
- Periodic Day Interrupt
- Periodic Hour Interrupt
- Periodic Minute Interrupt
- Periodic Second Interrupt
- Periodic Millisecond Interrupt

NOTE: To use the external wakeup interrupt, you must set the WU_DIR bit in the RTCPMGT register to 0.

When an RTC interrupt is generated, the RTC’s interrupt is directed to two places (see Figure 11-2).

1. System Clock Wakeup Logic: the interrupt will cause the Master Clock Gater to enable the Master Clock from IDLE3 mode.
2. The RTC interrupt can be directed to the CPU if RTCINTEN = 1. The CPU's RTC interrupt must be unmasked for the CPU to respond to the interrupt.
   - If the CPU is idled, the interrupt will cause the CPU to exit idle. If interrupts are globally enabled, the CPU will execute the RTC ISR.
   - If the CPU is not idled and interrupts are globally enabled, the CPU will execute the RTC ISR.

Figure 11-2. RTC Interrupt and Wakeup Logic



11.2.6.2 Interrupt Flag Bits

When the interrupts are enabled in Section 11.2.6.1 and the event occurs, the equivalent flag is set in the RTC Interrupt Flag Register (RTCINTFL). See Section 11.3 for complete details of the RTC registers. The flagged event is cleared when the programmer writes a "1" to the flag bit.

There is also an RTC Lost Power Register (RTCNOPWR). If this flag is set the RTC has lost power and requires a software reset. NOTE: at least 3 RTC clock cycles must elapse after power-up in order to read valid data since the synchronization logic between the CPU and RTC consumes 3 RTC clock cycles.

If the RTC Interrupt enable bit is set and any of the active events occur then an RTC interrupt is sent to the CPU. The RTC interrupt is asserted as long as at least one of the interrupt flag bits are set. When an interrupt occurs from the RTC, the source of the interrupt can be determined by reading the flag bits in RTCINTFL.

11.2.6.3 Periodic Interrupt Request

Periodic Interrupts cause the RTC to make an interrupt request to the CPU periodically. The periodicity can be every millisecond, every second, every minute, hourly, or daily. The periodic interrupt rate is selected using the RTC Interrupts Register (RTCINTREG), see Table 11-4. Writing a 1 to these bits enables the periodic interrupt. Writing a 0 disables the interrupt. Once the interrupt occurs it will remain active until the corresponding flag bit in the RTC Status Register is cleared.

NOTE: The interrupt occurs whenever that particular time value is incremented.

Table 11-4. Periodic Interrupts

RTCINTREG bits	Periodic Interrupt Rate
Bit 0	Every Millisecond
Bit 1	Every Second
Bit 2	Every Minute
Bit 3	Every Hour
Bit 4	Every Day

To use the RTC Periodic interrupt:

- Select the desired interrupt period by enabling the proper interrupt in the RTCINTREG
- Enable the RTC interrupt to the CPU by setting bit 0 of RTCINTEN

When the periodic interrupt occurs, the corresponding interrupt flag will be set in the RTC Interrupt Flag (RTCINTFL) register and the interrupt is sent to the CPU.

11.2.6.4 Alarm Interrupt Request

The RTC alarm interrupt can be used to generate an interrupt to the CPU at a specific time. The alarm interrupt occurs when the alarm time programmed in the RTC alarm registers match the current time. For information about programming an alarm time, see Section 11.2.5.

To use the RTC alarm interrupt:

- Select the desired alarm time by configuring the RTC alarm registers.
- Enable the RTC alarm interrupt by setting bit 15 of the RTCINTREG.
- Enable the RTC interrupt to the CPU by setting bit 0 of RTCINTEN

When the alarm interrupt occurs, the Alarm Flag (bit 15) in the RTCINTFL register will be set and the RTC interrupt is sent to the CPU.

11.2.6.5 WAKEUP Interrupt Request

The external WAKEUP signal or RTC alarm trigger sends a WAKEUP event to the System Clock Wakeup Logic. This asynchronously clears the clock gate which gates the Master Clock and enables the Master Clock. When the DSP wakes up due to an RTC alarm, periodic interrupt, or by the external WAKEUP signal, the DSP latches the RTC interrupt. Because there is only one interrupt line for the RTC, the user must look at the RTC status register to determine which RTC event caused the wake-up.

```
// This example shows how to reactivate the LDOs via WAKEUP input pin
    // Enable RTC Registers as writeable
    asm("    *port(#0x1C27) = #0x0001 "); // un-isolate CVDD_RTC power domain
    asm("    *port(#0x196C) = #0xF1E0 "); // RTC writeable unlock key
    asm("    *port(#0x196D) = #0x95A4 ");
    // Bit 5 = 1 is EXTINTEN which control the external WAKEUP input to generate a DSP IN as
shown in Figure 4-2
    asm("    *port(#0x1924) = #0x0020 ");
    // Disable LDOs and POR and Bandgap to idle
    asm("    *port(#0x1930) = #0x0012 ");
    // Clear RTC writeable key and isolate RTC CVDD_RTC power domain
    asm("    *port(#0x196C) = #0x0000 "); // clear out unlock key to lock
    asm("    *port(#0x196D) = #0x0000 ");
    asm("    *port(#0x1C27) = #0x0000 "); // isolate CVDD_RTC power domain
    // WAKEUP signal high will activate LDOs
```

11.2.7 Reset Considerations

The RTC can be reset by the RTCRESET bit located in the RTC oscillator register (RTCOSC). The RTC can also be reset by an internal POR circuit that monitors VDD_RTC. Neither the RESETN pin nor the DSP's POR can reset the RTC.

11.2.7.1 Software Reset Considerations

The DSP can cause a software reset of the RTC when the RTCRESET bit is set to 1. When this occurs, all RTC registers are reset to the default settings. The RTC will not be reset when the RESETN pin goes low. After a RTC software reset, do not access any RTC register for three 32.768kHz clock cycles after setting the software reset bit.

11.2.7.2 Hardware Reset Considerations

The RTC has a hardware reset that is tied to a POR circuit that monitors the VDD_RTC. The RTC is not reset with the RESETN pin or the DSP's POR.

11.3 Registers

11.3.1 Overview

This section describes the memory-mapped registers for the Real Time Clock (RTC).

Control of the RTC is maintained through a set of I/O memory mapped registers. The first two registers, RTCINTEN and RTCUPDATE, are located in the DSP core power domain, while the remaining registers in Table 5 are located in the RTC power domain.

Writes to registers in the RTC power domain are synchronized to the RTC 32.768-kHz clock and can therefore take many CPU clock cycles to complete. The CPU clock must run at least three-times faster than the RTC, and writes to registers in the RTC domain will not be evident for up to two 32.768kHz clock cycles. If the RTC oscillator is disabled, the RTC system control register is low, or the gate-keeper register does not contain the correct key, no RTC register in the RTC power domain can be written.

11.3.2 RTC Registers

Table 11-5 lists the memory-mapped registers for the RTC. All register offset addresses not listed in Table 11-5 should be considered as reserved locations and the register contents should not be modified.

Table 11-5. RTC REGISTERS

CPU Word Address	Acronym	Register Name	Section
1900h	RTCINTEN	RTC Interrupt Enable Register	Section 11.3.2.1
1901h	RTCUPDATE	RTC Update Register	Section 11.3.2.2
1904h	RTCMIL	Milliseconds Register	Section 11.3.2.3
1905h	RTCMILA	Milliseconds Alarm Register	Section 11.3.2.4
1908h	RTCSEC	Seconds Register	Section 11.3.2.5
1909h	RTCSECA	Seconds Alarm Register	Section 11.3.2.6
190Ch	RTCMIN	Minutes Register	Section 11.3.2.7
190Dh	RTCMINA	Minutes Alarm Register	Section 11.3.2.8
1910h	RTCHOUR	Hours Register	Section 11.3.2.9
1911h	RTCHOURA	Hours Alarm Register	Section 11.3.2.10
1914h	RTCDAY	Days Register	Section 11.3.2.11
1915h	RTCDAYA	Days Alarm Register	Section 11.3.2.12
1918h	RTCMONTH	Months Register	Section 11.3.2.13
1919h	RTCMONTHA	Months Alarm Register	Section 11.3.2.14
191Ch	RTCYEAR	Year Register	Section 11.3.2.15
191Dh	RTCYEARA	Years Alarm Register	Section 11.3.2.16
1920h	RTCINTFL	RTC Interrupt Flag Register	Section 11.3.2.17
1921h	RTCNOPWR	RTC Lost Power Status	Section 11.3.2.18
1924h	RTCINTREG	RTC Interrupt Register	Section 11.3.2.19
1928h	RTCDRIFT	RTC Compensation Register	Section 11.3.2.20
192Ch	RTCOSC	RTC Oscillator Register	Section 11.3.2.21
1930h	RTCPMGT	RTC Power Management Register	Section 11.3.2.22
1960h	RTCSCR1	RTC LSW Scratch Register 1	Section 11.3.2.23
1961h	RTCSCR2	RTC MSW Scratch Register 2	Section 11.3.2.24
1964h	RTCSCR3	RTC LSW Scratch Register 3	Section 11.3.2.25
1965h	RTCSCR4	RTC MSW Scratch Register 4	Section 11.3.2.26
196Ch	RGKR_LSW	RTC Gate-Keeper Register LSW	Section 11.3.2.27
196Dh	RGKR_MSW	RTC Gate-Keeper Register MSW	Section 11.3.2.28

11.3.2.1 RTCINTEN Register (offset = 1900h) [reset = 0h]

RTCINTEN is shown in Figure 11-3 and described in Table 11-6.

RTCINTEN contains the bit field to allow RTC events (alarm, periodic interrupts and external wakeup) to interrupt the CPU. The RTCINTEN bit does not affect the ability of the RTC events to enable clocks and wake up the device from IDLE modes.

Figure 11-3. RTCINTEN Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved							RTCINTEN
R-0h							R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-6. RTCINTEN Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	Reserved	R	0h	Reserved
0	RTCINTEN	R/W	0h	RTC interrupt enable 0x0 = RTC interrupt is disabled 0x1 = RTC interrupt is enabled

11.3.2.2 RTCUPDATE Register (offset = 1901h) [reset = 0h]

RTCUPDATE is shown in Figure 11-4 and described in Table 11-7.

RTC Update Register

Figure 11-4. RTCUPDATE Register

15	14	13	12	11	10	9	8
TIMEUPDT	ALARMUPDT	Reserved					
R/W-0h	R/W-0h	R-0h					

7	6	5	4	3	2	1	0
Reserved							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-7. RTCUPDATE Register Field Descriptions

Bit	Field	Type	Reset	Description
15	TIMEUPDT	R/W	0h	Initiates the time updates 0x0 = RTC time registers updated 0x1 = Initiate update of the time registers
14	ALARMUPDT	R/W	0h	Initiates the alarm updates 0x0 = RTC alarm registers updated 0x1 = Initiate update of the alarm registers
13-0	Reserved	R	0h	Reserved

11.3.2.3 RTCMIL Register (offset = 1904h) [reset = 0h]

RTCMIL is shown in Figure 11-5 and described in Table 11-8.

Milliseconds Register

Figure 11-5. RTCMIL Register

15	14	13	12	11	10	9	8
Reserved			MS3	MS2			
R-0h			R/W-0h	R/W-0h			
7	6	5	4	3	2	1	0
MS1				MS0			
R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-8. RTCMIL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved
12	MS3	R/W	0h	Milliseconds in BCD format in digit 3, value 0 to 1. 0x0 = Digit 3 of the MS BCD is 0 0x1 = Digit 3 of the MS BCD is 1
11-8	MS2	R/W	0h	Milliseconds in BCD format in digit 2, value 0 to 9.
7-4	MS1	R/W	0h	Milliseconds in BCD format in digit 1, value 0 to 9.
3-0	MS0	R/W	0h	Milliseconds in BCD format in digit 0, value 0 to 9.

11.3.2.4 RTCMILA Register (offset = 1905h) [reset = 0h]

RTCMILA is shown in Figure 11-6 and described in Table 11-9.

Milliseconds Alarm Register

Figure 11-6. RTCMILA Register

15	14	13	12	11	10	9	8
Reserved			MSA3	MSA2			
R-0h			R/W-0h	R/W-0h			
7	6	5	4	3	2	1	0
MSA1				MSA0			
R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-9. RTCMILA Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved
12	MSA3	R/W	0h	MS Alarm in digit 3 (BCD), value 0 to 1. 0x0 = Digit 3 of the MS Alarm BCD is 0 0x1 = Digit 3 of the MS Alarm BCD is 1
11-8	MSA2	R/W	0h	MS Alarm in digit 2 (BCD), value 0 to 9.
7-4	MSA1	R/W	0h	MS Alarm in digit 1 (BCD), value 0 to 9.
3-0	MSA0	R/W	0h	MS Alarm in digit 0 (BCD), value 0 to 9.

11.3.2.5 RTCSEC Register (offset = 1908h) [reset = 0h]

RTCSEC is shown in Figure 11-7 and described in Table 11-10.

Seconds Register

Figure 11-7. RTCSEC Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved	SEC1			SEC0			
R-0h	R/W-0h			R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-10. RTCSEC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	Reserved	R	0h	Reserved
6-4	SEC1	R/W	0h	Seconds in BCD format in digit 1, value 0 to 5.
3-0	SEC0	R/W	0h	Seconds in BCD format in digit 0, value 0 to 9.

11.3.2.6 RTCSECA Register (offset = 1909h) [reset = 0h]

RTCSECA is shown in Figure 11-8 and described in Table 11-11.

Seconds Alarm Register

Figure 11-8. RTCSECA Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved	SECA1			SECA0			
R-0h	R/W-0h			R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-11. RTCSECA Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	Reserved	R	0h	Reserved
6-4	SECA1	R/W	0h	Seconds Alarm in BCD format in digit 1, value 0 to 5.
3-0	SECA0	R/W	0h	Seconds Alarm in BCD format in digit 0, value 0 to 9.

11.3.2.7 RTCMIN Register (offset = 190Ch) [reset = 0h]

RTCMIN is shown in Figure 11-9 and described in Table 11-12.

Minutes Register

Figure 11-9. RTCMIN Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved	MIN1			MIN0			
R-0h	R/W-0h			R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-12. RTCMIN Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	Reserved	R	0h	Reserved
6-4	MIN1	R/W	0h	Minutes in BCD format in digit 1, value 0 to 5.
3-0	MIN0	R/W	0h	Minutes in BCD format in digit 0, value 0 to 9.

11.3.2.8 RTCMINA Register (offset = 190Dh) [reset = 0h]

RTCMINA is shown in Figure 11-10 and described in Table 11-13.

Minutes Alarm Register

Figure 11-10. RTCMINA Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved	MINA1			MINA0			
R-0h	R/W-0h			R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-13. RTCMINA Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	Reserved	R	0h	Reserved
6-4	MINA1	R/W	0h	Minutes Alarm in BCD format in digit 1, value 0 to 5.
3-0	MINA0	R/W	0h	Minutes Alarm in BCD format in digit 0, value 0 to 9.

11.3.2.9 RTCHOUR Register (offset = 1910h) [reset = 0h]

RTCHOUR is shown in Figure 11-11 and described in Table 11-14.

Hours Register

Figure 11-11. RTCHOUR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		HOUR1		HOUR0			
R-0h		R/W-0h		R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-14. RTCHOUR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	Reserved
5-4	HOUR1	R/W	0h	Hours in BCD format in digit 1, value 0 to 2.
3-0	HOUR0	R/W	0h	Hours in BCD format in digit 0, value 0 to 9.

11.3.2.10 RTCHOURA Register (offset = 1911h) [reset = 0h]

RTCHOURA is shown in Figure 11-12 and described in Table 11-15.

Hours Alarm Register

Figure 11-12. RTCHOURA Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		HOURA1		HOURA0			
R-0h		R/W-0h		R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-15. RTCHOURA Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	Reserved
5-4	HOURA1	R/W	0h	Hours Alarm. BCD format in digit 1, value 0 to 2.
3-0	HOURA0	R/W	0h	Hours Alarm. BCD format in digit 0, value 0 to 9.

11.3.2.11 RTCDAY Register (offset = 1914h) [reset = 1h]

RTCDAY is shown in Figure 11-13 and described in Table 11-16.

Days Register

Figure 11-13. RTCDAY Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		DAY1		DAY0			
R-0h		R/W-0h		R/W-1h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-16. RTCDAY Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	Reserved
5-4	DAY1	R/W	0h	Day in BCD format in digit 1, value 0 to 3.
3-0	DAY0	R/W	1h	Day in BCD format in digit 0, value 0 to 9.

11.3.2.12 RTCDAYA Register (offset = 1915h) [reset = 0h]

RTCDAYA is shown in Figure 11-14 and described in Table 11-17.

Days Alarm Register

Figure 11-14. RTCDAYA Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		DAYA1		DAYA0			
R-0h		R/W-0h		R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-17. RTCDAYA Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	Reserved
5-4	DAYA1	R/W	0h	Day Alarm. BCD format in digit 1, value 0 to 3.
3-0	DAYA0	R/W	0h	Day Alarm. BCD format in digit 0, value 0 to 9.

11.3.2.13 RTCMONTH Register (offset = 1918h) [reset = 1h]

RTCMONTH is shown in Figure 11-15 and described in Table 11-18.

Months Register

Figure 11-15. RTCMONTH Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved			MONTH1	MONTH0			
R-0h			R/W-0h	R/W-1h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-18. RTCMONTH Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0h	Reserved
4	MONTH1	R/W	0h	Month in BCD format in digit 1, value 0 to 1.
3-0	MONTH0	R/W	1h	Month in BCD format in digit 0, value 0 to 9.

11.3.2.14 RTCMONTHA Register (offset = 1919h) [reset = 0h]

RTCMONTHA is shown in Figure 11-16 and described in Table 11-19.

Months Alarm Register

Figure 11-16. RTCMONTHA Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved			MONTHA1	MONTHA0			
R-0h			R/W-0h	R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-19. RTCMONTHA Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0h	Reserved
4	MONTHA1	R/W	0h	Month Alarm. BCD format in digit 1, value 0 to 1.
3-0	MONTHA0	R/W	0h	Month Alarm. BCD format in digit 0, value 0 to 9.

11.3.2.15 RTCYEAR Register (offset = 191Ch) [reset = 0h]

RTCYEAR is shown in Figure 11-17 and described in Table 11-20.

Year Register

Figure 11-17. RTCYEAR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
YEAR1				YEAR0			
R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-20. RTCYEAR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-4	YEAR1	R/W	0h	Year in BCD format in digit 1 (20XX), value 0 to 9.
3-0	YEAR0	R/W	0h	Year in BCD format in digit 0 (20XX), value 0 to 9.

11.3.2.16 RTCYEARA Register (offset = 191Dh) [reset = 0h]

RTCYEARA is shown in Figure 11-18 and described in Table 11-21.

Years Alarm Register

Figure 11-18. RTCYEARA Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
YEARA1				YEARA0			
R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-21. RTCYEARA Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-4	YEARA1	R/W	0h	Year Alarm. BCD format in digit 1 (20XX), value 0 to 9.
3-0	YEARA0	R/W	0h	Year Alarm. BCD format in digit 0 (20XX), value 0 to 9.

11.3.2.17 RTCINTFL Register (offset = 1920h) [reset = 0h]

RTCINTFL is shown in Figure 11-19 and described in Table 11-22.

RTC Interrupt Flag Register

Figure 11-19. RTCINTFL Register

15	14	13	12	11	10	9	8
ALARMFL	Reserved						
R/W-0h	R-0h						

7	6	5	4	3	2	1	0
Reserved		EXTFL	DAYFL	HOURFL	MINFL	SECFL	MSFL
R-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-22. RTCINTFL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	ALARMFL	R/W	0h	Indicates that an alarm interrupt has been generated 0x0 = Alarm interrupt did not occur 0x1 = Alarm interrupt occurred (write 1 to clear)
14-6	Reserved	R	0h	Reserved
5	EXTFL	R/W	0h	External event has occurred 0x0 = External event interrupt has not occurred 0x1 = External event interrupt occurred (write 1 to clear)
4	DAYFL	R/W	0h	Day event has occurred 0x0 = Periodic day event has not occurred 0x1 = Periodic Day event occurred (write 1 to clear)
3	HOURFL	R/W	0h	Hour event has occurred 0x0 = Periodic Hour event has not occurred 0x1 = Periodic Hour event occurred (write 1 to clear)
2	MINFL	R/W	0h	Minute event has occurred 0x0 = Periodic Minute event has not occurred 0x1 = Periodic Minute event occurred (write 1 to clear)
1	SECFL	R/W	0h	Second event occurred 0x0 = Periodic Second event has not occurred 0x1 = Periodic Second event occurred (write 1 to clear)
0	MSFL	R/W	0h	Millisecond event occurred 0x0 = Periodic Millisecond event has not occurred 0x1 = Periodic Millisecond event occurred (write 1 to clear)

11.3.2.18 RTCNOPWR Register (offset = 1921h) [reset = 1h]

RTCNOPWR is shown in Figure 11-20 and described in Table 11-23.

RTC Lost Power Status

Figure 11-20. RTCNOPWR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved							PWRUP
R-0h							R-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-23. RTCNOPWR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	Reserved	R	0h	Reserved
0	PWRUP	R	1h	Power lost status bit 0x0 = RTC has not lost power since software reset 0x1 = RTC has lost power and requires a software reset and initialization of the time registers to the current time and date. PWRUP is cleared by a read of RTCINTFL or RTCNOPWR. Therefore, read RTCNOPWR before reading RTCINTFL to obtain the correct PWRUP value.

11.3.2.19 RTCINTREG Register (offset = 1924h) [reset = 0h]

RTCINTREG is shown in Figure 11-21 and described in Table 11-24.

RTC Interrupt Register

Figure 11-21. RTCINTREG Register

15	14	13	12	11	10	9	8
ALARMINTEN	Reserved						
R/W-0h	R-0h						

7	6	5	4	3	2	1	0
Reserved		EXTINTEN	DAYINTEN	HOURINTEN	MININTEN	SECINTEN	MSINTEN
R-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-24. RTCINTREG Register Field Descriptions

Bit	Field	Type	Reset	Description
15	ALARMINTEN	R/W	0h	Alarm interrupt enable 0x0 = Alarm interrupt not enabled. 0x1 = Alarm interrupt enabled (write 1 to clear)
14-6	Reserved	R	0h	Reserved
5	EXTINTEN	R/W	0h	External event interrupt enable 0x0 = External event interrupt not enabled 0x1 = External event interrupt enabled
4	DAYINTEN	R/W	0h	Day event interrupt enable 0x0 = Periodic Day event not enabled 0x1 = Periodic Day event enabled
3	HOURINTEN	R/W	0h	hour event interrupt enable 0x0 = Periodic Hour event not enabled 0x1 = Periodic Hour event enabled
2	MININTEN	R/W	0h	Minute Event interrupt enable 0x0 = Periodic Minute event not enabled 0x1 = Periodic Minute event enabled
1	SECINTEN	R/W	0h	Second Event interrupt enable 0x0 = Periodic Second event not enabled 0x1 = Periodic Second event enabled
0	MSINTEN	R/W	0h	Millisecond event interrupt enable 0x0 = Periodic Millisecond event not enabled 0x1 = Periodic Millisecond event enabled

11.3.2.20 RTCDRIFT Register (offset = 1928h) [reset = 0h]

RTCDRIFT is shown in Figure 11-22 and described in Table 11-25.

Every hour on the hour, a positive or negative number of milliseconds is added to the milliseconds register to compensate for inaccuracy in the 32.768-kHz crystal based on the value of COMP[3:0]. If this value is 0 then no compensation will be applied.

Note: Any positive compensation value must not be a multiple of 10.

Figure 11-22. RTCDRIFT Register

15	14	13	12	11	10	9	8
DRIFT	Reserved		COMP3	COMP2			
R/W-0h	R-0h		R/W-0h	R/W-0h			
7	6	5	4	3	2	1	0
COMP1				COMP0			
R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-25. RTCDRIFT Register Field Descriptions

Bit	Field	Type	Reset	Description
15	DRIFT	R/W	0h	Positive or Negative Compensation 0x0 = Negative compensation 0x1 = Positive compensation
14-13	Reserved	R	0h	Reserved
12	COMP3	R/W	0h	Digit 3 of compensation in BCD format, value 0 to 1. 0x0 = BCD digit is 0 0x1 = BCD digit is 1
11-8	COMP2	R/W	0h	Digit 2 of compensation register in BCD format, value 0 to 9.
7-4	COMP1	R/W	0h	Digit 1 of compensation register in BCD format, value 0 to 9.
3-0	COMP0	R/W	0h	Digit 0 of compensation register in BCD format, value 0 to 9.

11.3.2.21 RTCOSC Register (offset = 192Ch) [reset = Bh]

RTCOSC is shown in Figure 11-23 and described in Table 11-26.

RTC Oscillator Register

Figure 11-23. RTCOSC Register

15	14	13	12	11	10	9	8
RTCRESET	Reserved						
R/W-0h	R-0h						

7	6	5	4	3	2	1	0
Reserved			OSC32KPWD	OSCRES			
R-0h			R/W-0h	R/W-Bh			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-26. RTCOSC Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RTCRESET	R/W	0h	RTC software reset. The RTC only resets when this bit is set. The RTC is not reset when RESETn goes low. Once set, this bit is cleared by the RTC. Do not access any RTC register for three 32 kHz clock cycles after setting this bit. 0x0 = RTC not reset 0x1 = RTC reset
14-5	Reserved	R	0h	Reserved
4	OSC32KPWD	R/W	0h	Control of 32 kHz Oscillator powerdown in functional mode. 0x0 = RTC 32 kHz oscillator enabled 0x1 = RTC 32 kHz oscillator disabled. External crystal connected through two internal buffers.
3-0	OSCRES	R/W	Bh	Value of the oscillator resistance, value 0 to Fh. The default (reset state) is 1011b and this gives faster startup but higher power. Once the oscillator is running it can be changed to 1000b for lower power consumption.

11.3.2.22 RTCPMGT Register

RTCPMGT is shown in Figure 11-24 and described in Table 11-27.

RTC Power Management Register

Figure 11-24. RTCPMGT Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved			WU_DOUT	WU_DIR	BG_PD	LDO_PD	RTCCLKOUTEN
R-0h			R/W-1h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-27. RTCPMGT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0h	Reserved
4	WU_DOUT	R/W	1h	Wakeup output, active low or open drain. 0x0 = WAKEUP pin driven low 0x1 = WAKEUP pin is open drain output
3	WU_DIR	R/W	0h	Wakup pin control, high or open drain. The WAKEUP pin, when configured as an input, is active high. When the WAKEUP pin is configured as an output, it is open drain and thus it should have an external pullup and it is active low. 0x0 = WAKEUP pin is configured as and active high input 0x1 = WAKEUP pin is configured as an open drain output
2	BG_PD	R/W	0h	Bandgap power down. 0x0 = Normal 0x1 = Request to powerdown bandgap
1	LDO_PD	R/W	0h	LDO powerdown, functional mode. 0x0 = Normal 0x1 = Request to powerdown LDO
0	RTCCLKOUTEN	R/W	0h	Clockout output enable. 0x0 = Clock output disabled 0x1 = Clock output enabled

11.3.2.23 RTCSCR1 Register (offset = 1960h) [reset = 0h]

RTCSCR1 is shown in Figure 11-25 and described in Table 11-28.

The RTC Scratch Registers are general-purpose memory that can be used to store the RTC configuration status.

Figure 11-25. RTCSCR1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCRATCH0															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-28. RTCSCR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	SCRATCH0	R/W	0h	Scratch registers, available to program

11.3.2.24 RTCSCR2 Register (offset = 1961h) [reset = 0h]

RTCSCR2 is shown in Figure 11-26 and described in Table 11-29.

The RTC Scratch Registers are general-purpose memory that can be used to store the RTC configuration status.

Figure 11-26. RTCSCR2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCRATCH2															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-29. RTCSCR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	SCRATCH2	R/W	0h	Scratch registers, available to program

11.3.2.25 RTCSCR3 Register (offset = 1964h) [reset = 0h]

RTCSCR3 is shown in Figure 11-27 and described in Table 11-30.

The RTC Scratch Registers are general-purpose memory that can be used to store the RTC configuration status.

Figure 11-27. RTCSCR3 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCRATCH3															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-30. RTCSCR3 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	SCRATCH3	R/W	0h	Scratch registers, available to program

11.3.2.26 RTCSCR4 Register (offset = 1965h) [reset = 0h]

RTCSCR4 is shown in Figure 11-28 and described in Table 11-31.

The RTC Scratch Registers are general-purpose memory that can be used to store the RTC configuration status.

Figure 11-28. RTCSCR4 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCRATCH4															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-31. RTCSCR4 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	SCRATCH4	R/W	0h	Scratch registers, available to program

11.3.2.27 RGKR_LSW Register (offset = 196Ch) [reset = 0h]

RGKR_LSW is shown in Figure 11-29 and described in Table 11-32.

The RTC Gate-Keeper registers allow the RTC registers to be silently blocked from unintended writes unless a specific 32-bit value (0x95A4F1E0) is written into the gate-keeper registers first. Reads to the RTC registers are unaffected and will occur normally regardless of the value in these registers. The RGKR_LSW register is updated by the 32-kHz clock, so the DSP must allow time after a write for the write to complete.

The amount of time that writes are unblocked should be minimized to reduce the chances of RTC time corruption in the event of a CV_{DD} powerdown.

Figure 11-29. RGKR_LSW Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-32. RGKR_LSW Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	GKC_LSW	R/W	0h	RTC Gate-Keeper Code LSW. 0xF1E0 = DSP writes to RTC registers are not blocked, and when GKC_MSW = 0x95A4. Value is 0 to FFFFh. Others = DSP writes to RTC registers are blocked.

11.3.2.28 RGKR_MSW Register (offset = 196Dh) [reset = 0h]

RGKR_MSW is shown in Figure 11-30 and described in Table 11-33.

The RTC Gate-Keeper registers allow the RTC registers to be silently blocked from unintended writes unless a specific 32-bit value (0x95A4F1E0) is written into the gate-keeper registers first. Reads to the RTC registers are unaffected and will occur normally regardless of the value in these registers. The RGKR_MSW register is updated by the 32-kHz clock, so the DSP must allow time after a write for the write to complete.

The amount of time that writes are unblocked should be minimized to reduce the chances of RTC time corruption in the event of a CV_{DD} powerdown.

Figure 11-30. RGKR_MSW Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 11-33. RGKR_MSW Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	GKC_MSW	R/W	0h	RTC Gate-Keeper Code MSW. 0x95A4 = DSP writes to RTC registers are not blocked, and when GKC_LSW = 0xF1E0. Value is 0 to FFFFh. Others = DSP writes to RTC registers are blocked.

Successive Approximation (SAR) Analog-to-Digital Converter (ADC)

This following sections provide an overview of the Successive Approximation Register (SAR) analog-to-digital Converter (ADC) on the digital signal processor (DSP). The SAR is a 10-bit ADC using a switched capacitor architecture that converts an analog input signal to a digital value at a maximum rate of 64 ksps for use by the DSP. This SAR module supports six channels that are connected to four general purpose analog pins (GPAIN [3:0]), which can also be used as general-purpose digital outputs.

Topic **Page**

12.1 Introduction

This following sections provide an overview of the successive approximation (SAR) analog-to-digital converter (ADC) on the digital signal processor (DSP).

12.1.1 Purpose of the 10-bit SAR

The SAR in the device is a 10-bit ADC using a switched capacitor architecture that converts an analog input signal to a digital value at a maximum rate of 64 kilo samples per second (ksps) for use by the DSP. This SAR module supports six channels that are connected to four general-purpose analog pins (GPAIN [3:0]) that can be used as general-purpose outputs.

12.1.2 Features

- Up to 64 ksps
- Single conversion and continuous back-to-back conversion modes
- Interrupt driven or polling conversion or DMA event generation
- Internal configurable reference voltages of: V_{DD_ANA} or bandgap_1.0V or bandgap_0.8V
- Software controlled power down
- Individually configurable general-purpose digital outputs

12.1.3 Supported Use Case Statement

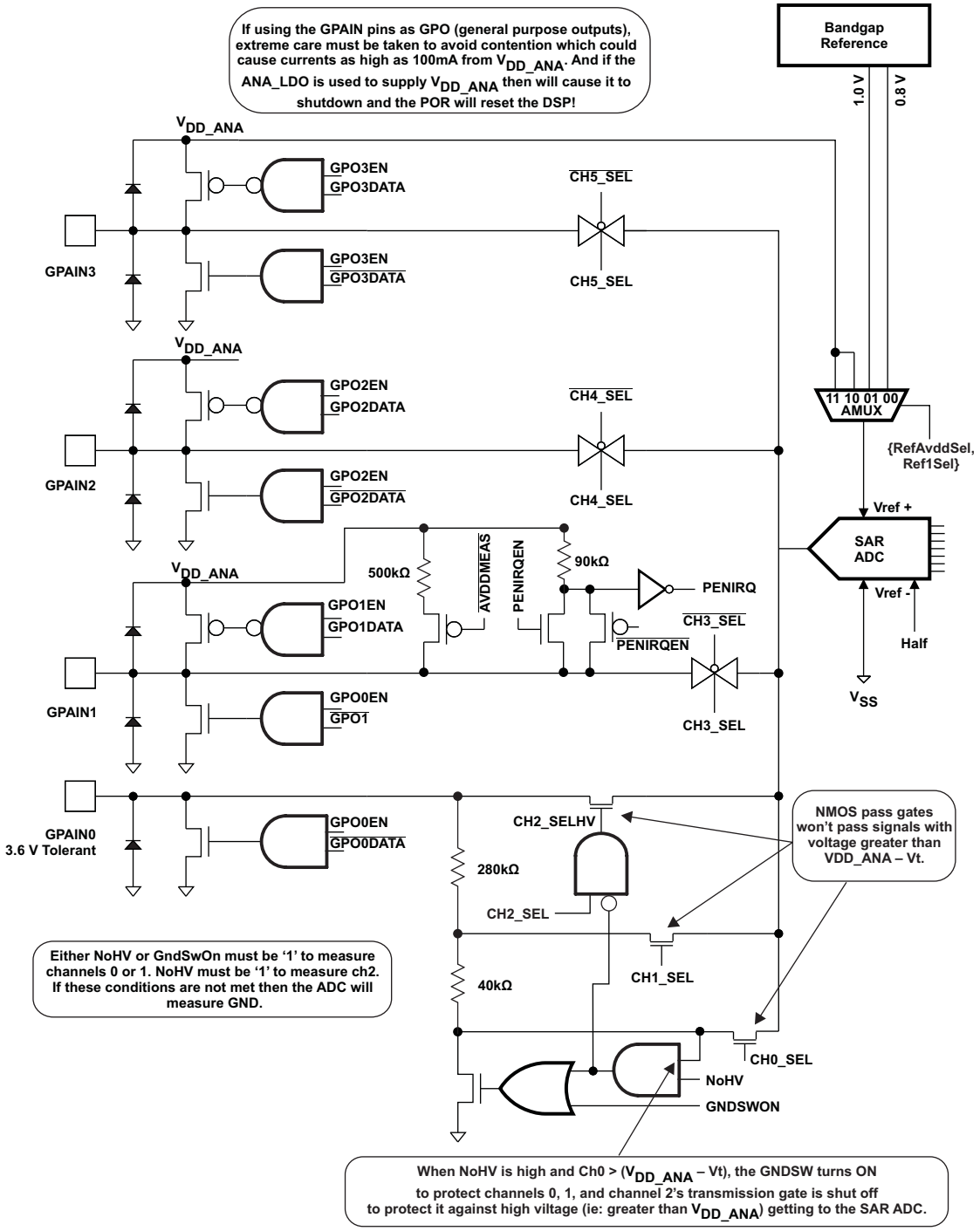
- Measure battery voltage, internal analog voltage (V_{DDA_ANA}), and volume control by measuring across a potentiometer
- 4-wire resistive touch screen coordinate pair measurement and pen down interrupt
- General-purpose outputs that can be driven high or low (except for GPAIN0, which only drives low)
- General-purpose voltage measurement

12.1.4 Industry Standard(s) Compliance Statement

This peripheral is not intended to conform to any specific industry standard.

12.1.5 Functional Block Diagram

Figure 12-1. SAR Converter



12.2 SAR Architecture

The 10-bit successive approximation analog-to-digital module in the device converts an analog input signal to a digital value for use by the DSP. The SAR module supports six channels, VIN[5:0]. These channels are connected to four general purpose analog pins, GPAIN[3:0]. (See Figure 12-1.) All general purpose analog pins can be used as general-purpose outputs by setting the corresponding GPIO bits in the SAR A/D Pin Control Register.

Once a conversion is initiated, the programmer must wait until the conversion completes before selecting another channel or initiating a new conversion. To indicate that a conversion is in progress, the ADCBUSY bit field is set. After the conversion completes, the ADCBUSY bit field changes from 1 to 0, indicating that the conversion data is available. The DSP can then read the data from the ADCDAT bits in the SAR A/D Data Register (SARDATA). The value of the CHSEL bit in SAR A/D Control Register (SARCTRL) is reproduced in the CHAN bit of the SARDATA register, so that the DSP can identify which samples were acquired from which channel.

A DMA event is also generated at the end of every conversion.

12.2.1 SAR Clock Control

The SAR A/D module can operate at a maximum clock rate of 2 MHz (500 ns) and takes 32 clocks cycles to convert a value. This results in a maximum sample rate of 64 ksps. The following equations describe the relationship between the A/D programmable control registers:

SAR A/D Clock Frequency = (System Clock Frequency) / (SystemClkDivisor + 1) ≤ 2 MHz

SAR A/D Conversion Time = (SAR A/D Clock Period * 32)

12.2.2 Memory Map

Table 12-1. SAR Memory Map

Address	Acronym	Description
7012h	SARCTRL	SAR A/D Control Register
7014h	SARDATA	SAR A/D Data Register
7016h	SARCLKCTRL	SAR A/D Clock Control Register
7018h	SARPINCTRL	SAR A/D Reference and Pin Control Register
701Ah	SARGPOCTRL	SAR A/D GPO Control Register

12.2.3 Signal Descriptions

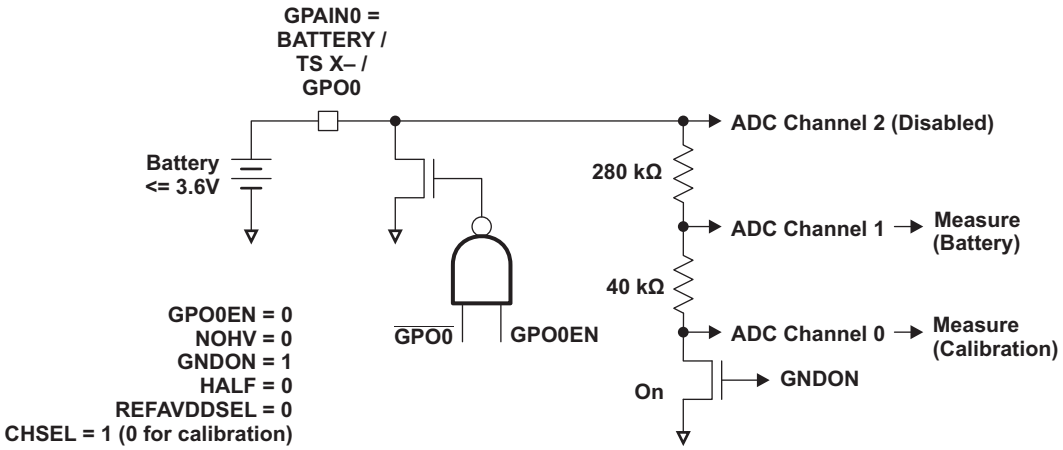
The device's GPAIN[3:0] pins can be configured as inputs to the SAR ADCs or they can be configured as general-purpose outputs that can be driven high or low (excluding GPAIN0 that can only be driven low). The SAR inputs can be used for battery measurement, internal voltage measurement, volume control, and touch screen control. GPAIN[0] is capable of accepting analog input voltage from 0 V up to 3.6 V while GPAIN[1:3] can accept a range of 0 V to V_{DDA_ANA}.

12.2.4 Battery Measurement

The SAR can be configured to measure a battery using GPAIN0.

To measure a battery that has less than 3.6 V, first connect the battery to GPAIN0 and set the ground switch (GNDON) bit of SAR channel 0. Next, calibrate the measurement by sampling the voltage at SAR Channel 0 (set CHAN to Channel 0). Channel 0 should now be tied to ground with GNDON set to 1. If there is an offset on Channel 0, this needs to be applied to the battery reading that can be obtained by switching to channel 1 and sampling the battery voltage through the voltage divider. The voltage divider divides the value from channel 1 by a factor of 8 (see data manual for limits) before being sampled by the SAR ADC. (See Figure 12-2.) After measuring the battery, the ground switch transistor should be shut off to eliminate current draw from the battery.

Figure 12-2. Battery Measurement

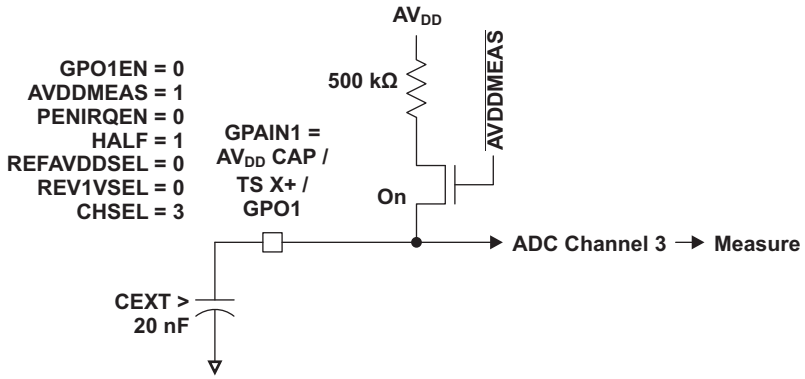


12.2.5 Internal Voltage Measurement

Using GPAIN1, the SAR can measure the internal voltage of V_{DDA_ANA} on AVDDMEAS channel 3 of the SAR.

To measure the internal AV_{DD}, set the internal voltage reference by setting in SAR A/D Reference and Pin Control Register (SARPINCTRL). A 20 nF cap is recommended to be connected between GPAIN1 and GND to provide low pass filtering and less measurement noise. Next, sample SAR channel 3. Selecting HALF = 1 has the effect of reducing the ADC's input sampled voltage in half. Therefore, with AV_{DD} (that is, V_{DDA_ANA}) at its max of 1.43 V, divided by two is 0.715 V. Then, with the ADC's VREF set to bandgap_0.8 V, the dynamic range is optimum. See Figure 12-3.

Figure 12-3. Voltage Measurement

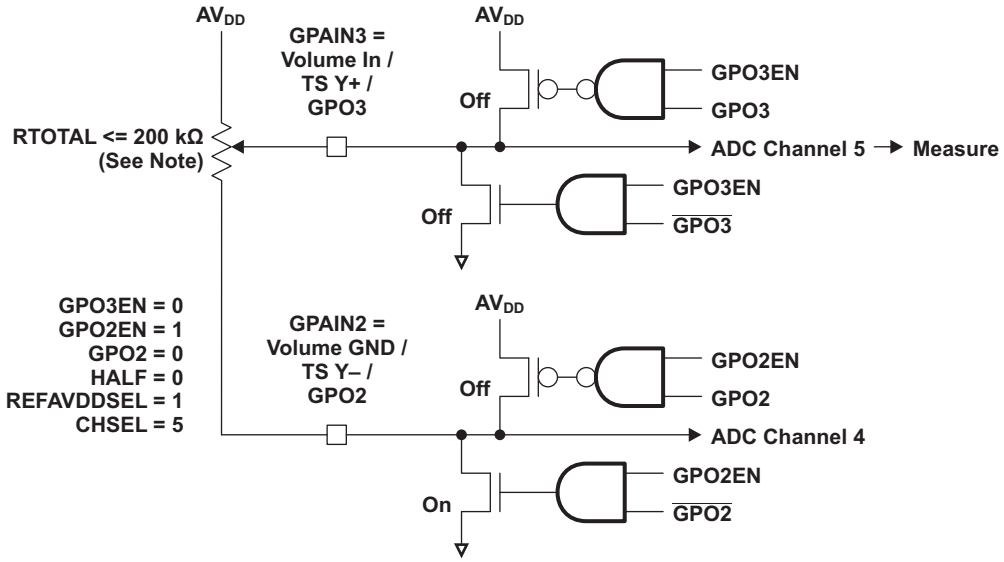


12.2.6 Volume Control

The SAR can be used to sample a volume control potentiometer connected across GPAIN2 and GPAIN3. Note that other combinations of the GPAIN[3:0] could be used to perform this function. We have chosen GPAIN2 and GPAIN3 for this example.

To use the SAR for volume control, place a potentiometer across GPAIN3 and GPAIN2, then ground GPAIN2 by clearing GPO2 and sample SAR channel 5 voltage. Use the settings in Figure 12-4.

Figure 12-4. Voltage Control

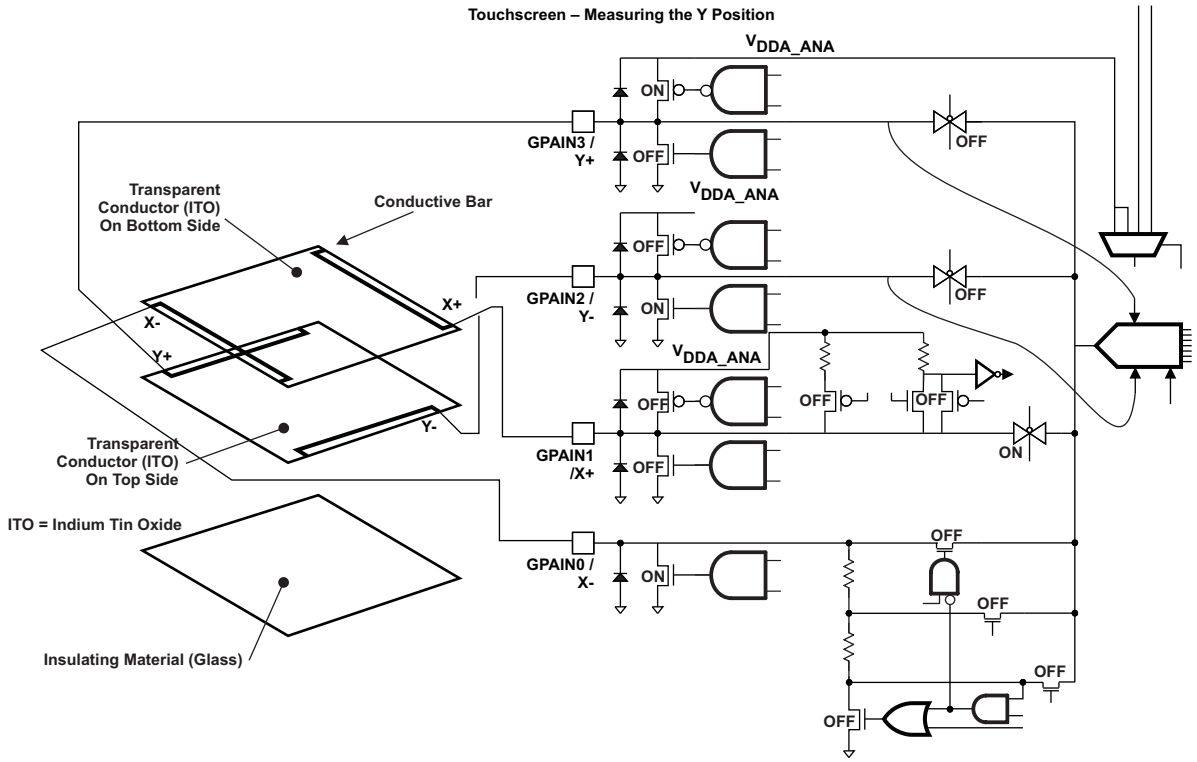


12.2.7 Touch Screen Digitizing

Using all 4 GPAIN pins, the SAR can be used for digitizing touch screen coordinates. With GPAIN3 as Y+, GPAIN2 as Y-, GPAIN1 as X+, and GPAIN0 as X-.

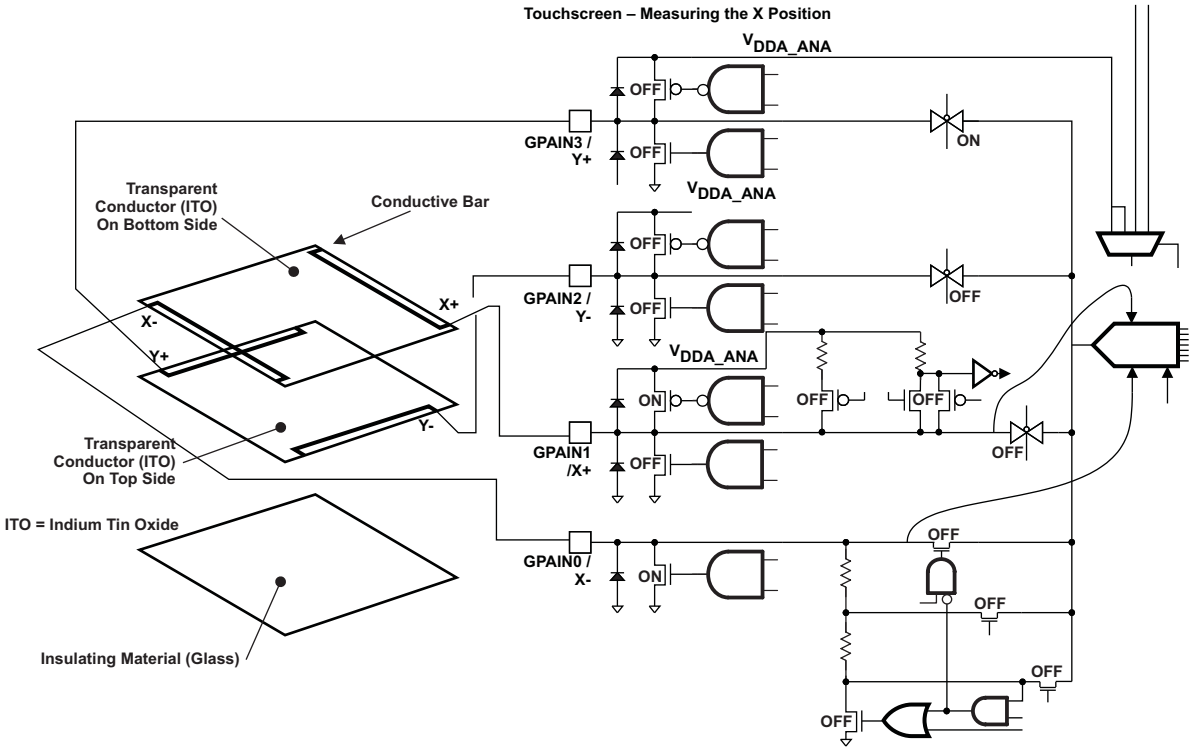
To measure Y position, enable GPAIN3 and GPAIN2 as general-purpose outputs using GPO3EN=1 and GPO2EN=1. Then, ground GPAIN2 by clearing GPO2 and drive GPAIN3 high by setting GPO3. Let the touch panel settle (duration depends on the bypass caps you have at the X+, X-, Y+, Y- terminals of the touch screen) and measure the voltage at GPAIN1 using SAR channel 3.

Figure 12-5. Y Position



To measure X position, enable GPAIN1 and GPAIN0 using GPO1EN=1 and GPO0EN=1. Then, ground GPAIN0 by clearing GPO0 and drive GPAIN1 high by setting GPO1. Let the touch panel settle, then measure the voltage at GPAIN3 using SAR channel 5.

Figure 12-6. X Position

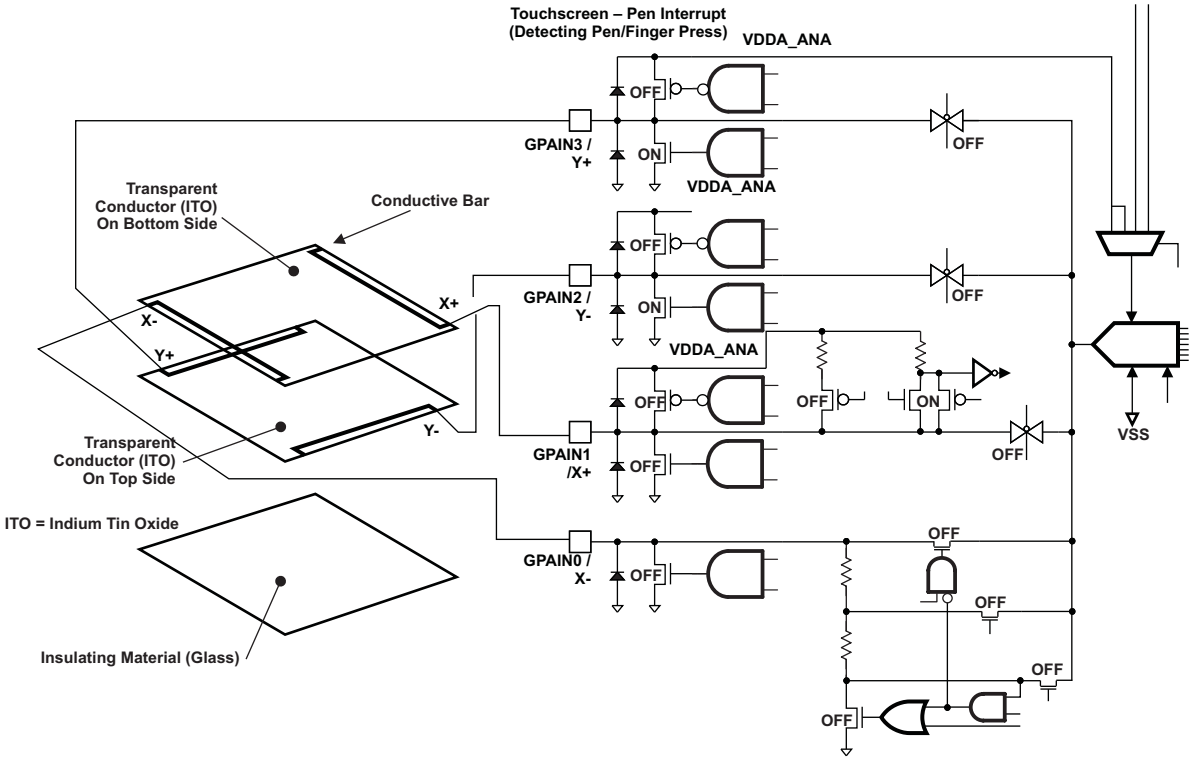


12.2.8 Touch Screen : Pen Press Interrupts

To detect when the touch screen is touched, the SAR peripheral has a PENIRQ feature. This feature makes it possible to detect touch events without continuous ADC polling.

The SAR should be configured as shown in Figure 12-7.

Figure 12-7. Pen Interrupt



The DSP should be configured to allow SAR interrupts. When the touch screen is not pressed, a pullup resistor biases the PENIRQ buffer high so that an interrupt is not generated. When the touch screen is pressed it creates a path to ground that is lower impedance than the pullup resistor. Therefore, the PENIRQ buffer generates an interrupt and the DSP can then take the steps necessary to digitize the X & Y coordinates.

12.2.9 General-Purpose Output

GPAIN[3:0] can be configured as general-purpose outputs. This is accomplished by enabling the GPAIN pin as an output in the SAR A/D GPO Control Register (SARGPOCTRL). After enabling the GPO you can set the output as grounded or driven high. In the case where V_{DDA_ANA} is supplied by the ANA_LDOO, care must be taken to avoid shorting GPAIN pins to ground as this will cause the maximum current of the ANA_LDOO to be exceeded and the POR circuit to reset the DSP. The total current from all GPAIN[3:0] pins to ground should never exceed Imax of the ANA_LDOO. For more information, see the device-specific data manual. In the case where V_{DDA_ANA} is supplied by some source other than the on-chip ANA_LDO, this is not an issue.

12.2.10 Reset Considerations

The SAR can only be reset by a hardware reset.

12.2.10.1 Software Reset Considerations

A software reset (such as a reset generated by the emulator) will not cause the SAR controller registers to be altered. After a software reset, the SAR controller continues to operate as it was configured prior to the reset.

There is no peripheral reset for the SAR.

12.2.10.2 Hardware Reset Considerations

A hardware reset of the processor causes the SAR controller registers to return to their default values after reset.

12.2.11 A/D Conversion

To start an analog to digital conversion the following steps must be executed:

1. Set SAR clock to be less than or equal to 2 MHz in the SAR A/D clock Control Register (SARCLKCTRL) for fastest conversion rate and operation of the A/D module.
   A/D function clock = (Sys Clk)/(ADCCLKDIV+1)
2. Write to the SARPINCTRL register (7018h) to power up the SAR circuits and select the SAR reference voltage.
3. Write a 1 to the ADCSTRT bit in the SARCTRL register and the desired channel for the conversion in the CHSEL bit field.
4. Read the ADCBUSY bit in the SARDATA register to ensure it is set to 1 to indicate the start of conversion. Due to delays between the CPU write instruction and the actual write to the SAR A/D registers the ADCBUSY bit must be set before proceeding.
5. ADCSTRT in the SARCTRL register and ADCBUSY bit in the SARDATA register are set to 0 to indicate the end of the conversion sequence.
6. Once ADCBUSY bit in the SARDATA register is set to 0, the SAR A/D Data Register contains the channel converted in the CHSEL bit field and the actual converted value in the ADCDAT bit field.

12.2.12 Interrupt Support

12.2.12.1 Interrupt Events and Requests

The SAR peripheral generates DSP interrupts every time an ADC conversion is completed and data is available to be read by the DSP. Additionally, when connected to a touch screen device, the SAR can be configured to detect when the touch screen is pressed and generate an interrupt without having to perform continuous conversion to poll the touch screen.

12.2.13 Emulation Considerations

The SAR controller is not affected by emulation halt events (such as breakpoints).

12.2.14 Conversion Example

To request a conversion the CPU must execute the following sequence of events:

1. Set SAR clock to be less than or equal to 2MHz in SAR Clock Control Register for fastest conversion rate and operation of the A/D module.
2. Write a “1” to the ADCSTRT bit of the SARCTRL register and the desired channel for conversion in the CHAN bit field in the SARDATA register.
3. ADCBUSY bit of the SARDATA register is set to “1” to indicate the start of A/D conversion.

4. Due to delays between the CPU write instruction and the actual write to the SAR A/D Registers, it is recommended to read the SARDATA register and verify the ADCBUSY bit is set to “1” before proceeding with step 6.
5. ADCSTRT and ADCBUSY bits are set to “0” to indicate the end of the conversion sequence. The SAR A/D module enters stand-by mode to conserve power until event 2 occurs over again.
6. Once ADCBUSY bit is set to “0”, the SARDATA register contains the channel converted in the CHAN bit field and the actual converted value in the ADCDAT bit field

12.3 SAR Registers

Table 12-2 list the memory mapped registers associated with the successive approximation (SAR) analog-to-digital converter (ADC).

Table 12-2. SAR Registers

CPU Word Address	Acronym	Register Name	Section
7012h	SARCTRL	SAR A/D Control Register	Section 12.3.1
7014h	SARDATA	SAR A/D Data Register	Section 12.3.2
7016h	SARCLKCTRL	SAR A/D Clock Control Register	Section 12.3.3
7018h	SARPINCTRL	SAR A/D Reference and Pin Control Register	Section 12.3.4
701Ah	SARGPOCTRL	SAR A/D GPO Control Register	Section 12.3.5

12.3.1 SARCTRL Register

The SAR A/D control register (SARCTRL) selects the channel number and indicates the start of a conversion.

The SAR A/D control register (SARCTRL) is shown in Figure 12-8 and described in Table 12-3.

Figure 12-8. SAR A/D Control Register (SARCTRL)

15	14 12	11	10	9 0
ADCSTRT	CHSEL	MULTICH	SNGLCONV	Reserved
R/W-0	R/W-000	R/W-0	R/W-0	R/W-0000000000

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-3. SARCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	ADCSTRT	R/W	0	Start Conversion 0 = No conversion. 1 = Start conversion cycle by setting START signal.
14-12	CHSEL	R/W	0	Channel Select 0 = Channel CH0 is selected. 1h = Channel CH1 is selected. 2h = Channel CH2 is selected. 3h = Channel CH3 is selected. 4h = Channel CH4 is selected. 5h = Channel CH5 is selected. 6h = All channels are off. 7h = All channels are off.
11	MULTICH	R/W	0	Multi Channel operation. 0 = Normal Mode. 1 = In this mode, the SAR state machine is optimized to give more time to sampling the analog input. This mode could possibly improve measurements in cases where the ADC input has abrupt voltage changes such as when changing from one input channel to another. The additional time given to sampling does not affect the 32 cycles for conversion, but it does come at the expense of settling time for the ADC's internal comparator. Therefore, this mode should not be used unless the above situation exists and it is determined to improve the measurements.
10	SNGLCONV	R/W	0	Single Conversion mode. 0 = Continuously perform back-to-back conversions, as long as ADCSTRT is set. 1 = Perform one conversion and stop. ADCSTRT must be cleared and then set high to perform another conversion.
9-0	Reserved	R	0	Reserved.

12.3.2 SARDATA Register

The SAR A/D data register (SARDATA) indicates if a conversion is in process, the actual digital data converted from the analog signal, and the channel belonging to this conversion.

The SAR A/D data register (SARDATA) is shown in Figure 12-9 and described in Table 12-4.

Figure 12-9. SAR A/D Data Register (SARDATA)

15	14 12	11 10	9 0
ADCBUSY	CHAN	Reserved	ADCDAT
R-0	R-111	R-00	R-0000000000

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-4. SARDATA Register Field Descriptions

Bit	Field	Type	Reset	Description
15	ADCBUSY	R	0	ADC Converter Busy. The ADCBUSY bit will be set three SAR clock cycles after the ADCSTRT bit is set. This will always read 0 when STATUSMASK=1. 0 = ADCDAT available. 1 = ADCBUSY performing a conversion. After ADCSTRT is high, the ADCBUSY becomes high.
14-12	CHAN	R	111	Channel Select. These bits will always read 000 when STATUSMASK = 1. 0 = Channel CH0 is selected. 1h = Channel CH1 is selected. 2h = Channel CH2 is selected. 3h = Channel CH3 is selected. 4h = Channel CH4 is selected. 5h = Channel CH5 is selected. 6h = Reserved. 7h = Reserved.
11-10	Reserved	R	0	Reserved.
9-0	ADCDAT	R	0	Converter Data from 0 to 3FFh.

12.3.3 SARCLKCTRL Register

The SAR A/D clock control register (SARCLKCTRL) sets the clock divider to control the speed of conversion. The clock rate of the SAR module must not exceed 2 MHz.

The SAR A/D clock control register (SARCLKCTRL) is shown in Figure 12-10 and described in Table 12-5.

Figure 12-10. SAR A/D Clock Control Register (SARCLKCTRL)

15	14	0
Reserved	ADCCLKDIV	
R-0	R/W-111111111111111	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-5. SARCLKCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	Reserved	R	0	0 = Reserved.
14-0	ADCCLKDIV	R/w	1111	0 = System Clock Divisor from 0 to 7FFFh. This specifies the divider rate of the system clock: F(SAR_Clock) = (F(System_Clock)) /(ADCCLKDIV[14:0] + 1) Allows for divide-by-1 up to divide-by-32768.

12.3.4 SARPINCTRL Register

The SAR A/D reference and pin control register (SARPINCTRL) controls the SAR's reference voltage and the circuits surrounding the GPAIN pins. The SAR's reference voltage determines the voltage at the ADC input that corresponds to the fullscale output code (ie: 1111111111b). Note, however, that due to the circuitry between the ADC input and the GPAIN[3:0] pins, the voltage at the ADC input isn't necessarily the same as the voltage at the GPAIN[3:0] pins. For example, the voltage divider on GPAIN0/Channel 1 scales the voltage of the signal by a factor of 8 before it arrives at the ADC. It is important to select the best voltage reference according to the voltage range of signal that will be digitized by the ADC so that the best resolution is obtained.

The SAR A/D reference and pin control register (SARPINCTRL) is shown in Figure 12-11 and described in Table 12-6.

Figure 12-11. SAR A/D Reference and Pin Control Register (SARPINCTRL)

15	14	13	12	11	10	9	8
Reserved	STATUSMASK	PWRUPBIAS	SARPWRUP	Reserved	REFBUFFEN	REFLVSEL	REFAVDDSEL
R-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0

7	5	4	3	2	1	0
Reserved		TOUCHSCREENMODE	AVDDMEAS	Reserved	GNDON	HALF
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-6. SARPINCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	Reserved	R	0	0 = Reserved.
14	STATUSMASK	R/W	0	Asserting this bit causes bits 12-15 of the SAR_DATA register to be forced to 0000. Those four bits correspond to the ChannelSelected and ADCBusy bits. The purpose for clearing/masking them is so that DMA transfers from the SAR to a memory buffer do not include those status bits and thus post-processing of the memory buffer is not required to strip the status bits out of the sample set. 0 = The SAR_DATA register includes the status info in bits 12-15. 1 = The SAR_DATA register bits 12-15 are always read as 0000.
13	PWRUPBIAS	R/W	0	Enables or disables the current bias circuit that is needed for the SAR to perform A/D conversions. 0 = Powered Down. Low power setting. 1 = Powered Up. Required setting for performing A/D conversions.
12	SARPWRUP	R/W	0	Enables or disables the analog power to the SAR. 0 = SAR analog Powered down. 1 = SAR analog power present.
11	Reserved	R	0	0 = Reserved.
10	REFBUFFEN	R/W	0	Reference Buffer enable. The reference buffer can be disabled to save power when the ADC's VREF is set to VDDA_ANA (REFAVDDSEL=1) or when VREF is provided by the TOUCHSCREENMODE pins (TOUCHSCREENMODE=1). The reference buffer must be enabled whenever one of the bandgap reference voltages are used (ie: REFAVDDSEL=0). REFBUFEN can be 0 or 1 when TOUCHSCREENMODE=1. 0 = Reference Buffer is disabled. Low power setting. 1 = Reference Buffer is enabled. Required when using bandgap generated VREF.
9	REFLVSEL	R/W	0	Bandgap-based reference voltage value select. The on-chip bandgap provides two references to the SAR peripheral: 0.8v & 1.0v. This register is used to select which bandgap reference voltage is used when REFAVDDSEL=0. In general, the lowest VREF should be used to get the best resolution from the converter. However, VREF should always be greater than the input signal else clipping will occur. 0 = Bandgap-Based Reference Voltage set to 0.8V. 1 = Bandgap-Based Reference Voltage set to 1V.

Table 12-6. SARPINCTRL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8	REFAVDDSEL	R/W	0	ADC Reference Voltage Select. When asserted, this register selects VDDA_ANA as the voltage reference for the SAR ADC. Otherwise, one of the two selectable bandgap references will be used. This register has no effect when TOUCHSCREENMODE=1. 0 = Reference Voltage based on Bandgap. The voltage value of the Bandgap reference is dictated by REFLVSEL. 1 = Reference Voltage set to Analog Voltage (VDD_ANA).
7-6	Reserved	R	0	0 = Reserved.
5	Reserved	R	0	0 = Reserved must write 0.
4	TOUCHSCREENMODE	R/W	0	Enables Touch Screen Mode. In this mode, the SAR detects which coordinate, X or Y, is being measured based on the GPOxEN and GPOxDATA settings, and switches the ADC's VREF+ and VREF- to the appropriate GPAIN[3:0] pins to reduce offset and gain errors caused by the dc current flowing thru the touch screen and the drop across the GPO output transistors. 0 = TOUCHSCREENMODE is Disabled. 1 = TOUCHSCREENMODE is Enabled.
3	AVDDMEAS	R/W	0	Enable measurement of internal analog voltage (VDD_ANA) on SAR Channel 3. 0 = PMOS switch on channel 3 is open, thus VDDA_ANA is not connected to channel 3 ADC input. 1 = PMOS switch on channel 3 is closed, thus VDDA_ANA is connected to channel 3 ADC input thru a pullup resistor to enable measuring the internal VDDA_ANA voltage. Note, when measuring VDDA_ANA, an independent voltage reference is needed for the ADC. So one of the two bandgap voltages should be used. Half mode will also be necessary since VDDA_ANA is greater than the two bandgap voltage references.
2	Reserved	R	0	0 = Reserved.
1	GNDON	R/W	0	Ground SAR Analog Channel 0 and introduce a voltage resistor divider network in SAR Channel 1. 0 = SAR Analog Channel 0 is not grounded. 1 = SAR Analog Channel 0 grounded. Introduces a divider into the SAR Channel 1 input of 1/8 * GPAIN0. See datasheet for tolerance specs on the resistor divider.
0	HALF	R/W	0	Divides the ADC analog input by two before doing the conversion. The attenuation is accomplished by only charging half of the SAR ADC's internal capacitive array during the sample phase, then the whole capacitive array is used for the successive approximation conversion. By sampling with half the capacitance and comparing against VREF with the full capacitance, the input voltage is attenuated by a factor of 2. 0 = A-to-D conversion is based on Vin. 1 = A-to-D conversion is based on Vin / 2.

12.3.5 SARGPOCTRL Register

The SAR A/D general-purpose output control register (SARGPOCTRL) sets the corresponding GPAIN pins to general-purpose outputs or analog inputs. In general-purpose output mode, the GPAIN pins can be individually driven high or low.

The SAR A/D GPO control register (SARGPOCTRL) is shown in Figure 12-12 and described in Table 12-7.

Figure 12-12. SAR A/D GPO Control Register (SARGPOCTRL)

15						10	9	8
Reserved							PENIRQ	PENIRQEN
R-00000							R/W-0	R/W-0
7	6	5	4	3	2	1	0	
GPO3EN	GPO2EN	GPO1EN	GPO0EN	GPO3	GPO2	GPO1	GPO0	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-7. SARGPOCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0	0 = Reserved.
9	PENIRQ	R/W	0	Pen Interrupt Request Status. 0 = No pen input detected. 1 = Pen input detected.
8	PENIRQEN	R/W	0	Pen Interrupt Request Enable. 0 = Disable the pen interrupt. 1 = Enable the pen interrupt and route to the CPU's SAR interrupt signal.
7	GPO3EN	R/W	0	Enable General Purpose Output on GPAIN3. Allows using GPAIN3 as a general output. 0 = GPAIN3 output driver disabled. 1 = GPAIN3 used as output.
6	GPO2EN	R/W	0	Enable General Purpose Output on GPAIN2. Allows using GPAIN2 as a general output. 0 = GPAIN2 output driver disabled. 1 = GPAIN2 used as output.
5	GPO1EN	R/W	0	Enable General Purpose Output on GPAIN1. Allows using GPAIN1 as a general output. 0 = GPAIN1 output driver disabled. 1 = GPAIN1 used as output.
4	GPO0EN	R/W	0	Enable General Purpose Output on GPAIN0. Allows using GPAIN0 as a general output 0 = GPAIN0 output driver disabled. 1 = GPAIN0 used as output.
3	GPO3	R/W	0	Drive high or low GPAIN3 when set as General Purpose Output. 0 = GPAIN3 grounded. 1 = GPAIN3 driven high.
2	GPO2	R/W	0	Drive high or low GPAIN2 when set as General Purpose Output. 0 = GPAIN2 grounded. 1 = GPAIN2 driven high.
1	GPO1	R/W	0	Drive high or low GPAIN1 when set as General Purpose Output. 0 = GPAIN1 grounded. 1 = GPAIN1 driven high.
0	GPO0	R/W	0	Ground GPAIN0 when set as General Purpose Output. 0 = GPAIN0 grounded. 1 = GPAIN0 driven high.

Serial Peripheral Interface (SPI)

This chapter describes the features and operations of the serial peripheral interface (SPI).

Topic **Page**

13.1 Introduction

The following sections describe the serial peripheral interface (SPI) in the digital signal processor (DSP).

13.1.1 Purpose of the Peripheral

The SPI is a high-speed synchronous serial input/output port that allows a serial bit stream of programmed length (1 to 32 bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI supports multi-chip operation of up to four SPI slave devices. The SPI can operate as a master device only.

The SPI is normally used for communication between the DSP and external peripherals. Typical applications include an interface to external I/O or peripheral expansion via devices such as shift registers, display drivers, SPI EEPROMs, and analog-to-digital converters.

13.1.2 Features

The SPI has the following features:

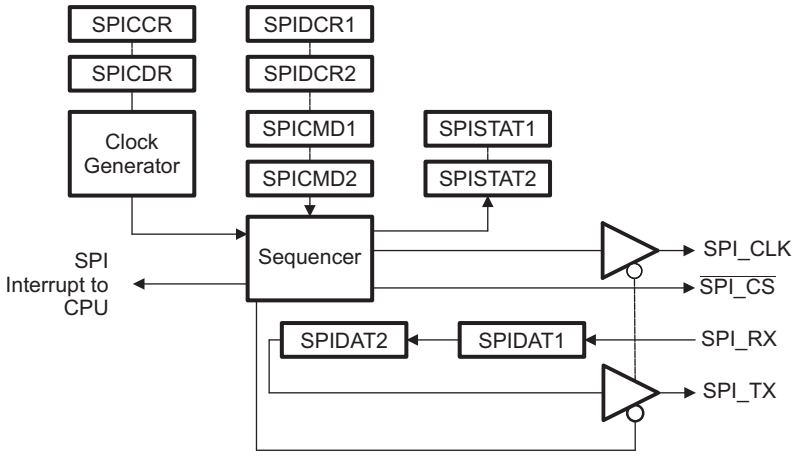
- Programmable divider for serial data clock generation.
- Four pin interface (SPI_CLK, SPI_CS*n*, SPI_TX and SPI_RX).
- Programmable data length (1 to 32 bits).
- 4 external chip select signals.
- Programmable transfer or frame size (1 to 4096 characters).
- Optional interrupt generation on character completion or frame completion.
- Programmable SPI_CS*n* to SPI_TX delay insertion from 0 to 3 SPI_CLK cycles.
- Programmable signal polarities.
- Programmable active clock edge.
- Internal loopback mode for testing.

The SPI can only operate in master mode only, slave mode is not supported.

13.1.3 Functional Block Diagram

Figure 13-1 illustrates the main components of the SPI.

Figure 13-1. Serial Peripheral Interface (SPI) Block Diagram

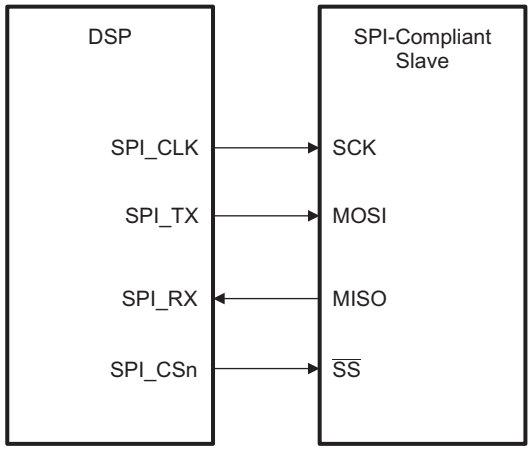


13.1.4 Supported Use Case Statement

The SPI is intended for communication between the DSP and up to four SPI-complaint slave devices. Typical applications include an interface to external I/O or peripheral expansion via devices such as shift registers, display drivers, SPI EEPROMs, and analog-to-digital converters. The programmable configuration capability of the SPI allows it to interface to a variety of SPI format devices without the need for glue logic.

A typical SPI interface with a single slave device is shown in Figure 13-2. The DSP controls the flow of communication by providing shift-clock (SPI_CLK) and slave-select signals (SPI_CS*n*).

Figure 13-2. Typical SPI Interface



13.1.5 Industry Standard(s) Compliance Statement

The SPI does not conform to a specific industry standard.

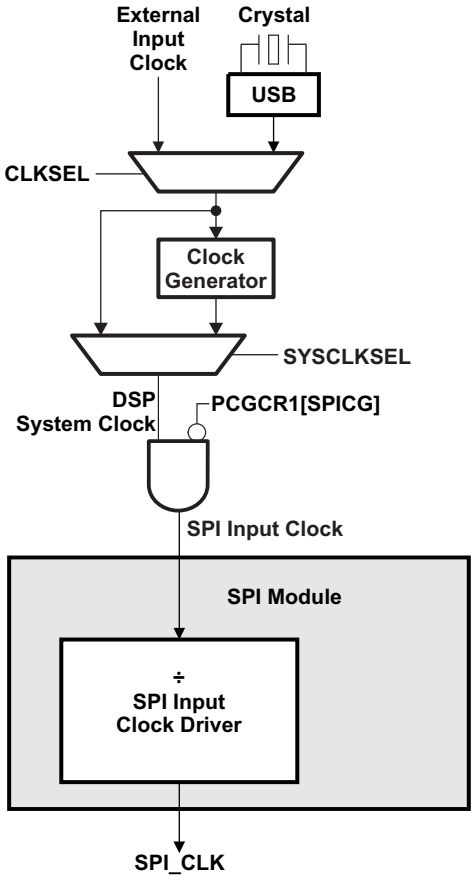
13.2 Serial Peripheral Interface Architecture

13.2.1 Clock Control

As shown in Figure 13-3, the clock generator receives either the USB oscillator or a signal from an external clock source and produces the DSP system clock. This internal clock is used by the DSP CPU and peripherals. A programmable clock divider in the SPI module divides down the SPI input clock to produce the SPI interface output clock (SPI_CLK). For proper device operation, the frequency of the SPI input clock must be at least four times greater than the frequency of SPI_CLK.

