TMS320C5515/14/05/04/VC05/VC04 DSP Multimedia Card (MMC)/Secure Digital (SD) Card Controller

Reference Guide

Literature Number: SPRUFO6A
November 2009–Revised July 2010

Contents

List of Figures

List of Tables

Read This First

This document describes the Multimedia Card (MMC)/Secure Digital (SD) Card Controller on the TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP).

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.
- Registers in this document are shown in figures and described in tables.
  - Each register figure shows a rectangle divided into fields that represent the fields of the register. Each field is labeled with its bit name, its beginning and ending bit numbers above, and its read/write properties below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure designate a bit that is used for future device expansion.

Related Documentation From Texas Instruments

The following documents describe the TMS320C5515/14/05/04 Digital Signal Processor (DSP) Digital Signal Processor (DSP). Copies of these documents are available on the internet at http://www.ti.com.

SWPU073 — TMS320C55x 3.0 CPU Reference Guide. This manual describes the architecture, registers, and operation of the fixed-point TMS320C55x digital signal processor (DSP) CPU.

SPRU652 — TMS320C55x DSP CPU Programmer's Reference Supplement. This document describes functional exceptions to the CPU behavior.

SPRUFO0 — TMS320VC5505/5504 Digital Signal Processor (DSP) Universal Serial Bus 2.0 (USB) User's Guide. This document describes the universal serial bus 2.0 (USB) in the TMS320VC5505/5504 Digital Signal Processor (DSP) devices. The USB controller supports data throughput rates up to 480 Mbps. It provides a mechanism for data transfer between USB devices.

SPRUFO1A — TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) Inter-Integrated Circuit (I2C) Peripheral User's Guide. This document describes the inter-integrated circuit (I2C) peripheral in the TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) devices. The I2C peripheral provides an interface between the device and other devices compliant with Phillips Semiconductors Inter-IC bus (I2C-bus) specification version 2.1 and connected by way of an I2C-bus. This document assumes the reader is familiar with the I2C-bus specification.

SPRUFO2 — TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) Timer/Watchdog Timer User's Guide. This document provides an overview of the three 32-bit timers in the TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) devices. The 32-bit timers of the device are software programmable timers that can be configured as general-purpose (GP) timers. Timer 2 can be configured as a GP, a Watchdog (WD), or both simultaneously.

SPRUFO3 — TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) Serial Peripheral Interface (SPI) User's Guide. This document describes the serial peripheral interface (SPI) in the TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) devices. 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.

SPRUFO4 — TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) General-Purpose Input/Output (GPIO) User's Guide. This document describes the general-purpose input/output (GPIO) on the TMS320C5515/14/05/04/VC05/VC04 digital signal processor (DSP) devices. The GPIO peripheral provides dedicated general-purpose pins that can be configured as either inputs or outputs. When configured as an input, you can detect the state of an internal register. When configured as an output you can write to an internal register to control the state driven on the output pin.

SPRUFO5 — TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) Universal Asynchronous Receiver/Transmitter (UART) User's Guide. This document describes the universal asynchronous receiver/transmitter (UART) peripheral in the TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) devices. 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.

SPRUFO6 — TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) Multimedia Card (MMC)/Secure Digital (SD) Card Controller User's Guide. This document describes the Multimedia Card (MMC)/Secure Digital (SD) Card Controller on the TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) devices. 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 external MMC and SD cards.

SPRUF07 — TMS320VC5505/5504 Digital Signal Processor (DSP) Real-Time Clock (RTC) User's Guide. This document describes the operation of the Real-Time Clock (RTC) module in the TMS320VC5505/5504 Digital Signal Processor (DSP) devices. The RTC also has the capability to wake-up the power management and apply power to the rest of the device through an alarm, periodic interrupt, or external WAKEUP signal.

SPRUFO8A — TMS320VC5505/5504 Digital Signal Processor (DSP) External Memory Interface (EMIF) User's Guide. This document describes the operation of the external memory interface (EMIF) in the TMS320VC5505/5504 Digital Signal Processor (DSP) devices. The purpose of the EMIF is to provide a means to connect to a variety of external devices.

SPRUFO9 — TMS320VC5505/5504 Digital Signal Processor (DSP) Direct Memory Access (DMA) Controller User's Guide. This document describes the features and operation of the DMA controller that is available on the TMS320VC5505/5504 Digital Signal Processor (DSP) devices. The DMA controller is used to move data among internal memory, external memory, and peripherals without intervention from the CPU and in the background of CPU operation.

SPRUFP0 — TMS320VC5505 Digital Signal Processor (DSP) System User's Guide. This document describes various aspects of the TMS320VC5505/5504 digital signal processor (DSP) including: system memory, device clocking options and operation of the DSP clock generator, power management features, interrupts, and system control.

SPRUGL6 — TMS320VC5504 Digital Signal Processor (DSP) System User's Guide. This document describes various aspects of the TMS320VC5505/5504 digital signal processor (DSP) including: system memory, device clocking options and operation of the DSP clock generator, power management features, interrupts, and system control.

SPRUFP1 — TMS320C5515/05/VC05 Digital Signal Processor (DSP) Successive Approximation (SAR) Analog to Digital Converter (ADC) User's Guide. This document provides an overview of the Successive Approximation (SAR) Analog to Digital Converter (ADC) on the TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) devices. The SAR is a 10-bit ADC using a switched capacitor architecture which converts an analog input signal to a digital value.

SPRUFP3 — TMS320C5515/05/VC05 Digital Signal Processor (DSP) Liquid Crystal Display Controller (LCDC) User's Guide. This document describes the liquid crystal display controller (LCDC) in the TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP) devices. The LCD controller includes a LCD Interface Display Driver (LIDD) controller.

SPRUFP4 — TMS320VC5505/5504 Digital Signal Processor (DSP) Inter-IC Sound (I2S) Bus User's Guide. This document describes the features and operation of Inter-IC Sound (I2S) Bus in the TMS320VC5505/5504 Digital Signal Processor (DSP) devices. This peripheral allows serial transfer of full duplex streaming data, usually streaming audio, between DSP and an external I2S peripheral device such as an audio codec.

Multimedia Card (MMC)/Secure Digital (SD) Card Controller

1.1 Introduction

This document describes the Multimedia Card (MMC)/Secure Digital (SD) Card Controller in the TMS320C5515/14/05/04/VC05/VC04 Digital Signal Processor (DSP).

1.1.1 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.

1.1.2 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 TMS320VC5505/5504 device includes two independent MMC/SD card controllers.

1.1.3 Functional Block Diagram

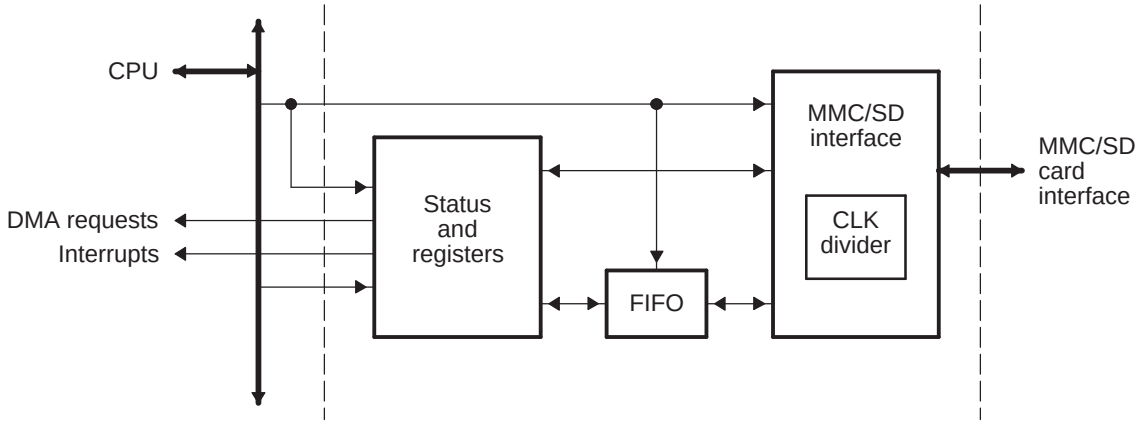
The MMC/SD card controller transfers data between the CPU, DMA, and MMC/SD as shown in Figure 1-1. The CPU and DMA controller can read/write the data in the card by accessing the registers in the MMC/SD controller.

1.1.4 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 1-1. MMC/SD Card Controller Block Diagram



1.1.5 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.

1.2 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 1-2 summarizes the MMC/SD mode interface. Figure 1-3 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 Section 1.4.1).
- CLK: This pin provides the clock to the memory card from the MMC/SD controller.

Figure 1-2. MMC/SD Controller Interface Diagram

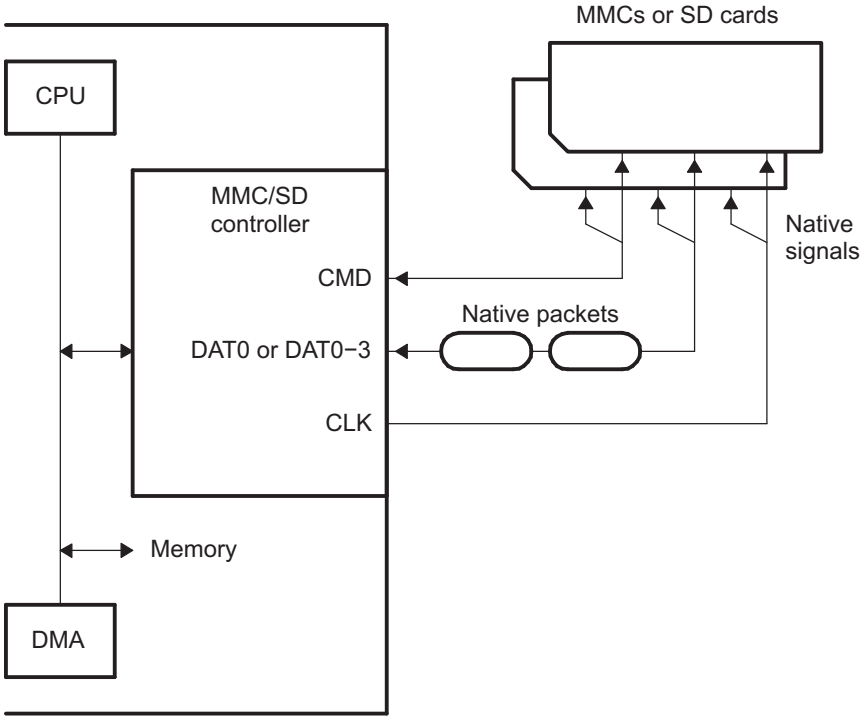
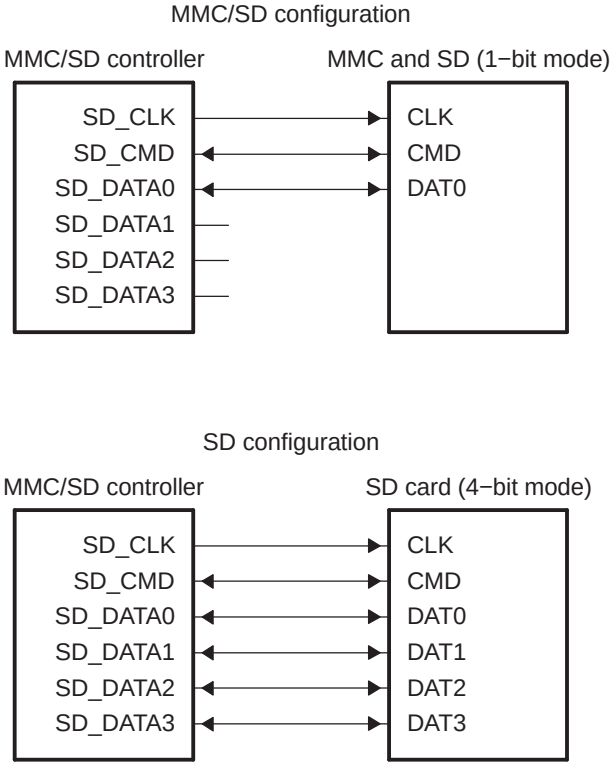


