TMS320VC5509 DSP MultiMediaCard / SD Card Controller Reference Guide

Literature Number: SPRU593A
September 2007

Texas Instruments

Preface

Read This First

About This Manual

TMS320VC5509 digital signal processors (DSPs) in the TMS320C55x™ (C55x™) DSP generation contain a peripheral for controlling memory cards that conform to the MultiMediaCard™ standard or the SD (Secure Digital) Memory Card standard. This manual describes the features of this MultiMediaCard / SD card controller and how to use it. Throughout this document, the name of this peripheral is abbreviated as *MMC controller*.

Notational Conventions

This document uses the following conventions:

- In most cases, hexadecimal numbers are shown with the suffix h. For example, the following number is a hexadecimal 40 (decimal 64):

  40h

  Similarly, binary numbers often are shown with the suffix b. For example, the following number is the decimal number 4 shown in binary form:

  0100b

- If a signal or pin is active low, it has an overbar. For example, the $\overline{\text{RESET}}$ signal is active low.

- Bits and signals are sometimes referenced with the following notations:

Notation	Description	Example
Register(n−m)	Bits n through m of Register	R(15−0) represents the 16 least significant bits of register R.
Bus[n:m]	Signals n through m of Bus	A[21:1] represents signals 21 through 1 of bus A.

Related Documentation From Texas Instruments

The following documents describe the C55x devices and related support tools. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the keyword search box provided at www.ti.com.

TMS320VC5509 Fixed-Point Digital Signal Processor Data Manual (literature number SPRS163) describes the features of the TMS320VC5509 fixed-point DSP and provides signal descriptions, pinouts, electrical specifications, and timings for the device.

TMS320VC5509A Fixed-Point Digital Signal Processor Data Manual (literature number SPRS205) describes the features of the TMS320VC5509A fixed-point DSP and provides signal descriptions, pinouts, electrical specifications, and timings for the device.

TMS320C55x Technical Overview (literature number SPRU393) introduces the TMS320C55x DSPs, the latest generation of fixed-point DSPs in the TMS320C5000™ DSP platform. Like the previous generations, this processor is optimized for high performance and low-power operation. This book describes the CPU architecture, low-power enhancements, and embedded emulation features.

TMS320C55x DSP CPU Reference Guide (literature number SPRU371) describes the architecture, registers, and operation of the CPU for the TMS320C55x DSPs.

TMS320C55x DSP Peripherals Overview Reference Guide (literature number SPRU317) introduces the peripherals, interfaces, and related hardware that are available on TMS320C55x DSPs.

TMS320C55x DSP Algebraic Instruction Set Reference Guide (literature number SPRU375) describes the TMS320C55x DSP algebraic instructions individually. It also includes a summary of the instruction set, a list of the instruction opcodes, and a cross-reference to the mnemonic instruction set.

TMS320C55x DSP Mnemonic Instruction Set Reference Guide (literature number SPRU374) describes the TMS320C55x DSP mnemonic instructions individually. It also includes a summary of the instruction set, a list of the instruction opcodes, and a cross-reference to the algebraic instruction set.

TMS320C55x Optimizing C/C++ Compiler User's Guide (literature number SPRU281) describes the TMS320C55x™ C/C++ Compiler. This C/C++ compiler accepts ISO standard C and C++ source code and produces assembly language source code for TMS320C55x devices.

TMS320C55x Assembly Language Tools User's Guide (literature number SPRU280) describes the assembly language tools (assembler, linker, and other tools used to develop assembly language code), assembler directives, macros, common object file format, and symbolic debugging directives for TMS320C55x devices.

TMS320C55x DSP Programmer's Guide (literature number SPRU376) describes ways to optimize C and assembly code for the TMS320C55x DSPs and explains how to write code that uses special features and instructions of the DSPs.

Trademarks

Contents

Figures

Tables

Chapter 1

Features and Operation

This chapter describes the core functional behavior of the MultiMediaCard / SD card controller. The controller supports both the MultiMediaCard protocol and the SD (Secure Digital) Memory Card protocol, but for brevity, this document refers to the peripheral as the MMC controller.

Topic **Page**

1.1 Overview

The MMC controller includes:

- ❑ Support for a MultiMediaCard (MMC) or a Secure Digital Memory Card (SD card)
- ❑ A programmable frequency for the operation of the MMC controller
- ❑ A programmable frequency for the clock that controls the timing of transfers between the MMC controller and the memory card

The MMC controller does not support the SPI protocol. It cannot communicate with a memory card that is in its SPI mode.

Note:

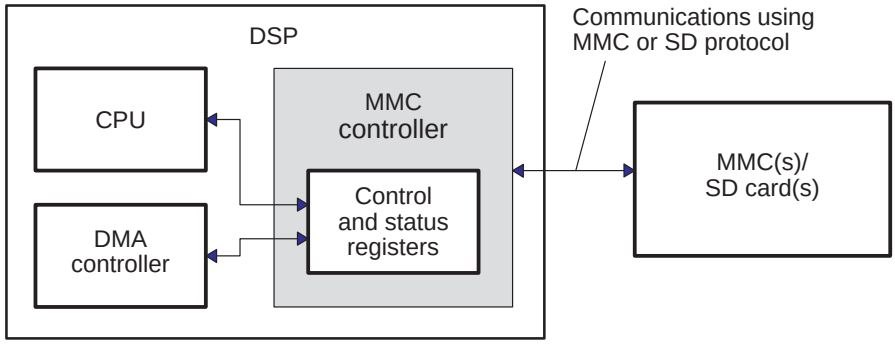
Each of the two MMC controllers shares pins with a multichannel buffered serial port (McBSP) in the DSP. You select the MMC/SD mode or the McBSP mode for the pins by programming serial port mode bits in the external bus selection register (EBSR). For more details, see the device-specific data manual.

1.2 Role of the MMC Controller

As shown in Figure 1−1, the MMC controller passes data between the CPU or the DMA controller on one side and one or more a memory cards on the other side. The CPU or the DMA controller can read from or write to the control and status registers in the MMC controller. As necessary, the CPU and/or the DMA controller can store or retrieve data in the DSP memory or in the registers of other peripherals. The CPU can monitor data activity by reading the status registers and responding to interrupt requests (see section 1.5 on page 1-9). The DMA controller can be notified of data reception/transmission status with the two DMA events (see section 1.6 on page 1-11).

Data transfers between the MMC controller and a memory card can use one bidirectional data line (for the MMC protocol) or four parallel data lines (for the SD protocol). If multiple cards are connected, the MMC controller uses commands of the MMC/SD protocol to select and communicate with one card at a time.

Figure 1−1. Role of the MMC Controller



1.3 MMC Controller Interface

Table 1−1 describes the seven pins of the MMC controller and indicates which pins are used for the MMC protocol and the SD protocol. For a visual comparison of the MMC and SD configurations, see Figure 1−2.

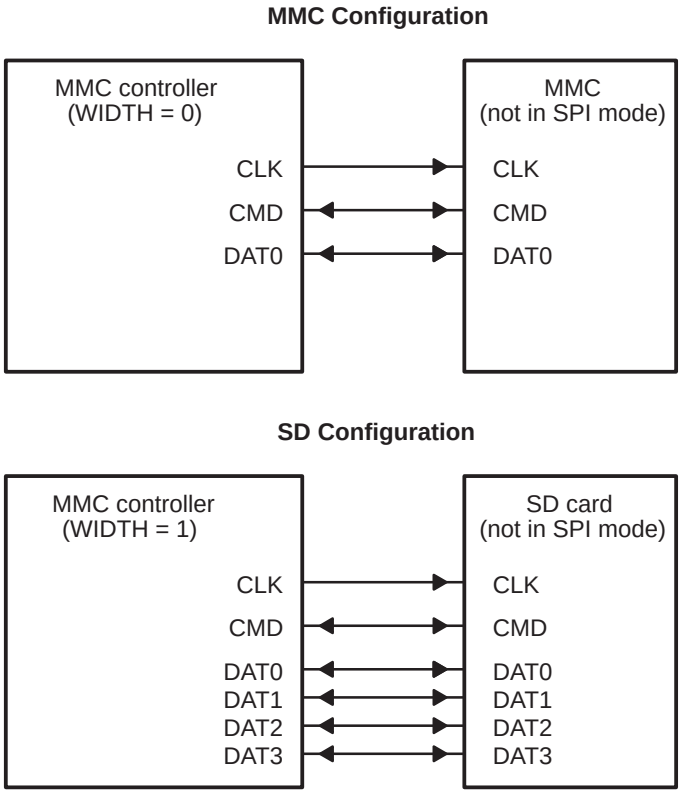
Because the command and data lines are separate, sequential and multiple-block read/write operations are possible. The next command to the card may be sent at the same time as data associated with the previous command.

Table 1−1. MMC Controller Pins

		Protocol		
Pin	**Type**†	**MMC**	**SD**	**Description**
CLK	O	Clock line	Clock line	CLK provides a clock signal to time the transfers on the other pins.
CMD	I/O/Z	Command line	Command line	CMD is used for two-way control communication with the memory card or cards connected to the interface. On CMD, the MMC controller drives commands followed by arguments, and the memory card drives responses to the commands.
DAT0	I/O/Z	Data line	Data line 0	One data line (DAT0) is used for an MMC. All four data lines are needed for an SD card. You configure the number of DAT pins (the data bus width) when you initialize the WIDTH bit of MMCCTL.
DAT1	I/O/Z	(Not used)	Data line 1	
DAT2	I/O/Z	(Not used)	Data line 2	
DAT3	I/O/Z	(Not used)	Data line 3	

† I = Input to the MMC controller; O = Output from the MMC controller; Z = High-impedance

Figure 1−2. MMC Configuration Versus SD Configuration



1.3.1 Write Operation

Figure 1−3 and Table 1−2 describe the signal activity when the MMC controller is writing data to a memory card. The same block length must be defined in the MMC controller and in the card. In a successful write sequence:

1) The controller sends a write command to the card.

2) The card sends a response to acknowledge the command.

3) The controller sends a block of data to the card.

4) The card sends the CRC status to the controller.

5) The card sends low BUSY bits until all the data have been programmed into the flash memory inside the card.

Figure 1−3. Write Operation

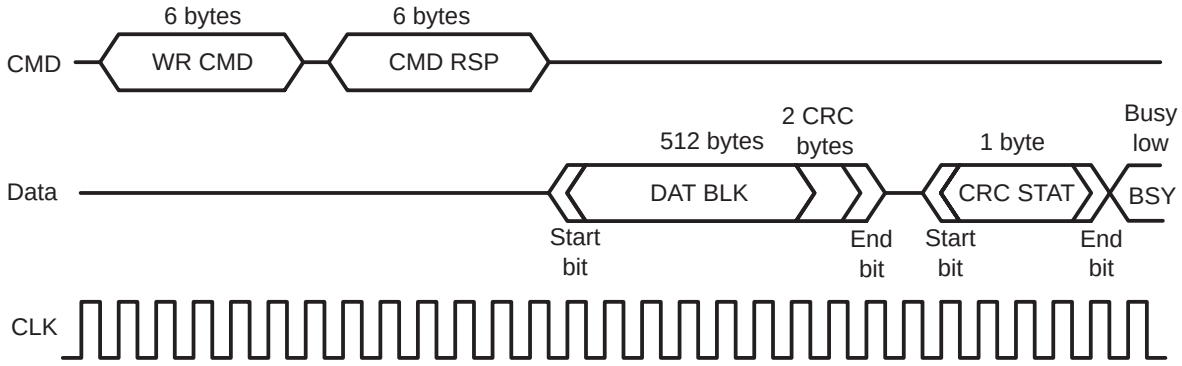


Table 1−2. Write Operation Description

Portion of the Sequence	Description
WR CMD	Write command. A 6-byte WRITE_BLOCK command token is sent from the DSP to the card.
CMD RSP	Command response. The card sends a 6-byte response of type R1 to the DSP, to acknowledge the WRITE_BLOCK command.
DAT BLK	Data block. The DSP 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 one byte of CRC status information to the DSP. This byte indicates whether the data has been accepted by the card or rejected due to a CRC error. The CRC status content is preceded by one start bit and followed by one end bit.
BSY	Busy bits. The CRC status information is followed by a continuous stream of low busy bits until all of the data have been programmed into the flash memory on the card.

1.3.2 Read Operation

Figure 1−4 and Table 1−3 describe the signal activity when the MMC controller is reading data from a memory card. The same block length must be defined in the MMC controller and in the card. In a successful read sequence:

1) The controller sends a read command to the card.

2) The card sends a response to acknowledge the command.

3) The card sends a block of data to the DSP.

Figure 1−4. Read Operation

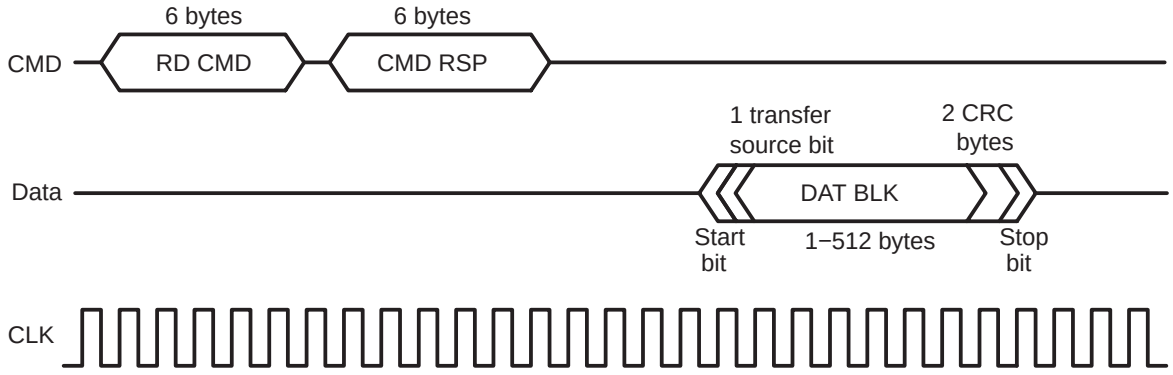


Table 1−3. Read Operation Description

Portion of the Sequence	Description
RD CMD	Read command. A 6-byte READ_SINGLE_BLOCK command token is sent from the DSP to the card.
CMD RSP	Command response. The card sends a 6-byte response of type R1 to the DSP, to acknowledge the READ_SINGLE_BLOCK command.
DAT BLK	Data block. The card sends a block of data to the DSP. The data content is preceded by a start bit and then a transfer source bit. The data content is followed by two CRC bytes and then a stop bit.