The DSP device includes logic which can be used to gate the clock to its on-chip peripherals, including the SPI. The input clock to the SPI can be enabled and disabled through the peripheral clock gating configuration register 1 (PCGCR1).

Figure 13-3. Clocking Diagram for the SPI



13.2.2 Signal Descriptions

Table 13-1 shows the SPI pins used to interface to external devices. A typical SPI interface with a single slave device is shown in Figure 13-2.

Table 13-1. Serial Peripheral Interface (SPI) Pins

Pin	Type	Function
SPI_CLK	Output	Serial clock output pin (also referred to as SCK)
SPI_TX	Output	Serial data output pin (also referred to as Master Output - Slave Input, or MOSI)
SPI_RX	Input	Serial data input pin (also referred to Master Input - Slave Output, or MISO)
SPI_CS0	Output	Slave 0 chip select pin (also referred to as Slave Select, or $\overline{SS}$)
SPI_CS1	Output	Slave 1 chip select pin (also referred to as Slave Select, or $\overline{SS}$)
SPI_CS2	Output	Slave 2 chip select pin (also referred to as Slave Select, or $\overline{SS}$)
SPI_CS3	Output	Slave 3 chip select pin (also referred to as Slave Select, or $\overline{SS}$)

13.2.3 Units of Data: Characters and Frames

This documentation describes SPI module communication using two terms: characters and frames. Characters are the smallest unit of data that can be transferred by the SPI module. During a transfer, the SPI generates enough clock cycles to send a character of data. The length of the character is specified by the CLEN bits of SPICMD2. The character length can be from 1 to 32 bits and can be of different size each time the SPI initiates a character transfer.

The total number of characters transmitted by the SPI module is referred to as a frame. At the beginning of a frame, the SPI module will assert the chip select pin specified by the chip select bits (CSNUM) of SPICMD2 and transfer a character of data. The SPI module will keep the chip select pin asserted until all the characters in the frame have been transferred. The frame length bits (FLEN) of SPICMD1 define the total number of characters in a frame. A frame can have up to 4096 characters. Please note that you should complete a frame transfer before using a different chip select pin.

The character count bits (CCNT) of SPISTAT2 keep track of the total number of times a character has been transferred by the SPI module. The character count is not affected by the size of the character. For example, a transmission of an 8-bit character followed by a 16-bit character increments CCNT by two.

The frame complete bit (FC) of SPISTAT1 is set after all the requested characters in a frame have been transferred. This bit is reset when SPISTAT1 is read or when a new SPI transfer is initiated via a write of a read or write command to CMD in SPICMD2.

13.2.4 Chip Select Control

The SPI module initiates a slave access by asserting one of its four chip select pins. The chip select pin remains asserted until an entire frame of data has been transferred. You can specify which chip select pin is activated through the chip select bits (CSNUM) of SPICMD2. Please note that writing to SPICMD2 immediately initiates a slave access; therefore you should only write to CSNUM when you are ready to initiate a slave access. Also, after initiating an access to a particular slave, you must complete the entire frame transfer to that slave before activating a different chip select pin.

The polarity of each of the four chip select pins can be configured through the CSP*n* bits of SPIDCR1 and SPIDCR2. Setting CSP*n* = 0 configures the corresponding SPI_CS*n* pin as active low while setting CSP*n* = 1 configures the SPI_CS*n* pin as active high.

13.2.5 Clock Polarity and Phase

Before communication between the SPI module and a slave can begin, you must specify the clock polarity and clock phase to be used. Together, the clock polarity and clock phase specify on which clock edge data is shifted in and out as well as the default state of the clock signal.

The configuration of the clock polarity and clock phase is referred to as the SPI mode. There are a total of four SPI modes (see Table 13-2), all which are supported by the SPI module. The clock polarity and clock phase must be configured correctly for successful communication between the SPI module and slave.

The clock polarity and phase can be specified through the CKP*n* and CKPH*n* bits of the SPI device configuration register (SPIDC). You can program a different clock polarity and phase for each slave.

Table 13-2. Definition of SPI Modes

SPI Mode	Clock Polarity	Clock Phase
0	Active low (base value of clock is low)	Data shifted out on the falling edge, input captured on the rising edge.
1	Active low (base value of clock is low)	Data shifted out on the rising edge, input captured on the falling edge.
2	Active high (base value of clock is high)	Data shifted out on the rising edge, input captured on the falling edge.
3	Active high (base value of clock is high)	Data shifted out on the falling edge, input captured on the rising edge.

The timing diagrams for the four possible SPI modes are shown in Figure 13-4 through Figure 13-7. Please note the following about these figures:

- Although the timing diagrams show an 8-bit character transfer, the character length can be set to 1 through 32 bits. The character length is selected with the CLEN bits SPICMD2.
- The number of characters transferred during one slave access is specified through the FLEN bits of SPICMD1. The figures show the case of FLEN = 0 (1 character).
- The polarity of the chip select pins (SPI_CS*n*) can be configured through the CSP*n* bits of SPIDCR1 and SPIDCR2. The figures show a chip select polarity of active low.
- The SPI module automatically delays the first clock edge with respect to the activation of the SPI_CS*n* pin by half a SPI_CLK cycle plus a system clock cycle. Additional clock delay cycles can be added using the data delay bits (DD*n*) of SPIDCR1 and SPIDCR2. The figures below show the case of DD*n* = 0 (zero data delay) and CLKDV is odd.

Figure 13-4. SPI Mode 0 Transfer (CKP*n* = 0, CKPH*n* = 0)

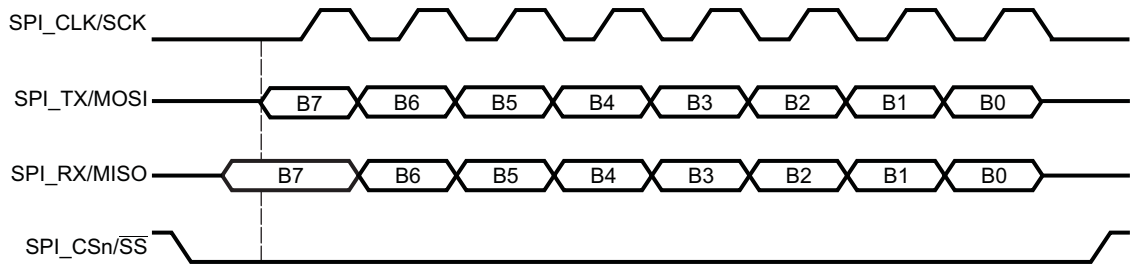


Figure 13-5. SPI Mode 1 Transfer (CKP*n* = 0, CKPH*n* = 1)

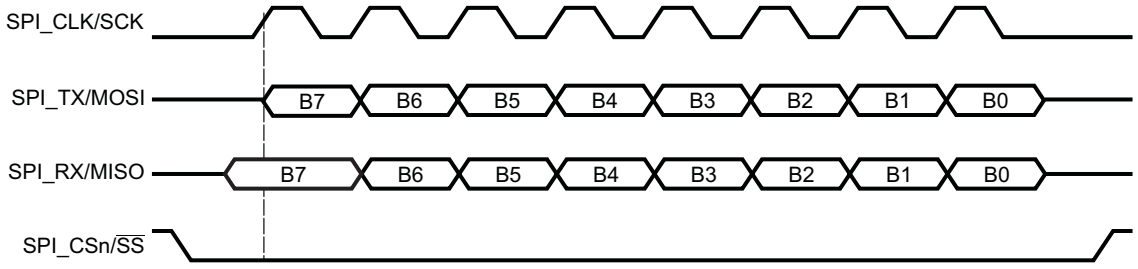


Figure 13-6. SPI Mode 2 Transfer (CKP*n* = 1, CKPH*n* = 0)

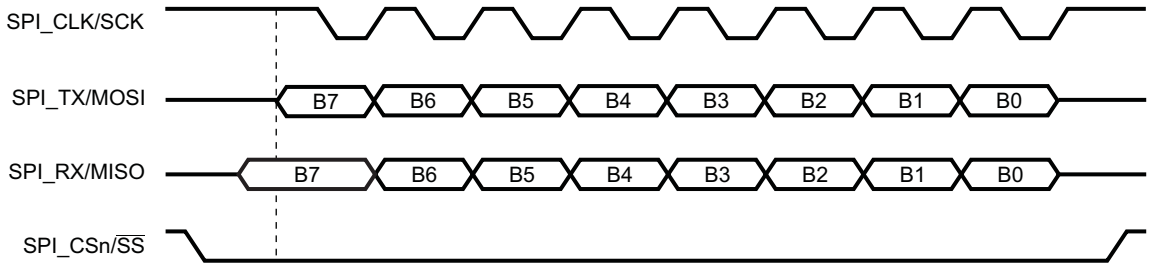
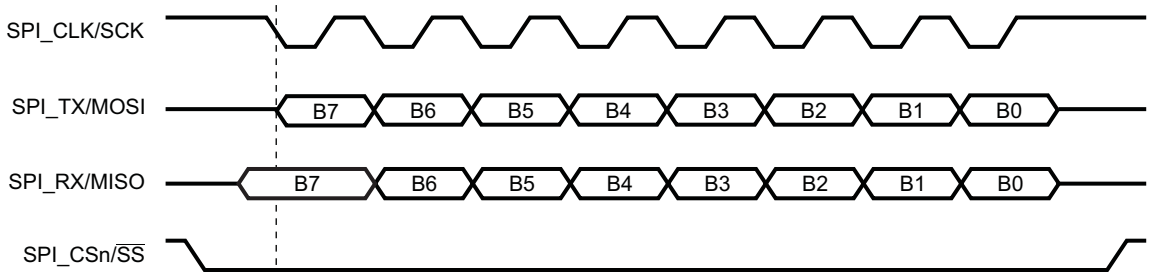


Figure 13-7. SPI Mode 3 Transfer (CKP*n* = 1, CKPH*n* = 1)

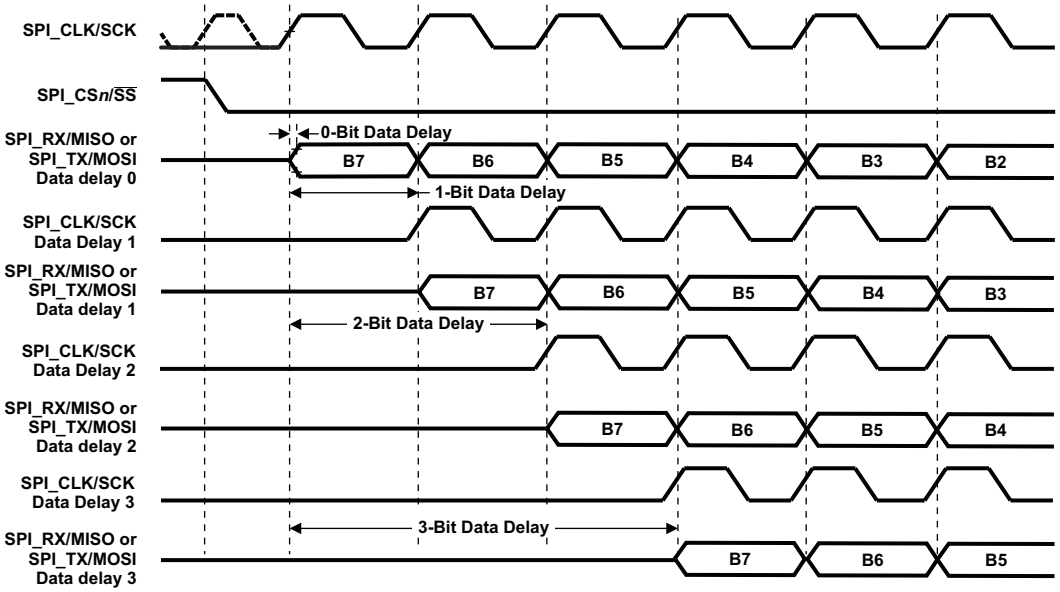


13.2.6 Data Delay

As described in the previous section, the SPI module automatically delays the first clock edge with respect to the activation of the SPI_CS*n* pin by half a SPI_CLK cycle plus a system clock cycle. You can program the SPI module to insert additional clock delay cycles using the data delay bits (DD*n*) of SPIDCR1 and SPIDCR2 to determine the number of clock delay cycles to insert. The data delay can be specified from zero to three clock cycles (DD*n* = 00b - 11b). Figure 13-8 below shows the effect of the data delay bits. Please note the following about Figure 13-8:

- The data transferred is an 8-bit character with bits labeled B7, B6, B5, and so on. The character length can be set to 1 through 32 bits through the CLEN bits of the SPICMD2).
- The polarity of the chip select pins (SPI_CS*n*) can be configured through the CSP*n* bits of SPIDCR1 and SPIDCR2. The figures show a chip select polarity of active low.
- The clock polarity and clock phase can be configured through the CKP*n* and CKPH*n* bits of SPIDCR1 and SPIDCR2. Figure 13-8 shows an example CKP*n* = 0 and CKPH*n* = 1 (SPI Mode 1).
- The first clock edge is automatically delayed by half a SPI_CLK cycle plus a system clock cycle by the SPI module.

Figure 13-8. Range of Programmable Data Delay



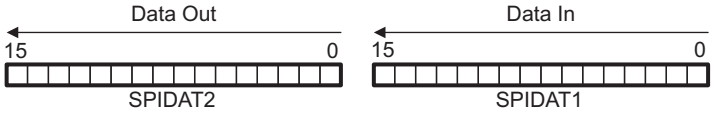
13.2.7 Data Input and Output

The data registers (SPIDAT1 and SPIDAT2) hold the data that is either shifted in or shifted out during a slave access. The SPIDAT1 and SPIDAT2 registers are treated as a single 32-bit shift register. Data received by the SPI is shifted into the least-significant bit of SPIDAT1 and the contents of both registers are shifted to the left. Similarly, data transferred by the SPI is shifted out of the most-significant bit of SPIDAT2 and the contents of both registers are shifted to the left. This process is illustrated in Figure 13-9.

The CLEN bits of SPICMD2 determine the length of the character. The character length can be set to any value between 1 and 32 bits.

The data registers are not cleared between reads or writes, and old may exist in the registers. Therefore, you must : (1) clear the data registers prior to initiating a read; and (2) mask off any unneeded data upon completing the read.

Figure 13-9. Data Shift Process



13.2.8 Loopback Mode

The SPI includes a loopback mode which can be used for testing purposes. In the loopback mode, the SPI_TX and SPI_RX are internally connected to each other. All SPI modes are usable when loopback mode is enabled.

13.2.9 Monitoring SPI Activity

The status registers (SPISTAT1 and SPISTAT2) contain indicators that allow you to monitor the progression of a frame transfer. These bits are described in Table 13-3. Additionally, the SPI module can be configured to generate interrupts after a frame or character has been transferred. These interrupts are described in Section 13.2.13.

Table 13-3. SPI Module Status Bits

Status Bit	Register	Description
FC	SPISTAT1	The frame complete bit. This bit is set after all the requested characters in a frame have been transferred. This bit is reset when SPISTAT1 is read or when a new SPI transfer is initiated via a write of a read or write command to CMD in SPICMD2.
CC	SPISTAT1	Character complete bit. This bit is set after each transfer is completed. This bit is reset when SPISTAT1 is read or when a new SPI transfer is initiated via a write of a read or write command to CMD in SPICMD2.
CCNT	SPISTAT2	Character count bits. These bits keep track of the total number of times a character has been transferred by the SPI module. The CCNT bits are not affected by the size of the character. For example, a transmission of an 8-bit character followed by a 16-bit character increments CCNT by two.
BUSY	SPISTAT1	Busy bit. This bit is set during an active character transfer. Between characters this bit will be cleared to signal that the data registers can be accessed.

13.2.10 Slave Access

After you have initialized the SPI following the steps outlined in Section 13.2.12, you can follow these steps to initiate a slave access.

1. Specify the total number of characters you intend to transfer by setting the FLEN bits of SPICMD1. You must finish transferring this number of characters before initiating transfers with a different slave.
2. For writes, load the output data value into SPIDAT1 and SPIDAT2. For reads, clear SPIDAT1 and SPIDAT2. The SPI module will shift data out starting with the most-significant bit of SPIDAT2. Similarly, the SPI will shift data in starting at the least-significant bit of SPIDAT1. The amount of bits the SPI module shifts in or out is determined by the CLEN bits of SPICMD2.
3. Write to the command register (SPICMD2) to start the SPI access. The value you write SPICMD2 must set CSNUM, CLEN, and CMD to the chip select, character length, and command to be used.
4. Poll SPISTAT1 until CC = 1 or wait for the character complete interrupt (CIRQ).
5. Poll SPISTAT1 until BUSY = 0. This ensures that a character is not being transferred.
6. For reads, the data is loaded into SPIDAT1 and SPIDAT2. You must mask off any invalid bits when reading SPIDAT1 and SPIDAT2. The number of valid bits is determined by CLEN in SPICMD2.
7. Repeat steps 2 though 5 until all the characters have been transferred.

Please note the following points:

- The chip select pin specified in SPICMD2 will be activated as soon as the first character transfer is initiated. The chip select pin will remain activated until all the characters specified by FLEN have been transferred.
- SPIDAT1 and SPIDAT2 are treated as a 32-bit shift register where SPIDAT2 holds the most-significant word and SPIDAT1 holds the least-significant word. During writes, data is shifted out starting with the most-significant bit of SPIDAT2. For reads, data is shifted in starting at the least-significant bit of SPIDAT1.
- It is important for you to always load SPIDAT1 and SPIDAT2 with the values you intend to shift out before initiating an SPI write. It is equally important to always mask off invalid bits after reading values from SPIDAT1 and SPIDAT2 after every SPI read.
- You must finish transferring the number of characters specified by FLEN in SPICMD1 before initiating transfers with a different slave.
- SPI_CLK will only be activated while data is being shifted out.

Figure 13-10 illustrates the steps described above.

Figure 13-10. Flow Diagram for SPI Read or Write

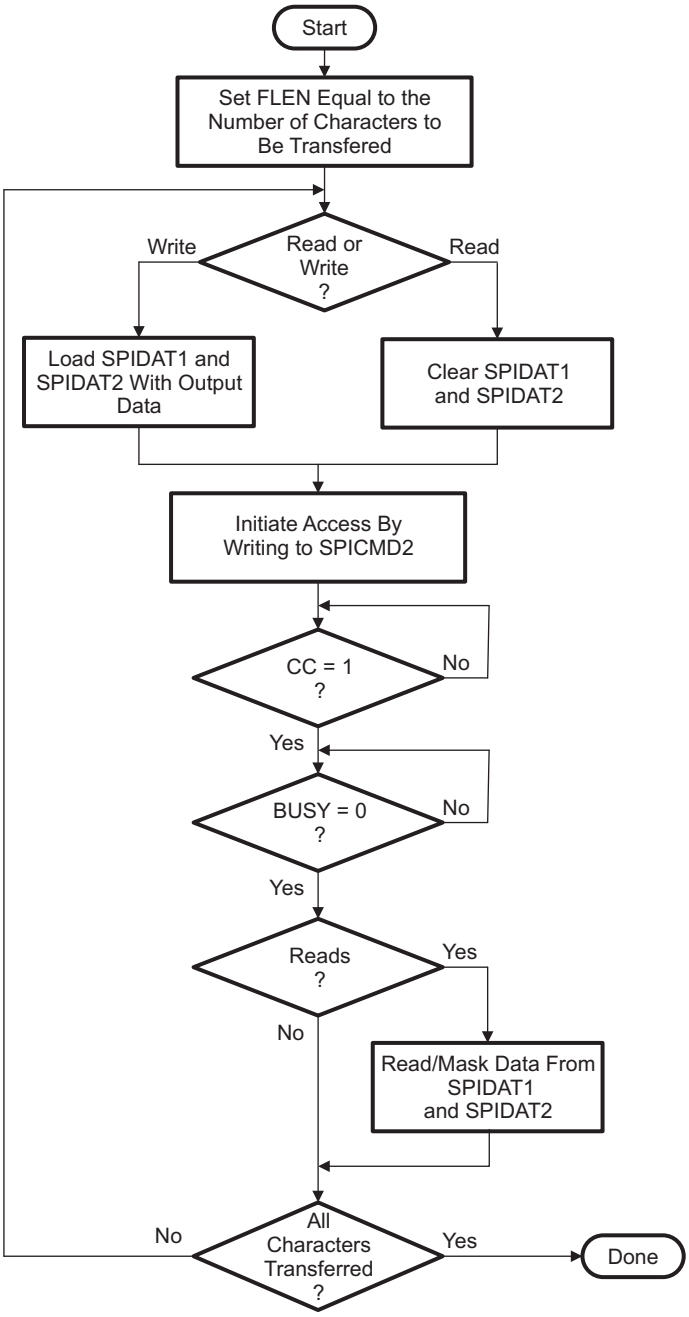
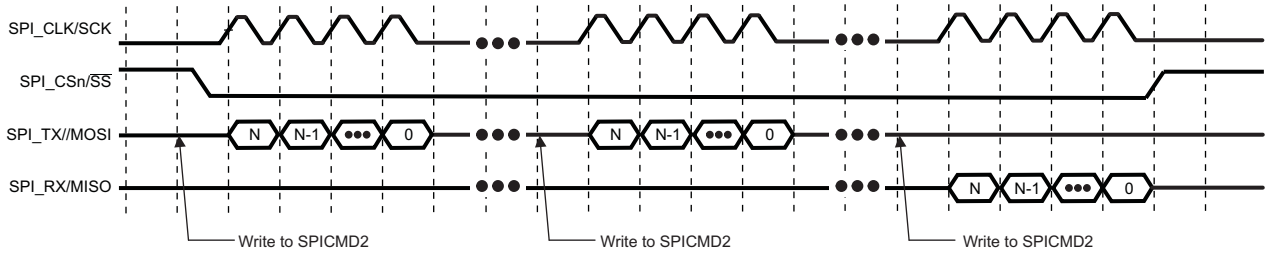


Figure 13-11 illustrates the activity at the pins of the SPI module.

Figure 13-11. SPI Access



13.2.11 Reset Considerations

The SPI module has two reset sources: software reset and hardware reset.

13.2.11.1 Software Reset Considerations

The SPI module can be reset by software through the RST bit in the clock control register (SPICCR) or through the PG4_RST bit in the peripheral reset control register (PRCR).

When RST is set to 1, the SPI module is reset.

The entire SPI module may also be reset through the PG4_RST bit of PRCR. When this reset is asserted, all SPI registers are set to their default values. The SPI remains inactive until programmed by software. Note that from the SPI perspective, this reset appears as a hardware reset.

NOTE: The PG4_RST bit of PRCR resets other peripherals besides the SPI. For more details on this bit and register, refer to the device-specific data manual.

13.2.11.2 Hardware Reset Considerations

When the SPI is reset due to a device reset, all the SPI registers are set to their default values. The SPI module remains inactive until programmed by software.

13.2.12 Initialization

The following initialization procedure describes the basic setup of the SPI module. Please note that the exact configuration will depend on the characteristics of the slave device. For specific examples, please refer to Section 13.3.

1. Ensure the SPI is out reset by setting SPI_RST = 0 in the peripheral reset control register (PRCR). See the device-specific data manual for more information on PRCR.
2. Enable the SPI input clock by setting SPICG to 0 in the peripheral clock gating configuration register (PCGCR1). See the device-specific data manual for more information on PCGCR1.
3. Set CLKEN = 0 to disable SPI_CLK. The CLKEN bit is part of SPICCR.
4. Set CLKDV in SPICDR to provide the appropriate SPI_CLK frequency. Note that for proper device operation, CLKDV must be greater than or equal to 3. Also, note that the clock frequency selected in this step applies to all slave devices. If a different frequency is required for each slave, you must reprogram the clock register each time you access a different slave.
5. Set CLKEN = 1 to enable SPI_CLK.
6. Program the device configuration registers (SPIDCR1 and SPIDCR2) with the required clock phase, clock polarity, chip select pin polarity, and data delay settings for each slave.
7. Program the SPI module to generate an interrupt after a frame of characters has been transferred (FIRQ = 1) or after a single character has been transferred (CIRQ = 1), If you want to use interrupts. The FIRQ and CIRQ bits are located in SPICMD1.

Enable the SPI pins through the external bus selection register (EBSR). EBSR controls the pin multiplexing options of the device. Refer to the device-specific data manual for more information on EBSR.

13.2.13 Interrupt Support

13.2.13.1 Interrupt Events and Requests

The SPI module is capable of interrupting the CPU after every character transfer and after every frame transfer. The SPI interrupts are enabled through the FIRQ and CIRQ bits of the command register 1 (SPICMD1).

13.2.13.2 Interrupt Multiplexing

The SPI module generates a single interrupt to the CPU. The SPI interrupt is not multiplexed with any other interrupt source.

13.2.14 DMA Event Support

The SPI module does not generate any DMA events; it must be fully serviced by the CPU. Furthermore, none of the device DMA controllers have access to the SPI memory-mapped registers.

13.2.15 Power Management

The SPI module can be clock-gated to conserve power during periods of no activity. The SPI input clock can be turned off by using the peripheral clock gating configuration register (PCGCR). For detailed information on PCGCR, see the device-specific data manual.

13.2.16 Emulation Considerations

The SPI module is not interrupted by emulation events such as an emulation breakpoint.

13.3 Interfacing the SPI to an SPI EEPROM

The SPI module can be used to communicate with an SPI EEPROM. This section gives an example of interfacing the SPI a 256K-bit SPI EEPROM from Catalyst Semiconductor (CAT25C256). Software sequences for common tasks such as data block reads and writes are also included.

13.3.1 Operational Description

The SPI module is totally controlled by the CPU. The SPI initiates transfers with SPI devices when the CPU writes to the SPICMD2 register. The SPI can interrupt the CPU when it has completed a character transfer and/or when it has completed a frame transfer. Alternatively, the CPU can poll the SPI status registers. The SPI cannot generate DMA events; therefore, it is the task of the CPU to move data to and from the SPI data registers.

Communication with SPI EEPROM is carried out completely by the SPI module. The SPI module drives the chip select pin of the memory device and allows the clocks to shift data in and out.

The SPI clock is derived from the chip system clock. The clock divider bits (CLKDV) of SPICDR are used to configure the frequency of the SPI clock (SPI_CLK) as follows:

SPI_CLK frequency = SPI module input clock / (CLKDV + 1)

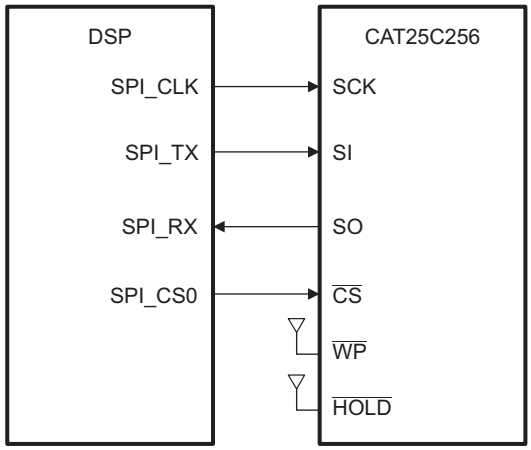
The number of characters sent or received by the SPI module is specified through the FLEN bits of SPICMD1. During communication with a SPI device, the SPI module will assert the chip select pin until all the characters specified through FLEN have been transferred.

13.3.2 Hardware Interface

Figure 13-12 shows the required hardware interface. Please note the following:

- The $\overline{WP}$ pin is shown as tied high (write-protect feature disabled); however, you can choose to tie the $\overline{WP}$ pin high or low depending on whether or not you want to use the write-protect feature of the CAT25C256 device.
- The $\overline{HOLD}$ pin is not used in this example and is pulled high.
- The example described in this section assumes chip-select 0 is used; however, you can use any other chip select.

Figure 13-12. Hardware Interface



13.3.3 SW Configuration

The following sections describe the software sequence for common tasks such as data block writes and data block reads.

13.3.3.1 Basic Initialization

The following procedure describes the basic steps required to setup the SPI module for communication with the CAT25C256 device.

1. Ensure the SPI is out reset by setting SPI_RST=0 in the peripheral reset control register (PRCR). See the device-specific data manual for more information on PRCR.
2. Enable the SPI input clock by setting SPICG to 0 in the peripheral clock gating configuration register (PCGCR1). See the device-specific data manual for more information on PCGCR1.
3. Set CLKEN = 0 to disable SPI_CLK. The CLKEN bit is part of SPICCR .
4. Set CLKDV in SPICDR to provide the appropriate SPI_CLK frequency. Note that for proper device operation, CLKDV must be greater than or equal to 3 .
5. Set CLKEN = 1 to enable SPI_CLK.
6. The CAT25C256 device latches input data on the rising edge of the clock and shifts out data on the falling edge of the clock. Therefore, the SPI must be programmed to shift out data on the falling edge of the clock and shift in data on the rising edge of the clock. According to Table 13-2, this corresponds to SPI mode 0. To enable SPI mode 0, set CKP0 = 0 and CKPH0 = 0 in SPIDCR1. Also, the data delay field should be programmed such that the chip select setup requirement of the CAT25C256 is met. A 1-bit data delay is recommended when running the SPI_CLK at frequencies greater than 2.5 MHz; otherwise a 0-bit data delay can be used.
7. Program the SPI module to generate an interrupt after a frame of characters has been transferred (FIRQ = 1) or after a single character has been transferred (CIRQ = 1), If you want to use interrupts. The FIRQ and CIRQ bits are located in SPICMD1. This example assumes interrupts are not used.

8. Enable the SPI pins through the external bus selection register (EBSR). EBSR controls the pin multiplexing options of the device. The SPI pins are multiplexed with other peripherals on the device and the pin multiplexing configuration you choose will depend on your use-case scenario. Refer to the device-specific data manual for more information on EBSR.

13.3.3.2 Reading the SPI EEPROM Status Register

The status register of the CAT25C256 device provides useful information including the write status and write protect feature status. The SPI module can be easily used to read the contents of this register by following these steps:

1. Ensure the SPI is not busy with another transfer by polling the CC and BUSY bits in SPISTAT1.
2. Reading the EEPROM status register requires 8 clock cycles for the read status register command (RDSR) and eight cycles for the actual data. Therefore, set FLEN = 1 (2 character transfer) in SPICMD1.
3. Load the opcode for the read status register command (05h) into the upper byte of SPIDAT2.
4. Write to the command register (SPICMD2) to start the SPI write. The value you write SPICMD2 must set CSNUM = 0 (chip-select 0 active), CLEN = 7 (8-bit character length), and CMD = 10b (write).
5. Poll SPISTAT1 until CC = 1 or wait for the character complete interrupt (CIRQ).
6. Poll SPISTAT1 until BUSY = 0. This ensures that a character is not being transferred.
7. Prepare SPIDAT1 and SPIDAT2 for data reception by loading both registers with 0000h.
8. Write to SPICMD2 to start an SPI read. The value you write SPICMD2 must set CSNUM = 0 (chip-select 0 active), CLEN = 7 (8-bit character length), and CMD = 01b (read).
9. Poll SPISTAT1 until CC = 1 or wait for the character complete interrupt (CIRQ).
10. Poll SPISTAT1 until BUSY = 0. This ensures that a character is not being transferred.
11. Read the value shifted out by the SPI EEPROM from the lower byte of SPIDAT1 (mask lower byte in data read from SPIDAT1).

13.3.3.3 Enabling and Disabling Writes

Before writing data to the CAT25C256 device, writes must be enabled. Similarly, to prevent inadvertent writes, writes should be disabled. Writes can be enabled by sending the write enable command (WREN) and writes can be disabled by sending the write disable command (WRDI). Both of these commands can be easily sent to the SPI EEPROM using the SPI module.

1. Ensure the SPI is not busy with another transfer by polling the CC and BUSY bits in SPISTAT1.
2. Sending the WREN or WRDI command to the EEPROM requires a total of 8 clock cycles. Therefore, set FLEN = 0 (1 character transfer) in SPICMD1.
3. Load the opcode for either WREN (06h) or WRDI (04h) into the upper byte of SPIDAT2.
4. Write to the command register (SPICMD2) to start the SPI write. The value you write SPICMD2 must set CSNUM = 0 (chip-select 0 active), CLEN = 7 (8-bit character length), and CMD = 10b (write).
5. Poll SPISTAT1 until CC = 1 or wait for the character complete interrupt (CIRQ).
6. Poll SPISTAT1 until BUSY = 0. This ensures that a character is not being transferred.

You can read the EEPROM status register to verify the WREN or WRDI commands were received successfully. See Section 13.3.3.2 for more details on reading the status register of the SPI EEPROM.

13.3.3.4 Writing a Block of Data to a SPI EEPROM

Once the CAT25C256 device has been configured to accept writes, you can use the write data memory command (WRITE) to program the contents of the device. Please note that the CAT25C256 device allows for a page write sequence in which you are allowed to write up to 64 bytes (a page) while the EEPROM automatically increments its address counter. The only restriction during page writes is that page boundaries must not be crossed. If the edge of a page boundary is reached, the address counter rolls over to the beginning of the page. The following steps outline the procedure for conducting a page write to the SPI EEPROM.

1. Configure the SPI EEPROM to allow writes through the use of the WREN command. See Section 13.3.3.3 for more details.
2. Ensure the SPI is not busy with another transfer by polling the CC and BUSY bits in SPISTAT1.
3. Writing a page of data to the EEPROM requires 8 clock cycles for the WRITE command, 16 clock cycles for the address, and eight cycles for each data byte. Therefore, set FLEN = 1 + 2 + N – 1, where N is the number of data bytes to be written. For example, if you want to write 64 data bytes, set FLEN = 66.
4. Load the opcode for the WRITE command (02h) into the upper byte of SPIDAT2.
5. Write to the command register (SPICMD2) to start the SPI write. The value you write SPICMD2 must set CSNUM = 0 (chip-select 0 active), CLEN = 7 (8-bit character length), and CMD = 10b (write).
6. Poll SPISTAT1 until CC = 1 or wait for the character complete interrupt (CIRQ).
7. Poll SPISTAT1 until BUSY = 0. This ensures that a character is not being transferred.
8. Load the most-significant byte of the SPI EEPROM address into the upper byte of SPIDAT2.
9. Write to the command register (SPICMD2) to start the SPI write. The value you write SPICMD2 must set CSNUM = 0 (chip-select 0 active), CLEN = 7 (8-bit character length), and CMD = 10b (write).
10. Poll SPISTAT1 until CC = 1 or wait for the character complete interrupt (CIRQ).
11. Poll SPISTAT1 until BUSY = 0. This ensures that a character is not being transferred.
12. Load the least-significant byte of the SPI EEPROM address into the upper byte of SPIDAT2.
13. Write to the command register (SPICMD2) to start the SPI write. The value you write SPICMD2 must set CSNUM = 0 (chip-select 0 active), CLEN = 7 (8-bit character length), and CMD = 10b (write).
14. Poll SPISTAT1 until CC = 1 or wait for the character complete interrupt (CIRQ).
15. Poll SPISTAT1 until BUSY = 0. This ensures that a character is not being transferred.
16. Load the data byte to be written to the upper byte of SPIDAT2. The SPI module will shift data out starting with the most-significant bit of SPIDAT2.
17. Write to SPICMD2 to start an SPI write. The value you write SPICMD2 must set CSNUM = 0 (chip-select 0 active), CLEN = 7 (8-bit character length), and CMD = 10b (write).
18. Poll SPISTAT1 until CC = 1 or wait for the character complete interrupt (CIRQ).
19. Poll SPISTAT1 until BUSY = 0. This ensures that a character is not being transferred.
20. Repeat steps 16 through 19 for the remainder of the bytes.

At the end of this sequence, you can poll the $\overline{\text{RDY}}$ bit in the SPI EEPROM status register (see Section 13.3.3.2) to determine when the EEPROM has completed the writes. At that time, you can restart this sequence to write more bytes to the EEPROM at a different address.

13.3.3.5 Reading a Block of Data from a SPI EEPROM

Unlike a page write, the CAT25C256 device allows you to read the entire memory contents without regard for page boundaries. The read data from memory command (READ) is used to read data from the SPI EEPROM. The following steps outline the procedure required to read a block of data from the memory device.

1. Ensure the SPI is not busy with another transfer by polling the CC and BUSY bits in SPISTAT1.
2. Reading a block of data from the EEPROM requires 8 clock cycles for the READ command, 16 clock cycles for the address, and eight cycles for each data byte. Therefore, set FLEN = 1 + 2 + N – 1, where N is the number of bytes to be read. For example, if you want to read 64 bytes, set FLEN = 66.
3. Load the opcode for the READ command (03h) into the upper byte of SPIDAT2.
4. Write to the command register (SPICMD2) to start the SPI write. The value you write SPICMD2 must set CSNUM = 0 (chip-select 0 active), CLEN = 7 (8-bit character length), and CMD = 10b (write).
5. Poll SPISTAT1 until CC = 1 or wait for the character complete interrupt (CIRQ).
6. Poll SPISTAT1 until BUSY = 0. This ensures that a character is not being transferred.
7. Load the most-significant byte of the SPI EEPROM address into the upper byte of SPIDAT2.
8. Write to the command register (SPICMD2) to start the SPI write. The value you write SPICMD2 must set CSNUM = 0 (chip-select 0 active), CLEN = 7 (8-bit character length), and CMD = 10b (write).
9. Poll SPISTAT1 until CC = 1 or wait for the character complete interrupt (CIRQ).
10. Poll SPISTAT1 until BUSY = 0. This ensures that a character is not being transferred.
11. Load the least-significant byte of the SPI EEPROM address into the upper byte of SPIDAT2.
12. Write to the command register (SPICMD2) to start the SPI write. The value you write SPICMD2 must set CSNUM = 0 (chip-select 0 active), CLEN = 7 (8-bit character length), and CMD = 10b (write).
13. Poll SPISTAT1 until CC = 1 or wait for the character complete interrupt (CIRQ).
14. Poll SPISTAT1 until BUSY = 0. This ensures that a character is not being transferred.
15. Prepare SPIDAT1 and SPIDAT2 for data reception by loading both registers with 0000h.
16. Write to SPICMD2 to start an SPI read. The value you write SPICMD2 must set CSNUM = 0 (chip-select 0 active), CLEN = 7 (8-bit character length), and CMD = 01b (read).
17. Poll SPISTAT1 until CC = 1 or wait for the character complete interrupt (CIRQ).
18. Poll SPISTAT1 until BUSY = 0. This ensures that a character is not being transferred.
19. Read the value shifted out by the SPI EEPROM from the lower byte of (mask lower byte in data read from SPIDAT1) SPIDAT1.
20. Repeat steps 15 through 19 for the remainder of the bytes.

13.4 SPI Registers

Table 13-4 lists the memory-mapped registers for the SPI. All register offset addresses not listed in Table 13-4 should be considered as reserved locations and the register contents should not be modified.

Table 13-4. SPI REGISTERS

CPU Word Address	Acronym	Register Name	Section
3000h	SPICDR	Clock Divider Register	Section 13.4.1
3001h	SPICCR	Clock Control Register	Section 13.4.2
3002h	SPIDCR1	Device Configuration Register 1	Section 13.4.3
3003h	SPIDCR2	Device Configuration Register 2	Section 13.4.4
3004h	SPICMD1	Command Register 1	Section 13.4.5
3005h	SPICMD2	Command Register 2	Section 13.4.6
3006h	SPISTAT1	Status Register 1	Section 13.4.7
3007h	SPISTAT2	Status Register 2	Section 13.4.8
3008h	SPIDAT1	Data Register 1	Section 13.4.9

Table 13-4. SPI REGISTERS (continued)

CPU Word Address	Acronym	Register Name	Section
3009h	SPIDAT2	Data Register 2	Section 13.4.10

13.4.1 SPICDR Register (offset = 3000h) [reset = 0h]

SPICDR is shown in Figure 13-13 and described in Table 13-5.

The clock divider register (SPICDR) specifies the clock divider value for the SPI input clock.

Figure 13-13. SPICDR Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 13-5. SPICDR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	CLKDV	RW	0h	Clock divider bits. The SPI input clock is divided down to generate the SPI interface clock (SPI_CLK). You can specify the divider value through the CLKDV bits. The frequency SPI_CLK is: SPI_CLK frequency = (SPI input clock frequency) / (CLKDV + 1) The duty cycle of SPI_CLK depends on the value of CLKDV + 1. When CLKDV + 1 is odd, the resulting duty cycle is approximately 33%. When CLKDV + 1 is even, the duty cycle is approximately 50%. NOTE: CLKDV must be set to a value greater than or equal to 2 such that the frequency of SPI_CLK is at least three times slower than the frequency of the SPI input clock.

13.4.2 SPICCR Register (offset = 3001h) [reset = 0h]

SPICCR is shown in Figure 13-14 and described in Table 13-6.

The clock control register (SPICCR) is used to enable the clock output and to initiate a soft reset to the SPI module.

Figure 13-14. SPICCR Register

15	14	13	12	11	10	9	8
CLKEN	RST	Reserved					
RW-0h	W-0h	R-0h					

7	6	5	4	3	2	1	0
Reserved							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 13-6. SPICCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	CLKEN	RW	0h	Clock enable bit. This bit is used to enable and disable the SPI interface clock (SPI_CLK). 0x0 = SPI_CLK is disabled and held at the logic value specified by the clock polarity bit of SPIDC. 0x1 = SPI_CLK is enabled.
14	RST	W	0h	Soft reset bit. Writing a 1 to this bit will reset the SPI module. This bit is self-clearing and will deactivate the soft reset on the next SPI input clock cycle after this bit is set to 1. 0x0 = Soft reset is released. 0x1 = Soft reset is asserted.
13-0	Reserved	R	0h	Reserved

13.4.3 SPIDCR1 Register (offset = 3002h) [reset = 0h]

SPIDCR1 is shown in Figure 13-15 and described in Table 13-7.

The device configuration registers (SPIDCR1 and SPIDCR2) are used to specify the clock phase, clock polarity, and data delay for each SPI slave connected to the SPI chip select pins.

Figure 13-15. SPIDCR1 Register

15	14	13	12	11	10	9	8
Reserved			DD1		CKPH1	CSP1	CKP1
R-0h			RW-0h		RW-0h	RW-0h	RW-0h
7	6	5	4	3	2	1	0
Reserved			DD0		CKPH0	CSP0	CKP0
R-0h			RW-0h		RW-0h	RW-0h	RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 13-7. SPIDCR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved
12-11	DD1	RW	0h	Data delay for chip select 1 pin (SPI_CS1). 0x0 = Data is output on the same cycle as the SPI_CS1 goes active. 0x1 = Data is output one SPI_CLK cycle after the SPI_CS1 goes active. 0x2 = Data is output two SPI_CLK cycles after the SPI_CS1 goes active. 0x3 = Data is output three SPI_CLK cycles after the SPI_CS1 goes active.
10	CKPH1	RW	0h	Clock phase for chip select 1 pin (SPI_CS1). The clock phase bit, in conjunction with the clock polarity bit (CKP1), controls the clock-data relationship between master and slave. 0x0 = When CKP1 = 0, data shifted out on falling edge, input captured on rising edge. When CKP1 = 1, data shifted out on rising edge, input captured on falling edge. 0x1 = When CKP1 = 0, data shifted out on rising edge, input captured on falling edge. When CKP1 = 1, data shifted out on falling edge, input captured on rising edge.
9	CSP1	RW	0h	Polarity for chip select 1 pin (SPI_CS1). 0x0 = Active low. 0x1 = Active high.
8	CKP1	RW	0h	Clock polarity inactive state for the clock pin during accesses to chip select 1. 0x0 = When data is not being transferred, a steady state low value is produced at the SCK pin. 0x1 = When data is not being transferred, a steady state high value is produced at the SCK pin.
7-5	Reserved	R	0h	Reserved
4-3	DD0	RW	0h	Data delay for chip select 0 pin (SPI_CS0). 0x0 = Data is output on the same cycle as the SPI_CS0 goes active. 0x1 = Data is output one SPI_CLK cycle after the SPI_CS0 goes active. 0x2 = Data is output two SPI_CLK cycles after the SPI_CS0 goes active. 0x3 = Data is output three SPI_CLK cycles after the SPI_CS0 goes active.

Table 13-7. SPIDCR1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	CKPH0	RW	0h	Clock phase for chip select 0 pin (SPI_CS0). The clock phase bit, in conjunction with the clock polarity bit (CKP0), controls the clock-data relationship between master and slave. 0x0 = When CKP0 = 0, data shifted out on falling edge, input captured on rising edge. When CKP0 = 1, data shifted out on rising edge, input captured on falling edge. 0x1 = When CKP0 = 0, data shifted out on rising edge, input captured on falling edge. When CKP0 = 1, data shifted out on falling edge, input captured on rising edge.
1	CSP0	RW	0h	Polarity for chip select 0 pin (SPI_CS0). 0x0 = Active low. 0x1 = Active high.
0	CKP0	RW	0h	Clock polarity inactive state for the clock pin during accesses to chip select 0. 0x0 = When data is not being transferred, a steady state low value is produced at the SCK pin. 0x1 = When data is not being transferred, a steady state high value is produced at the SCK pin.

13.4.4 SPIDCR2 Register (offset = 3003h) [reset = 0h]

SPIDCR2 is shown in Figure 13-16 and described in Table 13-8.

The device configuration registers (SPIDCR1 and SPIDCR2) are used to specify the clock phase, clock polarity, and data delay for each SPI slave connected to the SPI chip select pins.

Figure 13-16. SPIDCR2 Register

15	14	13	12	11	10	9	8
LPBK	Reserved		DD3		CKPH3	CSP3	CKP3
RW-0h	R-0h		RW-0h		RW-0h	RW-0h	RW-0h
7	6	5	4	3	2	1	0
Reserved			DD2		CKPH2	CSP2	CKP2
R-0h			RW-0h		RW-0h	RW-0h	RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 13-8. SPIDCR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	LPBK	RW	0h	Loopback mode enable bit. 0x0 = Loopback mode is disabled. 0x1 = Loopback mode is enabled.
14-13	Reserved	R	0h	Reserved
12-11	DD3	RW	0h	Data delay for chip select 3 pin (SPI_CS3). 0x0 = Data is output on the same cycle as the SPI_CS3 goes active. 0x1 = Data is output one SPI_CLK cycle after the SPI_CS3 goes active. 0x2 = Data is output two SPI_CLK cycles after the SPI_CS3 goes active. 0x3 = Data is output three SPI_CLK cycles after the SPI_CS3 goes active.
10	CKPH3	RW	0h	Clock phase for chip select 3 pin (SPI_CS3). The clock phase bit, in conjunction with the clock polarity bit (CKP3), controls the clock-data relationship between master and slave. 0x0 = When CKP3 = 0, data shifted out on falling edge, input captured on rising edge. When CKP3 = 1, data shifted out on rising edge, input captured on falling edge. 0x1 = When CKP3 = 0, data shifted out on rising edge, input captured on falling edge. When CKP3 = 1, data shifted out on falling edge, input captured on rising edge.
9	CSP3	RW	0h	Polarity for chip select 3 pin (SPI_CS3). 0x0 = Active low. 0x1 = Active high.
8	CKP3	RW	0h	Clock polarity inactive state for the clock pin during accesses to chip select 3. 0x0 = When data is not being transferred, a steady state low value is produced at the SCK pin. 0x1 = When data is not being transferred, a steady state high value is produced at the SCK pin.
7-5	Reserved	R	0h	Reserved
4-3	DD2	RW	0h	Data delay for chip select 2 pin (SPI_CS2). 0x0 = Data is output on the same cycle as the SPI_CS2 goes active. 0x1 = Data is output one SPI_CLK cycle after the SPI_CS2 goes active. 0x2 = Data is output two SPI_CLK cycles after the SPI_CS2 goes active. 0x3 = Data is output three SPI_CLK cycles after the SPI_CS2 goes active.

Table 13-8. SPIDCR2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	CKPH2	RW	0h	Clock phase for chip select 2 pin (SPI_CS2). The clock phase bit, in conjunction with the clock polarity bit (CKP2), controls the clock-data relationship between master and slave. 0x0 = When CKP2 = 0, data shifted out on falling edge, input captured on rising edge. When CKP2 = 1, data shifted out on rising edge, input captured on falling edge. 0x1 = When CKP2 = 0, data shifted out on rising edge, input captured on falling edge. When CKP2 = 1, data shifted out on falling edge, input captured on rising edge.
1	CSP2	RW	0h	Polarity for chip select 2 pin (SPI_CS2). 0x0 = Active low. 0x1 = Active high.
0	CKP2	RW	0h	Clock polarity inactive state for the clock pin during accesses to chip select 2. 0x0 = When data is not being transferred, a steady state low value is produced at the SCK pin. 0x1 = When data is not being transferred, a steady state high value is produced at the SCK pin.

13.4.5 SPICMD1 Register (offset = 3004h) [reset = 0h]

SPICMD1 is shown in Figure 13-17 and described in Table 13-9.

Used to enable character and frame complete interrupts and specify the number of characters in a frame.

Figure 13-17. SPICMD1 Register

15	14	13	12	11	10	9	8
FIRQ	CIRQ	Reserved		FLEN			
R-0h	RW-0h	R-0h		RW-0h			

7	6	5	4	3	2	1	0
FLEN							
RW-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 13-9. SPICMD1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	FIRQ	R	0h	Frame count interrupt enable. 0x0 = No interrupt generated at the end of the frame count. 0x1 = Interrupt generated at the end of the frame count.
14	CIRQ	RW	0h	Character interrupt enable. 0x0 = No interrupt generated at the end of the character transfer. 0x1 = Interrupt generated at the end of the character transfer.
13-12	Reserved	R	0h	Reserved
11-0	FLEN	RW	0h	Frame length bits. These bits are used to specify the length of entire transfer. The total number of characters transferred equals FLEN + 1. For example, if FLEN = 63, a frame consists of a total of 64 characters.

13.4.6 SPICMD2 Register (offset = 3005h) [reset = 0h]

SPICMD2 is shown in Figure 13-18 and described in Table 13-10.

The command to use (read or write), character length, and chip select pin used during SPI transfers are specified through SPICMD2. Writing to SPICMD2 will cause the SPI to execute the command specified by the transfer command bits (CMD).

Figure 13-18. SPICMD2 Register

15	14	13	12	11	10	9	8
Reserved		CSNUM		Reserved			
R-0h		RW-0h		R-0h			

7	6	5	4	3	2	1	0
CLEN					Reserved	CMD	
RW-0h					R-0h	RW-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 13-10. SPICMD2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	Reserved	R	0h	Reserved
13-12	CSNUM	RW	0h	Device select. Sets the active chip select for the transfer. 0x0 = Chip select 0 is active. 0x1 = Chip select 1 is active. 0x2 = Chip select 2 is active. 0x3 = Chip select 3 is active.
11-8	Reserved	R	0h	Reserved
7-3	CLEN	RW	0h	Character length. Sets the transfer size of the individual transfer elements from 1 to 32 bits. The character length is set to CLEN + 1. For example, if CLEN = 7, the character length is set to 8 bits.
2	Reserved	R	0h	Reserved
1-0	CMD	RW	0h	Transfer command bits. These bits specify the type of transaction being used. 0x0 = Reserved. 0x1 = Read. 0x2 = Write. 0x3 = Reserved.

13.4.7 SPISTAT1 Register (offset = 3006h) [reset = 0h]

SPISTAT1 is shown in Figure 13-19 and described in Table 13-11.

The status registers (SPISTAT1 and SPISTAT2) contain indicators to allow you to monitor the progression of a frame transfer.

Figure 13-19. SPISTAT1 Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved					FC	CC	BSY
R-0h					R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 13-11. SPISTAT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-3	Reserved	R	0h	Reserved
2	FC	R	0h	Frame complete. This bit is set after all the requested characters have been transferred. This bit is reset when SPISTATU is read and when SPICMDU is written to. 0x0 = Transfer is not complete. 0x1 = All characters have been transferred.
1	CC	R	0h	Character complete. This bit is set after each character transfer is completed. This bit is reset when SPISTATU is read and when SPICMDU is written to. 0x0 = Character transfer is not complete. 0x1 = Character transfer is complete.
0	BSY	R	0h	Busy bit. This bit is set during an active character transfer. Between characters this bit will be cleared to signal that the data registers can be accessed. 0x0 = Idle, no character transfers in progress. 0x1 = Active, a character transfer is in progress.

13.4.8 SPISTAT2 Register (offset = 3007h) [reset = 0h]

SPISTAT2 is shown in Figure 13-20 and described in Table 13-12.

The status registers (SPISTAT1 and SPISTAT2) contain indicators to allow you to monitor the progression of a frame transfer.

Figure 13-20. SPISTAT2 Register

15	14	13	12	11	10	9	8
Reserved				CCNT			
R-0h				R-0h			

7	6	5	4	3	2	1	0
CCNT							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 13-12. SPISTAT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	Reserved	R	0h	Reserved
11-0	CCNT	R	0h	Character count. These bits reflect the total number of characters transferred minus one. For example, when CCNT = 63, a total of 64 characters have been transferred.

13.4.9 SPIDAT1 Register (offset = 3008h) [reset = 0h]

SPIDAT1 is shown in Figure 13-21 and described in Table 13-13.

The data registers (SPIDAT1 and SPIDAT2) are treated as a 32-bit shift register. Data received by the SPI is shifted into the least-significant bit of SPIDAT1 and the contents of both registers are shifted to the left. Similarly, data transferred by the SPI is shifted out of the most-significant bit of SPIDAT2 and the contents of both registers are shifted to the left.

Figure 13-21. SPIDAT1 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 13-13. SPIDAT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	RW	0h	Low part of data during read and write operations.

13.4.10 SPIDAT2 Register (offset = 3009h) [reset = 0h]

SPIDAT2 is shown in Figure 13-22 and described in Table 13-14.

The data registers (SPIDAT1 and SPIDAT2) are treated as a 32-bit shift register. Data received by the SPI is shifted into the least-significant bit of SPIDAT1 and the contents of both registers are shifted to the left. Similarly, data transferred by the SPI is shifted out of the most-significant bit of SPIDAT2 and the contents of both registers are shifted to the left.

Figure 13-22. SPIDAT2 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 13-14. SPIDAT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	RW	0h	High part of data during read and write operations.

32-Bit Timer/Watchdog Timer

Topic **Page**

14.1 Introduction

The following information describes the operation of the three 32-bit software programmable timers in the digital signal process (DSP). Each timer can be used as a general-purpose (GP) timer. Timer2 also contains a 16-bit Watchdog (WD) timer.

14.1.1 Purpose of the Timers

General purpose timers are typically used to provide interrupts to the CPU to schedule periodic tasks or a delayed task. The general-purpose (GP) timers are 32-bit timers with a 13-bit prescaler that divides the source clock and uses this scaled value as a reference clock. These timers can be used to generate periodic interrupts.

Watchdog timers are used to reset the CPU in the event of a deadlocked state, such as a non-exiting code loop. This device includes a timer that functions as a timer or a watchdog timer simultaneously, Timer2. This watchdog timer is a 32-bit timer composed of a 16-bit counter with a 16-bit prescaler that divides the CPU system clock and uses this scaled value as a reference clock. The programmer must continuously service the watchdog timer to prevent it from resetting the device. Once the code fails to service the watchdog timer due to a deadlock or non-exiting code loop, the watchdog expires and resets the device.

14.1.2 Features

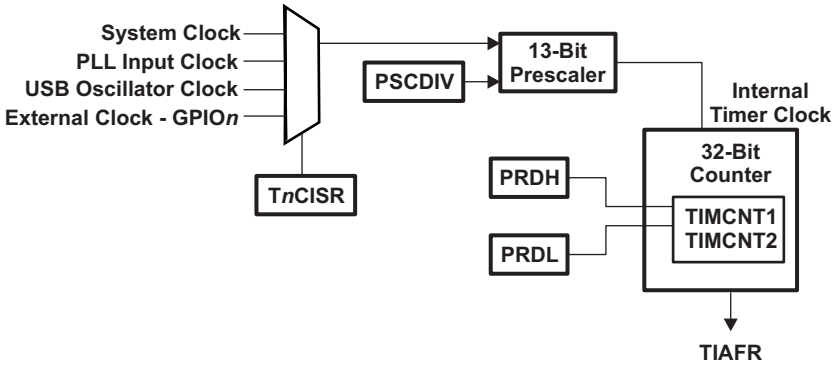
32-bit Timers:

- 32-bit programmable count down timer
- 13-bit prescaler divider
- Timer Modes:
  - 32-bit general-purpose timer
  - 16-bit Watchdog Timer (Timer2 only)
- Auto reload option
- Wide range of clock sources: CPU system clock (PLL output), USB oscillator, PLL input clock, or GPIOn. Each timer may use a different clock source.
- Generates a single interrupt to the CPU. The interrupt is individually latched to determine which timer triggered the interrupt
- Interrupt can be used for DMA event
- Option to generate a non-maskable interrupt to the CPU

14.1.3 Functional Timer Block Diagram

A block diagram of the timer is shown in Figure 14-1. Detailed information about the architecture and operation of the watchdog timer is in Section 14.3.

Figure 14-1. Architecture and Operation of GP Timers



NOTE: The timer architecture also includes logic, not shown, to synchronize parts of the timer circuit running on the internal timer clock with an asynchronous input clock source.

14.2 General-Purpose Timer

All three timers (Timer0, Timer1, and Timer2) can be used as GP timers. Timer2 can also be used as a WD timer. To use the WD timer see Section 14.3.

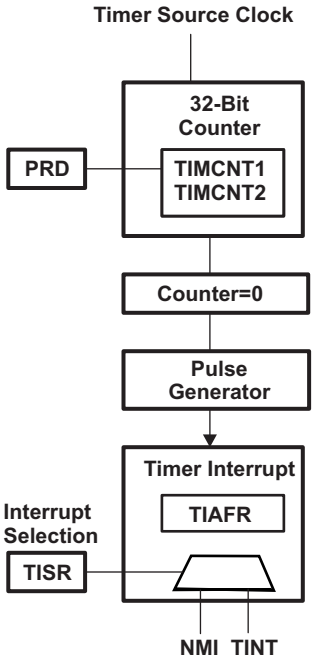
14.2.1 General-Purpose Timer Clock Control

The clock source to the GP timer and to the WD timer is driven by the selected input clock, CPU system clock (PLL output), USB oscillator, PLL input clock, or GPIOn. This clock source determines the speed of the timer since the timer counts down in units of source clock cycles. The source clock for the GP timer can be divided down by a 13-bit prescaler and uses this scaled value as the reference clock of the timer. Each GP timer has its own 13-bit prescaler. When determining the period and prescaler setting for the timer, choose the desired period in units of source clock cycles.

14.2.2 Using the 32-bit General Purpose Timer

The general-purpose timers consist of a 32-bit timer with a 13-bit prescaler. Figure 14-2 shows a high-level diagram of the timer.

Figure 14-2. 32-Bit GP Timer With a 13-Bit Prescaler



Each GP timer has a count register (TIMCNT*n*) which consists of two 16-bit words (TIMCNT1 and TIMCNT2) and a period register (TIMPRD*n*) which also consists of two 16-bit words (TIMPRD1 and TIMPRD2). When the timer is set to start the contents of the TIMPRD*n* register is loaded into the TIMCNT register and begins to count down. A timer control register (TCR) controls the operation of the timer.

The Prescaler Divider (PSCDIV), located in the TCR, is used to divide the timer source clock.

When the START bit is set to 1 in the TCR, the contents of the Timer Period Registers (TIMPRD1 and TIMPRD2) are loaded into the concatenated Timer Counter Registers (TIMCNT1 and TIMCNT2) and the timer starts to count down with every cycle of the prescale divided clock. When TIMCNT1 and TIMCNT2 reach 0, the timer sets an interrupt flag, TIAFR, and sends an interrupt request to the CPU. Two types of interrupt requests are possible by setting the timer interrupt selection register (TISR): a maskable timer interrupt request (TINT) along with a DMA Event to the four DMAs, or a higher priority non-maskable interrupt (NMI) to the CPU.

The timer can be configured in auto-reload mode by setting the AUTORELOAD bit in the TCR. In this mode, the timer counter is reloaded with the timer period register, TIMPRD*n*, when the timer counter reaches 0 and the timer re-starts its count down.

NOTE: The synchronization blocks within the Timer affect the occurrence of the very first interrupt from when the START bit is set to 1 in TCR. Due to extra timer input source clock cycles needed to synchronize the timer logic with the asynchronous input clock source, the first interrupt will be delayed by 4 or 5 timer input source clock cycles. All subsequent interrupts will occur as expected from the programmed values.

14.3 Watchdog Timer

This section describes the timer in the watchdog (WD) timer mode. Only Timer2 can be used as a Watchdog timer. This timer can also be used as general-purpose timer. To use it as a general-purpose timer, see Section 14.2.

14.3.1 Watchdog Timer Function

The watchdog timer function consists of a 16-bit main counter preceded by prescaler. The combination of the 16-bit counter with the 16-bit prescaler allows for a maximum countdown value of 4,294,967,296. As a watchdog, the timer can be used to prevent system lockup when the software becomes deadlocked or trapped in a loop with no controlled exit. When the counter expires, a hardware reset is generated. To prevent the hardware reset from being generated, the watchdog must be serviced periodically before the counter expires. The service reinitializes the counter to its starting value and starts counting down again.

After a hardware reset occurs, the watchdog timer is disabled. To configure the watchdog timer, each control register has a corresponding lock register that requires a specific key sequence (with two or three keys written in order) to unlock that control register. Once the appropriate key sequence has been written to the lock register, then the corresponding control register is unlocked and can be written. After the control register is written, it is automatically locked. To write the register, the sequence must be repeated to unlock the register. If the wrong value is written for any of the keys, then it is necessary to restart at the first key. Each control and lock register pair has its own state machine to keep track of the key sequence. Therefore, it is possible to write the 1st key to each of the 4 lock registers, and then proceed to write the 2nd key to each of the 4 lock registers, and so on until all of the keys have been written. In other words, each control register must have it's specific keys written in order; but the CPU can perform other things while writing the keys in order. In the watchdog timer mode, the timer requires a periodic execution of this unlock/write sequence to reinitialize the watchdog counter to its starting value. Without this periodic servicing, the watchdog timer counter reaches zero and causes a watchdog timeout event.

When the timeout event occurs, the watchdog timer resets the entire chip with the exception of the RTC (real time clock).

14.3.2 Watchdog Timer Operation

The Watchdog (WD) timer function is enabled when:

- The Start Value and Prescale registers have been unlocked.
- The Start Value and Prescale registers have been programmed.
- The Watchdog Enable Lock register is unlocked.
- A 1 is written to bit 0 of the Watchdog Enable register.

To enable the watchdog timer, a certain sequence of events must be followed. First, to configure the watchdog timer, each control register has a lock register that requires a specific key sequence (with two or three keys written in the proper order) to unlock the corresponding control register. Table 14-1 details the unlock sequence for the watchdog lock registers:

Table 14-1. Unlock Sequence for the Watchdog Lock Registers

Address	Register	First Key Sequence	Second Key Sequence	Third Key Sequence
1880h	Watchdog Kick Lock Register	5555h	AAAAh	n/a
1884h	Watchdog Start Value Lock Register	6666h	BBBBh	n/a
1888h	Watchdog Enable Lock Register	7777h	CCCCh	DDDDh
188Ch	Watchdog Prescale Lock Register	5A5Ah	A5A5h	n/a

Once the Start Value and Prescale register have been unlocked and programmed, the Watchdog is enabled by unlocking the Enable Lock Register with a sequence of three words (7777h, CCCCh, and DDDDh) and writing a 1 on the EN bit (bit 0) of the WDEN register. The Watchdog starts counting down by unlocking the Kick Lock register and writing any value into the Kick register. At this time, the counter starts to count down . The counter will reach zero when the counter value and pre-scalar value are exhausted, consequently, triggering a hardware reset of the device. To prevent the hardware reset from occurring, the Kick Lock register must be unlocked and the Kick register bit 0 set to 1 to restart the countdown before the countdown is exhausted.

Once the Watchdog is enabled, it can be disabled with software by unlocking the Enable Lock Register with the three word sequence and setting the EN bit (bit 0) of the WDEN register to 0.

14.4 Reset Considerations

The timers will be reset upon a hardware reset.

When a hardware reset is asserted, all timer registers, including the TIAFR register, are set to their default values.

14.5 Interrupt Support

The general-purpose timer has a timer interrupt signal. The timer interrupt request is sent to the CPU when the main count register (TIMCNT1 and TIMCNT2) counts down to 0. The same interrupt signal is also routed to the DMAs and can be used as a DMA trigger event.

The interrupt for each timer can be configured as a timer interrupt (TINT) or as a non-maskable interrupt (NMI) to the CPU using the Timer Interrupt Selection Register (TISR).

The TIAFR latches each timer's interrupt signal when the timer counter expires. Using this register, the programmer can determine which of the three timers generated the timer interrupt because the bits in the TIAFR are OR'ed together and sent to the DSP as a single interrupt. Each timer interrupt flag needs to be cleared by the CPU with a write of "1" to the corresponding flag bit.

A false timer interrupt is triggered when the timer interrupt flag corresponding to a nonmaskable interrupt (NMI) timer interrupt is cleared. This NMI timer interrupt should have pre-empted a regular timer interrupt ISR and before the timer ISR has had a chance to clear the timer interrupt flag.

For example, if two timers are set up as Timer0 for TINT (timer interrupt) to CPU and Timer1 for NMI interrupt to CPU. The CPU receives TINT first and starts servicing TINT ISR. But before ISR can clear the Timer0 flag in the Timer Interrupt Aggregation (TIA) register, the CPU receives an NMI interrupt from Timer1. The NMI interrupt pre-empts TINT ISR and the CPU starts servicing the corresponding NMI ISR.

After the NMI ISR clears the Timer1 flag in the TIA register, another TINT interrupt is generated to the CPU due to the uncleared Timer0 interrupt flag. When NMI ISR returns to TINT ISR and TINT ISR clears the Timer0 flag in the TIA register and returns to main function. The CPU immediately starts servicing the second (false) TINT interrupt again, but the Timer0 interrupt flag is not set in the TIA register.

To work around the false interrupt, see the *TMS320C5517 Fixed-Point DSP Silicon Errata* [literature number SPRZ383].

14.6 Registers

14.6.1 TIMER 0 CONTROLLER Registers

Table 14-2 lists the memory-mapped registers for the TIMER 0 CONTROLLER. All register offset addresses not listed in Table 14-2 should be considered as reserved locations and the register contents should not be modified.

Table 14-2. TIMER 0 CONTROLLER REGISTERS

CPU Word Address	Acronym	Register Name	Section
1810h	T0CR	Timer 0 Control Register	Section 14.6.1.1
1812h	TIM0PRD1	Timer 0 Period Register 1	Section 14.6.1.2
1813h	TIM0PRD2	Timer 0 Period Register 2	Section 14.6.1.3
1814h	TIM0CNT1	Timer 0 Counter Register 1	Section 14.6.1.4
1815h	TIM0CNT2	Timer 0 Counter Register 2	Section 14.6.1.5
1816h	T0INSR	Timer 0 Input Selection Register	Section 14.6.1.6

14.6.1.1 T0CR Register (offset = 1810h) [reset = 0h]

T0CR is shown in Figure 14-3 and described in Table 14-3.

Timer 0 Control Register

Figure 14-3. T0CR Register

15	14	13	12	11	10	9	8
TIMEN	Reserved						
RW-0h	R-0h						

7	6	5	4	3	2	1	0
Reserved		PSCDIV				AUTORELOAD	START
R-0h		RW-0h				RW-0h	RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-3. T0CR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	TIMEN	RW	0h	Enables and disables the clocks for the counters to run. The TIMEN bit starts both the prescaler and the main counter. 0x0 = Timer 0 counters are disabled. 0x1 = Timer 0 counters are enabled and the prescaler and counter will begin to count down.
14-6	Reserved	R	0h	Reserved
5-2	PSCDIV	RW	0h	Prescaler divider. The range is: 0000 = divide by 2 to 1100 = divide by 8192.
1	AUTORELOAD	RW	0h	Automatically reloads the counter when it reaches 0. 0x0 = Auto-reload is disabled. 0x1 = Auto-reload is enabled.
0	START	RW	0h	When the START bit is written to, the bit loads and starts the counter. If the device is already in the process of counting down, and a "1" is written to this bit, the counter will be re-loaded with the start value and begin counting down again. When read, the bit will be 1 when the counter is counting down. The register will read 0 when the counter has stopped.

14.6.1.2 TIM0PRD1 Register (offset = 1812h) [reset = FFFFh]

TIM0PRD1 is shown in Figure 14-4 and described in Table 14-4.

Timer 0 Period Register 1

Figure 14-4. TIM0PRD1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRD1															
RW-FFFFh															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-4. TIM0PRD1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	PRD1	RW	FFFFh	The timer 0 period register is 32-bits wide. This is the LSW for the timer 0 period, value 0000 to FFFFh.

14.6.1.3 TIM0PRD2 Register (offset = 1813h) [reset = FFFFh]

TIM0PRD2 is shown in Figure 14-5 and described in Table 14-5.

Timer 0 Period Register 2

Figure 14-5. TIM0PRD2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRD2															
RW-FFFFh															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-5. TIM0PRD2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	PRD2	RW	FFFFh	The timer 0 period register is 32-bits wide. This is the MSW for the timer 0 period, value 0000 to FFFFh.

14.6.1.4 TIM0CNT1 Register (offset = 1814h) [reset = FFFFh]

TIM0CNT1 is shown in Figure 14-6 and described in Table 14-6.

Timer 0 Counter Register 1

Figure 14-6. TIM0CNT1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM1															
RW-FFFFh															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-6. TIM0CNT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TIM1	RW	FFFFh	The timer 0 counter is 32-bits wide. This is the LSW for the timer 0 counter, value 0000 to FFFFh.

14.6.1.5 TIM0CNT2 Register (offset = 1815h) [reset = FFFFh]

TIM0CNT2 is shown in Figure 14-7 and described in Table 14-7.

Timer 0 Counter Register 2

Figure 14-7. TIM0CNT2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM2															
RW-FFFFh															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-7. TIM0CNT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TIM2	RW	FFFFh	The timer 0 counter is 32-bits wide. This is the MSW for the timer 0 counter, value 0000 to FFFFh.

14.6.1.6 T0INSR Register (offset = 1816h) [reset = 0h]

T0INSR is shown in Figure 14-8 and described in Table 14-8.

Timer 0 Input Selection Register

Figure 14-8. T0INSR Register

15	14	13	12	11	10	9	8
TIMRST	Reserved						
RW-0h	R-0h						

7	6	5	4	3	2	1	0
Reserved					TINSEL		
R-0h					RW-0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-8. T0INSR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	TIMRST	RW	0h	Drives the corresponding timer 0 module reset signal low (asserted) or high (deasserted). 0x0 = Timer 0 counter reset is deasserted. Timer 0 module is out of reset. 0x1 = Timer 0 counter reset is asserted. Timer 0 module is in reset.
14-3	Reserved	R	0h	Reserved
2-0	TINSEL	RW	0h	Timer 0 Clock Input Source Selection. 000b = Timer 0 clock is driven by the System Clock (PLL output). 001b = Timer 0 clock is driven by GPIO0, S0. 010b = Timer 0 clock is driven by GPIO6, S10. 011b = Timer 0 clock is driven by GPIO12, PD[2]. 100b = Timer 0 clock is driven by GPIO18, PD[8]. 101b = Timer 0 clock is driven by GPIO24, A[18]. 110b = Timer 0 clock is driven by USB Oscillator. 111b = Timer 0 clock is driven by PLL input clock.

14.6.2 TIMER 1 CONTROLLER Registers

Table 14-9 lists the memory-mapped registers for the TIMER 1 CONTROLLER. All register offset addresses not listed in Table 14-9 should be considered as reserved locations and the register contents should not be modified.

Table 14-9. TIMER 1 CONTROLLER REGISTERS

CPU Word Address	Acronym	Register Name	Section
1850h	T1CR	Timer 1 Control Register	Section 14.6.2.1
1852h	TIM1PRD1	Timer 1 Period Register 2	Section 14.6.2.2
1853h	TIM1PRD2	Timer 1 Period Register 2	Section 14.6.2.3
1854h	TIM1CNT1	Timer 1 Counter Register 1	Section 14.6.2.4
1855h	TIM1CNT2	Timer 1 Counter Register 2	Section 14.6.2.5
1856h	T1INSR	Timer 1 Input Selection Register	Section 14.6.2.6

14.6.2.1 T1CR Register (offset = 1850h) [reset = 0h]

T1CR is shown in Figure 14-9 and described in Table 14-10.

Timer 1 Control Register

Figure 14-9. T1CR Register

15	14	13	12	11	10	9	8
TIMEN	Reserved						
RW-0h	R-0h						

7	6	5	4	3	2	1	0
Reserved		PSCDIV				AUTORELOAD	START
R-0h		RW-0h				RW-0h	RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-10. T1CR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	TIMEN	RW	0h	Enables and disables the clocks for the counters to run. The TIMEN bit starts both the prescaler and the main counter. 0x0 = Timer 1 counters are disabled. 0x1 = Timer 1 counters are enabled and the prescaler and counter will begin to count down.
14-6	Reserved	R	0h	Reserved
5-2	PSCDIV	RW	0h	Prescaler divider. The range is: 0000 = divide by 2 to 1100 = divide by 8192.
1	AUTORELOAD	RW	0h	Automatically reloads the counter when it reaches 0. 0x0 = Auto-reload is disabled. 0x1 = Auto-reload is enabled.
0	START	RW	0h	When the START bit is written to, the bit loads and starts the counter. If the device is already in the process of counting down, and a "1" is written to this bit, the counter will be re-loaded with the start value and begin counting down again. When read, the bit will be 1 when the counter is counting down. The register will read 0 when the counter has stopped.

14.6.2.2 TIM1PRD1 Register (offset = 1852h) [reset = FFFFh]

TIM1PRD1 is shown in Figure 14-10 and described in Table 14-11.

Timer 1 Period Register 2

Figure 14-10. TIM1PRD1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRD1															
RW-FFFFh															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-11. TIM1PRD1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	PRD1	RW	FFFFh	The timer 1 period register is 32-bits wide. This is the LSW for the timer 1 period, value 0000 to FFFFh.

14.6.2.3 TIM1PRD2 Register (offset = 1853h) [reset = FFFFh]

TIM1PRD2 is shown in Figure 14-11 and described in Table 14-12.

Timer 1 Period Register 2

Figure 14-11. TIM1PRD2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRD2															
RW-FFFFh															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-12. TIM1PRD2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	PRD2	RW	FFFFh	The timer 1 period register is 32-bits wide. This is the MSW for the timer period, value 0000 to FFFFh.

14.6.2.4 TIM1CNT1 Register (offset = 1854h) [reset = FFFFh]

TIM1CNT1 is shown in Figure 14-12 and described in Table 14-13.

Timer 1 Counter Register 1

Figure 14-12. TIM1CNT1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM1															
RW-FFFFh															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-13. TIM1CNT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TIM1	RW	FFFFh	The timer 1 counter is 32-bits wide. This is the LSW for the timer 1 counter, value 0000 to FFFFh.

14.6.2.5 TIM1CNT2 Register (offset = 1855h) [reset = FFFFh]

TIM1CNT2 is shown in Figure 14-13 and described in Table 14-14.

Timer 1 Counter Register 2

Figure 14-13. TIM1CNT2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM2															
RW-FFFFh															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-14. TIM1CNT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TIM2	RW	FFFFh	The timer 1 counter is 32-bits wide. This is the MSW for the timer 1 counter, value 0000 to FFFFh.

14.6.2.6 T1INSR Register (offset = 1856h) [reset = 0h]

T1INSR is shown in Figure 14-14 and described in Table 14-15.