Figure 1-3. MMC Configuration and SD Configuration Diagram



1.2.1 Clock Control

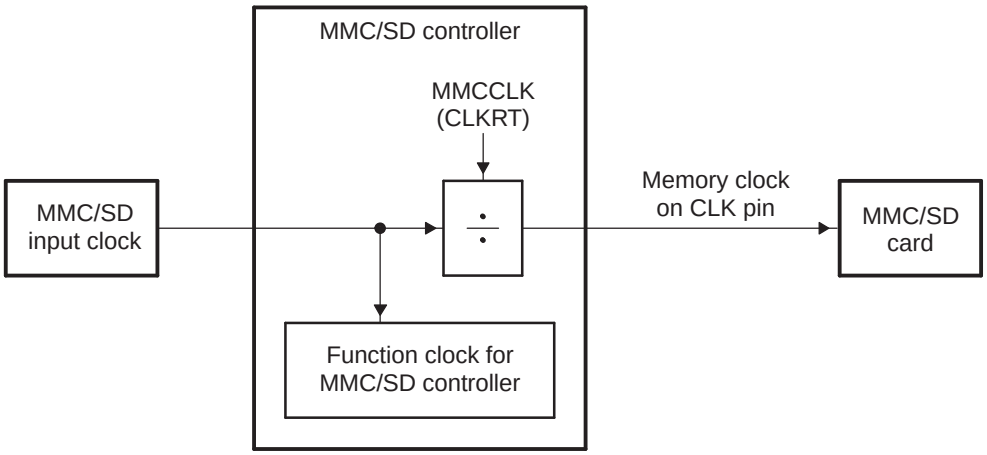
The MMC/SD controller has two clocks: the function clock and the memory clock (see Figure 1-4).

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 MMC/SD controller is capable of operating with a function clock up to 100 MHz.

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 1-4. 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.

1.2.2 Signal Descriptions

Table 1-1 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 1-1. 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.

1.2.3 Pin Multiplexing

The MMC/SD card controller pins are multiplexed with other peripherals on the TMS320C5505/C5504 DSPs. 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 the *TMS320VC5505/5504 DSP System Guide* (SPRUFP0).

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.

1.2.4 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 1.1.5.

1.2.4.1 MMC/SD Mode Write Sequence

Figure 1-5 and Table 1-2 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 1-5. MMC/SD Mode Write Sequence Timing Diagram

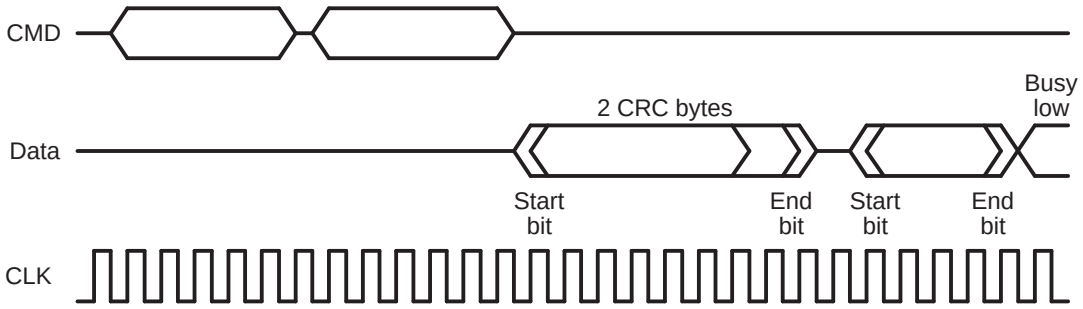


Table 1-2. MMC/SD Mode Write Sequence

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.

1.2.4.2 MMC/SD Mode Read Sequence

Figure 1-6 and Table 1-3 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 1-6. MMC/SD Mode Read Sequence Timing Diagram

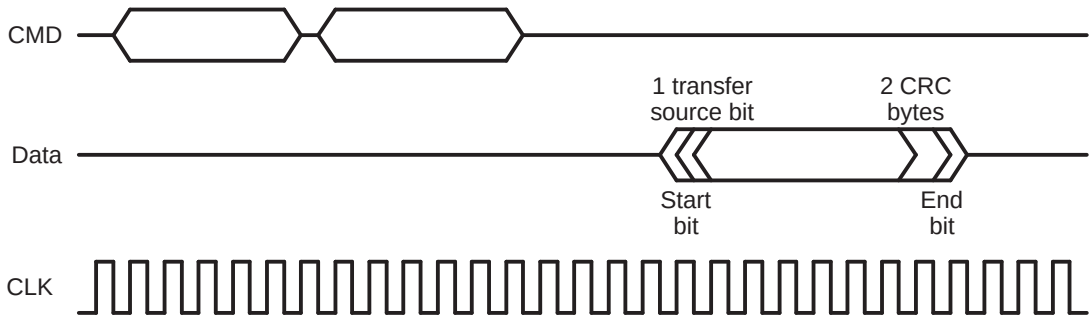


Table 1-3. MMC/SD Mode Read Sequence

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.

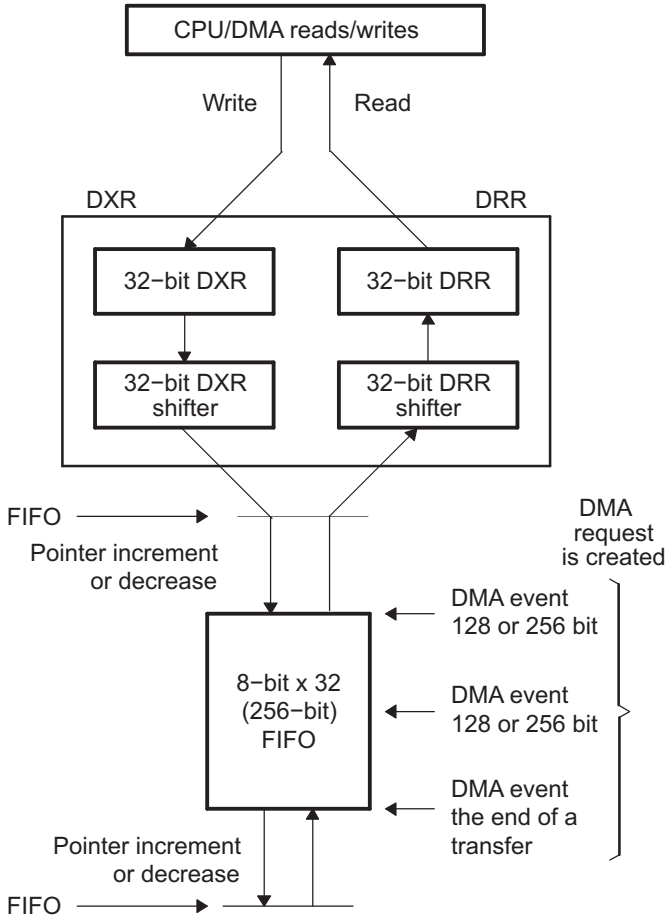
1.2.5 Data Flow in the Input/Output FIFO

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 1-7). 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 1-7. 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

1.2.6 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, wite 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 1-8 and Figure 1-9 shows how the data-size difference is handled by the data registers.

Figure 1-8. Little-Endian Access to MMCDXR/MMCDRR1 and 2 From the CPU or the DMA

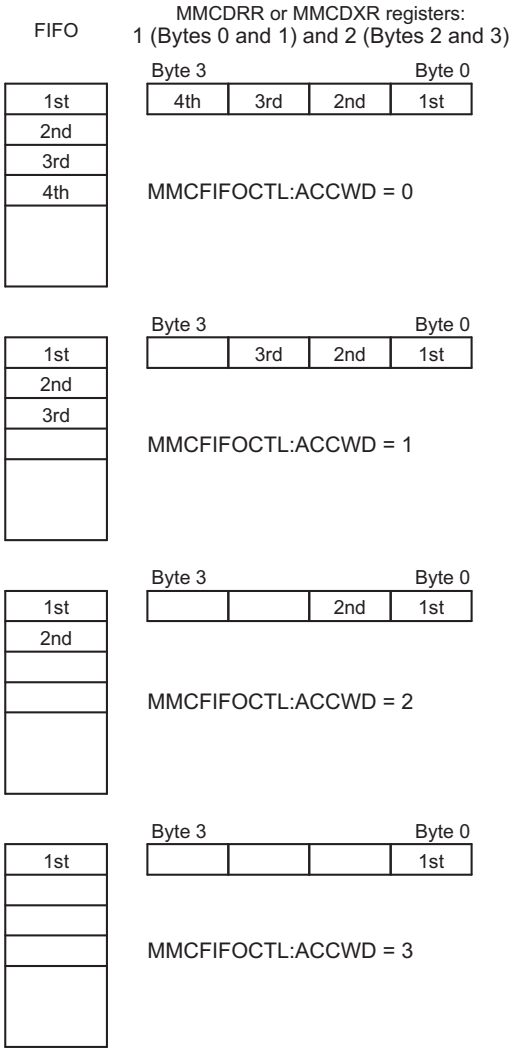
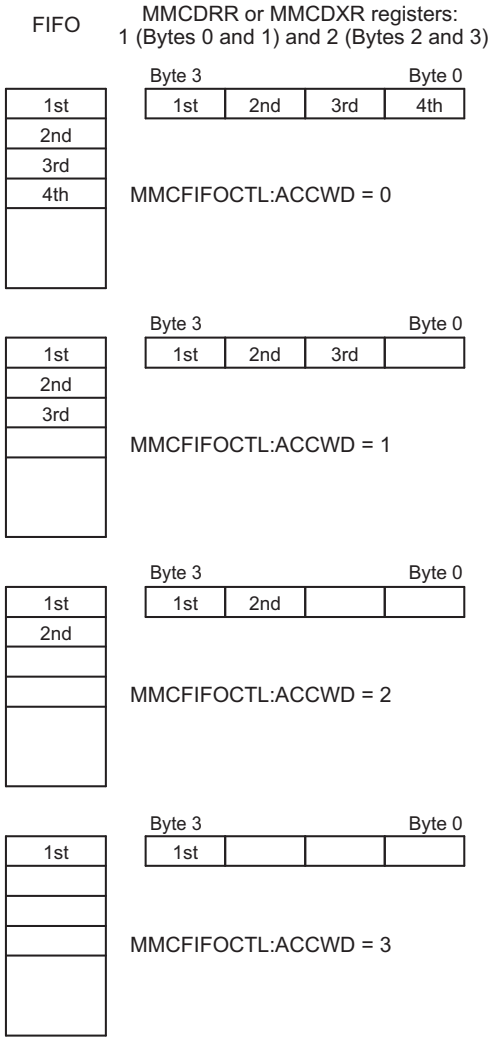


Figure 1-9. Big-Endian Access to MMCDXR/MMCDRR1 and 2 From the CPU or the DMA



1.2.7 FIFO Operation During Card Read Operation

The MMC/SD controller supports 1-, 2-, 3-, or 4-byte reads as shown in Figure 1-8 and Figure 1-9. The CPU makes 16-bit and the DMA makes 32-bit accesses to the MMCDRR registers.