1.4 Function Clock and Memory Clock

You must set the desired frequencies for the function clock and for the memory clock in the MMC controller.

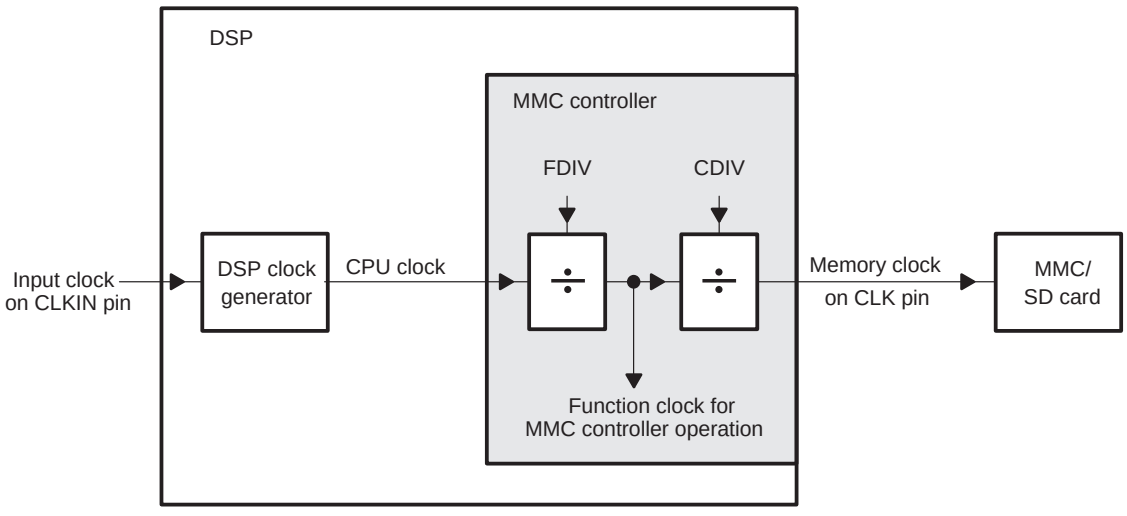
The **function clock** determines the frequency at which the MMC controller operates. Figure 1−5 shows the source of this clock. The DSP clock generator receives a signal from an external clock source and produces a CPU clock with a programmed frequency. A programmable clock divider in the MMC controller divides down the CPU clock to produce the function clock. To specify the divide-down value, initialize the FDIV field of the function clock control register, MMCFCLK. The resulting frequency is:

$$\textit{function clock frequency} = \frac{\textit{CPU clock frequency}}{(\textit{FDIV} + 1)}$$

The **memory clock** appears on the CLK pin of the MMC controller interface. This clock controls the timing of communication between the MMC controller and the attached memory card(s). As shown in Figure 1−5, a second clock divider in the MMC controller divides down the function clock to produce the memory clock. Load the divide-down value into the CDIV field of the clock control register, MMCCLK. The resulting frequency is:

$$\textit{memory clock frequency} = \frac{\textit{function clock frequency}}{2\ (\textit{CDIV} + 1)} = \frac{\textit{CPU clock frequency}}{2\ (\textit{FDIV} + 1)\ (\textit{CDIV} + 1)}$$

Figure 1−5. Clocking Diagram for the MMC Controller



1.5 Interrupt Activity in the MMC Controller

Each MMC controller can generate the interrupt requests described in Table 1−4 and shown in Figure 1−6. When an interrupt event occurs, its flag bit is set in status register 0 (MMCST0). If the corresponding enable bit is set in the interrupt enable register (MMCIE), an interrupt request is generated. All such requests are multiplexed to a single MMC interrupt request for the CPU.

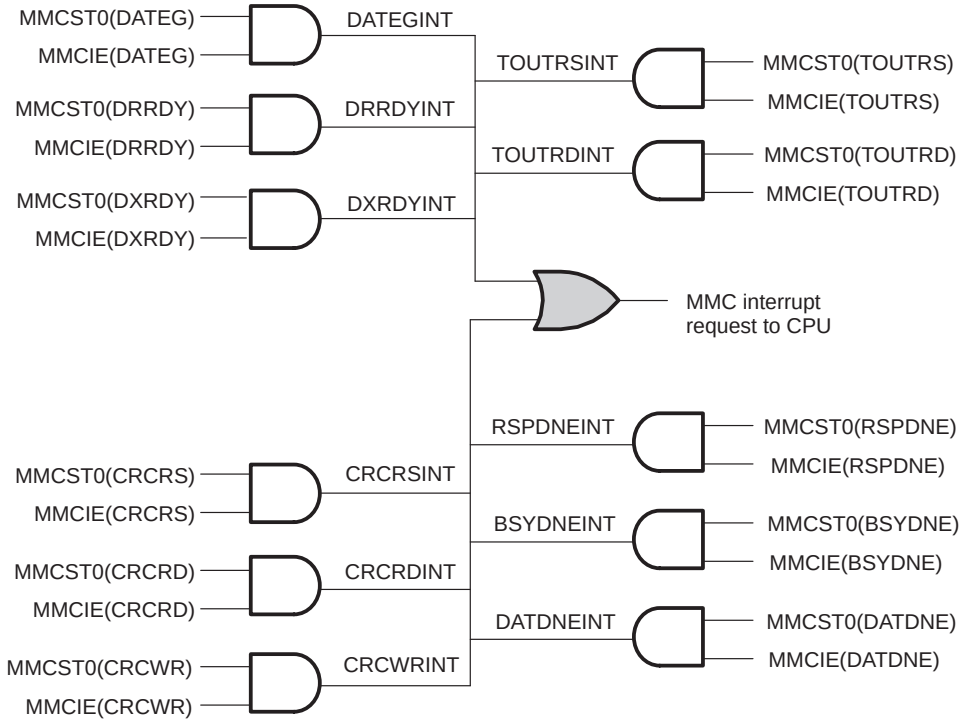
The MMC interrupt is one of the maskable interrupts of the CPU. As with any maskable interrupt request, if it is properly enabled in the CPU, the CPU executes the corresponding interrupt service routine (ISR). The ISR for the MMC interrupt can determine the event that caused the interrupt by checking the bits in MMCST0. When the CPU reads MMCST0, all of the register's bits are automatically cleared except for DRRDY and DXRDY. DRRDY and DXRDY remain set until your code explicitly clears them.

Table 1−4. Descriptions of the MMC Interrupt Requests

Interrupt Request	Interrupt Event
DATEGINT	An edge was detected on the DAT3 pin.
DRRDYINT	MMCDRR is ready to be read (data received).
DXRDYINT	MMCDXR is ready for new data (data transmitted).
CRCRSINT	A CRC error was detected in a response from the memory card.
CRCRDINT	A CRC error was detected while data was being read from the memory card.
CRCWRINT	A CRC error was detected while data was being written to the memory card.
TOUTRSINT	A time-out condition occurred while the MMC controller was waiting for a response to a command.
TOUTRDINT	A time-out condition occurred while the MMC controller was waiting for 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.

Interrupt Request	Interrupt Event
BSYDNEINT	The memory card is no longer sending 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.

Figure 1−6. Enable Paths of the MMC Interrupt Requests



1.6 DMA Events Generated by the MMC Controller

If the DMA event enable bit is set (DMAEN = 1 in MMCCTL), the MMC controller can generate the two DMA events described in Table 1−5. These events are sent to the DMA controller in the DSP. Activity in each DMA channel can be synchronized to respond to one of the two DMA events from the MMC controller.

Table 1−5. DMA Events Generated by the MMC Controller

DMA Event	Description
MMC receive event	New data is available to be read from the data receive register (MMCDRR).
MMC transmit event	The data transmit register (MMCDXR) is ready to accept new data for transmission.

1.7 Data Flow in the Data Registers (MMCDRR and MMCDXR)

The DSP (via the CPU or the DMA controller) reads 16 bits at a time from the data receive register (MMCDRR) and writes 16 bits at a time to the data transmit register (MMCDXR). However, the memory cards are 8-bit devices; they receive or transmit one byte at a time. Figure 1−7 and Figure 1−8 show how this difference in data size is handled via the data registers.

In most cases, once MMCDRR is filled with two bytes, the MMC controller generates a data receive ready (DRRDY) event. If an odd number of bytes is received, the last byte is loaded into the low half of MMCDRR and a DRRDY event is generated.

During transmission, the DSP typically loads two bytes to MMCDXR. When the second byte leaves MMCDXR, a data transmit ready (DXRDY) event is generated. If an odd number of bytes is transmitted, the DSP writes the last byte right aligned (see Figure 1−8). The transmission of the last byte causes a DXRDY event.

Figure 1−7. Data Flow in the Data Receive Register (MMCDRR)

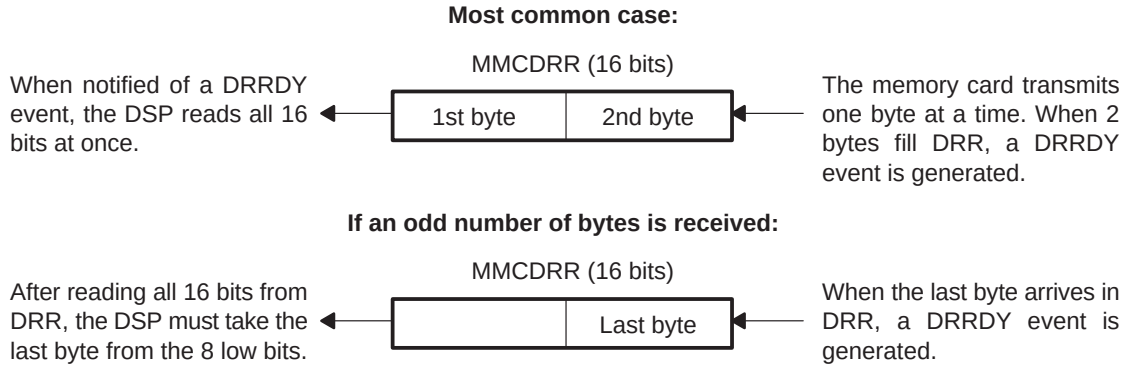
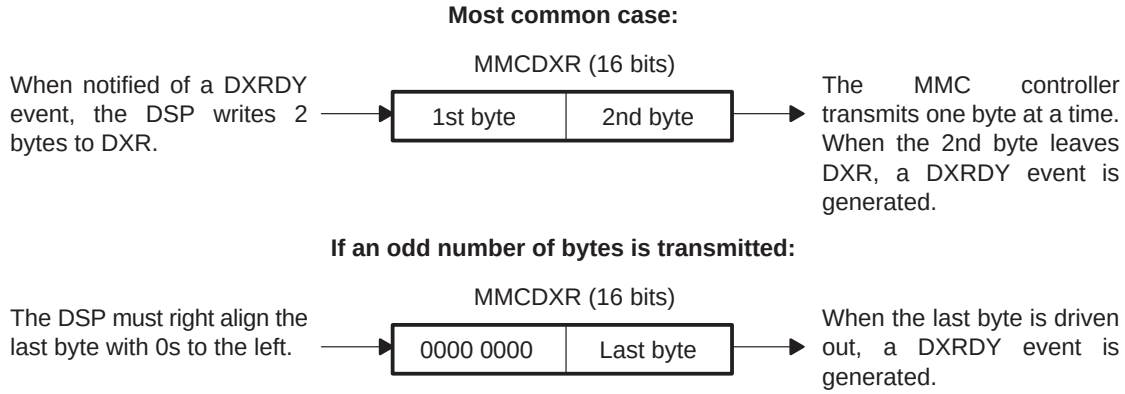


Figure 1−8. Data Flow in the Data Transmit Register (MMCDXR)



1.8 Power, Emulation, and Reset Considerations

1.8.1 Conserving Power

The DSP is divided into a number of idle domains. To minimize power consumption, you can choose which domains are active and which domains are idle at any given time. The *TMS320C55x DSP Peripherals Overview Reference Guide* (SPRU317) points to the power management documentation that describes how to control the idle domains.

If the peripherals domain is idle, the MMC controller may enter an inactive, low-power mode, depending on whether the IDLEEN bit is set in MMCFCLK. If the peripherals domain is idle and IDLEEN = 1, the MMC controller becomes idle. All of its activity stops immediately and only continues when the peripherals domain is reactivated. If IDLEEN = 0, the MMC controller remains active regardless of whether the peripherals domain is idle.

Keep in mind that idle domains other than the peripherals domain can affect the MMC controller. For example, if the clock generator domain is idle, the MMC controller has no clocks for operation.

1.8.2 Effect of Emulation Suspend Condition

An emulation suspend condition causes the MMC controller to halt its activity immediately.

1.8.3 Resetting the MMC Controller

The MMC controller is reset when one of the following occurs:

- The entire TMS320VC5509 DSP is reset with the $\overline{\text{RESET}}$ pin.
- An MMC controller software reset occurs. This occurs when a 1 is written to the CMDRST bit and to the DATRST bit of the control register (MMCCTL).

In either case, the state machines of the MMC controller are reset. One result is that all communications with memory cards stop immediately. In addition, the registers of the MMC controller are forced to the default values shown in the figures in Chapter 5. The transition of DATRST from 0 to 1 resets the two data-ready status bits, DRRDY and DXRDY. These bits are in status register 0 (MMCST0).

While CMDRST = 1 and DATRST = 1, the MMC controller is in its reset state and is disabled. To enable the MMC controller, clear both bits simultaneously.

Chapter 2

Procedures for Common Operations

This chapter describes how to program the MMC controller to send common command sequences to memory cards.

Note:

1) The procedures in this chapter are written for the MMC protocol. If you plan to use the SD protocol, check the appropriate SD card specifications to determine which of these procedures are supported and what modifications must be made to the supported procedures.