Timer 1 Input Selection Register

Figure 14-14. T1INSR Register

15	14	13	12	11	10	9	8
TIMRST	Reserved						
RW-0h	R-0h						

7	6	5	4	3	2	1	0
Reserved					TINSEL		
R-0h					RW-0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-15. T1INSR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	TIMRST	RW	0h	Drives the corresponding timer 1 module reset signal low (asserted) or high (deasserted). 0x0 = Timer 1 counter reset is deasserted. Timer 1 module is out of reset. 0x1 = Timer 1 counter reset is asserted. Timer 1 module is in reset.
14-3	Reserved	R	0h	Reserved
2-0	TINSEL	RW	0h	Timer 1 Clock Input Source Selection. 000b = Timer 1 clock is driven by the System Clock (PLL output). 001b = Timer 1 clock is driven by GPIO1, S1. 010b = Timer 1 clock is driven by GPIO7, S11. 011b = Timer 1 clock is driven by GPIO13, PD[3]. 100b = Timer 1 clock is driven by GPIO19, PD[9]. 101b = Timer 1 clock is driven by GPIO25, A[19]. 110b = Timer 1 clock is driven by USB Oscillator. 111b = Timer 1 clock is driven by PLL input clock.

14.6.3 TIMER 2 CONTROLLER Registers

Table 14-16 lists the memory-mapped registers for the TIMER 2 CONTROLLER. All register offset addresses not listed in Table 14-16 should be considered as reserved locations and the register contents should not be modified.

Table 14-16. TIMER 2 CONTROLLER REGISTERS

CPU Word Address	Acronym	Register Name	Section
1890h	T2CR	Timer 2 Control Register	Section 14.6.3.1
1892h	TIM2PRD1	Timer 2 Period Register 1	Section 14.6.3.2
1893h	TIM2PRD2	Timer 2 Period Register 2	Section 14.6.3.3
1894h	TIM2CNT1	Timer 2 Counter Register 1	Section 14.6.3.4
1895h	TIM2CNT2	Timer 2 Counter Register 2	Section 14.6.3.5
1896h	T2INSR	Timer 2 Input Selection Register	Section 14.6.3.6

14.6.3.1 T2CR Register (offset = 1890h) [reset = 0h]

T2CR is shown in Figure 14-15 and described in Table 14-17.

Timer 2 Control Register

Figure 14-15. T2CR Register

15	14	13	12	11	10	9	8
TIMEN	Reserved						
RW-0h	R-0h						

7	6	5	4	3	2	1	0
Reserved		PSCDIV				AUTORELOAD	START
R-0h		RW-0h				RW-0h	RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-17. T2CR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	TIMEN	RW	0h	Enables and disables the clocks for the counters to run. The TIMEN bit starts both the prescaler and the main counter. 0x0 = Timer 2 counters are disabled. 0x1 = Timer 2 counters are enabled and the prescaler and counter will begin to count down.
14-6	Reserved	R	0h	Reserved
5-2	PSCDIV	RW	0h	Prescaler divider. The range is: 0000 = divide by 2 to 1100 = divide by 8192.
1	AUTORELOAD	RW	0h	Automatically reloads the counter when it reaches 0. 0x0 = Auto-reload is disabled. 0x1 = Auto-reload is enabled.
0	START	RW	0h	When the START bit is written to, the bit loads and starts the counter. If the device is already in the process of counting down, and a "1" is written to this bit, the counter will be re-loaded with the start value and begin counting down again. When read, the bit will be 1 when the counter is counting down. This register will read 0 when the counter has stopped.

14.6.3.2 TIM2PRD1 Register (offset = 1892h) [reset = FFFFh]

TIM2PRD1 is shown in Figure 14-16 and described in Table 14-18.

Timer 2 Period Register 1

Figure 14-16. TIM2PRD1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRD1															
RW-FFFFh															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-18. TIM2PRD1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	PRD1	RW	FFFFh	The timer 2 period register is 32-bits wide. This is the LSW for the timer 2 period, value 0000 to FFFFh.

14.6.3.3 TIM2PRD2 Register (offset = 1893h) [reset = FFFFh]

TIM2PRD2 is shown in Figure 14-17 and described in Table 14-19.

Timer 2 Period Register 2

Figure 14-17. TIM2PRD2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRD2															
RW-FFFFh															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-19. TIM2PRD2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	PRD2	RW	FFFFh	The timer 2 period register is 32-bits wide. This is the MSW for the timer 2 period, value 0000 to FFFFh.

14.6.3.4 TIM2CNT1 Register (offset = 1894h) [reset = FFFFh]

TIM2CNT1 is shown in Figure 14-18 and described in Table 14-20.

Timer 2 Counter Register 1

Figure 14-18. TIM2CNT1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM1															
RW-FFFFh															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-20. TIM2CNT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TIM1	RW	FFFFh	The timer 2 counter is 32-bits wide. This is the LSW for the timer 2 counter, value 0000 to FFFFh.

14.6.3.5 TIM2CNT2 Register (offset = 1895h) [reset = FFFFh]

TIM2CNT2 is shown in Figure 14-19 and described in Table 14-21.

Timer 2 Counter Register 2

Figure 14-19. TIM2CNT2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM2															
RW-FFFFh															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-21. TIM2CNT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TIM2	RW	FFFFh	The timer 2 counter is 32-bits wide. This is the MSW for the timer 2 counter, value 0000 to FFFFh.

14.6.3.6 T2INSR Register (offset = 1896h) [reset = 0h]

T2INSR is shown in Figure 14-20 and described in Table 14-22.

Timer 2 Input Selection Register

Figure 14-20. T2INSR Register

15	14	13	12	11	10	9	8
TIMRST	Reserved						
RW-0h	R-0h						

7	6	5	4	3	2	1	0
Reserved					TINSEL		
R-0h					RW-0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-22. T2INSR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	TIMRST	RW	0h	Drives the corresponding Timer 2 module reset signal low (asserted) or high (deasserted). 0x0 = Timer 2 counter reset deasserted. Timer 2 module is out of reset. 0x1 = Timer 2 counter reset asserted. Timer 2 module is in reset.
14-3	Reserved	R	0h	Reserved
2-0	TINSEL	RW	0h	Timer 2 Clock Input Source Selection. 000b = Timer 2 clock is driven by the System Clock (PLL output). 001b = Timer 2 clock is driven by GPIO2, S2. 010b = Timer 2 clock is driven by GPIO8, S12. 011b = Timer 2 clock is driven by GPIO14, PD[4]. 100b = Timer 2 clock is driven by GPIO20, PD[10]. 101b = Timer 2 clock is driven by GPIO26, A[20]. 110b = Timer 2 clock is driven by USB Oscillator. 111b = Timer 2 clock is driven by PLL input clock.

14.6.4 TISR Registers

Table 14-23 lists the memory-mapped registers for the TISR. All register offset addresses not listed in Table 14-23 should be considered as reserved locations and the register contents should not be modified.

Table 14-23. TISR Registers

Offset	Acronym	Register Name	Section
1C3Eh	TISR	Timer Interrupt Selection Register	Section 14.6.4.1

14.6.4.1 TISR Register (offset = 1C3Eh) [reset = 0h]

TISR is shown in Figure 14-21 and described in Table 14-24.

The timer interrupt selection register allows the timer interrupt for each timer (Timer0, Timer1 and Timer2) to be configured as a timer interrupt (TINT) or as a non-maskable interrupt (NMI) to the CPU.

Figure 14-21. TISR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved					TIM2INTSEL	TIM1INTSEL	TIM0INTSEL
R-0h					R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-24. TISR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-3	Reserved	R	0h	
2	TIM2INTSEL	R/W	0	Timer 2 Interrupt Selection. 0 = Timer 2 interrupt is routed to CPU TINT. 1 = Timer 2 interrupt is routed to CPU NMI.
1	TIM1INTSEL	R/W	0	Timer 1 Interrupt Selection. 0 = Timer 1 interrupt is routed to CPU TINT. 1 = Timer 1 interrupt is routed to CPU NMI.
0	TIM0INTSEL	R/W	0	Timer 0 Interrupt Selection. 0 = Timer 0 interrupt is routed to CPU timer interrupt (TINT). 1 = Timer 0 interrupt is routed to CPU non-maskable interrupt (NMI).

14.6.5 TIAFR Registers

Table 14-25 lists the memory-mapped registers for the TIAFR. All register offset addresses not listed in Table 14-25 should be considered as reserved locations and the register contents should not be modified.

Table 14-25. TIAFR Registers

Offset	Acronym	Register Name	Section
1C14h	TIAFR	Timer Interrupt Aggregation Flag Register	Section 14.6.5.1

14.6.5.1 TIAFR Register (offset = 1C14h) [reset = 0h]

Timer Interrupt Aggregation Flag is shown in Figure 14-22 and described in Table 14-26.

The timer interrupt aggregation flag register latches each timer (Timer0, Timer1, and Timer2) interrupt signal when the timer counter expires. Using this register, the programmer can determine which Timer generated the single Timer CPU interrupt signal. Each timer flag needs to be cleared by the CPU with a write of '1' to the corresponding flag bit. Note that the corresponding Timer Interrupt Register must also be cleared.

Figure 14-22. TIAFR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved					TIM2FLAG	TIM1FLAG	TIM0FLAG
R-0h					R/W1C-0h	R/W1C-0h	R/W1C-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-26. TIAFR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-3	Reserved	R	0	
2	TIM2FLAG	R/W1C	0	Timer 2 interrupt flag bit. This bit latches the timer interrupt signal when the timer counter expires. You can clear this flag bit by writing a 1 to it. 0 = Timer has not generated an interrupt. 1 = Timer interrupt has occurred.
1	TIM1FLAG	R/W1C	0	Timer 1 interrupt flag bit. This bit latches the timer interrupt signal when the timer counter expires. You can clear this flag bit by writing a 1 to it. 0 = Timer has not generated an interrupt. 1 = Timer interrupt has occurred.
0	TIM0FLAG	R/W1C	0	Timer 0 interrupt flag bit. This bit latches the timer interrupt signal when the Timer counter expires. You can clear this flag bit by writing a 1 to it. 0 = Timer has not generated an interrupt. 1 = Timer interrupt has occurred.

14.6.6 WATCHDOG TIMER Registers

Table 14-27 lists the memory-mapped registers for the WDT. All register offset addresses not listed in Table 14-27 should be considered as reserved locations and the register contents should not be modified.

Table 14-27. WATCHDOG TIMER REGISTERS

CPU Word Address	Acronym	Register Name	Section
1880h	WDKCKLK	Watchdog Kick Lock Register	Section 14.6.6.1
1882h	WDKICK	Watchdog Kick Register	Section 14.6.6.2
1884h	WDSVLR	Watchdog Start Value Lock Register	Section 14.6.6.3
1886h	WDSVR	Watchdog Start Value Register	Section 14.6.6.4
1888h	WDENLOK	Watchdog Enable Lock Register	Section 14.6.6.5
188Ah	WDEN	Watchdog Enable Register	Section 14.6.6.6
188Ch	WDPSLR	Watchdog Prescaler Lock Register	Section 14.6.6.7
188Eh	WDPS	Watchdog Prescaler Register	Section 14.6.6.8

14.6.6.1 WDKCKLK Register (offset = 1880h) [reset = 0h]

WDKCKLK is shown in Figure 14-23 and described in Table 14-28.

Unlocks the WD Kick Register.

Figure 14-23. WDKCKLK Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-28. WDKCKLK Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	KICKLOK	R/W	0h	Used to unlock the Watchdog Kick Register. A two-word key sequence must be written to this register. The following keys must be written this order Key 1 = 0x5555 and Key 2 = 0xAAAA. When this is written the Kick register can now be started. When reading back the KICKLOK register, the value that is returned in bits [1:0] give the current state of the lock state machine, where: 00 = Waiting for Key 1. 01 = Waiting for key 2. 11 = Unlocked.

14.6.6.2 WDKICK Register (offset = 1882h) [reset = 0h]

WDKICK is shown in Figure 14-24 and described in Table 14-29.

Resets the counter to the start value.

Figure 14-24. WDKICK Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-29. WDKICK Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	KICK	R/W	0h	A write to the kick register when it is unlocked causes the watchdown counter to be reloaded with the value in the WD Start Value register and start counting down again. It does not matter what value is written. Reading the register returns the current value of the counter.

14.6.6.3 WDSVLR Register (offset = 1884h) [reset = 0h]

WDSVLR is shown in Figure 14-25 and described in Table 14-30.

Unlocks the WD start value register.

Figure 14-25. WDSVLR Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-30. WDSVLR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	STVALLOK	R/W	0h	Used to unlock the Watchdog Start Value Register. A two-word key sequence must be written to this register. The following keys must be written this order Key 1 = 0x6666 and Key 2 = 0xBBBB. When this is written the WD Start Value register can now be an input. When reading back the STVALLOK register, the value that is returned in bits [1:0] give the current state of the lock state machine, where: 00 = Waiting for Key 1 01 = Waiting for key 2 11 = Unlocked.

14.6.6.4 WDSVR Register (offset = 1886h) [reset = 0h]

WDSVR is shown in Figure 14-26 and described in Table 14-31.

Defines the start value for the WD counter.

Figure 14-26. WDSVR Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-31. WDSVR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	STRTVAL	R/W	0h	The value written to this register is what is loaded into the WD counter when the kick register is written to. A read of this register will return the the Start Value for the counter.

14.6.6.5 WDENLOK Register (offset = 1888h) [reset = 0h]

WDENLOK is shown in Figure 14-27 and described in Table 14-32.

Unlocks the WD enable register.

Figure 14-27. WDENLOK Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-32. WDENLOK Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	ENLOK	R/W	0h	Used to unlock the Watchdog Enable Register. A three-word key sequence must be written to this register. The keys must be written in the following order: Key 1 = 0x7777, Key 2 = 0xCCCC, and Key 3 = 0xDDDD. When this is written the WD Enable register can be used as an input. When reading back the ENLOK register, the value that is returned in bits [1:0] give the current state of the lock state machine, where: 00 = Waiting for Key 1. 01 = Waiting for key 2. 10 = Waiting for Key 3. 11 = Unlocked.

14.6.6.6 WDEN Register (offset = 188Ah) [reset = 0h]

WDEN is shown in Figure 14-28 and described in Table 14-33.

WD enable/disable bit.

Figure 14-28. WDEN Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							

7	6	5	4	3	2	1	0
RESERVED							EN
R-0h							R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-33. WDEN Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	RESERVED	R	0h	Reserved
0	EN	R/W	0h	Used to enable/disable the WD timer. When enabled the counter begins counting down if the counter is allowed to reach 0 the chip will be reset. 0x0 = Watchdog Timer is disabled 0x1 = Watchdog Timer is enabled and begins counting down.

14.6.6.7 WDPSLR Register (offset = 188Ch) [reset = 0h]

WDPSLR is shown in Figure 14-29 and described in Table 14-34.

WD enable/disable bit.

Figure 14-29. WDPSLR Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-34. WDPSLR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	PSLOK	R/W	0h	Used to unlock the Watchdog Prescaler Register. A two-word key sequence must be written to this register. The keys must be written in the following order: Key 1 = 0x5A5A and Key 2 = 0xA5A5. When this is written the WD Prescaler register can now be loaded. When reading back the PSLOK register, the value that is returned in bits [1:0] give the current state of the lock state machine, where: 00 = Waiting for Key 1. 01 = Waiting for key 2. 11 = Unlocked.

14.6.6.8 WDPS Register (offset = 188Eh) [reset = 0h]

WDPS is shown in Figure 14-30 and described in Table 14-35.

WD enable/disable bit.

Figure 14-30. WDPS Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 14-35. WDPS Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	PS	R/W	0h	The WD prescaler register stores the start value for the WD Prescaler. Each time the PS register counts down to 0 the WD counter is decremented by 1. A read will return the last value written to this register (the prescaler start value)

Universal Asynchronous Receiver/Transmitter (UART)

Topic **Page**

15.1 Introduction

The following sections describe the features of the universal asynchronous receiver/transmitter (UART) peripheral.

15.1.1 Purpose of the Peripheral

The UART performs serial-to-parallel conversions on data received from a peripheral device and parallel-to-serial conversion on data received from the CPU. The CPU can read the UART status at any time. The UART includes control capability and a processor interrupt system that can be tailored to minimize software management of the communications link.

The UART includes a programmable baud generator capable of dividing the UART input clock by divisors from 1 to 65,535 and producing a 16 x reference clock for the internal transmitter and receiver logic. For detailed timing and electrical specifications for the UART, see the device-specific data manual.

The UART peripheral is based on the industry standard TL16C550 asynchronous communications element, which in turn is a functional upgrade of the TL16C450. Functionally similar to the TL16C450 on power up (single character or TL16C450 mode), the UART can be placed in an alternate FIFO (TL16C550) mode. This relieves the CPU of excessive software overhead by buffering received and transmitted characters. The receiver and transmitter FIFOs store up to 16 bytes including three additional bits of error status per byte for the receiver FIFO.

15.1.2 Features

The UART peripheral has the following features:

- Programmable baud rates (frequency pre-scale values from 1 to 65535).
- Fully programmable serial interface characteristics:
  - 5, 6, 7, or 8-bit characters.
  - Even, odd, or no PARITY bit generation and detection.
  - 1, 1.5, or 2 STOP bit generation.
- 16-byte depth transmitter and receiver FIFOs:
  - The UART can be operated with or without the FIFOs.
  - 1, 4, 8, or 14 byte selectable receiver FIFO trigger level for autoflow control and DMA.
- DMA signaling capability for both received and transmitted data.
- CPU interrupt capability for both received and transmitted data.
- False START bit detection.
- Line break generation and detection.
- Internal diagnostic capabilities:
  - Loopback controls for communications link fault isolation.
  - Break, parity, overrun, and framing error simulation.
- Programmable autoflow control using CTS and RTS signals.

Table 15-1 summarizes the capabilities supported on the UART.

Table 15-1. UART Supported Features/Characteristics by Instance

Feature	Support
5, 6, 7 or 8-bit characters	Supported
Even, odd, or no PARITY bit	Supported
1, 1.5, or 2 STOP bit generation	Supported
Line break generation and detection	Supported
Internal loop back	Supported
DMA sync events for both received and transmitted data	Supported
1, 4, 8, or 14 byte selectable receiver FIFO trigger level	Supported
Polling/Interrupt	Supported
Modem control functions using CTS and RTS	Supported
Autoflow control using CTS and RTS	Supported
DTR and DSR	Not supported
Ring indication	Not supported
Carrier detection	Not supported
Single-character transfer mode (mode 0) in DMA mode	Not supported

15.1.3 Functional Block Diagram

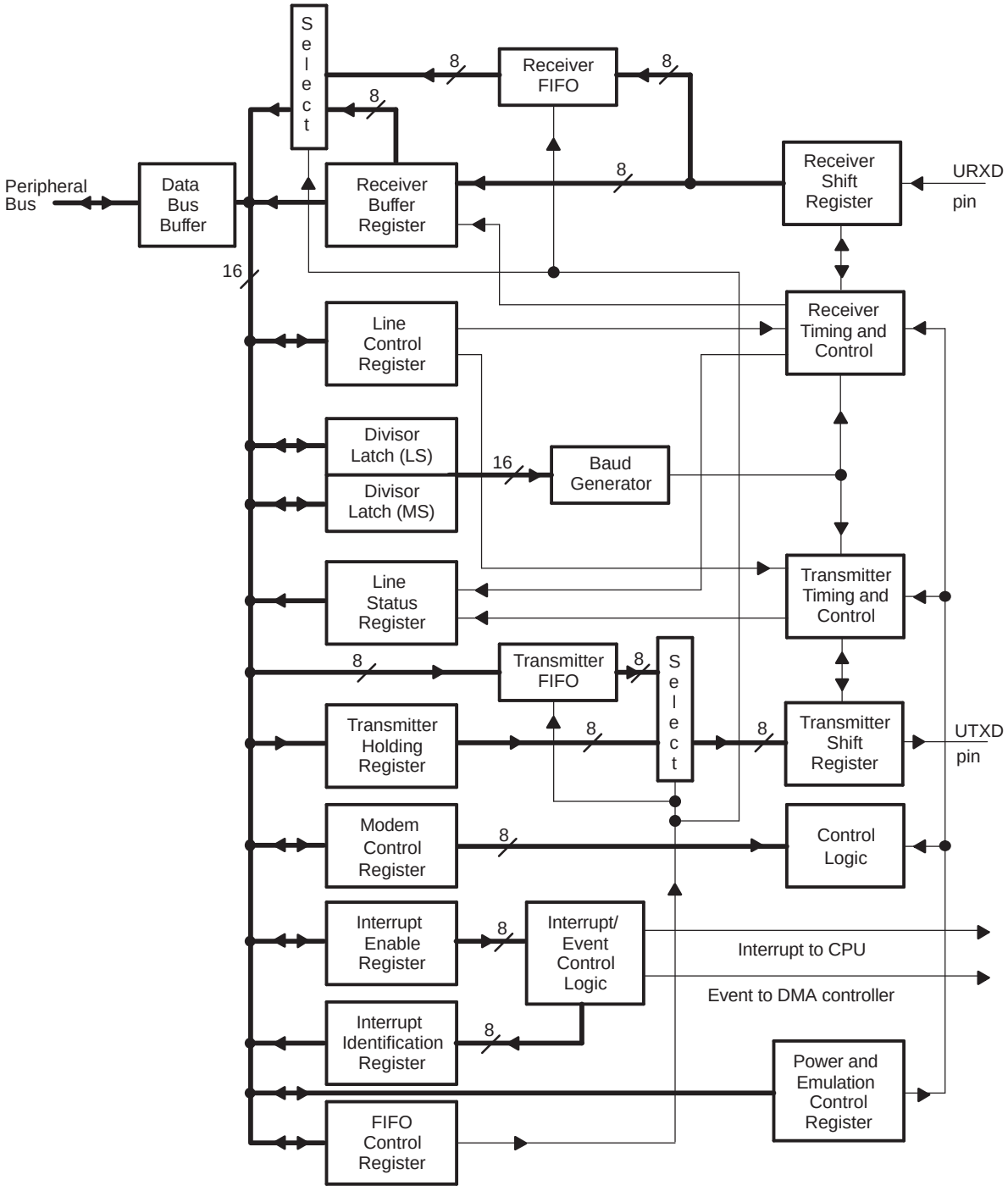
A functional block diagram of the UART is shown in Figure 15-1.

15.1.4 Industry Standard(s) Compliance Statement

The UART peripheral is based on the industry standard TL16C550 asynchronous communications element, which is a functional upgrade of the TL16C450. Any deviations in supported functions are indicated in Table 15-1.

The information in this document assumes the reader is familiar with these standards.

Figure 15-1. UART Block Diagram



15.2 Peripheral Architecture

15.2.1 Clock Generation and Control

The UART bit clock is derived from the internal system clock. Figure 15-2 is a conceptual clock generation diagram for the UART. The clock generator receives either the USB oscillator or a signal from an external clock source and produces DSP system clock. This clock is used by the DSP CPU and peripherals.

The UART contains a programmable baud generator that takes the UART input clock and divides it by a divisor in the range between 1 and (2^{16} - 1) to produce a baud clock (BCLK). The frequency of BCLK is sixteen times (16 ×) the baud rate; each received or transmitted bit lasts 16 BCLK cycles. When the UART is receiving, the bit is sampled in the 8th BCLK cycle. The formula to calculate the divisor is:

$$\text{Divisor} = \frac{\text{UART input clock frequency}}{\text{Desired baud rate} \times 16}$$

Two 8-bit register fields (DLH and DLL), called divisor latches, hold this 16-bit divisor. DLH holds the most significant bits of the divisor, and DLL holds the least significant bits of the divisor. For information about these register fields, see Section 15.3.1. These divisor latches must be loaded during initialization of the UART in order to ensure desired operation of the baud generator. Writing to the divisor latches results in two wait states being inserted during the write access while the baud generator is loaded with the new value.

Figure 15-2 summarizes the relationship between the transferred data bit, BCLK, and the UART input clock.

Example baud rates and divisor values relative to a 60- and 100-MHz UART input clock are shown in Table 15-2 and Table 15-3 respectively. Refer to the device-specific data sheet to determine the maximum baud rate supported on the DSP.

The device DSP includes logic which can be used to gate the clock to its on-chip peripherals. The UART input clock can be enabled and disabled through the peripheral clock gating configuration register 1 (PCGCR1).

Figure 15-2. UART Clock Generation Diagram

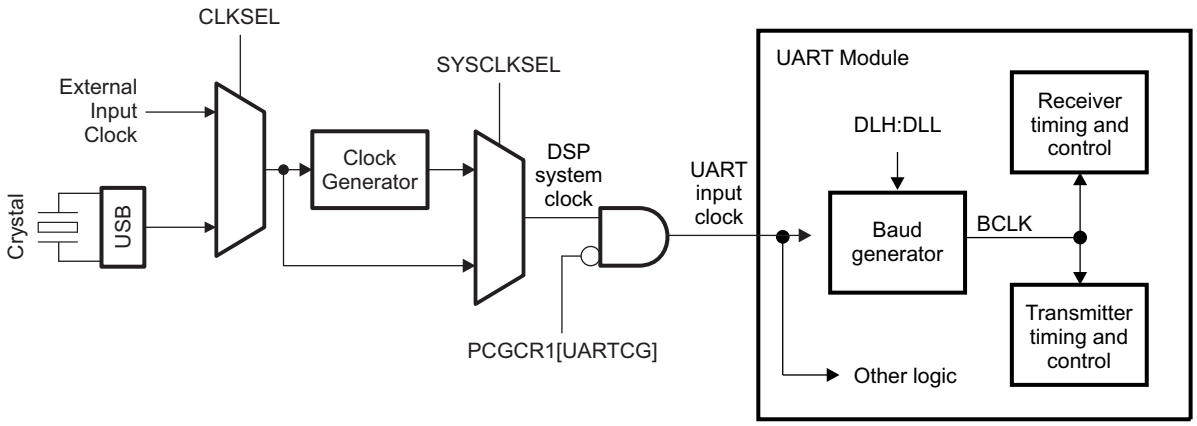


Figure 15-3. Relationship between Data Bit, BCLK, and UART Input Clock

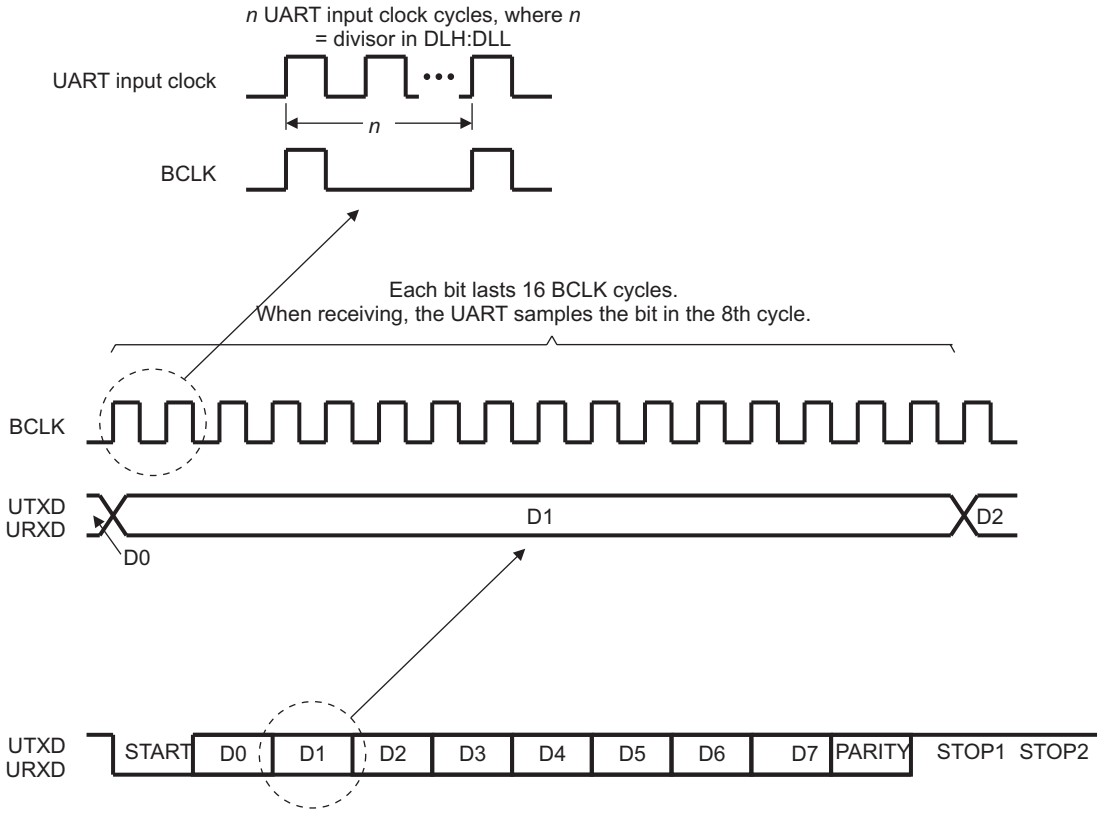


Table 15-2. Baud Rate Examples for 60-MHz UART Input Clock

Baud Rate	Divisor Value	Actual Baud Rate	Error (%)
2400	1563	2399.23	-0.032
4800	781	4801.54	0.032
9600	391	9590.79	-0.096
19200	195	19230.77	0.16
38400	98	38265.31	-0.351
56000	67	55970.15	-0.053
128000	29	129310.34	1.024

Table 15-3. Baud Rate Examples for 100-MHz UART Input Clock

Baud Rate	Divisor Value	Actual Baud Rate	Error (%)
2400	2604	2400.15	0.006
4800	1302	4800.31	0.006
9600	651	9600.61	0.006
19200	326	19171.78	-0.147
38400	163	38343.56	-0.147
56000	112	55803.57	-0.351
128000	49	127551.02	-0.351

15.2.2 Signal Descriptions

The UARTs utilize a minimal number of signal connections to interface with external devices. The UART signal descriptions are included in Table 15-4.

Table 15-4. UART Signal Descriptions

Signal Name	Signal Type	Function
UTXD	Output	Serial data transmit
URXD	Input	Serial data receive
UCTS	Input	Clear-to-Send handshaking signal
URTS	Output	Request-to-Send handshaking signal

15.2.3 Pin Multiplexing

The UART pins are multiplexed with other peripherals on the DSP device. To enable UART pin functionality, software must set the parallel port mode bits of the external bus selection register (EBSR) to either 001b, 100b, or 101b. For more information on the pin multiplexing options of the device DSP, please refer to the device-specific data manual.

15.2.4 Protocol Description

15.2.4.1 Transmission

The UART transmitter section includes a transmitter hold register (THR) and a transmitter shift register (TSR). When the UART is in the FIFO mode, THR is a 16-byte FIFO. Transmitter section control is a function of the UART line control register (LCR). Based on the settings chosen in LCR, the UART transmitter sends the following to the receiving device:

- 1 START bit.
- 5, 6, 7, or 8 data bits.
- 1 PARITY bit (optional).
- 1, 1.5, or 2 STOP bits.

15.2.4.2 Reception

The UART receiver section includes a receiver shift register (RSR) and a receiver buffer register (RBR). When the UART is in the FIFO mode, RBR is a 16-byte FIFO. Receiver section control is a function of the UART line control register (LCR). Based on the settings chosen in LCR, the UART receiver accepts the following from the transmitting device:

- 1 START bit.
- 5, 6, 7, or 8 data bits.
- 1 PARITY bit (optional).
- 1 STOP bit (any other STOP bits transferred with the above data are not detected).

15.2.4.3 Data Format

The UART transmits in the following format:

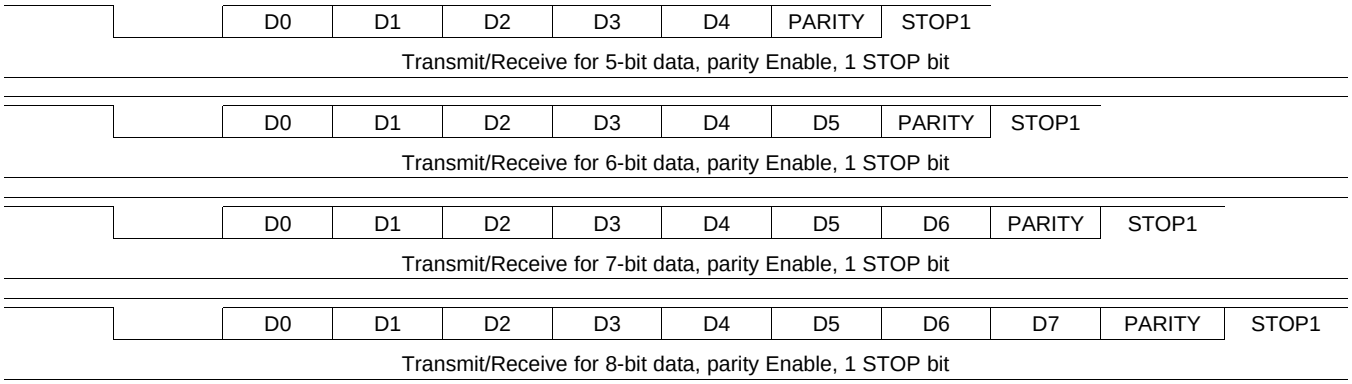
1 START bit, 5, 6, 7, or 8 data bits, depending on the data width selection. 1 PARITY bit, if parity is selected; 1, 1.5, or 2 STOP bits, depending on the STOP bit selection.

The UART receives in the following format:

1 START bit, 5, 6, 7, or 8 data bits, depending on the data width selection. 1 PARITY bit, if parity is selected; 1 STOP bit.

Examples of different protocol formats are shown in Figure 15-4.

Figure 15-4. UART Example Protocol Formats



15.2.5 Operation

15.2.5.1 Transmission

The UART transmitter section includes a transmitter hold register (THR) and a transmitter shift register (TSR). When the UART is in the FIFO mode, THR is a 16-byte FIFO. Transmitter section control is a function of the UART line control register (LCR). Based on the settings chosen in LCR, the UART transmitter sends the following to the receiving device:

- 1 START bit.
- 5, 6, 7, or 8 data bits.
- 1 PARITY bit (optional).
- 1, 1.5, or 2 STOP bits.

THR receives data from the internal data bus, and when TSR is ready, the UART moves the data from THR to TSR. The UART serializes the data in TSR and transmits the data on the TX pin. In the non-FIFO mode, if THR is empty and the THR empty interrupt is enabled in the interrupt enable register (IER), an interrupt is generated. This interrupt is cleared when a character is loaded into THR. In the FIFO mode, the interrupt is generated when the transmitter FIFO is empty, and it is cleared when at least one byte is loaded into the FIFO.

15.2.5.2 Reception

The UART receiver section includes a receiver shift register (RSR) and a receiver buffer register (RBR). When the UART is in the FIFO mode, RBR is a 16-byte FIFO. Timing is supplied by the receiver clock. Receiver section control is a function of the UART line control register (LCR). Based on the settings chosen in LCR, the UART receiver accepts the following from the transmitting device:

- 1 START bit.
- 5, 6, 7, or 8 data bits.
- 1 PARITY bit (optional).
- 1 STOP bit (any other STOP bits transferred with the above data are not detected).

RSR receives the data bits from the RX pin. Then RSR concatenates the data bits and moves the resulting value into RBR (or the receiver FIFO). The UART also stores three bits of error status information next to each received character, to record a parity error, framing error, or break.

In the non-FIFO mode, when a character is placed in RBR and the receiver data-ready interrupt is enabled in the interrupt enable register (IER), an interrupt is generated. This interrupt is cleared when the character is read from RBR. In the FIFO mode, the interrupt is generated when the FIFO is filled to the trigger level selected in the FIFO control register (FCR), and it is cleared when the FIFO contents drop below the trigger level.

15.2.5.3 FIFO Modes

The following two modes can be used for servicing the receiver and transmitter FIFOs:

- FIFO interrupt mode. The FIFO is enabled and the associated interrupts are enabled. Interrupts are sent to the CPU to indicate when specific events occur.
- FIFO poll mode. The FIFO is enabled but the associated interrupts are disabled. The CPU polls status bits to detect specific events.

Because the receiver FIFO and the transmitter FIFO are controlled separately, either one or both can be placed into the interrupt mode or the poll mode.

15.2.5.3.1 FIFO Interrupt Mode

When the receiver FIFO is enabled in the FIFO control register (FCR) and the receiver interrupts are enabled in the interrupt enable register (IER), the interrupt mode is selected for the receiver FIFO. The following are important points about the receiver interrupts:

- The receiver data-ready interrupt is issued to the CPU when the FIFO has reached the trigger level that is programmed in FCR. It is cleared when the CPU or the DMA controller reads enough characters from the FIFO such that the FIFO drops below its programmed trigger level.
- The receiver line status interrupt is generated in response to an overrun error, a parity error, a framing error, or a break. This interrupt has higher priority than the receiver data-ready interrupt. For details, see Section 15.2.9.
- The data-ready (DR) bit in the line status register (LSR) indicates the presence or absence of characters in the receiver FIFO. The DR bit is set when a character is transferred from the receiver shift register (RSR) to the empty receiver FIFO. The DR bit remains set until the FIFO is empty again.

- A receiver time-out interrupt occurs if all of the following conditions exist:
  - At least one character is in the FIFO,
  - The most recent character was received more than four continuous character times ago. A character time is the time allotted for 1 START bit, *n* data bits, 1 PARITY bit, and 1 STOP bit, where *n* depends on the word length selected with the WLS bits in the line control register (LCR). See Table 15-5.
  - The most recent read of the FIFO has occurred more than four continuous character times before.
- Character times are calculated by using the baud rate.
- When a receiver time-out interrupt has occurred, it is cleared and the time-out timer is cleared when the CPU or the DMA controller reads one character from the receiver FIFO. The interrupt is also cleared if a new character is received in the FIFO or if the URRST bit is cleared in the power and emulation management register (PWREMU_MGMT).
- If a receiver time-out interrupt has not occurred, the time-out timer is cleared after a new character is received or after the CPU or DMA reads the receiver FIFO.

NOTE: If the interrupt is caused by the time-out timer, the interrupt can be cleared when a start bit is received. The user must also check the receive data when the time-out interrupt is cleared.

When the transmitter FIFO is enabled in FCR and the transmitter holding register empty interrupt is enabled in IER, the interrupt mode is selected for the transmitter FIFO. The transmitter holding register empty interrupt occurs when the transmitter FIFO is empty. It is cleared when the transmitter hold register (THR) is loaded (1 to 16 characters may be written to the transmitter FIFO while servicing this interrupt).

Table 15-5. Character Time for Word Lengths

Word Length (*n*)	Character Time	Four Character Times
5	Time for 8 bits	Time for 32 bits
6	Time for 9 bits	Time for 36 bits
7	Time for 10 bits	Time for 40 bits
8	Time for 11 bits	Time for 44 bits

15.2.5.3.2 FIFO Poll Mode

When the receiver FIFO is enabled in the FIFO control register (FCR) and the receiver interrupts are disabled in the interrupt enable register (IER), the poll mode is selected for the receiver FIFO. Similarly, when the transmitter FIFO is enabled and the transmitter interrupts are disabled, the transmitter FIFO is in the poll mode. In the poll mode, the CPU detects events by checking bits in the line status register (LSR):

- The RXFIFOE bit indicates whether there are any errors in the receiver FIFO.
- The TEMT bit indicates that both the transmitter holding register (THR) and the transmitter shift register (TSR) are empty.
- The THRE bit indicates when THR is empty.
- The BI (break), FE (framing error), PE (parity error), and OE (overrun error) bits specify which error or errors have occurred.
- The DR (data-ready) bit is set as long as there is at least one byte in the receiver FIFO.

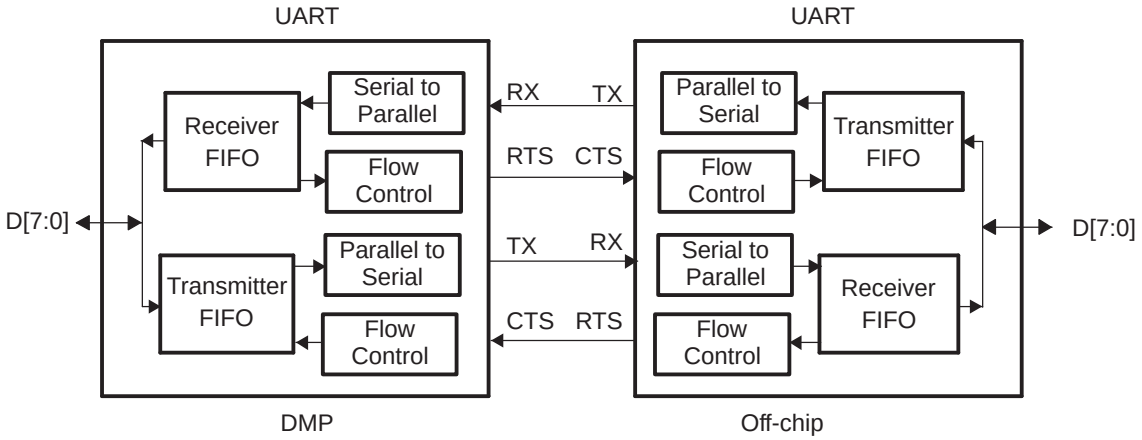
Also, in the FIFO poll mode:

- The interrupt identification register (IIR) is not affected by any events because the interrupts are disabled.
- The UART does not indicate when the receiver FIFO trigger level is reached or when a receiver time-out occurs.

15.2.5.4 Autoflow Control

The UART can employ autoflow control by connecting the CTS and RTS signals. The CTS input must be active before the transmitter FIFO can transmit data. The RTS becomes active when the receiver needs more data and notifies the sending device. When RTS is connected to CTS, data transmission does not occur unless the receiver FIFO has space for the data. Therefore, when two UARTs are connected as shown in Figure 15-5 with autoflow enabled, overrun errors are eliminated.

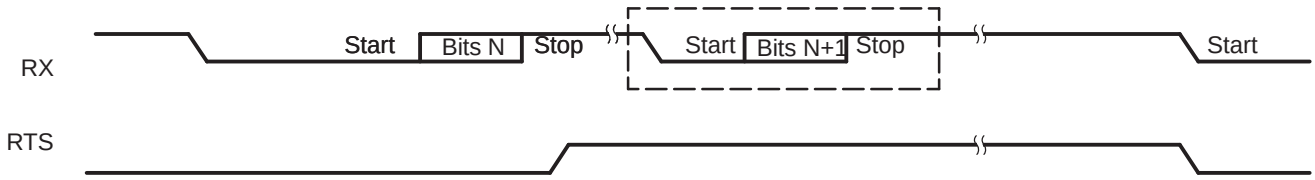
Figure 15-5. UART Interface Using Autoflow Diagram



15.2.5.4.1 RTS Behavior

RTS data flow control originates in the receiver block (see Figure 15-1). When the receiver FIFO level reaches a trigger level of 1, 4, 8, or 14 (see Figure 15-6), RTS is deasserted. The sending UART may send an additional byte after the trigger level is reached (assuming the sending UART has another byte to send), because it may not recognize the deassertion of RTS until after it has begun sending the additional byte. For trigger level 1, 4, and 8, RTS is automatically reasserted once the receiver FIFO is emptied. For trigger level 14, RTS is automatically reasserted once the receiver FIFO drops below the trigger level.

Figure 15-6. Autoflow Functional Timing Waveforms for RTS



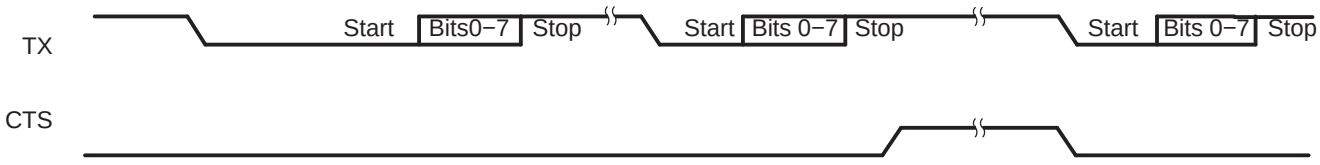
(1) N = Receiver FIFO trigger level.

(2) The two blocks in dashed lines cover the case where an additional byte is sent.

15.2.5.4.2 CTS Behavior

The transmitter checks CTS before sending the next data byte. If CTS is active, the transmitter sends the next byte. To stop the transmitter from sending the following byte, CTS must be released before the middle of the last STOP bit that is currently being sent (see Figure 15-7). When flow control is enabled, CTS level changes do not trigger interrupts because the device automatically controls its own transmitter. Without autoflow control, the transmitter sends any data present in the transmitter FIFO and a receiver overrun error may result.

Figure 15-7. Autoflow Functional Timing Waveforms for CTS



(1) When CTS is active (low), the transmitter keeps sending serial data out.

(2) When CTS goes high before the middle of the last STOP bit of the current byte, the transmitter finishes sending the current byte but it does not send the next byte.

(3) When CTS goes from high to low, the transmitter begins sending data again.

15.2.5.5 Loopback Control

The UART can be placed in the diagnostic mode using the LOOP bit in the modem control register (MCR), which internally connects the UART output back to the UART input. In this mode, the transmit and receive data paths, the transmitter and receiver interrupts, and the modem control interrupts can be verified without connecting to another UART.

15.2.6 Exception Processing

15.2.6.1 Divisor Latch Not Programmed

Since the processor reset signal has no effect on the divisor latch, the divisor latch will have an unknown value after power up. If the divisor latch is not programmed after power up, the baud clock (BCLK) will not operate and will instead be set to a constant logic 1 state.

The divisor latch values should always be reinitialized following a processor reset.

15.2.6.2 Changing Operating Mode During Busy Serial Communication

Since the serial link characteristics are based on how the control registers are programmed, the UART will expect the control registers to be static while it is busy engaging in a serial communication. Therefore, changing the control registers while the module is still busy communicating with another serial device will most likely cause an error condition and should be avoided.

15.2.7 Reset Considerations

The UART peripheral has two reset sources: software reset and hardware reset.

15.2.7.1 Software Reset Considerations

The UART peripheral can be reset by software through the transmitter reset (UTRST) and the receiver reset (URRST) bits of the UART power and emulation management register (PWREMU_MGMT) or through the UART_RST bit in the peripheral reset control register (PRCR).

The UTRST bit controls the transmitter part of the UART only. When UTRST is cleared to 0, the transmitter is disabled and placed in reset. When UTRST is set to 1, the transmitter is enabled. The URRST bit controls the receiver portion of the UART in a similar fashion. In each case, placing the receiver and/or transmitter in reset will reset the state machine of the affected portion but does not affect the UART registers.

When PG4_RST in the peripheral reset control register (PRCR) is set to 1, a hardware reset is forced on the UART. The effects of a hardware reset are described in the next section. Please note that the UART input clock must be enabled when using UART_RST (see Section 15.2.1). Refer to the device-specific data manual for more details on PRCR.

15.2.7.2 Hardware Reset Considerations

A hardware reset is always initiated during a full chip reset. Alternatively, software can force an UART hardware reset through the PG4_RST bit of the peripheral reset control register (PRCR). See the device data manual for more details on PRCR.

NOTE: PRCR resets other peripherals besides the UART. For more details on this bit and register, refer to the device-specific data manual.

When a hardware reset occurs, all the registers of the UART peripheral are set to their default values and the UART peripheral remains disabled until the transmitter reset (UTRST) and the receiver reset (URRST) bits of the UART power and emulation management register (PWREMU_MGMT) are changed to 1.

15.2.8 Initialization

The following steps are required to initialize the UART:

1. Perform the necessary device pin multiplexing setup (see Section 15.2.3 for more details).
2. Ensure the UART is out of reset by waiting until UART_RST = 0 in the peripheral reset control register (PRCR).
3. Enable the UART input clock by setting UARTCG to 0 in the peripheral clock gating configuration register (PCGCR1). See the device-specific data manual for more information on PCGCR1.
4. Place the UART transmitter and receiver in reset by setting UTRST and URRST to 0 in the UART power and emulation management register (PWREMU_MGMT).
5. Set the desired baud rate by writing the appropriate clock divisor values to the divisor latch registers (DLL and DLH).
6. Select the desired trigger level and enable the FIFOs by writing the appropriate values to the FIFO control register (FCR) if the FIFOs are used. The FIFOEN bit in FCR must be set first, before the other bits in FCR are configured. Be sure to set the DMAMODE1 bit to 1 as required for proper operation between the DMA and UART.
7. Choose the desired protocol settings by writing the appropriate values to the line control register (LCR).
8. Write appropriate values to the modem control register (MCR) if autoflow control is desired. Note that all UARTs do not support autoflow control, see the device-specific data manual for supported features.
9. Choose the desired response to emulation suspend events by setting the UTRST and URRST bits in the power and emulation management register (PWREMU_MGMT).

15.2.9 Interrupt Support

15.2.9.1 Interrupt Events and Requests

The UART generates the interrupt requests described in Table 15-6. All requests are multiplexed through an arbiter to a single UART interrupt request to the CPU, as shown in Figure 15-8. Each of the interrupt requests has an enable bit in the interrupt enable register (IER) and is recorded in the interrupt identification register (IIR).

If an interrupt occurs and the corresponding enable bit is set to 1, the interrupt request is recorded in IIR and is forwarded to the CPU. If an interrupt occurs and the corresponding enable bit is cleared to 0, the interrupt request is blocked. The interrupt request is neither recorded in IIR nor forwarded to the CPU.

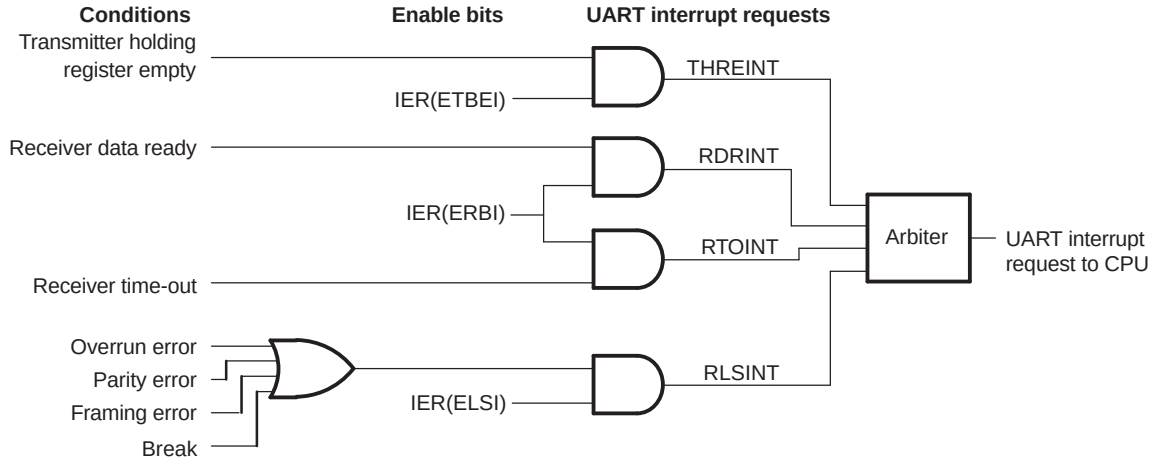
Table 15-6. UART Interrupt Requests Descriptions

UART Interrupt Request	Interrupt Source	Comment
THREINT	THR-empty condition: The transmitter holding register (THR) or the transmitter FIFO is empty. All of the data has been copied from THR to the transmitter shift register (TSR).	If THREINT is enabled in IER, by setting the ETBEI bit, it is recorded in IIR. As an alternative to using THREINT, the CPU can poll the THRE bit in the line status register (LSR).

Table 15-6. UART Interrupt Requests Descriptions (continued)

UART Interrupt Request	Interrupt Source	Comment
RDAINT	Receive data available in non-FIFO mode or trigger level reached in the FIFO mode.	If RDAINT is enabled in IER, by setting the ERBI bit, it is recorded in IIR. As an alternative to using RDAINT, the CPU can poll the DR bit in the line status register (LSR). In the FIFO mode, this is not a functionally equivalent alternative because the DR bit does not respond to the FIFO trigger level. The DR bit only indicates the presence or absence of unread characters.
RTOINT	Receiver time-out condition (in the FIFO mode only): No characters have been removed from or input to the receiver FIFO during the last four character times (see Table 15-5), and there is at least one character in the receiver FIFO during this time.	The receiver time-out interrupt prevents the UART from waiting indefinitely, in the case when the receiver FIFO level is below the trigger level and thus does not generate a receiver data-ready interrupt. If RTOINT is enabled in IER, by setting the ERBI bit, it is recorded in IIR. There is no status bit to reflect the occurrence of a time-out condition.
RLSINT	Receiver line status condition: An overrun error, parity error, framing error, or break has occurred.	If RLSINT is enabled in IER, by setting the ELSI bit, it is recorded in IIR. As an alternative to using RLSINT, the CPU can poll the following bits in the line status register (LSR): overrun error indicator (OE), parity error indicator (PE), framing error indicator (FE), and break indicator (BI).

Figure 15-8. UART Interrupt Request Enable Paths



15.2.9.2 Interrupt Multiplexing

The UART has a dedicated interrupt signal to the CPU that is not multiplexed with any other interrupt source.

15.2.10 DMA Event Support

In the FIFO mode, the UART generates the following two DMA events:

- **Receive event (URXEVT):** The trigger level for the receiver FIFO (1, 4, 8, or 14 characters) is set with the RXFIFTL bit in the FIFO control register (FCR). Every time the trigger level is reached or a receiver time-out occurs, the UART sends a receive event to the DMA controller. In response, the DMA controller reads the data from the receiver FIFO by way of the receiver buffer register (RBR).
- **Transmit event (UTXEVT):** When the transmitter FIFO is empty (when the last byte in the transmitter FIFO has been copied to the transmitter shift register), the UART sends an UTXEVT signal to the DMA controller. In response, the DMA controller refills the transmitter FIFO by way of the transmitter holding register (THR). The UTXEVT signal is also sent to the DMA controller when the UART is taken out of

reset using the UTRST bit in the power and emulation management register (PWREMU_MGMT).

Activity in DMA channels can be synchronized to these events. In the non-FIFO mode, the UART generates no DMA events. Any DMA channel synchronized to either of these events must be enabled at the time the UART event is generated. Otherwise, the DMA channel will miss the event and, unless the UART generates a new event, no data transfer will occur.

15.2.11 Power Management

The UART peripheral can be clock-gated to conserve power during periods of no activity. The input clock of the UART can be turned off by using the peripheral clock gating configuration register (PCGCR). For detailed information on PCGCR, see the device-specific data manual.

15.2.12 Emulation Considerations

Note also that emulator accesses are essentially transparent to UART operation. Emulator read operations do not affect any register contents, status bits, or operating states. Emulator writes, however, may affect register contents and may affect UART operation, depending on what register is accessed and what value is written.

The UART registers can be read from or written to during emulation suspend events, even if the UART activity has stopped.

15.3 UART Registers

The system programmer has access to and control over any of the UART registers. These registers, which control UART operations, receive data, and transmit data, and can be accessed by the CPU at the word address specified. Note that the CPU accesses all peripheral registers through its I/O space.

The following registers share one address:

- RBR, THR, and DLL. When the DLAB bit in LCR is 0, reading from the address gives the content of RBR, and writing to the address modifies THR. When DLAB = 1, all accesses at the address read or modify DLL. DLL can also be accessed by the CPU at word address 1B10h.
- IER and DLH. When DLAB = 0, all accesses read or modify IER. When DLAB = 1, all accesses read or modify DLH. DLH can also be accessed by the CPU at word address 1B12h.
- IIR and FCR share one address. Regardless of the value of the DLAB bit, reading from the address gives the content of IIR, and writing modifies FCR.

15.3.1 UART Registers

Table 15-7 lists the memory-mapped registers for the UART. All register offset addresses not listed in Table 15-7 should be considered as reserved locations and the register contents should not be modified.

Table 15-7. UART REGISTERS

CPU Word Address	Acronym	Register Name	Section
1B00h	RBR	Receiver Buffer Register	Section 15.3.1.1
1B00h	THR	Transmitter Holding Register	Section 15.3.1.2
1B02h	IER	Interrupt Enable Register	Section 15.3.1.3
1B04h	IIR	Interrupt Identification Register	Section 15.3.1.4
1B04h	FCR	FIFO Control Register	Section 15.3.1.5
1B06h	LCR	Line Control Register	Section 15.3.1.6
1B08h	MCR	Modem Control Register	Section 15.3.1.7
1B0Ah	LSR	Line Status Register	Section 15.3.1.8
1B0Eh	SCR	Scratch Register	Section 15.3.1.9
1B10h	DLL	Divisor LSB Latch	Section 15.3.1.10
1B12h	DLH	Divisor MSB Latch	Section 15.3.1.11
1B18h	PWREMU_MGMT	Power Management and Emulation Register	Section 15.3.1.12

15.3.1.1 RBR Register (offset = 1B00h) [reset = 0h]

RBR is shown in Figure 15-9 and described in Table 15-8.

RSR receives serial data from the RX pin. Then RSR concatenates the data and moves it into RBR (or the receiver FIFO).

Figure 15-9. RBR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
DATA							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 15-8. RBR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	
7-0	DATA	R	0h	Received data

15.3.1.2 THR Register (offset = 1B00h) [reset = 0h]

THR is shown in Figure 15-10 and described in Table 15-9.

THR receives data from the internal data bus and when TSR is idle, the UART moves the data from THR to TSR. The UART serializes the data in TSR and transmits the data on the TX pin.

Figure 15-10. THR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							
7	6	5	4	3	2	1	0
DATA							
W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 15-9. THR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	
7-0	DATA	W	0h	Data to transmit

15.3.1.3 IER Register (offset = 1B02h) [reset = 0h]

IER is shown in Figure 15-11 and described in Table 15-10.

The interrupt enable register (IER) is used to individually enable or disable each type of interrupt request that can be generated by the UART. Each interrupt request that is enabled in IER is forwarded to the CPU.

Figure 15-11. IER Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved				Reserved	ELSI	ETBEI	ERBI
R-0h				R-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 15-10. IER Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	Reserved	R	0h	
3	Reserved	R	0h	
2	ELSI	R/W	0h	Receiver line status interrupt enable. 0x0 = DISABLE: Receiver line status interrupt is disabled. 0x1 = ENABLE: Receiver line status interrupt is enabled.
1	ETBEI	R/W	0h	Transmitter holding register empty interrupt enable. 0x0 = DISABLE: Transmitter line status interrupt is disabled. 0x1 = ENABLE: Transmitter line status interrupt is enabled.
0	ERBI	R/W	0h	Receiver data available interrupt and character timeout indication interrupt enable. 0x0 = DISABLE: Receiver data available interrupt and character timeout indication interrupt is disabled. 0x1 = ENABLE: Receiver data available interrupt and character timeout indication interrupt is enabled.

15.3.1.4 IIR Register (offset = 1B04h) [reset = 1h]

IIR is shown in Figure 15-12 and described in Table 15-11.

When an interrupt is generated and enabled in the interrupt enable register (IER), IIR indicates that an interrupt is pending in the IPEND bit and encodes the type of interrupt in the INTID bits.

Figure 15-12. IIR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
FIFOEN		Reserved		INTID			IPEND
R-0h		R-0h		R-0h			R-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 15-11. IIR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	
7-6	FIFOEN	R	0h	FIFOs enabled. 0x0 = DISABLED: Non-FIFO mode. 0x1 = ENABLED: FIFOs are enabled. FIFOEN bit in the FIFO control register (FCR) is set to 1.
5-4	Reserved	R	0h	
3-1	INTID	R	0h	Interrupt type. 0x0 = RSV0: Reserved 0x1 = THRE: Transmitter holding register empty (priority 3). 0x2 = RDA: Receiver data available (priority 2). 0x3 = RLS: Receiver line status (priority 1, highest). 0x4 = RSV4: Reserved 0x5 = RSV5: Reserved 0x6 = CTI: Character timeout indication (priority 2). 0x7 = RSV7: Reserved
0	IPEND	R	1h	Interrupt pending. When any UART interrupt is generated and is enabled in IER, IPEND is forced to 0. IPEND remains 0 until all pending interrupts are cleared or until a hardware reset occurs. If no interrupts are enabled, IPEND is never forced to 0. 0x0 = PENDING: Interrupts are pending. 0x1 = NONE: No interrupts are pending.

15.3.1.5 FCR Register (offset = 1B04h) [reset = 0h]

FCR is shown in Figure 15-13 and described in Table 15-12.

Use FCR to enable and clear the FIFOs and to select the receiver FIFO trigger level FCR.

Figure 15-13. FCR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
RXFIFTL		Reserved		DMAMODE1	TXCLR	RXCLR	FIFOEN
W-0h		R-0h		W-0h	W-0h	W-0h	W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 15-12. FCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	
7-6	RXFIFTL	W	0h	Receiver FIFO trigger level. RXFIFTL sets the trigger level for the receiver FIFO. When the trigger level is reached, a receiver data-ready interrupt is generated (if the interrupt request is enabled). Once the FIFO drops below the trigger level, the interrupt is cleared. 0x0 = CHAR1: 1 byte 0x1 = CHAR4: 4 bytes 0x2 = CHAR8: 8 bytes 0x3 = CHAR14: 14 bytes
5-4	Reserved	R	0h	
3	DMAMODE1	W	0h	DMA MODE1 enable if FIFOs are enabled. Always write 1 to DMAMODE1. After a hardware reset, change DMAMODE1 from 0 to 1. DMAMODE1 = 1 is a requirement for proper communication between the UART and the DMA controller. 0x0 = DISABLE: DMA mode is disabled. 0x1 = ENABLE: DMA mode is enabled.
2	TXCLR	W	0h	Transmitter FIFO clear. Write a 1 to TXCLR to clear the bit.
1	RXCLR	W	0h	Receiver FIFO clear. Write a 1 to RXCLR to clear the bit.
0	FIFOEN	W	0h	Transmitter and receiver FIFOs mode enable. FIFOEN must be set before other FCR bits are written to or the FCR bits are not programmed. Clearing this bit clears the FIFO counters. 0x0 = DISABLE: Non-FIFO mode. The transmitter and receiver FIFOs are disabled, and the FIFO pointers are cleared. 0x1 = ENABLE: FIFO mode. The transmitter and receiver FIFOs are enabled.

15.3.1.6 LCR Register (offset = 1B06h) [reset = 0h]

LCR is shown in Figure 15-14 and described in Table 15-13.

The system programmer controls the format of the asynchronous data communication exchange by using LCR. In addition, the programmer can retrieve, inspect, and modify the content of LCR; this eliminates the need for separate storage of the line characteristics in system memory.

Figure 15-14. LCR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
DLAB	BC	SP	EPS	PEN	STB	WLS	
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 15-13. LCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	
7	DLAB	R/W	0h	Divisor latch access bit. The divisor latch registers (DLL and DLH) can be accessed at dedicated addresses or at addresses shared by RBR, THR, and IER. Using the shared addresses requires toggling DLAB to change which registers are selected. If you use the dedicated addresses, you can keep DLAB = 0. 0x0 = DLABON: Allows access to the receiver buffer register (RBR), the transmitter holding register (THR), and the interrupt enable register (IER) selected. At the address shared by RBR, THR, and DLL, the CPU can read from RBR and write to THR. At the address shared by IER and DLH, the CPU can read from and write to IER. 0x1 = DLABOFF: Allows access to the divisor latches of the baud generator during a read or write operation (DLL and DLH). At the address shared by RBR, THR, and DLL, the CPU can read from and write to DLL. At the address shared by IER and DLH, the CPU can read from and write to DLH.
6	BC	R/W	0h	Break control 0x0 = DISABLE: Break condition is disabled 0x1 = ENABLE: Break condition is transmitted to the receiving UART. A break condition is a condition where the UART_TX signal is forced to the spacing (cleared) state.
5	SP	R/W	0h	Stick parity. The SP bit works in conjunction with the EPS and PEN bits. 0x0 = DISABLE: Stick parity is disabled. 0x1 = ENABLE: Stick parity is enabled. - When odd parity is selected (EPS = 0), the PARITY bit is transmitted and checked as set. - When even parity is selected (EPS = 1), the PARITY bit is transmitted and checked as cleared.
4	EPS	R/W	0h	Even parity select. Selects the parity when parity is enabled (PEN = 1). The EPS bit works in conjunction with the SP and PEN bits. 0x0 = ODD: Odd parity is selected (an odd number of logic 1s is transmitted or checked in the data and PARITY bits). 0x1 = EVEN: Even parity is selected (an even number of logic 1s is transmitted or checked in the data and PARITY bits).
3	PEN	R/W	0h	Parity enable. The PEN bit works in conjunction with the SP and EPS bits. 0x0 = DISABLE: No PARITY bit is transmitted or checked. 0x1 = ENABLE: Parity bit is generated in transmitted data and is checked in received data between the last data word bit and the first STOP bit.

Table 15-13. LCR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	STB	R/W	0h	Number of STOP bits generated. STB specifies 1, 1.5, or 2 STOP bits in each transmitted character. When STB = 1, the WLS bit determines the number of STOP bits. The receiver clocks only the first STOP bit, regardless of the number of STOP bits selected. 0x0 = 1STOPBIT: 1 STOP bit is generated. 0x1 = MULTSTOPBIT: WLS bit determines the number of STOP bits: - When WLS = 0, 1.5 STOP bits are generated. - When WLS = 1h, 2h, or 3h, 2 STOP bits are generated.
1-0	WLS	R/W	0h	Word length select. Number of bits in each transmitted or received serial character. When STB = 1, the WLS bit determines the number of STOP bits. 0x0 = BITS5: 5 bits 0x1 = BITS6: 6 bits 0x2 = BITS7: 7 bits 0x3 = BITS8: 8 bits

15.3.1.7 MCR Register (offset = 1B08h) [reset = 0h]

MCR is shown in Figure 15-15 and described in Table 15-14.

The modem control register provides the ability to enable/disable the autoflow functions, and enable/disable the loopback function for diagnostic purposes.

Figure 15-15. MCR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved		AFE	LOOP	Reserved		RTS	Reserved
R-0h		R/W-0h	R/W-0h	R-0h		R/W-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 15-14. MCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	
5	AFE	R/W	0h	Autoflow control enable. Autoflow control allows the RTS and CTS signals to provide handshaking between UARTs during data transfer. When AFE = 1, the RTS bit determines the autoflow control enabled. 0x0 = DISABLE: Autoflow control is disabled. 0x1 = ENABLE: Autoflow control is enabled. - When RTS = 0, CTS is only enabled. - When RTS = 1, RTS and CTS are enabled.
4	LOOP	R/W	0h	Loop back mode enable. LOOP is used for the diagnostic testing using the loop back feature 0x0 = DISABLE: Loop back mode is disabled. 0x1(Read) = ENABLE: Loop back mode is enabled. When LOOP is set, the following occu- The UART_TX signal is set high. - The UART_RX pin is disconnected - The output of the transmitter shift register (TSR) is lopped back in to the receiver shift register (RSR) input.
3-2	Reserved	R	0h	
1	RTS	R/W	0h	RTS control bit. When AFE = 1, the RTS bit determines the autoflow control enabled. 0x0 = CTSEN: RTS is disabled, CTS is only enabled. 0x1 = CTSRTSEN: RTS and CTS are enabled.
0	Reserved	R	0h	

15.3.1.8 LSR Register (offset = 1B0Ah) [reset = 60h]

LSR is shown in Figure 15-16 and described in Table 15-15.

LSR provides information to the CPU concerning the status of data transfers. LSR is intended for read operations only; do not write to this register. Bits 1 through 4 record the error conditions that produce a receiver line status interrupt.

Figure 15-16. LSR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
RXFIFOE	TEMT	THRE	BI	FE	PE	OE	DR
R-0h	R-1h	R-1h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 15-15. LSR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	
7	RXFIFOE	R	0h	Receiver FIFO error. In non-FIFO mode: 0 = There has been no error, or RXFIFOE was cleared because the CPU read the erroneous character from the receiver buffer register (RBR). 1 = There is a parity error, framing error, or break indicator in the receiver buffer register (RBR). In FIFO mode: 0 = There has been no error, or RXFIFOE was cleared because the CPU read the erroneous character from the receiver FIFO and there are no more errors in the receiver FIFO. 1 = At least one parity error, framing error, or break indicator in the receiver FIFO.
6	TEMT	R	1h	Transmitter empty (TEMT) indicator. In non-FIFO mode: 0 = Either the transmitter holding register (THR) or the transmitter shift register (TSR) contains a data character. 1 = Both the transmitter holding register (THR) and the transmitter shift register (TSR) are empty. In FIFO mode: 0 = Either the transmitter FIFO or the transmitter shift register (TSR) contains a data character. 1 = Both the transmitter FIFO and the transmitter shift register (TSR) are empty.
5	THRE	R	1h	Transmitter holding register empty (THRE) indicator. If the THRE bit is set and the corresponding interrupt enable bit is set (ETBEI = 1 in IER), an interrupt request is generated. In non-FIFO mode: 0 = Transmitter holding register (THR) is not empty. THR has been loaded by the CPU. 1 = Transmitter holding register (THR) is empty (ready to accept a new character). The content of THR has been transferred to the transmitter shift register (TSR). In FIFO mode: 0 = Transmitter FIFO is not empty. At least one character has been written to the transmitter FIFO. You can write to the transmitter FIFO if it is not full. 1 = Transmitter FIFO is empty. The last character in the FIFO has been transferred to the transmitter shift register (TSR).

Table 15-15. LSR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	BI	R	0h	Break indicator. The BI bit is set whenever the receive data input (RX) was held low for longer than a full-word transmission time. A full-word transmission time is defined as the total time to transmit the START, data, PARITY, and STOP bits. If the BI bit is set and the corresponding interrupt enable bit is set (ELSI = 1 in IER), an interrupt request is generated. In non-FIFO mode: 0 = No break has been detected, or the BI bit was cleared because the CPU read the erroneous character from the receiver buffer register (RBR). 1 = A break has been detected with the character in the receiver buffer register (RBR). In FIFO mode: 0 = No break has been detected, or the BI bit was cleared because the CPU read the erroneous character from the receiver FIFO and the next character to be read from the FIFO has no break indicator. 1 = A break has been detected with the character at the top of the receiver FIFO.
3	FE	R	0h	Framing error (FE) indicator. A framing error occurs when the received character does not have a valid STOP bit. In response to a framing error, the UART sets the FE bit and waits until the signal on the RX pin goes high. Once the RX signal goes high, the receiver is ready to detect a new START bit and receive new data. If the FE bit is set and the corresponding interrupt enable bit is set (ELSI = 1 in IER), an interrupt request is generated. In non-FIFO mode: 0 = No framing error has been detected, or the FE bit was cleared because the CPU read the erroneous data from the receiver buffer register (RBR). 1 = A framing error has been detected with the character in the receiver buffer register (RBR). In FIFO mode: 0 = No framing error has been detected, or the FE bit was cleared because the CPU read the erroneous data from the receiver FIFO and the next character to be read from the FIFO has no framing error. 1 = A framing error has been detected with the character at the top of the receiver FIFO.
2	PE	R	0h	Parity error (PE) indicator. A parity error occurs when the parity of the received character does not match the parity selected with the EPS bit in the line control register (LCR). If the PE bit is set and the corresponding interrupt enable bit is set (ELSI = 1 in IER), an interrupt request is generated. In non-FIFO mode: 0 = No parity error has been detected, or the PE bit was cleared because the CPU read the erroneous data from the receiver buffer register (RBR). 1 = A parity error has been detected with the character in the receiver buffer register (RBR). In FIFO mode: 0 = No parity error has been detected, or the PE bit was cleared because the CPU read the erroneous data from the receiver FIFO and the next character to be read from the FIFO has no parity error. 1 = A parity error has been detected with the character at the top of the receiver FIFO.

Table 15-15. LSR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	OE	R	0h	Overrun error (OE) indicator. An overrun error in the non-FIFO mode is different from an overrun error in the FIFO mode. If the OE bit is set and the corresponding interrupt enable bit is set (ELSI = 1 in IER), an interrupt request is generated. In non-FIFO mode: 0 = No overrun error has been detected, or the OE bit was cleared because the CPU read the content of the line status register (LSR). 1 = Overrun error has been detected. Before the character in the receiver buffer register (RBR) could be read, it was overwritten by the next character arriving in RBR. In FIFO mode: 0 = No overrun error has been detected, or the OE bit was cleared because the CPU read the content of the line status register (LSR). 1 = Overrun error has been detected. If data continues to fill the FIFO beyond the trigger level, an overrun error occurs only after the FIFO is full and the next character has been completely received in the shift register. An overrun error is indicated to the CPU as soon as it happens. The new character overwrites the character in the shift register, but it is not transferred to the FIFO.
0	DR	R	0h	Data-ready (DR) indicator for the receiver. If the DR bit is set and the corresponding interrupt enable bit is set (ERBI = 1 in IER), an interrupt request is generated. In non-FIFO mode: 0 = Data is not ready, or the DR bit was cleared because the character was read from the receiver buffer register (RBR). 1 = Data is ready. A complete incoming character has been received and transferred into the receiver buffer register (RBR). In FIFO mode: 0 = Data is not ready, or the DR bit was cleared because all of the characters in the receiver FIFO have been read. 1 = Data is ready. There is at least one unread character in the receiver FIFO. If the FIFO is empty, the DR bit is set as soon as a complete incoming character has been received and transferred into the FIFO. The DR bit remains set until the FIFO is empty again.

15.3.1.9 SCR Register (offset = 1B0Eh) [reset = 0h]

SCR is shown in Figure 15-17 and described in Table 15-16.

The scratch register (SCR) is intended for programmer's use as a scratch pad in the sense that it temporarily holds programmer's data without affecting any other UART operation.

Figure 15-17. SCR Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
DATA							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 15-16. SCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	
7-0	DATA	R/W	0h	Scratch pad data.

15.3.1.10 DLL Register (offset = 1B10h) [reset = 0h]

DLL is shown in Figure 15-18 and described in Table 15-17.

Figure 15-18. DLL Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
DLL							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 15-17. DLL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	
7-0	DLL	R/W	0h	The 8 least-significant bits (LSBs) of the 16-bit divisor for generation of the baud clock in the baud rate generator.

15.3.1.11 DLH Register (offset = 1B12h) [reset = 0h]

DLH is shown in Figure 15-19 and described in Table 15-18.

Two 8-bit register fields (DLL and DLH), called divisor latches, store the 16-bit divisor for generation of the baud clock in the baud generator. The latches are in DLH and DLL. DLH holds the most-significant bits of the divisor, and DLL holds the least-significant bits of the divisor.

Figure 15-19. DLH Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							
7	6	5	4	3	2	1	0
DLH							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 15-18. DLH Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	
7-0	DLH	R/W	0h	The 8 most-significant bits (MSBs) of the 16-bit divisor for generation of the baud clock in the baud rate generator.

15.3.1.12 PWREMU_MGMT Register (offset = 1B18h) [reset = 0h]

PWREMU_MGMT is shown in Figure 15-20 and described in Table 15-19.

Used to reset the receiver and transmitter of the UART and as well as to enable its FREE mode.

Figure 15-20. PWREMU_MGMT Register

15	14	13	12	11	10	9	8
Reserved	UTRST	URRST	Reserved				
R-0h	R/W-0h	R/W-0h	R-0h				

7	6	5	4	3	2	1	0
Reserved							FREE
R-0h							R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 15-19. PWREMU_MGMT Register Field Descriptions

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	
14	UTRST	R/W	0h	UART transmitter reset. Resets and enables the transmitter 0x0 = RESET: Transmitter is disabled and in reset state 0x1 = ENABLE: Transmitter is enabled
13	URRST	R/W	0h	UART receiver reset. Resets and enables the receiver 0x0 = RESET: Receiver is disabled and in reset state 0x1 = ENABLE: Receiver is enabled
12-1	Reserved	R	0h	
0	FREE	R/W	0h	Free-running enable mode bit. This bit determines the emulation mode functionality of the UART. In suspended mode, the UART can handle register read/write requests, but does not generate any transmission/reception, interrupts or events. 0x0 = STOP: If a transmission is not in progress, the UART stops immediately. If a transmission is in progress, the UART stops after completion of the one word transmission 0x1 = RUN: Free-running mode is enabled; UART continues to run normally

Host Port Interface (UHPI)

Topic **Page**

16.1 Introduction

The universal host port interface (UHPI) provides a parallel port interface through which an external host processor can directly access the processor's resources (configuration and program/data memories). The external host device is asynchronous to the CPU clock and functions as a master to the UHPI interface. The UHPI enables a host device and the processor to exchange information via internal memory of the DSP. Dedicated address (UHPIA) and data (UHPID) registers within the UHPI provide the data path between the external host interface and the processor resources. An UHPI control register (UHPIC) is available to the host and the CPU for various configuration and interrupt functions.

16.1.1 Purpose of the Peripheral

The UHPI enables an external host processor (host) to directly access program/data memory on the device using a parallel interface. The primary purpose is to provide a mechanism to move data to and from the device. In addition to data transfer, the host can also use the UHPI to bootload the device by downloading program and data information to the processor's memory after power-up.