1.2.7.1 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 1-10 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.

1.2.7.2 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.

1.2.8 FIFO Operation During Card Write Operation

The MMC/SD controller supports 1-, 2-, 3-, or 4-byte writes as shown in Figure 1-8 and Figure 1-9. The CPU makes 16-bit and the DMA makes 32-bit accesses to the MMCDXR registers.

1.2.8.1 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 (MMCCMD2) after the desired write command is written to the MMCCMD1 register.

Figure 1-11 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.

1.2.8.2 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 1-10. FIFO Operation During Card Read Diagram

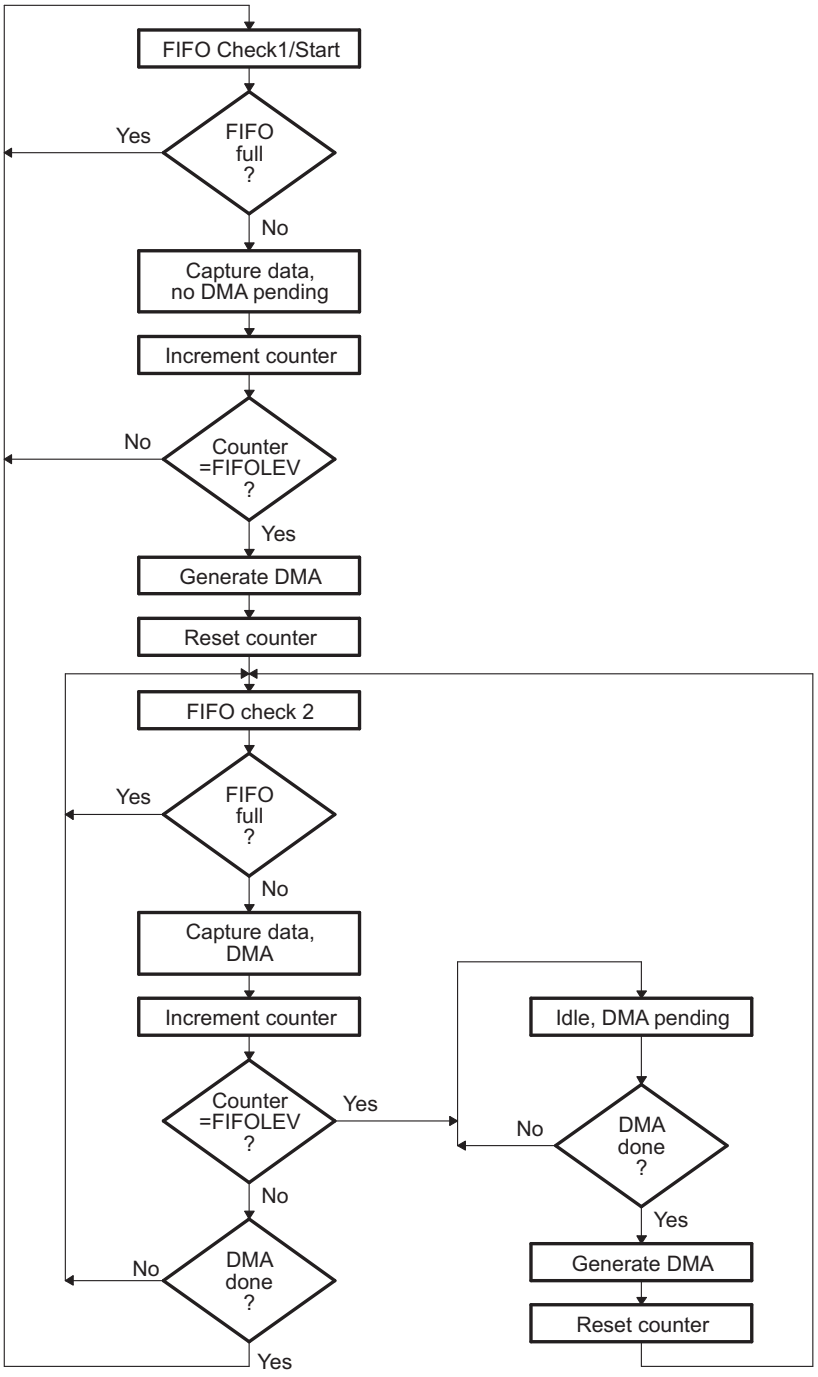
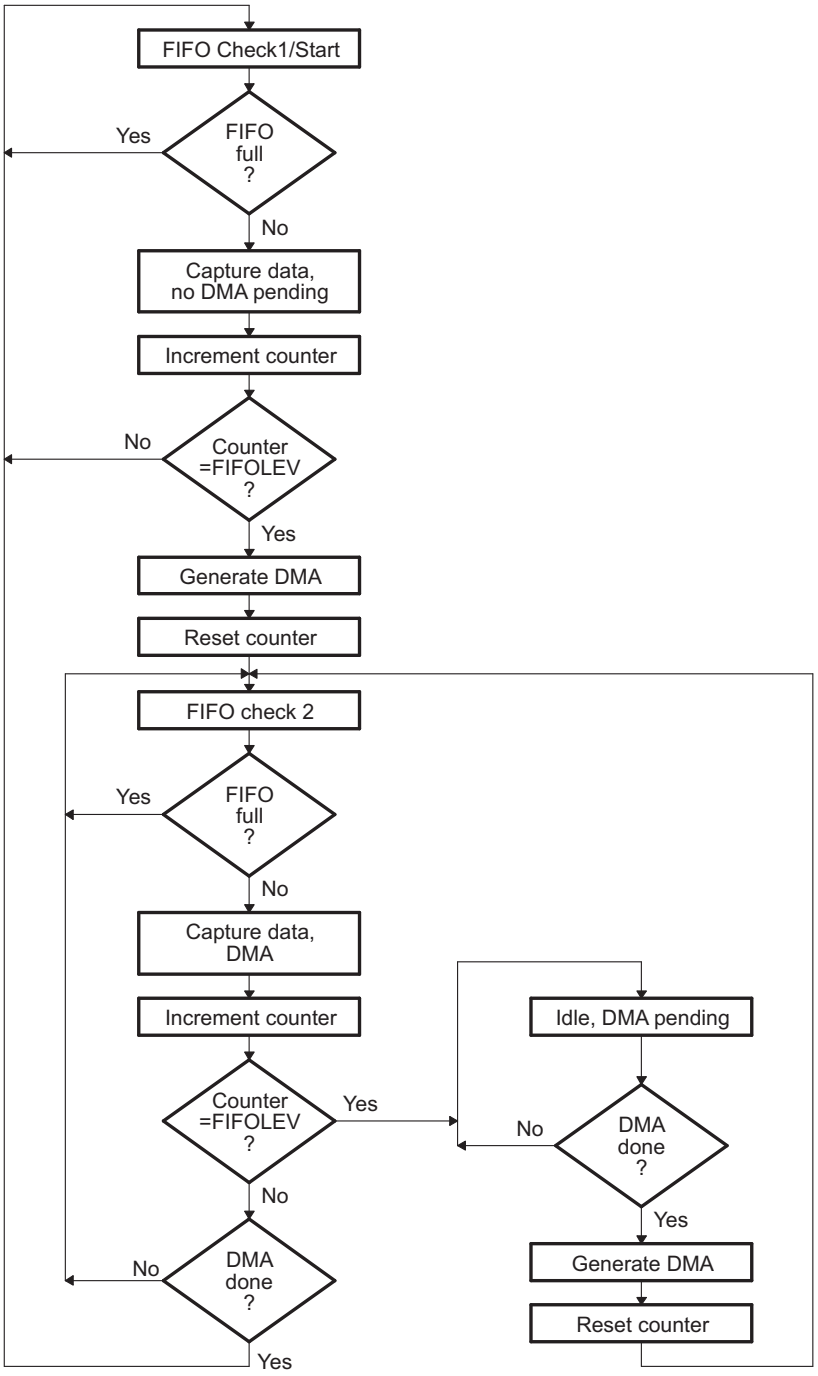


Figure 1-11. FIFO Operation During Card Write Diagram



1.2.9 Reset Considerations

The MMC/SD controller has two reset sources: hardware reset and software reset.

1.2.9.1 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.

1.2.9.2 Hardware Reset Considerations

A hardware reset of the processor will cause the MMC/SD controller registers to return to their default values after reset.

1.2.10 Programming and Using the MMCSD Controller

1.2.10.1 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 1.2.3 before the controller can communicate with the MMC/SD card.

1.2.10.2 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.

1.2.10.3 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.

1.2.10.4 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.

1.2.10.5 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.

1.2.10.6 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.

1.2.10.7 Using the Command Registers (MMCCMD1 and MMCCMD2)

The MMCCMD1 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 MMCCMD2 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 MMCCMD2 after the desired write command has been written to the MMCCMD1 register. If this bit is written to at any other time, the controller will resend the last command (configured in the MMCCMD1 register) to the card.

1.2.10.8 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).

1.2.10.8.1 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).

1.2.10.8.2 Detecting Level Changes on the DAT3 Pin

The DAT3ST bit in MMCST1 monitors the signal level on the SD_DATA3 pin.

1.2.10.8.3 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 a MMCDDR register.

1.2.10.8.4 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).

1.2.10.8.5 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).

1.2.10.8.6 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).

1.2.10.8.7 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).

1.2.10.8.8 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 MMCCMD). 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.

1.2.10.8.9 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).