2) To determine which of these procedures is supported for a given MMC, check the MMC manufacturer's documentation.

Topic **Page**

2.1 Card Identification Operation

Before the MMC controller can start data transfers to or from memory cards, it must first identify and configure all cards that are connected to it. This section describes the card identification operation, assuming the MMC protocol is used.

The MMC controller must first reset all the cards with a GO_IDLE_STATE command (CMD0). Second, the controller must issue a SEND_OP_COND broadcast command (CMD1) with the desired voltage range as the argument. Incompatible cards enter the inactive state, and compatible cards respond simultaneously, providing a wired-AND result that informs the controller of all supported voltage ranges.

Next, the controller must read the 128-bit card identification (CID) number of each compatible card and assign the card a 16-bit relative card address (RCA). This address is used by the controller to identify the card in all future commands that involve the card. The MMC controller issues an ALL_SEND_CID broadcast command (CMD2), and all cards compete to respond. When the winning card responds, the MMC controller assigns the card an RCA by issuing the SET_RELATIVE_ADDR command (CMD3). After a card has been assigned an RCA, it does not respond to future ALL_SEND_CID commands. The MMC controller repeats the process until no card responds.

The procedure for programming the MMC controller for the card identification operation follows. Figure 2−1 illustrates the procedure and includes recommended values to write to MMCCMD for each command.

1) Use MMCCMD to send a GO_IDLE_STATE command (CMD0). This puts all cards in the idle state.

2) Load the desired voltage range to MMCARGH:MMCARGL, and use MMCCMD to send a SEND_OP_COND broadcast command (CMD1). The result of all responses can be read from MMCCIDX and MMCRSP7−MMCRSP5.

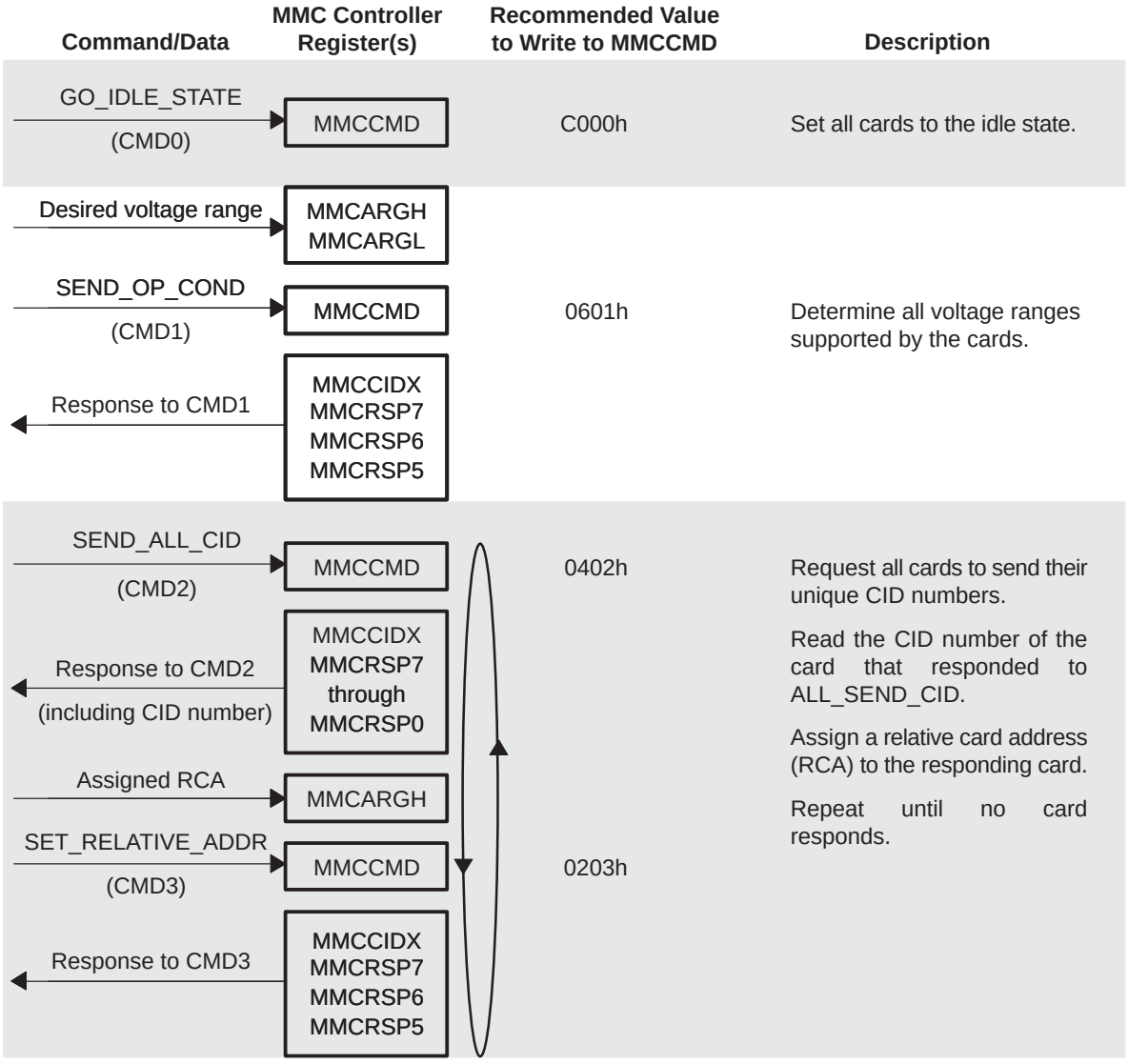
3) Use MMCCMD to send a SEND_ALL_CID command (CMD2).

4) Wait for a card to respond. If a card responds, go to step 5. Otherwise, stop.

5) Read the response from MMCCIDX and MMCRSP7−MMCRSP0. If the CID number has been successfully received, continue.

6) Write a 16-bit RCA to MMCARGH (the bits in MMCARGL are don't cares). Then use MMCCMD to send a SET_RELATIVE_ADDR command (CMD3). The response can be read from MMCCIDX and MMCRSP7−MMCRSP5. The response should indicate that the card is in its stand-by (stby) state. Go to step 4.

Figure 2−1. Card Identification Operation (MMC Protocol)



2.2 Single-Block Read Operation

To read a single block of data from a memory card, use the following procedure, which is also illustrated in Figure 2−2. This procedure assumes the MMC protocol is used. It also assumes the MMC controller has completed the card identification operation and the card you want to access is in its stand-by (stby) state. The same block length must be defined in the MMC controller and in the card.

1) Write the RCA of the card to argument register MMCARGH (the bits in MMCARGL are don't cares). Then use MMCCMD to send a SELECT/DESELECT_CARD command (CMD7) to select the addressed card and deselect the others.

2) Check the BSYDNE bit of MMCST0 or the BUSY bit of MMCST1 to determine whether the card is busy. If the card is busy, wait. Otherwise, read the response from MMCCIDX and MMCRSP7−MMCRSP5. The response should indicate that the card is in its transfer (tran) state.

3) If the block length is different from the length used in the previous operation, set the block length in the MMC controller and in the card. For the MMC controller, load the block length into MMCBLEN. For the card, load the block length to MMCARGH:MMCARGL, and use MMCCMD to send a SET_BLOCKLEN command (CMD16). The response can be read from MMCCIDX and MMCRSP7−MMCRSP5.

4) Load MMCARGH and MMCARGL with the memory start address. Write the upper 16 bits to MMCARGH and the lower 16 bits to MMCARGL. Then use MMCCMD to send a READ_SINGLE_BLOCK command (CMD17). The response can be read from MMCCIDX and MMCRSP7−MMCRSP5.

5) Monitor MMCST0 to determine when a new byte has been successfully received in MMCDRR.

6) Read the new byte of data from MMCDRR.

7) If more bytes are to be read, go to step 5. Otherwise, stop.

Figure 2−2. Single-Block Read Operation (MMC Protocol)

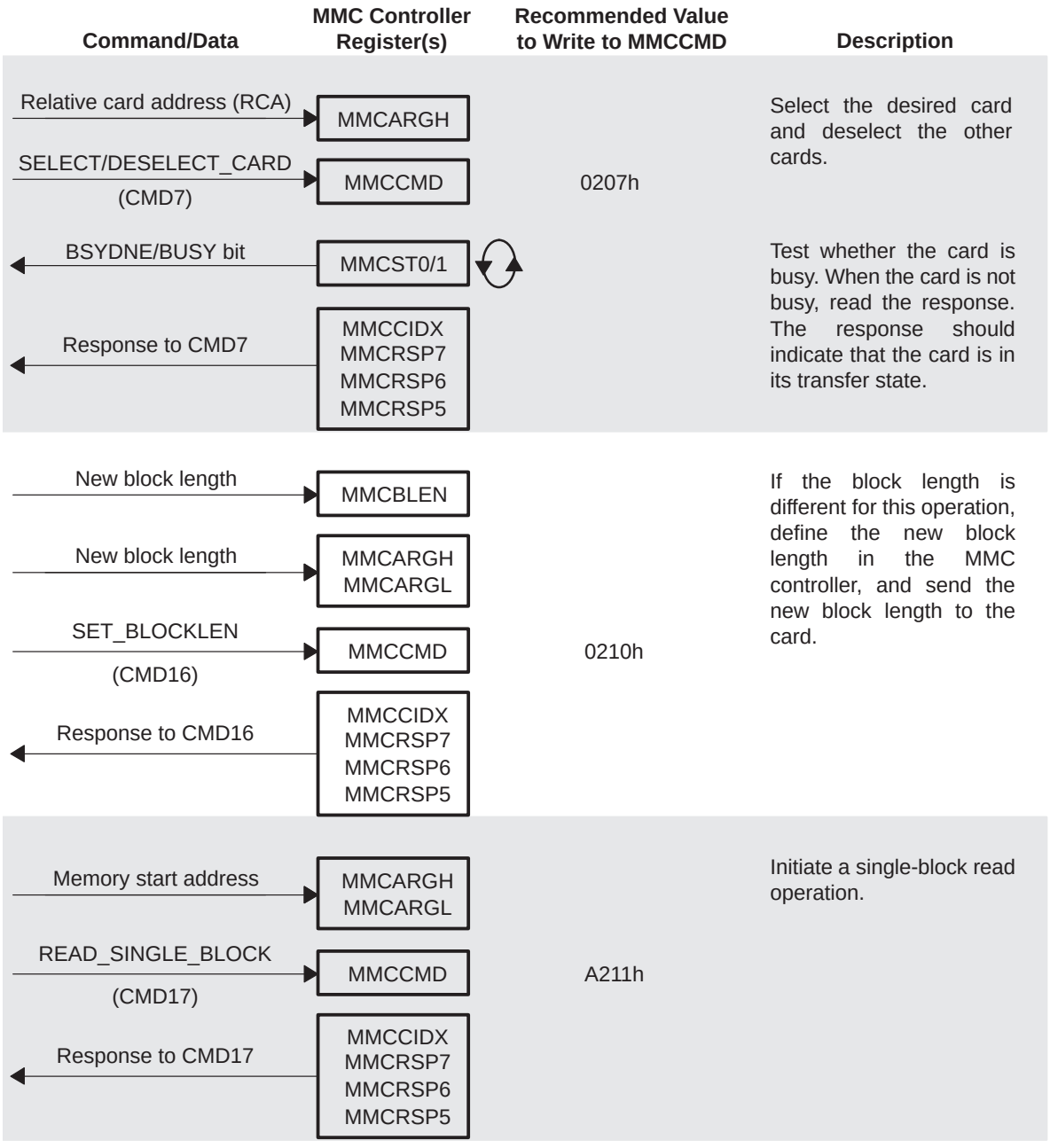
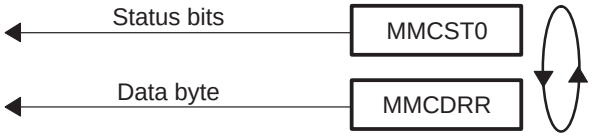


Figure continued on next page

Figure 2−2. Single-Block Read Operation (MMC Protocol) (Continued)



2.3 Multiple-Block Read Operation

To read multiple blocks of data from a memory card, use the following procedure (see also Figure 2−3). This procedure assumes the MMC protocol is used. It also assumes the MMC controller has completed the card identification operation and the card you want to access is in its stand-by (stby) state. The same block length must be defined in the MMC controller and in the card.

1) Write the RCA of the card to MMCARGH (the bits in MMCARGL are don't cares). Then use MMCCMD to send a SELECT/DESELECT_CARD command (CMD7) to select the addressed card and deselect the others.

2) Check the BSYDNE bit of MMCST0 or the BUSY bit of MMCST1 to determine whether the card is busy. If the card is busy, wait. Otherwise, read the response from MMCCIDX and MMCRSP7−MMCRSP5. The response should indicate that the card is in its transfer (tran) state.

3) If the block length is different from the length used in the previous operation, set the block length in the MMC controller and in the card. For the MMC controller, load the block length to MMCBLEN. For the card, load the block length to MMCARGH:MMCARGL, and use MMCCMD to send a SET_BLOCKLEN command (CMD16). The response can be read from MMCCIDX and MMCRSP7−MMCRSP5.

4) If the number of blocks is different from the number used in the previous operation, write the new number to MMCNBLK. The number of blocks can be in the range 1−65535 or can be defined as "infinite" (MMCNBLK = 0). The content of MMCNBLK is copied to MMCNBLC, which is decremented after each block transfer.

5) Load MMCARGH and MMCARGL with the memory start address. Write the upper 16 bits to MMCARGH and the lower 16 bits to MMCARGL. Then use MMCCMD to send a READ_MULTIPLE_BLOCK command (CMD18). The response can be read from MMCCIDX and MMCRSP7−MMCRSP5.

6) Monitor MMCST0 to determine when a new byte has been successfully received in MMCDRR.

7) Read the new byte of data from MMCDRR.

8) If more bytes are to be read, go to step 6. Otherwise, use MMCCMD to send a STOP_TRANSMISSION command (CMD12). When the card responds to the command, the response can be read from MMCCIDX and MMCRSP7−MMCRSP5.