16.1.2 Features

The UHPI supports the following features:

- Multiplexed address/data
- 16-bit-wide host data bus interface
- Internal data bursting using 8-word read and write first-in, first-out (FIFO) buffers
- UHPI control register (UHPIC) accessible by the external host
- UHPI address register (UHPIA) accessible by the external host
- Separate UHPI address registers for read (UHPIAR) and write (UHPIAW) with configurable option for operating as a single UHPI address register
- UHPI data register (UHPID)/FIFOs providing data-path between external host interface and CPU resources
- Multiple strobes and control signals to allow flexible host connection
- Asynchronous UHPI_HRDY output to allow the UHPI to insert wait states to the host
- Software control of data prefetching to the UHPID/FIFOs
- Processor-to-Host interrupt output signal controlled by UHPIC accesses
- Host-to-Processor interrupt controlled by UHPIC accesses
- Register controlled UHPIA and UHPIC ownership and FIFO timeout
- Memory-mapped peripheral identification register (PID)

16.1.3 Functional Block Diagram

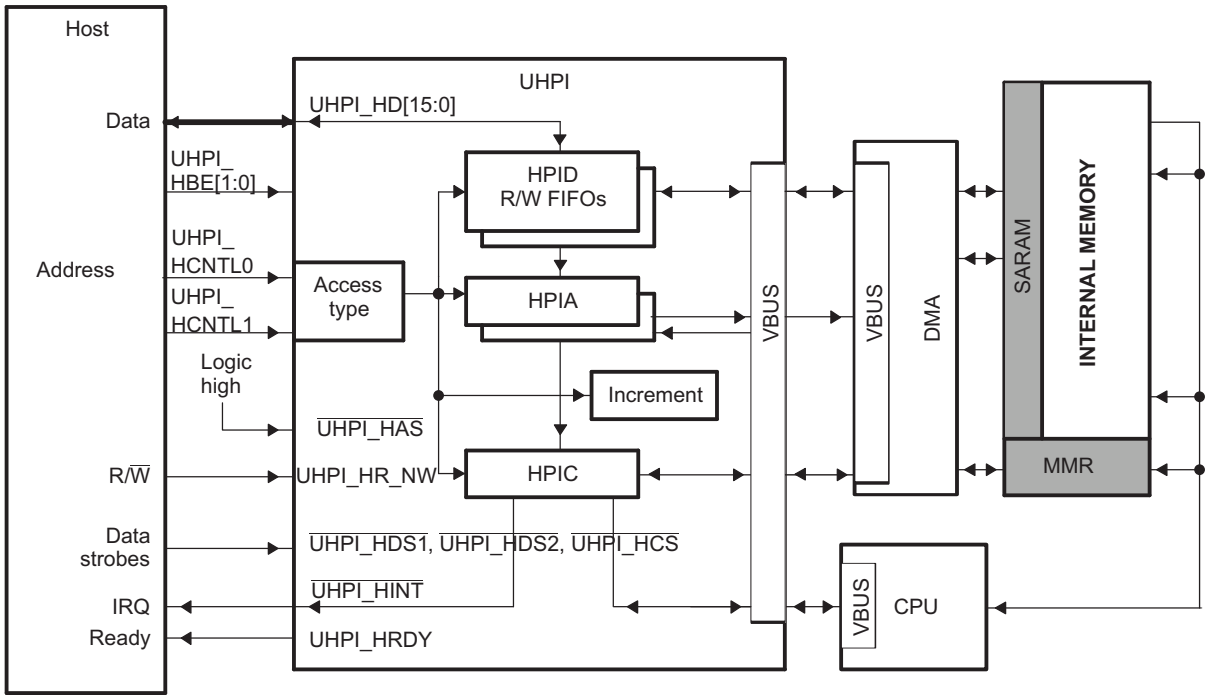
Figure 16-1 is a high-level block diagram showing how the UHPI connects a host (left side of figure) and the processor internal memory (right side of figure). Host activity is asynchronous to the internal processor clock that drives the UHPI. The host functions as a master to the UHPI. When UHPI resources are temporarily busy or unavailable, the UHPI communicates this to the host by deasserting the UHPI ready (UHPI_HRDY) output signal.

The UHPI supports multiplexed operation meaning the data bus is used for both address and data. Each host cycle consists of two consecutive 16-bit transfers. When the host drives an address on the bus, the address is stored in a 32-bit address register (UHPIA) in the UHPI, so that the bus can then be used for data. The UHPI contains two address registers (UHPIAR and UHPIAW), which can be used as separate address registers for read accesses and write accesses (for details, see Section 16.2.5.1).

A control register (UHPIC) is accessible by the CPU and the host. The CPU uses UHPIC to send an interrupt request to the host, to clear an interrupt request from the host, and to monitor the UHPI. The host uses UHPIC to configure and monitor the UHPI, to send an interrupt request to the CPU, and to clear an interrupt request from the CPU.

Data flow between the host and the UHPI uses a temporary storage register, the 32-bit data register (UHPID). Data arriving from the host is held in UHPID until the data can be stored elsewhere in the processor. Data to be sent to the host is held in UHPID until the UHPI is ready to perform the transfer. When address autoincrementing is used, read and write FIFOs are used to store burst data. If autoincrementing is not used, the FIFO memory acts as a single register (only one location is used).

Figure 16-1. UHPI Block Diagram



16.1.4 Industry Standard(s) Compliance Statement

The UHPI is not an industry standard interface that is developed and monitored by an international organization. It is a generic parallel interface that can be configured to gluelessly interface to a variety of parallel devices.

16.1.5 Terminology Used in This Document

The following is a brief explanation of some terms used in this document:

Term	Meaning
CPU	DSP CPU
host	External host device
processor	Entire system-on-chip

16.2 Architecture

16.2.1 Memory Map

The UHPI can be used by the host to access on-chip device memory, peripheral, and memory-mapped registers. See your device-specific data manual for more detailed information.

16.2.2 Signal Descriptions

Table 16-1 shows the a description of the UHPI signals.

Table 16-1. UHPI Pins

Pin	Type	Host Connection	Function
UHPI_HCNTL[1:0]	I	Address or control pins	**UHPI access control inputs**. The logic level of these pins is latched–in on the falling edge of UHPI_HAS or internal HSTRB. The four binary states of these pins determine the access mode of the current transfer (HPIC, HPID with increment, HPIA, HPID).
UHPI_HCS	I	Chip select pin	**UHPI chip select**. UHPI_HCS must be low for the UHPI to be selected by the host. UHPI_HCS can be kept low between accesses. UHPI_HCS normally precedes an active UHPI_HDS (data strobe) signal, but can be connected to a UHPI_HDS pin for simultaneous select and strobe activity.
UHPI_HR_NW	I	R/W strobe pin	**UHPI read/write**. Indicates to the UHPI on the falling edge of UHPI_HAS or internal HSTRB whether the current access is a read or write operation. Driving UHPI_HR_NW high indicates the transfer is a read from the UHPI, while driving UHPI_HR_NW low indicates a write to the UHPI.
UHPI_HHWIL	I	Address or control pins	**Halfword identification control input**. This bit identifies the first and second half-words of a dual half-word cycle operation. HHWIL=0 identifies the first cycle and HHWIL=1 identifies the second cycle.
UHPI_HAS	I	None	**Address strobe**. Connect to logic high.
UHPI_HINT	O/Z	Host interrupt pin	**Host Interrupt**. The DSP can interrupt the host device by writing a 1 to the HINT bit of the UHPIC register. Before subsequent UHPI_HINT interrupts can occur, must clear previous interrupts by writing a 1 to the HINT bit of the UHPIC register. This pin is active-low (that is, when an interrupt is asserted from the host, the state of this signal is low) and inverted from the HINT bit value in the UHPIC register. The pin is Hi-Z when UHPI is in ZPOR reset or when HPIENA=0.
UHPI_HDS1 and UHPI_HDS2	I	Read strobe and write strobe pins or any data strobe pin	**UHPI data strobe pins**. These pins are used for strobing data in and out of the UHPI (for data strobing details, see Section 16.2.5.3). The direction of the data transfer depends on the logic level of the UHPI_HR_NW signal. The UHPI_HDS signals are also used to latch control information on the falling edge. During a UHPID write access, data is latched into the UHPID register on the rising edge of UHPI_HDS. During read operations, these pins act as output-enable pins of the host data bus.
UHPI_HD[15:0]	I/O/Z	Data bus	**UHPI host data bus**. The UHPI data bus carries the data to and from the UHPI module. These pins are Hi-Z when UHPI is in ZPOR reset, when there are no read accesses occurring, or when HPIENA=0.
UHPI_HRDY	O/Z	Asynchronous ready input	**UHPI-ready signal**. When the UHPI drives UHPI_HRDY high, the host has permission to complete the current host cycle. When the UHPI drives UHPI_HRDY low, the UHPI is not ready for the current host cycle to complete. The pin is Hi-Z when UHPI is in ZPOR reset or when HPIENA=0.
UHPI_HBE[1:0]	I	Host byte enables	**UHPI byte enables.** These active-low byte enables perform single byte writes to CPU memory if supported. These enables only apply to HPID and FIFO writes and not to HPIA, HPIC or XHPIA accesses.

16.2.3 Pin Multiplexing

Extensive pin multiplexing is used to accommodate the largest number of peripheral functions in the smallest possible package. Pin multiplexing is controlled using a combination of hardware configuration at device reset and software programmable register settings. See the EBSR Register in Section 1.7.3.1 for how to select the pins for UHPI mode.

16.2.4 Protocol Description

The UHPI does not conform to any industry standard protocol.

16.2.5 Operation

16.2.5.1 Using the Address Registers

The UHPI contains two 32-bit address registers: one for read operations (UHPIAR) and one for write operations (UHPIAW). These roles are unchanging from the viewpoint of the UHPI logic.

However, unlike the UHPI logic, the host can choose how to interact with the two UHPI address registers. Using the DUAL_UHPIA bit in the UHPI control register (UHPIC), the host determines whether UHPIAR and UHPIAW act as a single 32-bit register (single-UHPIA mode) or as two independent 32-bit registers (dual-UHPIA mode).

Note that the addresses loaded into the UHPI address registers must be byte addresses, and must be 32-bit word aligned (with the least-significant two bits equal to zero), for use in addressing memory space within the DSP.

16.2.5.1.1 Single-UHPIA Mode

When DUAL_UHPIA = 0 in UHPIC, UHPIAR and UHPIAW become a single UHPI address register (UHPIA) from the perspective of the host. In this mode:

- A host UHPIA write cycle (UHPI_HCNTL[1:0] = 10b, UHPI_HR_NW = 0) updates UHPIAR and UHPIAW with the same value.
- Both UHPI address registers are incremented during autoincrement read/write cycles (UHPI_HCNTL[1:0] = 01b).
- An UHPIA read cycle (UHPI_HCNTL[1:0] = 10b, UHPI_HR_NW = 1) returns the content of UHPIAR, which should be identical to the content of UHPIAW.

To maintain consistency between the contents of UHPIAR and UHPIAW, the host should always reinitialize the UHPI address registers after changing the state of the DUAL_UHPIA bit. In addition, when DUAL_UHPIA = 0, the host must always reinitialize the UHPI address registers when it changes the data direction (from an UHPID read cycle to an UHPID write cycle, or conversely). Otherwise, the memory location accessed by the UHPI DMA logic might not be the location intended by the host.

16.2.5.1.2 Dual-UHPIA Mode

When DUALHPIA = 1 in UHPIC, UHPIAR and UHPIAW are two, independent UHPI address registers from the perspective of the host. In this mode:

- A host UHPIA access (UHPI_HCNTL[1:0] = 10b) reads or updates either UHPIAR or UHPIAW, depending on the value of the UHPIA read/write select (UHPIA_RW_SEL) bit in UHPIC. This bit is programmed by the host. While UHPIA_RW_SEL = 1, only UHPIAR is read or updated by the host. While UHPIA_RW_SEL = 0, only UHPIAW is read or updated by the host. The UHPIA_RW_SEL bit is only meaningful in the dual-UHPIA mode.

  Note: The UHPIA_RW_SEL bit does not affect the UHPI DMA logic. Regardless of the value of UHPIA_RW_SEL, the UHPI DMA logic uses UHPIAR when reading from memory and UHPIAW when writing to memory.
- A host UHPID access with autoincrementing (UHPI_HCNTL[1:0] = 01b) causes only the relevant UHPIA value to be incremented to the next consecutive memory address. In an autoincrement read cycle, UHPIAR is incremented after it has been used to perform the current read from memory. In an autoincrement write cycle, UHPIAW is incremented after it has been used for the write operation.

16.2.5.2 UHPI Configuration and Data Flow

The host accomplishes a multiplexed access in the following manner:

1. The host writes to the UHPI control register (UHPIC) to properly configure the UHPI. Typically, this means programming the halfword order bit (HWOB) and the UHPIA-related bits (DUAL_UHPIA and UHPIA_RW_SEL). This step is normally performed once before the initial data access.
2. The host writes the desired internal memory address to an address register (UHPIAR and/or UHPIAW). For an introduction to the two UHPI address registers and the two ways the host can interact with them, see Section 16.2.5.1.
3. The host reads from or writes to the data register (UHPID). Data transfers between UHPID and the internal memory of the device are handled by the UHPI DMA logic.

Each step of the access uses the same bus. Therefore, the host must drive the appropriate levels on the UHPI_HCNTL1 and UHPI_HCNTL0 signals to indicate which register is to be accessed. The host must also drive the appropriate level on the UHPI_HR_NW signal to indicate the data direction (read or write) and must drive other control signals as appropriate. When UHPI resources are temporarily busy or unavailable, the UHPI can communicate this to the host by deasserting the UHPI-ready (UHPI_HRDY) output signal.

When performing an access, the UHPI first latches the levels on UHPI_HCNTL[1:0], UHPI_HR_NW, and other control signals. This latching can occur on the falling edge of the internal strobe signal (for details, see Section 16.2.5.3). After the control information is latched, the UHPI initiates an access based on the control signals.

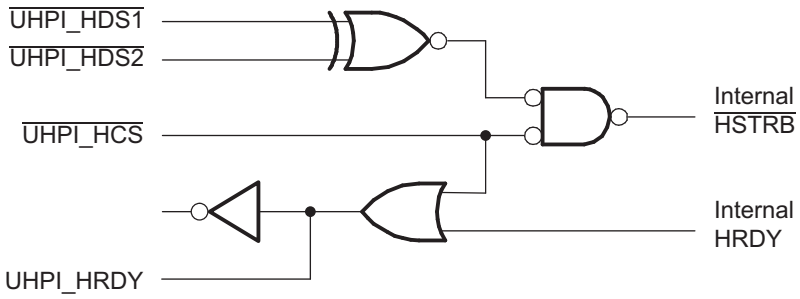
If the host wants to read data from internal resources (see Section 16.2.1), the UHPI DMA logic reads the resource address from UHPIAR and retrieves the data from the addressed memory location. When the data has been placed in UHPID, the UHPI drives the data onto its UHPI_HD bus. The UHPI_HRDY signal informs the host whether the data on the UHPI_HD bus is valid (UHPI_HRDY high) or not valid yet (UHPI_HRDY low). When the data is valid, the host should latch the data and drive the connected data strobe ($\overline{\text{UHPI\_HDS1}}$ or $\overline{\text{UHPI\_HDS2}}$) inactive, which, in turn, will cause the internal strobe (internal $\overline{\text{HSTRB}}$) signal to transition from low to high.

If the host wants to write data to internal resources (see Section 16.2.1), the operation is similar. After the host determines that the UHPI is ready to latch the data (UHPI_HRDY is low), it must cause internal $\overline{\text{HSTRB}}$ to transition from low to high, which causes the data to be latched into UHPID. Once the data is in UHPID, the UHPI DMA logic reads the memory address from UHPIAW and transfers the data from UHPID to the addressed memory location.

16.2.5.3 UHPI_HDS2, UHPI_HDS1, and UHPI_HCS: Data Strobing and Chip Selection

As shown in Figure 16-2, the strobing logic is a function of three key inputs: the chip select pin (UHPI_HCS) and two data strobe signals (UHPI_HDS1 and UHPI_HDS2). The internal strobe signal, which is referred to as internal HSTRB throughout this document, functions as the actual strobe signal inside the UHPI. UHPI_HCS must be low (UHPI selected) during strobe activity on the UHPI_HDS pins. If UHPI_HCS remains high (UHPI not selected), activity on the UHPI_HDS pins is ignored.

Figure 16-2. UHPI Strobe and Select Logic



Strobe connections between the host and the UHPI depend in part on the number and types of strobe pins available on the host. Table 16-2 describes some options for connecting to the UHPI_HDS pins.

Notice in Figure 16-2 that UHPI_HRDY is also gated by UHPI_HCS. If UHPI_HCS goes high (UHPI not selected), UHPI_HRDY goes high, regardless of whether the current internal transfer is completed in the device.

NOTE: The UHPI_HCS input and one UHPI_HDS strobe input can be tied together and driven with a single strobe signal from the host. This technique selects the UHPI and provides the strobe, simultaneously. When using this method, be aware that UHPI_HRDY is gated by UHPI_HCS as previously described.

It is not recommended to tie both UHPI_HDS1 and UHP

I_HDS2 to static logic levels and use UHPI_HCS as a strobe.

Table 16-2. Options for Connecting Host and UHPI Data Strobe Pins

Available Host Data Strobe Pins	Connections to UHPI Data Strobe Pins
Host has separate read and write strobe pins, both active-low	Connect one strobe pin to UHPI_HDS1 and the other to UHPI_HDS2 (1). Since such a host might not provide a R/W line, take care to satisfy UHPI_HR_NW timings as stated in your device-specific data manual. This could possibly be done using a host address line.
Host has separate read and write strobe pins, both active-high	Connect one strobe pin to UHPI_HDS1 and the other to UHPI_HDS2 (1). Since such a host might not provide a R/W line, take care to satisfy UHPI_HR_NW timings as stated in your device-specific data manual. This could possibly be done using a host address line.
Host has one active-low strobe pin	Connect the strobe pin to UHPI_HDS1 or UHPI_HDS2, and connect the other pin to logic-level 1.
Host has one active-high strobe pin	Connect the strobe pin to UHPI_HDS1 or UHPI_HDS2, and connect the other strobe pin to logic-level 0.

(1) The UHPI_HR_NW signal could be driven by a host address line in this case.

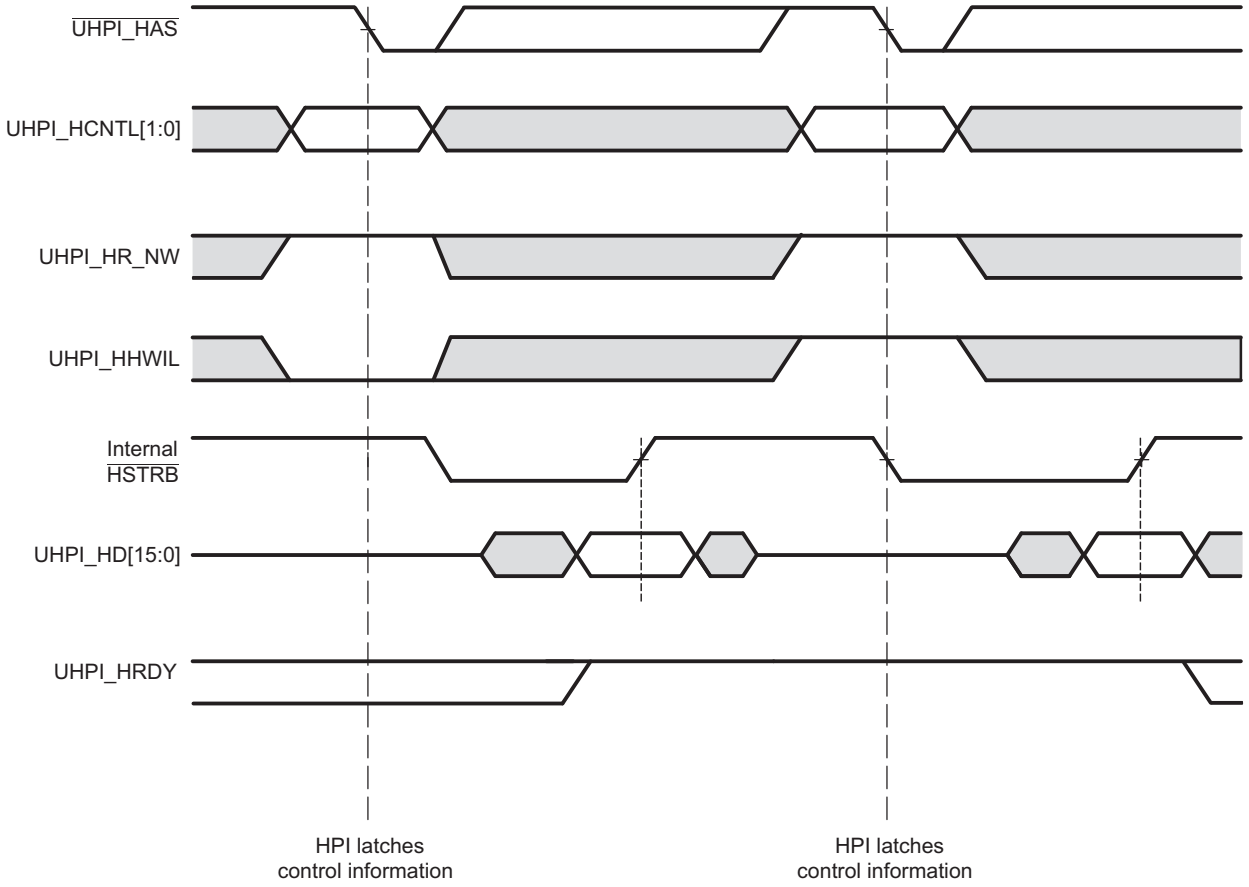
16.2.5.4 Host Accesses Using UHPI_HAS

The host address strobe (UHPI_HAS) is only valid in multiplexed mode. The UHPI_HAS signal supports host processors with multiplexed address and data buses. The falling edge of UHPI_HAS is used to latch the UHPI_HCNTL[1:0], UHPI_HR_NW and UHPI_HHWIL states into the UHPI.

First, the host selects the cycle type by driving the UHPI_HCNTL[1:0], UHPI_HR_NW and UHPI_HHWIL pins to select the desired access mode. The Internal HSTRB is derived from UHPI_HCS, UHPI_HDS1 and UHPI_HDS2, as described in Section 16.2.5.3.

The high-to-low transition of UHPI_HAS must precede the falling edge of the internal HSTRB. The UHPI_HAS input is not gated by UHPI_HCS, therefore allowing time for the host to perform the subsequent access. The UHPI_HAS signal may be brought high after UHPI_HDS is driven low, indicating the data access is about to occur. UHPI_HAS is not required to be driven high at any time during the cycle but must eventually transition high before performing another cycle with different values for UHPI_HCNTL[1:0], UHPI_HR_NW, or UHPI_HHWIL. Most of the timing diagrams in this document assume that UHPI_HAS is not used. Any valid multiplexed mode cycles may also be performed using UHPI_HAS as described in this section. Figure 16-3 illustrates a read-access using UHPI_HAS.

Figure 16-3. Host Read Using UHPI_HAS (Dual Half-Word Cycle)

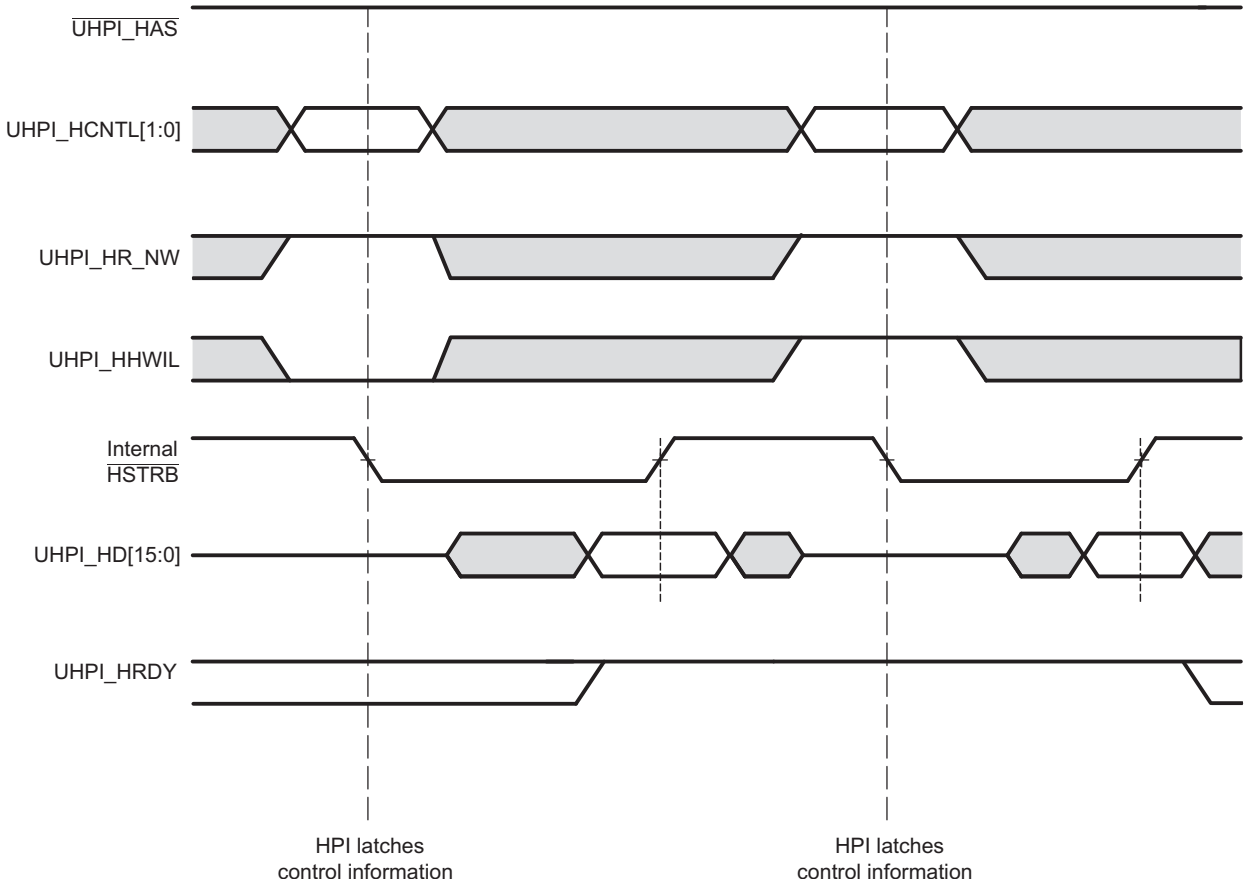


16.2.5.5 Host Accesses without UHPI_HAS

In cases where the host processor has dedicated signals (or bit I/O) capable of driving the UHPI_HCNTL[1:0] pins, the UHPI_HAS signal is not required. These dedicated signals can connect directly to the UHPI_HCNTL[1:0] and UHPI_HR_NW pins. This connection considerably simplifies access because there is no setup and strobing of the UHPI_HCNTL[1:0] data relative to UHPI_HAS. In this case, the falling edge of the internal HSTRB is used to latch the UHPI_HCNTL[1:0], UHPI_HR_NW and UHPI_HHWIL states into the UHPI. The internal HSTRB is derived from UHPI_HCS, UHPI_HDS1, and UHPI_HDS2, as described in Section 16.2.5.3.

When UHPI_HAS is not used, it is expected to be tied to logic high. Figure 16-4 shows the read-access sequence with UHPI_HAS tied high. All of the following timing diagrams in this document assume that UHPI_HAS is not used. Any valid multiplexed mode cycles may also be performed using UHPI_HAS, as described in Section 16.2.5.4.

Figure 16-4. Host Read without UHPI_HAS (Dual Half-Word Cycle)



16.2.5.6 UHPI_HCNTL[1:0] and UHPI_HR_NW: Indicating the Cycle Type

The cycle type consists of:

- The access type that the host selects by driving the appropriate levels on the UHPI_HCNTL[1:0] pins of the UHPI. Table 16-3 describes the four available access types.
- The transfer direction that the host selects with the UHPI_HR_NW pin. The host must drive the UHPI_HR_NW signal high (read) or low (write).

A summary of cycle types is in Table 16-4. The UHPI samples the UHPI_HCNTL levels at the falling edge of the internal strobe signal $\overline{\text{HSTRB}}$.

Table 16-3. Access Types Selectable With the UHPI_HCNTL Signals

UHPI_HCNTL1	UHPI_HCNTL0	Access Type
0	0	**UHPIC access.** The host requests to access the UHPI control register (UHPIC).
0	1	**UHPID access with autoincrementing.** The host requests to access the UHPI data register (UHPID) and to have the appropriate UHPI address register (UHPIAR and/or UHPIAW) automatically incremented by 1 after the access.
1	0	**UHPIA access.** The host requests to access the appropriate UHPI address register (UHPIAR and/or UHPIAW).
1	1	**UHPID access without autoincrementing.** The host requests to access the UHPI data register (UHPID) but requests no automatic post-increment of the UHPI address register.

Table 16-4. Cycle Types Selectable With the UHPI_HCNTL and UHPI_HR_NW Signals

UHPI_HCNTL1	UHPI_HCNTL0	UHPI_HR_NW	Cycle Type
0	0	0	UHPIC write cycle
0	0	1	UHPIC read cycle
0	1	0	UHPID write cycle with autoincrementing
0	1	1	UHPID read cycle with autoincrementing
1	0	0	UHPIA write cycle
1	0	1	UHPIA read cycle
1	1	0	UHPID write cycle without autoincrementing
1	1	1	UHPID read cycle without autoincrementing

16.2.5.7 UHPI_HHWIL: Identifying the First and Second Halfwords in Multiplexed Mode Transfers

Each host cycle consists of two consecutive halfword transfers. For each transfer, the host must specify the cycle type with UHPI_HCNTL[1:0] and UHPI_HR_NW, and the host must use UHPI_HHWIL to indicate whether the first or second halfword is being transferred. For UHPID and UHPIA accesses, UHPI_HHWIL must always be driven low for the first halfword transfer and high for the second halfword transfer. Results are undefined if the sequence is broken. For examples of using UHPI_HHWIL, see Section 16.2.5.8.

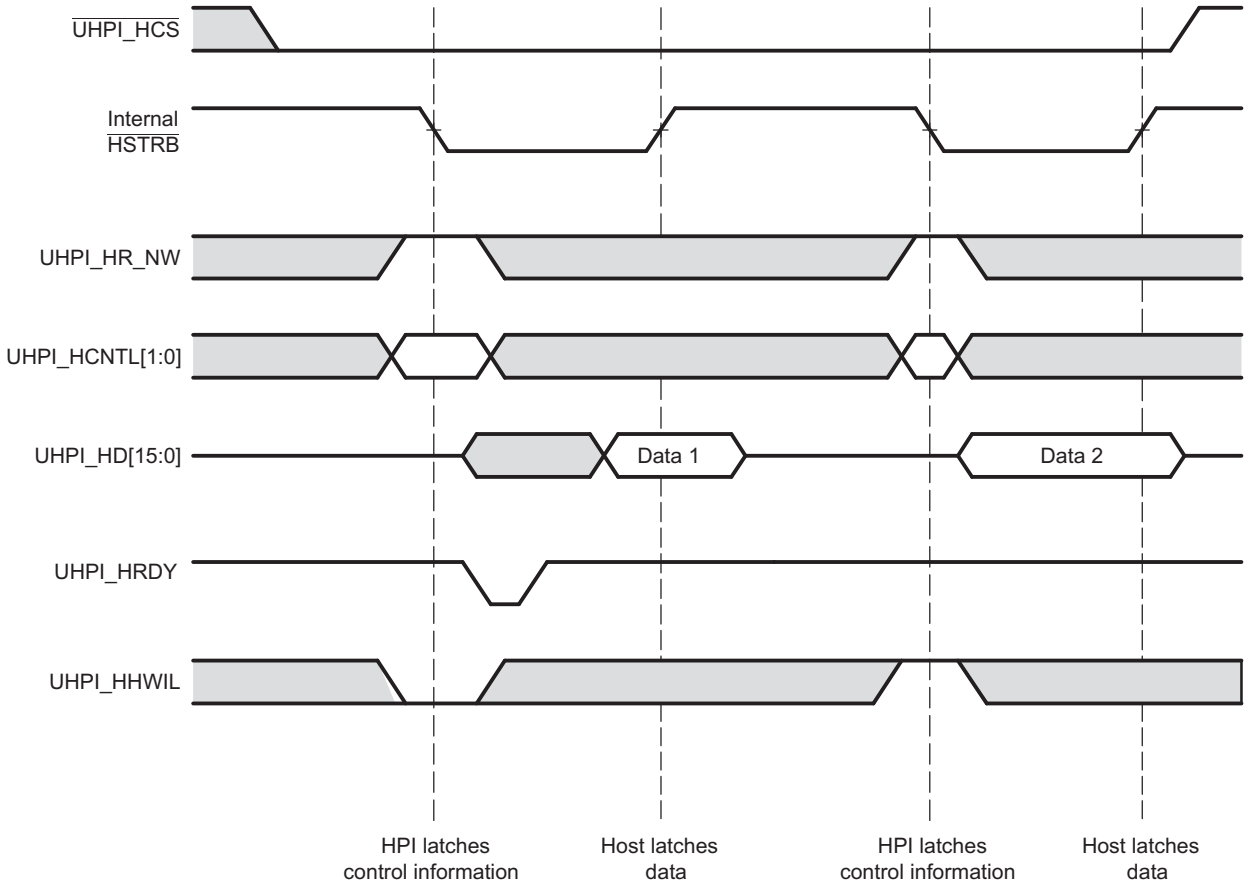
When the host sends the two halfwords of a 32-bit word in this manner, the host can send the most-significant and the least-significant halfwords of the word in either order (most-significant halfword first or most-significant halfword second). However, the host must inform the UHPI of the selected order before beginning the host cycle. This is done by programming the halfword order (HWOB) bit in UHPIC. Although HWOB is written at bit 0 in UHPIC, its current value is readable at both bit 0 and bit 8 (HWOBSTAT). Thus, the host can determine the current halfword order configuration by checking the least-significant bit of either half of UHPIC.

There is one case when the UHPI does not require a dual halfword access with UHPI_HHWIL low for the first halfword and UHPI_HHWIL high for the second halfword. This is the case when accessing the UHPIC register. When accessing UHPIC, the state of UHPI_HHWIL is ignored and the same 16-bit UHPIC register is accessed regardless of whether the host performs a single or dual access. For an example timing diagram of this case, see Section 16.2.5.9.

16.2.5.8 Performing a Multiplexed Access

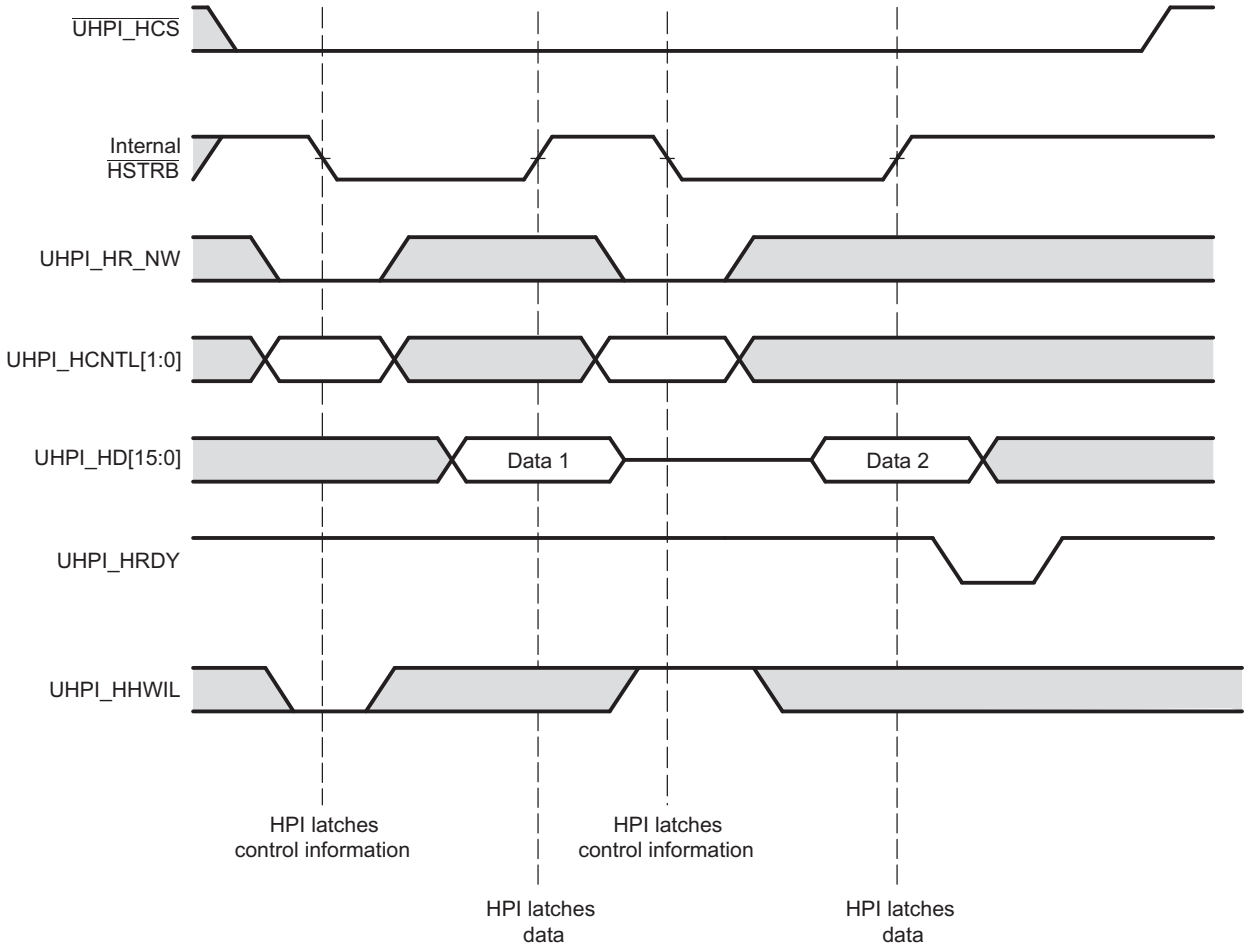
Figure 16-5 and Figure 16-6 show typical UHPI signal activity when performing a read and write transfer, respectively. In these cases, the falling edge of internal $\overline{\text{HSTRB}}$ is used to latch the UHPI_HCNTL[1:0], UHPI_HR_NW, and UHPI_HHWIL states into the UHPI. Internal $\overline{\text{HSTRB}}$ is derived from $\overline{\text{UHPI\_HCS}}$, $\overline{\text{UHPI\_HDS1}}$, and $\overline{\text{UHPI\_HDS2}}$ as described in Section 16.2.5.3.

Figure 16-5. Multiplexed-Mode Host Read Cycle



NOTE: Depending on the type of write operation (UHPID without autoincrementing, UHPIA, UHPIC, or UHPID with autoincrementing) and the state of the FIFO, transitions on UHPI_HRDY may or may not occur. For more information, see Section 16.2.5.10.

Figure 16-6. Multiplexed-Mode Host Write Cycle

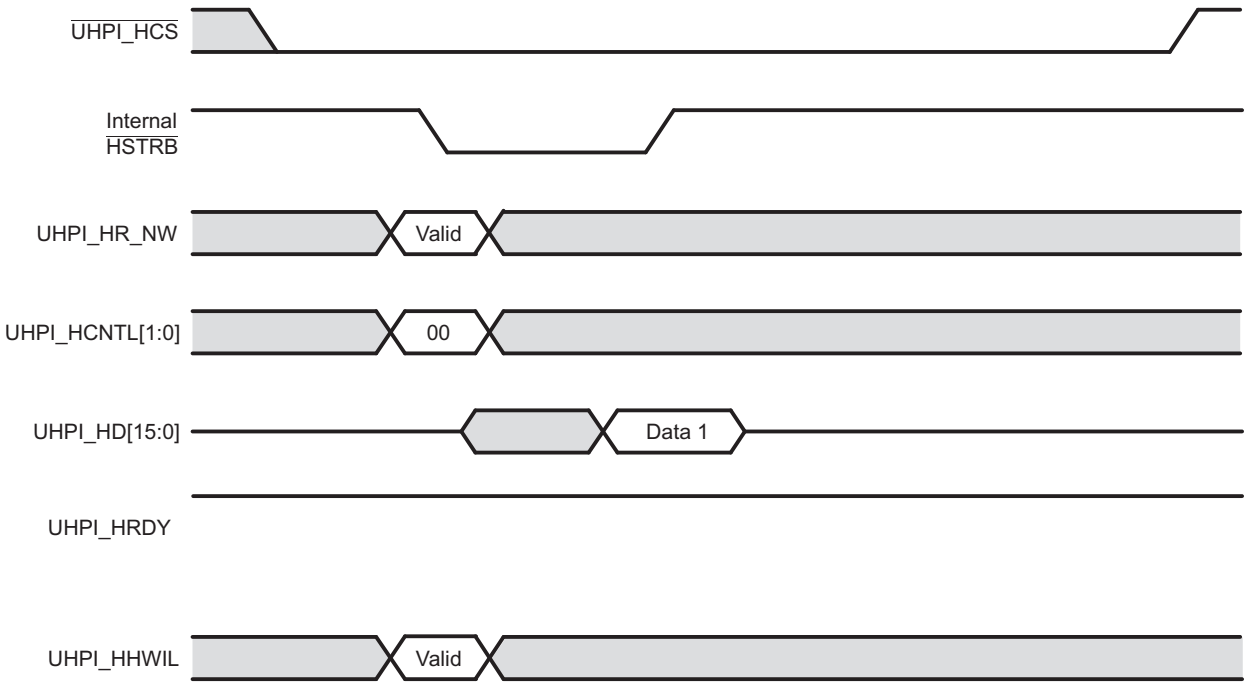


NOTE: Depending on the type of write operation (UHPID without autoincrementing, UHPIA, UHPIC, or UHPID with autoincrementing) and the state of the FIFO, transitions on UHPI_HRDY may or may not occur. For more information, see Section 16.2.5.10.

16.2.5.9 Single-Halfword UHPIC Cycle

Figure 16-7 shows the special case (see Section 16.2.5.7) when the host performs a single-halfword cycle to access the UHPIC. The state of UHPI_HHWIL is ignored and if a dual-halfword access is performed, then the same UHPIC register is accessed twice.

Figure 16-7. Multiplexed-Mode Single-Halfword UHPIC Cycle (Read or Write)



16.2.5.10 Hardware Handshaking Using the UHPI-Ready (UHPI_HRDY) Signal

The UHPI uses its ready signal, UHPI_HRDY, to tell the host whether it is ready to complete an access. During a read cycle, the UHPI is ready (UHPI_HRDY is high) when it has data available for the host. During a write cycle, the UHPI is ready (UHPI_HRDY is high) when it is ready to latch data from the host. If the UHPI is not ready, it can drive UHPI_HRDY low to insert wait states. These wait states indicate to the host that read data is not yet valid (read cycle) or that the UHPI is not ready to latch write data (write cycle). The number of wait states that must be inserted by the UHPI is dependent upon the state of the resource that is being accessed.

When the UHPI is not ready to complete the current cycle (UHPI_HRDY is low), the host can begin a new host cycle by forcing the UHPI to latch new control information. However, once the cycle has been initiated, the host must wait until UHPI_HRDY goes high before causing a rising edge on the internal strobe signal (internal $\overline{\text{HSTRB}}$) to complete the cycle. If internal $\overline{\text{HSTRB}}$ goes high when the UHPI is not ready, the cycle will be terminated with invalid data being returned (read cycle) or written (write cycle).

One reason the UHPI may drive UHPI_HRDY low is a not-ready condition in one of its first-in, first-out buffers (FIFOs). For example, any UHPID access that occurs while the write FIFO is full or the read FIFO is empty may result in some number of wait states being inserted by the UHPI. The FIFOs are explained in Section 16.2.5.11.

The following sections describe the behavior of UHPI_HRDY during UHPI register accesses. In all cases, the chip select signal, $\overline{\text{UHPI\_HCS}}$, must be asserted for UHPI_HRDY to go high.

16.2.5.10.1 UHPI_HRDY Behavior During Multiplexed-Mode Read Operations

Figure 16-8 shows an UHPIC (UHPI_HCNTL[1:0] = 00b) or UHPIA (UHPI_HCNTL[1:0] = 10b) read cycle. Neither an UHPIC read cycle nor an UHPIA read cycle causes UHPI_HRDY to go low. For this type of access, the state of UHPI_HHWIL is ignored, so if a dual halfword access is performed, the same register will be accessed twice.

Figure 16-8. UHPI_HRDY Behavior During an UHPIC or UHPIA Read Cycle in the Multiplexed Mode

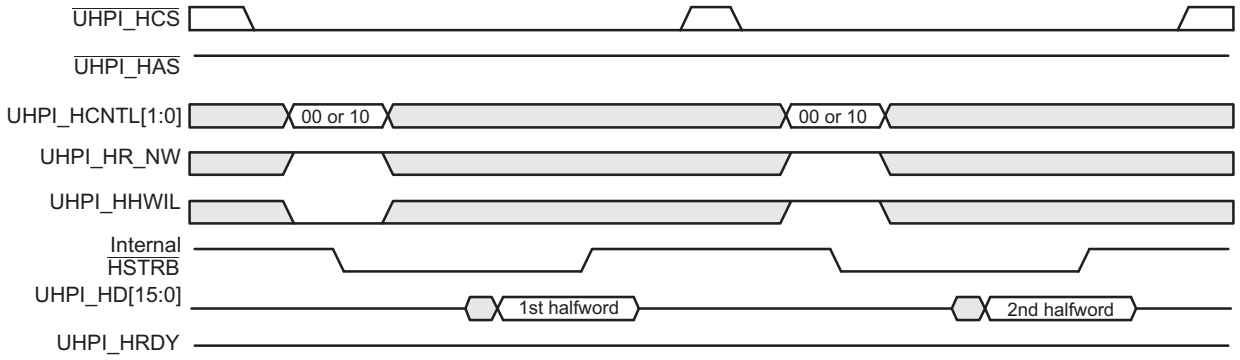


Figure 16-9 includes an UHPID read cycle without autoincrementing. The host writes the memory address during the UHPIA (UHPI_HCNTL[1:0] = 10b) write cycle, and the host reads the data during the UHPID (UHPI_HCNTL[1:0] = 11b) read cycle. UHPI_HRDY goes low for each UHPIA halfword access, but UHPI_HRDY goes low for only the first halfword access in each UHPID read cycle.

Figure 16-9. UHPI_HRDY Behavior During a Data Read Operation in the Multiplexed Mode (Case 1: UHPIA Write Cycle Followed by Nonautoincrement UHPID Read Cycle)

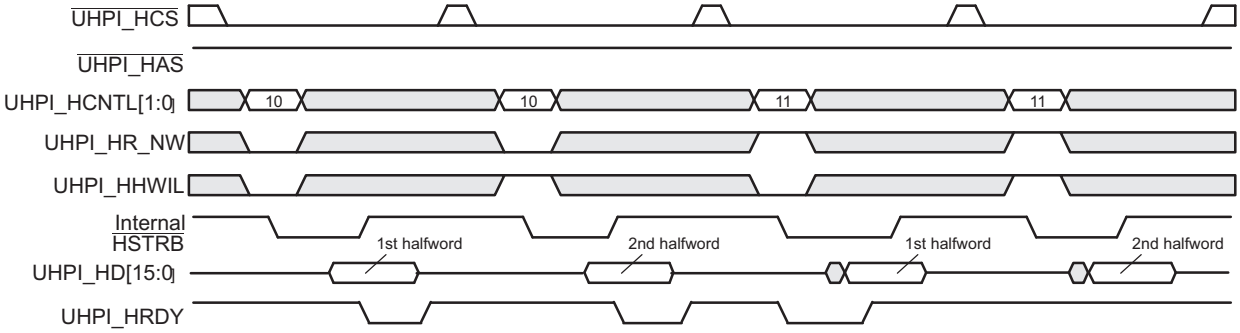
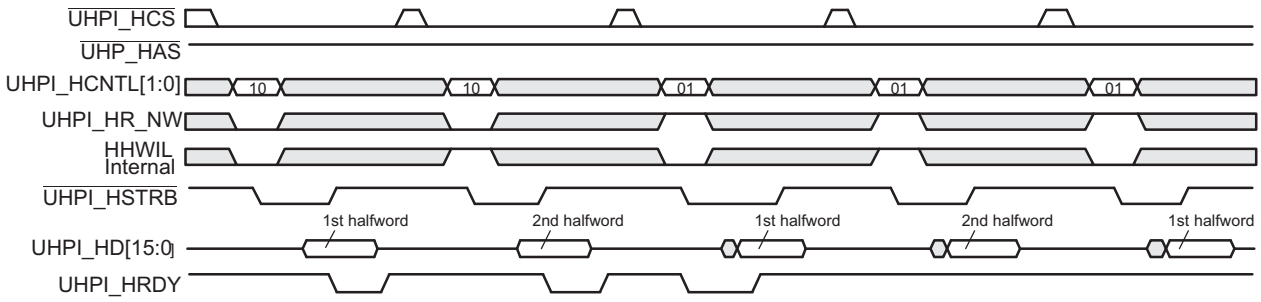


Figure 16-10 includes an autoincrement UHPID read cycle. The host writes the memory address while asserting UHPI_HCNTL[1:0] = 10b and reads the data while asserting UHPI_HCNTL[1:0] = 01b. During the first UHPID read cycle, UHPI_HRDY goes low for only the first halfword access, and subsequent UHPID read cycles do not cause UHPI_HRDY to go low.

Figure 16-10. UHPI_HRDY Behavior During a Data Read Operation in the Multiplexed Mode (Case 2: UHPIA Write Cycle Followed by Autoincrement UHPID Read Cycles)



16.2.5.10.2 UHPI_HRDY Behavior During Multiplexed-Mode Write Operations

Figure 16-11 shows an UHPIC (UHPI_HCNTL[1:0] = 00b) write cycle operation. An UHPIC write cycle does not cause UHPI_HRDY to go low and the state of UHPI_HHWIL is ignored. Firmware is not required to perform a dual access to access UHPIC.

Figure 16-11. UHPI_HRDY Behavior During an UHPIC Write Cycle in the Multiplexed Mode

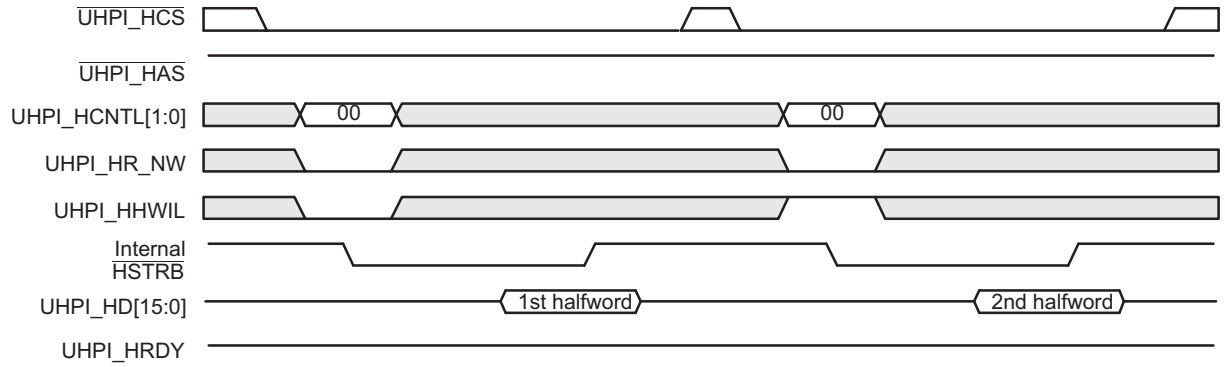


Figure 16-12 includes a UHPID write cycle without autoincrementing. The host writes the memory address while UHPI_HCNTL[1:0] = 10b and writes the data while UHPI_HCNTL[1:0] = 11b. During the UHPID write cycle, UHPI_HRDY goes low only for the second halfword access.

Figure 16-12. UHPI_HRDY Behavior During a Data Write Operation in the Multiplexed Mode (Case 1: No Autoincrementing)

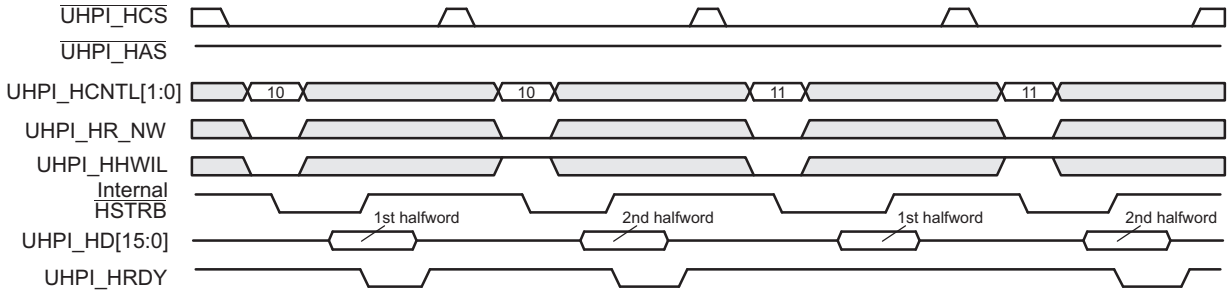


Figure 16-13 shows autoincrement UHPID write cycles when the write FIFO is empty prior to the UHPIA write. The host writes the memory address while UHPI_HCNTL[1:0] = 10b and writes the data while UHPI_HCNTL[1:0] = 01b. UHPI_HRDY does not go low during any of the UHPID write cycles until the FIFO is full.

Figure 16-13. UHPI_HRDY Behavior During a Data Write Operation in the Multiplexed Mode (Case 2: Autoincrementing Selected, FIFO Empty Before Write)

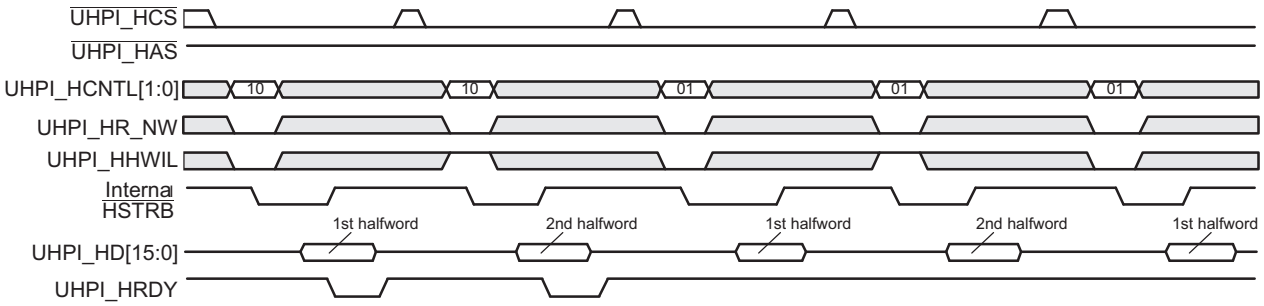
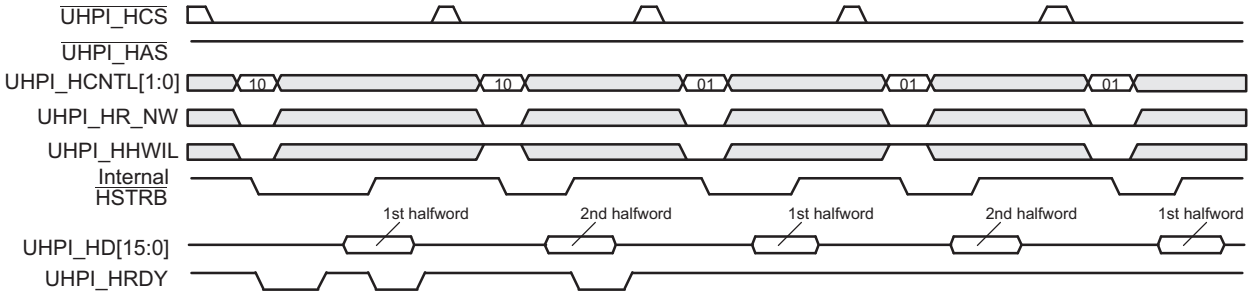


Figure 16-14 shows a case similar to that of Figure 16-13. However, in the case of Figure 16-14, the write FIFO is not empty when the UHPIA access is made. UHPI_HRDY goes low twice for the first halfword access of the UHPIA write cycle. The first UHPI_HRDY low period is due to the nonempty FIFO. The data currently in the FIFO must first be written to the memory. This results in UHPI_HRDY going low immediately after the falling edge of the data strobe ($\overline{\text{HSTRB}}$). The second and third UHPI_HRDY low periods occur for the writes to the UHPIA. UHPI_HRDY remains high for the UHPID accesses.

Figure 16-14. UHPI_HRDY Behavior During a Data Write Operation in the Multiplexed Mode (Case 3: Autoincrementing Selected, FIFO Not Empty Before Write)

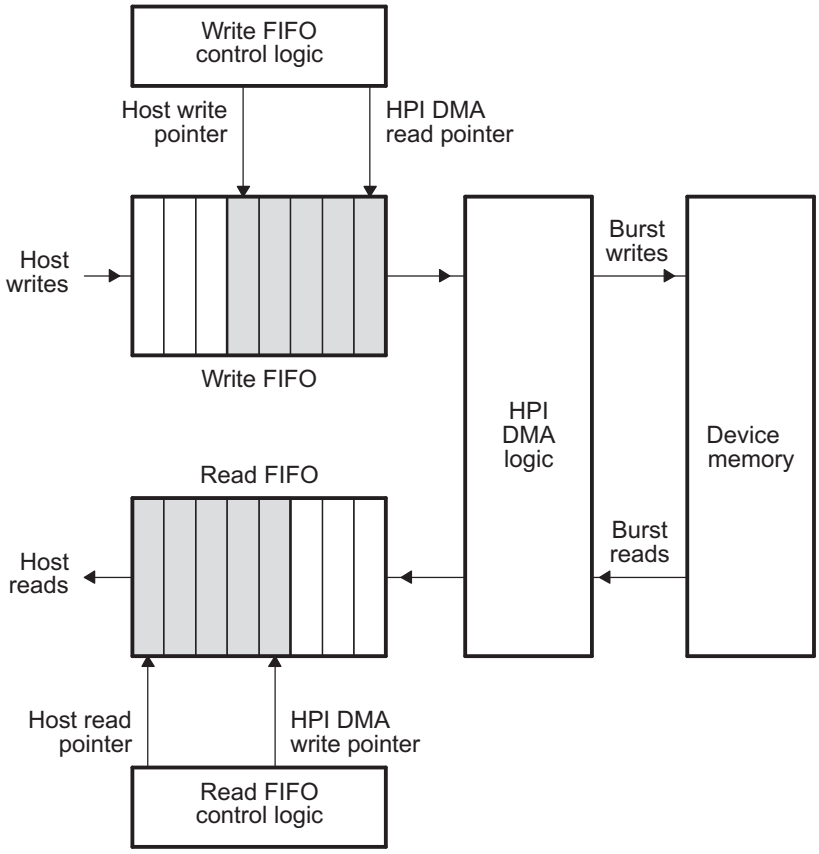


16.2.5.11 FIFOs and Bursting

The UHPI data register (UHPID) is a port through which the host accesses two first-in, first-out buffers (FIFOs). As shown in Figure 16-15, a read FIFO supports host read cycles, and a write FIFO supports host write cycles. Both read and write FIFOs are 8-words deep (each word is 32 bits). If the host is performing multiple reads or writes to consecutive memory addresses (autoincrement UHPID cycles), the FIFOs are used for bursting. The UHPI DMA logic reads or writes a burst of four words at a time when accessing one of the FIFOs.

Bursting is essentially invisible to the host because the host interface signaling is not affected. Its benefit to the host is that the UHPI_HRDY signal is deasserted less often when there are multiple reads or writes to consecutive addresses.

Figure 16-15. FIFOs in the UHPI



16.2.5.11.1 Read Bursting

When the host writes to the read address register (UHPIAR), the read FIFO is flushed. Any host read data that was in the read FIFO is discarded (the read FIFO pointers are reset). If an UHPI DMA write to the read FIFO is in progress at the time of a flush request, the UHPI allows this write to complete and then performs the flush.

Read bursting can begin in one of two ways: the host initiates an UHPID read cycle with autoincrementing, or the host initiates issues a FETCH command (writes 1 to the FETCH bit in UHPIC).

If the host initiates an UHPID read cycle with autoincrementing, the UHPI DMA logic performs two 4-word burst operations to fill the read FIFO. The host is initially held off by the deassertion of the UHPI_HRDY signal until data is available to be read from the read FIFO. Once data is available in the read FIFO, the host can read data from the read FIFO by performing subsequent reads of UHPID with autoincrementing. Once the initial read has been performed, the UHPI DMA logic continues to perform 4-word burst operations to consecutive memory addresses every time there are four empty word locations in the read FIFO. The UHPI DMA logic continues to prefetch data to keep the read FIFO full, until the occurrence of an event that causes a read FIFO flush (see Section 16.2.5.11.3).

As mentioned, the second way that read bursting may begin is with a FETCH command. The host should always precede the FETCH command with the initialization of the UHPIAR register or a nonautoincrement access, so that the read FIFO is flushed beforehand. When the host initiates a FETCH command, the UHPI DMA logic begins to prefetch data to keep the read FIFO full, as described in the previous paragraph. The FETCH bit in UHPIC does not actually store the value that is written to it; rather, the decoding of a host write of 1 to this bit is considered a FETCH command.

The FETCH command can be helpful if the host wants to minimize a stall condition on the interface. The host can initiate prefetching by writing 1 to the FETCH bit and later perform a read. The host can make use of the time it takes to load the read FIFO with read data, during which the UHPI was not ready, by using the CPU to service other tasks.

Both types of continuous or burst reads described in the previous paragraphs begin with a write to the UHPI address register, which causes a read FIFO flush. This is the typical way of initiating read cycles, because the initial read address needs to be specified.

NOTE: A UHPID read cycle without autoincrementing does not initiate any prefetching activity. Instead, it causes the read FIFO to be flushed and causes the UHPI DMA logic to perform a single-word read from the internal memory. As soon as the host activates a read cycle without autoincrementing, prefetching activity ceases until the occurrence of a FETCH command or an autoincrement read cycle.

16.2.5.11.2 Write Bursting

A write to the write address register (UHPIAW) causes the write FIFO to be flushed. This means that any write data in the write FIFO is forced to its destination in the internal memory (the UHPI DMA logic performs burst operations until the write FIFO is empty). When the FIFO has been flushed, the only action that will cause the UHPI DMA logic to perform burst writes is a host write to UHPID with autoincrementing. The initial host-write data is stored in the write FIFO. An UHPI DMA write is not requested until there are four words in the write FIFO. As soon as four words have been written to the FIFO via UHPID write cycles with autoincrementing, the UHPI DMA logic performs a 4-word burst operation to the internal memory. The burst operations continue as long as there are at least four words in the FIFO. If the FIFO becomes full (eight words are waiting in the FIFO), the UHPI holds off the host by deasserting UHPI_HRDY until at least one empty word location is available in the FIFO.

Because excessive time might pass between consecutive burst operations, the UHPI has a time-out counter. If there are fewer than four words in the write FIFO and the time-out counter expires, the UHPI DMA logic empties the FIFO immediately by performing a 2-word or 3-word burst, or a single-word write, as necessary. Every time new data is written to the write FIFO, the time-out counter is automatically reset to begin its count again. The time-out period is set to a value of 160.

NOTE: A UHPID write cycle without autoincrementing does not initiate any bursting activity. Instead, it causes the write FIFO to be flushed and causes the UHPI DMA logic to perform a single-word write to the internal memory. As soon as the host activates a write cycle without autoincrementing, bursting activity ceases until the occurrence of an autoincrement write cycle. A nonautoincrement write cycle always should be preceded by the initialization of UHPIAW or by another nonautoincrement access, so that the write FIFO is flushed beforehand.

16.2.5.11.3 FIFO Flush Conditions

When specific conditions occur within the UHPI, the read or write FIFO must be flushed to prevent the reading of stale data from the FIFOs. When a read FIFO flush condition occurs, all current host accesses and direct memory accesses (DMAs) to the read FIFO are allowed to complete. This includes DMAs that have been requested but not yet initiated. The read FIFO pointers are then reset, causing any read data to be discarded.

Similarly, when a write FIFO flush condition occurs, all current host accesses and DMAs to the write FIFO are allowed to complete. This includes DMAs that have been requested but not yet initiated. All posted writes in the FIFO are then forced to completion with a final burst or single-word write, as necessary.

If the host initiates an UHPID host cycle during a FIFO flush, the cycle is held off with the deassertion of UHPI_HRDY until the flush is complete and the FIFO is ready to be accessed.

The following conditions cause the read and write FIFOs to be flushed:

- Read FIFO flush conditions:
  - A value from the host is written to the read address register (UHPIAR).
  - The host performs an UHPID read cycle without autoincrementing.
- Write FIFO flush conditions:
  - A value from the host is written to the write address register (UHPIAW).
  - The host performs an UHPID write cycle without autoincrementing.
  - The write-burst time-out counter expires.

When operating with DUAL_UHPIA = 0, any read or write flush condition causes both read and write FIFOs to be flushed. In addition, the following scenarios cause both FIFOs to be flushed when DUAL_UHPIA = 0:

- The host performs a write to the UHPIA register.
- The host performs an UHPID write cycle with autoincrementing while the read FIFO is not empty (the read FIFO still contains data from prefetching or an UHPID read cycle with autoincrementing).
- The host performs an UHPID read cycle with autoincrementing while the write FIFO is not empty (there is still posted write data in the write FIFO).

This is useful in providing protection against reading stale data by reading a memory address when a previous write cycle has not been completed at the same address. Similarly, this protects against overwriting data at a memory address when a previous read cycle has not been completed at the same address.

When operating with DUAL_UHPIA = 1 (UHPIAR and UHPIAW are independent), there is no such protection. However, when DUAL_UHPIA = 1, data flow can occur in both directions without flushing both FIFOs simultaneously, thereby improving UHPI bandwidth.

16.2.5.11.4 FIFO Behavior When a Hardware Reset or Software Reset Occurs

A hardware reset (RESET pin driven low) or an UHPI software reset causes the FIFOs to be reset. The FIFO pointers are cleared, so that all data in the FIFOs are discarded. In addition, all associated FIFO logic is reset.

If a host cycle is active when a hardware or UHPI software reset occurs, the UHPI_HRDY signal is asserted (driven high), allowing the host to complete the cycle. When the cycle is complete, UHPI_HRDY is deasserted (driven low). Any access interrupted by a reset may result in corrupted read data or a lost write data (if the write does not actually update the intended memory or register). Although data may be lost, the host interface protocol is not violated. While either of reset condition is true, and the host is idle (internal $\overline{\text{HSTRB}}$ is held high), the FIFOs are held in reset, and host transactions are held off with an inactive UHPI_HRDY signal.

16.2.6 Reset Considerations

The UHPI has two reset sources: software reset and hardware reset.

16.2.6.1 Software Reset Considerations

The UHPI is not affected by a software reset issued by the emulator.

16.2.6.2 Hardware Reset Considerations

When the entire device is reset with the RESET pin:

- If the internal strobe signal, internal $\overline{\text{HSTRB}}$, is high (host is inactive), UHPI_HRDY is driven high and remains low until the reset condition is over.
- If internal $\overline{\text{HSTRB}}$ is low (host cycle is active), UHPI_HRDY is driven low, allowing the host to complete the cycle. When internal $\overline{\text{HSTRB}}$ goes high (cycle is complete), UHPI_HRDY is driven high and remains low until the reset condition is over. If the active cycle was a write cycle, the memory or register may not have been correctly updated. If the active cycle was a read cycle, the fetched value may not be valid.
- The UHPI registers are reset to their default values (see Section 16.6).
- The read and write FIFOs and the associated FIFO logic are reset (this includes a flush of the FIFOs).
- Host-to-CPU and CPU-to-host interrupts are cleared.

16.2.7 Initialization

The following steps are required to configure the UHPI after a hardware reset:

1. Perform the necessary device pin multiplexing setup. See Section 1.1 for pin selection/multiplexing at chip level.
2. Configure the UHPIENA and UHPIBYTEAD bits in the chip configuration 1 register (CFGCHIP1).
3. Choose how UHPIAR and UHPIAW will be controlled by configuring the DUAL_UHPIA bit in UHPIC.
4. Choose how halfword ordering will be handled by configuring the HWOB bit in UHPIC.
5. Choose how the UHPI will respond to emulation suspend events by configuring the FREE and SOFT bits in PWREMU_MGMT.
6. Choose the desired initial addresses and write the addresses to UHPIAW and UHPIAR, appropriately.
7. Release the UHPI logic from reset by clearing the UHPIRST bit in UHPIC.

The UHPI is now ready to perform data transactions.

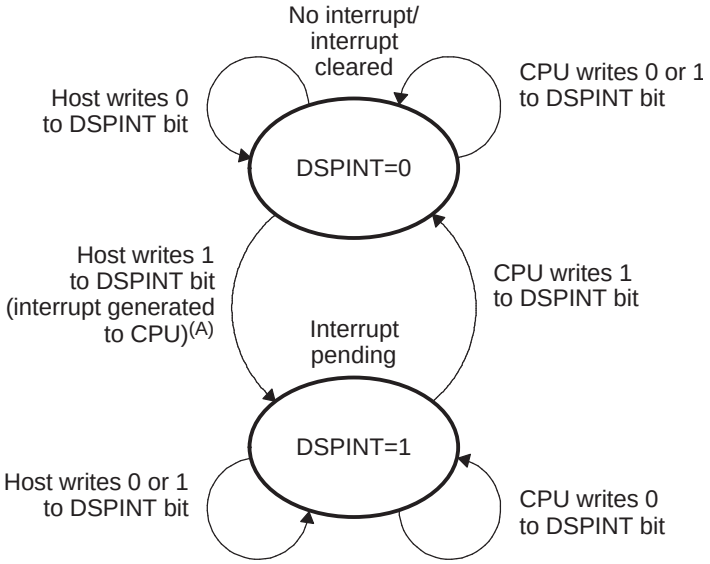
16.2.8 Interrupt Support

The host can interrupt the CPU via the DSPINT bit in UHPIC, as described in Section 16.2.8.1. The CPU can send an interrupt to the host by using the HINT bit in UHPIC, as described in Section 16.2.8.2.

16.2.8.1 DSPINT Bit: Host-to-CPU Interrupts

The DSPINT bit in UHPIC allows the host to send an interrupt request to the CPU. The use of the DSPINT bit is summarized in Figure 16-16.

Figure 16-16. Host-to-CPU Interrupt State Diagram



A When the DSPINT bit transitions from 0 to 1, an interrupt is generated to the CPU. No new interrupt can be generated until the CPU has cleared the bit (DSPINT = 0).

To interrupt the CPU, the host must:

1. Drive both UHPI_HCNTL1 and UHPI_HCNTL0 low to request a write to UHPIC.
2. Write 1 to the DSPINT bit in UHPIC.

When the host sets the DSPINT bit, the UHPI generates an interrupt pulse to the CPU. If this maskable interrupt is properly enabled in the CPU, the CPU executes the corresponding interrupt service routine (ISR).

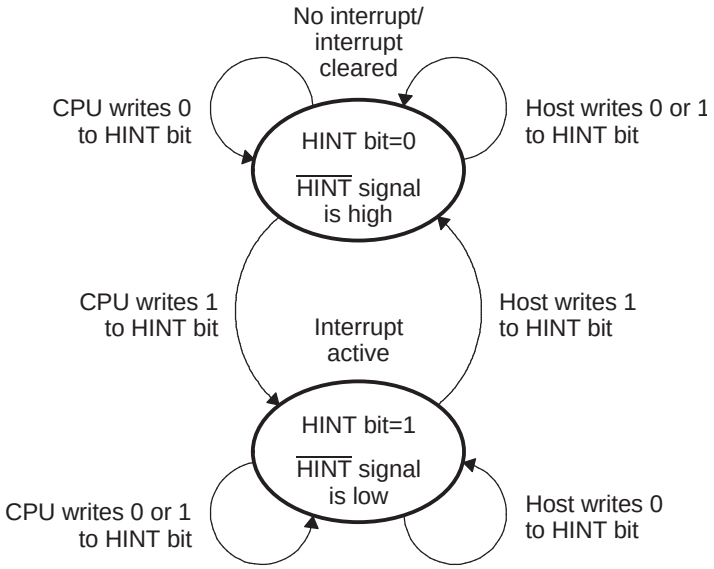
Before the host can use DSPINT to generate a subsequent interrupt to the CPU, the CPU must acknowledge the current interrupt by writing a 1 to the DSPINT bit. When the CPU writes 1, DSPINT is forced to 0. The host should verify that DSPINT = 0 before generating subsequent interrupts. While DSPINT = 1, host writes to the DSPINT bit do not generate an interrupt pulse.

Writes of 0 have no effect. A hardware reset immediately clears DSPINT and thus clears an active host-to-CPU interrupt.

16.2.8.2 HINT Bit: CPU-to-Host Interrupts

The HINT bit in UHPIC allows the CPU to send an interrupt request to the host. The use of the HINT bit is summarized in Figure 16-17.

Figure 16-17. CPU-to-Host Interrupt State Diagram



If the CPU writes 1 to the HINT bit of UHPIC, the UHPI drives the $\overline{\text{UHPI\_HINT}}$ signal low, indicating an interrupt condition to the host. Before the CPU can use the HINT bit generate a subsequent interrupt to host, the host must acknowledge the current interrupt by writing 1 to the HINT bit. When the host does this, the UHPI clears the HINT bit (HINT = 0), and this drives the $\overline{\text{UHPI\_HINT}}$ signal high. The CPU should read UHPIC and make sure HINT = 0 before generating subsequent interrupts.

Writes of 0 have no effect. A hardware reset immediately clears the HINT bit and thus clears an active CPU-to-host interrupt.

16.2.9 Emulation Considerations

The FREE and SOFT bits in the power and emulation management register (PWREMU_MGMT) determine the response of the UHPI to an emulation suspend condition. If FREE = 1, the UHPI is not affected, and the SOFT bit has no effect. If FREE = 0 and SOFT = 0, the UHPI is not affected. If FREE = 0 and SOFT = 1:

- The UHPI DMA logic halts after the current host and UHPI DMA operations are completed.
- The external host interface functions as normal throughout the emulation suspend condition. The host may access the control register (UHPIC). The host may also access the UHPIA registers and may perform data reads until the read FIFO is empty or data writes until the write FIFO is full. As in normal operation, UHPI_HRDY is driven low during a host cycle that cannot be completed due to the write FIFO being full or the read FIFO being empty. If this occurs, UHPI_HRDY continues to be driven low, holding off the host, until the emulation suspend condition is over, and the FIFOs are serviced by the UHPI DMA logic, allowing the host cycle to complete.
- When the emulation suspend condition is over, the appropriate requests by the UHPI DMA logic are made to process any posted host writes in the write FIFO or to fill the read FIFO as necessary. UHPI operation then continues as normal.

16.3 UHPIA Register Settings

The UHPIA setting consists of two 16-bit registers: UHPIAWL and UHPIAWU. The values in two 16-bit registers form a 32-bit address that defines a location inside the memory of the DSP that a UHPI-host will write to or read from. Since the transfer from the FIFO of UHPI to the internal memory of the DSP is by DMA, the settings must be calculated according to the following formula:

(SARAM_byte_address + 0x80000)/4

For example, to transfer the UHPI data to the SARAM0 that has the byte address as 0x10000, the setting can be calculated as follows:

(0x10000 + 0x80000)/4 = 0x24000

Based on the calculation result, the 32-bit address is 0x24000. As a result, “0x4000” and “0x0002” should be set to the UHPIAWL and UHPIAWU, respectively.

16.4 Chip-Level Configuration for the UHPI Mode

In UHPI mode, the pin UHPI_HBE[1:0] of the device must be driven as “0” (connected to ground). These two pins are the byte enable signal when the device is set to UHPI mode. Since the device supports only the 16-bit multiplex UHPI mode, so (UHPI_HBE[1:0]) must be set as “00”.

Some software configuration needs to be done on the chip in order to set the device into UHPI mode. The main steps are:

1. Set the pin mux mode
2. Enable the UHPI clock
3. Software reset the UHPI module to flush the FIFO of UHPI

The following code is an example for the UHPI initialization program:

```
EBSR &= 0x8FFF;                          // set pin mux to select the UHPI
PCGCR2 = 0x0000;                         // clock gating control

for (i=0; i<20; i++) {__asm("    NOP");}

CLKSTOPL &= 0xFBFF;                      //clear bit-10 to enable UHPI
PSRCR = 0x000A;                          //reset control counter

PRCR = 0x0008;                           //reset UHPI

for (i=0; i<20; i++) {__asm("    NOP");} // Delay
```

16.5 UHPI Configuration Register at Top Level

The UHPI configuration register is at 0x1C4E. The following tables list the definition of this register.