1.2.10.8.10 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.

1.2.10.8.11 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.

1.2.10.8.12 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.

1.2.10.8.13 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.

1.2.10.8.14 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.

1.2.11 Interrupt Support

1.2.11.1 Interrupt Events and Requests

The MMC/SD controller generates the interrupt requests described in Table 1-4. 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 MMC/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 1-4. 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.

1.2.12 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.

1.2.13 Emulation Considerations

The MMC/SD controller is not affected by emulation halt events (such as breakpoints).

1.3 Procedures for Common Operations

1.3.1 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.

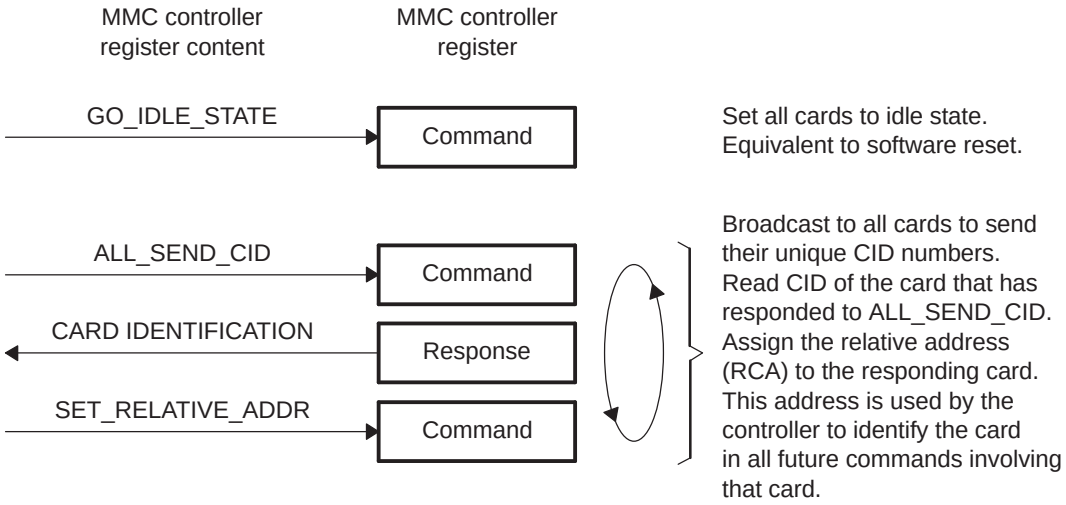
Only one card completes the response to ALL_SEND_CID at any one time. The absence of any response to ALL_SEND_CID indicates that all cards have been identified and configured.

The procedure for a card identification operation is:

1. Use MMCCMD1 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 MMCCMD1 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-MMCRSP7) 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 1-12.

Figure 1-12. Card Identification (Native MMC/SD Mode)



1.3.2 MMC/SD Mode Single-Block Write Operation Using CPU

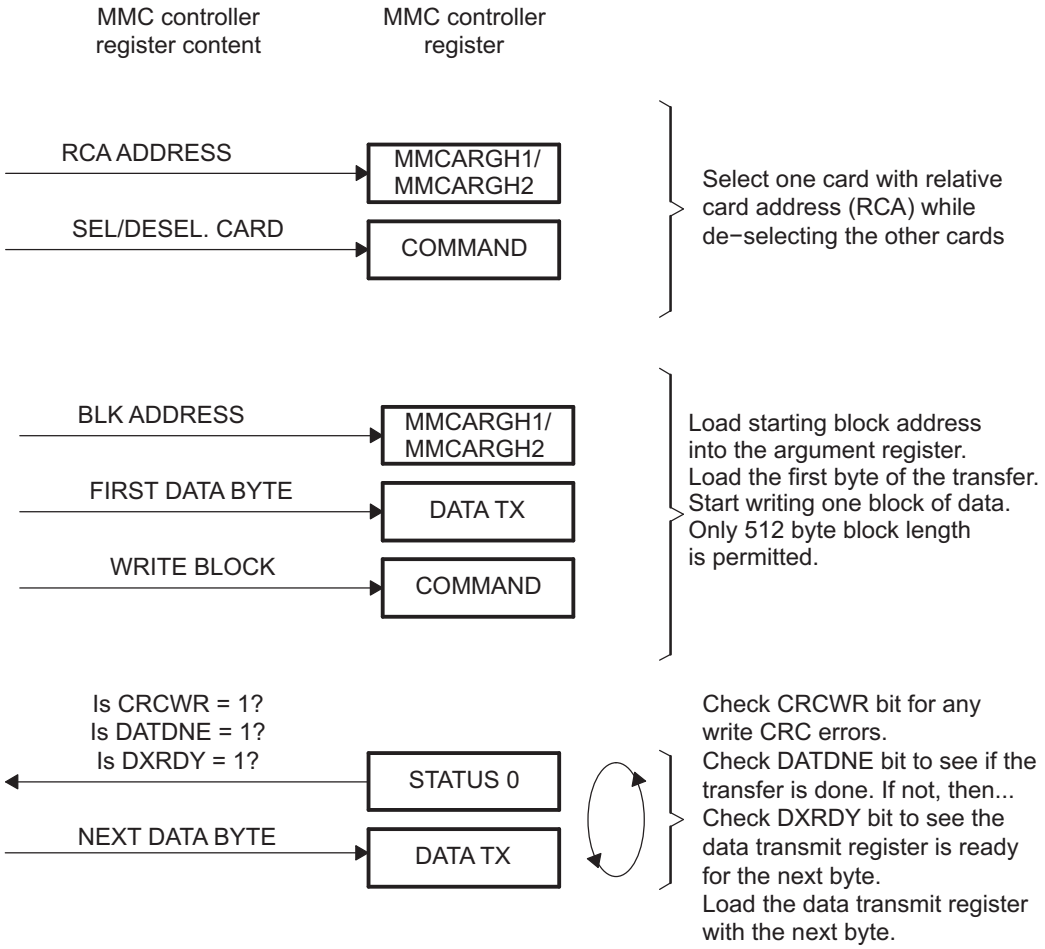
To perform a single-block write, the block length must be 512 bytes and the same length needs to be set in both the MMC/SD controller and the memory card. The procedure for this operation is:

1. Write the card's relative address to the MMC argument registers (MMCARG1/MMCARG2).
2. Use the MMCCMD1 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 MMCCMD1 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 MMCCMD1 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 1-13.

Figure 1-13. MMC/SD Mode Single-Block Write Operation



1.3.3 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 MMCCMD1 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 MMCCMD1 to send the WRITE _BLOCK command to the card.
10. Use MMCCMD2 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.

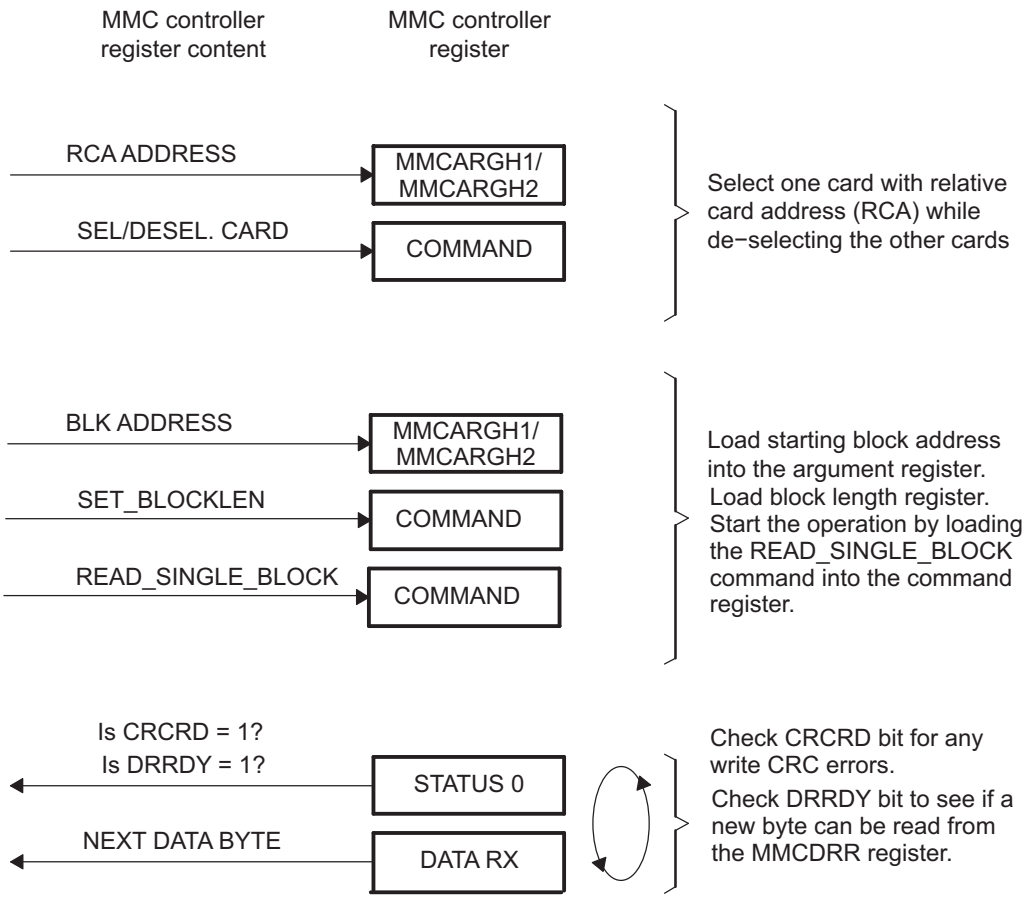
1.3.4 MMC/SD Mode Single-Block Read Operation Using CPU

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 MMCCMD1 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 MMCCMD1 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 MMCCMD1 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 1-14.

Figure 1-14. MMC/SD Mode Single-Block Read Operation



1.3.5 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 MMCCMD1 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 MMCCMD1 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.

1.3.6 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.