Figure 2−3. Multiple-Block Read Operation (MMC Protocol)

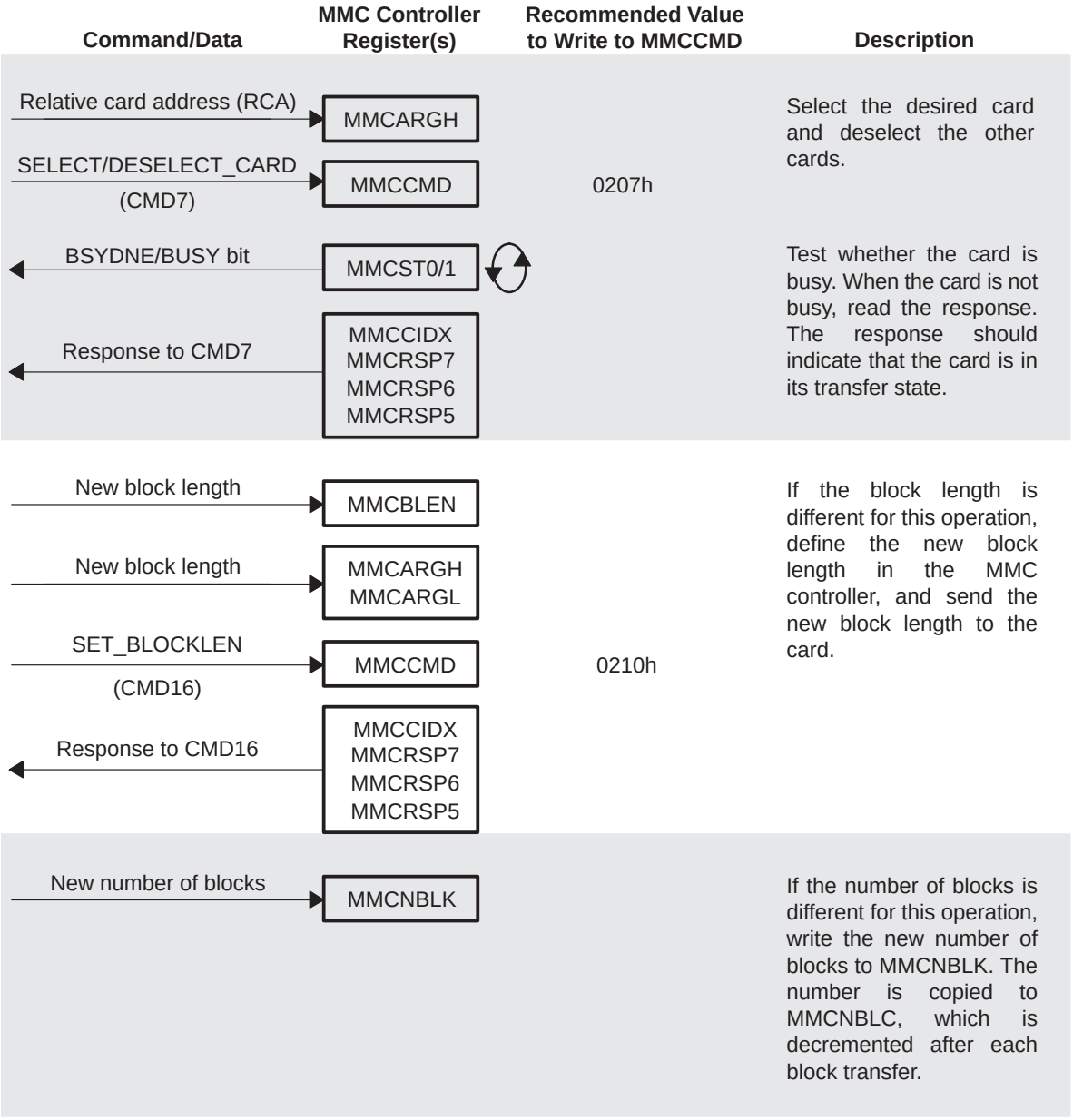
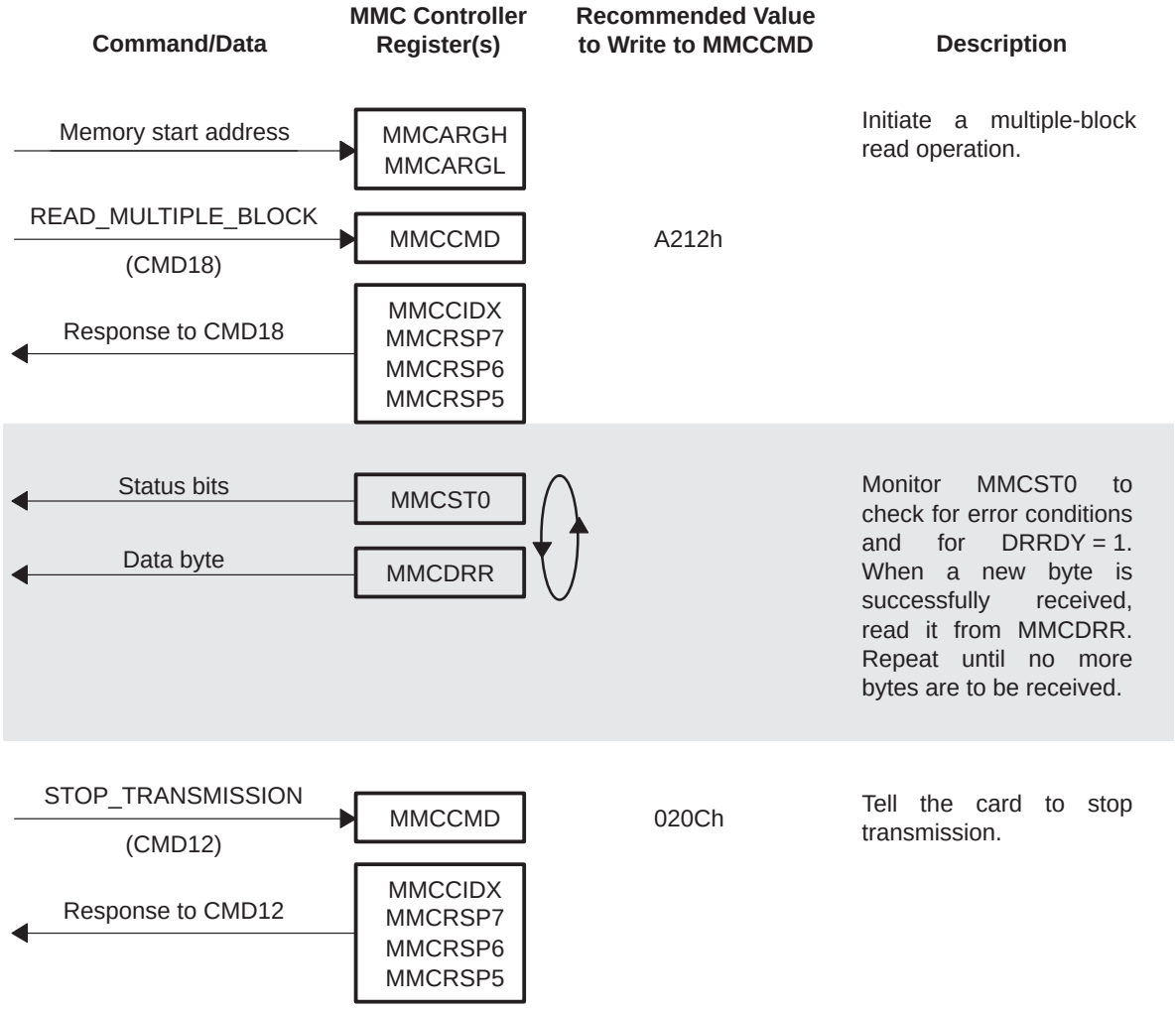


Figure continued on next page

Figure 2−3. Multiple-Block Read Operation (MMC Protocol) (Continued)



2.4 Stream Read Operation

To read a continuous stream of data from a memory card, use the following procedure (see also Figure 2−4). This procedure assumes the MMC protocol is used. It also assumes the MMC controller has completed the card identification operation and the card you want to access is in its stand-by (stby) state.

A stream read operation does not use blocks. Because the operation is not block oriented, no CRC bits are included with the data.

The stream read procedure follows:

1) Write the RCA of the card to MMCARGH (the bits in MMCARGL are don't cares). Then use MMCCMD to send a SELECT/DESELECT_CARD command (CMD7) to select the addressed card and deselect the others.

2) Check the BSYDNE bit of MMCST0 or the BUSY bit of MMCST1 to determine whether the card is busy. If the card is busy, wait. Otherwise, read the response from MMCCIDX and MMCRSP7−MMCRSP5. The response should indicate that the card is in its transfer (tran) state.

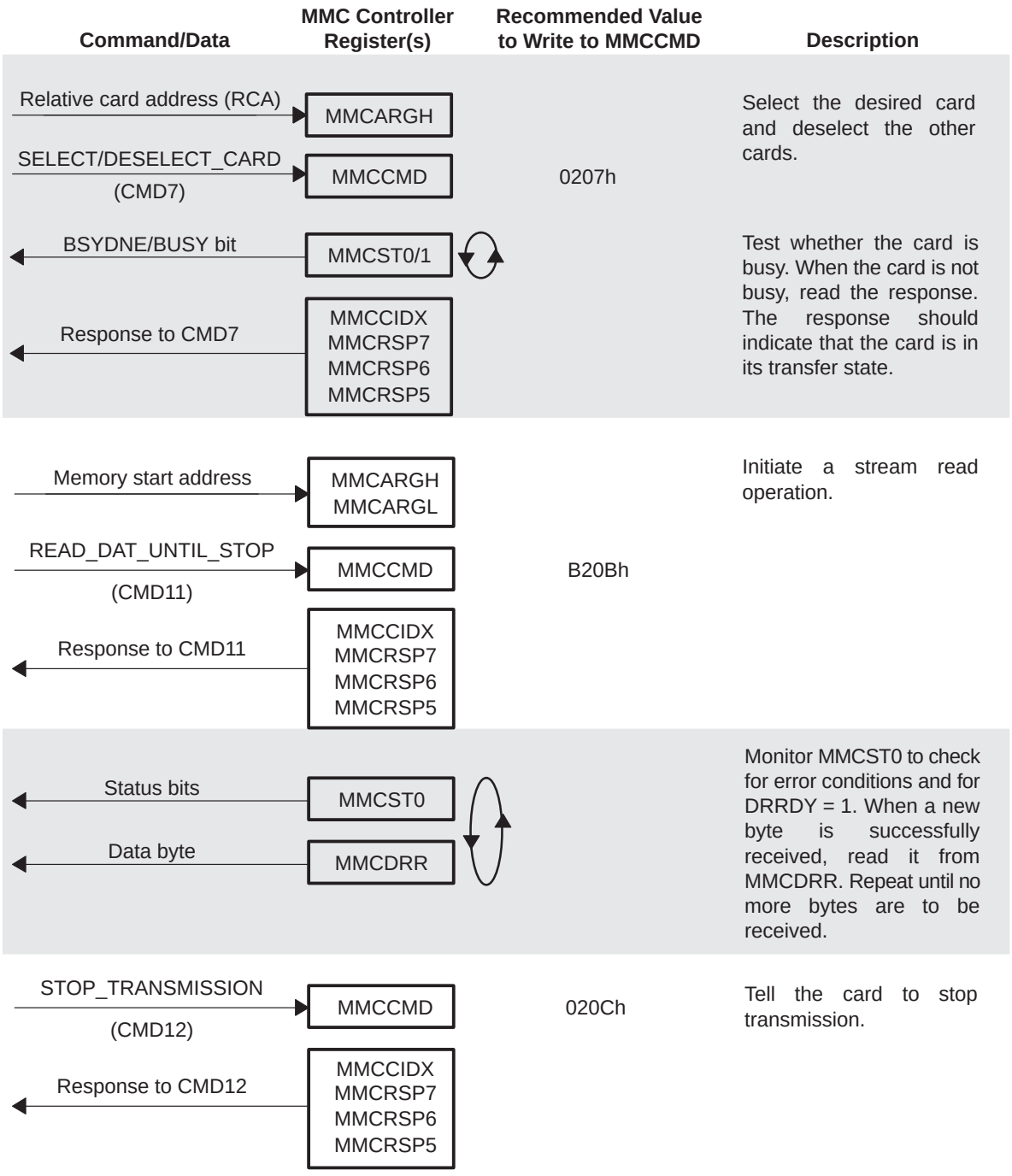
3) Load MMCARGH and MMCARGL with the memory start address. Write the upper 16 bits to MMCARGH and the lower 16 bits to MMCARGL. Then use MMCCMD to send a READ_DAT_UNTIL_STOP command (CMD11). The response can be read from MMCCIDX and MMCRSP7−MMCRSP5.

4) Monitor MMCST0 to determine when a new byte has been successfully received in MMCDRR.

5) Read the new byte of data from MMCDRR.

6) If more bytes are to be read, go to step 4. Otherwise, use MMCCMD to send a STOP_TRANSMISSION command (CMD12). When the card responds to the command, the response can be read from MMCCIDX and MMCRSP7−MMCRSP5.

Figure 2−4. Stream Read Operation (MMC Protocol)



2.5 Single-Block Write Operation

To write a single block of data to a memory card, use the following the procedure, which is also illustrated in Figure 2−5. This procedure assumes the MMC protocol is used. It also assumes the MMC controller has completed the card identification operation and the card you want to access is in its stand-by (stby) state. The same block length must be defined in the MMC controller and in the card.

1) Write the RCA of the card to MMCARGH (the bits in MMCARGL are don't cares). Then use MMCCMD to send a SELECT/DESELECT_CARD command (CMD7) to select the addressed card and deselect the others.

2) Check the BSYDNE bit of MMCST0 or the BUSY bit of MMCST1 to determine whether the card is busy. If the card is busy, wait. Otherwise, read the response from MMCCIDX and MMCRSP7−MMCRSP5. The response should indicate that the card is in its transfer (tran) state.