Figure 16-18. UHPI Configuration Register (0x1C4E)

15	3	2	1	0
Reserved		Byte Addr	Full Word	NMux
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-5. UHPI Configuration Register Bit Field Descriptions

Bit	Field	Value	Description
15-3	Reserved		
2	Byte Addr		UHPI byte/word address mode select
		0	Word addressing mode. Host address corresponds to a word (16-bit) address.
		1	Byte addressing mode. Host address corresponds to a byte (8-bit) address.
1	Full Word		UHPI dual- or half-word mode select
		0	Full word (16-bit) operation. This setting needs to be used in conjunction with 16-bit multiplexed mode in the Parallel Port Mode of the External Bus Selection Register.
		1	Double-word (32-bit) operation. (Not supported on this device.)
0	NMux		UHPI multiplexed or non-multiplexed mode select
		0	UHPI uses multiplexed mode.
		1	UHPI uses non-multiplexed mode. The EMIF address bus is used as UHPI address pins. This setting prevents the EMIF from using the address and data pins. (Not supported on this device.)

16.6 UHPI Registers

Table 16-6 lists the memory-mapped registers for the UHPI. All register offset addresses not listed in Table 16-6 should be considered as reserved locations and the register contents should not be modified.

Table 16-6. UHPI REGISTERS

CPU Word Address	Acronym	Register Name	Section
2E00h	PIDL	This Register stores version information used to identify the specific UHPI peripheral implemented in a particular device	Section 16.6.1
2E01h	PIDU	This Register stores version information used to identify the specific UHPI peripheral implemented in a particular device	Section 16.6.2
2E04h	PWREMU_MGMT	This Register control the operation of the peripheral during emulation suspend and / or idle states	Section 16.6.3
2E08h	GPINT_CTRLL	GPIO Interrupts Control Register	Section 16.6.4
2E09h	GPINT_CTRLU	GPIO Interrupts Control Register	Section 16.6.5
2E0Ch	GPIO_EN	GPIO Enable Register	Section 16.6.6
2E10h	GPIO_DIR1	GPIO Direction Control Register	Section 16.6.7
2E14h	GPIO_DAT1	GPIO Data Register	Section 16.6.8
2E18h	GPIO_DIR2	GPIO Direction Control Register	Section 16.6.9
2E1Ch	GPIO_DAT2	GPIO Data Register	Section 16.6.10
2E30h	UHPICL	Host Port Interface Control Register Lower	Section 16.6.11
2E34h	UHPIAWL	Register stores address UHPI write operation	Section 16.6.12
2E35h	UHPIAWU	Register stores address UHPI write operation	Section 16.6.13
2E38h	UHPIARL	Register stores address UHPI read operation	Section 16.6.14
2E39h	UHPIARU	Register stores address UHPI read operation	Section 16.6.15

Table 16-6. UHPI REGISTERS (continued)

CPU Word Address	Acronym	Register Name	Section
2E3Ch	XUHPIAWL	eXtended Host Port Interface Write Address Register Lower	Section 16.6.16
2E3Dh	XUHPIAWU	eXtended Host Port Interface Write Address Register Upper	Section 16.6.17
2E40h	XUHPIARL	eXtended Host Port Interface Read Address Register Lower	Section 16.6.18
2E41h	XUHPIARU	eXtended Host Port Interface Read Address Register Upper	Section 16.6.19

16.6.1 PIDL Register (word address = 2E00h) [reset = 10Ah]

PIDL is shown in Figure 16-19 and described in Table 16-7.

This Register stores versioning information used to identify the specific UHPI peripheral implemented in a particular device

Figure 16-19. PIDL Register

15	14	13	12	11	10	9	8
RTL					MAJOR		
R-0h					R-1h		

7	6	5	4	3	2	1	0
CUSTOM		MINOR					
R-0h		R-Ah					

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 16-7. PIDL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-11	RTL	R	0h	RTL Version.
10-8	MAJOR	R	1h	Major Revision. 001b for module releases corresponding to this functional spec.
7-6	CUSTOM	R	0h	Non-custom.
5-0	MINOR	R	Ah	Minor Revision. 001010b for module releases corresponding to this functional spec.

16.6.2 PIDU Register (word address = 2E01h) [reset = 4421h]

PIDU is shown in Figure 16-20 and described in Table 16-8.

This Register stores versioning information used to identify the specific UHPI peripheral implemented in a particular device

Figure 16-20. PIDU Register

15	14	13	12	11	10	9	8
TYPE		Reserved		FUNC			
R-1h		R-0h		R-421h			

7	6	5	4	3	2	1	0
FUNC							
R-421h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 16-8. PIDU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	TYPE	R	1h	Used to distinguish between old Scheme and current. Spare bit to encode future schemes.
13-12	Reserved	R	0h	Reserved
11-0	FUNC	R	421h	Function. 421h for UHPI.

16.6.3 PWREMU_MGMT Register (word address = 2E04h) [reset = 0h]

PWREMU_MGMT is shown in Figure 16-21 and described in Table 16-9.

This Register control the operation of the peripheral during emulation suspend and / or idle states

Figure 16-21. PWREMU_MGMT Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved						SOFT	FREE
R-0h						RW-0h	RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 16-9. PWREMU_MGMT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	SOFT	RW	0h	Emulation Suspend bit 0x0 = Upon emulation suspend the peripheral operation is not affected. 0x1 = Upon emulation suspend current host and Vbus operations will complete, after which the Vbus interfaces will remain in the idle state
0	FREE	RW	0h	Free run mode operation bit 0x0 = SOFT bit takes effect 0x1 = Upon emulation suspend the peripheral operation will not be affected (regardless of SOFT).

16.6.4 GPINT_CTRLL Register (word address = 2E08h) [reset = 0h]

GPINT_CTRLL is shown in Figure 16-22 and described in Table 16-10.

GPIO interrupts control Register

Figure 16-22. GPINT_CTRLL Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved						GPINT_EN2	Reserved
R-0h						RW-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 16-10. GPINT_CTRLL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	GPINT_EN	RW	0h	GPIO Interrupt Enable bit 0x0 = Enabled /HAS as a GPINT sourcing the CPU Interrupt 0x1 = Disabled /HAS as a GPINT sourcing the CPU Interrupt
0	Reserved	R	0h	Reserved

16.6.5 GPINT_CTRLU Register (word address = 2E09h) [reset = 0h]

GPINT_CTRLU is shown in Figure 16-23 and described in Table 16-11.

GPIO interrupts control Register

Figure 16-23. GPINT_CTRLU Register

15	14	13	12	11	10	9	8
Reserved							
R-0h							

7	6	5	4	3	2	1	0
Reserved						GPINT_INV2	Reserved
R-0h						RW-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 16-11. GPINT_CTRLU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	GPINT_INV	RW	0h	GPINT Invert bit 0x0 = GPINT not inverted 0x1 = GPINT inverted
0	Reserved	R	0h	Reserved

16.6.6 GPIO_EN Register (word address = 2E0Ch) [reset = 0h]

GPIO_EN is shown in Figure 16-24 and described in Table 16-12.

GPIO Enable Register

Figure 16-24. GPIO_EN Register

15	14	13	12	11	10	9	8
Reserved							GPIO_EN8
R-0h							RW-0h
7	6	5	4	3	2	1	0
GPIO_EN7	GPIO_EN6	GPIO_EN5	GPIO_EN4	GPIO_EN3	GPIO_EN2	GPIO_EN1	GPIO_EN0
RW-0h	RW-0h	RW-0h	RW-0h	RW-0h	RW-0h	RW-0h	RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 16-12. GPIO_EN Register Field Descriptions

Bit	Field	Type	Reset	Description
15-9	Reserved	R	0h	Reserved
8	GPIO_EN8	RW	0h	GPIO Enable bit 8 0x0 = HD[15:8] pins will work as UHPI. 0x1 = HD[15:8] pins will work as GPIO.
7	GPIO_EN7	RW	0h	GPIO Enable bit 7 0x0 = HD[7:0] pins will work as UHPI. 0x1 = HD[7:0] pins will work as GPIO.
6	GPIO_EN6	RW	0h	GPIO Enable bit 6 0x0 = /HINT pin will work as UHPI. 0x1 = /HINT pin will work as GPIO.
5	GPIO_EN5	RW	0h	GPIO Enable bit 5 0x0 = HRDY and /HRDY pins will work as UHPI. 0x1 = HRDY and /HRDY pins will work as GPIO.
4	GPIO_EN4	RW	0h	GPIO Enable bit 4 0x0 = HHWIL pin will work as UHPI. 0x1 = HHWIL pin will work as GPIO.
3	GPIO_EN3	RW	0h	GPIO Enable bit 3 0x0 = /HBE[3:0] pins will work as UHPI. 0x1 = /HBE[3:0] pins will work as GPIO.
2	GPIO_EN2	RW	0h	GPIO Enable bit 2 0x0 = /HAS pin will work as UHPI. 0x1 = /HAS pin will work as GPIO.
1	GPIO_EN1	RW	0h	GPIO Enable bit 1 0x0 = HCNTL[B:A] pins will work as UHPI. 0x1 = HCNTL[B:A] pins will work as GPIO.
0	GPIO_EN0	RW	0h	GPIO Enable bit 0 0x0 = /HCS, /HDS1, /HDS2 and HR/W pins will work as UHPI. 0x1 = /HCS, /HDS1, /HDS2 and HR/W pins will work as GPIO.

16.6.7 GPIO_DIR1 Register (word address = 2E10h) [reset = 0h]

GPIO_DIR1 is shown in Figure 16-25 and described in Table 16-13.

GPIO Direction Control Register

Figure 16-25. GPIO_DIR1 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 16-13. GPIO_DIR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DIR	RW	0h	The DIR bits control the direction of the HD [15:0] pins when they are configured as general purpose I/O. DIRx=1 configures the HDx pin as an output and DIRx=0 configures the HDx pin as an input.

16.6.8 GPIO_DAT1 Register (word address = 2E14h) [reset = 0h]

GPIO_DAT1 is shown in Figure 16-26 and described in Table 16-14.

GPIO Data Register

Figure 16-26. GPIO_DAT1 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 16-14. GPIO_DAT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DAT	RW	0h	The value written in this will reflect on HD [15:0], when configured as output and will read the logical level on the pin when configured as input.

16.6.9 GPIO_DIR2 Register (word address = 2E18h) [reset = 0h]

GPIO_DIR2 is shown in Figure 16-27 and described in Table 16-15.

GPIO Direction Control Register

Figure 16-27. GPIO_DIR2 Register

15	14	13	12	11	10	9	8
Reserved			DIR12	DIR11	Reserved	DIR9	DIR8
R-0h			RW-0h	RW-0h	R-0h	RW-0h	RW-0h
7	6	5	4	3	2	1	0
DIR7	DIR6	DIR5	DIR4	DIR3	DIR2	DIR1	DIR0
RW-0h	RW-0h	RW-0h	RW-0h	RW-0h	RW-0h	RW-0h	RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 16-15. GPIO_DIR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved
12	DIR12	RW	0h	Direction control for /HBE1 pin when configured as GPIO 0x0 = /HBE1 pin configured as input. 0x1 = /HBE1 pin configured as output.
11	DIR11	RW	0h	Direction control for /HBE0 pin when configured as GPIO 0x0 = /HBE0 pin configured as input. 0x1 = /HBE0 pin configured as output.
10	Reserved	R	0h	Reserved
9	DIR9	RW	0h	Direction control for HRDY pin when configured as GPIO 0x0 = HRDY pin configured as input. 0x1 = HRDY pin configured as output.
8	DIR8	RW	0h	Direction control for /HINT pin when configured as GPIO 0x0 = /HINT pin configured as input. 0x1 = /HINT pin configured as output.
7	DIR7	RW	0h	Direction control for HCNTLA pin when configured as GPIO 0x0 = HCNTLA pin configured as input. 0x1 = HCNTLA pin configured as output.
6	DIR6	RW	0h	Direction control for HCNTLB pin when configured as GPIO 0x0 = HCNTLB pin configured as input. 0x1 = HCNTLB pin configured as output.
5	DIR5	RW	0h	Direction control for HHWIL pin when configured as GPIO 0x0 = HHWIL pin configured as input. 0x1 = HHWIL pin configured as output.
4	DIR4	RW	0h	Direction control for HR/W pin when configured as GPIO 0x0 = HR/W pin configured as input. 0x1 = HR/W pin configured as output.
3	DIR3	RW	0h	Direction control for /HDS2 pin when configured as GPIO 0x0 = /HDS2 pin configured as input. 0x1 = /HDS2 pin configured as output.
2	DIR2	RW	0h	Direction control for /HDS1 pin when configured as GPIO 0x0 = /HDS1 pin configured as input. 0x1 = /HDS1 pin configured as output.
1	DIR1	RW	0h	Direction control for /HCS pin when configured as GPIO 0x0 = /HCS pin configured as input. 0x1 = /HCS pin configured as output.

Table 16-15. GPIO_DIR2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	DIR0	RW	0h	Direction control for /HAS pin when configured as GPIO 0x0 = /HAS pin configured as input. 0x1 = /HAS pin configured as output.

16.6.10 GPIO_DAT2 Register (word address = 2E1Ch) [reset = 0h]

GPIO_DAT2 is shown in Figure 16-28 and described in Table 16-16.

GPIO Data Register

Figure 16-28. GPIO_DAT2 Register

15	14	13	12	11	10	9	8
Reserved			DAT12	DAT11	Reserved	DAT9	DAT8
R-0h			RW-0h	RW-0h	R-0h	RW-0h	RW-0h
7	6	5	4	3	2	1	0
DAT7	DAT6	DAT5	DAT4	DAT3	DAT2	DAT1	DAT0
RW-0h	RW-0h	RW-0h	RW-0h	RW-0h	RW-0h	RW-0h	RW-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 16-16. GPIO_DAT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved
12	DAT12	RW	0h	Controls the output level of the /HBE1 pin when configured as output and reads the status of /HBE1 when configured as input 0x0 = /HBE1 pin is at logical low or driven low 0x1 = /HBE1 pin is at logical high or driven high
11	DAT11	RW	0h	Controls the output level of the /HBE0 pin when configured as output and reads the status of /HBE0 when configured as input 0x0 = /HBE0 pin is at logical low or driven low 0x1 = /HBE0 pin is at logical high or driven high
10	Reserved	R	0h	Reserved
9	DAT9	RW	0h	Controls the output level of the HRDY pin when configured as output and reads the status of HRDY when configured as input 0x0 = HRDY pin is at logical low or driven low 0x1 = HRDY pin is at logical high or driven high
8	DAT8	RW	0h	Controls the output level of the /HINT pin when configured as output and reads the status of /HINT when configured as input 0x0 = /HINT pin is at logical low or driven low 0x1 = /HINTpin is at logical high or driven high
7	DAT7	RW	0h	Controls the output level of the HCNTLA pin when configured as output and reads the status of HCNTLA when configured as input 0x0 = HCNTLA pin is at logical low or driven low 0x1 = HCNTLA pin is at logical high or driven high
6	DAT6	RW	0h	Controls the output level of the HCNTLB pin when configured as output and reads the status of HCNTLB when configured as input 0x0 = HCNTLB pin is at logical low or driven low 0x1 = HCNTLB pin is at logical high or driven high
5	DAT5	RW	0h	Controls the output level of the HHWIL pin when configured as output and reads the status of HHWIL when configured as input 0x0 = HHWIL pin is at logical low or driven low 0x1 = HHWIL pin is at logical high or driven high
4	DAT4	RW	0h	Controls the output level of the HR/W pin when configured as output and reads the status of HR/W when configured as input 0x0 = HR/W pin is at logical low or driven low 0x1 = HR/W pin is at logical high or driven high
3	DAT3	RW	0h	Controls the output level of the /HDS2 pin when configured as output and reads the status of /HDS2 when configured as input 0x0 = /HDS2 pin is at logical low or driven low 0x1 = /HDS2 pin is at logical high or driven high

Table 16-16. GPIO_DAT2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	DAT2	RW	0h	Controls the output level of the /HDS1 pin when configured as output and reads the status of /HDS1 when configured as input 0x0 = /HDS1 pin is at logical low or driven low 0x1 = /HDS1 pin is at logical high or driven high
1	DAT1	RW	0h	Controls the output level of the /HCS pin when configured as output and reads the status of /HCS when configured as input 0x0 = /HCS pin is at logical low or driven low 0x1 = /HCS pin is at logical high or driven high
0	DAT0	RW	0h	Controls the output level of the /HAS pin when configured as output and reads the status of /HAS when configured as input 0x0 = /HAS pin is at logical low or driven low 0x1 = /HAS pin is at logical high or driven high

16.6.11 UHPICL Register (word address = 2E30h) [reset = C0h]

UHPICL is shown in Figure 16-29 and described in Table 16-17.

Host Port Interface Control Register-Lower

Figure 16-29. UHPICL Register

15	14	13	12	11	10	9	8
NRDY_UHPID_WR	NRDY_UHPIDP_WR	NRDY_UHPID_RD	NRDY_UHPIA_WR	UHPIA_RW_SEL	LB_MODE	DUAL_UHPIA	HWOB_STAT
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
UHPI_RST	RESET	XUHPIA	FETCH	HRDY	HINT	DSP_INT	HWOB
RW-1h	R-1h	R-0h	R-0h	R-0h	RW-0h	RW-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 16-17. UHPICL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	NRDY_UHPID_WR	R	0h	UHPID Non-Incrementing Write Not Ready 0x0 = It is safe to perform a single UHPID non-incrementing write because the write FIFO is empty. 0x1 = The write FIFO is not empty or is flushing, therefore an UHPID non-incrementing write would result in HRDY pin deassertion. The write data is lost or corrupted and the UHPI interface could be corrupted.
14	NRDY_UHPIDP_WR	R	0h	HPID+ Incrementing Write Not Ready 0x0 = It is safe to perform a single UHPID+ incrementing write because the write FIFO is not full. 0x1 = The write FIFO is full. Therefore UHPID+ incrementing write would result in HRDY pin deassertion. The write data is lost or corrupted and subsequent writes may be out of sync or the UHPI interface could be corrupted.
13	NRDY_UHPID_RD	R	0h	UHPID Read Not Ready 0x0 = It is safe to perform a single UHPID read (incrementing or non-incrementing) because the read FIFO is not empty. 0x1 = The read FIFO is empty or flushing. Therefore any UHPID read would result in HRDY pin deassertion. The read data is not guaranteed and subsequent reads may also be out of sync or the UHPI interface could be corrupted.
12	NRDY_UHPIA_WR	R	0h	UHPIA Write Not Ready 0x0 = It is safe to perform a single UHPIA write because any previous UHPIA write has already been fully synchronized. 0x1 = A previous UHPIA write has not finished synchronizing. Therefore any UHPIA write could result in HRDY pin deassertion and the UHPIA value and the host interface could be corrupted.
11	UHPIA_RW_SEL	R	0h	UHPIA Register select bit, When DUAL_UHPIA=1, this bit is used to select the UHPIA Register to be accessed 0x0 = UHPIA-W (write address) Register is selected 0x1 = UHPIA-R (read address) Register selected.
10	LB_MODE	R	0h	Loop-back mode configuration bit, valid only when HOSTLESS=1 0x0 = Loopback is enabled 0x1 = Loopback is disabled

Table 16-17. UHPICL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	DUAL_UHPIA	R	0h	Dual UHPIA mode configuration bit. The CPU can access both UHPIA registers separately regardless of DUAL_UHPIA setting. Regardless of this bit, dual UHPIA mode is implied when the device has ownership of a (X)UHPIA register. 0x0 = Dual UHPIA mode is disabled. If this bit is 0, the two UHPIA registers operate as a single virtual UHPIA register in terms of host accesses. 0x1 = Dual UHPIA mode enabled.
8	HWOB_STAT	R	0h	HWOB status. The value of the HWOB bit is also stored in this bit position. A write to the UHPIC bit position 0 will also update this bit. 0x0 = The HWOB bit is zero 0x1 = The HWOB bit is one
7	UHPI_RST	RW	1h	UHPI_RESET bit 0x0 = CLEAR 0x1 = The FIFO pointers are held in reset as soon as the pending vbus transactions are complete. Even after writing 1 to UHPI_RST, the CPU will read back a 0 from this bit position until the FIFOs are reset. Only the CPU may write to this bit.
6	RESET	R	1h	CPU core reset bit 0x0 = Reset signal to the CPU core is not active 0x1 = Reset signal to the CPU core is active
5	XUHPIA	R	0h	Extended address enable. 0x0 = Extended address disabled. 0x1 = Extended address enabled.
4	FETCH	R	0h	Host data fetch request bit. 0x0 = No prefetch. 0x1 = Request Vbus master to prefetch data into the read FIFO.
3	HRDY	R	0h	The logic level of the internal HRDY signal appears in this field. The host can read this bit for software polling of HRDY. 0x0 = UHPI has not completed the current data access 0x1 = UHPI has completed the current data access
2	HINT	RW	0h	DSP to host interrupt. The host must write a 1 to HINT to clear the HINT pin. Writing a 0 to the HINT bit by the host or DSP has no effect. 0x0 = DSP to host interrupt bit has no effect. 0x1 = DSP to host interrupt has been generated.
1	DSP_INT	RW	0h	Host to DSP interrupt 0x0 = Host to DSP interrupt is not generated 0x1 = Host to DSP interrupt is generated
0	HWOB	R	0h	Half-word Ordering Bit. HWOB affects both data and address transfers. HWOB must be initialized before the first data or address register access. 0x0 = First half-word is most significant. 0x1 = First half-word is least significant.

16.6.12 UHPIAWL Register (word address = 2E34h) [reset = 0h]

UHPIAWL is shown in Figure 16-30 and described in Table 16-18.

Register stores address UHPI write operation

Figure 16-30. UHPIAWL Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 16-18. UHPIAWL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	UHPIAW	R	0h	Memory Address used for host writes when in muxed mode.

16.6.13 UHPIAWU Register (word address = 2E35h) [reset = 0h]

UHPIAWU is shown in Figure 16-31 and described in Table 16-19.

Register stores address UHPI write operation

Figure 16-31. UHPIAWU Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 16-19. UHPIAWU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	UHPIAW	R	0h	Memory Address used for host writes when in muxed mode.

16.6.14 UHPIARL Register (word address = 2E38h) [reset = 0h]

UHPIARL is shown in Figure 16-32 and described in Table 16-20.

Register stores address UHPI read operation

Figure 16-32. UHPIARL Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 16-20. UHPIARL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	UHPIAR	R	0h	Memory Address used for host reads when in muxed mode.

16.6.15 UHPIARU Register (word address = 2E39h) [reset = 0h]

UHPIARU is shown in Figure 16-33 and described in Table 16-21.

Register stores address UHPI read operation

Figure 16-33. UHPIARU Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 16-21. UHPIARU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	UHPIAR	R	0h	Memory Address used for host reads when in muxed mode.

16.6.16 XUHPIAWL Register (word address = 2E3Ch) [reset = 0h]

XUHPIAWL is shown in Figure 16-34 and described in Table 16-22.

Register stores extended address UHPI write operation

Figure 16-34. XUHPIAWL Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 16-22. XUHPIAWL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	XUHPIAW	R	0h	Extended memory address used for host writes when in muxed mode. Value range is 0000 to FFFFh.

16.6.17 XUHPIAWU Register (word address = 2E3Dh) [reset = 0h]

XUHPIAWU is shown in Figure 16-35 and described in Table 16-23.

Register stores extended address UHPI write operation

Figure 16-35. XUHPIAWU Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 16-23. XUHPIAWU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	XUHPIAW	R	0h	Extended memory address used for host writes when in muxed mode. Value range is 0000 to FFFFh.

16.6.18 XUHPIARL Register (word address = 2E40h) [reset = 0h]

XUHPIARL is shown in Figure 16-36 and described in Table 16-24.

Register stores extended address UHPI read operation

Figure 16-36. XUHPIARL Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 16-24. XUHPIARL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	XUHPIAR	R	0h	Extended memory address used for host reads when in muxed mode. Value range is 0000 to FFFFh.

16.6.19 XUHPIARU Register (word address = 2E41h) [reset = 0h]

XUHPIARU is shown in Figure 16-37 and described in Table 16-25.

Register stores extended address UHPI read operation

Figure 16-37. XUHPIARU Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 16-25. XUHPIARU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	XUHPIAR	R	0h	Extended memory address used for host reads when in muxed mode. Value range is 0000 to FFFFh.

Universal Serial Bus (USB) Controller

Topic **Page**

17.1 Introduction

This document describes the universal serial bus (USB) controller. The controller complies with the USB 2.0 standard high-speed and full-speed functions. In addition, the four test modes for high-speed operation described in the USB 2.0 specification are supported. It also allows options that allow the USB controller to be forced into full-speed mode and high-speed mode that may be used for debug purposes.

17.1.1 Purpose of the Peripheral

The USB controller provides a low-cost connectivity solution for consumer portable devices by providing a mechanism for data transfer between USB devices up to 480 Mbps. With the USB controller, you can use the DSP to create a high-speed USB slave device that is complaint with the Universal Serial Bus Specification version 2.0.

17.1.2 Features

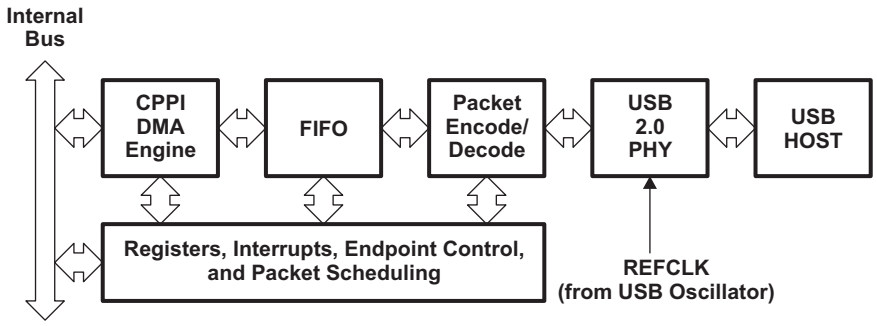
The USB has the following features:

- Operating as a peripheral, it complies with the USB 2.0 standard for high-speed (480 Mbps) and full-speed (12 Mbps) operation with a host
- Supports 4 simultaneous RX and TX endpoints, in addition to control endpoint, more devices can be supported by dynamically switching endpoints states
- Each endpoint (other than endpoint 0) can support all transfer types (control, bulk, interrupt, and isochronous)
- Includes a 4K endpoint FIFO RAM, and supports programmable FIFO sizes
- Includes a DMA controller that supports 4 TX and 4 RX DMA channels
- Includes four types of Communications Port Programming Interface (CPPI) 4.1 DMA compliant transfer modes, Transparent, Generic RNDIS, RNDIS, and Linux CDC mode of DMA for accelerating RNDIS type protocols using short packet termination over USB
- DMA supports single data transfer size up to 4Mbytes

17.1.3 Functional Block Diagram

The USB functional block diagram is shown in Figure 17-1.

Figure 17-1. Functional Block Diagram



17.1.4 Industry Standard(s) Compliance Statement

This device conforms to USB 2.0 Specification.

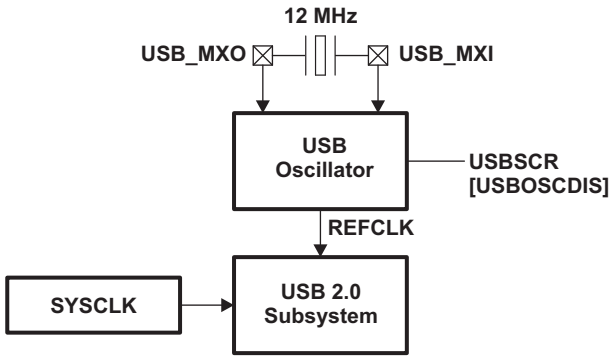
17.2 Architecture

17.2.1 Clock Control

Figure 17-2 shows the clock connections for the USB2.0 module. Note that there is a built-in oscillator that generates a 12 MHz reference clock for the internal PLL of the USB 2.0 subsystem. The USB2.0 subsystem peripheral bus clock is sourced from the system clock (SYSCLK).

NOTE: The device system clock (SYSCLK) must be at least 30 MHz for proper USB operation.

Figure 17-2. USB Clocking Diagram



17.2.2 Signal Descriptions

The USB controller provides the I/O signals listed in Table 17-1.

Table 17-1. USB Terminal Functions

Name	I/O[1]	Description
USB_DP	A I/O/Z	USB D+ (differential signal pair)
USB_DM	A I/O/Z	USB D- (differential signal pair)
USB_VBUS	A I	Five volt input that signifies that VBUS is connected.
USB_REXT	A I/O/Z	External resistor connect.
USB_MXI	I	12 MHz crystal oscillator input.
USB_MXO	O	12 MHz crystal oscillator output.
USB_LDOO	PWR	USB module LDO output. This output is regulated to 1.3V.
USB_LDOI	PWR	USB module LDO input. This input handles a voltage range of 1.8V to 3.6V.
VSS_USBOSC	PWR	3.3V USB oscillator power supply.
VDD_USBPLL	PWR	3.3V USB PLL power supply.
VDDA_USBXCVR	PWR	3.3V USB transceiver power supply.
VDDA_USB	PWR	1.3V USB analog power supply.
VDD_USB	PWR	1.3V USB PLL and oscillator digital power supply.
VSS_USBOSC	GND	USB oscillator ground.
VSS_USBPLL	GND	USB PLL ground.
VSSA_USBXCVR	GND	USB transceiver ground.
VSS_USBXCVR	GND	USB ground for reference circuits.
VSSA_USB	GND	USB analog ground.
VSS_USB	GND	USB PLL and oscillator digital ground.

(1) I = Input, O = Output, Z = High impedance, GND = Ground, A = Analog signal, PWR = Power supply pin.

17.2.3 Memory Map

The USB controller can access internal single-access RAM (SARAM) and external memory. It cannot access dual-access RAM (DARAM). The starting address for SARAM and external memory is different from the point-of-view of the CPU and USB controller. The memory map, as seen by the USB controller and the CPU, is shown in Table 17-2.

Table 17-2. USB Controller Memory Map

USB Start *Byte* Address	CPU Start *Word* Address	CPU Memory Map	USB Controller Memory Map
0001 0000h[1]	00 0000h[1]	DARAM	Reserved
0009 0000h	00 8000h	SARAM	SARAM
0100 0000h	02 8000h	EMIF CS0	EMIF CS0
0200 0000h	40 0000h	EMIF CS2	EMIF CS2
0300 0000h	60 0000h	EMIF CS3	EMIF CS3
0400 0000h	70 0000h	EMIF CS4	EMIF CS4
0500 0000h	78 0000h	EMIF CS5	EMIF CS5

(1) CPU word addresses 00 0000h - 00 005Fh (which correspond to byte addresses 00 0000h - 00 00BFh) are reserved for the memory-mapped registers (MMRs) of the DSP CPU.

17.2.4 USB_DP/USB_DM Polarity Inversion

The polarity of the USB data pins (USB_DP and USB_DM) can be inverted through the USBDATPOL bit of the USB system control register (USBSCR). Since USB_DP is equal to the inverse of USB_DM (they form a differential pair), inverting these pins allows you to effectively swap their function. This allows flexibility in board design by allowing different USB connector configurations. In particular, this allows for mounting the connector on either side of the board and for arranging the data pins so they do not physically cross each other.

17.2.5 Indexed and Non-Indexed Registers

The USB controller provides two mechanisms of accessing the endpoint control and status registers:

- Indexed Endpoint Control/Status Registers: These registers are located at I/O address 8410h to 841Fh. The endpoint is selected by programming the INDEX register of the controller.
- Non-indexed Endpoint Control/Status Registers: These registers are located at I/O address 8500h to 854Fh. Registers at address 8500h to 850Fh map to Endpoint 0; at address 8510h to 851Fh map to Endpoint 1, and so on.

For detailed information about the USB controller registers, see Section 17.3.

17.2.6 USB PHY Initialization

The general procedure for USB PHY initialization consists of enabling the USB on-chip oscillator, configuring PHY parameters, and finally resetting the PHY. The detailed USB PHY initialization sequence is as follows:

1. The bits USBOSCBIASDIS and USBOSCDIS in the USB system control register (USBSCR) should be cleared to 0 to enable the on-chip USB oscillatory if not enabled already.
2. Wait cycles for the on-chip oscillator to stabilize. Refer to the device-specific data manual for oscillator stabilization time.
3. To configure the PHY for normal operation, the bits USBPWDN, USBSESSEND, and USBPLLEN in USBSCR should be cleared to 0, the USBVBUSDET bit should be set to 1, and the USBDATPOL bit should be set according to the system requirements (set to 1 for normal operation).
4. Enable the USB clock by clearing USBCG to 0 in the peripheral clock gating configuration register 2 (PCGCR2) register.
5. Set the USBCLKSTPREQ bit.
6. Set COUNT = 20h in the peripheral software reset counter register (PSRCR).
7. Reset the USB controller by setting USB_RST to 1 in the peripheral reset control register (PRCR). This bit will self-clear once the reset has been completed.

For more information on the PCGCR2, CLKSTOP, PSRCR, and PRCR, see Section 1.1, *System Control* .

During the normal operation, the USB PHY PLL should run at 60 MHz. This clock can be probed on the device. One can monitor different clocks of the device by probing the CLKOUT pin for debugging purposes. Different clocks can be routed to this pin by setting both register CLKOUTCR (1C24h at IO space) and CPU register ST3_55.

To monitor the USB PHY PLL, the setting of register CLKOUTCR should be as follows:

- When CLKOUTCR = 0x000F; USB PHY clock (60 MHz) is routed to the CLKOUT pin

The setting of bit-2 (CLKOFF) of register ST3_55:

- When CLKOFF = 0, the CLKOUT pin is enabled
- When CLKOFF = 1, the CLKOUT pin is disabled

After these settings, the 60 MHz clock can be probed on CLKOUT pin.

17.2.6.1 USB System Control Register (USBSCR)

The USB system control register is used to disable the USB on-chip oscillator and to power-down the USB.

The USB system control register (USBSCR) is shown in Figure 17-3 and described in Table 17-3.

Figure 17-3. USB System Control Register (USBSCR) [1C32h]

15	14	13	12	11	8
USBPWDN	USBSESSEND	USBVBUSDET	USBPLLEN	Reserved	
R/W-1	R/W-0	R/W-1	R/W-0	R-0	

7	6	5	4	3	2	1	0
Reserved	USBDATPOL	Reserved		USBOSCBIASDIS	USBOSCDIS	BYTEMODE	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-3. USB System Control Register (USBSCR) Field Descriptions

Bit	Field	Value	Description
15	USBPWDN		USB module power.
		0	USB module is powered.
		1	USB module is powered-down.
14	USBSESSEND		USB VBUS session end comparator enable.
		0	USB VBUS session end comparator is disabled.
		1	USB VBUS session end comparator is enabled.
13	USBVBUSDET		USB VBUS detect enable.
		0	USB VBUS detect comparator is disabled.
		1	USB VBUS detect comparator is enabled.
12	USBPLLEN		USB PLL enable.
		0	Normal USB operation.
		1	Override USB suspend end behavior and force release of PLL from suspend state.
11-7	Reserved	0	Reserved. Always write 0 to these bits.
6	USBDATPOL		USB data polarity bit.
		0	Reverse polarity on DP and DM signals.
		1	Normal polarity.
5-4	Reserved	0	Reserved.
3	USBOSCBIASDIS		USB internal oscillator bias resistor disable.
		0	Internal oscillator bias resistor enabled (normal operating mode).
		1	Internal oscillator bias resistor disabled.
2	USBOSCDIS		USB oscillator disable bit.
		0	USB internal oscillator enable.
		1	USB internal oscillator disabled.
1-0	BYTEMODE		USB byte mode select bits.
		0	Word accesses by the CPU are allowed.
		1h	Byte accesses by the CPU are allowed (high byte is selected).
		2h	Byte accesses by the CPU are allowed (low byte is selected).
		3h	Reserved.

17.2.7 Dynamic FIFO Sizing

The USB controller supports a total of 4K RAM to dynamically allocate FIFO to all endpoints. The allocation of FIFO space to the different endpoints requires the specification for each Tx and Rx endpoint of:

- The start address of the FIFO within the RAM block
- The maximum size of packet to be supported
- Whether double-buffering is required.

These details are specified through four registers, which are added to the indexed area of the memory map. That is, the registers for the desired endpoint are accessed after programming the INDEX register with the desired endpoint value. Section 17.3.46, Section 17.3.47, Section 17.3.48, and Section 17.3.50 provide details of these registers.

NOTE: The option of setting FIFO sizes dynamically only applies to Endpoints 1 to 4. Endpoint 0 FIFO has a fixed size (64 bytes) and a fixed location (start address 0).

It is the responsibility of the firmware to ensure that all the Tx and Rx endpoints that are active in the current USB configuration have a block of RAM assigned exclusively to that endpoint that is at least as large as the maximum packet size set for that endpoint.

17.2.8 USB Controller Peripheral Mode Operation

The USB controller can be used as a high-speed or a full-speed USB peripheral device attached to a conventional USB host (such as a PC).

The USB2.0 controller will transition to session when it sees power (must be greater or equal to 4.01V) on the USB0_VBUS pin, assuming that the firmware has set the SOFTCONN bit in the POWER register and has enabled the data lines and there is an external host sourcing power on the USB0_VBUS line. The USB 2.0 controller will then set the SESSION bit upon detecting the power on the USB0_VBUS line and it will connect its 1.5Kohm pull-up resistor so it signifies to the external host out it is a Full-Speed device. Note that even when operating as a High-Speed; it has to first come up as Full-Speed. The USB2.0 controller will then wait for a reset signal from the host.

17.2.8.1 USB Interrupts

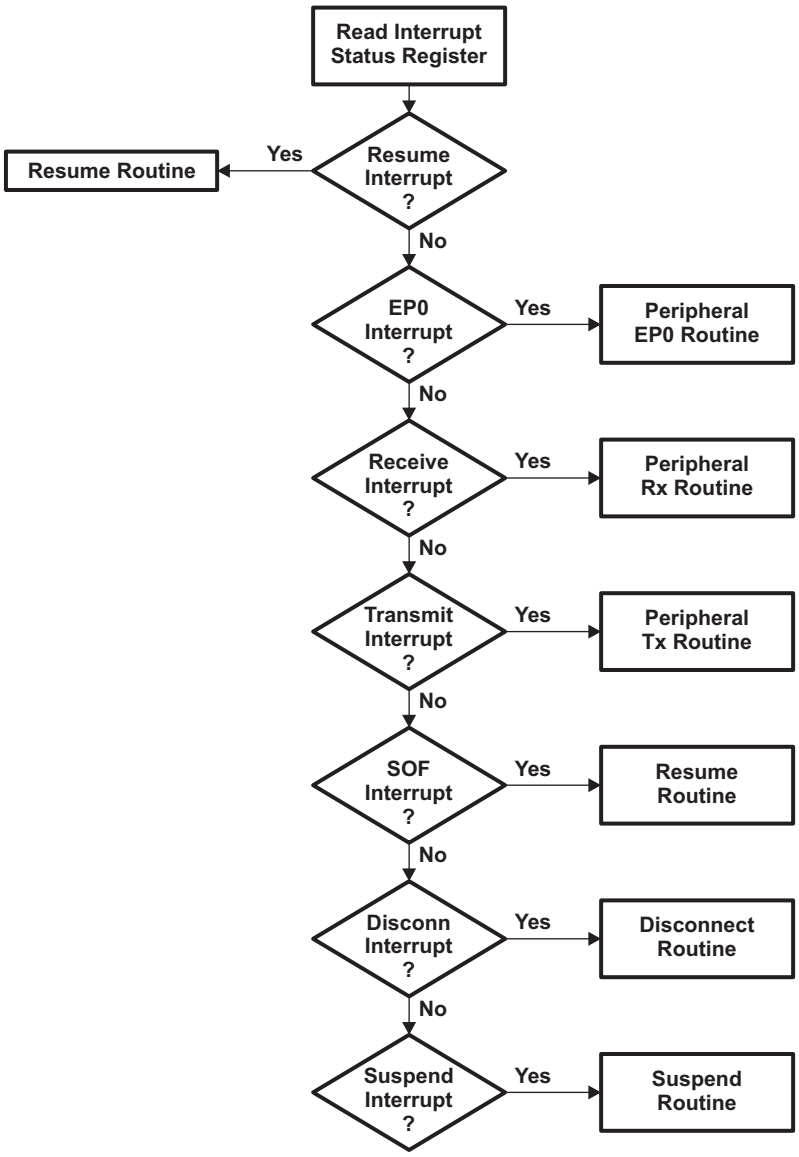
The USB controller interrupts the CPU on completion of the data transfer on any of the endpoints or on detecting reset, resume, suspend, connect, disconnect, or start-of-frame (SOF) on the bus.

When the CPU is interrupted with a USB interrupt, it needs to read the interrupt status register to determine the endpoints that have caused the interrupt and jump to the appropriate routine. If multiple endpoints have caused the interrupt, endpoint 0 should be serviced first followed by the other endpoints. The suspend interrupt should be serviced last.

The flowchart in Figure 17-4 describes the interrupt service routine for the USB module.

The following sections describe the programming of USB controller in peripheral mode.

Figure 17-4. Interrupt Service Routine Flow Chart



17.2.8.2 Connect, Suspend Mode, and Reset Signaling

The following sections describe the operation of the USB controller during connect, suspend mode, and USB reset.

17.2.8.2.1 Soft Connect

After a reset, the SOFTCONN bit in the POWER register is cleared to 0. The controller will therefore appear disconnected until the software has set the SOFTCONN bit to 1. The application software can then choose when to set the PHY into its normal mode. Systems with a lengthy initialization procedure may use this to ensure that initialization is complete and the system is ready to perform enumeration before connecting to the USB.

Once the SOFTCONN bit of the POWER register has been set, the software can also simulate a disconnect by clearing this bit to 0.

17.2.8.2.2 Suspend Mode

The controller monitors activity on the bus and when no activity has occurred for 3 ms, it goes into Suspend mode. If the Suspend interrupt has been enabled, an interrupt will be generated.

At this point, the controller can be left active (and hence able to detect when Resume signaling occurs on the USB), or the application may arrange to disable the controller by stopping its clock. However, the controller will not then be able to detect Resume signaling on the USB. As a result, some external hardware will be needed to detect Resume signaling (by monitoring the DM and DP signals) so that the clock to the controller can be restarted.

When Resume signaling occurs on the bus, first the clock to the controller must be restarted if necessary. Then the controller will automatically exit Suspend mode. If the Resume interrupt is enabled, an interrupt will be generated.

If the software wants to initiate a remote wake-up while the controller is in Suspend mode, it should write to the POWER register to set the RESUME bit to 1. The software should then leave this bit set for approximately 10 ms (minimum of 2 ms, a maximum of 15 ms) before resetting it to 0.

NOTE: No resume interrupt will be generated when the software initiates a remote wake-up.

17.2.8.2.3 Reset Signaling

If the HSENA bit in the POWER register was set, the controller also tries to negotiate for high-speed operation.

Whether high-speed operation is selected is indicated by the HSMODE bit of the POWER register.

When the application software receives a reset interrupt, it should close any open pipes and wait for bus enumeration to begin.

17.2.8.3 Control Transactions

Endpoint 0 is the main control endpoint of the core. The software is required to handle all the standard device requests that may be sent or received via endpoint 0. These are described in Universal Serial Bus Specification, Revision 2.0, Chapter 9. The protocol for these device requests involves different numbers and types of transactions per transfer. To accommodate this, the software needs to take a state machine approach to command decoding and handling.

The Standard Device Requests received by a USB peripheral device can be divided into three categories: Zero Data Requests (in which all the information is included in the command), Write Requests (in which the command will be followed by additional data), and Read Requests (in which the device is required to send data back to the host).

This section looks at the sequence of actions that the software must perform to process these different types of device request.

NOTE: The Setup packet associated with any standard device request should include an 8-byte command. Any setup packet containing a command field of anything other than 8 bytes will be automatically rejected by the controller.

17.2.8.3.1 Zero Data Requests

Zero data requests have all their information included in the 8-byte command and require no additional data to be transferred. Examples of Zero Data standard device requests are:

- SET_FEATURE
- CLEAR_FEATURE
- SET_ADDRESS
- SET_CONFIGURATION
- SET_INTERFACE

The sequence of events will begin, as with all requests, when the software receives an endpoint 0 interrupt. The RXPKTRDY bit of the PERI_CSR0 register will also have been set. The 8-byte command should then be read from the endpoint 0 FIFO, decoded, and the appropriate action taken.

For example, if the command is SET_ADDRESS, the 7-bit address value contained in the command should be written to the FADDR register. The PERI_CSR0 register should then be written to set the SERV_RXPKTRDY bit (indicating that the command has been read from the FIFO) and to set the DATAEND bit (indicating that no further data is expected for this request). The interval between setting the SERV_RXPKTRDY bit and setting the DATAEND bit should be very small to avoid getting a SetupEnd error condition.

When the host moves to the status stage of the request, a second endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software. The second interrupt is just a confirmation that the request completed successfully. For SET_ADDRESS command, the address should be set in the FADDR register only after the status stage interrupt is received.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it has been decoded, the PERI_CSR0 register should be written to set the SERV_RXPKTRDY bit and to set the SENDSTALL bit. When the host moves to the status stage of the request, the controller will send a STALL to tell the host that the request was not executed. A second endpoint 0 interrupt will be generated and the SENTSTALL bit in the PERI_CSR0 register will be set.

If the host sends more data after the DATAEND bit has been set, then the controller will send a STALL. An endpoint 0 interrupt will be generated and the SENTSTALL bit in the PERI_CSR0 register will be set.

NOTE: DMA is not supported for endpoint 0, so the command should be read by accessing the endpoint 0 FIFO register.

17.2.8.3.2 Write Requests

Write requests involve an additional packet (or packets) of data being sent from the host after the 8-byte command. An example of a Write standard device request is: SET_DESCRIPTOR.

The sequence of events will begin, as with all requests, when the software receives an endpoint 0 interrupt. The RXPKTRDY bit of PERI_CSR0 will also have been set. The 8-byte command should then be read from the Endpoint 0 FIFO and decoded.

As with a zero data request, the PERI_CSR0 register should then be written to set the SERV_RXPKTRDY bit (indicating that the command has been read from the FIFO) but in this case the DATAEND bit should not be set (indicating that more data is expected).

When a second endpoint 0 interrupt is received, the PERI_CSR0 register should be read to check the endpoint status. The RXPKTRDY bit in the PERI_CSR0 register should be set to indicate that a data packet has been received. The COUNT0 register should then be read to determine the size of this data packet. The data packet can then be read from the endpoint 0 FIFO.

If the length of the data associated with the request (indicated by the wLength field in the command) is greater than the maximum packet size for endpoint 0, further data packets will be sent. In this case, PERI_CSR0 should be written to set the SERV_RXPKTRDY bit, but the DATAEND bit should not be set.

When all the expected data packets have been received, the PERI_CSR0 register should be written to set the SERV_RXPKTRDY bit and to set the DATAEND bit (indicating that no more data is expected).

When the host moves to the status stage of the request, another endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software, the interrupt is just a confirmation that the request completed successfully.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it has been decoded, the PERI_CSR0 register should be written to set the SERV_RXPKTRDY bit and to set the SENDSTALL bit. When the host sends more data, the controller will send a STALL to tell the host that the request was not executed. An endpoint 0 interrupt will be generated and the SENTSTALL bit in the PERI_CSR0 register will be set.

If the host sends more data after the DATAEND bit has been set, then the controller will send a STALL. An endpoint 0 interrupt will be generated and the SENTSTALL bit will be set.

17.2.8.3.3 Read Requests

Read requests have a packet (or packets) of data sent from the function to the host after the 8-byte command. Examples of Read Standard Device Requests are:

- GET_CONFIGURATION
- GET_INTERFACE
- GET_DESCRIPTOR
- GET_STATUS
- SYNCH_FRAME

The sequence of events will begin, as with all requests, when the software receives an endpoint 0 interrupt. The RXPKTRDY bit in the PERI_CSR0 register will also have been set. Next, the 8-byte command should be read from the endpoint 0 FIFO and decoded. The PERI_CSR0 register should then be written to set the SERV_RXPKTRDY bit (indicating that the command has read from the FIFO).

The data to be sent to the host should be written to the endpoint 0 FIFO. If the data to be sent is greater than the maximum packet size for endpoint 0, only the maximum packet size should be written to the FIFO. The PERI_CSR0 register should then be written to set the TXPKTRDY bit (indicating that there is a packet in the FIFO to be sent). When the packet has been sent to the host, another endpoint 0 interrupt will be generated and the next data packet can be written to the FIFO.

When the last data packet has been written to the FIFO, the PERI_CSR0 register should be written to set the TXPKTRDY bit and to set the DATAEND bit (indicating that there is no more data after this packet).

When the host moves to the status stage of the request, another endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software: the interrupt is just a confirmation that the request completed successfully.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it has been decoded, the PERI_CSR0 register should be written to set the SERV_RXPKTRDY bit and to set the SENDSTALL bit. When the host requests data, the controller will send a STALL to tell the host that the request was not executed. An endpoint 0 interrupt will be generated and the SENTSTALL bit in the PERI_CSR0 register will be set.

If the host requests more data after the DATAEND bit has been set, then the controller will send a STALL. An endpoint 0 interrupt will be generated and the SENTSTALL bit will be set.

17.2.8.3.4 Endpoint 0 States

The endpoint 0 control needs three modes – IDLE, TX, and RX – corresponding to the different phases of the control transfer and the states endpoint 0 enters for the different phases of the transfer (described in later sections).

The default mode on power-up or reset should be IDLE. The RXPKTRDY bit in the PERI_CSR0 register becoming set when endpoint 0 is in IDLE state indicates a new device request. Once the device request is unloaded from the FIFO, the controller decodes the descriptor to find whether there is a data phase and, if so, the direction of the data phase of the control transfer (in order to set the FIFO direction). See Figure 17-5.

Depending on the direction of the data phase, endpoint 0 goes into either TX state or RX state. If there is no Data phase, endpoint 0 remains in IDLE state to accept the next device request.

The actions that the CPU needs to take at the different phases of the possible transfers (for example, loading the FIFO, setting TXPKTRDY) are indicated in Figure 17-6 .

NOTE: The controller changes the FIFO direction, depending on the direction of the data phase independently of the CPU.

Figure 17-5. CPU Actions at Transfer Phases

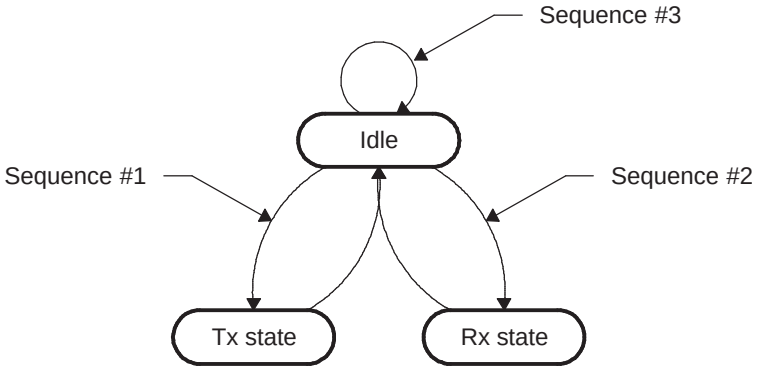
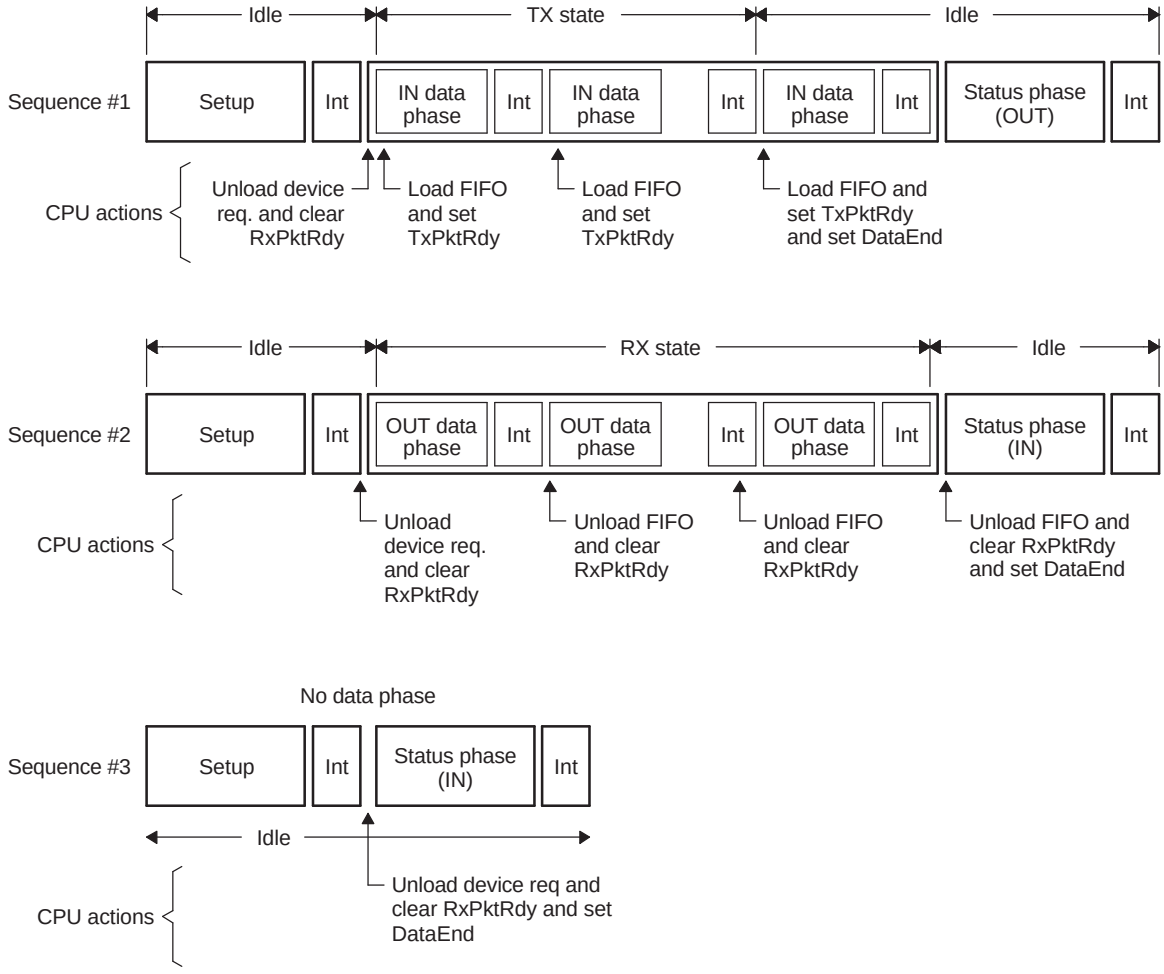


Figure 17-6. Sequence of Transfer



17.2.8.3.5 Endpoint 0 Service Routine

An Endpoint 0 interrupt is generated when:

- The controller sets the RXPKTRDY bit in the PERI_CSR0 register after a valid token has been received and data has been written to the FIFO.
- The controller clears the TXPKTRDY bit of PERI_CSR0 after the packet of data in the FIFO has been successfully transmitted to the host.
- The controller sets the SENTSTALL bit of PERI_CSR0 after a control transaction is ended due to a protocol violation.
- The controller sets the SETUPEND bit of PERI_CSR0 because a control transfer has ended before DATAEND is set.

Whenever the endpoint 0 service routine is entered, the software must first check to see if the current control transfer has been ended due to either a STALL condition or a premature end of control transfer. If the control transfer ends due to a STALL condition, the SENTSTALL bit would be set. If the control transfer ends due to a premature end of control transfer, the SETUPEND bit would be set. In either case, the software should abort processing the current control transfer and set the state to IDLE.

Once the software has determined that the interrupt was not generated by an illegal bus state, the next action taken depends on the endpoint state. Figure 17-7 shows the flow of this process.

If endpoint 0 is in IDLE state, the only valid reason an interrupt can be generated is as a result of the controller receiving data from the bus. The service routine must check for this by testing the RXPKTRDY bit of PERI_CSR0. If this bit is set, then the controller has received a SETUP packet. This must be unloaded from the FIFO and decoded to determine the action the controller must take. Depending on the command contained within the SETUP packet, endpoint 0 will enter one of three states:

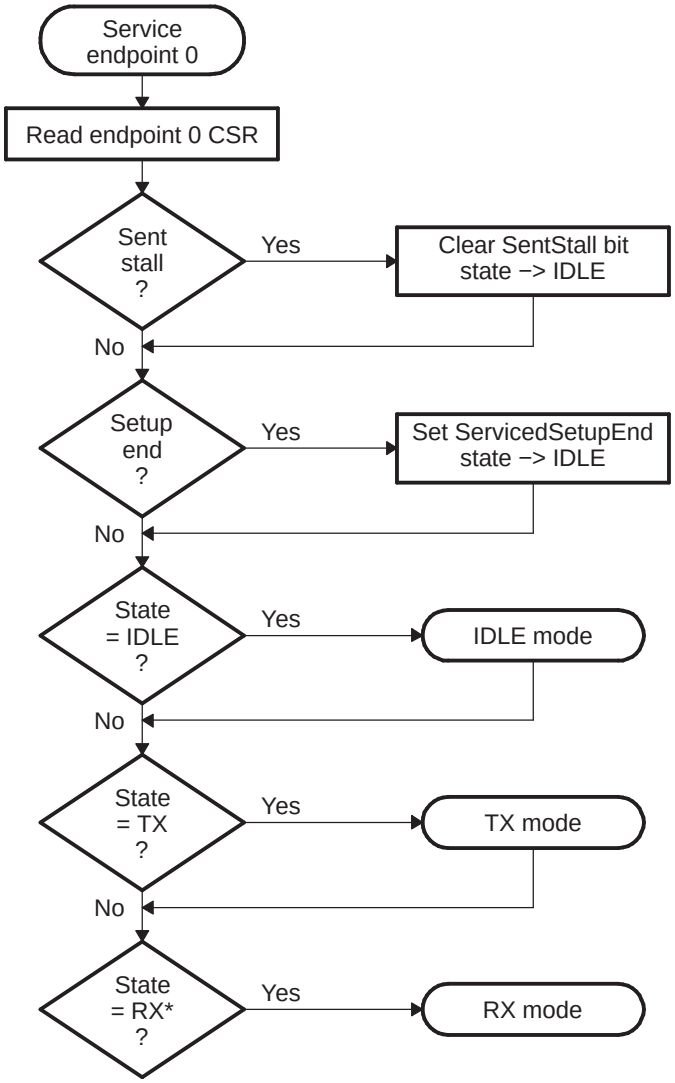
- If the command is a single packet transaction (SET_ADDRESS, SET_INTERFACE etc.) without any data phase, the endpoint will remain in IDLE state.
- If the command has an OUT data phase (SET_DESCRIPTOR etc.), the endpoint will enter RX state.
- If the command has an IN data phase (GET_DESCRIPTOR etc.), the endpoint will enter TX state.

If the endpoint 0 is in TX state, the interrupt indicates that the core has received an IN token and data from the FIFO has been sent. The software must respond to this either by placing more data in the FIFO if the host is still expecting more data or by setting the DATAEND bit to indicate that the data phase is complete. Once the data phase of the transaction has been completed, endpoint 0 should be returned to IDLE state to await the next control transaction.

NOTE: All command transactions include a field that indicates the amount of data the host expects to receive or is going to send.

If the endpoint is in RX state, the interrupt indicates that a data packet has been received. The software must respond by unloading the received data from the FIFO. The software must then determine whether it has received all of the expected data. If it has, the software should set the DATAEND bit and return endpoint 0 to IDLE state. If more data is expected, the firmware should set the SERV_RXPKTRDY bit of PERI_CSR0 to indicate that it has read the data in the FIFO and leave the endpoint in RX state.

Figure 17-7. Service Endpoint 0 Flow Chart

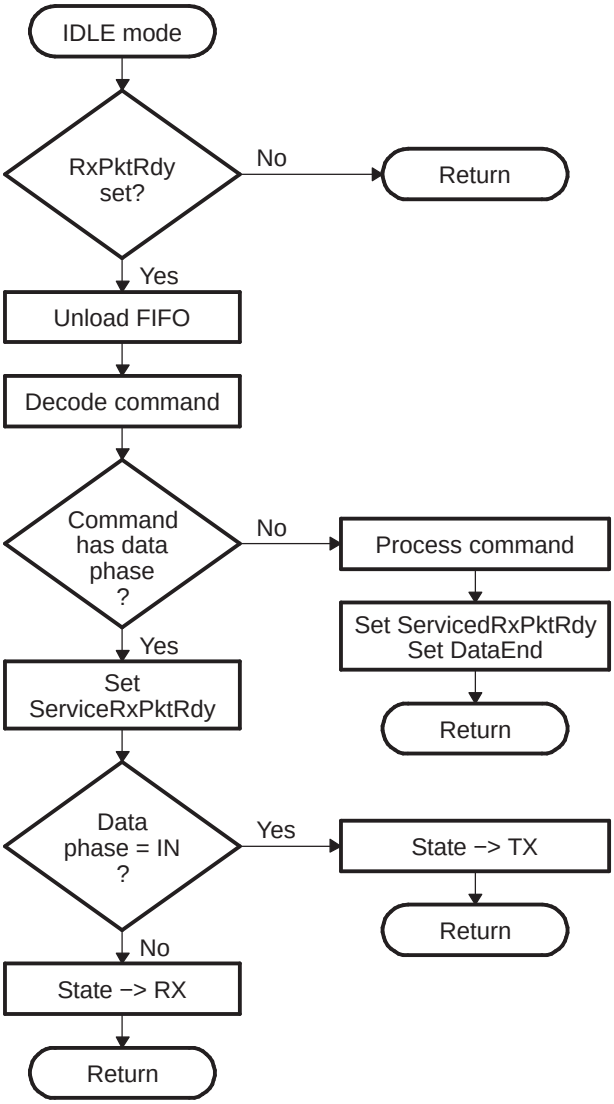


* By default

17.2.8.3.5.1 IDLE Mode

IDLE mode is the mode the endpoint 0 control must select at power-on or reset and is the mode to which the endpoint 0 control should return when the RX and TX modes are terminated. It is also the mode in which the SETUP phase of control transfer is handled (as outlined in Figure 17-8).

Figure 17-8. IDLE Mode Flow Chart



17.2.8.3.5.2 TX Mode

When the endpoint is in TX state all arriving IN tokens need to be treated as part of a data phase until the required amount of data has been sent to the host. If either a SETUP or an OUT token is received while the endpoint is in the TX state, this will cause a SetupEnd condition to occur as the core expects only IN tokens. See Figure 17-9.

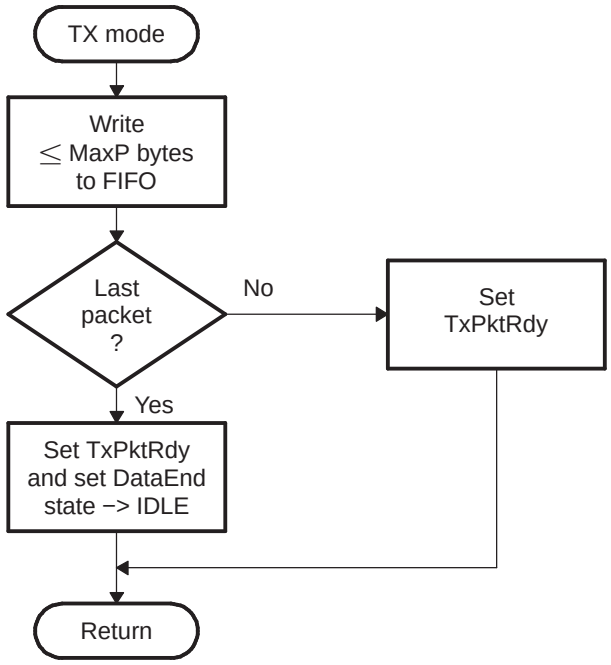
Three events can cause TX mode to be terminated before the expected amount of data has been sent:

1. The host sends an invalid token causing a SETUPEND condition (bit 4 of PERI_CSR0 set).
2. The software sends a packet containing less than the maximum packet size for endpoint 0.
3. The software sends an empty data packet.

Until the transaction is terminated, the software simply needs to load the FIFO when it receives an interrupt that indicates a packet has been sent from the FIFO. (An interrupt is generated when TXPKTRDY is cleared.)

When the software forces the termination of a transfer (by sending a short or empty data packet), it should set the DATAEND bit of PERI_CSR0 (bit 3) to indicate to the core that the data phase is complete and that the core should next receive an acknowledge packet.

Figure 17-9. TX Mode Flow Chart



17.2.8.3.5.3 RX Mode

In RX mode, all arriving data should be treated as part of a data phase until the expected amount of data has been received. If either a SETUP or an IN token is received while the endpoint is in RX state, a SetupEnd condition will occur as the controller expects only OUT tokens.

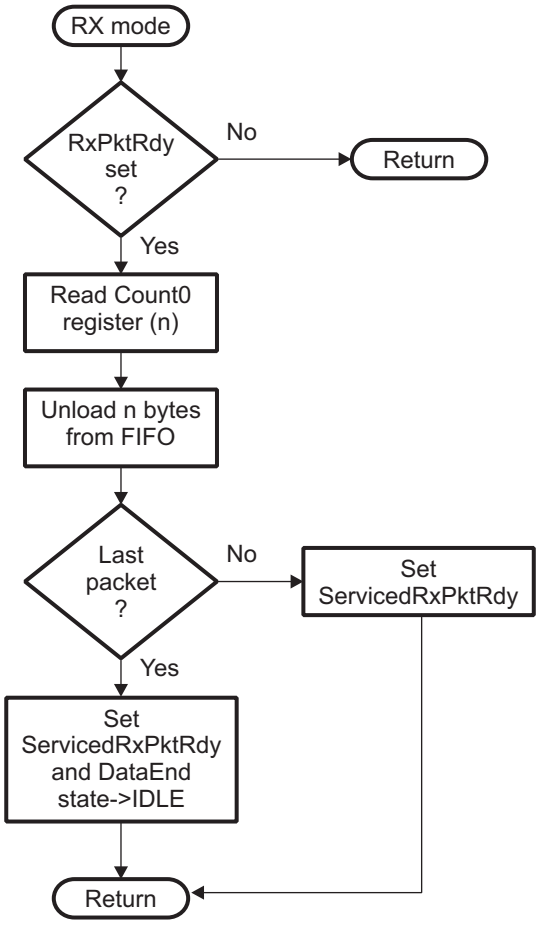
Three events can cause RX mode to be terminated before the expected amount of data has been received as shown in Figure 17-10:

1. The host sends an invalid token causing a SETUPEND condition (setting bit 4 of PERI_CSR0).
2. The host sends a packet which contains less than the maximum packet size for endpoint 0.
3. The host sends an empty data packet.

Until the transaction is terminated, the software unloads the FIFO when it receives an interrupt that indicates new data has arrived (setting RXPKTRDY bit of PERI_CSR0) and to clear RXPKTRDY by setting the SERV_RXPKTRDY bit of PERI_CSR0 (bit 6).

When the software detects the termination of a transfer (by receiving either the expected amount of data or an empty data packet), it should set the DATAEND bit (bit 3 of PERI_CSR0) to indicate to the controller that the data phase is complete and that the core should receive an acknowledge packet next.

Figure 17-10. RX Mode Flow Chart



17.2.8.3.5.4 Error Handling

A control transfer may be aborted due to a protocol error on the USB, the host prematurely ending the transfer, or if the software wishes to abort the transfer (for example, because it cannot process the command).

The controller automatically detects protocol errors and sends a STALL packet to the host under the following conditions:

- The host sends more data during the OUT Data phase of a write request than was specified in the command. This condition is detected when the host sends an OUT token after the DATAEND bit (bit 3 of PERI_CSR0) has been set.
- The host requests more data during the IN Data phase of a read request than was specified in the command. This condition is detected when the host sends an IN token after the DATAEND bit in the PERI_CSR0 register has been set.
- The host sends more than Max Packet Size data bytes in an OUT data packet.
- The host sends a non-zero length DATA1 packet during the STATUS phase of a read request.

When the controller has sent the STALL packet, it sets the SENTSTALL bit (bit 2 of PERI_CSR0) and generates an interrupt. When the software receives an endpoint 0 interrupt with the SENTSTALL bit set, it should abort the current transfer, clear the SENTSTALL bit, and return to the IDLE state.

If the host prematurely ends a transfer by entering the STATUS phase before all the data for the request has been transferred, or by sending a new SETUP packet before completing the current transfer, then the SETUPEND bit (bit 4 of PERI_CSR0) will be set and an endpoint 0 interrupt generated. When the software receives an endpoint 0 interrupt with the SETUPEND bit set, it should abort the current transfer, set the SERV_SETUPEND bit (bit 7 of PERI_CSR0), and return to the IDLE state. If the RXPKTRDY bit (bit 0 of PERI_CSR0) is set this indicates that the host has sent another SETUP packet and the software should then process this command.

If the software wants to abort the current transfer, because it cannot process the command or has some other internal error, then it should set the SENDSTALL bit (bit 5 of PERI_CSR0). The controller will then send a STALL packet to the host, set the SENTSTALL bit (bit 2 of PERI_CSR0) and generate an endpoint 0 interrupt.

17.2.8.3.5.5 Additional Conditions

The controller automatically responds to certain conditions on the USB bus or actions by the host. The details are:

- Stall Issued to Control Transfers
  - 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 controller when the host sends an OUT token (instead of an IN token) after the software has unloaded the last OUT packet and set DataEnd.
  - 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 controller when the host sends an IN token (instead of an OUT token) after the software has cleared TXPKTRDY and set DataEnd in response to the ACK issued by the host to what should have been the last packet.
  - The host sends more than MaxPktSize data with an OUT data token.
  - The host sends the wrong PID for the OUT Status phase of a Control transfer.
  - The host sends more than a zero length data packet for the OUT Status phase.
- Zero Length Out Data Packets In Control Transfer
  - 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 (that is, after the software has set DataEnd). If, however, the host sends a zero length OUT data packet before the entire length of device request has been transferred, this signals the premature end of the transfer. In this case, the controller will automatically flush any IN token loaded by software ready for the Data phase from the FIFO and set SETUPEND bit (bit 4 of PERI_CSR0).

17.2.8.4 Bulk Transactions

17.2.8.4.1 Bulk In Transactions

A Bulk IN transaction is used to transfer non-periodic data from the USB peripheral device to the host.

The following optional features are available for use with a Tx endpoint for Bulk IN transactions:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO awaiting transmission to the host. Double packet buffering is enabled by setting the DPB bit of TXFIFOSZ register (bit 4).
- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint is able to accept another packet in its FIFO. This feature allows the DMA controller to load packets into the FIFO without processor intervention.

  When DMA is enabled and DMAMODE bit of PERI_TXCSR is set, an endpoint interrupt is not generated for completion of the packet transfer. An endpoint interrupt is generated only in the error conditions.

17.2.8.4.1.1 Setup

In configuring a TX endpoint for bulk transactions, the TXMAXP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint and the PERI_TXCSR register should be set as shown in Table 17-4 when using DMA:

Table 17-4. PERI_TXCSR Register Bit Configuration for Bulk IN Transactions

Bit Position	Bit Field Name	Configuration
Bit 14	ISO	Cleared to 0 for bulk mode operation.
Bit 13	MODE	Set to 1 to make sure the FIFO is enabled (only necessary if the FIFO is shared with an RX endpoint).
Bit 12	DMAEN	Set to 1 if DMA requests must be enabled.
Bit 11	FRCDATATOG	Cleared to 0 to allow normal data toggle operations.
Bit 10	DMAMODE	Set to 1 when DMA is enabled and EP interrupt is not needed for each packet transmission.

When the endpoint is first configured (following a SET_CONFIGURATION or SET_INTERFACE command on Endpoint 0), the lower byte of PERI_TXCSR should be written to set the CLRDATATOG bit (bit 6). This will ensure that the data toggle (which is handled automatically by the controller) starts in the correct state.

Also if there are any data packets in the FIFO, indicated by the FIFONOTEMPTY bit (bit 1 of PERI_TXCSR) being set, they should be flushed by setting the FLUSHFIFO bit (bit 3 of PERI_TXCSR).

NOTE: It may be necessary to set this bit twice in succession if double buffering is enabled.

17.2.8.4.1.2 Operation

When data is to be transferred over a Bulk IN pipe, a data packet needs to be loaded into the FIFO and the PERI_TXCSR register written to set the TXPKTRDY bit (bit 0). When the packet has been sent, the TXPKTRDY bit will be cleared by the USB controller and an interrupt generated so that the next packet can be loaded into the FIFO. If double packet buffering is enabled, then after the first packet has been loaded and the TXPKTRDY bit set, the TXPKTRDY bit will immediately be cleared by the USB controller and an interrupt generated so that a second packet can be loaded into the FIFO. The software should operate in the same way, loading a packet when it receives an interrupt, regardless of whether double packet buffering is enabled or not.

In the general case, the packet size must not exceed the size specified by the lower 11 bits of the TXMAXP register. This part of the register defines the payload (packet size) for transfers over the USB and is required by the USB Specification to be either 8, 16, 32, 64 (Full-Speed or High-Speed) or 512 bytes (High-Speed only).

The host may determine that all the data for a transfer has been sent by knowing the total amount of data that is expected. Alternatively it may infer that all the data has been sent when it receives a packet which is smaller than the stated payload (TXMAXP[10-0]). In the latter case, if the total size of the data block is a multiple of this payload, it will be necessary for the function to send a null packet after all the data has been sent. This is done by setting TXPKTRDY when the next interrupt is received, without loading any data into the FIFO.

If large blocks of data are being transferred, then the overhead of calling an interrupt service routine to load each packet can be avoided by using DMA.

17.2.8.4.1.3 Error Handling

If the software wants to shut down the Bulk IN pipe, it should set the SENDSTALL bit (bit 4 of PERI_TXCSR). When the controller receives the next IN token, it will send a STALL to the host, set the SENTSTALL bit (bit 5 of PERI_TXCSR) and generate an interrupt.

When the software receives an interrupt with the SENTSTALL bit (bit 5 of PERI_TXCSR) set, it should clear the SENTSTALL bit. It should however leave the SENDSTALL bit set until it is ready to re-enable the Bulk IN pipe.

NOTE: If the host failed to receive the STALL packet for some reason, it will send another IN token, so it is advisable to leave the SENDSTALL bit set until the software is ready to re-enable the Bulk IN pipe. When a pipe is re-enabled, the data toggle sequence should be restarted by setting the CLRDATATOG bit in the PERI_TXCSR register (bit 6).

17.2.8.4.2 Bulk OUT Transactions

A Bulk OUT transaction is used to transfer non-periodic data from the host to the function controller.

The following optional features are available for use with an Rx endpoint for Bulk OUT transactions:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO on reception from the host. Double packet buffering is enabled by setting the DPB bit of the RXFIFOSZ register (bit 4).
- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint has a packet in its FIFO. This feature can be used to allow the DMA controller to unload packets from the FIFO without processor intervention.

  When DMA is enabled, endpoint interrupt will not be generated for completion of packet reception. Endpoint interrupt will be generated only in the error conditions.

17.2.8.4.2.1 Setup

In configuring an Rx endpoint for Bulk OUT transactions, the RXMAXP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint. In addition, the relevant interrupt enable bit in the INTRRXE register should be set (if an interrupt is required for this endpoint) and the PERI_RXCSR register should be set as shown in Table 17-5.

Table 17-5. PERI_RXCSR Register Bit Configuration for Bulk OUT Transactions

Bit Position	Bit Field Name	Configuration
Bit 14	ISO	Cleared to 0 to enable Bulk protocol.
Bit 13	DMAEN	Set to 1 if a DMA request is required for this endpoint.
Bit 12	DISNYET	Cleared to 0 to allow normal PING flow control. This will affect only high speed transactions.
Bit 11	DMAMODE	Always clear this bit to 0.

When the endpoint is first configured (following a SET_CONFIGURATION or SET_INTERFACE command on Endpoint 0), the lower byte of PERI_RXCSR should be written to set the CLRDATATOG bit (bit 7). This will ensure that the data toggle (which is handled automatically by the USB controller) starts in the correct state.

Also if there are any data packets in the FIFO (indicated by the RXPKTRDY bit (bit 0 of PERI_RXCSR) being set), they should be flushed by setting the FLUSHFIFO bit (bit 4 of PERI_RXCSR).

NOTE: It may be necessary to set this bit twice in succession if double buffering is enabled.