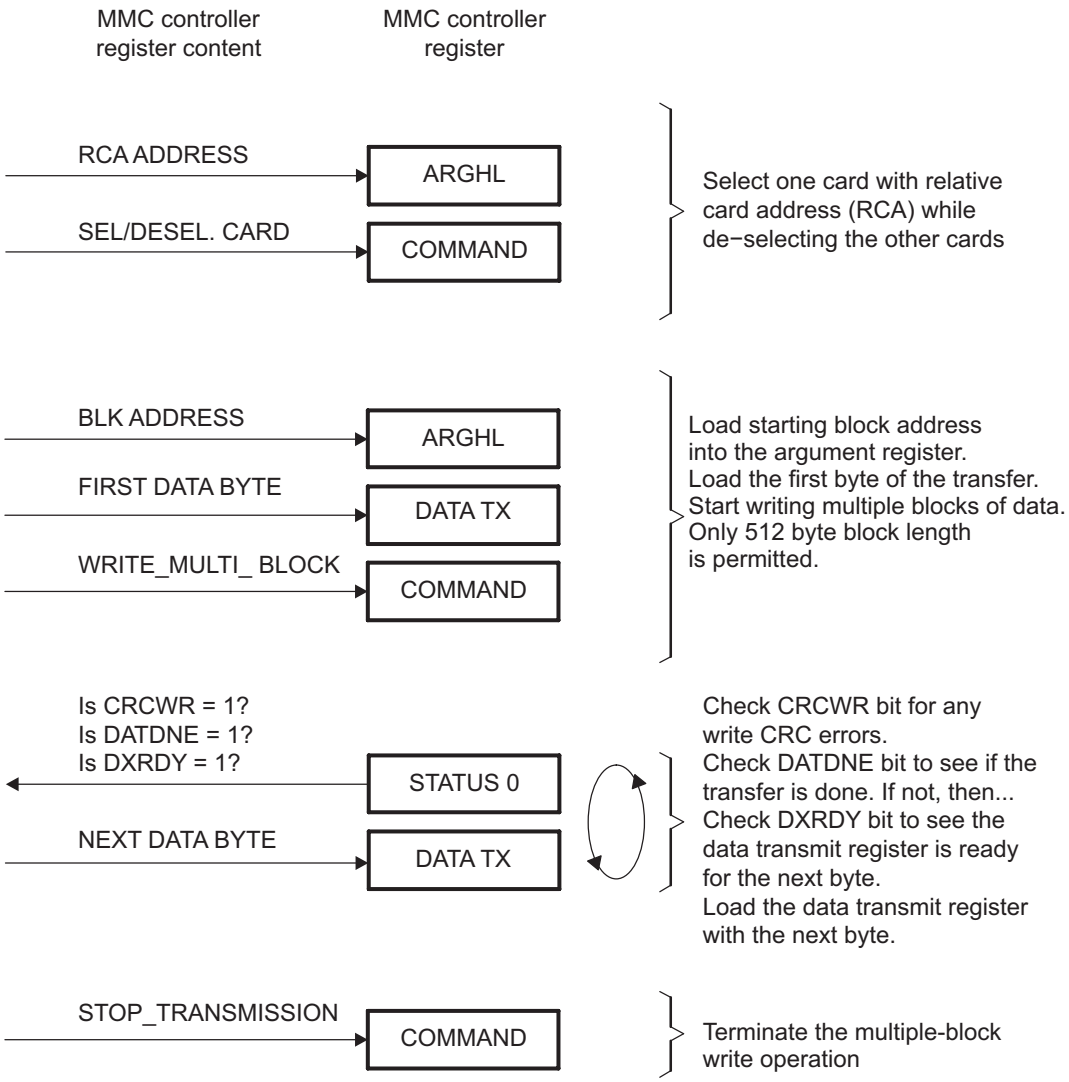
NOTE: The procedure in this section uses a STOP_TRANSMISSION command to end the block transfer. This assumes that the value in the MMC number of blocks counter register (MMCNBLK) is 0. A multiple-block operation terminates itself if you load MMCNBLK with the exact number of blocks you want transferred.

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 MMCCMD1 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 MMCCMD1 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 MMCCMD1 to send the STOP_TRANSMISSION command.

The sequence of events in this operation is shown in Figure 1-15.

Figure 1-15. MMC/SD Multiple-Block Write Operation



1.3.7 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 MMCCMD1 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 MMCCMD1 to send the WRITE_MULTI_BLOCK command to the card.

10. Use MMCCMD2 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 MMCCMD1 to send the STOP_TRANSMISSION command.

1.3.8 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.

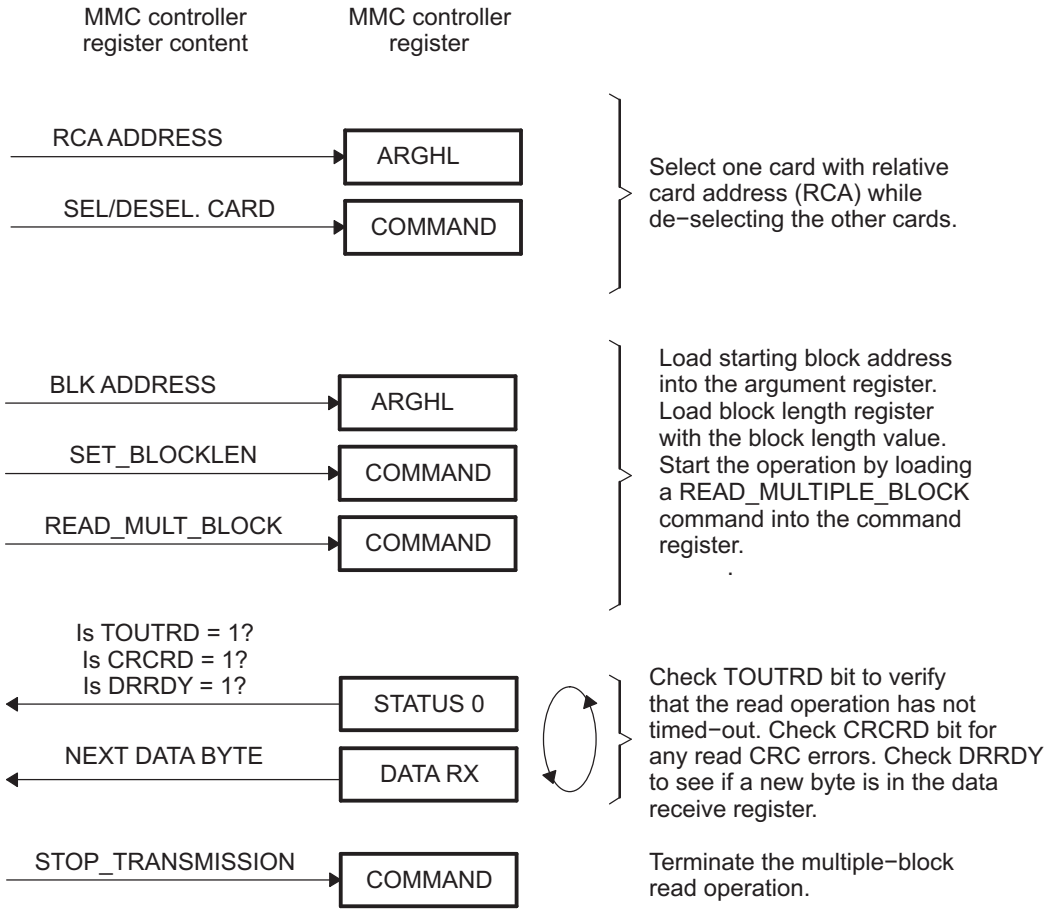
NOTE: The procedure in this section uses a STOP_TRANSMISSION command to end the block transfer. This assumes that the value in the MMC number of blocks counter register (MMCNBLK) is 0. A multiple-block operation terminates itself if you load MMCNBLK with the exact number of blocks you want transferred.

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 MMCCMD1 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 MMCCMD1 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 MMCCMD1 to send the STOP_TRANSMISSION command.

The sequence of events in this operation is shown in Figure 1-16.

Figure 1-16. MMC/SD Mode Multiple-Block Read Operation



1.3.9 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 MMCCMD1 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 MMCCMD1 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 MMCCMD1 to send the STOP_TRANSMISSION command.

1.3.10 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/SD 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 MMCSD 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 MMCSD clock rate up to 50 MHz.
13. Follow normal Read/Write operation.

1.3.11 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.

1.3.11.1 SDIO Control Register (SDIOCTL)

The SDIO card control register (SDIOCTL) is used to configure the read wait operation using the SD_DATA2 line.

1.3.11.2 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.

1.3.11.3 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.

1.4 Registers

Table 1-5 list the memory-mapped registers associated with the two Multimedia Card/Secure Digital 0 (MMC/SD0) controllers and Table 1-6 list the memory-mapped registers associated with the two Multimedia Card/Secure Digital 1 (MMC/SD1) controllers. Note that the CPU accesses all peripheral registers through its I/O space. All other register addresses not listed should be considered as reserved locations and the register contents should not be modified.

Table 1-5. Multimedia Card/Secure Digital 0 (MMC/SD0) Card Controller Registers

CPU Word Address	Acronym	Register Description	Section
3A00h	MMCCTL	MMC Control Register	Section 1.4.1
3A04h	MMCCLK	MMC Memory Clock Control Register	Section 1.4.2
3A08h	MMCST0	MMC Status Register 0	Section 1.4.3
3A0Ch	MMCST1	MMC Status Register 1	Section 1.4.4
3A10h	MMCIM	MMC Interrupt Mask Register	Section 1.4.5
3A14h	MMCTOR	MMC Response Time-Out Register	Section 1.4.6
3A18h	MMCTOD	MMC Data Read Time-Out Register	Section 1.4.7
3A1Ch	MMCBLEN	MMC Block Length Register	Section 1.4.8
3A20h	MMCNBLK	MMC Number of Blocks Register	Section 1.4.9
3A24h	MMCNBLC	MMC Number of Blocks Counter Register	Section 1.4.10
3A28h	MMCDRR1	MMC Data Receive Register 1	Section 1.4.11
3A29h	MMCDRR2	MMC Data Receive Register 2	Section 1.4.11
3A2Ch	MMCDXR1	MMC Data Transmit Register 1	Section 1.4.12
3A2Dh	MMCDXR2	MMC Data Transmit Register 2	Section 1.4.12
3A30h	MMCCMD1	MMC Command Register 1	Section 1.4.13
3A31h	MMCCMD2	MMC Command Register 2	Section 1.4.13
3A34h	MMCARG1	MMC Argument Register 1	Section 1.4.14
3A35h	MMCARG2	MMC Argument Register 2	Section 1.4.14
3A38h	MMCRSP0	MMC Response Register 0	Section 1.4.15
3A39h	MMCRSP1	MMC Response Register 1	Section 1.4.15
3A3Ch	MMCRSP2	MMC Response Register 2	Section 1.4.15
3A3Dh	MMCRSP3	MMC Response Register 3	Section 1.4.15
3A40h	MMCRSP4	MMC Response Register 4	Section 1.4.15
3A41h	MMCRSP5	MMC Response Register 5	Section 1.4.15
3A44h	MMCRSP6	MMC Response Register 6	Section 1.4.15
3A45h	MMCRSP7	MMC Response Register 7	Section 1.4.15
3A48h	MMCDRSP	MMC Data Response Register	Section 1.4.16
3A50h	MMCCIDX	MMC Command Index Register	Section 1.4.17
3A64h	SDIOCTL	SDIO Control Register	Section 1.4.18
3A68h	SDIOST0	SDIO Status Register 0	Section 1.4.19
3A6Ch	SDIOIEN	SDIO Interrupt Enable Register	Section 1.4.20
3A70h	SDIOIST	SDIO Interrupt Status Register	Section 1.4.21
3A74h	MMCFIFOCTL	MMC FIFO Control Register	Section 1.4.22