3) If the block length is different than the length used in the previous operation, set the block length in the MMC controller and in the card. For the MMC controller, load the block length into MMCBLEN. For the card, load the block length to MMCARGH:MMCARGL, and use MMCCMD to send a SET_BLOCKLEN command (CMD16). The response can be read from MMCCIDX and MMCRSP7−MMCRSP5.

4) Write the first byte of the data block to MMCDXR.

5) Load MMCARGH and MMCARGL with the memory start address. Write the upper 16 bits to MMCARGH and the lower 16 bits to MMCARGL. Then use MMCCMD to send a WRITE_BLOCK command (CMD24). The response can be read from MMCCIDX and MMCRSP7−MMCRSP5.

6) Monitor MMCST0 to determine when the current byte has been successfully transferred out of MMCDXR.

7) If more bytes of the block remain to be transmitted, write the next byte of the data block to MMCDXR, and go to step 6. Otherwise, stop. The CRC status token from the memory card can be read from MMCDRSP.

Figure 2−5. Single-Block Write Operation (MMC Protocol)

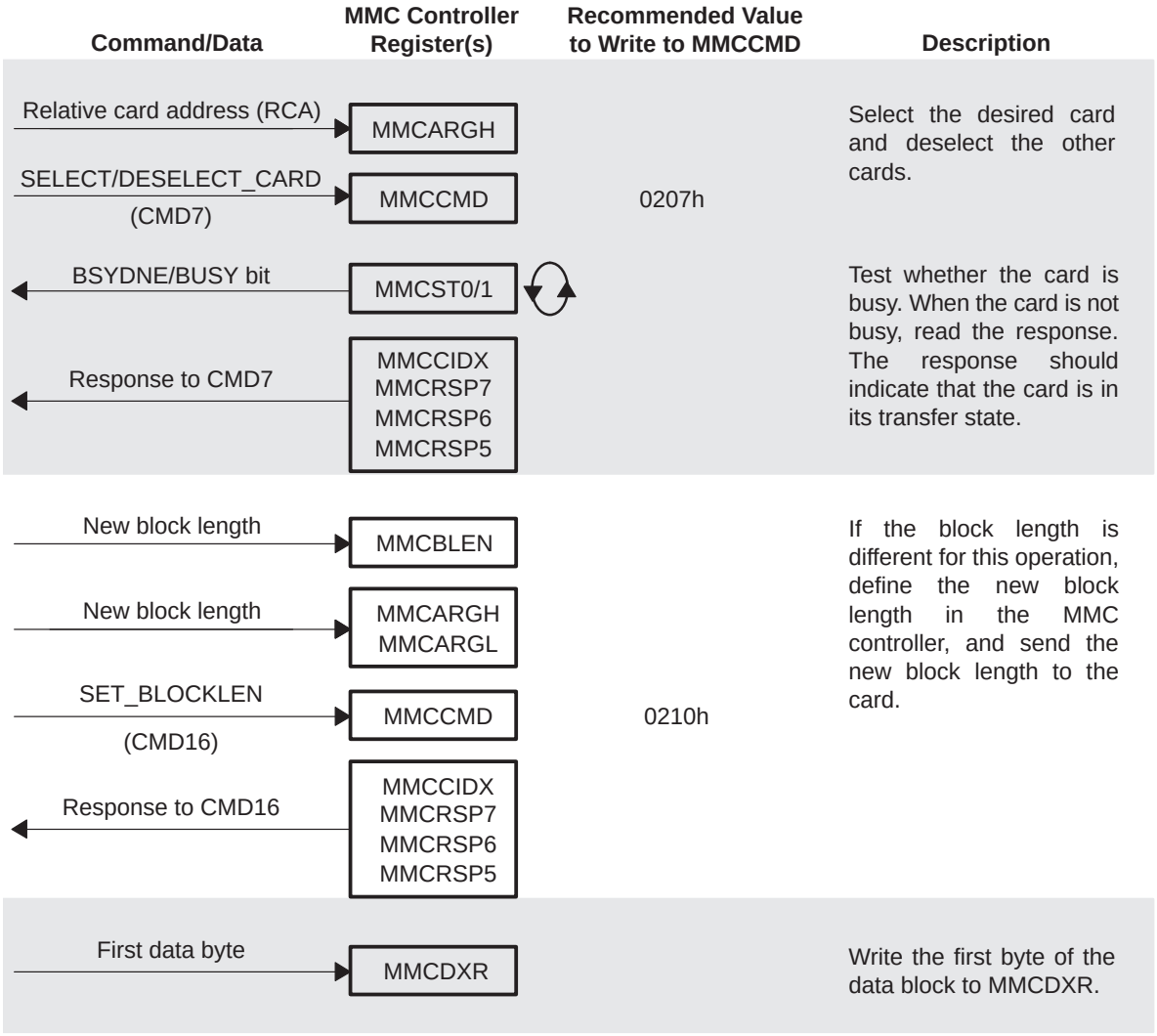
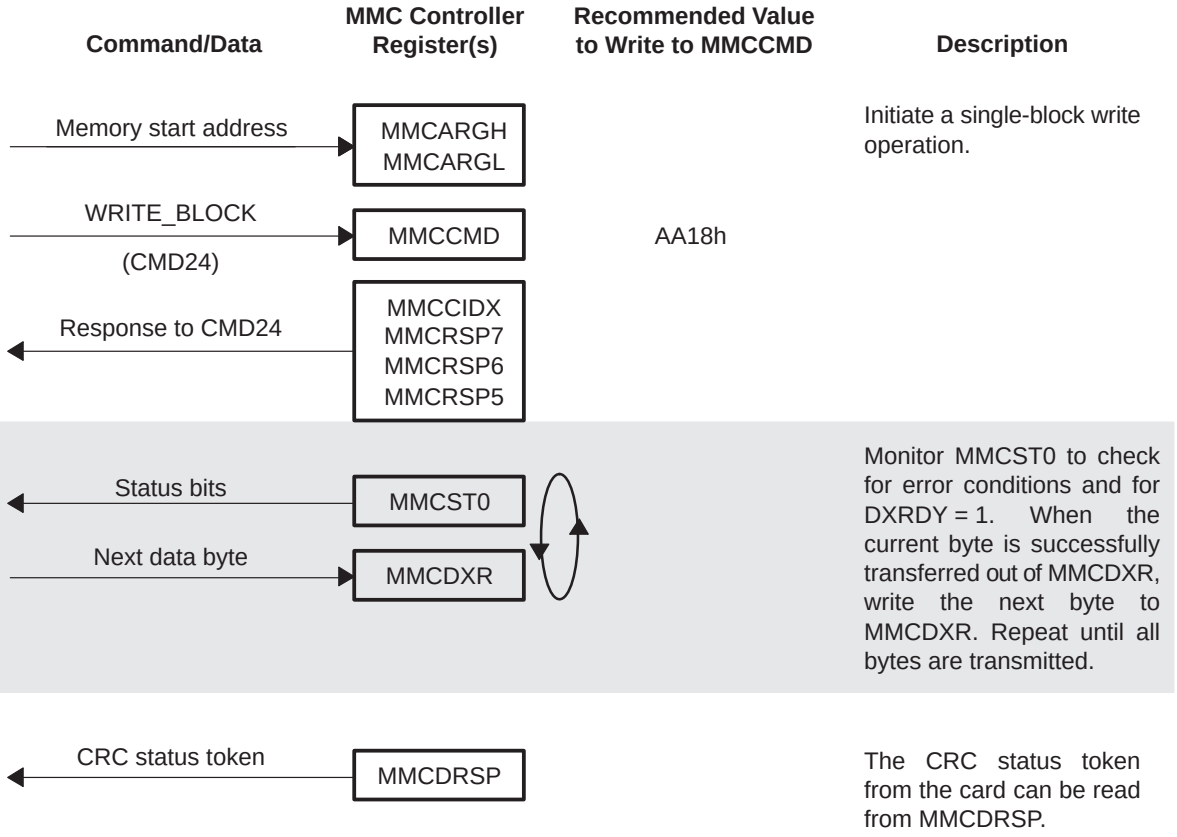


Figure continued on next page

Figure 2−5. Single-Block Write Operation (MMC Protocol) (Continued)



2.6 Multiple-Block Write Operation

To write multiple blocks of data to a memory card, use the following procedure (see also Figure 2−6). This procedure assumes the MMC protocol is used. It also assumes the MMC controller has completed the card identification operation and the card you want to access is in its stand-by (stby) state. The same block length must be defined in the MMC controller and in the card.

1) Write the RCA of the card to MMCARGH (the bits in MMCARGL are don't cares). Then use MMCCMD to send a SELECT/DESELECT_CARD command (CMD7) to select the addressed card and deselect the others.

2) Check the BSYDNE bit of MMCST0 or the BUSY bit of MMCST1 to determine whether the card is busy. If the card is busy, wait. Otherwise, read the response from MMCCIDX and MMCRSP7−MMCRSP5. The response should indicate that the card is in its transfer (tran) state.

3) If the block length is different from the length used in the previous operation, set the block length in the MMC controller and in the card. For the MMC controller, load the block length to MMCBLEN. For the card, load the block length to MMCARGH:MMCARGL, and use MMCCMD to send a SET_BLOCKLEN command (CMD16). The response can be read from MMCCIDX and MMCRSP7−MMCRSP5.

4) If the number of blocks is different from the number used in the previous operation, write the new number to MMCNBLK. The number of blocks can be in the range 1−65535 or can be defined as "infinite" (MMCNBLK = 0). The content of MMCNBLK is copied to MMCNBLC, which is decremented after each block transfer.

5) Write the first byte of the first data block to MMCDXR.

6) Load MMCARGH and MMCARGL with the memory start address. Write the upper 16 bits to MMCARGH and the lower 16 bits to MMCARGL. Then use MMCCMD to send a WRITE_MULTIPLE_BLOCK command (CMD25). The response can be read from MMCCIDX and MMCRSP7−MMCRSP5.

7) Monitor MMCST0 to determine when the current byte has been successfully transferred out of MMCDXR.

8) If more bytes are to be transmitted, write the next byte of data to MMCDXR, and go to step 7. Otherwise, go to step 9. At the end of each block, the CRC status token from the memory card can be read from MMCDRSP.

9) Send a STOP_TRANSMISSION command (CMD12). After sending this command, check the BSYDNE bit of MMCST0 or the BUSY bit of MMCST1 to determine whether the card is busy. If the card is busy, wait. When the card responds to the command, the response can be read from MMCCIDX and MMCRSP7−MMCRSP5.

Figure 2−6. Multiple-Block Write Operation (MMC Protocol)

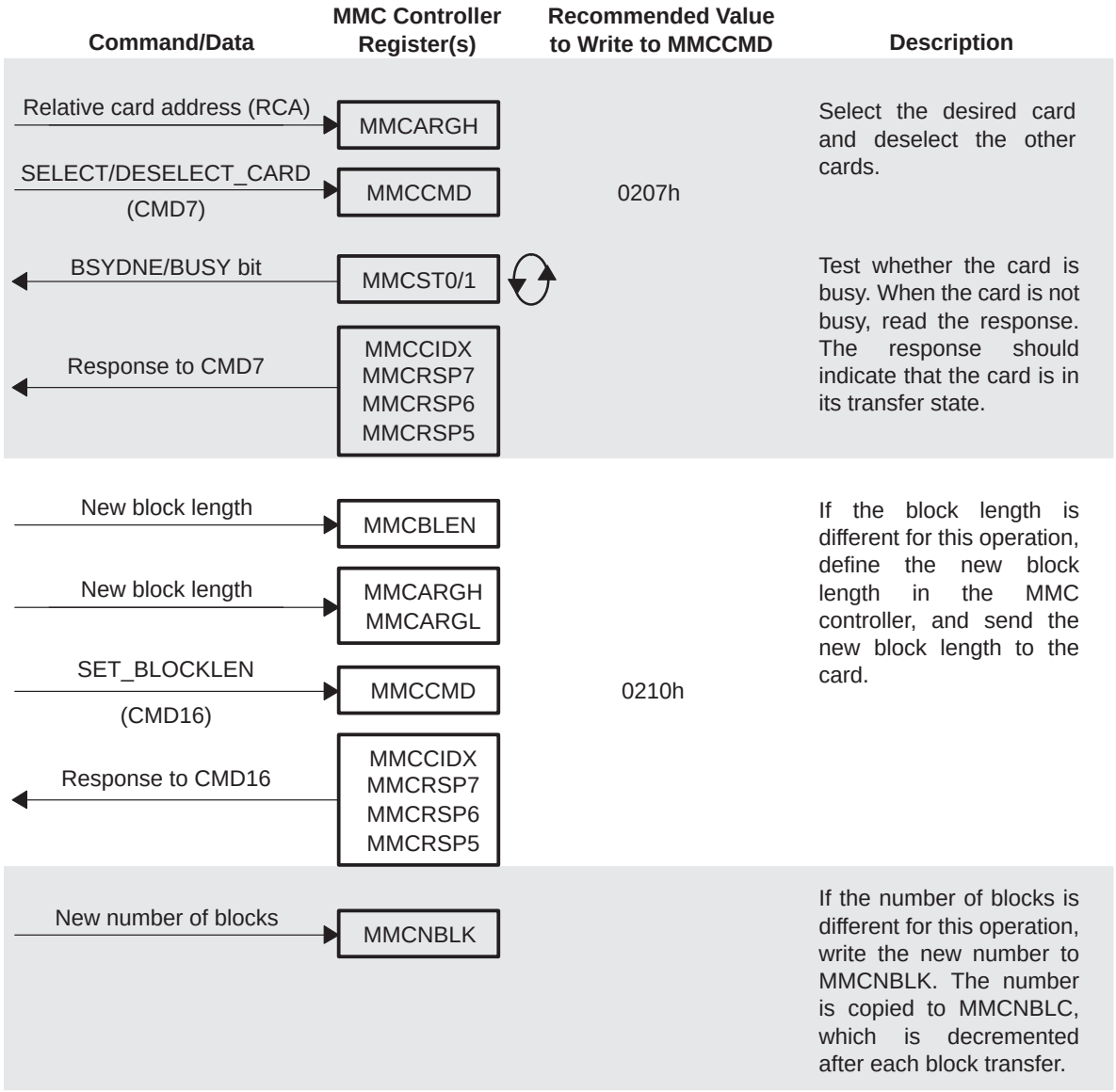
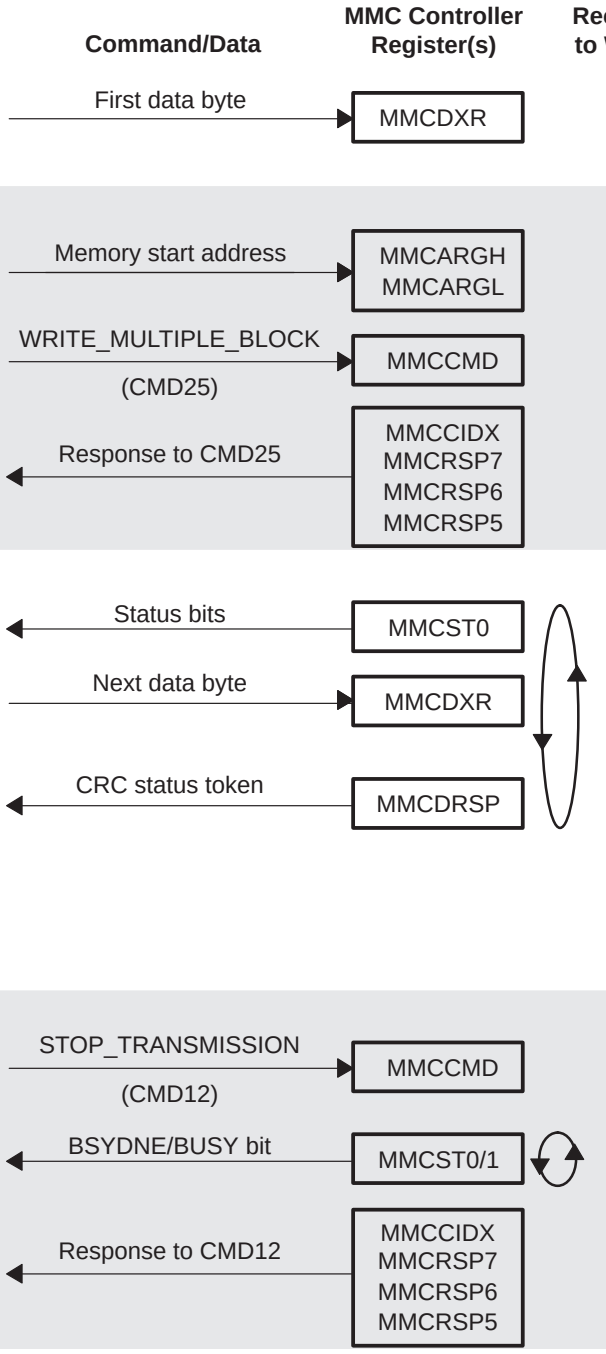