17.2.8.4.2.2 Operation

When a data packet is received by a Bulk Rx endpoint, the RXPKTRDY bit (bit 0 of PERI_RXCSR) is set and an interrupt is generated. The software should read the RXCOUNT register for the endpoint to determine the size of the data packet. The data packet should be read from the FIFO, then the RXPKTRDY bit should be cleared.

The packets received should not exceed the size specified in the RXMAXP register (as this should be the value set in the wMaxPacketSize field of the endpoint descriptor sent to the host). When a block of data larger than wMaxPacketSize needs to be sent to the function, it will be sent as multiple packets. All the packets will be wMaxPacketSize in size, except the last packet which will contain the residue. The software may use an application specific method of determining the total size of the block and hence when the last packet has been received. Alternatively it may infer that the entire block has been received when it receives a packet which is less than wMaxPacketSize in size. (If the total size of the data block is a multiple of wMaxPacketSize, a null data packet will be sent after the data to signify that the transfer is complete.)

In the general case, the application software will need to read each packet from the FIFO individually. If large blocks of data are being transferred, the overhead of calling an interrupt service routine to unload each packet can be avoided by using DMA.

17.2.8.4.2.3 Error Handling

If the software wants to shut down the Bulk OUT pipe, it should set the SENDSTALL bit (bit 5 of PERI_RXCSR). When the controller receives the next packet it will send a STALL to the host, set the SENTSTALL bit (bit 6 of PERI_RXCSR) and generate an interrupt.

When the software receives an interrupt with the SENTSTALL bit (bit 6 of PERI_RXCSR) set, it should clear this bit. It should however leave the SENDSTALL bit set until it is ready to re-enable the Bulk OUT pipe.

NOTE: If the host failed to receive the STALL packet for some reason, it will send another packet, so it is advisable to leave the SENDSTALL bit set until the software is ready to re-enable the Bulk OUT pipe. When a Bulk OUT pipe is re-enabled, the data toggle sequence should be restarted by setting the CLRDATATOG bit (bit 7) in the PERI_RXCSR register.

17.2.8.5 Interrupt Transactions

An Interrupt IN transaction uses the same protocol as a Bulk IN transaction and can be used the same way. Similarly, an Interrupt OUT transaction uses almost the same protocol as a Bulk OUT transaction and can be used the same way.

Tx endpoints in the USB controller have one feature for Interrupt IN transactions that they do not support in Bulk IN transactions. In Interrupt IN transactions, the endpoints support continuous toggle of the data toggle bit.

This feature is enabled by setting the FRCDATATOG bit in the PERI_TXCSR register (bit 11). When this bit is set, the controller will consider the packet as having been successfully sent and toggle the data bit for the endpoint, regardless of whether an ACK was received from the host.

Another difference is that interrupt endpoints do not support PING flow control. This means that the controller should never respond with a NYET handshake, only ACK/NAK/STALL. To ensure this, the DISNYET bit in the PERI_RXCSR register (bit 12) should be set to disable the transmission of NYET handshakes in high-speed mode.

Though DMA can be used with an interrupt OUT endpoint, it generally offers little benefit as interrupt endpoints are usually expected to transfer all their data in a single packet.

17.2.8.6 Isochronous Transactions

17.2.8.6.1 Isochronous IN Transactions

An Isochronous IN transaction is used to transfer periodic data from the function controller to the host.

The following optional features are available for use with a Tx endpoint for Isochronous IN transactions:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO awaiting transmission to the host. Double packet buffering is enabled by setting the DPB bit of TXFIFOSZ register (bit 4).

  NOTE: Double packet buffering is generally advisable for Isochronous transactions in order to avoid Underrun errors as described in later section.

- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint is able to accept another packet in its FIFO. This feature allows the DMA controller to load packets into the FIFO without processor intervention.

  However, this feature is not particularly useful with Isochronous endpoints because the packets transferred are often not maximum packet size and the PERI_TXCSR register needs to be accessed following every packet to check for Underrun errors.

  When DMA is enabled and DMAMODE bit of PERI_TXCSR is set, endpoint interrupt will not be generated for completion of packet transfer. Endpoint interrupt will be generated only in the error conditions.

17.2.8.6.1.1 Setup

In configuring a Tx endpoint for Isochronous IN transactions, the TXMAXP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint. In addition, the relevant interrupt enable bit in the INTRTXE register should be set (if an interrupt is required for this endpoint) and the PERI_TXCSR register should be set as shown in Table 17-6.

Table 17-6. PERI_TXCSR Register Bit Configuration for Isochronous IN Transactions

Bit Position	Bit Field Name	Configuration
Bit 14	ISO	Set to 1 to enable Isochronous transfer protocol.
Bit 13	MODE	Set to 1 to ensure the FIFO is enabled (only necessary if the FIFO is shared with an Rx endpoint).
Bit 12	DMAEN	Set to 1 if DMA Requests have to be enabled.
Bit 11	FRCDATATOG	Ignored in Isochronous mode.
Bit 10	DMAMODE	Set it to 1, when DMA is enabled and EP interrupt is not needed for each packet transmission.

17.2.8.6.1.2 Operation

An Isochronous endpoint does not support data retries, so if data underrun is to be avoided, the data to be sent to the host must be loaded into the FIFO before the IN token is received. The host will send one IN token per frame (or microframe in High-speed mode), however the timing within the frame (or microframe) can vary. If an IN token is received near the end of one frame and then at the start of the next frame, there will be little time to reload the FIFO. For this reason, double buffering of the endpoint is usually necessary.

An interrupt is generated whenever a packet is sent to the host and the software may use this interrupt to load the next packet into the FIFO and set the TXPKTRDY bit in the PERI_TXCSR register (bit 0) in the same way as for a Bulk Tx endpoint. As the interrupt could occur almost any time within a frame(/microframe), depending on when the host has scheduled the transaction, this may result in irregular timing of FIFO load requests. If the data source for the endpoint is coming from some external hardware, it may be more convenient to wait until the end of each frame(/microframe) before loading the FIFO as this will minimize the requirement for additional buffering. This can be done by using either the SOF interrupt or the external SOF_PULSE signal from the controller to trigger the loading of the next data packet. The SOF_PULSE is generated once per frame(/microframe) when a SOF packet is received. (The controller also maintains an external frame(/microframe) counter so it can still generate a SOF_PULSE when the SOF packet has been lost.) The interrupts may still be used to set the TXPKTRDY bit in PERI_TXCSR (bit 0) and to check for data overruns/underruns.

Starting up a double-buffered Isochronous IN pipe can be a source of problems. Double buffering requires that a data packet is not transmitted until the frame(/microframe) after it is loaded. There is no problem if the function loads the first data packet at least a frame(/microframe) before the host sets up the pipe (and therefore starts sending IN tokens). But if the host has already started sending IN tokens by the time the first packet is loaded, the packet may be transmitted in the same frame(/microframe) as it is loaded, depending on whether it is loaded before, or after, the IN token is received. This potential problem can be avoided by setting the ISOUPDATE bit in the POWER register (bit 7). When this bit is set, any data packet loaded into an Isochronous Tx endpoint FIFO will not be transmitted until after the next SOF packet has been received, thereby ensuring that the data packet is not sent too early.

17.2.8.6.1.3 Error Handling

If the endpoint has no data in its FIFO when an IN token is received, it will send a null data packet to the host and set the UNDERRUN bit in the PERI_TXCSR register (bit 2). This is an indication that the software is not supplying data fast enough for the host. It is up to the application to determine how this error condition is handled.

If the software is loading one packet per frame(/microframe) and it finds that the TXPKTRDY bit in the PERI_TXCSR register (bit 0) is set when it wants to load the next packet, this indicates that a data packet has not been sent (perhaps because an IN token from the host was corrupted). It is up to the application how it handles this condition: it may choose to flush the unsent packet by setting the FLUSHFIFO bit in the PERI_TXCSR register (bit 3), or it may choose to skip the current packet.

17.2.8.6.2 Isochronous OUT Transactions

An Isochronous OUT transaction is used to transfer periodic data from the host to the function controller.

Following optional features are available for use with an Rx endpoint for Isochronous OUT transactions:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO on reception from the host. Double packet buffering is enabled by setting the DPB bit of RXFIFOSZ register (bit 4).

  NOTE: Double packet buffering is generally advisable for Isochronous transactions in order to avoid Overrun errors.

- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint has a packet in its FIFO. This feature can be used to allow the DMA controller to unload packets from the FIFO without processor intervention.

  However, this feature is not particularly useful with Isochronous endpoints because the packets transferred are often not maximum packet size and the PERI_RXCSR register needs to be accessed following every packet to check for Overrun or CRC errors.

  When DMA is enabled, endpoint interrupt will not be generated for completion of packet reception. Endpoint interrupt will be generated only in the error conditions.

17.2.8.6.2.1 Setup

In configuring an Rx endpoint for Isochronous OUT transactions, the RXMAXP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint. In addition, the relevant interrupt enable bit in the INTRRXE register should be set (if an interrupt is required for this endpoint) and the PERI_RXCSR register should be set as shown in Table 17-7.

Table 17-7. PERI_RXCSR Register Bit Configuration for Isochronous OUT Transactions

Bit Position	Bit Field Name	Configuration
Bit 14	ISO	Set to 1 to enable isochronous protocol.
Bit 13	DMAEN	Set to 1 if a DMA request is required for this endpoint.
Bit 12	DISNYET	Ignored in isochronous transfers.
Bit 11	DMAMODE	Always clear this bit to 0.

17.2.8.6.2.2 Operation

An Isochronous endpoint does not support data retries so, if a data overrun is to be avoided, there must be space in the FIFO to accept a packet when it is received. The host will send one packet per frame (or microframe in High-speed mode); however, the time within the frame can vary. If a packet is received near the end of one frame(/microframe) and another arrives at the start of the next frame, there will be little time to unload the FIFO. For this reason, double buffering of the endpoint is usually necessary.

An interrupt is generated whenever a packet is received from the host and the software may use this interrupt to unload the packet from the FIFO and clear the RXPKTRDY bit in the PERI_RXCSR register (bit 0) in the same way as for a Bulk Rx endpoint. As the interrupt could occur almost any time within a frame(/microframe), depending on when the host has scheduled the transaction, the timing of FIFO unload requests will probably be irregular. If the data sink for the endpoint is going to some external hardware, it may be better to minimize the requirement for additional buffering by waiting until the end of each frame(/microframe) before unloading the FIFO. This can be done by using either the SOF interrupt or the external SOF_PULSE signal from the controller to trigger the unloading of the data packet. The SOF_PULSE is generated once per frame(/microframe) when a SOF packet is received. (The controller also maintains an external frame(/microframe) counter so it can still generate a SOF_PULSE when the SOF packet has been lost.) The interrupts may still be used to clear the RXPKTRDY bit in PERI_RXCSR and to check for data overruns/underruns.

17.2.8.6.2.3 Error Handling

If there is no space in the FIFO to store a packet when it is received from the host, the OVERRUN bit in the PERI_RXCSR register (bit 2) will be set. This is an indication that the software is not unloading data fast enough for the host. It is up to the application to determine how this error condition is handled.

If the controller finds that a received packet has a CRC error, it will still store the packet in the FIFO and set the RXPKTRDY bit (bit 0 of PERI_RXCSR) and the DATAERROR bit (bit 3 of PERI_RXCSR). It is left up to the application how this error condition is handled.

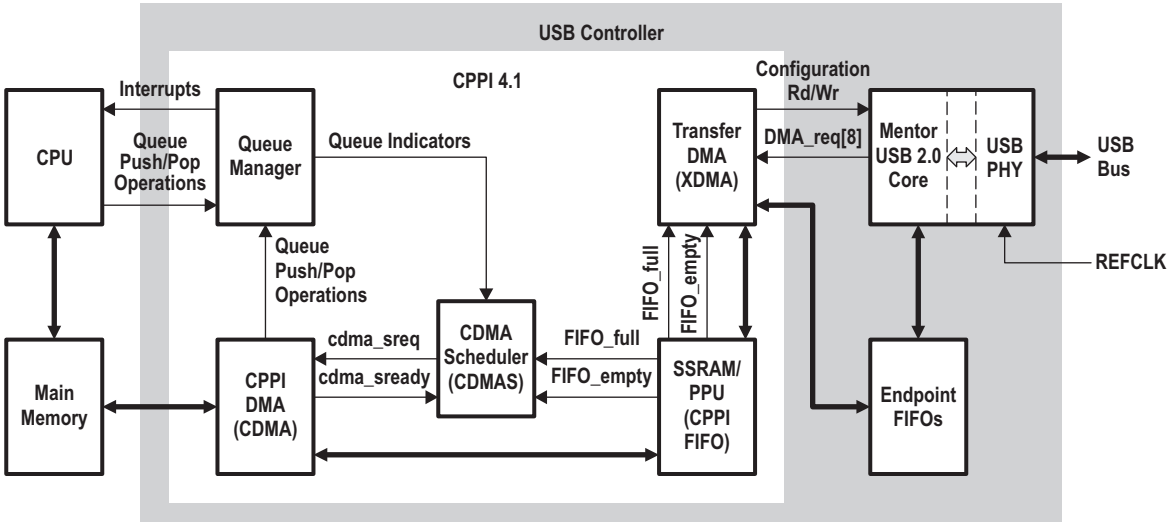
17.2.9 Communications Port Programming Interface (CPPI) 4.1 DMA Overview for TMS320C5515

The CPPI DMA module supports the transmission and reception of USB packets. The CPPI DMA is designed to facilitate the segmentation and reassembly of CPPI compliant packets to/from smaller data blocks that are natively compatible with the specific requirements of each networking port. Multiple Tx and Rx channels are provided within the DMA which allow multiple segmentation or reassembly operations to be effectively performed in parallel (but not actually simultaneously). The DMA controller maintains state information for each of the ports/channels which allows packet segmentation and reassembly operations to be time division multiplexed between channels in order to share the underlying DMA hardware. A DMA scheduler is used to control the ordering and rate at which this multiplexing occurs.

The CPPI (version 4.1) DMA controller sub-module is a common 4 dual-port DMA Controller. It supports 4 Tx and 4 Rx Ports and each port attaches to the associated endpoint in the controller. Port 1 maps to endpoint 1 and Port 2 maps to endpoint 2 and Port 3 maps to endpoint 3 and Port 4 maps to endpoint 4, while endpoint 0 can not utilize the DMA and the firmware is responsible to load or offload the endpoint 0 FIFO via CPU.

Figure 17-11 displays the USB controller block diagram.

Figure 17-11. USB Controller Block Diagram



Host— The host is an intelligent system resource that configures and manages each communications control module. The host is responsible for allocating memory, initializing all data structures, and responding to port interrupts.

Main Memory— The area of data storage managed by the CPU. The CPPI DMA (CDMA) reads and writes CPPI packets from and to main memory. This memory can exist internal or external from the device.

Queue Manager (QM)— The QM is responsible for accelerating management of a variety of Packet Queues and Free Descriptor / Buffer Queues. It provides status indications to the CDMA Scheduler when queues are empty or full.

CPPI DMA (CDMA)— The CDMA is responsible for transferring data between the CPPI FIFO and Main Memory. It acquires free Buffer Descriptor from the QM (Receive Submit Queue) for storage of received data, posts received packets pointers to the Receive Completion Queue, transmits packets stored on the Transmit Submit Queue (Transmit Queue) , and posts completed transmit packets to the Transmit Completion Queue.

CDMA Scheduler (CDMAS)— The CDMAS is responsible for scheduling CDMA transmit and receive operations. It uses Queue Indicators from the QM and the CDMA to determine the types of operations to schedule.

CPPI FIFO— The CPPI FIFO provides 8 FIFO interfaces (one for each of the 4 transmit and receive endpoints). Each FIFO contains two 64-byte memory storage elements (ping-pong buffer storage).

Transfer DMA (XDMA)— The XDMA receives DMA requests from the Mentor USB 2.0 Core and initiates DMAs to the CPPI FIFO.

Endpoint FIFOs— The Endpoint FIFOs are the USB packet storage elements used by the Mentor USB 2.0 Core for packet transmission or reception. The XDMA transfers data between the CPPI FIFO and the Endpoint FIFOs for transmit operations and between the Endpoint FIFOs and the CPPI FIFO for receive operations.

Mentor USB 2.0 Core— This controller is responsible for processing USB bus transfers (control, bulk, interrupt, and isochronous). It supports 4 transmit and 4 receive endpoints in addition to endpoint 0 (control).

17.2.9.1 CPPI Terminology

The following terms are important in the discussion of DMA CPPI.

Port— A port is the communications module (peripheral hardware) that contains the control logic for Direct Memory Access for a single transmit/receive interface or set of interfaces. Each port may have multiple communication channels that transfer data using homogenous or heterogeneous protocols. A port is usually subdivided into transmit and receive pairs which are independent of each other. Each endpoint, excluding endpoint 0, has its own dedicated port.

Channel— A channel refers to the sub-division of information (flows) that is transported across ports. Each channel has associated state information. Channels are used to segregate information flows based on the protocol used, scheduling requirements (example: CBR, VBR, ABR), or concurrency requirements (that is, blocking avoidance). All four ports have dedicated single channels, channel 0, associated for their use in a USB application.

Data Buffer— A data buffer is a single data structure that contains payload information for transmission to or reception from a port. A data buffer is a byte aligned contiguous block of memory used to store packet payload data. A data buffer may hold any portion of a packet and may be linked together (via descriptors) with other buffers to form packets. Data buffers may be allocated anywhere within the device memory map. The Buffer Length field of the packet descriptor indicates the number of valid data bytes in the buffer. There may be from 1 to 4M-1 valid data bytes in each buffer.

Host Buffer Descriptor— A buffer descriptor is a single data structure that contains information about one or more data buffers. This type of descriptor is required when more than one descriptor is needed to define an entire packet, that is, it either defines the middle of a packet or end of a packet.

Host Packet Descriptor— A packet descriptor is another name for the first buffer descriptor within a packet. Some fields within a data buffer descriptor are only valid when it is a packet descriptor including the tags, packet length, packet type, and flags. This type of descriptor is always used to define a packet since it provides packet level information that is useful to both the ports and the Host in order to properly process the packet. It is the only descriptor used when single descriptor solely defines a packet. When multiple descriptors are needed to define a packet, the packet descriptor is the first descriptor used to define a packet.

Free Descriptor/Buffer Queue— A free descriptor/buffer queue is a hardware managed list of available descriptors with pre-linked empty buffers that are to be used by the receive ports for host type descriptors. Free Descriptor/Buffer Queues are implemented by the Queue Manager.

Teardown Descriptor— Teardown Descriptor is a special structure which is not used to describe either a packet or a buffer but is instead used to describe the completion of a channel halt and teardown event. Channel teardown is an important function because it ensures that when a connection is no longer needed that the hardware can be reliably halted and any remaining packets which had not yet been transmitted can be reclaimed by the Host without the possibility of losing buffer or descriptor references (which results in a memory leak).

Packet Queue— A packet queue is hardware managed list of valid (that is, populated) packet descriptors that is used for forwarding a packet from one entity to another for any number of purposes.

Queue Manager— The queue manager is a hardware module that is responsible for accelerating management of the packet queues. Packets are added to a packet queue by writing the 32-bit descriptor address to a particular memory mapped location in the Queue Manager module. Packets are de-queued by reading the same location for that particular queue. A single Queue Manager is used for a USB application.

NOTE: All descriptors (regardless of type) must be allocated at addresses that are naturally aligned to the smallest power of 2 that is equal to or greater than the descriptor size.

17.2.9.2 Host Packet Descriptor (SOP Descriptor)

Host Packet Descriptors are designed to be used when USB like application requires support for true, unlimited fragment count scatter/gather type operations. The Host Packet Descriptor is the first descriptor on multiple descriptors setup or the only descriptor in a single descriptors setup. The Host Packet Descriptor contains the following information:

- Indicator which identifies the descriptor as a Host Packet Descriptor (always 10h)
- Source and Destination Tags (Reserved)
- Packet Type
- Packet Length
- Protocol Specific Region Size
- Protocol Specific Control/Status Bits
- Pointer to the first valid byte in the SOP data buffer
- Length of the SOP data buffer
- Pointer to the next buffer descriptor in the packet

Host Packet Descriptors can vary in size of their defined fields from 32 bytes up to 104 bytes. Within this range, Host Packet Descriptors always contain 32 bytes of required information and may also contain 8 bytes of software specific tagging information and up to 64 bytes (indicated in 4 byte increments) of protocol specific information. How much protocol specific information (and therefore the allocated size of the descriptors) is application dependent.

NOTE: Descriptors can be located anywhere within the 16MB address space of the device (except for DARAM, which is not accessible by the USB controller). However, all descriptors must be placed in a single contiguous block of up to 64KW.

The Host Packet Descriptor layout is shown in Figure 17-12.

Figure 17-12. Host Packet Descriptor Layout

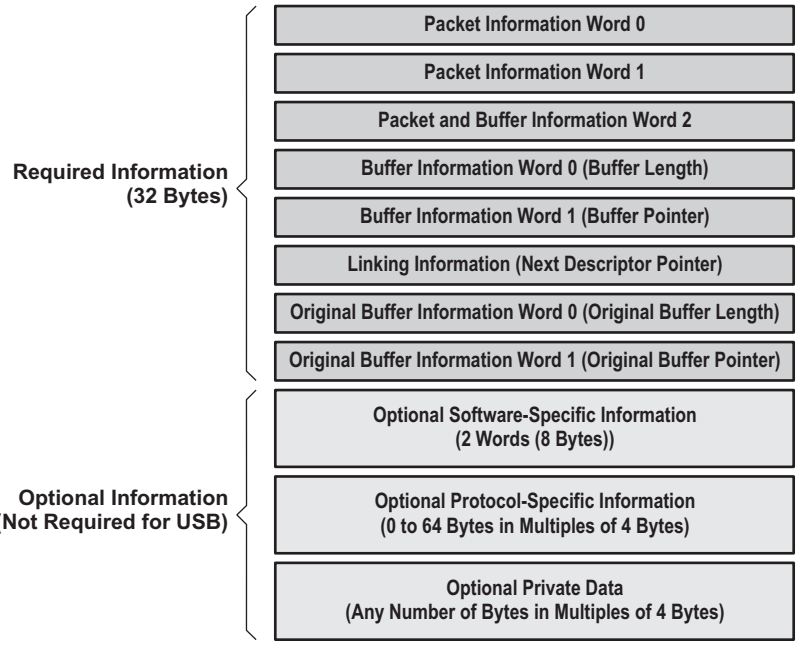


Table 17-8. Host Packet Descriptor Word 0 (HPD Word 0)

Bits	Name	Description
31-27	Descriptor Type	The Host Packet Descriptor Type is 16 decimal (10h). The CPU initializes this field.
26-22	Protocol Specific Valid Word Count	This field indicates the valid numberof 32-bit words in the protocol specific region. The CPU initializes this field. This is encoded in increments of 4 bytes as: 0 = 0 byte 1 = 4 bytes ... 16 = 64 bytes 17-31 = Reserved
21-0	Packet Length	The length of the packet in bytes. If the Packet Length is less than the sum of the buffer lengths, then the packet data will be truncated. A Packet Length greater than the sum of the buffers is an error. The valid range for the packet length is 0 to (4M - 1) bytes. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception.

Table 17-9. Host Packet Descriptor Word 1 (HPD Word 1)

Bits	Name	Description
31-27	Source Tag: Port #	This field indicates the port number (0-31) from which the packet originated. The DMA overwrites this field on packet reception. This is the RX Endpoint number from which the packet originated.
26-21	Source Tag: Channel #	This field indicates the channel number within the port from which the packet originated. The DMA overwrites this field on packet reception. This field is always 0-63.
20-16	Source Tag: Sub-channel #	This field indicates the sub-channel number (0-31) within the channel from which the packet originated. The DMA overwrites this field on packet reception. This field is always 0.
15-0	Destination Tag	This field is application specific. This field is always 0.

Table 17-10. Host Packet Descriptor Word 2 (HPD Word 2)

Bits	Name	Description
31	Packet Error	This bit indicates if an error occurred during reception of this packet (0 = No error occurred, 1 = Error occurred). The DMA overwrites this field on packet reception. Additional information about different errors may be encoded in the protocol specific fields in the descriptor.
30-26	Packet Type	This field indicates the type of this packet. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception. This field is encoded as: 0-4 = Reserved 5 = USB 6-31 = Reserved
25-20	Reserved	Reserved
19	Zero-length packet indicator	If a zero-length USB packet is received, the XDMA will send the CDMA a data block with a byte count of 0 and this bit is set. The CDMA will then perform normal EOP termination of the packet without transferring data. For transmit, if a packet has this bit set, the XDMA will ignore the CPPI packet size and send a zero-length packet to the USB controller.
18-16	Protocol Specific	This field contains protocol specific flags/information that can be assigned based on the packet type. Not used for USB.
15	Return Policy	This field indicates the return policy for this packet. The CPU initializes this field. 0 = Entire packet (still linked together) should be returned to the queue specified in bits 13-0. 1 = Each buffer should be returned to the queue specified in bits 13-0 of Word 2 in their respective descriptors. The Tx DMA will return each buffer in sequence.
14	On-chip	This field indicates whether or not this descriptor is in a region which is in on-chip memory space (1) or in external memory (0).
13-12	Packet Return Queue Mgr #	This field indicates which queue manager in the system the descriptor is to be returned to after transmission is complete. This field is not altered by the DMA during transmission or reception and is initialized by the CPU. There is only 1 Queue Manager in the USB HS/FS Device Controller, this field must always be 0.
11-0	Packet Return Queue #	This field indicates the queue number within the selected queue manager that the descriptor is to be returned to after transmission is complete. This field is not altered by the DMA during transmission or reception and is initialized by the CPU.

Table 17-11. Host Packet Descriptor Word 3 (HPD Word 3)

Bits	Name	Description
31-22	Reserved	Reserved
21-0	Buffer 0 Length	The Buffer Length field indicates how many valid data bytes are in the buffer. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception.

Table 17-12. Host Packet Descriptor Word 4 (HPD Word 4)

Bits	Name	Description
31-0	Buffer 0 Pointer	The Buffer Pointer is the byte aligned memory address of the buffer associated with the buffer descriptor. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception.

Table 17-13. Host Packet Descriptor Word 5 (HPD Word 5)

Bits	Name	Description
31-0	Next Descriptor Pointer	The 32-bit word aligned memory address of the next buffer descriptor in the packet. If the value of this pointer is zero, then the current buffer is the last buffer in the packet. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception.

Table 17-14. Host Packet Descriptor Word 6 (HPD Word 6)

Bits	Name	Description
31-22	Reserved	Reserved
21-0	Original Buffer 0 Length	The Buffer Length field indicates the original size of the buffer in bytes. This value is not overwritten during reception. This value is read by the Rx DMA to determine the actual buffer size as allocated by the CPU at initialization. Since the buffer length in Word 3 is overwritten by the Rx port during reception, this field is necessary to permanently store the buffer size information.

Table 17-15. Host Packet Descriptor Word 7 (HPD Word 7)

Bits	Name	Description
31-22	Reserved	Reserved
21-0	Original Buffer 0 Pointer	The Buffer Pointer is the byte aligned memory address of the buffer associated with the buffer descriptor. This value is not overwritten during reception. This value is read by the Rx DMA to determine the actual buffer location as allocated by the CPU at initialization. Since the buffer pointer in Word 4 is overwritten by the Rx port during reception, this field is necessary to permanently store the buffer pointer information.

17.2.9.3 Host Buffer Descriptor (Non-SOP Descriptor)

The Host Buffer Descriptor is identical in size and organization to a Host Packet Descriptor but does not include valid information in the packet level fields and does not include a populated region for protocol specific information. The packet level fields is not needed since the SOP descriptor contain this information and additional copy of this data is not needed/necessary.

Host Buffer Descriptors are designed to be linked onto a Host Packet Descriptor or another Host Buffer Descriptor to provide support for unlimited scatter / gather type operations. Host Buffer Descriptors provide information about a single corresponding data buffer. Every Host buffer descriptor stores the following information:

- Pointer to the first valid byte in the data buffer
- Length of the data buffer
- Pointer to the next buffer descriptor in the packet

Host Buffer Descriptors always contain 32 bytes of required information. Since it is a requirement that it is possible to convert a Host descriptor between a Buffer Descriptor and a Packet Descriptor (by filling in the appropriate fields) in practice, Host Buffer Descriptors will be allocated using the same sizes as Host Packet Descriptors. In addition, since the 5 LSBs of the Descriptor Pointers are used in CPPI 4.1 for the purpose of indicating the length of the descriptor, the minimum size of a descriptor is always 32 bytes. (For more information on Descriptor Size, see Section 17.3.80).

NOTE: Descriptors can be located anywhere within the 16MB address space of the device (except for DARAM, which is not accessible by the USB controller). However, all descriptors must be placed in a single contiguous block of up to 64KW.

The descriptor layout is shown in Figure 17-13.

Figure 17-13. Host Buffer Descriptor Layout

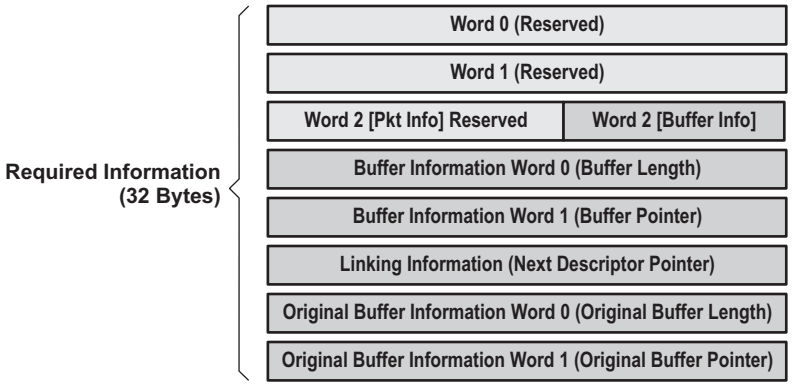


Table 17-16. Host Buffer Descriptor Word 0 (HBD Word 0)

Bits	Name	Description
31-0	Reserved	Reserved

Table 17-17. Host Buffer Descriptor Word 1 (HBD Word 1)

Bits	Name	Description
31-0	Reserved	Reserved

Table 17-18. Host Buffer Descriptor Word 2 (HBD Word 2)

Bits	Name	Description
31-15	Reserved	Reserved
14	On-chip	This field indicates whether or not this descriptor is in a region which is in on-chip memory space (1) or in external memory (0).
13-12	Packet Return Queue Mgr #	This field indicates which queue manager in the system the descriptor is to be returned to after transmission is complete. This field is not altered by the DMA during transmission or reception and is initialized by the CPU. There is only 1 Queue Manager in the USB HS/FS Device Controller, this field must always be 0.
11-0	Packet Return Queue #	This field indicates the queue number within the selected queue manager that the descriptor is to be returned to after transmission is complete. This field is not altered by the DMA during transmission or reception and is initialized by the CPU.

Table 17-19. Host Buffer Descriptor Word 3 (HBD Word 3)

Bits	Name	Description
31-22	Reserved	Reserved
21-0	Buffer 0 Length	The Buffer Length field indicates how many valid data bytes are in the buffer. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception.

Table 17-20. Host Buffer Descriptor Word 4 (HBD Word 4)

Bits	Name	Description
31-0	Buffer 0 Pointer	The Buffer Pointer is the byte aligned memory address of the buffer associated with the buffer descriptor. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception.

Table 17-21. Host Buffer Descriptor Word 5 (HBD Word 5)

Bits	Name	Description
31-0	Next Descriptor Pointer	The 32-bit word aligned memory address of the next buffer descriptor in the packet. If the value of this pointer is zero, then the current descriptor is the last descriptor in the packet. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception.

Table 17-22. Host Buffer Descriptor Word 6 (HBD Word 6)

Bits	Name	Description
31-22	Reserved	Reserved
21-0	Original Buffer 0 Length	The Buffer Length field indicates the original size of the buffer in bytes. This value is not overwritten during reception. This value is read by the Rx DMA to determine the actual buffer size as allocated by the CPU at initialization. Since the buffer length in Word 3 is overwritten by the Rx port during reception, this field is necessary to permanently store the buffer size information.

Table 17-23. Host Buffer Descriptor Word 7 (HBD Word 7)

Bits	Name	Description
31-0	Original Buffer 0 Pointer	The Buffer Pointer is the byte aligned memory address of the buffer associated with the buffer descriptor. This value is not overwritten during reception. This value is read by the Rx DMA to determine the actual buffer location as allocated by the CPU at initialization. Since the buffer pointer in Word 4 is overwritten by the Rx port during reception, this field is necessary to permanently store the buffer pointer information.

17.2.9.4 Teardown Descriptor

The Teardown Descriptor is not like the Host Packet or Buffer Descriptors since it is not used to describe either a packet or a buffer. The Teardown Descriptor is always 32 bytes long and is comprised of 4 bytes of actual teardown information and 28 bytes of pad (see Figure 17-14). Since the 5 LSBs of the Descriptor Pointers are used in CPPI 4.1 for the purpose of indicating the length of the descriptor, the minimum size of a descriptor is 32 bytes.

Teardown Descriptor is used to describe a channel halt and teardown event. Channel teardown ensures that when a connection is no longer needed that the hardware can be reliably halted and any remaining packets which had not yet been transmitted can be reclaimed by the Host without the possibility of losing buffer or descriptor references (which results in a memory leak).

NOTE: Descriptors can be located anywhere within the 16MB address space of the device (except for DARAM, which is not accessible by the USB controller). However, all descriptors must be placed in a single contiguous block of up to 64KW.

The Teardown Descriptor contains the following information:

- Indicator which identifies the descriptor as a Teardown Packet Descriptor
- DMA Controller Number where teardown occurred
- Channel number within DMA where teardown occurred
- Indicator of whether this teardown was for the Tx or Rx channel

Figure 17-14. Teardown Descriptor Layout

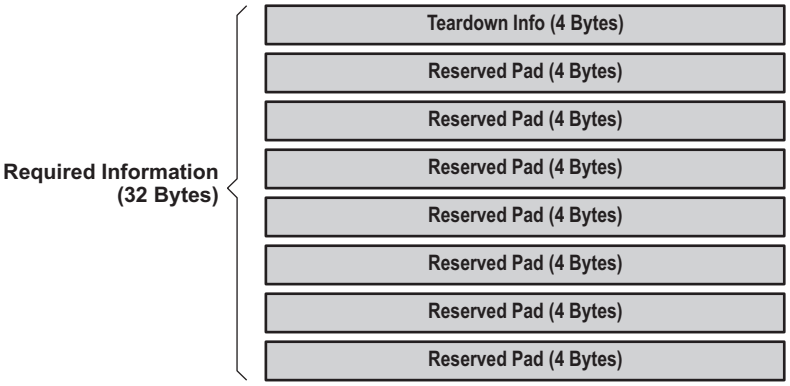


Table 17-24. Teardown Descriptor Word 0

Bits	Name	Description
31-27	Descriptor Type	The teardown descriptor type is 19 decimal (13h).
26-17	Reserved	Reserved
16	TX_RX	Indicates whether teardown is a TX (0) or RX (1).
15-10	DMA Number	Indicates the DMA number for this teardown.
9-6	Reserved	Reserved
5-0	Channel Number	Indicates the channel number within the DMA that was torn down.

Table 17-25. Teardown Descriptor Words 1-7

Bits	Name	Description
31-0	Reserved	Reserved

Teardown operation of an endpoint requires three operations. The teardown register in the CPPI DMA must be written, the corresponding endpoint bit in TEARDOWN of the USB module must be set, and the FlushFIFO bit in the Mentor USB controller TX/RXCSR register must be set. Writing to TEARDOWN in the USB2.0 module resets the CPPI FIFO occupancy value and pointers to 0. It also resets the current state of the XDMA for the endpoint selected, after any current operations have been completed. Note that due to VBUSP bridge latency, the CPPI FIFO occupancy values will not be reset immediately upon writing of TEARDOWN.

17.2.9.5 Queues

Several types of queues exist (a total of 64 queues) within the CPPI 4.1 DMA. Regardless of the type of queue a queue is, queues are used to hold pointers to host or buffer packet descriptors while they are being passed between the Host and / or any of the ports in the system. All queues are maintained within the Queue Manager module.

The following type of Queues exist:

- Receive Free Descriptor/Buffer Queue
- Receive Completion (Return) Queue
- Transmit Submit Queue (also referred as Transmit Queue)
- Transmit Completion (Return) Queue
- Free Descriptor Queue (Unassigned: Can be used for Completion or Application Specific purposes)

Table 17-26 displays the allocation (partition) of the available Queues.

Table 17-26. Allocation of Queues

Starting Queue Number	Number of Queues	Function
0	16	RX +Free Descriptor/Buffer (submit) queues
16	2	USB Endpoint 1 TX (submit) queues
18	2	USB Endpoint 2 TX (submit) queues
20	2	USB Endpoint 3 TX (submit) queues
22	2	USB Endpoint 4 TX (submit) queues
24	2	TX Completion (return) queues
26	2	RX Completion (return) queues
28	36	Unassigned (application-defined) queues

17.2.9.5.1 Queuing Packets

Prior to queuing packets, the host/firmware should construct data buffer as well host packet/buffer descriptors within the 16MB address space of the device (except for DARAM which is not accessible by the USB controller).

NOTE: Descriptors must be placed in a single contiguous block of up to 64KW anywhere in the 16MB address space of the device, except DARAM which is not accessible by the USB controller.

Queuing of packets onto a packet queue is accomplished by writing a pointer to the Packet Descriptor into a specific address within the selected queue (Register D of Queue N). Packet is always queued onto the tail of the queue. The Queue Manager provides a unique set of addresses for adding packets for each queue that it manages.

NOTE: The control register D for each queue is split up into two registers (CTRL1D and CTRL2D). To load a descriptor pointer into a queue, use a single double word write to CTRL1D.

17.2.9.5.2 De-Queuing Packets

De-queuing of packets from a packet queue is accomplished by reading the head packet pointer from a specific address within the selected queue (Register D of Queue N). After the head pointer has been read, the Queue Manager will invalidate the head pointer and will replace it with the next packet pointer in the queue. This functionality, which is implemented in the Queue Manager, prevents the ports from needing to traverse linked lists and allows for certain optimizations to be performed within the Queue Manager.

NOTE: The control register D for each queue is split up into two registers (CTRL1D and CTRL2D). To unload a descriptor pointer into a queue, use a single word read from CTRL1D. The return value will be the lower 16 bits of the descriptor address. Since all descriptors must be within a 64KW memory range, a read from CTRL2D is not necessary.

17.2.9.5.3 Type of Queues

Several types of queues exist and all are managed by the Queue Manager which is part of the CPPI 4.1 DMA. All accesses to the queues are through memory mapped registers and no external memory setup is required by the firmware.

17.2.9.5.3.1 Receive Free Descriptor/Buffer (Submit) Queue

Receive ports use queues referred to as "receive free descriptor / buffer queues" to forward completed receive packets to the host or another peer port entity. The entries on the Free Descriptor / Buffer Queues have pre-attached empty buffers whose size and location are described in the "original buffer information" fields in the descriptor. The host is required to allocate both the descriptor and buffer and pre-link them prior to adding (submitting) a descriptor to one of the available receive free descriptor / buffer queue. The first 16 queues (Queue 0 up to Queue 15) are reserved for all four receive ports to handle incoming packets.

17.2.9.5.3.2 Transmit (Submit) Queue

Transmit ports use packet queues referred to as "transmit (submit) queues" to store the packets that are waiting to be transmitted. Each port has dedicated queues (2 queues per port) that are reserved exclusively for a use by a single port. Multiple queues per port/channel are allocated to facilitate Quality of Service (QoS) for applications that require QoS. Queue 16 and 17 are allocated for port 1, Queue 18 and 19 are allocated for port 2 and Queue 20 and Queue 21 are allocated for port 3 and Queue 22 and 23 are allocated for port 4.

17.2.9.5.3.3 Transmit Completion Queue

Transmit ports also use packet queues referred to as "transmit completion queues" to return packets to the host after they have been transmitted. Even though, non-allocated queues can be used for this purpose, a total of two dedicated queues (Queue 24 and Queue 25), that is to be shared amongst all four transmit ports, have been reserved for returning transmit packets after end of transmit operation when the firmware desires to receive interrupt when transmission completes.

17.2.9.5.3.4 Receive Completion Queue

Receive ports also use packet queues referred to as "receive completion queues" to return packets to the port after they have been received. Even though, non-allocated queues can be used for this purpose, a total of two dedicated queues (Queue 26 and Queue 27), that is to be shared amongst all four transmit ports, have been reserved for returning received packets to the receive ports after end of receive operation when the firmware desires to receive interrupt when transmission completes.

17.2.9.5.3.5 Unassigned (Application Defined) Queue

Thirty-six additional queues (Queue 28 to Queue 63) exist that have not been dedicated for exclusive use. The user can use these queues as a Completion Queues or Free Descriptor/Buffer queue.

When these queues are used as Completion Queues, interrupt will not be generated. However, the queues will have the list of descriptor pointers for the packets that have completed transmission or reception. The firmware can use polling method by continually performing the de-queuing technique onto the particular unassigned queue used to identify if the reception or transmission has completed.

When unassigned queues are used as free descriptor/buffer queue, the user can use these queues to queue/store available descriptors for future receive and transmit operations by the firmware popping the respective assigned queue and retrieving and populating descriptor prior to submitting the updated descriptor.

17.2.9.5.3.6 Teardown Queue

The Teardown Queue is used by the DMA to communicate a completion of a channel teardown after a channel teardown is invoked on to a channel. The pointer to the teardown descriptor is written to the teardown queue, which is also the Completion Queue, when the channel teardown completes.

17.2.9.5.3.7 Diverting Queue Packets from one Queue to Another

The host can move the entire contents of one queue to another queue by writing the source queue number and the destination queue number to the Queue Diversion Register. When diverting packets, the host can choose whether the source queue contents should be pushed onto the tail of the destination queue.

17.2.9.6 Memory Regions and Linking RAM

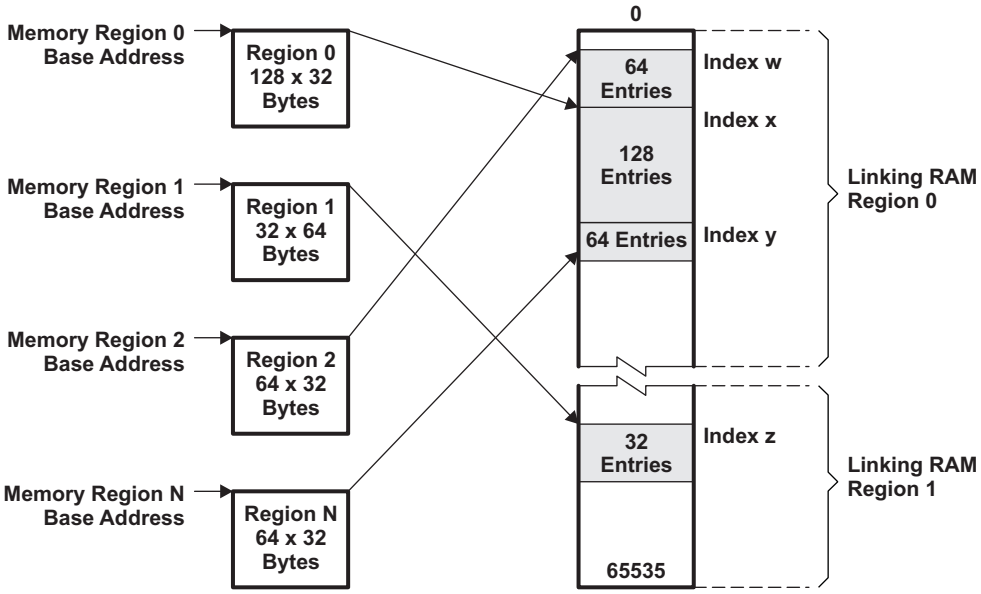
In addition to allocating memory for raw data, the host is responsible for allocating additional memory for exclusive use of the CPPI DMA as well as the Queue Manager. The Queue Manager has the capability of managing up to 16 Memory Regions. These Memory regions are used to allocate descriptors of variable sizes. The total number of descriptors that can be managed by the Queue Manager should not exceed 64K. Each memory region has descriptors of one configurable size. These 64K descriptors are referenced internally in the queue manager by a 16-bit quantity index.

The queue manager uses a linking RAM to store information (16-bit quantity index) about how the descriptors are logically connected to one another. A total of two Linking RAMs exists to be used by all Memory Regions. Each location in the linking RAM stores information for one descriptor index. The linking information for all descriptors in a given memory region is stored in a contiguous fashion in the linking RAM. The host, when it initializes the memory regions, also writes the index number corresponding to the first descriptor in a given region.

This information is used by the queue manager to compute where exactly in memory a particular descriptor is stored. The size of the linking RAM to be allotted by the host/firmware should be large enough to contain information for the amount of descriptor defined within the total Memory Regions. A total of 4 bytes of RAM is required for each descriptor. Figure 17-15 illustrates the relationship between memory regions and linking RAM.

NOTE: The reason for the existence of the two Linking RAMs is for the case when the user desires to allocate Linking RAMs in both internal and external memory. There is no restriction as to the placement of the Linking RAM.

Figure 17-15. Relationship Between Memory Regions and Linking RAM



17.2.9.7 Zero Length Packets

A special case is the handling of null packets with the CPPI 4.1 compliant DMA controller. Upon receiving a zero length USB packet, the XFER DMA will send a data block to the DMA controller with byte count of zero and the zero byte packet bit of INFO Word 2 set. The DMA controller will then perform normal End of Packet termination of the packet, without transferring data.

If a zero-length USB packet is received, the XDMA will send the CDMA a data block with a byte count of 0 and this bit set. The CDMA will then perform normal EOP termination of the packet without transferring data. For transmit, if a packet has this bit set, the XDMA will ignore the CPPI packet size and send a zero-length packet to the USB controller.

17.2.9.8 CPPI DMA Scheduler

The CPPI DMA scheduler is responsible for controlling the rate and order between the different Tx and Rx threads that are provided in the CPPI DMA controller. The scheduler table RAM exists within the scheduler.

17.2.9.8.1 CPPI DMA Scheduler Initialization

Before the scheduler can be used, the host is required to initialize and enable the block. This initialization is performed as follows:

1. The Host initializes entries within an internal memory array in the scheduler. This array contains up to 256 entries and each entry consists of a DMA channel number and a bit indicating if this is a Tx or Rx opportunity. These entries represent both the order and frequency that various Tx and Rx channels will be processed. A table size of 256 entries allows channel bandwidth to be allocated with a maximum precision of 1/256th of the total DMA bandwidth. The more entries that are present for a given channel, the bigger the slice of the bandwidth that channel will be given. Larger tables can be accommodated to allow for more precision. This array can only be written by the Host, it cannot be read.
2. If the application does not need to use the entire 256 entries, firmware can initialize the portion of the 256 entries and indicate the last entry used by writing to the LAST_ENTRY bits in the CDMA Scheduler Control Register 1 (DMA_SCHED_CTRL1) in the scheduler.
3. The host writes to the ENABLE bit in DMA_SCHED_CTRL1 to enable the scheduler. The scheduler is not required to be disabled in order to change the scheduler array contents.

17.2.9.8.2 Scheduler Operation

Once the scheduler is enabled it will begin processing the entries in the table and when appropriate passing credits to the DMA controller to perform a Tx or Rx operation. The operation of the DMA controller is as follows:

1. After the DMA scheduler is enabled it begins with the table index set to 0.
2. The scheduler reads the entry pointed to by the index and checks to see if the channel in question is currently in a state where a DMA operation can be accepted. The following must both be true:
   - The DMA channel must be enabled.
   - The CPPI FIFO that the channel talks to has free space on TX (FIFO full signal is not asserted) or a valid block on Rx (FIFO empty signal is not asserted).
3. If the DMA channel is capable of processing a credit to transfer a block, the DMA scheduler will issue that credit via the DMA scheduling interface. These are the steps:
   (a) The DMA controller may not be ready to accept the credit immediately and it may stall the scheduler until it can accept the credit. The DMA controller only accepts credits when it is in the IDLE state.
   (b) Once a credit has been accepted, the scheduler will increment the index to the next entry and will start at step 2.
4. If the channel in question is not currently capable of processing a credit, the scheduler will increment the index in the scheduler table to the next entry and will start at step 2.
5. When the scheduler attempts to increment its index to the value programmed in the table size register, the index will reset to 0.

17.2.9.9 CPPI DMA Transfer Interrupt Handling

The CPPI DMA 4.1 Interrupt handling mechanism does not go through the PDR Interrupt handler built into the core. The DMA interrupt line is directly routed to the Interrupt Dispatcher in a PDR compliant manner. The DMA interrupt is not maskable. The firmware needs to use queues not reserved by H/W as Completion Queues if require for DMA interrupt to be generated on a completion of a transfer.

Queues 24 and 25 are reserved by H/W for DMA transmit operations and queues 26 and 27 are reserved by H/W for DMA receive operations. If firmware uses these queues as completion queues, interrupt will be generated when the transfer completes. If need not to generate an interrupt, firmware is required to use queues that are not reserved as completion queues (queues 28 to 67).

17.2.9.10 DMA State Registers

The port must store and maintain state information for each transmit and receive port/channel. The state information is referred to as the Tx DMA State and Rx DMA State.

17.2.9.10.1 Transmit DMA State Registers

The Tx DMA State is a combination of control fields and protocol specific port scratchpad space used to manipulate data structures and transmit packets. Each transmit channel has two queues. Each queue has a one head descriptor pointer and one completion pointer. There are four Tx DMA State registers; one for each port/channel.

The following information is stored in the Tx DMA State:

- Tx Queue Head Descriptor Pointer(s)
- Tx Completion Pointer(s)
- Protocol specific control/status (port scratchpad)

17.2.9.10.2 Receive DMA State Registers

The Rx DMA State is a combination of control fields and protocol specific port scratchpad space used to manipulate data structures in order to receive packets. Each receive channel has only one queue. Each channel queue has one head descriptor pointer and one completion pointer. There are four Rx DMA State registers; one for each port/channel.

The following information is stored in the Rx DMA State:

- Rx Queue Head Descriptor Pointer
- Rx Queue Completion Pointer
- Rx Buffer Offset

17.2.9.11 USB DMA Protocols Supported

Four different type of DMA transfers are supported by the CPPI 4.1 DMA; Transparent, RNDIS, Generic RNDIS, and Linux CDC. The following sections will outline the details on these DMA transfer types.

17.2.9.11.1 Transparent DMA

Transparent Mode DMA operation is the default DMA mode where DMA interrupt is generated whenever a DMA packet is transferred. In the transparent mode, DMA packet size cannot be greater than USB MaxPktSize for the endpoint. This transfer type is ideal for transfer (not packet) sizes that are less than a max packet size.

Transparent DMA Transfer Setup

The following will configure all four ports/channels for Transparent DMA Transfer type.

- Make sure that RNDIS Mode is disabled globally. RNDIS bit in the control register (CTRLR) is cleared to 0.
- Configure the endpoint mode control fields in the DMA Mode Registers (MODE1 and MODE2) for Transparent Mode (RX*n*_MODE and TX*n*_MODE = 0h).

17.2.9.11.2 RNDIS

RNDIS mode DMA is used for large transfers (that is, total data size to be transferred is greater than USB MaxPktSize where the MzxPktSize is a multiple of 64 bytes) that requires multiple USB packets. This is accomplished by breaking the larger packet into smaller packets, where each packet size being USB MaxPktSize except the last packet where its size is less than USB MaxPktSize, including zero bytes. This implies that multiple USB packets of MaxPktSize will be received and transferred together as a single large DMA transfer and the DMA interrupt is generated only at the end of the complete reception of DMA transfer. The protocol defines the end of the complete transfer by receiving a short USB packet (smaller than USB MaxPktSize as mentioned in USB specification 2.0). If the DMA packet size is an exact multiple of USB MaxPktSize, the DMA controller waits for a zero byte packet at the end of complete transfer to signify the completion of the transfer.

NOTE: RNDIS Mode DMA is supported only when USB MaxPktSize is an integral multiple of 64 bytes.

RNDIS DMA Transfer Setup

The following will configure all four ports/channels for RNDIS DMA Transfer type. If all endpoints are to be configured with the same RNDIS DMA transfer type, then you can enable for RNDIS mode support from the Control Register and the content of the Mode Register will be ignored.

If you need to enable RNDIS support globally.

- Enable RNDIS Mode globally. RNDIS bit in the control register (CTRLR) is set to 1.

If you need to enable RNDIS support at the port/channel (endpoint) level.

- Disable RNDIS Mode globally. RNDIS bit in the control register (CTRLR) is cleared to 0.
- Configure the endpoint mode control fields in the DMA Mode Registers (MODE1 and MODE2) for RNDIS Mode (RX*n*_MODE and TX*n*_MODE = 1h).

The above two setups yield the same result.

17.2.9.11.3 Generic RNDIS

Generic RNDIS DMA transfer mode is identical to the normal RNDIS mode in nearly all respects, except for the exception case where the last packet of the transfer can either be a short packet or the MaxPktSize. Generic RNDIS transfer makes use of a RNDIS EP Size register (there exists a register for each endpoint) that must be programmed with a value that is an integer multiple of the endpoint size for the DMA to know the end of the transfer when the last packet size is equal to the USB MaxPktSize. For example, it the Tx/RxMaxP is programmed with a value of 64, the Generic RNDIS EP Size register for that endpoint must be programmed with a value that is an integer multiple of 64 (for example, 64, 128, 192, 256, etc.).

In other words, when using Generic RNDIS mode and the DMA is tasked to transfer data transfer size that is less than a value programmed within the RNDIS EP Size register and this transfer will be resulting with a short packet, the DMA will terminate the transfer when encountering the short packet behaving exactly as the RNDIS DMA transfer type.

This means that Generic RNDIS mode will perform data transfer in the same manner as RNDIS mode, closing the CPPI packet when a USB packet is received that is less than the USB MaxPktSize size. Otherwise, the packet will be closed when the value in the Generic RNDIS EP Size register is reached.

Using RNDIS EP Size register, a packet of up to 64K bytes can be transferred. This is to allow the host software to program the USB module to transfer data that is an exact multiple of the USB MaxPktSize (Tx/RxMaxP programmed value) without having to send an additional short packet to terminate.

NOTE: As in RNDIS mode, the USB max packet size of any Generic RNDIS mode enabled endpoints must be a multiple of 64 bytes. Generic RNDIS acceleration should not be enabled for endpoints where the max packet size is not a multiple of 64 bytes. Only transparent mode should be used for such endpoints.

Generic RNDIS DMA Transfer Setup

The following will configure all four ports/channels for Generic RNDIS DMA Transfer type.

- Disable RNDIS Mode globally. RNDIS bit in the control register (CTRLR) is cleared to 0.
- Configure the endpoint mode control fields in the DMA Mode Registers (MODE1 and MODE2) for Generic RNDIS Mode (RX*n*_MODE and TX*n*_MODE = 3h).

17.2.9.11.4 Linux CDC

Linux CDC DMA transfer mode acts in the same manner as RNDIS packets, except for the case where the last data matches the max USB packet size. If the last data packet of a transfer is a short packet where the data size is greater than zero and less the USB MaxPktSize, then the behavior of the Linux CDC DMA transfer type is identical with the RNDIS DMA transfer type. The only exception is when the short packet length terminating the transfer is a Null Packet. In this case, instead of transferring the Null Packet, it will transfer a data packet of size 1 byte with the data value of 0h.

In transmit operation, if an endpoint is configured or CDC Linux mode, upon receiving a Null Packet from the CPPI DMA, the XFER DMA will then generate a packet containing 1 byte of data, whose value is 0h, indicating the end of the transfer. During receive operation, the XFER DMA will recognize the one byte zero packet as a termination of the data transfer, and sends a block of data with the EOP indicator set and a byte count of one to the CPPI DMA controller. The CPPI DMA realizing the end of the transfer termination will not update/increase the packet size count of the Host Packet Descriptor.

Linux CDC DMA Transfer Setup

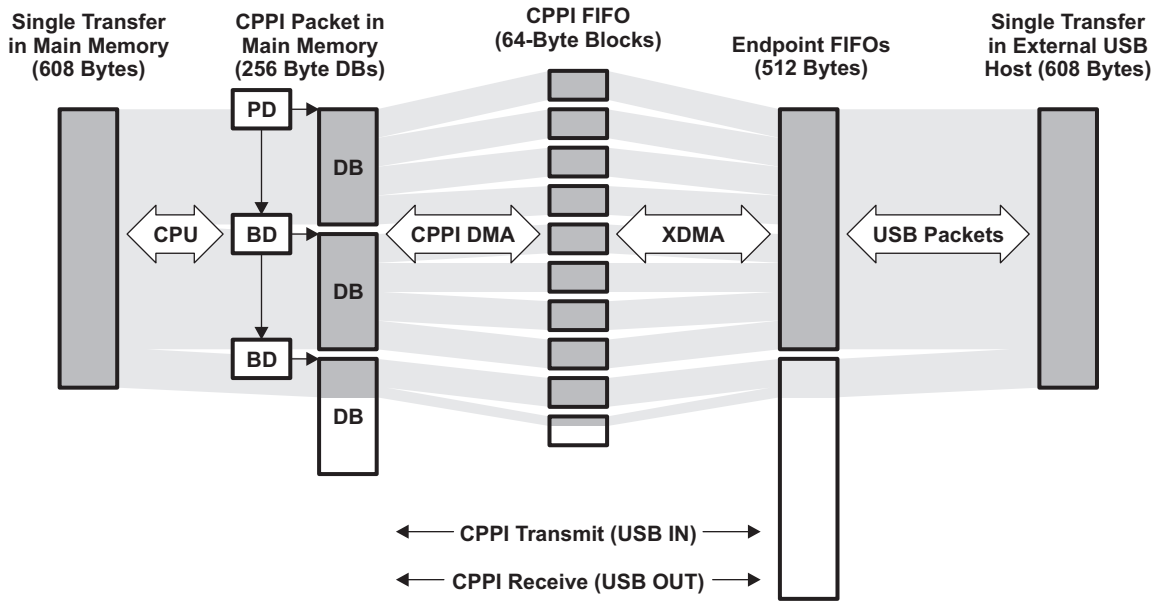
The following will configure all four ports/channels for Linux CDC DMA Transfer type.

- Disable RNDIS Mode globally. RNDIS bit in the control register (CTRLR) is cleared to 0.
- Configure the endpoint mode control fields in the DMA Mode Registers (MODE1 and MODE2) for Linux CDC Mode (RX*n*_MODE and TX*n*_MODE = 2h).

17.2.9.12 USB Data Flow Using DMA

The necessary steps required to perform a USB data transfer using the CPPI 4.1 DMA is expressed using an example for both transmit and receive cases. Assume a device is ready to perform a data transfer of size 608 bytes (see Figure 17-16).

Figure 17-16. High-Level Transmit and Receive Data Transfer Example



Example assumptions:

- The CPPI data buffers are 256 bytes in length.
- The USB endpoint 1 Tx and Rx endpoint 1 size are 512 bytes.
- A single transfer length is 608 bytes.
- The SOP offset is 0.

This translates to the following:

- Transmit Case:
  - 1 Host Packet Descriptor with Packet Length field of 608 bytes and a Data Buffer of size 256 Bytes linked to the 1st Host Buffer Descriptor.
  - First Host Buffer Descriptor with a Data Buffer size of 256 Bytes linked to the 2nd Buffer Descriptor.
  - Second Host Buffer Descriptor with a Data Buffer size of 96 bytes (can be greater, the Packet Descriptor contain the size of the packet) linked with its link word set to Null.
- Receive Case:
  - Two Host Buffer Descriptors with 256 bytes of Data Buffer Size
  - One Host Buffer Descriptor with 96 bytes (can be greater) of Data Buffer size

Within the rest of this section, the following nomenclature is used.

BD— Host Buffer Descriptor

DB— Data Buffer Size of 256 Bytes

PBD— Pointer to Host Buffer Descriptor

PD— Host Packet Descriptor

PPD— Pointer to Host Packet Descriptor

RXCQ— Receive Completion Queue or Receive Return Queue (for all Rx EPs, use 26 or 27)

RXSQ— Receive Free Packet/Buffer Descriptor Queue or Receive Submit Queue. (for all Rx EPs, use 0 to 15)

TXCQ— Transmit Completion Queue or Transmit Return Queue (for all Tx EPs, use 24 or 25)

TXSQ— Transmit Queue or Transmit Submit Queue (for EP1, use 16 or 17)

17.2.9.12.1 Transmit USB Data Flow Using DMA

The transmit descriptors and queue status configuration prior to the transfer taking place is shown in Figure 17-17. An example of initialization for a transmit USB data flow is shown in Figure 17-18.

Figure 17-17. Transmit Descriptors and Queue Status Configuration

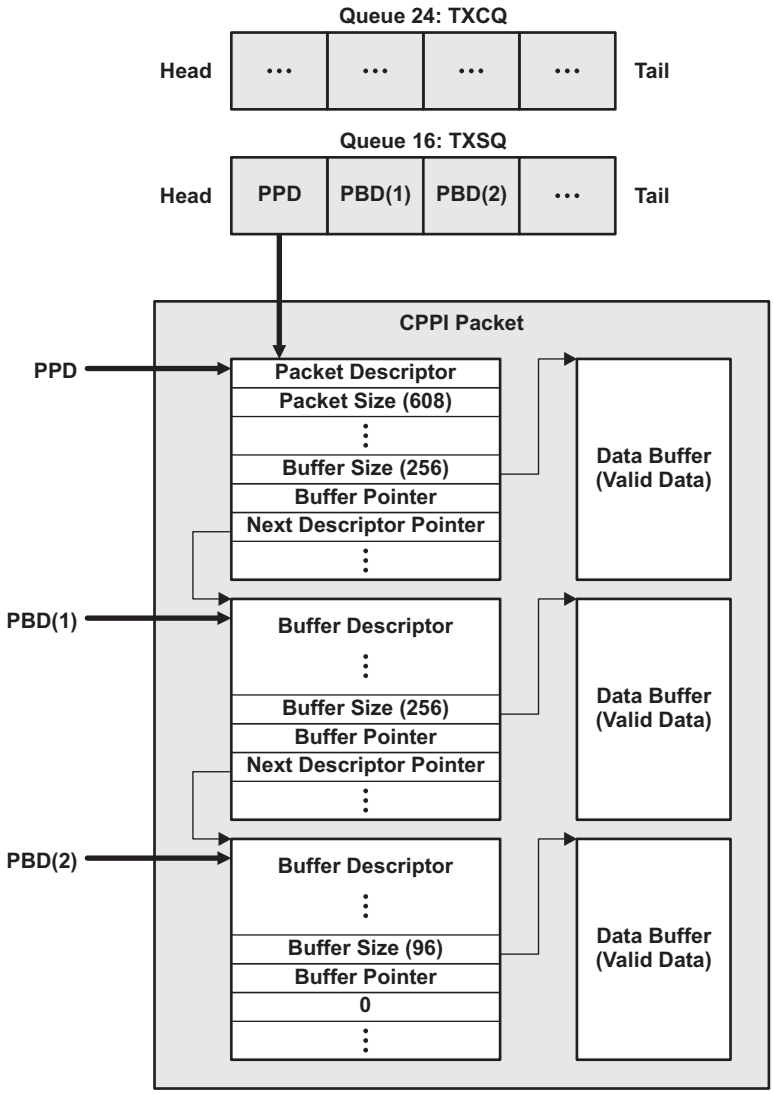
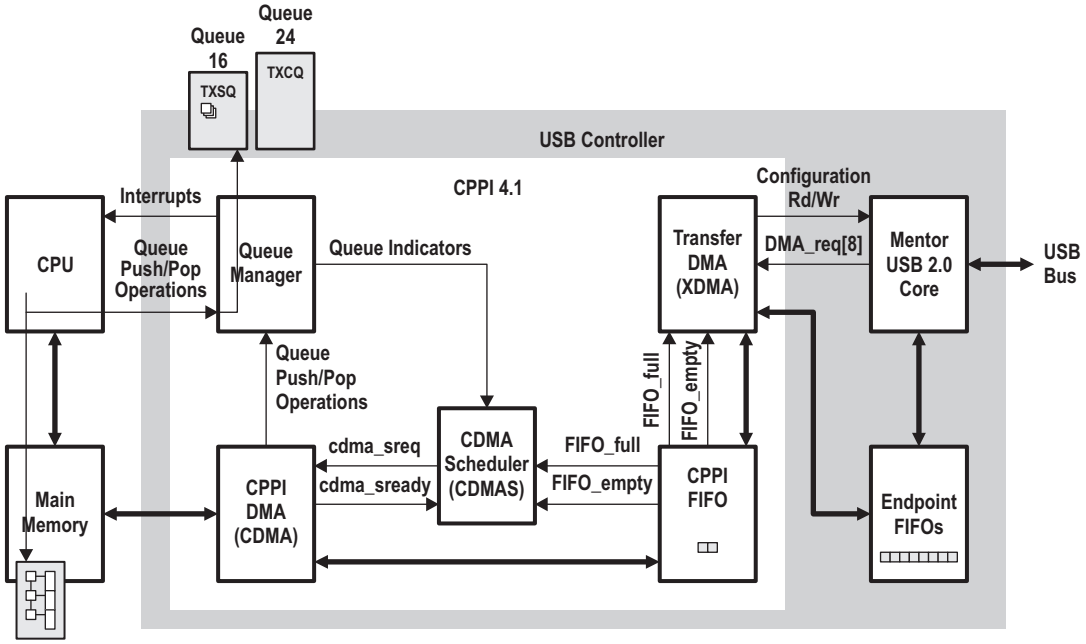


Figure 17-18. Transmit USB Data Flow Example (Initialization)



Step 1 (Initialization for Tx):

1. The CPU initializes Queue Manager with the Memory Region 0 base address and Memory Region 0 size, Link RAM0 Base address, Link RAM0 data size, and Link RAM1 Base address.
2. The CPU creates PD, BDs, and DBs in main memory and link as indicated in Figure 17-18.
3. It then initializes and configures the Queue Manager, Channel Setup, DMA Scheduler, and Mentor USB 2.0 Core.
4. It then adds (pushes) the PPD and the two PBDs to the TXSQ

NOTE: You can create more BD/DB pairs and push them on one of the unassigned queues. The firmware can pop a BD/DP pair from this chosen queue and can create its HPD or HBDs and pre link them prior to submitting the pointers to the HPD and HBD on to the TXSQ.

Step 2 (CDMA and XDMA transfers packet data into Endpoint FIFO for Tx):

1. The Queue Manager informs the CDMAS that the TXSQ is not empty.
2. CDMAS checks that the CPPI FIFO FIFO_full is not asserted, then issues a credit to the CDMA.
3. CDMA reads the packet descriptor pointer and descriptor size hint from the Queue Manager.
4. CMDA reads the packet descriptor from memory.
5. For each 64-byte block of data in the packet data payload:
   (a) The CDMA transfers a max burst of 64-byte block from the data to be transferred in main memory to the CPPI FIFO.
   (b) The XDMA sees FIFO_empty not asserted and transfers 64-byte block from CPPI FIFO to Endpoint FIFO.
   (c) The CDMA performs the above 2 steps 3 more times since the data size of the HPD is 256 bytes.
6. The CDMA reads the first buffer descriptor pointer.
7. CDMA reads the buffer descriptor from memory.

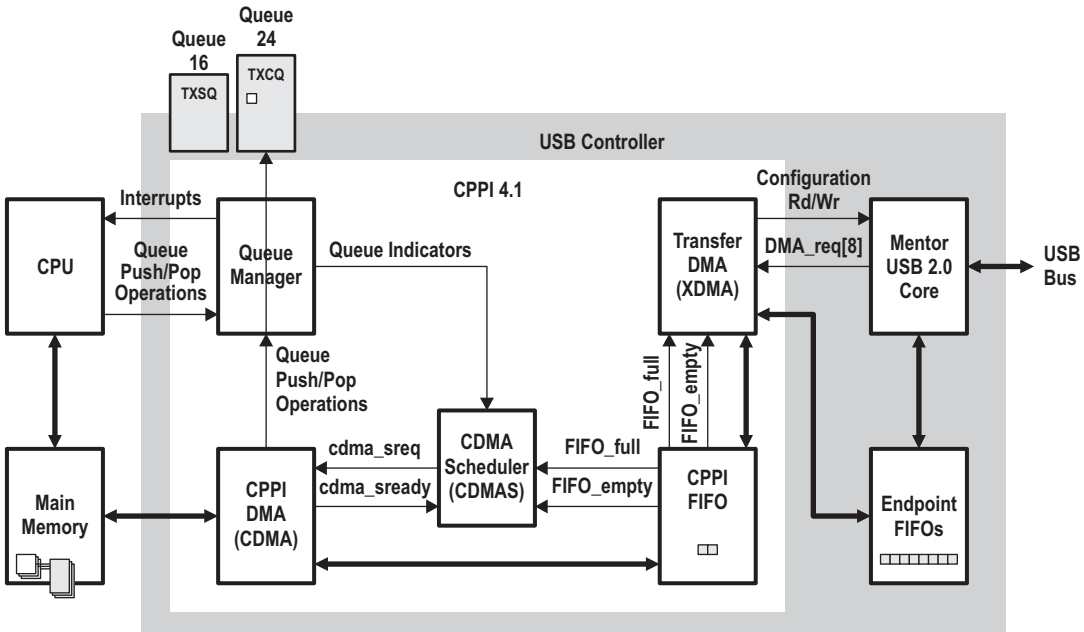
8. For each 64-byte block of data in the packet data payload:
   (a) The CDMA transfers a max burst of 64-byte block from the data to be transferred in main memory to the CPPI FIFO.
   (b) The XDMA sees FIFO_empty not asserted and transfers 64-byte block from CPPI FIFO to Endpoint FIFO.
   (c) The CDMA performs the above 2 steps 2 more times since data size of the HBD is 256 bytes.
9. The CDMA reads the second buffer descriptor pointer.
10. CDMA reads the buffer descriptor from memory.
11. For each 64-byte block of data in the packet data payload:
   (a) The CDMA transfers a max burst of 64-byte block from the data to be transferred in main memory to the CPPI FIFO.
   (b) The XDMA sees FIFO_empty not asserted and transfers 64-byte block from CPPI FIFO to Endpoint FIFO.
   (c) The CDMA transfers the last remaining 32-byte from the data to be transferred in main memory to the CPPI FIFO.
   (d) The XDMA sees FIFO_empty not asserted and transfers 32-byte block from CPPI FIFO to Endpoint FIFO.

Step 3 (Mentor USB 2.0 Core transmits USB packets for Tx):

1. Once the XDMA has transferred enough 64-byte blocks of data from the CPPI FIFO to fill the Endpoint FIFO, it signals the Mentor USB 2.0 Core that a TX packet is ready (sets the endpoint's TxPktRdy bit).
2. The Mentor USB 2.0 Core will transmit the packet from the Endpoint FIFO out on the USB BUS when it receives a corresponding IN request from the attached USB Host.
3. After the USB packet is transferred, the Mentor USB 2.0 Core issues a TX DMA_req to the XDMA.
4. This process is repeated until the entire packet has been transmitted. The XDMA will also generate the required termination packet depending on the termination mode configured for the endpoint.

An example of the completion for a transmit USB data flow is shown in Figure 17-19.

Figure 17-19. Transmit USB Data Flow Example (Completion)



Step 4 (Return packet to completion queue and interrupt CPU for Tx):

1. After all data for the packet has been transmitted (as specified by the packet size field), the CDMA will write the pointer to the packet descriptor to the TX Completion Queue specified in the return queue manager / queue number fields of the packet descriptor.
2. The Queue Manager then indicates the status of the TXSQ (empty) to the CDMAS and the TXCQ to the CPU via an interrupt.

17.2.9.12.2 Receive USB Data Flow Using DMA

The receive descriptors and queue status configuration prior to the transfer taking place is shown in Figure 17-20. An example of initialization for a receive USB data flow is shown in Figure 17-21.

Figure 17-20. Receive Descriptors and Queue Status Configuration

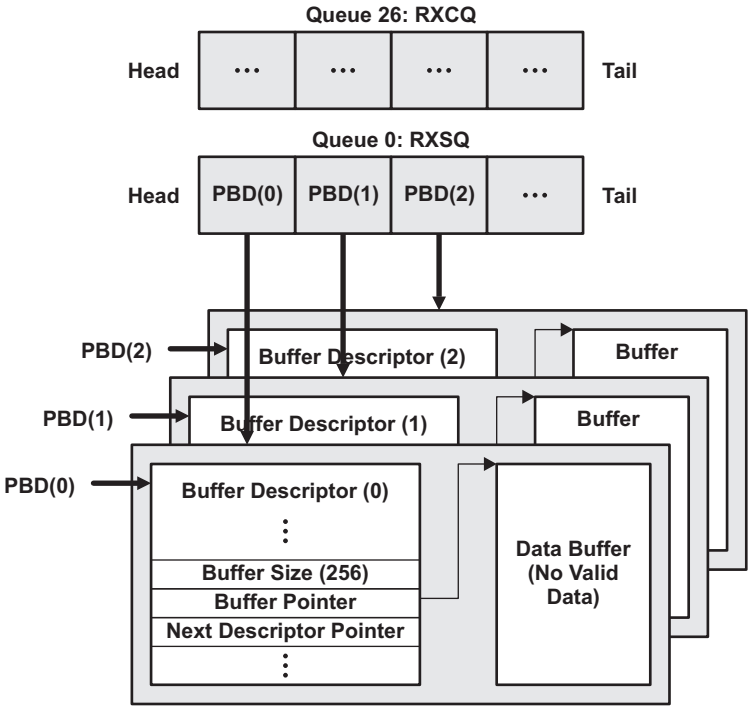
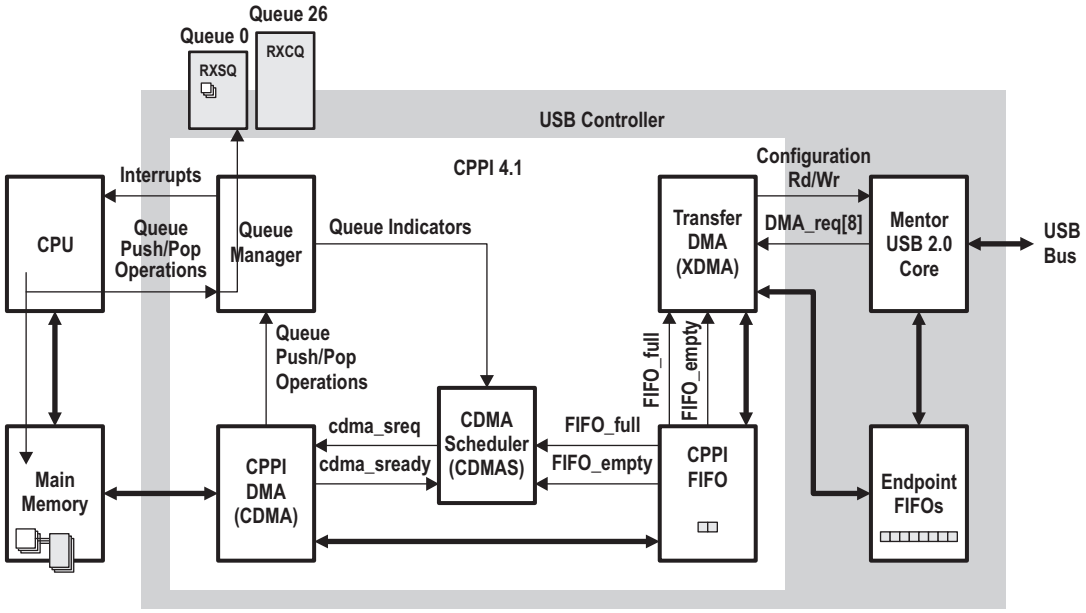


Figure 17-21. Receive USB Data Flow Example (Initialization)



Step 1 (Initialization for Rx):

1. The CPU initializes Queue Manager with the Memory Region 0 base address and Memory Region 0 size, Link RAM0 Base address, Link RAM0 data size, and Link RAM1 Base address.
2. The CPU creates BDs, and DBs in main memory and link them as indicated in Figure 17-21.
3. It then initializes the RXCQ queue and configures the Queue Manager, Channel Setup, DMA Scheduler, and Mentor USB 2.0 Core.
4. It then adds (pushes) the address of the three PHDs into the RXSQ.