Table 1-6. Multimedia Card/Secure Digital 1 (MMC/SD1) Card Controller Registers

CPU Word Address	Acronym	Register Description	Section
3B00h	MMCCTL	MMC Control Register	Section 1.4.1
3B04h	MMCCLK	MMC Memory Clock Control Register	Section 1.4.2
3B08h	MMCST0	MMC Status Register 0	Section 1.4.3
3B0Ch	MMCST1	MMC Status Register 1	Section 1.4.4
3B10h	MMCIM	MMC Interrupt Mask Register	Section 1.4.5
3B14h	MMCTOR	MMC Response Time-Out Register	Section 1.4.6
3B18h	MMCTOD	MMC Data Read Time-Out Register	Section 1.4.7
3B1Ch	MMCBLEN	MMC Block Length Register	Section 1.4.8
3B20h	MMCNBLK	MMC Number of Blocks Register	Section 1.4.9
3B24h	MMCNBLC	MMC Number of Blocks Counter Register	Section 1.4.10
3B28h	MMCDRR1	MMC Data Receive Register 1	Section 1.4.11
3B29h	MMCDRR2	MMC Data Receive Register 2	Section 1.4.11
3B2Ch	MMCDXR1	MMC Data Transmit Register 1	Section 1.4.12
3B2Dh	MMCDXR2	MMC Data Transmit Register 2	Section 1.4.12
3B30h	MMCCMD1	MMC Command Register 1	Section 1.4.13
3B31h	MMCCMD2	MMC Command Register 2	Section 1.4.13
3B34h	MMCARG1	MMC Argument Register 1	Section 1.4.14
3B35h	MMCARG2	MMC Argument Register 2	Section 1.4.14
3B38h	MMCRSP0	MMC Response Register 0	Section 1.4.15
3B39h	MMCRSP1	MMC Response Register 1	Section 1.4.15
3B3Ch	MMCRSP2	MMC Response Register 2	Section 1.4.15
3B3Dh	MMCRSP3	MMC Response Register 3	Section 1.4.15
3B40h	MMCRSP4	MMC Response Register 4	Section 1.4.15
3B41h	MMCRSP5	MMC Response Register 5	Section 1.4.15
3B44h	MMCRSP6	MMC Response Register 6	Section 1.4.15
3B45h	MMCRSP7	MMC Response Register 7	Section 1.4.15
3B48h	MMCDRSP	MMC Data Response Register	Section 1.4.16
3B50h	MMCCIDX	MMC Command Index Register	Section 1.4.17
3B64h	SDIOCTL	SDIO Control Register	Section 1.4.18
3B68h	SDIOST0	SDIO Status Register 0	Section 1.4.19
3B6Ch	SDIOIEN	SDIO Interrupt Enable Register	Section 1.4.20
3B70h	SDIOIST	SDIO Interrupt Status Register	Section 1.4.21
3B74h	MMCFIFOCTL	MMC FIFO Control Register	Section 1.4.22

1.4.1 MMC Control Register (MMCCTL)

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. MMCCTL is shown in Figure 1-17 and described in Table 1-7.

Figure 1-17. MMC Control Register (MMCCTL)

15	11	10	9	8
Reserved		PERMDX	PERMDR	Reserved
R-0		R/W-0	R/W-0	R-0

7	6	5	3	2	1	0
DATEG		Reserved		WIDTH	CMDRST	DATRST
R/W-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 1-7. MMC Control Register (MMCCTL) Field Descriptions

Bit	Field	Value	Description
15-11	Reserved	0	Reserved.
10	PERMDX		Endian select enable when writing.
		0	Little Endian is selected.
		1	Big Endian is selected.
9	PERMDR		Endian select enable when reading.
		0	Little Endian is selected.
		1	Big Endian is selected.
8	Reserved	0	Reserved.
7-6	DATEG	0-3h	DAT3 edge detection select.
		0	DAT3 edge detection is disabled.
		1h	DAT3 rising-edge detection is enabled.
		2h	DAT3 falling-edge detection is enabled.
		3h	DAT3 rising-edge and falling-edge detections are enabled.
5-3	Reserved	0	Reserved.
2	WIDTH		Data bus width (MMC mode only).
		0	Data bus has 1 bit (only DAT0 is used).
		1	Data bus has 4 bits (all DAT0-3 are used).
1	CMDRST		CMD logic reset.
		0	CMD line portion is enabled.
		1	CMD line portion is disabled and in reset state.
0	DATRST		DAT logic reset.
		0	DAT line portion is enabled.
		1	DAT line portion is disabled and in reset state.

1.4.2 MMC Memory Clock Control Register (MMCCLK)

The MMC memory clock control register (MMCCLK) is used to:

- Select whether the CLK pin is enabled or disabled (CLKEN bit).
- 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. For more details about clock generation, see Section 1.2.1.

MMCCLK is shown in Figure 1-18 and described in Table 1-8.

Figure 1-18. MMC Memory Clock Control Register (MMCCLK)

15	9	8	7	0
Reserved		CLKEN	CLKRT	
R-0		R/W-0	R/W−FFh	

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

Table 1-8. MMC Memory Clock Control Register (MMCCLK) Field Descriptions

Bit	Field	Value	Description
15-9	Reserved	0	Reserved.
8	CLKEN		CLK pin enable.
		0	CLK pin is disabled and fixed low.
		1	The CLK pin is enabled; it shows the memory clock signal.
7-0	CLKRT	0−FFh	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)*

1.4.3 MMC Status Register 0 (MMCST0)

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 described for them in Table 1-9, or in response to a hardware reset.

MMCST0 is shown in Figure 1-19 and described in Table 1-9.

Figure 1-19. MMC Status Register 0 (MMCST0)

15			13	12	11	10	9	8
Reserved				TRNDNE	DATED	DRRDY	DXRDY	Reserved
R-0				RC-0	RC-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0	
CRCRS	CRCRD	CRCWR	TOUTRS	TOUTRD	RSPDNE	BSYDNE	DATDNE	
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	

LEGEND: R = Read only; RC = Cleared to 0 when read; -*n* = value after reset

Table 1-9. MMC Status Register 0 (MMCST0) Field Descriptions

Bit	Field	Value	Description
15-13	Reserved	0	Reserved. Any writes to these bit(s) must always have a value of 0.
12	TRNDNE		Transfer done.
		0	No data transfer is done.
		1	Data transfer of specified length is done.
11	DATED		DAT3 edge detected. DATED is cleared when read by CPU.
		0	A DAT3 edge has not been detected.
		1	A DAT3 edge has been detected.
10	DRRDY		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).
		0	MMCDRR is not ready.
		1	MMCDRR is ready. New data has arrived and can be read by the CPU or by the DMA controller.
9	DXRDY		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).
		0	MMCDXR is not ready.
		1	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	0	Reserved.
7	CRCRS		Response CRC error.
		0	A response CRC error has not been detected.
		1	A response CRC error has been detected.
6	CRCRD		Read-data CRC error.
		0	A read-data CRC error has not been detected.
		1	A read-data CRC error has been detected.
5	CRCWR		Write-data CRC error.
		0	A write-data CRC error has not been detected.
		1	A write-data CRC error has been detected.

Table 1-9. MMC Status Register 0 (MMCST0) Field Descriptions (continued)

Bit	Field	Value	Description
4	TOUTRS		Response time-out event.
		0	A response time-out event has not occurred.
		1	A time-out event has occurred while the MMC controller was waiting for a response to a command.
3	TOUTRD		Read-data time-out event.
		0	A read-data time-out event has not occurred.
		1	A time-out event has occurred while the MMC controller was waiting for data.
2	RSPDNE		Command/response done.
		0	No response is received.
		1	Command has been sent without response or response has been received for the command sent.
1	BSYDNE		Busy done.
		0	No busy releasing is done.
		1	Released from busy state or expected busy is not detected.
0	DATDNE		Data done.
		0	The data has not been fully transmitted.
		1	The data has been fully transmitted.

NOTE: 1) 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.

2) Bit 12 (TRNDNE) indicates that the last byte of a transfer has been completed. Bit 0 (DATDNE) occurs at end of a transfer but not until the CRC check and programming has been completed.

1.4.4 MMC Status Register 1 (MMCST1)

The MMC status register 1 (MMCST1) records specific events or errors. There are no interrupts associated with these events or errors. MMCST1 is shown in Figure 1-20 and described in Table 1-10.

Figure 1-20. MMC Status Register 1 (MMCST1)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
Reserved	FIFOFUL	FIFOEMP	DAT3ST	DRFUL	DXEMP	CLKSTP	BUSY
R-0	R-0	R-0	R-0	R-0	R-0	R-1	R-0

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

Table 1-10. MMC Status Register 1 (MMCST1) Field Descriptions

Bit	Field	Value	Description
15-7	Reserved	0	Reserved.
6	FIFOFUL		FIFO is full.
		0	FIFO is not full.
		1	FIFO is full.
5	FIFOEMP		FIFO is empty.
		0	FIFO is not empty.
		1	FIFO is empty.
4	DAT3ST		DAT3 status.
		0	The signal level on the DAT3 pin is a logic-low level.
		1	The signal level on the DAT3 pin is a logic-high level.
3	DRFUL		Data receive register (MMCDRR) is full.
		0	A data receive register full condition is not detected.
		1	A data receive register full condition is detected.
2	DXEMP		Data transmit register (MMCDXR) is empty.
		0	A data transmit register empty condition is not detected. The data transmit shift register is not empty.
		1	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		Clock stop status.
		0	CLK is active. The memory clock signal is being driven on the pin.
		1	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		Busy.
		0	No busy signal is detected.
		1	A busy signal is detected (the memory card is busy).

1.4.5 MMC Interrupt Mask Register (MMCIM)

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. MMCIM is shown in Figure 1-21 and described in Table 1-11.