Figure continued on next page

Figure 2−6. Multiple-Block Write Operation (MMC Protocol) (Continued)



2.7 Stream Write Operation

To write a continuous stream of data from a memory card, use the following procedure (see also Figure 2−7). This procedure assumes the MMC protocol is used. It also assumes the MMC controller has completed the card identification operation and the card you want to access is in its stand-by (stby) state.

A stream write operation does not use blocks. Because the operation is not block oriented, no CRC bits are included with the data.

The stream write procedure follows:

1) Write the RCA of the card to MMCARGH (the bits in MMCARGL are don't cares). Then use MMCCMD to send a SELECT/DESELECT_CARD command (CMD7) to select the addressed card and deselect the others.

2) Check the BSYDNE bit of MMCST0 or the BUSY bit of MMCST1 to determine whether the card is busy. If the card is busy, wait. Otherwise, read the response from MMCCIDX and MMCRSP7−MMCRSP5. The response should indicate that the card is in its transfer (tran) state.

3) Write the first byte of data to MMCDXR.

4) Load MMCARGH and MMCARGL with the memory start address. Write the upper 16 bits to MMCARGH and the lower 16 bits to MMCARGL. Then use MMCCMD to send a WRITE_DAT_UNTIL_STOP command (CMD20). The response can be read from MMCCIDX and MMCRSP7−MMCRSP5.

5) Use MMCST0 to determine when the current byte has been successfully transferred out of MMCDXR.

6) If more bytes are to be transmitted, write the next byte of data to MMCDXR, and go to step 5. Otherwise, go to step 7.

7) Send a STOP_TRANSMISSION command (CMD12). After sending this command, check the BSYDNE bit of MMCST0 or the BUSY bit of MMCST1 to determine whether the card is busy. If the card is busy, wait. When the card responds to the command, the response can be read from MMCCIDX and MMCRSP7−MMCRSP5.

Figure 2−7. Stream Write Operation (MMC Protocol)

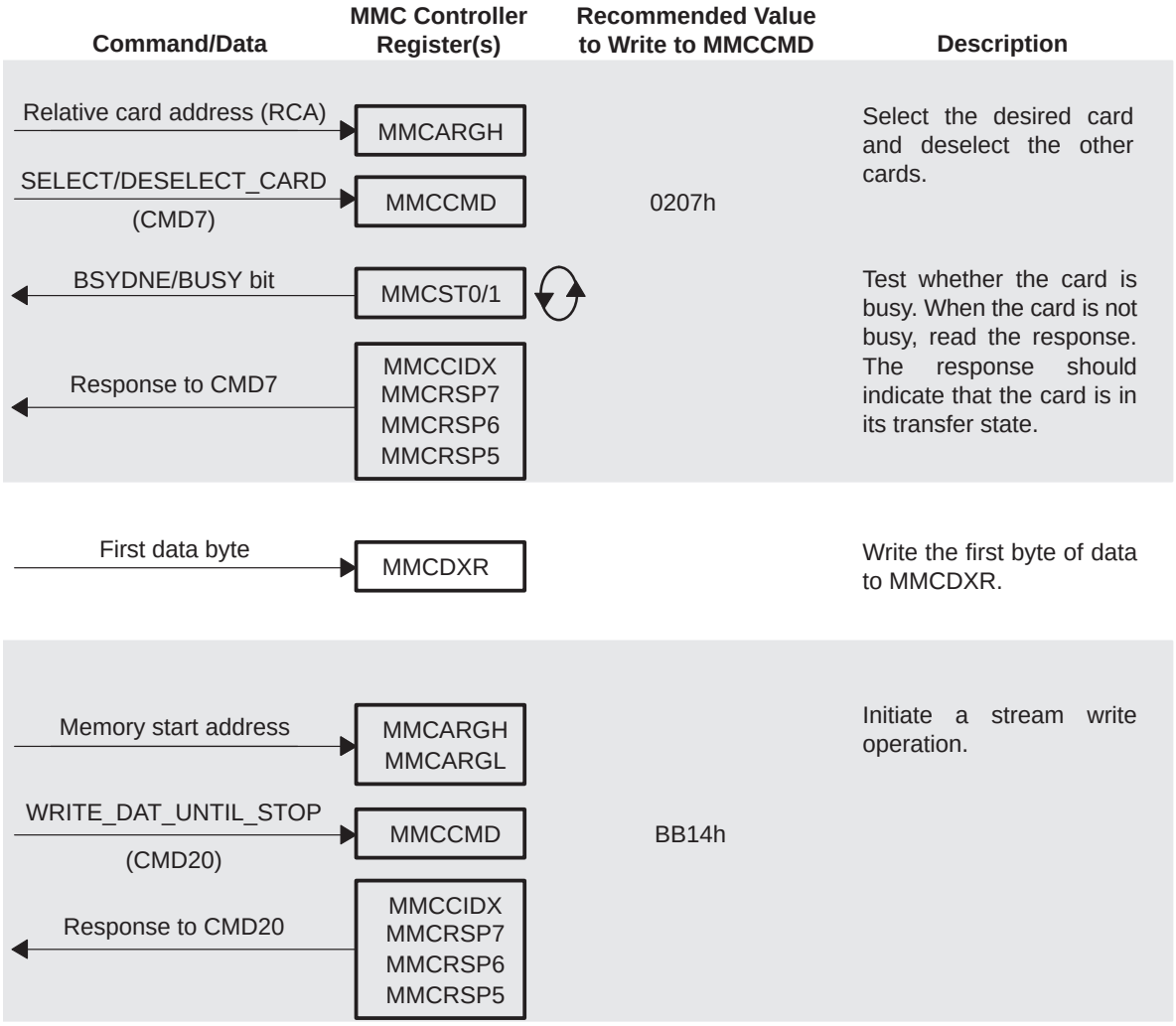
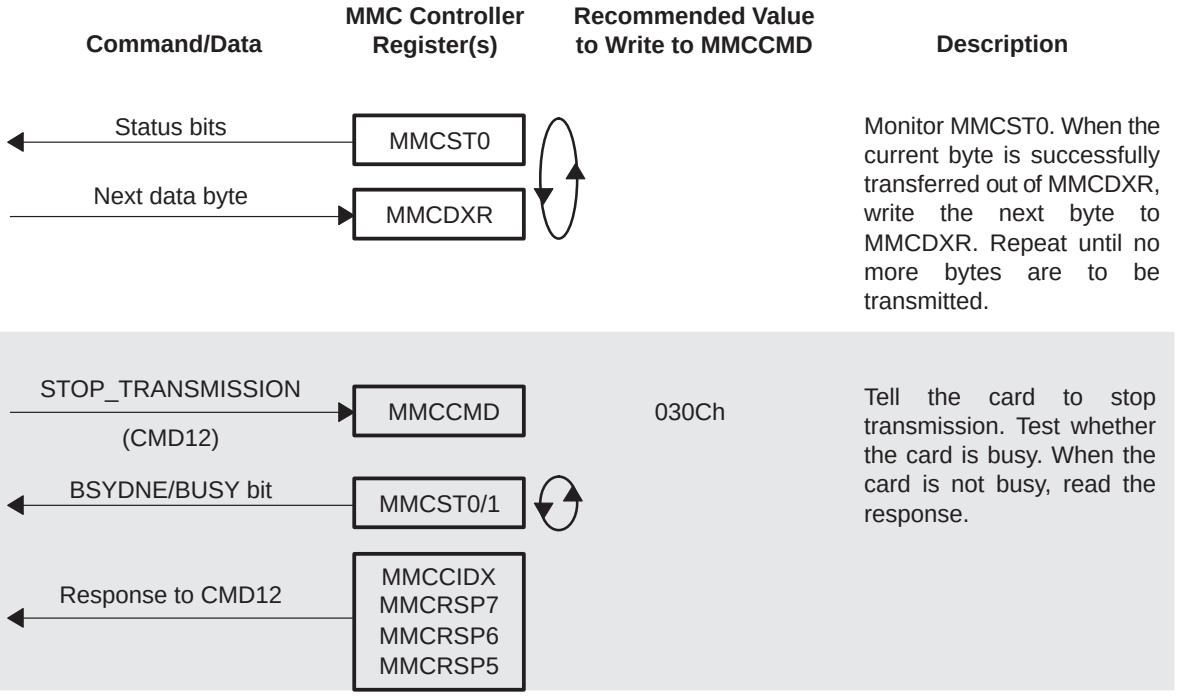


Figure continued on next page

Figure 2−7. Stream Write Operation (MMC Protocol) (Continued)



Chapter 3

Initializing the MMC Controller

This chapter describes how to program the registers that should be initialized before the MMC controller begins communications with attached memory cards.

Topic **Page**

3.1 Initializing the MMC Controller

The general procedure for initializing the MMC controller is given in the following steps.

1) Place the MMC controller in its reset state by setting the CMDRST bit of MMCCTL and the DATRST bit of MMCCTL. With the same register write operation, write the desired values to other bits in MMCCTL.

2) Write to other registers to complete the MMC controller configuration.

3) Clear the CMDRST bit and the DATRST bit to release the MMC controller from its reset state. Make sure you do not change the values you wrote to the other bits of MMCCTL in step 1.

4) Enable the CLK pin so that the memory clock is sent to the memory card.

3.2 Initializing the MMC Control Register (MMCCTL)

Figure 3−1 shows the bit fields in the MMC control register (MMCCTL).

Figure 3−1. MMCCTL

15	12	11	9	8
Reserved		Reserved†		DMAEN
R−0		R/W−0		R/W−0

7	6	5	3	2	1	0
DATEG		Reserved†		WIDTH	CMDRST	DATRST
R/W−00		R/W−0		R/W−0	R/W−0	R/W−0

Legend: R = Read; W = Write; -*n* = Value after hardware reset

† Keep the default value (0) in these reserved bits.

3.2.1 Enable/Disable DMA Events

Register(Field)	Value	Description
MMCCTL(DMAEN)	0	Disable DMA events.
	1	Enable DMA events.

Use DMAEN to disable for enable the MMC controller DMA events, which are described in section 1.6 (page 1-11).

3.2.2 Select a Type of Edge Detection (If Any) for the DAT3 Pin

Register(Field)	Value	Description
MMCCTL(DATEG)	00b	Disable DAT3 edge detection.
	01b	Enable DAT3 rising edge detection.
	10b	Enable DAT3 falling edge detection.
	11b	Enable DAT3 dual edge detection (detect both edges).

The DATEG control bit of MMCCTL enables or disables general-purpose edge detection on the DAT3 pin. If you enable edge detection and an edge is detected, the DATEG flag bit of MMCST0 is set. In addition, if DATEG = 1 in MMCIE, an interrupt request is generated.

3.2.3 Select a Data Bus Width

Register(Field)	Value	Description
MMCCTL(WIDTH)	0	Use a 1-bit data bus (DAT0 pin).
	1	Use a 4-bit data bus (pins DAT0−DAT3).

The MMC controller must know how wide the data bus must be for the memory card that is connected. If an MMC is connected, specify a 1-bit data bus (WIDTH = 0). If an SD card is connected, specify a 4-bit data bus (WIDTH = 1).