Step 2 (Mentor USB 2.0 Core receives a packet, XDMA starts data transfer for Receive):

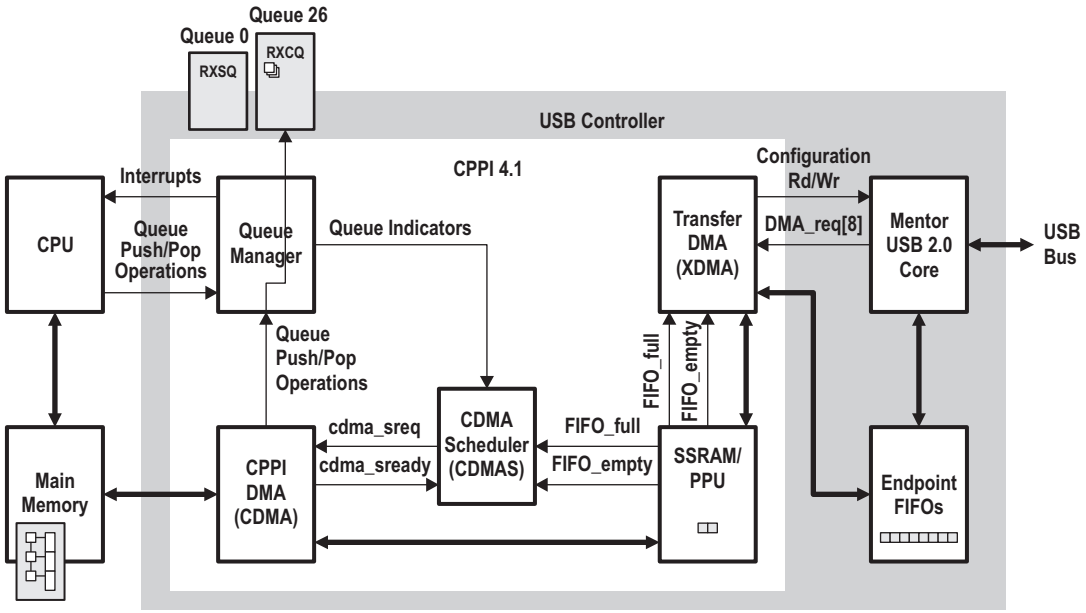
1. The Mentor USB 2.0 Core receives a USB packet from the USB Host and stores it in the Endpoint FIFO.
2. It then asserts a DMA_req to the XDMA informing it that data is available in the Endpoint FIFO.
3. The XDMA verifies the corresponding CPPI FIFO is not full via the FIFO_full signal, then starts transferring 64-byte data blocks from the Endpoint FIFO into the CPPI FIFO.

Step 3 (CDMA transfers data from SSRAM / PPU to main memory for Receive):

1. The CDMAS see FIFO_empty de-asserted (there is RX data in the FIFO) and issues a transaction credit to the CDMA.
2. The CDMA begins packet reception by fetching the first PBD from the Queue Manager using the Free Descriptor / Buffer Queue 0 (Rx Submit Queue) index that was initialized in the RX port DMA state for that channel.
3. The CDMA will then begin writing the 64-byte block of packet data into this DB.
4. The CDMA will continue filling the buffer with additional 64-byte blocks of data from the CPPI FIFO and will fetch additional PBD as needed using the Free Descriptor / Buffer Queue 1, 2, and 3 indexes for the 2nd, 3rd, and remaining buffers in the packet. After each buffer is filled, the CDMA writes the buffer descriptor to main memory.

An example of the completion for a receive USB data flow is shown in Figure 17-22 .

Figure 17-22. Receive USB Data Flow Example (Completion)



Step 4 (CDMA completes the packet transfer for Receive):

1. After the entire packet has been received, the CDMA writes the packet descriptor to main memory.
2. The CDMA then writes the packet descriptor to the RXCQ specified in the Queue Manager / Queue Number fields in the RX Global Configuration Register.
3. The Queue Manager then indicates the status of the RXCQ to the CPU via an interrupt.
4. The CPU can then process the received packet by popping the received packet information from the RXCQ and accessing the packet's data from main memory.

17.2.9.13 Interrupt Handling

Table 17-27 lists the interrupts generated by the USB controller.

Table 17-27. Interrupts Generated by the USB Controller

Interrupt	Description
Tx Endpoint [4-0]	Tx endpoint ready or error condition. For endpoints 4 to 0. (Rx and Tx for endpoint 0)
Rx Endpoint [4-1]	Rx endpoint ready or error condition. For endpoints 4 to 1. (Endpoint 0 has interrupt status in Tx interrupt)
USB Core[3-0]	Interrupts for 4 USB conditions
DMA Tx Completion [3-0]	Tx DMA completion interrupt for channel 3 to 0 using Queues 24 and 25
DMA Rx Completion [3-0]	Rx DMA completion interrupt for channel 3 to 0 using Queues 26 and 27

Whenever any of these interrupt conditions are generated, the host processor is interrupted. The software needs to read the different interrupt status registers (discussed in later section) to determine the source of the interrupt.

The USB interrupt conditions are listed in Table 17-28.

Table 17-28. USB Interrupt Conditions

Interrupt	Description
USB[3]	SOF started
USB[2]	Reset Signaling detected
USB[1]	Resume signaling detected
USB[0]	Suspend Signaling detected

17.2.9.13.1 USB Core Interrupts

Interrupt status can be determined using the INTSRCR (interrupt source) registers. These registers are non-masked. To clear the interrupt source, set the corresponding interrupt bit in INTCLRR registers. For debugging purposes, interrupt can be set manually through INTSETR registers.

The interrupt controller provides the option of masking the interrupts. A mask can be set using INTMSKSETR registers and can be cleared by setting the corresponding bit in the INTMSKCLRR registers. The mask can be read from INTMSKR registers. The masked interrupt status is determined using the INTMASKEDR registers.

The host processor software should write to the End Of Interrupt Register (EOIR) to acknowledge the completion of an interrupt.

NOTE: While EOIR is not written, the interrupt from the USB controller remains asserted.

17.2.10 BYTEMODE Bits of the USB System Control Register

The CPU cannot generate 8-bit accesses to its data or I/O space. This presents a problem given that some USB controller I/O registers are only 8 bits in width.

For these situations, the BYTEMODE bits of the USB system control register (USBSCR) can be used to program the DSP switched central resource (SCR) such that a CPU word access generates single byte access when reading or writing from USB controller I/O registers.

Table 17-29 summarizes the effect of the BYTEMODE bits for different CPU operations. For more details on USBSCR, please refer to Section 17.2.6.1.

Table 17-29. Effect of USBSCR BYTEMODE Bits on USB Access

BYTEMODE Setting	CPU Access to USB Register
BYTEMODE = 0h (16-bit word access)	Entire register contents are accessed.
BYTEMODE = 1h (8-bit access with high byte selected)	Only the upper byte of the register is accessed.
BYTEMODE = 2h (8-bit access with low byte selected)	Only the lower byte of the register is accessed.

17.2.11 Reset Considerations

The USB controller has two reset sources: hardware reset and the soft reset.

17.2.11.1 Software Reset Considerations

The USB controller can be reset by software through the RESET bit in the control register (CTRLR) or through the USB_RST bit in the peripheral reset control register (PCR).

When the RESET bit in the control register (CTRLR) is set, all the USB controller registers and DMA operations are reset. The bit is cleared automatically.

When USB_RST is set to 1, a hardware reset is forced on the USB controller. The effects of a hardware reset are described in the next section. Please note that the USB input clock must be enabled when using USB_RST (see Section 17.2.1).

17.2.11.2 Hardware Reset Considerations

A hardware reset is always initiated during a full chip reset. Alternatively, software can force an USB controller hardware reset through the USB_RST bits of the peripheral reset control register (PRCR). For more details on PRCR, see Section 1.1, System Control .

When a hardware reset is asserted, all the registers are set to their default values.

17.2.12 Interrupt Support

The USB controller is capable of interrupting the CPU. For more information on the mapping of interrupts, see Section 1.1, System Control .

17.2.13 DMA Event Support

The USB is an internal bus master peripheral and does not utilize system DMA events. The USB has its own dedicated DMA, CPPI 4.1 DMA, that it utilizes for DMA driven data transfer.

17.2.14 Power Management

The USB controller can be clock gated to conserve power during periods of no activity. The clock gating the peripheral is controlled by the CPU. For detailed information on power management procedures, see Section 1.1, System Control .

One of the main improvements of the device is that a dedicated LDO for USB has been integrated on the chip. This simplifies the power supply design on the system level. In order to use the internal USB LDO, the pin connections in Table 17-30 are required. The external LDO can also be used; the pin connections for using external LDO are included in Table 17-30 as well.

Table 17-30. LDO Pin Connections

Pin Number	Pin Name	Internal LDO	External LDO
C12	RSV0	L	L
D14	RSV3	L	L
D12	DSP_LDO_EN	L	H
D13	RSV16	L	L

Notes:

- *L* in the above table means voltage is at ground-level
- *H* in the above table means voltage is at the corresponding power supply level
- To use the internal USB LDO, the bit0 of register LDOCNTL (LDO Control Register) at 7004h need to be set as “1”
- Internal USB LDO is dedicated for USB module only; it should not be used for any other modules or external devices
- If an application requires the device to boot from an external USB device, then the external LDO must be used to power the USB module

17.3 Registers

17.3.1 USB Controller Register Summary

The following sections summarize the registers for the universal serial bus (USB) controller. Please note that the USB controller includes an USB2.0 mentor core and a communication part programming interface (CPPI) DMA, each with its own set of registers.

Due to the fact that the CDMA (CPPI DMA) is a native 32-bit module, and the device is a 16-bit DSP, it cannot do 32-bit read/write in one operation cycle. So, some cautions are needed when the device is accessing this module.

- When the device reads a 32-bit self-clean register, only the first read returns a valid 16-bit data. This is because when a reading action is applied to a 32-bit self-clean register; all 32 bit of the register is cleaned after the current read.
- When one needs to write a 32-bit data to a pair of 16-bit registers, which are split from a native 32-bit register, these two 16-bit write operations need to be consecutive and all the interrupts need to be disabled during these two write operations.

17.3.1.1 Universal Serial Bus (USB) Controller Registers

Table 17-31 lists the registers of the USB controller. Refer to the sections listed for detailed information on each register.

NOTE: Some USB controller registers are 8-bits wide. However, the CPU cannot generate 8-bit accesses to its data or I/O space. When accessing these registers, program the BYTEMODE bits of the USB system control register (USBSCR) to mask the upper or lower byte of of a word access. The BYTEMODE bits should be set to 00b (16-bit access) when accessing any other register. See Section 17.2.10 for more details on the BYTEMODE bits.

Table 17-31. Universal Serial Bus (USB) Registers

CPU Word Address	Acronym	Register Description	Section
8000h	REVID1	Revision Identification Register 1	Section 17.3.2
8001h	REVID2	Revision Identification Register 2	Section 17.3.2
8004h	CTRLR	Control Register	Section 17.3.3
800Ch	EMUR	Emulation Register	Section 17.3.4
8010h	MODE1	Mode Register 1	Section 17.3.5
8011h	MODE2	Mode Register 2	Section 17.3.5
8014h	AUTOREQ	Auto Request Register	Section 17.3.6
801Ch	TEARDOWN1	Teardown Register 1	Section 17.3.7
801Dh	TEARDOWN2	Teardown Register 2	Section 17.3.7
8020h	INTSRCR1	USB Interrupt Source Register 1	Section 17.3.8
8021h	INTSRCR2	USB Interrupt Source Register 2	Section 17.3.8
8024h	INTSETR1	USB Interrupt Source Set Register 1	Section 17.3.9
8025h	INTSETR2	USB Interrupt Source Set Register 2	Section 17.3.9
8028h	INTCLRR1	USB Interrupt Source Clear Register 1	Section 17.3.10
8029h	INTCLRR2	USB Interrupt Source Clear Register 2	Section 17.3.10
802Ch	INTMSKR1	USB Interrupt Mask Register 1	Section 17.3.11
802Dh	INTMSKR2	USB Interrupt Mask Register 2	Section 17.3.11
8030h	INTMSKSETR1	USB Interrupt Mask Set Register 1	Section 17.3.12
8031h	INTMSKSETR2	USB Interrupt Mask Set Register 2	Section 17.3.12
8034h	INTMSKCLRR1	USB Interrupt Mask Clear Register 1	Section 17.3.13
8035h	INTMSKCLRR2	USB Interrupt Mask Clear Register 2	Section 17.3.13
8038h	INTMASKEDR1	USB Interrupt Source Masked Register 1	Section 17.3.14

Table 17-31. Universal Serial Bus (USB) Registers (continued)

CPU Word Address	Acronym	Register Description	Section
8039h	INTMASKEDR2	USB Interrupt Source Masked Register 2	Section 17.3.14
803Ch	EOIR	USB End of Interrupt Register	Section 17.3.15
8040h	INTVECTR1	USB Interrupt Vector Register 1	Section 17.3.16
8041h	INTVECTR2	USB Interrupt Vector Register 2	Section 17.3.16
8050h	GREP1SZR1	Generic RNDIS EP1Size Register 1	Section 17.3.17
8051h	GREP1SZR2	Generic RNDIS EP1Size Register 2	Section 17.3.17
8054h	GREP2SZR1	Generic RNDIS EP2 Size Register 1	Section 17.3.18
8055h	GREP2SZR2	Generic RNDIS EP2 Size Register 2	Section 17.3.18
8058h	GREP3SZR1	Generic RNDIS EP3 Size Register 1	Section 17.3.19
8059h	GREP3SZR2	Generic RNDIS EP3 Size Register 2	Section 17.3.19
805Ch	GREP4SZR1	Generic RNDIS EP4 Size Register 1	Section 17.3.20
805Dh	GREP4SZR2	Generic RNDIS EP4 Size Register 2	Section 17.3.20

17.3.1.2 Mentor USB2.0 Core Registers

This section lists the registers of the Mentor USB2.0 core integrated in the USB controller.

NOTE: Some USB controller registers are 8-bits wide. However, the CPU cannot generate 8-bit accesses to its data or I/O space. When accessing these registers, program the BYTEMODE bits of the USB system control register (USBSCR) to mask the upper or lower byte of of a word access. The BYTEMODE bits should be set to 00b (16-bit access) when accessing any other register. See Section 17.2.10 for more details on the BYTEMODE bits.

17.3.1.2.1 Common USB Registers

Table 17-33 lists the common USB registers. Some common USB registers are 8-bit wide and share a word address with other 8-bit registers. Table 17-32 describes how the common USB registers are laid out in memory.

Table 17-32. Common USB Register Layout

CPU Word Address	Register	
	Byte 1	Byte 0
8401h	POWER	FADDR
8402h	INTRTX	
8405h	INTRRX	
8406h	INTRTXE	
8409h	INTRRXE	
840Ah	INTRUSBE	INTRUSB
840Dh	FRAME	
840Eh	TESTMODE	INDEX

Table 17-33. Common USB Registers

CPU Word Address	Acronym	Register Description	Section
8401h	FADDR_POWER	Function Address Register, Power Management Register	Section 17.3.21
8402h	INTRTX	Interrupt Register for Endpoint 0 plus Transmit Endpoints 1 to 4	Section 17.3.23
8405h	INTRRX	Interrupt Register for Receive Endpoints 1 to 4	Section 17.3.24

Table 17-33. Common USB Registers (continued)

CPU Word Address	Acronym	Register Description	Section
8406h	INTRTXE	Interrupt enable register for INTRTX	Section 17.3.25
8409h	INTRRXE	Interrupt Enable Register for INTRRX	Section 17.3.26
840Ah	INTRUSB_INTRUSBE	Interrupt Register for Common USB Interrupts, Interrupt Enable Register	Section 17.3.27
840Dh	FRAME	Frame Number Register	Section 17.3.29
840Eh	INDEX_TESTMODE	Index Register for Selecting the Endpoint Status and Control Registers, Register to Enable the USB 2.0 Test Modes	Section 17.3.30

17.3.1.2.2 Indexed Registers

Table 17-36 lists the index registers. These registers operate on the endpoint selected by the index register. (The index register is the low-8 bits of the INDEX_TESTMODE 16 bits register). Table 17-34 describes how the indexed USB registers are laid out in memory when endpoint 0 is selected in the index register (INDEX = 0). Similarly, Table 17-35 shows the layout of the indexed registers when endpoints 1-4 are selected in the index register (INDEX = 1 or 2 or 3 or 4).

Table 17-34. USB Indexed Register Layout when Index Register Set to Select Endpoint 0

CPU Word Address	Register	
	Byte 1	Byte 0
8411h	Reserved	
8412h	PERI_CSR0	
8415h	Reserved	
8416h	Reserved	
8419h	COUNT0	
841Ah	Reserved	
841Dh	Reserved	
841Eh	CONFIGDATA_INDX	Reserved

Table 17-35. USB Indexed Register Layout when Index Register Set to Select Endpoint 1-4

CPU Word Address	Register	
	Byte 1	Byte 0
8411h	TXMAXP	
8412h	PERI_TXCSR	
8415h	RXMAXP	
8416h	PERI_RXCSR	
8419h	RXCOUNY	
841Ah	Reserved	
841Dh	Reserved	
841Eh	Reserved	

Table 17-36. USB Indexed Registers

CPU Word Address	Acronym	Register Description	Section
8411h	TXMAXP	Maximum Packet Size for Peripheral/Host Transmit Endpoint. (Index register set to select Endpoints 1-4)	Section 17.3.32

Table 17-36. USB Indexed Registers (continued)

CPU Word Address	Acronym	Register Description	Section
8412h	PERI_CSR0	Control Status Register for Peripheral Endpoint 0. (Index register set to select Endpoint 0)	Section 17.3.33
	PERI_TXCSR	Control Status Register for Peripheral Transmit Endpoint. (Index register set to select Endpoints 1-4)	Section 17.3.34
8415h	RXMAXP	Maximum Packet Size for Peripheral/Host Receive Endpoint. (Index register set to select Endpoints 1-4)	Section 17.3.35
8416h	PERI_RXCSR	Control Status Register for Peripheral Receive Endpoint. (Index register set to select Endpoints 1-4)	Section 17.3.36
8419h	COUNT0	Number of Received Bytes in Endpoint 0 FIFO. (Index register set to select Endpoint 0)	Section 17.3.37
	RXCOUNT	Number of Bytes in Host Receive Endpoint FIFO. (Index register set to select Endpoints 1- 4)	Section 17.3.38
841Ah	-	Reserved	
841Dh	-	Reserved	
841Eh	CONFIGDATA_INDC (Upper byte of 841Dh)	Returns details of core configuration. (index register set to select Endpoint 0)	Section 17.3.39

17.3.1.2.3 FIFO Registers

Table 17-37 lists the FIFO registers of the USB2.0 Mentor core.

Table 17-37. USB FIFO Registers

CPU Word Address	Acronym	Register Description	Section
8421h	FIFO0R1	Transmit and Receive FIFO Register 1 for Endpoint 0	Section 17.3.40
8422h	FIFO0R2	Transmit and Receive FIFO Register 2 for Endpoint 0	Section 17.3.40
8425h	FIFO1R1	Transmit and Receive FIFO Register 1 for Endpoint 1	Section 17.3.41
8426h	FIFO1R2	Transmit and Receive FIFO Register 2 for Endpoint 1	Section 17.3.41
8429h	FIFO2R1	Transmit and Receive FIFO Register 1 for Endpoint 2	Section 17.3.42
842Ah	FIFO2R2	Transmit and Receive FIFO Register 2 for Endpoint 2	Section 17.3.42
842Dh	FIFO3R1	Transmit and Receive FIFO Register 1 for Endpoint 3	Section 17.3.43
842Eh	FIFO3R2	Transmit and Receive FIFO Register 2 for Endpoint 3	Section 17.3.43
8431h	FIFO4R1	Transmit and Receive FIFO Register 1 for Endpoint 4	Section 17.3.44
8432h	FIFO4R2	Transmit and Receive FIFO Register 2 for Endpoint 4	Section 17.3.44

17.3.1.2.4 Dynamic FIFO Control Registers

Table 17-39 lists the dynamic FIFO control registers of the US2.0 Mentor core. Some common USB registers are 8-bit wide and share a word address with other 8-bit registers. Table 17-38 describes how the common USB registers are laid out in memory.

Table 17-38. Dynamic FIFO Control Register Layout

CPU Word Address	Register	
	Byte 1	Byte 0
8461h	Reserved	
8462h	RXFIFOSZ	TXFIFOSZ
8465h	TXFIFOADDR	
8466h	RXFIFOADDR	
846Dh	HWVERS	

Table 17-39. Dynamic FIFO Control Registers

CPU Word Address	Acronym	Register Description	Section
8461h	-	Reserved	
8462h	TXFIFOSZ_RXFIFOSZ	Transmit Endpoint FIFO Size, Receive Endpoint FIFO Size (Index register set to select Endpoints 1-4)	Section 17.3.46
8465h	TXFIFOADDR	Transmit Endpoint FIFO Address (Index register set to select Endpoints 1-4)	Section 17.3.48
8466h	RXFIFOADDR	Receive Endpoint FIFO Address (Index register set to select Endpoints 1-4)	Section 17.3.50
846Dh	HWVERS	Hardware Version Register	Section 17.3.49

17.3.1.2.5 Control and Status Registers for Endpoints 0-4

Table 17-40 lists the control and status registers for endpoints 0-4 of the USB2.0 Mentor Core.

Table 17-40. Control and Status Registers for Endpoints 0-4

CPU Word Address	Acronym	Register Description	Section
		Control and Status Register for Endpoint 1	
8511h	TXMAXP	Maximum Packet Size for Peripheral/Host Transmit Endpoint	Section 17.3.32
8512h	PERI_TXCSR	Control Status Register for Peripheral Transmit Endpoint (peripheral mode)	Section 17.3.34
8515h	RXMAXP	Maximum Packet Size for Peripheral/Host Receive Endpoint	Section 17.3.35
8516h	PERI_RXCSR	Control Status Register for Peripheral Receive Endpoint (peripheral mode)	Section 17.3.36
8519h	RXCOUNT	Number of Bytes in Host Receive endpoint FIFO	Section 17.3.38
851Ah	-	Reserved	
851Dh	-	Reserved	
851Eh	-	Reserved	
		Control and Status Register for Endpoint 2	
8521h	TXMAXP	Maximum Packet Size for Peripheral/Host Transmit Endpoint	Section 17.3.32
8522h	PERI_TXCSR	Control Status Register for Peripheral Transmit Endpoint (peripheral mode)	Section 17.3.34
8525h	RXMAXP	Maximum Packet Size for Peripheral/Host Receive Endpoint	Section 17.3.35
8526h	PERI_RXCSR	Control Status Register for Peripheral Receive Endpoint (peripheral mode)	Section 17.3.36
8529h	RXCOUNT	Number of Bytes in Host Receive endpoint FIFO	Section 17.3.38
852Ah	-	Reserved	
852Dh	-	Reserved	
852Eh	-	Reserved	
		Control and Status Register for Endpoint 3	
8531h	TXMAXP	Maximum Packet Size for Peripheral/Host Transmit Endpoint	Section 17.3.32
8532h	PERI_TXCSR	Control Status Register for Peripheral Transmit Endpoint (peripheral mode)	Section 17.3.34
8535h	RXMAXP	Maximum Packet Size for Peripheral/Host Receive Endpoint	Section 17.3.35
8536h	PERI_RXCSR	Control Status Register for Peripheral Receive Endpoint (peripheral mode)	Section 17.3.36
8539h	RXCOUNT	Number of Bytes in Host Receive endpoint FIFO	Section 17.3.38
853Ah	-	Reserved	
853Dh	-	Reserved	
853Eh	-	Reserved	
		Control and Status Register for Endpoint 4	

Table 17-40. Control and Status Registers for Endpoints 0-4 (continued)

CPU Word Address	Acronym	Register Description	Section
8541h	TXMAXP	Maximum Packet Size for Peripheral/Host Transmit Endpoint	Section 17.3.32
8542h	PERI_TXCSR	Control Status Register for Peripheral Transmit Endpoint (peripheral mode)	Section 17.3.34
8545h	RXMAXP	Maximum Packet Size for Peripheral/Host Receive Endpoint	Section 17.3.35
8546h	PERI_RXCSR	Control Status Register for Peripheral Receive Endpoint (peripheral mode)	Section 17.3.36
8549h	RXCOUNT	Number of Bytes in Host Receive endpoint FIFO	Section 17.3.38
854Ah	-	Reserved	
854Dh	-	Reserved	
854Eh	-	Reserved	

17.3.1.3 Communications Port Programming Interface (CPPI) 4.1 DMA Registers

This section lists the registers of the communications port programming interface (CPPI) DMA. Refer to the sections listed for detailed information on each register.

17.3.1.3.1 CPPI DMA (CMDA) Registers

Table 17-41 lists the register of the CPPI DMA (CMDA).

Table 17-41. CPPI DMA (CMDA) Registers

CPU Word Address	Acronym	Register Description	Section
9000h	DMAREVID1	CDMA Revision Identification Register 1	Section 17.3.51
9001h	DMAREVID2	CDMA Revision Identification Register 2	Section 17.3.51
9004h	TDFDQ	CDMA Teardown Free Descriptor Queue Control Register	Section 17.3.52
9008h	DMAEMU	CDMA Emulation Control Register	Section 17.3.53
9800h	TXGCR1[0]	Transmit Channel 0 Global Configuration Register 1	Section 17.3.54
9801h	TXGCR2[0]	Transmit Channel 0 Global Configuration Register 2	Section 17.3.54
9808h	RXGCR1[0]	Receive Channel 0 Global Configuration Register 1	Section 17.3.55
9809h	RXGCR2[0]	Receive Channel 0 Global Configuration Register 2	Section 17.3.55
980Ch	RXHPCR1A[0]	Receive Channel 0 Host Packet Configuration Register 1 A	Section 17.3.56
980Dh	RXHPCR2A[0]	Receive Channel 0 Host Packet Configuration Register 2 A	Section 17.3.56
9810h	RXHPCR1B[0]	Receive Channel 0 Host Packet Configuration Register 1 B	Section 17.3.57
9811h	RXHPCR2B[0]	Receive Channel 0 Host Packet Configuration Register 2 B	Section 17.3.57
9820h	TXGCR1[1]	Transmit Channel 1 Global Configuration Register 1	Section 17.3.54
9821h	TXGCR2[1]	Transmit Channel 1 Global Configuration Register 2	Section 17.3.54
9828h	RXGCR1[1]	Receive Channel 1 Global Configuration Register 1	Section 17.3.55
9829h	RXGCR2[1]	Receive Channel 1 Global Configuration Register 2	Section 17.3.55
982Ch	RXHPCR1A[1]	Receive Channel 1 Host Packet Configuration Register 1 A	Section 17.3.56
982Dh	RXHPCR2A[1]	Receive Channel 1 Host Packet Configuration Register 2 A	Section 17.3.56
9830h	RXHPCR1B[1]	Receive Channel 1 Host Packet Configuration Register 1 B	Section 17.3.57
9831h	RXHPCR2B[1]	Receive Channel 1 Host Packet Configuration Register 2 B	Section 17.3.57
9840h	TXGCR1[2]	Transmit Channel 2 Global Configuration Register 1	Section 17.3.54
9841h	TXGCR2[2]	Transmit Channel 2 Global Configuration Register 2	Section 17.3.54
9848h	RXGCR1[2]	Receive Channel 2 Global Configuration Register 1	Section 17.3.55
9849h	RXGCR2[2]	Receive Channel 2 Global Configuration Register 2	Section 17.3.55
984Ch	RXHPCR1A[2]	Receive Channel 2 Host Packet Configuration Register 1 A	Section 17.3.56

Table 17-41. CPPI DMA (CMDA) Registers (continued)

CPU Word Address	Acronym	Register Description	Section
984Dh	RXHPCR2A[2]	Receive Channel 2 Host Packet Configuration Register 2 A	Section 17.3.56
9850h	RXHPCR1B[2]	Receive Channel 2 Host Packet Configuration Register 1 B	Section 17.3.57
9851h	RXHPCR2B[2]	Receive Channel 2 Host Packet Configuration Register 2 B	Section 17.3.57
9860h	TXGCR1[3]	Transmit Channel 3 Global Configuration Register 1	Section 17.3.54
9861h	TXGCR2[3]	Transmit Channel 3 Global Configuration Register 2	Section 17.3.54
9868h	RXGCR1[3]	Receive Channel 3 Global Configuration Register 1	Section 17.3.55
9869h	RXGCR2[3]	Receive Channel 3 Global Configuration Register 2	Section 17.3.55
986Ch	RXHPCR1A[3]	Receive Channel 3 Host Packet Configuration Register 1 A	Section 17.3.56
986Dh	RXHPCR2A[3]	Receive Channel 3 Host Packet Configuration Register 2 A	Section 17.3.56
9870h	RXHPCR1B[3]	Receive Channel 3 Host Packet Configuration Register 1 B	Section 17.3.57
9871h	RXHPCR2B[3]	Receive Channel 3 Host Packet Configuration Register 2 B	Section 17.3.57
A000h	DMA_SCHED_CTRL1	CDMA Scheduler Control Register 1	Section 17.3.58
A001h	DMA_SCHED_CTRL2	CDMA Scheduler Control Register 1	Section 17.3.58
A800h + 4 × *N*	ENTRYLSW[*N*]	CDMA Scheduler Table Word *N* Registers LSW (*N* = 0 to 63)	Section 17.3.59
A801h + 4 × *N*	ENTRYMSW[*N*]	CDMA Scheduler Table Word *N* Registers MSW (*N* = 0 to 63)	Section 17.3.59

17.3.1.3.2 Queue Manager (QMGR) Registers

Table 17-42 lists the registers of the queue manager.

Table 17-42. Queue Manager (QMGR) Registers

CPU Word Address	Acronym	Register Description	Section
C000h	QMGRREVID1	Queue Manager Revision Identification Register 1	Section 17.3.60
C001h	QMGRREVID2	Queue Manager Revision Identification Register 2	Section 17.3.60
C008h	DIVERSION1	Queue Manager Queue Diversion Register 1	Section 17.3.61
C009h	DIVERSION2	Queue Manager Queue Diversion Register 2	Section 17.3.61
C020h	FDBSC0	Queue Manager Free Descriptor/Buffer Starvation Count Register 0	Section 17.3.62
C021h	FDBSC1	Queue Manager Free Descriptor/Buffer Starvation Count Register 1	Section 17.3.63
C024h	FDBSC2	Queue Manager Free Descriptor/Buffer Starvation Count Register 2	Section 17.3.64
C025h	FDBSC3	Queue Manager Free Descriptor/Buffer Starvation Count Register 3	Section 17.3.65
C028h	FDBSC4	Queue Manager Free Descriptor/Buffer Starvation Count Register 4	Section 17.3.66
C029h	FDBSC5	Queue Manager Free Descriptor/Buffer Starvation Count Register 5	Section 17.3.67
C02Ch	FDBSC6	Queue Manager Free Descriptor/Buffer Starvation Count Register 6	Section 17.3.68
C02Dh	FDBSC7	Queue Manager Free Descriptor/Buffer Starvation Count Register 7	Section 17.3.69
C080h	LRAM0BASE1	Queue Manager Linking RAM Region 0 Base Address Register 1	Section 17.3.70
C081h	LRAM0BASE2	Queue Manager Linking RAM Region 0 Base Address Register 2	Section 17.3.70
C084h	LRAM0SIZE	Queue Manager Linking RAM Region 0 Size Register	Section 17.3.71
C085h	-	Reserved	
C088h	LRAM1BASE1	Queue Manager Linking RAM Region 1 Base Address Register 1	Section 17.3.72
C089h	LRAM1BASE2	Queue Manager Linking RAM Region 1 Base Address Register 2	Section 17.3.72
C090h	PEND0	Queue Manager Queue Pending 0	Section 17.3.73
C091h	PEND1	Queue Manager Queue Pending 1	Section 17.3.74
C094h	PEND2	Queue Manager Queue Pending 2	Section 17.3.75
C095h	PEND3	Queue Manager Queue Pending 3	Section 17.3.76
C098h	PEND4	Queue Manager Queue Pending 4	Section 17.3.77
C099h	PEND5	Queue Manager Queue Pending 5	Section 17.3.78

Table 17-42. Queue Manager (QMGR) Registers (continued)

CPU Word Address	Acronym	Register Description	Section
D000h + 16 × *R*	QMEMRBASE1[*R*]	Queue Manager Memory Region *R* Base Address Register 1 (*R* = 0 to 15)	Section 17.3.79
D001h + 16 × *R*	QMEMRBASE2[*R*]	Queue Manager Memory Region *R* Base Address Register 2 (*R* = 0 to 15)	Section 17.3.79
D004h + 16 × *R*	QMEMRCTRL1[*R*]	Queue Manager Memory Region *R* Control Register (*R* = 0 to 15)	Section 17.3.80
D005h + 16 × *R*	QMEMRCTRL2[*R*]	Queue Manager Memory Region *R* Control Register (*R* = 0 to 15)	Section 17.3.80
E00Ch + 16 × *N*	CTRL1D	Queue Manager Queue *N* Control Register 1 D (*N* = 0 to 63)	Section 17.3.81
E00Dh + 16 × *N*	CTRL2D	Queue Manager Queue *N* Control Register 2 D (*N* = 0 to 63)	Section 17.3.81
E800h + 16 × *N*	QSTATA	Queue Manager Queue *N* Status Register A (*N* = 0 to 63)	Section 17.3.82
E804h + 16 × *N*	QSTAT1B	Queue Manager Queue *N* Status Register 1 B (*N* = 0 to 63)	Section 17.3.83
E805h + 16 × *N*	QSTAT2B	Queue Manager Queue *N* Status Register 2 B (*N* = 0 to 63)	Section 17.3.83
E808h + 16 × *N*	QSTATC	Queue Manager Queue *N* Status Register C (*N* = 0 to 63)	Section 17.3.84

17.3.2 Revision Identification Registers (REVID1 and REVID2)

The revision identification registers (REVID1 and REVID2) contain the revision for the USB 2.0 controller module. The REVID1 is shown in Figure 17-23 and described in Table 17-43. The REVID2 is shown in Figure 17-24 and described in Table 17-44.

Figure 17-23. Revision Identification Register 1 (REVID1)

15	0
REVLSB	
R- 0800h	

LEGEND: R = Read only; -*n* = value after reset

Figure 17-24. Revision Identification Register 2 (REVID2)

15	0
REVMSB	
R-4EA1h	

LEGEND: R = Read only; -*n* = value after reset

Table 17-43. Revision Identification Register 1 (REVID1) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	REVLSB	R	0800h	Least significant bits of the revision ID of the USB module. Value: 0-FFFFh

Table 17-44. Revision Identification Register 2 (REVID2) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	REVMSB	R	R-4EA1h	Most significant bits of the revision ID of the USB module. Value: 0-FFFFh

17.3.3 Control Register (CTRLR)

The control register (CTRLR) allows the CPU to control various aspects of the module. The CTRLR is shown in Figure 17-25 and described in Table 17-45.

Figure 17-25. Control Register (CTRLR)

15	5	4	3	2	1	0
Reserved		RNDIS	UINT	Reserved	CLKFACK	RESET
R-0		R/W-0	R/W-0	R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-45. Control Register (CTRLR) Field Descriptions

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0	Reserved
4	RNDIS	RW	0	Global RNDIS mode enable for all endpoints. 0 = Global RNDIS mode is disabled. 1 = Global RNDIS mode is enabled.
3	UINT	RW	0	USB non-PDR interrupt handler enable. 0 = PDR interrupt handler is enabled. 1 = PDR interrupt handler is disabled
2	Reserved	R	0	Reserved
1	CLKFACK	RW	0	Clock stop fast ACK enable. 0 = Clock stop fast ACK is disabled. 1 = Clock stop fast ACK is enabled.
0	RESET	RW	0	Soft reset. 0 = No effect. 1 = Writing a 1 starts a module reset.

17.3.4 Emulation Register (EMUR)

The emulation register (EMUR) allows the CPU to configure the CBA 3.0 emulation interface. The EMUR is shown in Figure 17-26 and described in Table 17-46.

Figure 17-26. Emulation Register (EMUR)

15	3	2	1	0
Reserved		RT_SEL	SOFT	FREERUN
R-0		R/W-0	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-46. Emulation Register (EMUR) Field Descriptions

Bit	Field	Type	Reset	Description
15-3	Reserved	R	0	Reserved.
2	RT_SEL	RW	0	Real-time enable. 0 = Enable 1 = Disable
1	SOFT	RW	1	Soft stop. 0 = No effect. 1 = Soft stop enable.
0	FREERUN	RW	1	Free run. 0 = No effect. 1 = Free run enable

17.3.5 Mode Registers (MODE1 and MODE2)

The mode registers (MODE1 and MODE2) allow the CPU to individually enable RNDIS/Generic/CDC modes for each endpoint. Using the global RNDIS bit in the control register (CTRLR) overrides this register and enables RNDIS mode for all endpoints. The MODE1 is shown in Figure 17-27 and described in Table 17-47. The MODE2 is shown in Figure 17-28 and described in Table 17-48.

Figure 17-27. Mode Register 1 (MODE1)

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
Reserved	TX4_MODE	Reserved	TX3_MODE	Reserved	TX2_MODE	Reserved	TX1_MODE
R	R/W	R	R/W	R	R/W	R	R/W

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 17-28. Mode Register 2 (MODE2)

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
Reserved	RX4_MODE	Reserved	RX3_MODE	Reserved	RX2_MODE	Reserved	RX1_MODE
R	R/W	R	R/W	R	R/W	R	R/W

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-47. Mode Register 1 (MODE1) Field Descriptions

Bit	Field	Type	Reset	Description
15-14	Reserved	R	0	Reserved.
13-12	TX4_MODE	RW	0	Transmit endpoint 4 mode control. 0 = Transparent mode on Transmit endpoint 4. 1h = RNDIS mode on Transmit endpoint 4. 2h = CDC mode on Transmit endpoint 4. 3h = Generic RNDIS mode on Transmit endpoint 4.
11-10	Reserved	R	0	Reserved.
9-8	TX3_MODE	RW	0	Transmit endpoint 3 mode control. 0 = Transparent mode on Transmit endpoint 3. 1h = RNDIS mode on Transmit endpoint 3. 2h = CDC mode on Transmit endpoint 3. 3h = Generic RNDIS mode on Transmit endpoint 3.
7-6	Reserved	R	0	Reserved.
5-4	TX2_MODE	RW	0	Transmit endpoint 2 mode control. 0 = Transparent mode on Transmit endpoint 2. 1h = RNDIS mode on Transmit endpoint 2. 2h = CDC mode on Transmit endpoint 2. 3h = Generic RNDIS mode on Transmit endpoint 2.
3-2	Reserved	R	0	Reserved.
1-0	TX1_MODE	RW	0	Transmit endpoint 1 mode control. 0 = Transparent mode on Transmit endpoint 1. 1h = RNDIS mode on Transmit endpoint 1. 2h = CDC mode on Transmit endpoint 1. 3h = Generic RNDIS mode on Transmit endpoint 1.

Table 17-48. Mode Register 2 (MODE2) Field Descriptions

Bit	Field	Type	Reset	Description
15-14	Reserved	R	0	Reserved.

Table 17-48. Mode Register 2 (MODE2) Field Descriptions (continued)

Bit	Field	Type	Reset	Description
13-12	RX4_MODE	RW	0	Receive endpoint 4 mode control. 0 = Transparent mode on Receive endpoint 4. 1h = RNDIS mode on Receive endpoint 4. 2h = CDC mode on Receive endpoint 4. 3h = Generic RNDIS mode on Receive endpoint 4.
11-10	Reserved	R	0	Reserved.
9-8	RX3_MODE	RW	0	Receive endpoint 3 mode control. 0 = Transparent mode on Receive endpoint 3. 1h = RNDIS mode on Receive endpoint 3. 2h = CDC mode on Receive endpoint 3. 3h = Generic RNDIS mode on Receive endpoint 3.
7-6	Reserved	R	0	Reserved.
5-4	RX2_MODE	RW	0	Receive endpoint 2 mode control. 0 = Transparent mode on Receive endpoint 2. 1h = RNDIS mode on Receive endpoint 2. 2h = CDC mode on Receive endpoint 2. 3h = Generic RNDIS mode on Receive endpoint 2.
3-2	Reserved	R	0	Reserved.
1-0	RX1_MODE	RW	0	Receive endpoint 1 mode control. 0 = Transparent mode on Receive endpoint 1. 1h = RNDIS mode on Receive endpoint 1. 2h = CDC mode on Receive endpoint 1. 3h = Generic RNDIS mode on Receive endpoint 1.

17.3.6 Auto Request Register (AUTOREQ)

The auto request register (AUTOREQ) allows the CPU to enable an automatic IN token request generation for host mode RX operation per each RX endpoint. This feature has the DMA set the REQPKT bit in the control status register for host receive endpoint (HOST_RXCSR) when it clears the RXPKTRDY bit after reading out a packet. The REQPKT bit is used by the core to generate an IN token to receive data. By using this feature, the host can automatically generate an IN token after the DMA finishes receiving data and empties an endpoint buffer, thus receiving the next data packet as soon as possible from the connected device. Without this feature, the CPU will have to manually set the REQPKT bit for every USB packet.

There are two modes that auto request can function in: always or all except an EOP. The always mode sets the REQPKT bit after every USB packet the DMA receives thus generating a new IN token after each USB packet. The EOP mode sets the REQPKT bit after every USB packet that is not an EOP (end of packet) in the CPPI descriptor. For RNDIS, CDC, and Generic RNDIS modes, the auto request stops when the EOP is received (either via a short packet for RNDIS, CDC, and Generic RNDIS or the count is reached for Generic RNDIS), making it useful for starting a large RNDIS packet and having it auto generate IN tokens until the end of the RNDIS packet. For transparent mode, every USB packet is an EOP CPPI packet so the auto request never functions and acts like auto request is disabled.

The AUTOREQ is shown in Figure 17-29 and described in Table 17-49.

Figure 17-29. Auto Request Register (AUTOREQ)

15	8	7	6	5	4	3	2	1	0
Reserved		RX4_AUTREQ		RX3_AUTREQ		RX2_AUTREQ		RX1_AUTREQ	
R-0		R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-49. Auto Request Register (AUTOREQ) Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0	Reserved.
7-6	RX4_AUTREQ	RW	0	Receive endpoint 4 auto request enable. 0 = No auto request. 1h = Auto request on all but EOP. 2h = Reserved. 3h = Auto request always.
5-4	RX3_AUTREQ	RW	0	Receive endpoint 3 auto request enable. 0 = No auto request. 1h = Auto request on all but EOP. 2h = Reserved. 3h = Auto request always.
3-2	RX2_AUTREQ	RW	0	Receive endpoint 2 auto request enable. 0 = No auto request. 1h = Auto request on all but EOP. 2h = Reserved. 3h = Auto request always.
1-0	RX1_AUTREQ	RW	0	Receive endpoint 1 auto request enable. 0 = No auto request. 1h = Auto request on all but EOP. 2h = Reserved. 3h = Auto request always.

17.3.7 Teardown Registers (TEARDOWN1 and TEARDOWN2)

The teardown registers (TEARDOWN1 and TEARDOWN2) control the tearing down of receive and transmit FIFOs in the USB controller. When a 1 is written to a valid bit in TEARDOWN1 or TEARDOWN2, the CPPI FIFO pointers for that endpoint are cleared. TEARDOWN1 and TEARDOWN2 must be used in conjunction with the CPPI DMA teardown mechanism. The Host should also write the FLUSHFIFO bits in the TXCSR and RXCSR registers to ensure a complete teardown of the endpoint.

The TEARDOWN1 is shown in Figure 17-30 and described in Table 17-50. The TEARDOWN2 is shown in Figure 17-31 and described in Table 17-51.

Figure 17-30. Teardown Register 1 (TEARDOWN1)

15	5	4	1	0
Reserved		RX_TDOWN		Reserved
R-0		R/W-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 17-31. Teardown Register 2 (TEARDOWN2)

15	5	4	1	0
Reserved		TX_TDOWN		Reserved
R-0		R/W-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-50. Teardown Register 1 (TEARDOWN1) Field Descriptions

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0	Reserved.
4-1	RX_TDOWN	RW	0	Receive endpoint teardown. 0 = Disable. 1 = Enable.
0	Reserved	R	0	Reserved.

Table 17-51. Teardown Register 2 (TEARDOWN2) Field Descriptions

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0	Reserved.
4-1	TX_TDOWN	RW		Transmit endpoint teardown. 0 = Disable. 1 = Enable.
0	Reserved	R	0	Reserved.

17.3.8 USB Interrupt Source Registers (INTSRCR1 and INTSRCR2)

The USB interrupt source registers (INTSRCR1 and INTSRCR2) contain the status of the interrupt sources generated by the USB core (not the DMA). The INTSRCR1 is shown in Figure 17-32 and described in Table 17-52. The INTSRCR2 is shown in Figure 17-33 and described in Table 17-53.

Figure 17-32. USB Interrupt Source Register 1 (INTSRCR1)

15 13	12 9	8 5	4 0
Reserved	RX	Reserved	TX
	R-0		R-0

LEGEND: R = Read only; -*n* = value after reset

Figure 17-33. USB Interrupt Source Register 2 (INTSRCR2)

15 9	8 0
Reserved	USB
R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 17-52. USB Interrupt Source Register 1 (INTSRCR1) Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0	Reserved.
12	RX4	R	0	Receive interrupt source for EndPoint4. Value: 0/1
11	RX3	R	0	Receive interrupt source for EndPoint3. Value: 0/1
10	RX2	R	0	Receive interrupt source for EndPoint2. Value: 0/1
9	RX1	R	0	Receive interrupt source for EndPoint1. Value: 0/1
8-5	Reserved	R	0	Reserved
4	TX4	R	0	Transmit interrupt source for EndPoint4. Value: 0/1
3	TX3	R	0	Transmit interrupt source for EndPoint3. Value: 0/1
2	TX2	R	0	Transmit interrupt source for EndPoint2. Value: 0/1
1	TX1	R	0	Transmit interrupt source for EndPoint1. Value: 0/1
0	RX1/TX1	R	0	Both Receive and Transmit interrupt source for EndPoint0. Value: 0/1

Table 17-53. USB Interrupt Source Register 2 (INTSRCR2) Field Descriptions

Bit	Field	Type	Reset	Description
15-9	Reserved	R	0	Reserved.
8-0	USB	R	0	USB interrupt sources. (Please see Figure 17-64 for the definition of each bit here.) Value: 0-1FFh.

17.3.9 USB Interrupt Source Set Registers (INTSETR1 and INTSETR2)

The USB interrupt source set registers (INTSETR1 and INTSETR2) allow the USB interrupt sources to be manually triggered. A read of this register returns the USB interrupt source register value. The INTSETR1 is shown in Figure 17-34 and described in Table 17-54. The INTSETR2 is shown in Figure 17-35 and described in Table 17-55.

Figure 17-34. USB Interrupt Source Set Register 1 (INTSETR1)

15 13	12 9	8 5	4 0
Reserved	RX	Reserved	TX
R-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 17-35. USB Interrupt Source Set Register 2 (INTSETR2)

15 9	8 0
Reserved	USB
R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-54. USB Interrupt Source Set Register 1 (INTSETR1) Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0	Reserved.
12-8	RX	RW	0	Write a 1 to set equivalent Receive endpoint interrupt source. Allows the USB interrupt sources to be manually triggered. Value: 0-Fh
7-5	Reserved	R	0	Reserved.
4-0	TX	RW	0	Write a 1 to set equivalent Transmit endpoint interrupt source. Allows the USB interrupt sources to be manually triggered. Value: 0-Fh

Table 17-55. USB Interrupt Source Set Register 2 (INTSETR2) Field Descriptions

Bit	Field	Type	Reset	Description
15-9	Reserved	R	0	Reserved.
8-0	USB	RW	0	Write a 1 to set equivalent USB interrupt source. Allows the USB interrupt sources to be manually triggered. Value: 0-1FFh

17.3.10 USB Interrupt Source Clear Registers (INTCLRR1 and INTCLRR2)

The USB interrupt source clear registers (INTCLRR1 and INTCLRR2) allow the CPU to acknowledge an interrupt source and turn it off. A read of this register returns the USB interrupt source register value. The INTCLRR1 is shown in Figure 17-36 and described in Table 17-56. The INTCLRR2 is shown in Figure 17-37 and described in Table 17-57.

Figure 17-36. USB Interrupt Source Clear Register 1 (INTCLRR1)

15 13	12 9	8 5	4 0
Reserved	RX	Reserved	TX
R-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 17-37. USB Interrupt Source Clear Register 2 (INTCLRR2)

15 9	8 0
Reserved	USB
R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-56. USB Interrupt Source Clear Register 1 (INTCLRR1) Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0	Reserved.
12-8	RX	RW	0	Write a 1 to clear equivalent Receive endpoint interrupt source. Allows the CPU to acknowledge an interrupt source and turn it off. Value: 0-Fh
7-5	Reserved	R	0	Reserved.
4-0	TX	RW	0	Write a 1 to clear equivalent Transmit endpoint interrupt source. Allows the CPU to acknowledge an interrupt source and turn it off. Value: 0-1Fh

Table 17-57. USB Interrupt Source Clear Register 2 (INTCLRR2) Field Descriptions

Bit	Field	Type	Reset	Description
15-9	Reserved	R	0	Reserved.
8-0	USB	RW	0	Write a 1 to clear equivalent USB interrupt source. Allows the CPU to acknowledge an interrupt source and turn it off. Value: 0-1FFh

17.3.11 USB Interrupt Mask Registers (INTMSKR1 and INTMSKR2)

The USB interrupt mask registers (INTMSKR1 and INTMSKR2) contain the masks of the interrupt sources generated by the USB core (not the DMA). These masks are used to enable or disable interrupt sources generated on the masked source interrupts (the raw source interrupts are never masked). The bit positions are maintained in the same position as the interrupt sources in the USB interrupt source register.

The INTMSKR1 is shown in Figure 17-38 and described in Table 17-58. The INTMSKR2 is shown in Figure 17-39 and described in Table 17-59.

Figure 17-38. USB Interrupt Mask Register 1 (INTMSKR1)

15 13	12 9	8 5	4 0
Reserved	RX	Reserved	TX
R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Figure 17-39. USB Interrupt Mask Register 2 (INTMSKR2)

15 9	8 0
Reserved	USB
R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 17-58. USB Interrupt Mask Register 1 (INTMSKR1) Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0	Reserved.
12-8	RX	R	0	Receive endpoint interrupt source masks. Value: 0-Fh
7-5	Reserved	R	0	Reserved.
4-0	TX	R	0	Transmit endpoint interrupt source masks. Value: 0-1Fh

Table 17-59. USB Interrupt Mask Register 2 (INTMSKR2) Field Descriptions

Bit	Field	Type	Reset	Description
15-9	Reserved	R	0	Reserved.
8-0	USB	R	0	USB interrupt source masks. Value: 0-1FFh

17.3.12 USB Interrupt Mask Set Registers (INTMSKSETR1 and INTMSKSETR2)

The USB interrupt mask set registers (INTMSKSETR1 and INTMSKSETR2) allow the USB masks to be individually enabled. A read to this register returns the USB interrupt mask register value. The INTMSKSETR1 is shown in Figure 17-40 and described in Table 17-60. The INTMSKSETR2 is shown in Figure 17-41 and described in Table 17-61.

Figure 17-40. USB Interrupt Mask Set Register 1 (INTMSKSETR1)

15 13	12 9	8 5	4 0
Reserved	RX	Reserved	TX
R-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 17-41. USB Interrupt Mask Set Register 2 (INTMSKSETR2)

15 9	8 0
Reserved	USB
R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-60. USB Interrupt Mask Set Register 1 (INTMSKSETR1) Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0	Reserved.
12-8	RX	RW	0	Write a 1 to set equivalent Receive endpoint interrupt mask. Value: 0-Fh
7-5	Reserved	R	0	Reserved.
4-0	TX	RW	0	Write a 1 to set equivalent Transmit endpoint interrupt mask. Value: 0-1Fh

Table 17-61. USB Interrupt Mask Set Register 2 (INTMSKSETR2) Field Descriptions

Bit	Field	Type	Reset	Description
15-9	Reserved	R	0	Reserved.
8-0	USB	RW	0	Write a 1 to set equivalent USB interrupt mask. Value: 0-1FFh

17.3.13 USB Interrupt Mask Clear Registers (INTMSKCLRR1 and INTMSKCLRR2)

The USB interrupt mask clear registers (INTMSKCLRR1 and INTMSKCLRR2) allow the USB interrupt masks to be individually disabled. A read to this register returns the USB interrupt mask register value. The INTMSKCLRR1 is shown in Figure 17-42 and described in Table 17-62. The INTMSKCLRR2 is shown in Figure 17-43 and described in Table 17-63.

Figure 17-42. USB Interrupt Mask Clear Register 1 (INTMSKCLRR1)

15 13	12 9	8 5	4 0
Reserved	RX	Reserved	TX
R-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 17-43. USB Interrupt Mask Clear Register 2 (INTMSKCLRR2)

15 9	8 0
Reserved	USB
R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-62. USB Interrupt Mask Clear Register 1 (INTMSKCLRR1) Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0	Reserved.
12-8	RX	RW	0	Write a 1 to clear equivalent Receive endpoint interrupt mask. Value: 0-Fh
7-5	Reserved	R	0	Reserved.
4-0	TX	RW	0	Write a 1 to clear equivalent Transmit endpoint interrupt mask. Value: 0-1Fh

Table 17-63. USB Interrupt Mask Clear Register 2 (INTMSKCLRR2) Field Descriptions

Bit	Field	Type	Reset	Description
15-9	Reserved	R	0	Reserved.
8-0	USB	RW	0	Write a 1 to clear equivalent USB interrupt mask. Value: 0-1FFh

17.3.14 USB Interrupt Source Masked Registers (INTMASKEDR1 and INTMASKEDR2)

The USB interrupt source masked registers (INTMASKEDR1 and INTMASKEDR2) contain the status of the interrupt sources generated by the USB core masked by the USB interrupt mask register values. The INTMASKEDR1 is shown in Figure 17-44 and described in Table 17-64. The INTMASKEDR2 is shown in Figure 17-45 and described in Table 17-65.

Figure 17-44. USB Interrupt Source Masked Register 1 (INTMASKEDR1)

15 13	12 9	8 5	4 0
Reserved	RX	Reserved	TX
R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Figure 17-45. USB Interrupt Source Masked Register 2 (INTMASKEDR2)

15 9	8 0
Reserved	USB
R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 17-64. USB Interrupt Source Masked Register 1 (INTMASKEDR1) Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0	Reserved.
12-8	RX	RW	0	Receive endpoint interrupt sources masked. Value: 0-Fh
7-5	Reserved	R	0	Reserved.
4-0	TX	RW	0	Transmit endpoint interrupt sources masked. Value: 0-1Fh

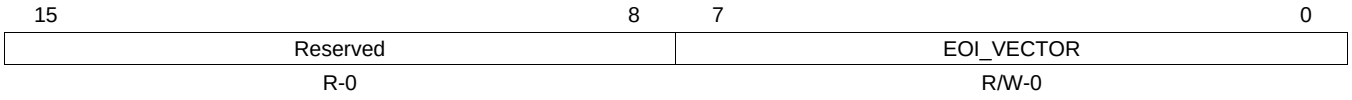
Table 17-65. USB Interrupt Source Masked Register 2 (INTMASKEDR2) Field Descriptions

Bit	Field	Type	Reset	Description
15-9	Reserved	R	0	Reserved.
8-0	USB	R	0	USB interrupt sources masked. Value: 0-1FFh

17.3.15 USB End of Interrupt Register (EOIR)

The USB end of interrupt register (EOIR) allows the CPU to acknowledge completion of an interrupt by writing 0 to the EOI_VECTOR bit. The EOIR is shown in Figure 17-46 and described in Table 17-66.

Figure 17-46. USB End of Interrupt Register (EOIR)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

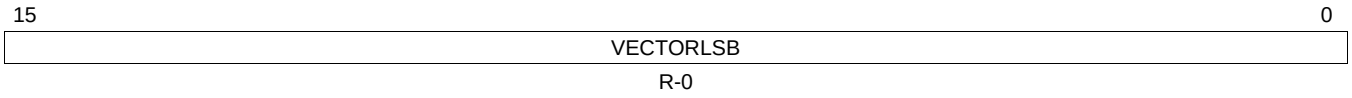
Table 17-66. USB End of Interrupt Register (EOIR) Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0	Reserved.
7-0	EOI_VECTOR	RW	0	EOI Vector. Value: 0-FFh

17.3.16 USB Interrupt Vector Registers (INTVECTR1 and INTVECTR2)

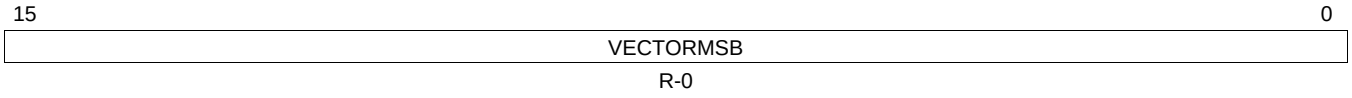
The USB interrupt vector registers (INTVECTR1 and INTVECTR2) recycle the Interrupt Vector input to be read by the CPU. The INTVECTR1 is shown in Figure 17-47 and described in Table 17-67. The INTVECTR2 is shown in Figure 17-48 and described in Table 17-68.

Figure 17-47. USB Interrupt Vector Register 1 (INTVECTR1)



LEGEND: R = Read only; -*n* = value after reset

Figure 17-48. USB Interrupt Vector Register 2 (INTVECTR2)



LEGEND: R = Read only; -*n* = value after reset

Table 17-67. USB Interrupt Vector Register 1 (INTVECTR1) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	VECTORLSB	R	0	Input Interrupt Vector. Together, INTVECTR1 and INTVECTR2 specify a 32 bit value. Value: 0-FFFFh

Table 17-68. USB Interrupt Vector Register 2 (INTVECTR2) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	VECTORMSB	R	0	Input Interrupt Vector. Together, INTVECTR1 and INTVECTR2 specify a 32 bit value. Value: 0-FFFFh

17.3.17 Generic RNDIS EP1 Size Registers (GREP1SZR1 and GREP1SZR2)

The generic RNDIS EP1 size registers (GREP1SZR1 and GREP1SZR2) are programmed with a RNDIS packet size in bytes. When EP1 is in Generic RNDIS mode, the received USB packets are collected into a single CPPI packet that is completed when the number of bytes equal to the value of this register have been received, or a *short* packet is received. The packet size must be an integer multiple of the endpoint size. The maximum packet size that can be used is 10000h, or 65536.

The GREP1SZR1 is shown in Figure 17-49 and described in Table 17-69. The GREP1SZR2 is shown in Figure 17-50 and described in Table 17-70.

Figure 17-49. Generic RNDIS EP1 Size Register 1 (GREP1SZR1)

15	0
EP1_SIZE_LSB	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 17-50. Generic RNDIS EP1 Size Register 2 (GREP1SZR2)

15	1	0
Reserved		EP1_SIZE_MSB
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-69. Generic RNDIS EP1 Size Register 1 (GREP1SZR1) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	EP1_SIZE_LSB	RW	0	Generic RNDIS packet size. Together, GREP1SZR1 and GREP1SZR2 specify the packet size. Value: 0-FFFFh

Table 17-70. Generic RNDIS EP1 Size Register 2 (GREP1SZR2) Field Descriptions

Bit	Field	Type	Reset	Description
15-1	Reserved	R	0	Reserved.
0	EP1_SIZE_MSB	RW	0	Generic RNDIS packet size. Together, GREP1SZR1 and GREP1SZR2 specify the packet size. Value: 0-1

17.3.18 Generic RNDIS EP2 Size Registers (GREP2SZR1 and GREP2SZR2)

The generic RNDIS EP2 size registers (GREP2SZR1 and GREP2SZR2) are programmed with a RNDIS packet size in bytes. When EP2 is in Generic RNDIS mode, the received USB packets are collected into a single CPPI packet that is completed when the number of bytes equal to the value of this register have been received, or a *short* packet is received. The packet size must be an integer multiple of the endpoint size. The maximum packet size that can be used is 10000h, or 65536.

The GREP2SZR1 is shown in Figure 17-51 and described in Table 17-71. The GREP2SZR2 is shown in Figure 17-52 and described in Table 17-72.

Figure 17-51. Generic RNDIS EP2 Size Register 1 (GREP2SZR1)

15	0
EP2_SIZE_LSB	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 17-52. Generic RNDIS EP2 Size Register 2 (GREP2SZR2)

15	1	0
Reserved		EP2_SIZE_MSB
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-71. Generic RNDIS EP2 Size Register 1 (GREP2SZR1) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	EP2_SIZE_LSB	RW	0	Generic RNDIS packet size. Together, GREP2SZR1 and GREP2SZR2 specify the packet size. Value: 0-FFFFh

Table 17-72. Generic RNDIS EP2 Size Register 2 (GREP2SZR2) Field Descriptions

Bit	Field	Type	Reset	Description
15-1	Reserved	R	0	Reserved.
0	EP2_SIZE_MSB	RW	0	Generic RNDIS packet size. Together, GREP2SZR1 and GREP2SZR2 specify the packet size. Value: 0-1

17.3.19 Generic RNDIS EP3 Size Registers (GREP3SZR1 and GREP3SZR2)

The generic RNDIS EP3 size registers (GREP3SZR1 and GREP3SZR2) are programmed with a RNDIS packet size in bytes. When EP3 is in Generic RNDIS mode, the received USB packets are collected into a single CPPI packet that is completed when the number of bytes equal to the value of this register has been received, or a *short* packet is received. The packet value must be an integer multiple of the endpoint size. The maximum packet size that can be used is 10000h, or 65536.

The GREP3SZR1 is shown in Figure 17-53 and described in Table 17-73. The GREP3SZR2 is shown in Figure 17-54 and described in Table 17-74.

Figure 17-53. Generic RNDIS EP3 Size Register 1 (GREP3SZR1)

15	0
EP3_SIZE_LSB	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 17-54. Generic RNDIS EP3 Size Register 2 (GREP3SZR2)

15	1	0
Reserved		EP3_SIZE_MSB
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-73. Generic RNDIS EP3 Size Register 1 (GREP3SZR1) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	EP3_SIZE_LSB	RW	0	Generic RNDIS packet size. Together, GREP3SZR1 and GREP3SZR2 specify the packet size. Value: 0-FFFFh

Table 17-74. Generic RNDIS EP3 Size Register 2 (GREP3SZR2) Field Descriptions

Bit	Field	Type	Value	Description
15-1	Reserved	R	0	Reserved.
0	EP3_SIZE_MSB	RW	0	Generic RNDIS packet size. Together, GREP3SZR1 and GREP3SZR2 specify the packet size. Value: 0-1

17.3.20 Generic RNDIS EP4 Size Registers (GREP4SZR1 and GREP4SZR2)

The generic RNDIS EP4 size registers (GREP4SZR1 and GREP4SZR2) are programmed with a RNDIS packet size in bytes. When EP4 is in Generic RNDIS mode, the received USB packets are collected into a single CPPI packet that is completed when the number of bytes equal to the value of this register has been received, or a *short* packet is received. The packet size must be an integer multiple of the endpoint size. The maximum packet size that can be used is 10000h, or 65536.

The GREP4SZR1 is shown in Figure 17-55 and described in Table 17-75. The GREP4SZR2 is shown in Figure 17-56 and described in Table 17-76.

Figure 17-55. Generic RNDIS EP4 Size Register 1 (GREP4SZR1)

15	0
EP4_SIZE_LSB	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 17-56. Generic RNDIS EP4 Size Register 2 (GREP4SZR2)

15	1	0
Reserved		EP4_SIZE_MSB
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-75. Generic RNDIS EP4 Size Register 1 (GREP4SZR1) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	EP4_SIZE_LSB	RW	0	Generic RNDIS packet size. Together, GREP4SZR1 and GREP4SZR2 specify the packet size. Value: 0-FFFFh

Table 17-76. Generic RNDIS EP4 Size Register 2 (GREP4SZR2) Field Descriptions

Bit	Field	Type	Reset	Description
15-1	Reserved	R	0	Reserved
0	EP4_SIZE_MSB	RW	0	Generic RNDIS packet size. Together, GREP4SZR1 and GREP4SZR2 specify the packet size. Value: 0-1

17.3.21 Function Address Register (FADDR)

The function address register (FADDR) is shown in Figure 17-57 and described in Table 17-77.

Figure 17-57. Function Address Register (FADDR)

7	6	0
Reserved	FUNCADDR	
R-0	R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-77. Function Address Register (FADDR) Field Descriptions

Bit	Field	Type	Reset	Description
7	Reserved	R	0	Reserved.
6-0	FUNCADDR	RW	0	7_bit address of the peripheral part of the transaction. Value: 0-7Fh This register should be written with the address received through a SET_ADDRESS command, which will then be used for decoding the function address in subsequent token packets. When used in Host mode, this register should be set to the value sent in a SET_ADDRESS command during device enumeration as the address for the peripheral device.

17.3.22 Power Management Register (POWER)

The power management register (POWER) is shown in Figure 17-58 and described in Table 17-78.

Figure 17-58. Power Management Register (POWER)

7	6	5	4	3	2	1	0
ISOUPDATE	SOFTCONN	HSEN	HSMODE	RESET	RESUME	SUSPENDM	ENSUSPM
R/W-0	R/W-0	R/W-1	R-0	R-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-78. Power Management Register (POWER) Field Descriptions

Bit	Field	Type	Reset	Description
7	ISOUPDATE	RW	0	When set, the USB controller will wait for an SOF token from the time TxPktRdy is set before sending the packet. If an IN token is received before an SOF token, then a zero length data packet will be sent. This bit only affects endpoints performing Isochronous transfers. Value: 0-1
6	SOFTCONN	RW	0	If Soft Connect/Disconnect feature is enabled, then the USB D+/D- lines are enabled when this bit is set and tri-stated when this bit is cleared. Value: 0-1
5	HSEN	RW	1	When set, the USB controller will negotiate for high-speed mode when the device is reset by the hub. If not set, the device will only operate in full-speed mode. Value: 0-1
4	HSMODE	R	0	This bit is set when the USB controller has successfully negotiated for high-speed mode. Value: 0-1
3	RESET	R	0	This bit is set when Reset signaling is present on the bus. Note: this bit is read-only. Value: 0-1
2	RESUME	RW	0	Set to generate Resume signaling when the controller is in Suspend mode. The bit should be cleared after 10 ms (a maximum of 15 ms) to end Resume signaling. In Host mode, this bit is also automatically set when Resume signaling from the target is detected while the USB controller is suspended. Value: 0-1
1	SUSPENDM	RW	0	This bit is set on entry into Suspend mode. It is cleared when the interrupt register is read, or the RESUME bit is set. Value: 0-1
0	ENSUSPM	RW	0	Set to enable the SUSPENDM output. Value: 0-1

17.3.23 Interrupt Register for Endpoint 0 Plus Transmit Endpoints 1 to 4 (INTRTX)

The interrupt register for endpoint 0 plus transmit endpoints 1 to 4 (INTRTX) is shown in Figure 17-59 and described in Table 17-79.

Figure 17-59. Interrupt Register for Endpoint 0 Plus Tx Endpoints 1 to 4 (INTRTX)

15	8
Reserved	
R-0	

7	5	4	3	2	1	0
Reserved		EP4TX	EP3TX	EP2TX	EP1TX	EP0
R-0		R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 17-79. Interrupt Register for Endpoint 0 Plus Transmit Endpoints 1 to 4 (INTRTX) Field Descriptions

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0	Reserved.
4	EP4TX	R	0	Transmit Endpoint 4 interrupt active. Value: 0-1
3	EP3TX	R	0	Transmit Endpoint 3 interrupt active. Value: 0-1
2	EP2TX	R	0	Transmit Endpoint 2 interrupt active. Value: 0-1
1	EP1TX	R	0	Transmit Endpoint 1 interrupt active. Value: 0-1
0	EP0	R	0	Endpoint 0 interrupt active. Value: 0-1

17.3.24 Interrupt Register for Receive Endpoints 1 to 4 (INTRRX)

The interrupt register for receive endpoints 1 to 4 (INTRRX) is shown in Figure 17-60 and described in Table 17-80.

Figure 17-60. Interrupt Register for Receive Endpoints 1 to 4 (INTRRX)

15	8
Reserved	
R-0	

7	5	4	3	2	1	0
Reserved		EP4RX	EP3RX	EP2RX	EP1RX	Reserved
R-0		R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 17-80. Interrupt Register for Receive Endpoints 1 to 4 (INTRRX) Field Descriptions

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0	Reserved.
4	EP4RX	R	0	Receive Endpoint 4 interrupt active. Value: 0-1
3	EP3RX	R	0	Receive Endpoint 3 interrupt active. Value: 0-1
2	EP2RX	R	0	Receive Endpoint 2 interrupt active. Value: 0-1
1	EP1RX	R	0	Receive Endpoint 1 interrupt active. Value: 0-1
0	Reserved	R	0	Reserved.

17.3.25 Interrupt Enable Register for INTRTX (INTRTXE)

The interrupt enable register for INTRTX (INTRTXE) is shown in Figure 17-61 and described in Table 17-81.

Figure 17-61. Interrupt Enable Register for INTRTX (INTRTXE)

15	8
Reserved	
R-0	

7	5	4	3	2	1	0
Reserved		EP4TX	EP3TX	EP2TX	EP1TX	EP0
R-0		R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-81. Interrupt Enable Register for INTRTX (INTRTXE) Field Descriptions

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0	Reserved.
4	EP4TX	RW	1	Transmit Endpoint 4 interrupt active. Value: 0-1
3	EP3TX	RW	1	Transmit Endpoint 3 interrupt active. Value: 0-1
2	EP2TX	RW	1	Transmit Endpoint 2 interrupt active. Value: 0-1
1	EP1TX	RW	1	Transmit Endpoint 1 interrupt active. Value: 0-1
0	EP0	RW	1	Endpoint 0 interrupt active. Value: 0-1

17.3.26 Interrupt Enable Register for INTRRX (INTRRXE)

The interrupt enable register for INTRRX (INTRRXE) is shown in Figure 17-62 and described in Table 17-82.

Figure 17-62. Interrupt Enable Register for INTRRX (INTRRXE)

15	8
Reserved	
R-0	

7	5	4	3	2	1	0
Reserved		EP4RX	EP3RX	EP2RX	EP1RX	Reserved
R-0		R/W-1	R/W-1	R/W-1	R/W-1	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-82. Interrupt Enable Register for INTRRX (INTRRXE) Field Descriptions

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0	Reserved.
4	EP4RX	RW	1	Receive Endpoint 4 interrupt active. Value: 0-1
3	EP3RX	RW	1	Receive Endpoint 3 interrupt active. Value: 0-1
2	EP2RX	RW	1	Receive Endpoint 2 interrupt active. Value: 0-1
1	EP1RX	RW	1	Receive Endpoint 1 interrupt active. Value: 0-1
0	Reserved	R	0	Reserved.

17.3.27 Interrupt Register for Common USB Interrupts (INTRUSB)

The interrupt register for common USB interrupts (INTRUSB) is shown in Figure 17-63 and described in Table 17-83.

NOTE: Unless the UINT bit of CTRLR is set, do not read or write this register directly. Use the INTSRCR register instead.

Figure 17-63. Interrupt Register for Common USB Interrupts (INTRUSB)

7	6	5	4	3	2	1	0
VBUSERR	SESSREQ	DISCON	CONN	SOF	RESET_BABBLE	RESUME	SUSPEND
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 17-83. Interrupt Register for Common USB Interrupts (INTRUSB) Field Descriptions

Bit	Field	Type	Reset	Description
7	VBUSERR	R	0	Set when VBus drops below the VBus valid threshold during a session. Only valid when the USB controller is 'A' device. All active interrupts will be cleared when this register is read. Value: 0-1
6	SESSREQ	R	0	Set when session request signaling has been detected. Only valid when USB controller is 'A' device. Value: 0-1
5	DISCON	R	0	Set when a session ends. Value: 0-1
4	CONN	R	0	Set when a device connection is detected. Only valid in host mode. Value: 0-1
3	SOF	R	0	Set when a new frame starts. Value: 0-1
2	RESET_BABBLE	R	0	Set when reset signaling is detected on the bus. Value: 0-1
1	RESUME	R	0	Set when resume signaling is detected on the bus while the USB controller is in suspend mode. Value: 0-1
0	SUSPEND	R	0	Set when suspend signaling is detected on the bus. Value: 0-1

17.3.28 Interrupt Enable Register for INTRUSB (INTRUSBE)

The interrupt enable register for INTRUSB (INTRUSBE) is shown in Figure 17-64 and described in Table 17-84.

NOTE: Unless the UINT bit of CTRLR is set, do not read or write this register directly. Use the INTSETR/INTCLRR registers instead.

Figure 17-64. Interrupt Enable Register for INTRUSB (INTRUSBE)

7	6	5	4	3	2	1	0
VBUSERR	SESSREQ	DISCON	CONN	SOF	RESET_BABBLE	RESUME	SUSPEND
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-0

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-84. Interrupt Enable Register for INTRUSB (INTRUSBE) Field Descriptions

Bit	Field	Type	Reset	Description
7	VBUSERR	RW	0	Vbus error interrupt enable. Value: 0-1
6	SESSREQ	RW	0	Session request interrupt enable. Value: 0-1
5	DISCON	RW	0	Disconnect interrupt enable. Value: 0-1
4	CONN	RW	0	Connect interrupt enable. Value: 0-1
3	SOF	RW	0	Start of frame interrupt enable. Value: 0-1
2	RESET_BABBLE	RW	1	Reset interrupt enable. Value: 0-1
1	RESUME	RW	1	Resume interrupt enable. Value: 0-1
0	SUSPEND	RW	0	Suspend interrupt enable. Value: 0-1

17.3.29 Frame Number Register (FRAME)

The frame number register (FRAME) is shown in Figure 17-65 and described in Table 17-85.

Figure 17-65. Frame Number Register (FRAME)

15	11	10	0
Reserved		FRAMENUMBER	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 17-85. Frame Number Register (FRAME) Field Descriptions

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0	Reserved.
10-0	FRAMENUMBER	R	0	Last received frame number. Value: 0-7FFh

17.3.30 Index Register for Selecting the Endpoint Status and Control Registers (INDEX)

The index register for selecting the endpoint status and control registers (INDEX) is shown in Figure 17-66 and described in Table 17-86.

Figure 17-66. Index Register for Selecting the Endpoint Status and Control Registers (INDEX)

7	4	3	0
Reserved		EPSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-86. Index Register for Selecting the Endpoint Status and Control Registers (INDEX) Field Descriptions

Bit	Field	Type	Reset	Description
7-4	Reserved	R	0	Reserved.
3-0	EPSEL	RW	0	Each transmit endpoint and each receive endpoint have their own set of control/status registers. EPSEL determines which endpoint control/status registers are accessed. Before accessing an endpoint's control/status registers, the endpoint number should be written to the Index register to ensure that the correct control/status registers appear in the memory-map. Value: 0-Fh

17.3.31 Register to Enable the USB 2.0 Test Modes (TESTMODE)

The register to enable the USB 2.0 test modes (TESTMODE) is shown in Figure 17-67 and described in Table 17-87.

Figure 17-67. Register to Enable the USB 2.0 Test Modes (TESTMODE)

7	6	5	4	3	2	1	0
FORCE_HOST	FIFO_ACCESS	FORCE_FS	FORCE_HS	TEST_PACKET	TEST_K	TEST_J	TEST_SE0_NAK
R/W-0	W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; W = Write only; -*n* = value after reset

Table 17-87. Register to Enable the USB 2.0 Test Modes (TESTMODE) Field Descriptions

Bit	Field	Type	Reset	Description
7	FORCE_HOST	RW	0	Set this bit to forcibly put the USB controller into Host mode when SESSION bit is set, regardless of whether it is connected to any peripheral. The controller remains in Host mode until the Session bit is cleared, even if a device is disconnected. And if the FORCE_HOST but remains set, it will re-enter Host mode next time the SESSION bit is set. The operating speed is determined using the FORCE_HS and FORCE_FS bits. Value: 0-1
6	FIFO_ACCESS	W	0	Set this bit to transfer the packet in EP0 Tx FIFO to EP0 Receive FIFO. It is cleared automatically. Value: 0-1
5	FORCE_FS	RW	0	Set this bit to force the USB controller into full-speed mode when it receives a USB reset. Value: 0-1
4	FORCE_HS	RW	0	Set this bit to force the USB controller into high-speed mode when it receives a USB reset. Value: 0-1
3	TEST_PACKET	RW	0	Set this bit to enter the Test_Packet test mode. In this mode, the USB controller repetitively transmits a 53-byte test packet on the bus, the form of which is defined in the Universal Serial Bus Specification Revision 2.0. Note: The test packet has a fixed format and must be loaded into the Endpoint 0 FIFO before the test mode is entered. Value: 0-1
2	TEST_K	RW	0	Set this bit to enter the Test_K test mode. In this mode, the USB controller transmits a continuous K on the bus. Value: 0-1
1	TEST_J	RW	0	Set this bit to enter the Test_J test mode. In this mode, the USB controller transmits a continuous J on the bus. Value: 0-1
0	TEST_SE0_NAK	RW	0	Set this bit to enter the Test_SE0_NAK test mode. In this mode, the USB controller remains in high-speed mode, but responds to any valid IN token with a NAK. Value: 0-1

17.3.32 Maximum Packet Size for Peripheral/Host Transmit Endpoint (TXMAXP)

The maximum packet size for peripheral/host transmit endpoint (TXMAXP) is shown in Figure 17-68 and described in Table 17-88.