Figure 1-21. MMC Interrupt Mask Register (MMCIM)

15	13	12	11	10	9	8
Reserved		ETRNDNE	EDATED	EDRRDY	EDXRDY	Reserved
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R-0

7	6	5	4	3	2	1	0
ECRCRS	ECRCRD	ECRCWR	ETOUTRS	ETOUTRD	ERSPDNE	EBSYDNE	EDATDNE
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 1-11. MMC Interrupt Mask Register (MMCIM) Field Descriptions

Bit	Field	Value	Description
15-13	Reserved	0	Reserved.
12	ETRNDNE		Transfer done (TRNDNE) interrupt enable.
		0	Transfer done interrupt is disabled.
		1	Transfer done interrupt is enabled.
11	EDATED		DAT3 edge detect (DATED) interrupt enable.
		0	DAT3 edge detect interrupt is disabled.
		1	DAT3 edge detect interrupt is enabled.
10	EDRRDY		Data receive register ready (DRRDY) interrupt enable.
		0	Data receive register ready interrupt is disabled.
		1	Data receive register ready interrupt is enabled.
9	EDXRDY		Data transmit register (MMCDXR) ready interrupt enable.
		0	Data transmit register ready interrupt is disabled.
		1	Data transmit register ready interrupt is enabled.
8	Reserved	0	Reserved.
7	ECRCRS		Response CRC error (CRCRS) interrupt enable.
		0	Response CRC error interrupt is disabled.
		1	Response CRC error interrupt is enabled.
6	ECRCRD		Read-data CRC error (CRCRD) interrupt enable.
		0	Read-data CRC error interrupt is disabled.
		1	Read-data CRC error interrupt is enabled.
5	ECRCWR		Write-data CRC error (CRCWR) interrupt enable.
		0	Write-data CRC error interrupt is disabled.
		1	Write-data CRC error interrupt is disabled.
4	ETOUTRS		Response time-out event (TOUTRS) interrupt enable.
		0	Response time-out event interrupt is disabled.
		1	Response time-out event interrupt is enabled.
3	ETOUTRD		Read-data time-out event (TOUTRD) interrupt enable.
		0	Read-data time-out event interrupt is disabled.
		1	Read-data time-out event interrupt is enabled.
2	ERSPDNE		Command/response done (RSPDNE) interrupt enable.
		0	Command/response done interrupt is disabled.
		1	Command/response done interrupt is enabled.

Table 1-11. MMC Interrupt Mask Register (MMCIM) Field Descriptions (continued)

Bit	Field	Value	Description
1	EBSYDNE		Busy done (BSYDNE) interrupt enable.
		0	Busy done interrupt is disabled.
		1	Busy done interrupt is enabled.
0	EDATDNE		Data done (DATDNE) interrupt enable.
		0	Data done interrupt is disabled.
		1	Data done interrupt is enabled.

1.4.6 MMC Response Time-Out Register (MMCTOR)

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.

MMCTOR is shown in Figure 1-22 and described in Table 1-12.

Figure 1-22. MMC Response Time-Out Register (MMCTOR)

15	8	7	0
TOD_23_16		TOR	
R/W-0		R/W-0	

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

Table 1-12. MMC Response Time-Out Register (MMCTOR) Field Descriptions

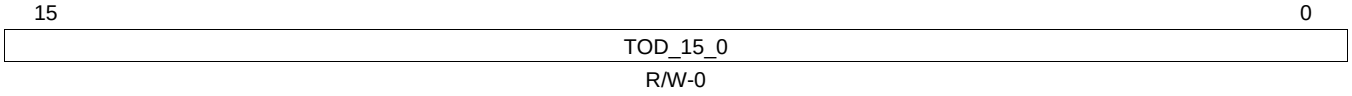
Bit	Field	Value	Description
15-8	TOD_23_16	0-1Fh	Data read time-out count upper 8 bits. Used in conjunction with the TOD_15_0 bits in MMCTOD to form a 24-bit count. See MMCTOD (Section 1.4.7).
7-0	TOR	0-FFh	Time-out count for response.
		0	No time out.
		1-FFh	1 CLK memory clock cycle to 255 CLK memory clock cycles.

1.4.7 MMC Data Read Time-Out Register (MMCTOD)

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.

MMCTOD is shown in Figure 1-23 and described in Table 1-13.

Figure 1-23. MMC Data Read Time-Out Register (MMCTOD)



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

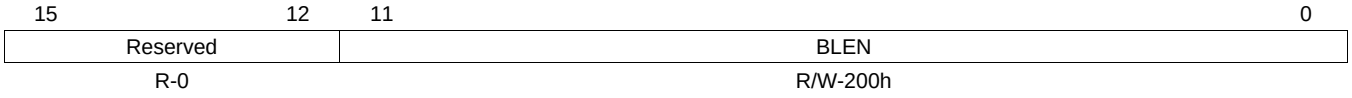
Table 1-13. MMC Data Read Time-Out Register (MMCTOD) Field Descriptions

Bit	Field	Value	Description
15-0	TOD_15_0	0-1F FFFFh	Data read time-out count. Used in conjunction with the TOD_23_16 bits in MMCTOR to form a 24-bit count. See MMCTOR (Section 1.4.6).
		0	No time out.
		1-FFFFh	1 CLK clock cycle to 64,000 CLK clock cycles. When used in conjunction with TOD_23_16, the value range will be 1-1F FFFF clock cycles.

1.4.8 MMC Block Length Register (MMCBLEN)

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. MMCBLEN is shown in Figure 1-24 and described in Table 1-14.

Figure 1-24. MMC Block Length Register (MMCBLEN)



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

Table 1-14. MMC Block Length Register (MMCBLEN) Field Descriptions

Bit	Field	Value	Description
15-12	Reserved	0	Reserved.
11−0	BLEN	1h−FFFh	Block length. This field is used to set the block length, which is the byte count of a data block. The value 0 is prohibited.

NOTE: 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.

1.4.9 MMC Number of Blocks Register (MMCNBLK)

The MMC number of blocks register (MMCNBLK) specifies the number of blocks for a multiple-block transfer. MMCNBLK is shown in Figure 1-25 and described in Table 1-15.

Figure 1-25. MMC Number of Blocks Register (MMCNBLK)

15	0
NBLK	
R/W-0	

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

Table 1-15. MMC Number of Blocks Register (MMCNBLK) Field Descriptions

Bit	Field	Value	Description
15-0	NBLK	0-FFFFh	Number of blocks. This field is used to set the total number of blocks to be transferred.
		0	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).
		1h−FFFFh	*n* 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).

1.4.10 MMC Number of Blocks Counter Register (MMCNBLC)

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. MMCNBLC is shown in Figure 1-26 and described in Table 1-16.

Figure 1-26. MMC Number of Blocks Counter Register (MMCNBLC)

15	0
NBLC	
R-FFFFh	

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

Table 1-16. MMC Number of Blocks Counter Register (MMCNBLC) Field Descriptions

Bit	Field	Value	Description
15-0	NBLC	0−FFFFh	Read this field to determine the number of blocks remaining to be transferred.

1.4.11 MMC Data Receive Register (MMCDRR1) and (MMCDRR2)

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. MMCDRR1 is shown in Figure 1-27 and described in Table 1-17. MMCDRR2 is shown in Figure 1-28 and described in Table 1-18.

Figure 1-27. MMC Data Receive Register (MMCDRR1)

15	0
DRR1	
R/W-0	

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

Table 1-17. MMC Data Receive Register (MMCDRR1) Field Descriptions

Bit	Field	Value	Description
15-0	DRR1	0-FFFFh	Data receive 1.

Figure 1-28. MMC Data Receive Register (MMCDRR2)

15	0
DRR2	
R/W-0	

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

Table 1-18. MMC Data Receive Register (MMCDRR2) Field Descriptions

Bit	Field	Value	Description
15-0	DRR2	0-FFFFh	Data receive 2.

1.4.12 MMC Data Transmit Registers (MMCDXR1) and (MMCDXR2)

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. MMCDXR1 is shown in Figure 1-29 and described in Table 1-19. MMCDXR2 is shown in Figure 1-30 and described in Table 1-20.

Figure 1-29. MMC Data Transmit Register (MMCDXR)

15	0
DXR1	
R/W-0	

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

Table 1-19. MMC Data Transmit Register (MMCDXR) Field Descriptions

Bit	Field	Value	Description
15-0	DXR1	0-FFFFh	Data transmit 1.

Figure 1-30. MMC Data Transmit Register (MMCDXR2)

15	0
DXR2	
R/W-0	

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

Table 1-20. MMC Data Transmit Register (MMCDXR2) Field Descriptions

Bit	Field	Value	Description
15-0	DXR2	0-FFFFh	Data transmit 2.

1.4.13 MMC Command Registers (MMCCMD1) and (MMCCMD2)

NOTE: 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) specifies 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. MMCCMD is shown in Figure 1-31 and described in Table 1-21.

When the CPU writes to MMCCMD1 and 2, the MMC controller sends the programmed commands, including any arguments in the MMCARG1/MMCARG2 registers. For the format of a command (index, arguments, and other bits), see Figure 1-33 and Table 1-23.

NOTE: Writes to MMCCMD2 should only occur after a write command has been written to the MMCCMD1 register for DMA data transfers.

Figure 1-31. MMC Command Register 1 (MMCCMD1)

15	14	13	12	11	10	9	8
DCLR	INITCK	WDATX	STRMTP	DTRW	RSPFMT		BSYEXP
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0		R/W-0
7	6	5					0
PPLEN	Reserved	CMD					
R/W-0	R-0	R/W-0					

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

Table 1-21. MMC Command Register 1 (MMCCMD1) Field Descriptions

Bit	Field	Value	Description
15	DCLR		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.
		0	Do not clear DRRDY and DXRDY bits in MMCST0.
		1	Clear DRRDY and DXRDY bits in MMCST0.
14	INITCK		Initialization clock cycles.
		0	Do not insert initialization clock cycles.
		1	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		Data transfer indicator.
		0	There is no data transfer associated with the command being sent.
		1	There is data transfer associated with the command being sent.
12	STRMTP		Stream transfer enable.
		0	If WDATX = 1, 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).
		1	If WDATX = 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		Data transfer write enable.
		0	If WDATX = 1, the data transfer is a read operation.
		1	If WDATX = 1, the data transfer is a write operation.

Table 1-21. MMC Command Register 1 (MMCCMD1) Field Descriptions (continued)

Bit	Field	Value	Description
10-9	RSPFMT	0-3h	Response format (expected type of response to the command).
		0	No response.
		1h	R1, R4, R5, or R6 response. 48 bits with CRC.
		2h	R2 response. 136 bits with CRC.
		3h	R3 response. 48 bits with no CRC.
8	BSYEXP		Busy expected. If an R1b (R1 with busy) response is expected, set RSPFMT = 1h and BSYEXP = 1.
		0	A busy signal is not expected.
		1	A busy signal is expected.
7	PPLEN		Push pull enable.
		0	Push pull driver of CMD line is disabled (open drain).
		1	Push pull driver of CMD line is enabled.
6	Reserved	0	Reserved.
5-0	CMD	0-3Fh	Command index. This field contains the command index for the command to be sent to the memory card.

Figure 1-32. MMC Command Register 2 (MMCCMD2)

15	1	0
Reserved		DMATRIG
R-0		W-0

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

Table 1-22. MMC Command Register 2 (MMCCMD2) Field Descriptions

Bit	Field	Value	Description
15-1	Reserved	0	Reserved.
0	DMATRIG		Generate a DMA event once to trigger the first DMA transfer for data write operations (subsequent DMA events are automatically generated).
		0	DMA transfer event generation is disabled.
		1	Trigger a DMA transfer event for the first data transfer to the FIFO.