3.2.4 Enable/Reset the MMC Controller

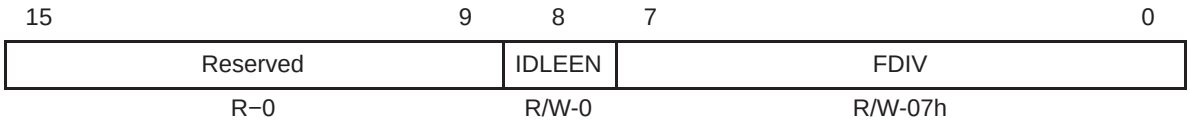
Register(Field)	Value	Description
MMCCTL(CMDRST)	0	Enable the CMD (command) logic of the MMC controller.
	1	Place the CMD logic of the MMC controller in its reset state.
MMCCTL(DATRST)	0	Enable the DAT (data) logic of the MMC controller.
	1	Place the DAT logic of the MMC controller in its reset state.

To place the MMC controller in its reset state and disable it, set the CMDRST and DATRST bits of MMCCTL. The first step of the MMC controller initialization process is to disable both sets of logic. When initialization is complete but before you enable the CLK pin, enable the MMC controller by clearing the CMDRST and DATRST bits.

3.3 Initializing the Clock Control Registers (MMCFCLK and MMCCLK)

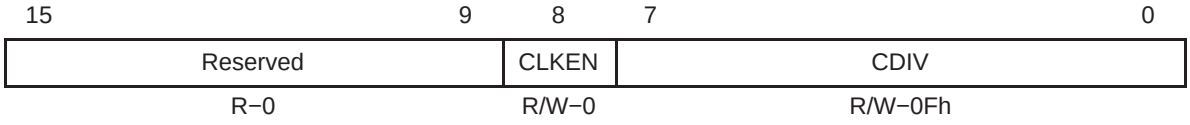
Figure 3−2 and Figure 3−3 show the bit fields in the function clock control register (MMCFCLK) and the clock control register (MMCCLK), respectively.

Figure 3−2. MMCFCLK



Legend: R = Read; W = Write; -*n* = Value after hardware reset

Figure 3−3. MMCCLK



Legend: R = Read; W = Write; -*n* = Value after hardware reset

3.3.1 Set the Function Clock and the Memory Clock

Register(Field)	Value	Description
MMCFCLK(FDIV)	1−255	Use this field to set the divide-down value for the function clock.
MMCCLK(CDIV)	0−255	Use this field to set the divide-down value for the memory clock.

To generate the function clock (the clock for activity inside the MMC controller), the MMC controller divides down the CPU clock as shown in the following equation. When you initialize MMCFCLK, you specify FDIV, a divide-down value in the range 1 through 255.

$$\textit{function clock frequency} = \frac{\textit{CPU clock frequency}}{(\textit{FDIV} + 1)}$$

The memory clock (the clock for the attached memory card) is a divided-down version of the function clock; see the following equation. When you initialize MMCCLK, you specify CDIV, a divide-down value in the range 0 through 255.

$$\textit{memory clock frequency} = \frac{\textit{function clock frequency}}{2\ (\textit{CDIV} + 1)} = \frac{\textit{CPU clock frequency}}{2\ (\textit{FDIV} + 1)\ (\textit{CDIV} + 1)}$$

For more information about the function clock and the memory clock, see section 1.4 on page 1-8.

3.3.2 Enable/Disable the Idle Capability

Register(Field)	Value	Description
MMCFCLK(IDLEEN)	0	The MMC controller cannot be made idle.
	1	If PERI = 1 (see just below), the MMC controller is idle (the function clock is stopped) after the IDLE instruction is executed.
ICR(PERI)	0	Any peripheral in the peripherals idle domain will be active after the IDLE instruction is executed.
	1	Any peripheral in the peripherals idle domain can be idle after the IDLE instruction is executed, depending on the state of that peripheral's idle enable bit.

The DSP is divided into a number of idle domains. The MMC controller is one of the peripherals in the peripherals idle domain. If you want the MMC controller to become idle in response to an IDLE instruction, make the following preparations:

1) Write 1 to the idle enable (IDLEEN) bit in MMCFCLK. This tells the DSP to stop the function clock of the MMC controller when the peripherals domain becomes idle.

2) Write 1 to the PERI bit in the idle control register (ICR) of the DSP. This tells the DSP to make the peripherals domain idle when an IDLE instruction is executed.

The *TMS320C55x DSP Peripherals Overview Reference Guide* (SPRU317) points to the power management documentation that describes how to control the idle domains.

3.3.3 Enable/Disable the CLK Pin

Register(Field)	Value	Description
MMCCLK(CLKEN)	0	Disable the CLK pin; drive a constant, low signal on the pin.
	1	Enable the CLK pin, so that it shows the memory clock signal.

The CLKEN bit determines whether the memory clock appears on the CLK pin.

3.4 Initializing the Interrupt Enable Register (MMCIE)

Register(Field)	Value	Description
MMCIE(11−0)	000h−FFFh	Use this field to select which of the MMC interrupt requests will be forwarded to the CPU.

The bits in MMCIE individually enable or disable the interrupt requests described in section 1.5 (page 1-9). Figure 3−4 shows the bit fields of MMCIE. Set one of these bits to enable the associated interrupt request. Clear one of these bit to disable the associated interrupt request.

Figure 3−4. MMCIE

15			12	11	10	9	8
Reserved				DATEG	DRRDY	DXRDY	Reserved†
R−0				R/W−0	R/W−0	R/W−0	R/W−0
7	6	5	4	3	2	1	0
CRCRS	CRCRD	CRCWR	TOUTRS	TOUTRD	RSPDNE	BSYDNE	DATDNE
R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

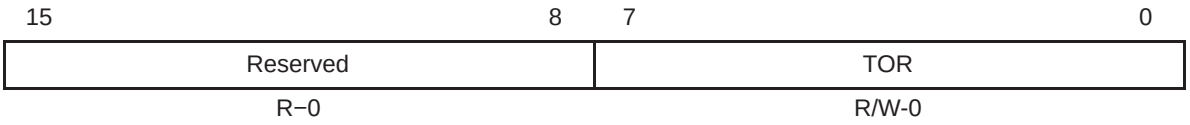
Legend: R = Read; W = Write; -*n* = Value after hardware reset

† Keep the default value (0) in this reserved bit.

3.5 Initializing the Time-Out Registers (MMCTOR and MMCTOD)

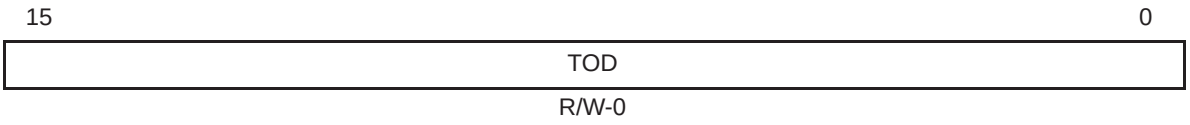
Specify the time-out period for responses (TOR, see Figure 3−5) and the time-out period for read data (TOD, see Figure 3−6) as described in the following subsections. If a memory card should require longer time-out periods than MMCTOR and MMCTOD can provide, software time-out mechanisms can be implemented.

Figure 3−5. MMCTOR



Legend: R = Read; W = Write; -*n* = Value after hardware reset

Figure 3−6. MMCTOD



Legend: R = Read; W = Write; -*n* = Value after hardware reset

3.5.1 Set the Time-Out Period for a Response

Register(Field)	Value	Description
MMCTOR(7−0)	0	Do not check for a response time-out condition.
	n = 1−255	If there is no response from the memory card in n CLK cycles, record a time-out condition.

When the MMC controller sends a command a memory card, it often must wait for a response. The controller can wait indefinitely or for up to 255 memory clock cycles. If you load 0 into MMCTOR during initialization, the controller waits for a response indefinitely. If you load a nonzero value into MMCTOR, the controller automatically stops waiting after the specified number of cycles and then records a response time-out condition. If the associated interrupt request is enabled, the controller also sends an interrupt request to the CPU.

3.5.2 Set the Time-Out Period for a Data Read Operation

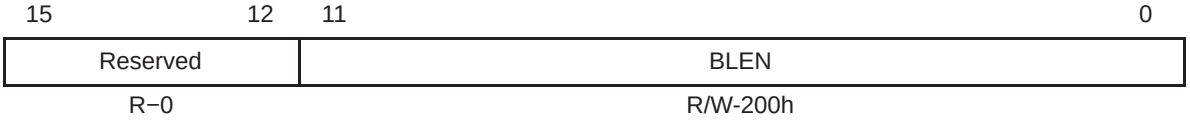
Register(Field)	Value	Description
MMCTOD(15−0)	0	Do not check for a data-read time-out condition.
	n = 1−65535	If no data is received from the memory card in n CLK cycles, record a time-out condition.

When the MMC 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 during initialization, the controller waits indefinitely. If you load a nonzero value *n* into MMCTOD, the controller waits *n* memory clock cycles and then records a data-read time-out condition in MMCST0. If the associated interrupt request is enabled, the controller also sends an interrupt request to the CPU.

3.6 Initializing the Data Block Registers (MMCBLEN and MMCNBLK)

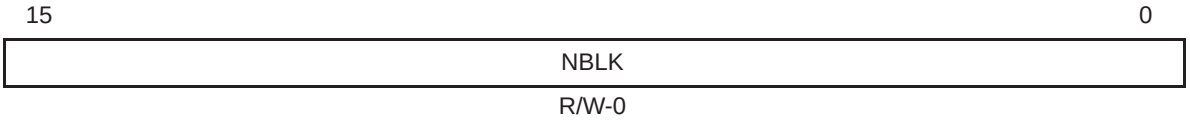
Specify the number of bytes in a data block in MMCBLEN (see Figure 3−7) and the number of blocks in a multiple-block transfer in MMCNBLK (see Figure 3−8). Details about these values are in the following subsections.

Figure 3−7. MMCBLEN



Legend: R = Read; W = Write; -*n* = Value after hardware reset

Figure 3−8. MMCNBLK



Legend: R = Read; W = Write; -*n* = Value after hardware reset

3.6.1 Set the Data Block Length

Register(Field)	Value	Description
MMCBLEN(11−0)	1−512	Use this field to set the number of bytes in a data block.

In MMCBLEN, you must define the size for each block of data transferred between the MMC controller and a memory card. The valid size depends on the type of read/write operation. A length of 0 bytes is prohibited.

3.6.2 Specify the Number of Blocks in a Multiple-Block Transfer

Register(Field)	Value	Description
MMCNBLK(15−0)	0	Transfer an infinite number of blocks.
	n = 1−65535	Transfer n blocks.

For multiple-block transfers, you must specify how many blocks of data are to be transferred between the MMC controller and a memory card. You can specify an infinite number of blocks by loading 0 into MMCNBLK. When MMCNBLK = 0, the MMC controller transfers blocks until you end the transferring with a STOP_TRANSMISSION command. If you need a specific number of blocks transferred, load MMCNBLK with a value from 1 through 65535.

Chapter 4

Monitoring the MMC Controller

This chapter describes registers and specific register bits that you can use to obtain the status of the MMC controller and its communications with memory cards.

Topic **Page**

4.1 Monitoring the DAT3 and CLK Pins

4.1.1 Detecting Edges and Level Changes on the DAT3 Pin

Register(Field)	Value	Description
MMCST0(DATEG)	0	No edge has been detected on the DAT3 pin.
	1	An edge has been detected on the DAT3 pin
MMCST1(DAT)	0	The signal level on DAT3 is low.
	1	The signal level on DAT3 is high.

Detecting edges. The MMC controller sets the DATEG flag of status register 0 (MMCST0) if DAT3 edge detection is enabled (DATEG is nonzero in MMCCTL) and the specified edge is detected. The CPU can also be notified of the DAT3 edge by an interrupt if you enable the interrupt request in the interrupt enable register (DATEG = 1 in MMCIE).

Detecting level changes. The DAT bit of status register 1 tracks the signal level on the DAT3 pin.

4.1.2 Checking the Status of the CLK Pin

Register(Field)	Value	Description
MMCST1(CLKSTP)	0	CLK is active. The memory clock signal is being driven on the pin.
	1	CLK is held low. Possible reasons are a manual stop (CLKEN = 0), a data receive full condition, or a data transmit empty condition.

Read CLKSTP to determine whether the memory clock has been stopped on the CLK pin.

4.2 Monitoring Data Transfers

4.2.1 Determining Whether New Data is Available in MMCDRR

Register(Field)	Value	Description
MMCST0(DRRDY)	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.

The MMC controller sets the DRRDY flag of MMCST0 when new data arrives in the data receive register (MMCDRR). The CPU can also be notified of the event by an interrupt if you enable the interrupt request (DRRDY = 1 in MMCIE).

4.2.2 Verifying That MMCDXR is Ready to Accept New Data

Register(Field)	Value	Description
MMCST0(DXRDY)	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.

The MMC controller sets the DXRDY flag of MMCST0 when data leaves the data transmit register (DXRDY). The CPU can also be notified of the event by an interrupt if you enable the interrupt request (DXRDY = 1 in MMCIE).

4.2.3 Checking for CRC Errors

Register(Field)	Value	Description
MMCST0(CRCRS)	0	No response CRC error has been detected.
	1	A response CRC error has been detected.
MMCST0(CRCRD)	0	No read-data CRC error has been detected.
	1	A read-data CRC error has been detected.
MMCST0(CRCWR)	0	No write-data CRC error has been detected.
	1	A write-data CRC error has been detected.

The MMC controller sets one of these flags in response to the corresponding CRC error. The CPU can also be notified of the CRC error by an interrupt if you enable the interrupt request (CRCRS/CRCRD/CRCWR = 1 in MMCIE).

4.2.4 Checking for Time-Out Events

Register(Field)	Value	Description
MMCST0(TOUTRS)	0	A response time-out event has not been detected.
	1	A response time-out event has been detected.
MMCST0(TOUTRD)	0	A read-data time-out event has not been detected.
	1	A read-data time-out event has been detected.

The MMC controller sets one of these flags in response to a the corresponding time-out event. The CPU can also be notified of the time-out event by an interrupt if you enable the interrupt request (TOUTRS/TOUTRD = 1 in MMCIE).