Figure 17-68. Maximum Packet Size for Peripheral/Host Transmit Endpoint (TXMAXP)

15	11	10	0
Reserved		MAXPAYLOAD	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-88. Maximum Packet Size for Peripheral/Host Transmit Endpoint (TXMAXP) Field Descriptions

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0	Reserved.
10-0	MAXPAYLOAD	RW	0	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 and high-speed operations. The value written to this register should match the wMaxPacketSize field of the Standard Endpoint Descriptor for the associated endpoint. A mismatch could cause unexpected results. Value: 0-FFh

17.3.33 Control Status Register for Peripheral Endpoint 0 (PERI_CSR0)

The control status register for peripheral endpoint 0 (PERI_CSR0) is shown in Figure 17-69 and described in Table 17-89.

Figure 17-69. Control Status Register for Peripheral Endpoint 0 (PERI_CSR0)

15						9	8
Reserved							FLUSHFIFO
R-0							W-0
7	6	5	4	3	2	1	0
SERV_SETUPEND	SERV_RXPKTRDY	SENDSTALL	SETUPEND	DATAEND	SENTSTALL	TXPKTRDY	RXPKTRDY
W-0	W-0	W-0	R-0	W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

Table 17-89. Control Status Register for Peripheral Endpoint 0 (PERI_CSR0) Field Descriptions

Bit	Field	Type	Reset	Description
15-9	Reserved	R	0	Reserved.
8	FLUSHFIFO	W	0	Set this bit to flush the next packet to be transmitted/read from the Endpoint 0 FIFO. The FIFO pointer is reset and the TXPKTRDY/RXPKTRDY bit is cleared. Value: 0-1 Note: FLUSHFIFO has no effect unless TXPKTRDY/RXPKTRDY is set.
7	SERV_SETUPEND	W	0	Set this bit to clear the SETUPEND bit. It is cleared automatically. Value: 0-1
6	SERV_RXPKTRDY	W	0	Set this bit to clear the RXPKTRDY bit. It is cleared automatically. Value: 0-1
5	SENDSTALL	W	0	Set this bit to terminate the current transaction. The STALL handshake will be transmitted and then this bit will be cleared automatically.Value: 0-1
4	SETUPEND	R	0	This bit will be set when a control transaction ends before the DATAEND bit has been set. An interrupt will be generated, and the FIFO will be flushed at this time. The bit is cleared by the writing a 1 to the SERV_SETUPEND bit. Value: 0-1
3	DATAEND	W	0	Set this bit to 1: 1. When setting TXPKTRDY for the last data packet. 2. When clearing RXPKTRDY after unloading the last data packet. 3. When setting TXPKTRDY for a zero length data packet. It is cleared automatically. Value: 0-1
2	SENTSTALL	RW	0	This bit is set when a STALL handshake is transmitted. This bit should be cleared. Value: 0-1
1	TXPKTRDY	RW	0	Set this bit after loading a data packet into the FIFO. It is cleared automatically when the data packet has been transmitted. An interrupt is generated (if enabled) when the bit is cleared. Value: 0-1
0	RXPKTRDY	R	0	This bit is set when a data packet has been received. An interrupt is generated when this bit is set. This bit is cleared by setting the SERV_RXPKTRDY bit. Value: 0-1

17.3.34 Control Status Register for Peripheral Transmit Endpoint (PERI_TXCSR)

The control status register for peripheral transmit endpoint (PERI_TXCSR) is shown in Figure 17-70 and described in Table 17-90.

Figure 17-70. Control Status Register for Peripheral Transmit Endpoint (PERI_TXCSR)

15	14	13	12	11	10	9 7
AUTOSET	ISO	MODE	DMAEN	FRCDATATOG	DMAMODE	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

6	5	4	3	2	1	0
CLRDATATOG	SENTSTALL	SENDSTALL	FLUSHFIFO	UNDERRUN	FIFONOTEMPTY	TXPKTRDY
W-0	R/W-0	R/W-0	W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

Table 17-90. Control Status Register for Peripheral Transmit Endpoint (PERI_TXCSR) Field Descriptions

Bit	Field	Type	Reset	Description
15	AUTOSET	RW	0	DMA Mode: The CPU needs to set the AUTOSET bit prior to enabling the Tx DMA. Value: 0 CPU Mode: If the CPU sets the AUTOSET bit, the TXPKTRDY bit will be automatically set when data of the maximum packet size (value in TXMAXP) is loaded into the Tx FIFO. If a packet of less than the maximum packet size is loaded, then the TXPKTRDY bit will have to be set manually. Value: 1
14	ISO	RW	0	Set this bit to enable the Tx endpoint for Isochronous transfers, and clear it to enable the Tx endpoint for Bulk or Interrupt transfers. Value: 0-1
13	MODE	RW	0	Set this bit to enable the endpoint direction as Tx, and clear the bit to enable it as Rx. Value: 0-1 Note: This bit has any effect only where the same endpoint FIFO is used for both Transmit and Receive transactions.
12	DMAEN	RW	0	Set this bit to enable the DMA request for the Tx endpoint. Value: 0-1
11	FRCDATATOG	RW	0	Set this bit to force the endpoint data toggle to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This can be used by Interrupt Tx endpoints that are used to communicate rate feedback for Isochronous endpoints. Value: 0-1
10	DMAMODE	RW	0	Set to 1 when DMA is enabled and EP interrupt is not needed for each packet transmission. Value: 0-1
9-7	Reserved	R	0	Reserved.
6	CLRDATATOG	W	0	Write a 1 to this bit to reset the endpoint data toggle to 0. Value: 0-1
5	SENTSTALL	RW	0	This bit is set automatically when a STALL handshake is transmitted. The FIFO is flushed and the TXPKTRDY bit is cleared. You should clear this bit. Value: 0-1
4	SENDSTALL	RW	0	Write a 1 to this bit to issue a STALL handshake to an IN token. Clear this bit to terminate the stall condition. Value: 0-1 Note: This bit has no effect where the endpoint is being used for Isochronous transfers.
3	FLUSHFIFO	W	0	Write a 1 to this bit to flush the next packet to be transmitted from the endpoint Tx FIFO. The FIFO pointer is reset and the TXPKTRDY bit is cleared. Value: 0-1 Note: FlushFIFO has no effect unless the TXPKTRDY bit is set. Also note that, if the FIFO is double-buffered, FlushFIFO may need to be set twice to completely clear the FIFO.
2	UNDERRUN	RW	0	This bit is set automatically if an IN token is received when TXPKTRDY is not set. You should clear this bit. Value: 0-1
1	FIFONOTEMPTY	RW	0	This bit is set when there is at least 1 packet in the Tx FIFO. You should clear this bit. Value: 0-1
0	TXPKTRDY	RW	0	Set this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. An interrupt is generated (if enabled) when the bit is cleared. Value: 0-1

17.3.35 Maximum Packet Size for Peripheral Receive Endpoint (RXMAXP)

The maximum packet size for peripheral receive endpoint (RXMAXP) is shown in Figure 17-71 and described in Table 17-91.

Figure 17-71. Maximum Packet Size for Peripheral Receive Endpoint (RXMAXP)

15	11	10	0
Reserved		MAXPAYLOAD	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-91. Maximum Packet Size for Peripheral Receive Endpoint (RXMAXP) Field Descriptions

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0	Reserved.
10-0	MAXPAYLOAD	RW	0	Defines the maximum amount of data that can be transferred through the selected Receive endpoint in a single frame/microframe (high-speed transfers). 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 and high-speed operations. The value written to this register should match the wMaxPacketSize field of the Standard Endpoint Descriptor for the associated endpoint. A mismatch could cause unexpected results. Value: 0-FFh

17.3.36 Control Status Register for Peripheral Receive Endpoint (PERI_RXCSR)

The control status register for peripheral receive endpoint (PERI_RXCSR) is shown in Figure 17-72 and described in Table 17-92.

Figure 17-72. Control Status Register for Peripheral Receive Endpoint (PERI_RXCSR)

15	14	13	12	11	10		8
AUTOCLEAR	ISO	DMAEN	DISNYET	DMAMODE	Reserved		
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0		
7	6	5	4	3	2	1	0
CLRDATATOG	SENTSTALL	SENDSTALL	FLUSHFIFO	DATAERROR	OVERRUN	FIFOFULL	RXPKTRDY
W-0	R/W-0	R/W-0	W-0	R-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

Table 17-92. Control Status Register for Peripheral Receive Endpoint (PERI_RXCSR) Field Descriptions

Bit	Field	Type	Reset	Description
15	AUTOCLEAR	RW	0	DMA Mode: The CPU sets the AUTOCLEAR bit prior to enabling the Rx DMA. Value: 0 CPU Mode: If the CPU sets the AUTOCLEAR bit, then the RXPKTRDY bit will be automatically cleared when a packet of RXMAXP bytes has been unloaded from the Receive FIFO. When packets of less than the maximum packet size are unloaded, RXPKTRDY will have to be cleared manually. Value: 1
14	ISO	RW	0	Set this bit to enable the Receive endpoint for Isochronous transfers, and clear it to enable the Receive endpoint for Bulk/Interrupt transfers. Value: 0-1
13	DMAEN	RW	0	Set this bit to enable the DMA request for the Receive endpoints. Value: 0-1
12	DISNYET	RW	0	DISNYET: Applies only for Bulk/Interrupt Transactions: The CPU sets this bit to disable the sending of NYET handshakes. When set, all successfully received Rx packets are ACK'd including at the point at which the FIFO becomes full. Value: 0 Note: This bit only has any effect in high-speed mode, in which mode it should be set for all Interrupt endpoints. PID_ERROR: Applies only for ISO Transactions: The core sets this bit to indicate a PID error in the received packet. Value: 1
11	DMAMODE	RW	0	The CPU clears the DMAMODE bit prior to enabling the Rx DMA. Value: 0-1
10-8	Reserved	R	0	Reserved.
7	CLRDATATOG	W	0	Write a 1 to this bit to reset the endpoint data toggle to 0. Value: 0-1
6	SENTSTALL	RW	0	This bit is set when a STALL handshake is transmitted. The FIFO is flushed and the TXPKTRDY bit is cleared. You should clear this bit. Value: 0-1
5	SENDSTALL	RW	0	Write a 1 to this bit to issue a STALL handshake. Clear this bit to terminate the stall condition. Value: 0-1 Note: This bit has no effect where the endpoint is being used for Isochronous transfers.
4	FLUSHFIFO	W	0	Write 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 RXPKTRDY bit is cleared. Value: 0-1 Note: FLUSHFIFO has no effect unless RXPKTRDY is set. Also note that, if the FIFO is double-buffered, FLUSHFIFO may need to be set twice to completely clear the FIFO.
3	DATAERROR	R	0	This bit is set when RXPKTRDY is set if the data packet has a CRC or bit-stuff error. It is cleared when RXPKTRDY is cleared. Value: 0-1 Note: This bit is only valid when the endpoint is operating in ISO mode. In Bulk mode, it always returns zero.
2	OVERRUN	RW	0	This bit is set if an OUT packet cannot be loaded into the Receive FIFO. You should clear this bit. Value: 0-1 Note: This bit is only valid when the endpoint is operating in ISO mode. In Bulk mode, it always returns zero.
1	FIFOFULL	R	0	This bit is set when no more packets can be loaded into the Receive FIFO. Value: 0-1
0	RXPKTRDY	RW	0	This bit is set when a data packet has been received. You should clear this bit when the packet has been unloaded from the Receive FIFO. An interrupt is generated when the bit is set. Value: 0-1

17.3.37 Count 0 Register (COUNT0)

The count 0 register (COUNT0) is shown in Figure 17-73 and described in Table 17-93.

Figure 17-73. Count 0 Register (COUNT0)

15	7	6	0
Reserved		EP0RXCOUNT	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 17-93. Count 0 Register (COUNT0) Field Descriptions

Bit	Field	Type	Reset	Description
15-7	Reserved	R	0	Reserved.
6-0	EP0RXCOUNT	R	0	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 RXPKTRDY of PERI_CSR0 is set. Value: 0-7Fh

17.3.38 Receive Count Register (RXCOUNT)

The receive count register (RXCOUNT) is shown in Figure 17-74 and described in Table 17-94.

Figure 17-74. Receive Count Register (RXCOUNT)

15	13	12	0
Reserved		EPRXCOUNT	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 17-94. Receive Count Register (RXCOUNT) Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0	Reserved.
12-0	EPRXCOUNT	R	0	Holds the number of received data bytes in the packet in the Receive FIFO. The value returned changes as the contents of the FIFO change and is only valid while RXPKTRDY of PERI_RXCSR or HOST_RXCSR is set. Value: 0-1FFFh

17.3.39 Configuration Data Register (CONFIGDATA)

The configuration data register (CONFIGDATA) is shown in Figure 17-75 and described in Table 17-95.

Figure 17-75. Configuration Data Register (CONFIGDATA)

7	6	5	4	3	2	1	0
MPRXE	MPTXE	BIGENDIAN	HBRXE	HBTXE	DYNFIFO	SOFTCONE	UTMIDATAWIDTH
R-0	R-0	R-0	R-0	R-0	R-1	R-1	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 17-95. Configuration Data Register (CONFIGDATA) Field Descriptions

Bit	Field	Type	Reset	Description
7	MPRXE	R	0	Indicates automatic amalgamation of bulk packets. 0 = Automatic amalgamation of bulk packets is not selected. 1 = Automatic amalgamation of bulk packets is selected.
6	MPTXE	R	0	Indicates automatic splitting of bulk packets. 0 = Automatic splitting of bulk packets is not selected. 1 = Automatic splitting of bulk packets is selected.
5	BIGENDIAN	R	0	Indicates endian ordering. 0 = Little-endian ordering is selected. 1 = Big-endian ordering is selected.
4	HBRXE	R	0	Indicates high-bandwidth Rx ISO endpoint support. 0 = High-bandwidth Rx ISO endpoint support is not selected. 1 = High-bandwidth Rx ISO endpoint support is selected.
3	HBTXE	R	0	Indicates high-bandwidth Tx ISO endpoint support. 0 = High-bandwidth Tx ISO endpoint support is not selected. 1 = High-bandwidth Tx ISO endpoint support is selected.
2	DYNFIFO	R	1	Indicates dynamic FIFO sizing. 0 = Dynamic FIFO sizing option is not selected. 1 = Dynamic FIFO sizing option is selected.
1	SOFTCONE	R	1	Indicates soft connect/disconnect. 0 = Soft connect/disconnect option is not selected. 1 = Soft connect/disconnect option is selected.
0	UTMIDATAWIDTH	R	0	Indicates selected UTMI data width. 0 = 8 bits. 1 = 16 bits.

17.3.40 Transmit and Receive FIFO Registers for Endpoint 0 (FIFO0R1 and FIFO0R2)

The transmit and receive FIFO register 1 for endpoint 0 (FIFO0R1) is shown in Figure 17-76 and described in Table 17-96. The transmit and receive FIFO register 2 for endpoint 0 (FIFO0R2) is shown in Figure 17-77 and described in Table 17-97.

Figure 17-76. Transmit and Receive FIFO Register 1 for Endpoint 0 (FIFO0R1)

15	0
DATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Figure 17-77. Transmit and Receive FIFO Register 2 for Endpoint 0 (FIFO0R2)

15	0
DATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-96. Transmit and Receive FIFO Register 1 for Endpoint 0 (FIFO0R1) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	RW	0	Writing to this address loads data into the Transmit FIFO for the corresponding endpoint. Reading from these addresses unloads data from the Receive FIFO for the corresponding endpoint. Value: 0-FFFFh

Table 17-97. Transmit and Receive FIFO Register 2 for Endpoint 0 (FIFO0R2) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	RW	0	Writing to this address loads data into the Transmit FIFO for the corresponding endpoint. Reading from these addresses unloads data from the Receive FIFO for the corresponding endpoint. Value: 0-FFFFh

17.3.41 Transmit and Receive FIFO Registers for Endpoint 1 (FIFO1R1 and FIFO1R2)

The transmit and receive FIFO register 1 for endpoint 1 (FIFO1R1) is shown in Figure 17-78 and described in Table 17-98. The transmit and receive FIFO register 2 for endpoint 1 (FIFO1R2) is shown in Figure 17-79 and described in Table 17-99.

Figure 17-78. Transmit and Receive FIFO Register 1 for Endpoint 1 (FIFO1R1)

15	0
DATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Figure 17-79. Transmit and Receive FIFO Register 2 for Endpoint 1 (FIFO1R2)

15	0
DATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-98. Transmit and Receive FIFO Register 1 for Endpoint 1 (FIFO1R1) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	RW	0	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. Value: 0-FFFFh

Table 17-99. Transmit and Receive FIFO Register 2 for Endpoint 1 (FIFO1R2) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	RW	0	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. Value: 0-FFFFh

17.3.42 Transmit and Receive FIFO Registers for Endpoint 2 (FIFO2R1 and FIFO2R2)

The transmit and receive FIFO register 1 for endpoint 2 (FIFO2R1) is shown in Figure 17-80 and described in Table 17-100. The transmit and receive FIFO register 2 for endpoint 2 (FIFO2R2) is shown in Figure 17-81 and described in Table 17-101.

Figure 17-80. Transmit and Receive FIFO Register 1 for Endpoint 2 (FIFO2R1)

15	0
DATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Figure 17-81. Transmit and Receive FIFO Register 2 for Endpoint 2 (FIFO2R2)

15	0
DATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-100. Transmit and Receive FIFO Register 1 for Endpoint 2 (FIFO2R1) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	RW	0	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. Value: 0-FFFFh

Table 17-101. Transmit and Receive FIFO Register 2 for Endpoint 2 (FIFO2R2) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	RW	0	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. Value: 0-FFFFh

17.3.43 Transmit and Receive FIFO Registers for Endpoint 3 (FIFO3R1 and FIFO3R2)

The transmit and receive FIFO register 1 for endpoint 3 (FIFO3R1) is shown in Figure 17-82 and described in Table 17-102. The transmit and receive FIFO register 2 for endpoint 3 (FIFO3R2) is shown in Figure 17-83 and described in Table 17-103.

Figure 17-82. Transmit and Receive FIFO Register 1 for Endpoint 3 (FIFO3R1)

15	0
DATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Figure 17-83. Transmit and Receive FIFO Register 2 for Endpoint 3 (FIFO3R2)

15	0
DATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-102. Transmit and Receive FIFO Register 1 for Endpoint 3 (FIFO3R1) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	RW	0	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. Value: 0-FFFFh

Table 17-103. Transmit and Receive FIFO Register 2 for Endpoint 3 (FIFO3R2) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	RW	0	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. Value: 0-FFFFh

17.3.44 Transmit and Receive FIFO Registers for Endpoint 4 (FIFO4R1 and FIFO4R2)

The transmit and receive FIFO register 1 for endpoint 4 (FIFO4R1) is shown in Figure 17-84 and described in Table 17-104. The transmit and receive FIFO register 2 for endpoint 4 (FIFO4R2) is shown in Figure 17-85 and described in Table 17-105.

Figure 17-84. Transmit and Receive FIFO Register 1 for Endpoint 4 (FIFO4R1)

15	0
DATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Figure 17-85. Transmit and Receive FIFO Register 2 for Endpoint 4 (FIFO4R2)

15	0
DATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-104. Transmit and Receive FIFO Register 1 for Endpoint 4 (FIFO4R1) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	RW	0	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. Value: 0-FFFFh

Table 17-105. Transmit and Receive FIFO Register 2 for Endpoint 4 (FIFO4R2) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DATA	RW	0	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. Value: 0-FFFFh

17.3.45 Device Control Register (DEVCTL)

The device control register (DEVCTL) is shown in Figure 17-86 and described in Table 17-106.

Figure 17-86. Device Control Register (DEVCTL)

7	6	5	4	3	2	1	0
BDEVICE	FSDEV	LSDEV	VBUS		HOSTMODE	Reserved	SESSION
R-0	R-0	R-0	R-0		R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 17-106. Device Control Register (DEVCTL) Field Descriptions

Bit	Field	Type	Reset	Description
7	BDEVICE	R	0	This read-only bit indicates whether the USB controller is operating as the 'A' device or the 'B' device. 0 = A device. 1 = B device. Only valid while a session is in progress.
6	FSDEV	R	0	This read-only bit is set when a full-speed or high-speed device has been detected being connected to the port (high-speed devices are distinguished from full-speed by checking for high-speed chirps when the device is reset). Only valid in Host mode. Host mode is not supported on the device. Value: 0-1
5	LSDEV	R	0	This read-only bit is set when a low-speed device has been detected being connected to the port. Only valid in Host mode. Host mode is not supported on the device. Value: 0-1
	VBUS	R	0	These read-only bits encode the current VBus level as follows: 0 = Below Session End. 1h = Above Session End, below AValid. 2h = Above AValid, below VBusValid. 3h = Above VBusValid.
2	HOSTMODE	R	0	This read-only bit is set when the USB controller is acting as a Host. Host mode is not supported on the device. Value: 0-1
1	Reserved	RW	0	Reserved.
0	SESSION	RW	0	When operating as an 'A' device, you must set or clear this bit start or end a session. When operating as a 'B' device, this bit is set/cleared by the USB controller when a session starts/ends. You must also set this bit to initiate the Session Request Protocol. When the USB controller is in Suspend mode, you may clear the bit to perform a software disconnect. A special software routine is required to perform SRP. Details will be made available in a later document version. Value: 0-1

17.3.46 Transmit Endpoint FIFO Size (TXFIFOSZ)

Section 17.2.7 describes dynamically setting endpoint FIFO sizes. The option of dynamically setting endpoint FIFO sizes only applies to Endpoints 1-4. The Endpoint 0 FIFO has a fixed size (64 bytes) and a fixed location (start address 0). It is the responsibility of the firmware to ensure that all the Tx and Rx endpoints that are active in the current USB configuration have a block of RAM assigned exclusively to that endpoint. The RAM must be at least as large as the maximum packet size set for that endpoint.

The transmit endpoint FIFO size (TXFIFOSZ) is shown in Figure 17-87 and described in Table 17-107.

Figure 17-87. Transmit Endpoint FIFO Size (TXFIFOSZ)

7	5	4	3	0
Reserved		DPB	SZ	
R-0		R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-107. Transmit Endpoint FIFO Size (TXFIFOSZ) Field Descriptions

Bit	Field	Type	Reset	Description
7-5	Reserved	R	0	Reserved.
4	DPB	RW	0	Double packet buffering enable. 0 = Single packet buffering is supported. 1 = Double packet buffering is enabled.
3-0	SZ	R	0	Maximum packet size to be allowed (before any splitting within the FIFO of Bulk packets prior to transmission). If m = SZ, the FIFO size is calculated as $2^{(m+3)}$ for single packet buffering and $2^{(m+4)}$ for dual packet buffering. Value: 0-Fh

17.3.47 Receive Endpoint FIFO Size (RXFIFOSZ)

Section 17.2.7 describes dynamically setting endpoint FIFO sizes. The option of dynamically setting endpoint FIFO sizes only applies to Endpoints 1-4. The Endpoint 0 FIFO has a fixed size (64 bytes) and a fixed location (start address 0). It is the responsibility of the firmware to ensure that all the Tx and Rx endpoints that are active in the current USB configuration have a block of RAM assigned exclusively to that endpoint. The RAM must be at least as large as the maximum packet size set for that endpoint.

The receive endpoint FIFO size (RXFIFOSZ) is shown in Figure 17-88 and described in Table 17-108.

Figure 17-88. Receive Endpoint FIFO Size (RXFIFOSZ)

7	5	4	3	0
Reserved		DPB	SZ	
R-0		R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-108. Receive Endpoint FIFO Size (RXFIFOSZ) Field Descriptions

Bit	Field	Type	Reset	Description
7-5	Reserved	R	0	Reserved.
4	DPB	RW	0	Double packet buffering enable. 0 = Single packet buffering is supported. 1 = Double packet buffering is enabled.
3-0	SZ	R	0	Maximum packet size to be allowed (before any splitting within the FIFO of Bulk packets prior to transmission). If m = SZ, the FIFO size is calculated as $2^{(m+3)}$ for single packet buffering and $2^{(m+4)}$ for dual packet buffering. Value: 0-Fh

17.3.48 Transmit Endpoint FIFO Address (TXFIFOADDR)

The transmit endpoint FIFO address (TXFIFOADDR) is shown in Figure 17-89 and described in Table 17-109.

Figure 17-89. Transmit Endpoint FIFO Address (TXFIFOADDR)

15	13	12	0
Reserved		ADDR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-109. Transmit Endpoint FIFO Address (TXFIFOADDR) Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0	Reserved.
12-0	ADDR	RW	0	Start Address of endpoint FIFO in units of 8 bytes. If m = ADDR, then the start address is 8 × m. Value: 0-1FFFh

17.3.49 Hardware Version Register (HWVERS)

The hardware version register (HWVERS) contains the RTL major and minor version numbers for the USB 2.0 controller module. The RTL version number is REVMAJ.REVMIN. The HWVERS is shown in Figure 17-90 and described in Table 17-110.

Figure 17-90. Hardware Version Register (HWVERS)

15	14	10	9	0
RC	REVMAJ		REVMIN	
R-0	R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 17-110. Hardware Version Register (HWVERS) Field Descriptions

Bit	Field	Type	Reset	Description
15	RC	R	0	Set to 1 if RTL is used from a Release Candidate, rather than from a full release of the core. Value: 0-1
14-10	REVMAJ	R	0	Major version of RTL. Range is 0-3.1. Value: 0-1Fh
9-0	REVMIN	R	0	Minor version of RTL. Range is 0-999. Value: 0-3E7h

17.3.50 Receive Endpoint FIFO Address (RXFIFOADDR)

The receive endpoint FIFO address (RXFIFOADDR) is shown in Figure 17-91 and described in Table 17-111.

Figure 17-91. Receive Endpoint FIFO Address (RXFIFOADDR)

15 13	12 0
Reserved	ADDR
R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-111. Receive Endpoint FIFO Address (RXFIFOADDR) Field Descriptions

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0	Reserved.
12-0	ADDR	RW	0	Start Address of endpoint FIFO in units of 8 bytes. If m = ADDR, then the start address is 8 × m. Value: 0-1FFFh

17.3.51 CDMA Revision Identification Registers (DMAREVID1 and DMAREVID2)

The CDMA revision identification registers (DMAREVID1 and DMAREVID2) contain the revision for the module. The DMAREVID1 is shown in Figure 17-92 and described in Table 17-112. The DMAREVID2 is shown in Figure 17-93 and described in Table 17-113.

Figure 17-92. CDMA Revision Identification Register 1 (DMAREVID1)

15 0
REV_LSB
R-1900h

LEGEND: R = Read only; -*n* = value after reset

Figure 17-93. CDMA Revision Identification Register 2 (DMAREVID2)

15 0
REV_MSB
R-0053h

LEGEND: R = Read only; -*n* = value after reset

Table 17-112. CDMA Revision Identification Register 1 (DMAREVID1) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	REV_LSB	R	1900h	Revision ID of the CPPI DMA (CDMA) module. Least significant bits. Value: 0-FFFFh

Table 17-113. CDMA Revision Identification Register 2 (DMAREVID2) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	REV_MSB	R	0053h	Revision ID of the CPPI DMA (CDMA) module. Most significant bits. Value: 0-FFFFh

17.3.52 CDMA Teardown Free Descriptor Queue Control Register (TDFDQ)

The CDMA teardown free descriptor queue control register (TDFDQ) is used to inform the DMA of the location in memory or descriptor array which is to be used for signaling of a teardown complete for each transmit and receive channel. The CDMA teardown free descriptor queue control register (TDFDQ) is shown in Figure 17-94 and described in Table 17-114.

Figure 17-94. CDMA Teardown Free Descriptor Queue Control Register (TDFDQ)

15 14	13 12	11 0
Reserved	TD_DESC_QMGR	TD_DESC_QNUM
R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-114. CDMA Teardown Free Descriptor Queue Control Register (TDFDQ) Field Descriptions

Bit	Field	Type	Reset	Description
15-14	Reserved	R	0	Reserved.
13-12	TD_DESC_QMGR	RW	0	Controls which of the four queue managers the DMA accesses to allocate a channel teardown descriptor from the teardown descriptor queue. Value: 0-3h
11-0	TD_DESC_QNUM	RW	0	Controls which of the 2K queues in the indicated queue manager should be read to allocate the channel teardown descriptors. Value: 0-FFFh

17.3.53 CDMA Emulation Control Register (DMAEMU)

The CDMA emulation controls the behavior of the DMA when an emulation suspend signal is asserted. The CDMA emulation control register (DMAEMU) is shown in Figure 17-95 and described in Table 17-115.

Figure 17-95. CDMA Emulation Control Register (DMAEMU)

15 2	1	0
Reserved	SOFT	FREE
R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-115. CDMA Emulation Control Register (DMAEMU) Field Descriptions

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0	Reserved.
1	SOFT	RW	0	Value : 0-1
0	FREE	RW	0	Value : 0-1

17.3.54 CDMA Transmit Channel n Global Configuration Registers (TXGCR1[n] and TXGCR2[n])

The transmit channel *n* configuration registers (TXGCR2[*n*] and TXGCR1[*n*]) initialize the behavior of each of the transmit DMA channels. There are four configuration register pairs, one for each transmit DMA channel.

The transmit channel *n* configuration registers TXGCR1[*n*]) and (TXGCR2[*n*] are shown in Figure 17-96 and Figure 17-97and described in Table 17-116 and Table 17-117. .

Figure 17-96. CDMA Transmit Channel *n* Global Configuration Register 1 (TXGCR1[*n*])

15 14	13 12	11 0
Reserved	TX_DEFAULT_QMGR	TX_DEFAULT_QNUM
R-0	W-0	W-0

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

Figure 17-97. CDMA Transmit Channel *n* Global Configuration Register 2 (TXGCR2[*n*])

15	14	13 0
TX_ENABLE	TX_TEARDOWN	Reserved
R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

Table 17-116. CDMA Transmit Channel *n* Global Configuration Register 1 (TXGCR1[*n*]) Field Descriptions

Bit	Field	Type	Reset	Description
15-14	Reserved	RW	0	Reserved.
13-12	TX_DEFAULT_QMGR	RW	0	Controls the default queue manager number that is used to queue teardown descriptors back to the host. Value: 0-3h
11-0	TX_DEFAULT_QNUM	RW	0	Controls the default queue number within the selected queue manager onto which teardown descriptors are queued back to the host. Value: 0-FFFh

Table 17-117. CDMA Transmit Channel *n* Global Configuration Register 2 (TXGCR2[*n*]) Field Descriptions

Bit	Field	Type	Reset	Description
15	TX_ENABLE	RW	0	Channel control. The TX_ENABLE field is cleared after a channel teardown is complete. 0 = Disables channel. 1 = Enables channel.
14	TX_TEARDOWN	RW	0	Setting this bit requests the channel to be torn down. The TX_TEARDOWN field remains set after a channel teardown is complete. Value: 0-1
13-0	Reserved	R	0	Reserved.

17.3.55 CDMA Receive Channel n Global Configuration Registers (RXGCR1[n] and RXGCR2[n])

The receive channel *n* global configuration registers (RXGCR1[*n*] and RXGCR2[*n*]) initialize the global (non-descriptor-type specific) behavior of each of the receive DMA channels. There are four configuration register pairs, one for each receive DMA channel. If the enable bit is being set, the receive channel *n* global configuration register should only be written after all of the other receive configuration registers have been initialized.

The receive channel *n* global configuration registers (RXGCR1[*n*] and RXGCR2[*n*]) are shown in Figure 17-98 and Figure 17-99 and are described in Table 17-118 and Table 17-119.

Figure 17-98. CDMA Receive Channel *n* Global Configuration Register 1 (RXGCR1[*n*])

15 14	13 12	11 0
RX_DEFAULT_DESC_TYPE	RX_DEFAULT_RQ_QMGR	RX_DEFAULT_RQ_QNUM
R-0	W-0	W-0

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

Figure 17-99. CDMA Receive Channel *n* Global Configuration Register 2 (RXGCR2[*n*])

15	14	13 9	8	7 0
RX_ENABLE	RX_TEARDOWN	Reserved	RX_ERROR_HANDLING	RX_SOP_OFFSET
R/W-0	R/W-0	R-0	W-0	W-0

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

Table 17-118. CDMA Receive Channel *n* Global Configuration Register 1 (RXGCR1[*n*]) Field Descriptions

Bit	Field	Type	Reset	Description
15-14	RX_DEFAULT_DESC_TYPE	R	0	Indicates the default descriptor type to use. The actual descriptor type that is used for reception can be overridden by information provided in the CPPI FIFO data block. 0 = Reserved. 1h = Host. 2h = Reserved. 3h = Reserved.
13-12	RX_DEFAULT_RQ_QMGR	W	0	Indicates the default receive queue manager that this channel should use. The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block. Value: 0-3h
11-0	RX_DEFAULT_RQ_QNUM	W	0	Indicates the default receive queue that this channel should use. The actual receive queue that is used for reception can be overridden by information provided in the CPPI FIFO data block. Value: 0-FFFh

Table 17-119. CDMA Receive Channel *n* Global Configuration Register 2 (RXGCR2[*n*]) Field Descriptions

Bit	Field	Type	Reset	Description
15	RX_ENABLE	RW	0	Channel control. Field is cleared after a channel teardown is complete. 0 = Disables channel. 1 = Enables channel.
14	RX_TEARDOWN	RW	0	Indicates whether a receive operation is complete. Field should be cleared when a channel is initialized. Field is set after a channel teardown is complete. Value: 0-1
13-9	Reserved	R	0	Reserved.

Table 17-119. CDMA Receive Channel *n* Global Configuration Register 2 (RXGCR2[*n*]) Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8	RX_ERROR_HANDLING	W	0	Controls the error handling mode for the channel and is only used when channel errors (that is, descriptor or buffer starvation occur): 0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to. 1 = Starvation errors result in subsequent retry of the descriptor allocation operation. In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface. This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.
7-0	RX_SOP_OFFSET	W	0	Specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload. This value must be less than the minimum size of a buffer in the system. Value: 0–FFh

17.3.56 CDMA Receive Channel n Host Packet Configuration Registers A (RXHPCR1A[n] and RXHPCR2A[n])

The receive channel *n* host packet configuration registers A (RXHPCR1A[n] and RXHPCR2A[n]) initialize the behavior of each of the receive DMA channels for reception of host type packets. There are four configuration A registers, one for each receive DMA channel.

The receive channel *n* host packet configuration register 1 A (RXHPCR1A[n]) are shown in Figure 17-100 and described in Table 17-120. The receive channel *n* host packet configuration register 2 A (RXHPCR2A[n]) is shown in Figure 17-101 and described in Table 17-121.

Figure 17-100. Receive Channel *n* Host Packet Configuration Register 1 A (RXHPCR1A[*n*])

15 14	13 12	11 0
Reserved	RX_HOST_FDQ0_QMGR	RX_HOST_FDQ0_QNUM
R-0	W-0	W-0

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Figure 17-101. Receive Channel *n* Host Packet Configuration Register 2 A (RXHPCR2A[*n*])

15 14	13 12	11 0
Reserved	RX_HOST_FDQ1_QMGR	RX_HOST_FDQ1_QNUM
R-0	W-0	W-0

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 17-120. Receive Channel *n* Host Packet Configuration Register 1 A (RXHPCR1A[*n*]) Field Descriptions

Bit	Field	Type	Reset	Description
15-14	Reserved	R	0	Reserved.
13-12	RX_HOST_FDQ0_QMGR	W	0	Specifies which buffer manager should be used for the first receive buffer in a host type packet. Value: 0-3h
11-0	RX_HOST_FDQ0_QNUM	W	0	Specifies which free descriptor/buffer pool should be used for the first receive buffer in a host type packet. Value: 0-FFFh

Table 17-121. Receive Channel *n* Host Packet Configuration Register 2 A (RXHPCR2A[*n*]) Field Descriptions

Bit	Field	Type	Reset	Description
15-14	Reserved	R	0	Reserved.
13-12	RX_HOST_FDQ1_QMGR	W	0	Specifies which buffer manager should be used for the second receive buffer in a host type packet. Value: 0-3h
11-0	RX_HOST_FDQ1_QNUM	W	0	Specifies which free descriptor/buffer pool should be used for the second receive buffer in a host type packet. Value: 0-FFFh

17.3.57 CDMA Receive Channel n Host Packet Configuration Registers B (RXHPCR1B[n] and RXHPCR2B[n])

The receive channel *n* host packet configuration registers B (RXHPCR1B[n] and RXHPCR2B[n]) initialize the behavior of each of the receive DMA channels for reception of host type packets. There are four configuration B register pairs, one for each receive DMA channel.

The receive channel *n* host packet configuration register 1 B (RXHPCR1B[*n*]) is shown in Figure 17-102 and described in Table 17-122. The receive channel *n* host packet configuration register 2 B (RXHPCR2B[*n*]) is shown in Figure 17-103 and described in Table 17-123.

Figure 17-102. Receive Channel *n* Host Packet Configuration Register 1 B (RXHPCR1B[*n*])

15	14	13	12	11	0
Reserved		RX_HOST_FDQ2_QMGR		RX_HOST_FDQ2_QNUM	
R-0		W-0		W-0	

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Figure 17-103. Receive Channel *n* Host Packet Configuration Register 2 B (RXHPCR2B[*n*])

15	14	13	12	11	0
Reserved		RX_HOST_FDQ3_QMGR		RX_HOST_FDQ3_QNUM	
R-0		W-0		W-0	

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 17-122. Receive Channel *n* Host Packet Configuration Register 1 B (RXHPCR1B[*n*]) Field Descriptions

Bit	Field	Type	Reset	Description
15-14	Reserved	R	0	Reserved.
13-12	RX_HOST_FDQ2_QMGR	W	0	Specifies which buffer manager should be used for the third receive buffer in a host type packet. Value: 0-3h
11-0	RX_HOST_FDQ2_QNUM	W	0	Specifies which free descriptor/buffer pool should be used for the third receive buffer in a host type packet. Value: 0-FFFh

Table 17-123. Receive Channel *n* Host Packet Configuration Register 2 B (RXHPCR2B[*n*]) Field Descriptions

Bit	Field	Value	Description
15-14	Reserved	0	Reserved.
13-12	RX_HOST_FDQ3_QMGR	0	Specifies which buffer manager should be used for the fourth or later receive buffer in a host type packet. Value: 0-3h
11-0	RX_HOST_FDQ3_QNUM	0	Specifies which free descriptor/buffer pool should be used for the fourth or later receive buffer in a host type packet. Value: 0-FFFh

17.3.58 CDMA Scheduler Control Register (DMA_SCHED_CTRL1 and DMA_SCHED_CTRL2)

The CDMA scheduler control registers (DMA_SCHED_CTRL1 and DMA_SCHED_CTRL2) enable the scheduler and indicate the last entry in the scheduler table. The CDMA scheduler control register 1 (DMA_SCHED_CTRL1) is shown in Figure 17-104 and described in Table 17-124. The CDMA scheduler control register 2 (DMA_SCHED_CTRL2) is shown in Figure 17-105 and described in Table 17-125.

Figure 17-104. CDMA Scheduler Control Register 1 (DMA_SCHED_CTRL1)

15	8	7	0
Reserved		LAST_ENTRY	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 17-105. CDMA Scheduler Control Register 2 (DMA_SCHED_CTRL2)

15	14	0
ENABLE	Reserved	
R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-124. CDMA Scheduler Control Register 1 (DMA_SCHED_CTRL1) Field Descriptions

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0	Reserved.
7-0	LAST_ENTRY	RW	0	Indicates the last valid entry in the scheduler table. There are 64 words in the table and there are 4 entries in each word. The table can be programmed with any integer number of entries from 1 to 256. The corresponding encoding for this field is as follows: 0 = 1 entry. 1h = 2 entries. 2h to FFh = 3 entries to 256 entries.

Table 17-125. CDMA Scheduler Control Register 2 (DMA_SCHED_CTRL2) Field Descriptions

Bit	Field	Type	Reset	Description
15	ENABLE	R	0	This is the enable bit for the scheduler and is encoded as follows: 0 = Scheduler is disabled and will no longer fetch entries from the scheduler table or pass credits to the DMA controller. 1 = Scheduler is enabled. This bit should only be set after the table has been initialized.
14-0	Reserved	RW	0	Reserved.

17.3.59 CDMA Scheduler Table Word n Registers (ENTRYLSW[n]-ENTRYMSW[n])

The CDMA scheduler table word *n* registers (ENTRYLSW[n]-ENTRYMSW[n]) provide information about the scheduler. The CDMA scheduler table word *n* registers (ENTRYLSW[*n*]) are shown in Figure 17-106 and described in Table 17-126. The CDMA scheduler table word *n* registers (ENTRYMSW[*n*]) are shown in Figure 17-107 and described in Table 17-127.

Figure 17-106. CDMA Scheduler Table Word *n* Registers (ENTRYLSW[*n*])

15	14 12	11 8	7	6 4	3 0
ENTRY1_RXTX	Reserved	ENTRY1_CHANNEL	ENTRY0_RXTX	Reserved	ENTRY0_CHANNEL
W-0	R-0	W-0	W-0	R-0	W-0

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Figure 17-107. CDMA Scheduler Table Word *n* Registers (ENTRYMSW[*n*])

15	14 12	11 8	7	6 4	3 0
ENTRY3_RXTX	Reserved	ENTRY3_CHANNEL	ENTRY2_RXTX	Reserved	ENTRY2_CHANNEL
W-0	R-0	W-0	W-0	R-0	W-0

LEGEND: R = Read only; W = Write only; -*n* = value after reset

Table 17-126. CDMA Scheduler Table Word *n* Registers (ENTRYLSW[*n*]) Field Descriptions

Bit	Field	Type	Reset	Description
15	ENTRY1_RXTX	W	0	This entry is for a transmit or a receive channel. 0 = Transmit channel 1 = Receive channel
14-12	Reserved	R	0	Reserved
11-8	ENTRY1_CHANNEL	W	0	Indicates the channel number that is to be given an opportunity to transfer data. If this is a transmit entry, the DMA will be presented with a scheduling credit for that exact transmit channel. If this is a receive entry, the DMA will be presented with a scheduling credit for the receive FIFO that is associated with this channel. For receive FIFOs which carry traffic for more than one receive DMA channel, the exact channel number that is given in the receive credit will actually be the channel number which is currently on the head element of that Rx FIFO, which is not necessarily the channel number given in the scheduler table entry. Value: 0-Fh
7	ENTRY0_RXTX	W	0	This entry is for a transmit or a receive channel. 0 = Transmit channel 1 = Receive channel
6-4	Reserved	R	0	Reserved
3-0	ENTRY0_CHANNEL	W	0	Indicates the channel number that is to be given an opportunity to transfer data. If this is a transmit entry, the DMA will be presented with a scheduling credit for that exact transmit channel. If this is a receive entry, the DMA will be presented with a scheduling credit for the receive FIFO that is associated with this channel. For receive FIFOs which carry traffic for more than one receive DMA channel, the exact channel number that is given in the receive credit will actually be the channel number which is currently on the head element of that Rx FIFO, which is not necessarily the channel number given in the scheduler table entry. Value: 0-Fh

Table 17-127. CDMA Scheduler Table Word *n* Registers (ENTRYMSW[*n*]) Field Descriptions

Bit	Field	Type	Reset	Description
15	ENTRY3_RXTX	W	0	This entry is for a transmit or a receive channel. 0 = Transmit channel 1 = Receive channel
14-12	Reserved	R	0	Reserved

Table 17-127. CDMA Scheduler Table Word *n* Registers (ENTRYMSW[*n*]) Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11-8	ENTRY3_CHANNEL	W	0	Indicates the channel number that is to be given an opportunity to transfer data. If this is a transmit entry, the DMA will be presented with a scheduling credit for that exact transmit channel. If this is a receive entry, the DMA will be presented with a scheduling credit for the receive FIFO that is associated with this channel. For receive FIFOs which carry traffic for more than one receive DMA channel, the exact channel number that is given in the receive credit will actually be the channel number which is currently on the head element of that Rx FIFO, which is not necessarily the channel number given in the scheduler table entry. Value: 0-Fh
7	ENTRY2_RXTX	W	0	This entry is for a transmit or a receive channel. 0 = Transmit channel 1 = Receive channel
6-4	Reserved	R	0	Reserved
3-0	ENTRY2_CHANNEL	W	0	Indicates the channel number that is to be given an opportunity to transfer data. If this is a transmit entry, the DMA will be presented with a scheduling credit for that exact transmit channel. If this is a receive entry, the DMA will be presented with a scheduling credit for the receive FIFO that is associated with this channel. For receive FIFOs which carry traffic for more than one receive DMA channel, the exact channel number that is given in the receive credit will actually be the channel number which is currently on the head element of that Rx FIFO, which is not necessarily the channel number given in the scheduler table entry. Value: 0-Fh

17.3.60 Queue Manager Revision Identification Registers (QMGRREVID1 and QMGRREVID2)

The queue manager revision identification registers (QMGRREVID1 and QMGRREVID2) contain the major and minor revisions for the module. The QMGRREVID1 is shown in Figure 17-108 and described in Table 17-128. The QMGRREVID2 is shown in Figure 17-109 and described in Table 17-129.

Figure 17-108. Queue Manager Revision Identification Register 1 (QMGRREVID1)

15	0
REV_LSB	
R-1200h	

LEGEND: R = Read only; -*n* = value after reset

Figure 17-109. Queue Manager Revision Identification Register 2 (QMGRREVID2)

15	0
REV_MSB	
R-0052h	

LEGEND: R = Read only; -*n* = value after reset

Table 17-128. Queue Manager Revision Identification Register 1 (QMGRREVID1) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	REV_LSB	R	1200H	Revision ID of the queue manager. Least-significant bits. Value: 0-FFFFh

Table 17-129. Queue Manager Revision Identification Register 2 (QMGRREVID2) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	REV_MSB	R	0052h	Revision ID of the queue manager. Most-significant bits. Value: 0-FFFFh

17.3.61 Queue Manager Queue Diversion Registers (DIVERSION1 and DIVERSION2)

The queue manager queue diversion registers (DIVERSION1 and DIVERSION2) are used to transfer the contents of one queue onto another queue. It does not support byte accesses. The queue manager queue diversion register 1 (DIVERSION1) is shown in Figure 17-110 and described in Table 17-130. The queue manager queue diversion register 2 (DIVERSION2) is shown in Figure 17-111 and described in Table 17-131.

Figure 17-110. Queue Manager Queue Diversion Register 1 (DIVERSION1)

15	14	13	0
Reserved		SOURCE_QNUM	
R-0		W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 17-111. Queue Manager Queue Diversion Register 2 (DIVERSION2)

15	14	13	0
HEAD_TAIL	Reserved	DEST_QNUM	
W-0	R-0	W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-130. Queue Manager Queue Diversion Register 1 (DIVERSION1) Field Descriptions

Bit	Field	Type	Reset	Description
15-14	Reserved	R	0	Reserved.
13-0	SOURCE_QNUM	W	0	Source Queue Number. Value: 0-3FFFh

Table 17-131. Queue Manager Queue Diversion Register 2 (DIVERSION2 Field Descriptions

Bit	Field	Type	Reset	Description
15	HEAD_TAIL	W	0	Indicates whether queue contents should be merged on to the head or tail of the destination queue. 0 = Head. 1 = Tail.
14	Reserved	R	0	Reserved.
13-0	DEST_QNUM	W	0	Destination Queue Number. Value: 0-3FFFh

17.3.62 Queue Manager Free Descriptor/Buffer Starvation Count Register 0 (FDBSC0)

The free descriptor/buffer queue starvation count register (FDBSC0) provides statistics about how many starvation events are occurring on the receive free descriptor/buffer queues. These registers do not support byte accesses. The free descriptor/buffer queue starvation count register (FDBSC0) is shown in Figure 17-112 and described in Table 17-132.

Figure 17-112. Queue Manager Free Descriptor/Buffer Starvation Count Register 0 (FDBSC0)

15	8	7	0
FDBQ1_STARVE_CNT		FDBQ0_STARVE_CNT	
RC-0		RC-0	

LEGEND: RC = Cleared on read; -*n* = value after reset

Table 17-132. Queue Manager Free Descriptor/Buffer Starvation Count Register 0 (FDBSC0) Field Descriptions

Bit	Field	Type	Reset	Description
15-8	FDBQ1_STARVE_CNT	RC	0	This field increments each time the Free Descriptor/Buffer Queue 1 is read while it is empty. This field is cleared when read by CPU. Value: 0-FFh
7-0	FDBQ0_STARVE_CNT	RC	0	This field increments each time the Free Descriptor/Buffer Queue 0 is read while it is empty. This field is cleared when read by CPU. Value: 0-FFh

17.3.63 Queue Manager Free Descriptor/Buffer Starvation Count Register 1 (FDBSC1)

The free descriptor/buffer queue starvation count register (FDBSC1) provides statistics about how many starvation events are occurring on the receive free descriptor/buffer queues. These registers do not support byte accesses. The free descriptor/buffer queue starvation count register (FDBSC1) is shown in Figure 17-113 and described in Table 17-133.

Figure 17-113. Queue Manager Free Descriptor/Buffer Starvation Count Register 1 (FDBSC1)

15	8	7	0
FDBQ3_STARVE_CNT		FDBQ2_STARVE_CNT	
RC-0		RC-0	

LEGEND: RC = Cleared on read; -*n* = value after reset

Table 17-133. Queue Manager Free Descriptor/Buffer Starvation Count Register 1 (FDBSC1) Field Descriptions

Bit	Field	Type	Reset	Description
15-8	FDBQ3_STARVE_CNT	RC	0	This field increments each time the Free Descriptor/Buffer Queue 3 is read while it is empty. This field is cleared when readby CPU. Value: 0-FFh
7-0	FDBQ2_STARVE_CNT	RC	0	This field increments each time the Free Descriptor/Buffer Queue 2 is read while it is empty. This field is cleared when readby CPU. Value: 0-FFh

17.3.64 Queue Manager Free Descriptor/Buffer Starvation Count Register 2 (FDBSC2)

The free descriptor/buffer queue starvation count register 2 (FDBSC2) provides statistics about how many starvation events are occurring on the receive free descriptor/buffer queues. These registers do not support byte accesses. The free descriptor/buffer queue starvation count register 2 (FDBSC2) is shown in Figure 17-114 and described in Table 17-134.

Figure 17-114. Queue Manager Free Descriptor/Buffer Starvation Count Register 2 (FDBSC2)

15	8	7	0
FDBQ5_STARVE_CNT		FDBQ4_STARVE_CNT	
RC-0		RC-0	

LEGEND: RC = Cleared on read; -*n* = value after reset

Table 17-134. Queue Manager Free Descriptor/Buffer Starvation Count Register 2 (FDBSC2) Field Descriptions

Bit	Field	Type	Reset	Description
15-8	FDBQ5_STARVE_CNT	RC	0	This field increments each time the Free Descriptor/Buffer Queue 5 is read while it is empty. This field is cleared when read by CPU. Value: 0-FFh
7-0	FDBQ4_STARVE_CNT	RC	0	This field increments each time the Free Descriptor/Buffer Queue 4 is read while it is empty. This field is cleared when read by CPU. Value: 0-FFh

17.3.65 Queue Manager Free Descriptor/Buffer Starvation Count Register 3 (FDBSC3)

The free descriptor/buffer queue starvation count register 3 (FDBSC3) provides statistics about how many starvation events are occurring on the receive free descriptor/buffer queues. These registers do not support byte accesses. The free descriptor/buffer queue starvation count register 3 (FDBSC3) is shown in Figure 17-115 and described in Table 17-135.

Figure 17-115. Queue Manager Free Descriptor/Buffer Starvation Count Register 3 (FDBSC3)

15	8	7	0
FDBQ7_STARVE_CNT		FDBQ6_STARVE_CNT	
RC-0		RC-0	

LEGEND: RC = Cleared on read; -*n* = value after reset

Table 17-135. Queue Manager Free Descriptor/Buffer Starvation Count Register 3 (FDBSC3) Field Descriptions

Bit	Field	Type	Reset	Description
15-8	FDBQ7_STARVE_CNT	RC	0	This field increments each time the Free Descriptor/Buffer Queue 7 is read while it is empty. This field is cleared when read by CPU. Value: 0-FFh
7-0	FDBQ6_STARVE_CNT	RC	0	This field increments each time the Free Descriptor/Buffer Queue 6 is read while it is empty. This field is cleared when read by CPU. Value: 0-FFh

17.3.66 Queue Manager Free Descriptor/Buffer Starvation Count Register 4 (FDBSC4)

The free descriptor/buffer queue starvation count register 4 (FDBSC4) provides statistics about how many starvation events are occurring on the receive free descriptor/buffer queues. These registers do not support byte accesses. The free descriptor/buffer queue starvation count register 4 (FDBSC4) is shown in Figure 17-116 and described in Table 17-136.

Figure 17-116. Queue Manager Free Descriptor/Buffer Starvation Count Register 4 (FDBSC4)

15	8	7	0
FDBQ9_STARVE_CNT		FDBQ8_STARVE_CNT	
RC-0		RC-0	

LEGEND: RC = Cleared on read; -*n* = value after reset

Table 17-136. Queue Manager Free Descriptor/Buffer Starvation Count Register 4 (FDBSC4) Field Descriptions

Bit	Field	Type	Reset	Description
15-8	FDBQ9_STARVE_CNT	RC	0	This field increments each time the Free Descriptor/Buffer Queue 9 is read while it is empty. This field is cleared when read by CPU. Value: 0-FFh
7-0	FDBQ8_STARVE_CNT	RC	0	This field increments each time the Free Descriptor/Buffer Queue 8 is read while it is empty. This field is cleared when read by CPU. Value: 0-FFh

17.3.67 Queue Manager Free Descriptor/Buffer Starvation Count Register 5 (FDBSC5)

The free descriptor/buffer queue starvation count register 5 (FDBSC5) provides statistics about how many starvation events are occurring on the receive free descriptor/buffer queues. These registers do not support byte accesses. The free descriptor/buffer queue starvation count register 5 (FDBSC5) is shown in Figure 17-117 and described in Figure 17-117.

Figure 17-117. Queue Manager Free Descriptor/Buffer Starvation Count Register 5 (FDBSC5)

15	8	7	0
FDBQ11_STARVE_CNT		FDB10_STARVE_CNT	
RC-0		RC-0	

LEGEND: RC = Cleared on read; -*n* = value after reset

Table 17-137. Queue Manager Free Descriptor/Buffer Starvation Count Register 5 (FDBSC5) Field Descriptions

Bit	Field	Type	Reset	Description
15-8	FDBQ11_STARVE_CNT	RC	0	This field increments each time the Free Descriptor/Buffer Queue 11 is read while it is empty. This field is cleared when read by CPU. Value: 0-FFh
7-0	FDBQ10_STARVE_CNT	RC	0	This field increments each time the Free Descriptor/Buffer Queue 10 is read while it is empty. This field is cleared when read by CPU. Value: 0-FFh

17.3.68 Queue Manager Free Descriptor/Buffer Starvation Count Register 6 (FDBSC6)

The free descriptor/buffer queue starvation count register 6 (FDBSC6) provides statistics about how many starvation events are occurring on the receive free descriptor/buffer queues. These registers do not support byte accesses. The free descriptor/buffer queue starvation count register 6 (FDBSC6) is shown in Figure 17-118 and described in Table 17-138.

Figure 17-118. Queue Manager Free Descriptor/Buffer Starvation Count Register 6 (FDBSC6)

15	8	7	0
FDBQ13_STARVE_CNT		FDBQ12_STARVE_CNT	
RC-0		RC-0	

LEGEND: RC = Cleared on read; -*n* = value after reset

Table 17-138. Queue Manager Free Descriptor/Buffer Starvation Count Register 6 (FDBSC6) Field Descriptions

Bit	Field	Value	Description
15-8	FDBQ13_STARVE_CNT	0-FFh	This field increments each time the Free Descriptor/Buffer Queue 13 is read while it is empty. This field is cleared when read by CPU.
7-0	FDBQ12_STARVE_CNT	0-FFh	This field increments each time the Free Descriptor/Buffer Queue 12 is read while it is empty. This field is cleared when read by CPU.

17.3.69 Queue Manager Free Descriptor/Buffer Starvation Count Register 7 (FDBSC7)

The free descriptor/buffer queue starvation count register 7 (FDBSC7) provides statistics about how many starvation events are occurring on the receive free descriptor/buffer queues. The registers do not support byte accesses. The free descriptor/buffer queue starvation count register 7 (FDBSC7) is shown in Figure 17-119 and described in Table 17-139.

Figure 17-119. Queue Manager Free Descriptor/Buffer Starvation Count Register 7 (FDBSC7)

15	8	7	0
FDBQ15_STARVE_CNT		FDB14_STARVE_CNT	
RC-0		RC-0	

LEGEND: RC = Cleared on read; -*n* = value after reset

Table 17-139. Queue Manager Free Descriptor/Buffer Starvation Count Register 7 (FDBSC7) Field Descriptions

Bit	Field	Type	Reset	Description
15-8	FDBQ15_STARVE_CNT	RC	0	This field increments each time the Free Descriptor/Buffer Queue 15 is read while it is empty. This field is cleared when read by CPU. Value: 0-FFh
7-0	FDBQ14_STARVE_CNT	RC	0	This field increments each time the Free Descriptor/Buffer Queue 14 is read while it is empty. This field is cleared when read by CPU. Value: 0-FFh

17.3.70 Queue Manager Linking RAM Region 0 Base Address Registers (LRAM0BASE1 and LRAM0BASE2)

The queue manager linking RAM region 0 base address registers (LRAM0BASE1 and LRAM0BASE2) set the base address for the first portion of the Linking RAM. This address must be 32-bit aligned. It is used by the Queue Manager to calculate the 32-bit linking address for a given descriptor index. These registers do not support byte accesses.

The queue manager linking RAM region 0 base address register 1 (LRAM0BASE1) is shown in Figure 17-120 and described in Table 17-140. The queue manager linking RAM region 0 base address register 2 (LRAM0BASE2) is shown in Figure 17-121 and described in Table 17-141.

Figure 17-120. Queue Manager Linking RAM Region 0 Base Address Register 1 (LRAM0BASE1)

15	0
REGION0_BASE_LSB	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Figure 17-121. Queue Manager Linking RAM Region 0 Base Address Register 2 (LRAM0BASE2)

15	0
REGION0_BASE_MSB	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-140. Queue Manager Linking RAM Region 0 Base Address Register 1 (LRAM0BASE1) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	REGION0_BASE_LSB	RW	0	This field stores the 16 least significant bits of the base address for the first region of the linking RAM. This may be anywhere in 32-bit address space but would be typically located in on-chip memory. Value: 0-FFFFh

Table 17-141. Queue Manager Linking RAM Region 0 Base Address Register 2 (LRAM0BASE2) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	REGION0_BASE_MSB	RW	0	This field stores the 16 most significant bits of the base address for the first region of the linking RAM. This may be anywhere in 32-bit address space but would be typically located in on-chip memory. Value: 0-FFFFh

17.3.71 Queue Manager Linking RAM Region 0 Size Register (LRAM0SIZE)

The queue manager linking RAM region 0 size register (LRAM0SIZE) sets the size of the array of linking pointers that are located in Region 0 of Linking RAM. The size specified the number of descriptors for which linking information is stored in this region. It does not support byte accesses. The queue manager linking RAM region 0 size register (LRAM0SIZE) is shown in Figure 17-122 and described in Table 17-142.

Figure 17-122. Queue Manager Linking RAM Region 0 Size Register (LRAM0SIZE)

15	14	13	0
Reserved		REGION0_SIZE	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-142. Queue Manager Linking RAM Region 0 Size Register (LRAM0SIZE) Field Descriptions

Bit	Field	Type	Reset	Description
15-14	Reserved	R	0	Reserved.
13-0	REGION0_SIZE	RW	0	This field indicates the number of entries that are contained in the linking RAM region 0. A descriptor with index less than region0_size value has its linking location in region 0. A descriptor with index greater than region0_size has its linking location in region 1. The queue manager will add the index (left shifted by 2 bits) to the appropriate regionX_base_addr to get the absolute 32-bit address to the linking location for a descriptor. Value: 0-3FFh

17.3.72 Queue Manager Linking RAM Region 1 Base Address Registers (LRAM1BASE1 and LRAM1BASE2)

The queue manager linking RAM region 1 base address registers (LRAM1BASE1 and LRAM1BASE2) are used to set the base address for the first portion of the Linking RAM. This address must be 32-bit aligned. These registers are used by the Queue Manager to calculate the 32-bit linking address for a given descriptor index. These registers do not support byte accesses.

The queue manager linking RAM region 1 base address register (LRAM1BASE1) is shown in Figure 17-123 and described in Table 17-143. The queue manager linking RAM region 1 base address register (LRAM1BASE2) is shown in Figure 17-124 and described in Table 17-144.

Figure 17-123. Queue Manager Linking RAM Region 1 Base Address Register 1 (LRAM1BASE1)

15	0
REGION1_BASE_LSB	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Figure 17-124. Queue Manager Linking RAM Region 1 Base Address Register 2 (LRAM1BASE2)

15	0
REGION1_BASE_MSB	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-143. Queue Manager Linking RAM Region 1 Base Address Register 1 (LRAM1BASE1) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	REGION1_BASE_LSB	RW	0	This field stores the least significant bits of the base address for the second region of the linking RAM. This may be anywhere in 32-bit address space but would be typically located in off-chip memory. Value: 0-FFFFh

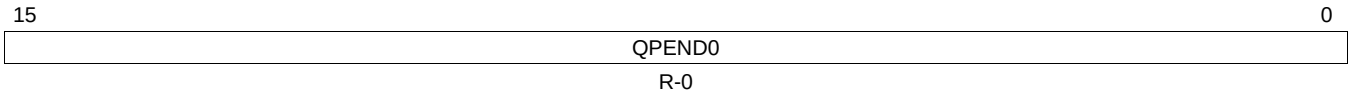
Table 17-144. Queue Manager Linking RAM Region 1 Base Address Register (LRAM1BASE2) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	REGION1_BASE_MSB	RW	0	This field stores the most significant bits of the base address for the second region of the linking RAM. This may be anywhere in 32-bit address space but would be typically located in off-chip memory. Value: 0-FFFFh

17.3.73 Queue Manager Queue Pending Register 0 (PEND0)

The queue pending register 0 (PEND0) can be read to find the pending status for queues 15 to 0. It does not support byte accesses. The queue pending register 0 (PEND0) is shown in Figure 17-125 and described in Table 17-145.

Figure 17-125. Queue Manager Queue Pending Register 0 (PEND0)



LEGEND: R = Read only; -*n* = value after reset

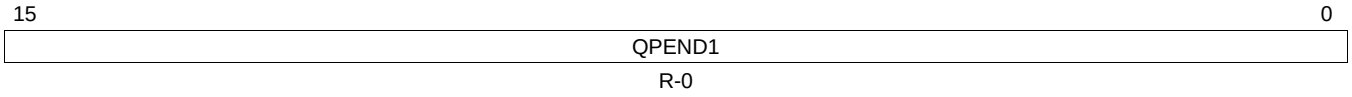
Table 17-145. Queue Manager Queue Pending Register 0 (PEND0) Field Descriptions

Bit	Field	Type	Recet	Description
15-0	QPEND0	R	0	This field indicates the queue pending status for queues 15-0. Value: 0-FFFFh

17.3.74 Queue Manager Queue Pending Register 1 (PEND1)

The queue pending register 1 (PEND1) can be read to find the pending status for queues 31 to 16. It does not support byte accesses. The queue pending register 1 (PEND1) is shown in Figure 17-126 and described in Table 17-146.

Figure 17-126. Queue Manager Queue Pending Register 1 (PEND1)



LEGEND: R = Read only; -*n* = value after reset

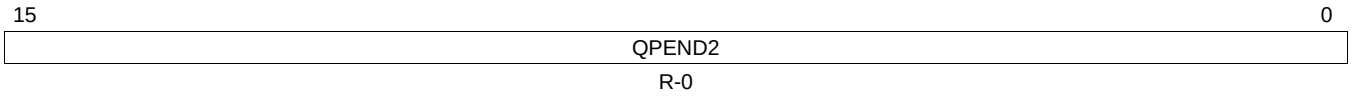
Table 17-146. Queue Manager Queue Pending Register 1 (PEND1) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	QPEND1	R	0	This field indicates the queue pending status for queues 31-16. Value: 0-FFFFh

17.3.75 Queue Manager Queue Pending Register 2 (PEND2)

The queue pending register 2 (PEND2) can be read to find the pending status for queues 47 to 32. It does not support byte accesses. The queue pending register 2 (PEND2) is shown in Figure 17-127 and described in Table 17-147.

Figure 17-127. Queue Manager Queue Pending Register 2 (PEND2)



LEGEND: R = Read only; -*n* = value after reset

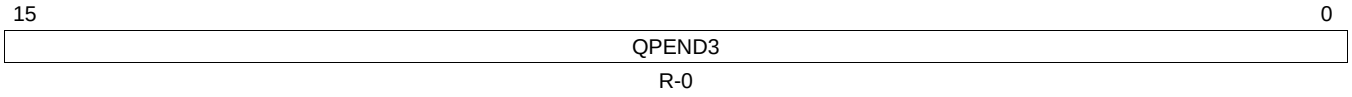
Table 17-147. Queue Manager Queue Pending Register 2 (PEND2) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	QPEND2	R	0	This field indicates the queue pending status for queues 47-32.Value: 0-FFFFh

17.3.76 Queue Manager Queue Pending Register 3 (PEND3)

The queue pending register 3 (PEND3) can be read to find the pending status for queues 63 to 48. It does not support byte accesses. The queue pending register 3 (PEND3) is shown in Figure 17-128 and described in Table 17-148.

Figure 17-128. Queue Manager Queue Pending Register 3 (PEND3)



LEGEND: R = Read only; -*n* = value after reset

Table 17-148. Queue Manager Queue Pending Register 3 (PEND3) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	QPEND3	R	0	This field indicates the queue pending status for queues 63-48. Value: 0-FFFFh

17.3.77 Queue Manager Queue Pending Register 4 (PEND4)

The queue pending register 4 (PEND4) can be read to find the pending status for queues 79 to 64. It does not support byte accesses. The queue pending register 4 (PEND4) is shown in Figure 17-129 and described in Table 17-149.

Figure 17-129. Queue Manager Queue Pending Register 4 (PEND4)

15	0
QPEND4	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 17-149. Queue Manager Queue Pending Register 4 (PEND4) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	QPEND4	R	0	This field indicates the queue pending status for queues 79-64. Value: 0-FFFFh

17.3.78 Queue Manager Queue Pending Register 5 (PEND5)

The queue pending register 5 (PEND5) can be read to find the pending status for queues 95 to 80. It does not support byte accesses. The queue pending register 5 (PEND5) is shown in Figure 17-130 and described in Table 17-150.

Figure 17-130. Queue Manager Queue Pending Register 5 (PEND5)

15	0
QPEND5	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 17-150. Queue Manager Queue Pending Register 5 (PEND5) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	QPEND5	R	0	This field indicates the queue pending status for queues 95-80. Value: 0-FFFFh

17.3.79 Queue Manager Memory Region *R* Base Address Registers (QMEMRBASE1[R] and QMEMRBASE2[R])

The memory region *R* base address registers (QMEMRBASE1[R] and QMEMRBASE2[R]) are written by the host to set the base address of memory region *R*, where *R* is 0-15. This memory region will store a number of descriptors of a particular size as determined by the memory region *R* control register. These registers do not support byte accesses.

The memory region *R* base address register (QMEMRBASE1[*R*]) is shown in Figure 17-131 and described in Table 17-151. The memory region *R* base address register (QMEMRBASE2[*R*]) is shown in Figure 17-132 and described in Table 17-152.

Figure 17-131. Queue Manager Memory Region *R* Base Address Register 1 (QMEMRBASE1[*R*])

15	0
REG_LSB	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Figure 17-132. Queue Manager Memory Region *R* Base Address Register 2 (QMEMRBASE2[*R*])

15	0
REG_MSB	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-151. Queue Manager Memory Region *R* Base Address Register 1 (QMEMRBASE1[*R*]) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	REG_LSB	RW	0	This field contains the least-significant bits of the base address of the memory region R. Value: 0-FFFFh

Table 17-152. Queue Manager Memory Region *R* Base Address Register 2 (QMEMRBASE2[*R*]) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	REG_MSB	RW	0	This field contains the most-significant bits of the base address of the memory region R. Value: 0-FFFFh

17.3.80 Queue Manager Memory Region *R* Control Registers (QMEMRCTRL1[R] and QMEMRCTRL2[R])

The memory region *R* control registers (QMEMRCTRL1[R] and QMEMRCTRL2[R]) are written by the host to configure various parameters of memory region *R*, where *R* is 0-15. These registers do not support byte accesses.

The memory region *R* control register (QMEMRCTRL1[*R*])) is shown in Figure 17-133 and described in Table 17-153. The memory region *R* control register (QMEMRCTRL2[*R*])) is shown in Figure 17-134 and described in Table 17-154.

Figure 17-133. Queue Manager Memory Region *R* Control Register 1 (QMEMRCTRL1[*R*])

15 12	11 8	7 3	2 0
Reserved	DESC_SIZE	Reserved	REG_SIZE
R-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 17-134. Queue Manager Memory Region *R* Control Register 2 (QMEMRCTRL2[*R*])

15 14	13 0
Reserved	START_INDEX
R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-153. Queue Manager Memory Region *R* Control Register 1 (QMEMRCTRL1[*R*]) Field Descriptions

Bit	Field	Type	Reset	Description
15-12	Reserved	R	0	Reserved.
11-8	DESC_SIZE	RW	0	This field indicates the size of each descriptor in this memory region. 0 = 32 1h = 64 2h = 128 3h = 256 4h = 512 5h = 1/k 6h = 2/k 7h = 4/k 8h = 8/k 9h-Fh = Reserved
7-3	Reserved	R	0	Reserved.
2-0	REG_SIZE	RW	0	This field indicates the size of the memory region (in terms of number of descriptors). 0 = 32 1h = 64 2h = 128 3h = 256 4h = 512 5h = 1/k 6h = 2/k 7h = 4/k 8h = 8/k 9h-Fh = Reserved

Table 17-154. Queue Manager Memory Region *R* Control Register 2 (QMEMRCTRL2[*R*]) Field Descriptions

Bit	Field	Type	Reset	Description
15-14	Reserved	R	0	Reserved.
13-0	START_INDEX	RW	0	This field indicates where in linking RAM the descriptor linking information corresponding to memory region R starts. Value: 0-3FFFh

17.3.81 Queue Manager Queue N Control Register D (CTRL1D[N] and CTRL2D[N])

The queue manager queue *N* control registers D (CTRL1D[N] and CTRL2D[N]) are written to add a packet to the queue and read to pop a packets off a queue. The packet is only pushed or popped to/from the queue when the queue manager queue *N* control register D is written. These registers do not support byte accesses.

The queue manager queue *N* control register 1 D (CTRL1D[*N*]) is shown in Figure 17-135 and described in Table 17-155. The queue manager queue *N* control register 2 D (CTRL2D[*N*]) is shown in Figure 17-136 and described in Table 17-156.

Figure 17-135. Queue Manager Queue *N* Control Register 1 D (CTRL1D[*N*])

15	5	4	0
DESC_PTR_LSB		DESC_SIZE	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; *-n* = value after reset

Figure 17-136. Queue Manager Queue *N* Control Register 2 D (CTRL2D[*N*])

15	0
DESC_PTR_MSB	
R/W-0	

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 17-155. Queue Manager Queue *N* Control Register 1 D (CTRL1D[*N*]) Field Descriptions

Bit	Field	Type	Reset	Description
15-5	DESC_PTR_LSB	RW	0	Descriptor Pointer (Least significant bits). 0 = Queue is empty. 1 = Indicates a 32-bit aligned address that points to a descriptor.
4-0	DESC_SIZE	RW	0	The descriptor size is encoded in 4-byte increments. This field returns a 0 when an empty queue is read. 0 = 24 bytes. 1h = 28 bytes. 2h = 32 bytes. 3h to 1Fh = 36 bytes to 148 bytes.

Table 17-156. Queue Manager Queue *N* Control Register 2 D (CTRL2D[*N*]) Field Descriptions

Bit	Field	Value	Description
15-0	DESC_PTR_MSB		Descriptor Pointer (Most significant bits).
		0	Queue is empty.
		1	Indicates a 32-bit aligned address that points to a descriptor.

17.3.82 Queue Manager Queue *N* Status Register A (QSTATA[N])

The queue manager queue *N* status register A (QSTATA[*N*]) is an optional register that is only implemented for a queue if the queue supports entry/byte count feature. The entry count feature provides a count of the number of entries that are currently valid in the queue. It does not support byte accesses. The queue manager queue *N* status register A (QSTATA[*N*]) is shown in Figure 17-137 and described in Table 17-157.

Figure 17-137. Queue Manager Queue *N* Status Register A (QSTATA[*N*])

15	14	13	0
Reserved		QUEUE_ENTRY_COUNT	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 17-157. Queue Manager Queue *N* Status Register A (QSTATA[*N*]) Field Descriptions

Bit	Field	Type	Reset	Description
15-14	Reserved	R	0	Reserved.
13-0	QUEUE_ENTRY_COUNT	R	0	This field indicates how many packets are currently queued on the queue. Value: 0-3FFFh

17.3.83 Queue Manager Queue *N* Status Registers B (QSTAT1B[N] and QSTAT2B[N])

The queue manager queue *N* status registers B (QSTAT1B[N] and QSTAT2B[N]) are optional registers that are only implemented for a queue if the queue supports a total byte count feature. The total byte count feature provides a count of the total number of bytes in all of the packets that are currently valid in the queue. The registers do not support byte accesses.

The queue manager queue *N* status register 1 B (QSTAT1B[*N*]) is shown in Figure 17-138 and described in Table 17-158. The queue manager queue *N* status register 2 B (QSTAT2B[*N*]) is shown in Figure 17-139 and described in Table 17-159.

Figure 17-138. Queue Manager Queue *N* Status Register 1 B (QSTAT1B[*N*])

15	0
QUEUE_BYTE_COUNT_LSB	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Figure 17-139. Queue Manager Queue *N* Status Register 2 B (QSTAT2B[*N*])

15	12	11	0
Reserved		QUEUE_BYTE_COUNT_MSB	

LEGEND: R = Read only; -*n* = value after reset

Table 17-158. Queue Manager Queue *N* Status Register 1 B (QSTAT1B[*N*]) Field Descriptions

Bit	Field	Type	Reset	Description
15-0	QUEUE_BYTE_COUNT_LSB	R	0	Together, QUEUE_BYTE_COUNT_MSB and QUEUE_BYTE_COUNT_LSB indicate how many bytes total are contained in all of the packets which are currently queued on this queue. Value: 0-FFFFh

Table 17-159. Queue Manager Queue *N* Status Register 2 B (QSTAT2B[*N*]) Field Descriptions

Bit	Field	Type	Reset	Description
15-12	Reserved	R	0	Reserved.

Table 17-159. Queue Manager Queue *N* Status Register 2 B (QSTAT2B[*N*]) Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11-0	QUEUE_BYTE_COUNT_MSB	R	0	Together, QUEUE_BYTE_COUNT_MSB and QUEUE_BYTE_COUNT_LSB indicate how many bytes total are contained in all of the packets which are currently queued on this queue. Value: 0-FFFh

17.3.84 Queue Manager Queue N Status Register C (QSTATC[N])

The queue manager queue *N* status register C (QSTATC[*N*]) specifies the packet size for the head element of a queue. It does not support byte accesses. The queue manager queue *N* status register C (QSTATC[*N*]) is shown in Figure 17-140 and described in Table 17-160.

Figure 17-140. Queue Manager Queue *N* Status Register C (QSTATC[*N*])

15	14	13	0
Reserved		PACKET_SIZE	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 17-160. Queue Manager Queue *N* Status Register C (QSTATC[*N*]) Field Descriptions

Bit	Field	Type	Reset	Description
15-14	Reserved	R	0	Reserved.
13-0	PACKET_SIZE	R	0	This field indicates how many packets are currently queued on the queue. Value: 0-3FFFh

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license 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 significant portions 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. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

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 Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265