Figure 1-33. Command Format

47	46	45 40	39 24
Start	Transmission	Command index	Argument, high part

23 8	7 1	0
Argument, low part	CRC7	End

Table 1-23. Command Format

Bit Position of Command	Register	Description
47	-	Start bit
46	-	Transmission bit
45-40	MMCCMD(5-0)	Command index (CMD)
39-24	MMCARG1	Argument, high part
23-8	MMCARG2	Argument, low part
7-1	-	CRC7
0	-	End bit

1.4.14 MMC Argument Registers (MMCARG1) and (MMCARG2)

NOTE: Do not modify the MMC argument registers (MMCARG1 and MMCARG2) while they are being used for an operation.

The MMC argument registers (MMCARG1 and MMCARG2) specifies 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. MMCARG1 is shown in Figure 1-34 and described in Table 1-24. MMCARG2 is shown in Figure 1-35 and described in Table 1-25 For the format of a command, see Figure 1-33 and Table 1-23.

Figure 1-34. MMC Data Transmit Register (MMCARG1)

15	0
ARG1	
R/W-0	

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

Table 1-24. MMC Argument Register (MMCARG1) Field Descriptions

Bit	Field	Value	Description
15-0	ARG1	0-FFFFh	Argument, high and low parts.

Figure 1-35. MMC Data Transmit Register (MMCARG2)

15	0
ARG2	
R/W-0	

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

Table 1-25. MMC Argument Register (MMCARG2) Field Descriptions

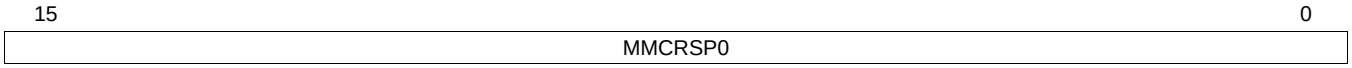
Bit	Field	Value	Description
15-0	ARG2	0-FFFFh	Argument, high and low parts.

1.4.15 MMC Response Registers (MMCRSP0-MMCRSP7)

Each command has a preset response type. When the MMC controller receives a response, it is stored in some or all of the 4 MMC response registers (MMCRSP0-MMCRSP7). The response registers are updated as the responses arrive, even if the CPU has not read the previous contents.

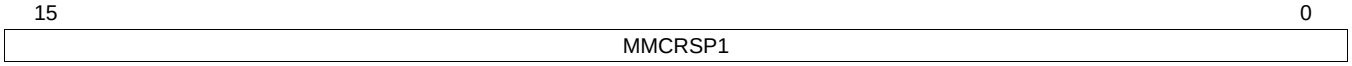
As shown in Figure 1-36 through Figure 1-43, each of the MMC response registers holds up to 16 bits. Table 1-26 and Table 1-27 show the format for each type of response and which MMC response registers are used for the bits of the response. The first byte of the response is a command index byte and is stored in the MMC command index register (MMCCIDX).

Figure 1-36. MMC Response Register 0 (MMCRSP0)



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

Figure 1-37. MMC Response Register 1 (MMCRSP1)



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

Figure 1-38. MMC Response Register 2 (MMCRSP2)

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

Figure 1-39. MMC Response Register 3 (MMCRSP3)

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

Figure 1-40. MMC Response Register 4 (MMCRSP4)

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

Figure 1-41. MMC Response Register 5 (MMCRSP5)

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

Figure 1-42. MMC Response Register 6 (MMCRSP6)

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

Figure 1-43. MMC Response Register 7 (MMCRSP7)

15	0
MMCRSP7	
R-0	

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

Table 1-26. R1, R3, R4, R5, or R6 Response (48 Bits)

Bit Position of Response	Register
47-40	MMCCIDX
39-24	MMCRSP7
23-8	MMCRSP6
7-0	MMCRSP5
-	MMCRSP4-0

Table 1-27. R2 Response (136 Bits)

Bit Position of Response	Register
135-128	MMCCIDX
127-112	MMCRSP7
111-96	MMCRSP6
95-80	MMCRSP5
79-64	MMCRSP4
63-48	MMCRSP3
47-37	MMCRSP2
31-16	MMCRSP1
15-0	MMCRSP0

1.4.16 MMC Data Response Register (MMCDRSP)

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, shown in Figure 1-44 and described in Table 1-28.

Figure 1-44. MMC Data Response Register (MMCDRSP)

15	8	7	0
Reserved		DRSP	
R-0		R/W-0	

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

Table 1-28. MMC Data Response Register (MMCDRSP) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7-0	DRSP	0-FFh	During a write operation (see Section 1.2.4.1), the CRC status token is stored in DRSP.

1.4.17 MMC Command Index Register (MMCCIDX)

The MMC command index register (MMCCIDX) stores the first byte of a response from a memory card. Table 1-26 and Table 1-27 show the format for each type of response. MMCCIDX is shown in Figure 1-45 and described in Table 1-29.

Figure 1-45. MMC Command Index Register (MMCCIDX)

15	8	7	6	5	0
Reserved		STRT	XMIT	CIDX	
R-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. MMC Command Index Register (MMCCIDX) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved.
7	STRT	0-1	Start bit. When the MMC controller receives a response, the start bit is stored in STRT.
6	XMIT	0-1	Transmission bit. When the MMC controller receives a response, the transmission bit is stored in XMIT.
5-0	CIDX	0-3Fh	Command index. When the MMC controller receives a response, the command index is stored in CIDX.

1.4.18 SDIO Control Register (SDIOCTL)

The SDIO control register (SDIOCTL) is shown in Figure 1-46 and described in Table 1-30.

Figure 1-46. SDIO Control Register (SDIOCTL)

15	2	1	0
Reserved		RDWTCR	RDWTRQ
R-0		R/W-0	R/W-0

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

Table 1-30. SDIO Control Register (SDIOCTL) Field Descriptions

Bit	Field	Value	Description
15-2	Reserved	0	Reserved.
1	RDWTCR		Read wait enable for CRC error. To end the read wait operation, write 0 to RDWTRQ. (No need to clear RDWTCR).
		0	Read wait is disabled.
		1	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		Read wait request. To end the read wait operation, write 0 to RDWTRQ.
		0	End read wait operation and release DAT[2].
		1	Set a read wait request. Read wait operation starts 2 clocks after the end of the read data block. MMC interface asserts low level on DAT[2] until RDWTRQ is cleared to 0.

1.4.19 SDIO Status Register 0 (SDIOST0)

The SDIO status register 0 (SDIOST0) is shown in Figure 1-47 and described in Table 1-31.

Figure 1-47. SDIO Status Register 0 (SDIOST0)

15	2	1	0
Reserved	RDWTST	INTPRD	DAT1
R-0	R-0	R-0	R-1

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

Table 1-31. SDIO Status Register 0 (SDIOST0) Field Descriptions

Bit	Field	Value	Description
15-3	Reserved	0	Reserved.
2	RDWTST		Read wait status.
		0	Read wait operation not in progress.
		1	Read wait operation in progress.
1	INTPRD		Interrupt period.
		0	Interrupt not in progress.
		1	Interrupt in progress.
0	DAT1		This bit reflects the external state of the SD_DATA1 pin.
		0	Logic-low level on the SD_DATA1 pin.
		1	Logic-high level on the SD_DATA1 pin.

1.4.20 SDIO Interrupt Enable Register (SDIOIEN)

The SDIO interrupt enable register (SDIOIEN) is shown in Figure 1-48 and described in Table 1-32.

Figure 1-48. SDIO Interrupt Enable Register (SDIOIEN)

15	2	1	0
Reserved		RWSEN	IOINTEN
R-0		R/W-0	R/W-0

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

Table 1-32. SDIO Interrupt Enable Register (SDIOIEN) Field Descriptions

Bit	Field	Value	Description
15-2	Reserved	0	Reserved.
1	RWSEN		Read wait interrupt enable.
		0	Read wait interrupt is disabled.
		1	Read wait interrupt is enabled.
0	IOINTEN		SDIO card interrupt enable.
		0	SDIO card interrupt is disabled.
		1	SDIO card interrupt is enabled.

1.4.21 SDIO Interrupt Status Register (SDIOIST)

The SDIO interrupt status register (SDIOIST) is shown in Figure 1-49 and described in Table 1-33.

Figure 1-49. SDIO Interrupt Status Register (SDIOIST)

15	2	1	0
Reserved		RWS	IOINT
R-0		R/W1C-0	R/W1C-0

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

Table 1-33. SDIO Interrupt Status Register (SDIOIST) Field Descriptions

Bit	Field	Value	Description
15-2	Reserved	0	Reserved.
1	RWS		Read wait interrupt status. Write a 1 to clear this bit.
		0	Read wait interrupt did not occur.
		1	Read wait interrupt occurred. Read wait operation starts and read wait interrupt is enabled (RWSEN = 1 in SDIOIEN).
0	IOINT		SDIO card interrupt status. Write a 1 to clear this bit.
		0	SDIO card interrupt did not occur.
		1	SDIO card interrupt occurred. SDIO card interrupt is detected and SDIO card interrupt is enabled (IOINTEN = 1 in SDIOIEN).

1.4.22 MMC FIFO Control Register (MMCFIFOCTL)

The MMC FIFO control register (MMCFIFOCTL) is shown in Figure 1-50 and described in Table 1-34.

Figure 1-50. MMC FIFO Control Register (MMCFIFOCTL)

15	8
Reserved	
R-0	

7	5	4	3	2	1	0
Reserved		ACCWD		FIFOLEV	FIFODIR	FIFORST
R-0		R/W-0		R/W-0	R/W-0	W-0

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

Table 1-34. MMC FIFO Control Register (MMCFIFOCTL) Field Descriptions

Bit	Field	Value	Description
15-5	Reserved	0	Reserved
4-3	ACCWD	0-3h	Access width. Used by FIFO control to determine full/empty flag.
		0	CPU/DMA access width of 4 bytes
		1h	CPU/DMA access width of 3 bytes
		2h	CPU/DMA access width of 2 bytes
		3h	CPU/DMA access width of 1 byte
2	FIFOLEV		FIFO level. Sets the threshold level that determines when the DMA request and the FIFO threshold interrupt are triggered.
		0	DMA request every 128 bits sent/received.
		1	DMA request every 256 bits sent/received.
1	FIFODIR		FIFO direction. Determines if the FIFO is being written to or read from.
		0	Read from FIFO.
		1	Write to FIFO.
0	FIFORST		FIFO reset. Resets the internal state of the FIFO.
		0	FIFO reset is disabled.
		1	FIFO reset is enabled.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Communications and Telecom	www.ti.com/communications
DSP	dsp.ti.com	Computers and Peripherals	www.ti.com/computers
Clocks and Timers	www.ti.com/clocks	Consumer Electronics	www.ti.com/consumer-apps
Interface	interface.ti.com	Energy	www.ti.com/energy
Logic	logic.ti.com	Industrial	www.ti.com/industrial
Power Mgmt	power.ti.com	Medical	www.ti.com/medical
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Space, Avionics & Defense	www.ti.com/space-avionics-defense
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video and Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless-apps

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265