4.2.5 Determining When a Response/Command is Done

Register(Field)	Value	Description
MMCST0(RSPDNE)		**If the command requires a response:**
	0	The response has not been fully received with no CRC error.
	1	The response has been fully received with no CRC error.
		If no response required:
	0	The command has not been sent.
	1	The command has been sent.

The MMC controller sets the RSPDNE flag when the response is done (or, for commands that do not require a response, when the command is done). The CPU can also be notified of the done condition by an interrupt if you enable the interrupt request (RSPDNE = 1 in MMCIE).

4.2.6 Determining Whether the Memory Card is Busy

Register(Field)	Value	Description
MMCST0(BSYDNE)	0	The memory card is busy.
	1	The memory card is no longer sending a busy signal.
MMCST1(BUSY)	0	The memory card has not sent a busy signal.
	1	The memory card is busy.

The card sends a busy signal either as an expected part of an R1b response or to indicate that the card is still programming the last write data into its flash memory. The MMC controller has two flags to tell you whether the memory card is sending a busy signal. The two flags are complements of each other:

- BSYDNE is set if the card did not send or is not sending a busy signal. As with the other bits in status register 0, this bit has an associated interrupt that can be enabled (BSYDNE = 1 in MMCIE).
- BUSY is set when a busy signal is received from the card.

4.2.7 Determining Whether a Data Transfer is Done

Register(Field)	Value	Description
MMCST0(DATDNE)		**When reading from memory card:**
	0	The data has not been fully received with no CRC error.
	1	The data has been fully received with no CRC error.
		When writing to memory card:
	0	The data has not been fully transmitted.
	1	The data has been fully transmitted.

The MMC controller sets the DATDNE flag when all the bytes of a data transfer have been transmitted/received. You can poll this bit 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 can also be notified of the data-done event by an interrupt if you enable the interrupt request (DATDNE = 1 in MMCIE).

4.2.8 Checking for a Data Transmit Empty Condition

Register(Field)	Value	Description
MMCST1(DXEMP)	0	A data-transmit-empty condition has not been detected.
	1	A data-transmit-empty condition has been detected.

During transmission, a data value is passed from the data transmit register (MMCDXR) to the data transmit shift register. Then the value is passed from this shift register to the memory card, one bit at a time. The DXEMP bit indicates when this shift register is empty (when there are no bits available to shift out to the memory card).

Typically, this bit is not used to control data transfers; it is checked during recovery from an error condition. There is no interrupt associated with the data-transmit-empty condition.

4.2.9 Checking for a Data Receive Full Condition

Register(Field)	Value	Description
MMCST1(DRFUL)	0	A data-receive-full condition has not been detected.
	1	A data-receive-full condition has been detected.

During reception, the data receive shift register accepts a data value, one bit at a time. Then the whole value is passed from this shift register to the data receive register (MMCDRR). The DRFUL bit indicates when this shift register is full. At that time, no new bits can be shifted in from the memory card.

Typically, this bit is not used to control data transfers; it is checked during recovery from an error condition. There is no interrupt associated with the data-receive-full condition.

4.2.10 Getting the CRC Status Token After a Block is Written

Register(Field)	Value	Description
MMCDRSP(7−0)	00h−FFh	CRC status token from the memory card.

After the MMC controller sends a data block to a memory card, the memory card returns a CRC status token. This token is stored in the data response register (MMCDRSP).

4.2.11 Getting the Remaining Block Count During a Multiple-Block Transfer

Register(Field)	Value	Description
MMCNBLC(15−0)	n = 1−65535	There are n blocks left to be transferred.

During a transfer of multiple data blocks, the block counter register (MMCNBLC) tells you how many blocks are left to be transferred.

Chapter 5

MMC Controller Registers

This chapter provides a summary and detailed descriptions of the registers in the MMC controller.

5.1 Summary of the MMC Controller Registers

The MMC controller registers are listed in Table 5−1. These registers are accessible at 16-bit addresses in the I/O space of the DSP. The x's in the Address column of Table 5−1 indicate the part of the address that is different for each of the two MMC controllers on the DSP. For the first MMC controller, the start address is 4800h. For the second MMC controller, the start address is 4C00h.

Table 5−1. MMC Controller I/O-Mapped Registers

Address	Name	Description	See ...
xx00h	MMCFCLK	Function Clock Control Register	Page 5-3
xx01h	MMCCTL	MMC Control Register	Page 5-4
xx02h	MMCCLK	Clock Control Register	Page 5-6
xx03h	MMCST0	Status Register 0	Page 5-7
xx04h	MMCST1	Status Register 1	Page 5-10
xx05h	MMCIE	Interrupt Enable Register	Page 5-12
xx06h	MMCTOR	Response Time-Out Register	Page 5-14
xx07h	MMCTOD	Data Read Time-Out Register	Page 5-15
xx08h	MMCBLEN	Block Length Register	Page 5-15
xx09h	MMCNBLK	Number of Blocks Register	Page 5-16
xx0Ah	MMCNBLC	Number of Blocks Counter Register	Page 5-16
xx0Bh	MMCDRR	Data Receive Register	Page 5-17
xx0Ch	MMCDXR	Data Transmit Register	Page 5-17

Table 5−1. MMC Controller I/O-Mapped Registers (Continued)

Address	Name	Description	See ...
xx0Dh	MMCCMD	Command Register	Page 5-18
xx0Eh	MMCARGL	Argument Register, Low	Page 5-20
xx0Fh	MMCARGH	Argument Register, High	Page 5-20
xx10h	MMCRSP0	Response Register 0	Page 5-22
xx11h	MMCRSP1	Response Register 1	Page 5-22
xx12h	MMCRSP2	Response Register 2	Page 5-22
xx13h	MMCRSP3	Response Register 3	Page 5-22
xx14h	MMCRSP4	Response Register 4	Page 5-22
xx15h	MMCRSP5	Response Register 5	Page 5-22
xx16h	MMCRSP6	Response Register 6	Page 5-22
xx17h	MMCRSP7	Response Register 7	Page 5-22
xx18h	MMCDRSP	Data Response Register	Page 5-24
xx19h	–	Reserved	–
xx1Ah	MMCCIDX	Command Index Register	Page 5-24

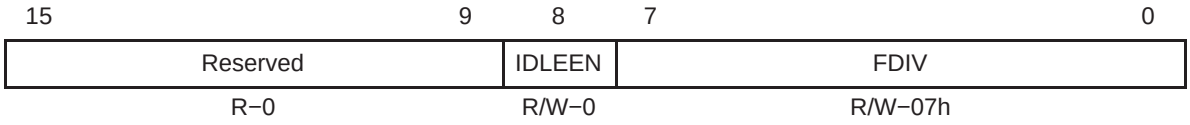
5.2 Function Clock Control Register (MMCFCLK)

Use MMCFCLK to:

- Select whether the MMC controller can be placed into an idle state when the peripherals domain of the DSP is turned off with an IDLE instruction (IDLEEN bit).
- Select how much the CPU clock is divided down to produce the function clock (FDIV bits). The MMC controller operates at the frequency of the function clock. For more details about clock generation, see section 1.4 on page 1-8.

Figure 5−1 and Table 5−2 summarize MMCFCLK.

Figure 5−1. Function Clock Control Register (MMCFCLK)



Legend: R = Read; W = Write; -*n* = Value after hardware reset

Table 5−2. Function Clock Control Register (MMCFCLK) Field Descriptions

Bit	Field	Value	Description
15−9	Reserved		These read-only reserved bits always return 0s.
8	IDLEEN		IDLE enable
		0	The function clock cannot be stopped by an IDLE instruction.
		1	If an IDLE instruction makes the peripherals domain idle, the MMC controller is idle (the function clock is stopped).
7−0	FDIV	1−255	Use this field to set the divide-down value for the function clock. The CPU clock is divided as follows to create the function clock: function clock frequency = CPU clock frequency/(FDIV + 1)

5.3 MMC Control Register (MMCCTL)

Use MMCCTL (see Figure 5−2 and Table 5−3) to enable or configure various modes of the MMC controller. Set or clear the DATRST and CMDRST bits at the same time to reset or enable the MMC controller.

Figure 5−2. MMC Control Register (MMCCTL)

15	12	11	9	8
Reserved		Reserved†		DMAEN
R−0		R/W−0		R/W−0

7	6	5	3	2	1	0
DATEG		Reserved†		WIDTH	CMDRST	DATRST
R/W−00		R/W−0		R/W−0	R/W−0	R/W−0

Legend: R = Read; W = Write; -*n* = Value after hardware reset

† Keep the default value (0) in these reserved bits.

Table 5−3. MMC Control Register (MMCCTL) Field Descriptions

Bit	Field	Value	Description
15−12	Reserved		These read-only reserved bits always return 0s.
11−9	Reserved	0	Keep the default value (0) in these reserved bits.
8	DMAEN		DMA event enable
		0	Disable DMA events.
		1	Enable DMA events.
7−6	DATEG		DAT3 edge detection select
		00b	DAT3 edge detection is disabled.
		01b	DAT3 rising edge detection is enabled.
		10b	DAT3 falling edge detection is enabled.
		11b	DAT3 dual edge detection is enabled (both edges detected).
5−3	Reserved	0	Keep the default value (0) in these reserved bits.
2	WIDTH		Data bus width select
		0	The data bus has 1 bit (DAT0 is used). The MMC protocol is selected.
		1	The data bus has 4 bits (DAT0−3 are used). The SD protocol is selected.

Table 5−3. MMC Control Register (MMCCTL) Field Descriptions (Continued)

Bit	Field	Value	Description
1	CMDRST		CMD (command) logic reset
		0	The CMD logic of the MMC controller is enabled.
		1	The CMD logic of the MMC controller is in the reset state.
0	DATRST		DAT (data) logic reset
		0	The DAT logic of the MMC controller is enabled.
		1	The DAT logic of the MMC controller is in the reset state.

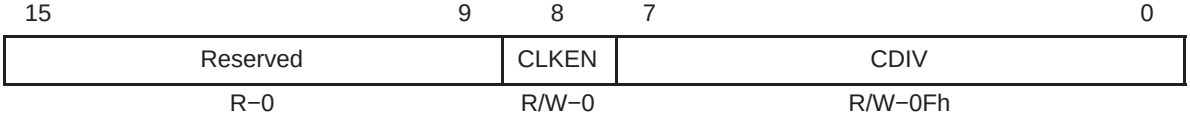
5.4 Clock Control Register (MMCCLK)

Use MMCCLK 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 (CDIV 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 1.4 on page 1-8.

Figure 5−3 and Table 5−4 summarize MMCCLK.

Figure 5−3. Clock Control Register (MMCCLK)



Legend: R = Read; W = Write; -*n* = Value after hardware reset

Table 5−4. Clock Control Register (MMCCLK) Field Descriptions

Bit	Field	Value	Description
15−9	Reserved		These read-only reserved bits always return 0s.
8	CLKEN		CLK pin enable
		0	The CLK pin is disabled and fixed low.
		1	The CLK pin is enabled; it shows the memory clock signal.
7−0	CDIV	0−255	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(CDIV + 1))

5.5 Status Register 0 (MMCST0)

The status bits in MMCST0 (see Figure 5−4 and Table 5−5) record specific events or errors. The transition from 0 to 1 of 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 MMCIE, which is described in section 5.7 (page 5-12).

In most cases, when a status bit is read, it is cleared. The two exceptions are the DRRDY bit and the DXRDY bit; these bits are cleared only in response to the functional events described for them in Table 5−5, or in response to a hardware reset.

Figure 5−4. Status Register 0 (MMCST0)

15			12	11	10	9	8
Reserved				DATEG	DRRDY	DXRDY†	Reserved†
R−0				R−0	R−0	R−1	R−0
7	6	5	4	3	2	1	0
CRCRS	CRCRD	CRCWR	TOUTRS	TOUTRD	RSPDNE	BSYDNE	DATDNE
R−0	R−0	R−0	R−0	R−0	R−0	R−0	R−0

Legend: R = Read; -*n* = Value after hardware reset
† The reset values shown for bits 9 and 8 are valid when the MMC controller stabilizes after reset.

Table 5−5. Status Register 0 (MMCST0) Field Descriptions

Bit	Field	Value	Description
15−12	Reserved		These read-only reserved bits always return 0s.
11	DATEG		DAT3 edge detected
		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), when a command is sent with data receive/transmit clear (DCLR = 1), or when data is read from MMCDRR.
		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.

Table 5−5. Status Register 0 (MMCST0) Field Descriptions (Continued)

Bit	Field	Value	Description
9	DXRDY		Data transmit ready
			DXRDY is set to 1 when the DAT logic is reset (DATRST = 1), when a command is sent with data receive/transmit clear (DCLR = 1), or when data is written to 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		This read-only reserved bit always returns 0.
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.
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.

Table 5−5. Status Register 0 (MMCST0) Field Descriptions (Continued)

Bit	Field	Value	Description
2	RSPDNE		Command/response done
			If the command requires a response:
		0	The response has not been fully received with no CRC error.
		1	The response has been fully received with no CRC error.
			If no response required:
		0	The command has not been sent.
		1	The command has been sent.
1	BSYDNE		Busy done
			BSYDNE is used for commands with an R1b response. BSYDNE is set to indicate that the card is no longer busy.
		0	The memory card is busy.
		1	The memory card is no longer sending a busy signal.
0	DATDNE		Data transfer done
			When reading from memory card:
		0	The data has not been fully received with no CRC error.
		1	The data has been fully received with no CRC error.
			When writing to memory card:
		0	The data has not been fully transmitted.
		1	The data has been fully transmitted.

5.6 Status Register 1 (MMCST1)

The status bits in MMCST1 (see Figure 5−5 and Table 5−6) record specific events or errors. There are no interrupts associated with these events or errors.

Figure 5−5. Status Register 1 (MMCST1)

15	8
Reserved	
R−0	

7	5	4	3	2	1	0
Reserved		DAT	DRFUL	DXEMP	CLKSTP	BUSY
R−0		R−0	R−0	R−0	R−0	R−0

Legend: R = Read; -*n* = Value after hardware reset

Table 5−6. Status Register 1 (MMCST1) Field Descriptions

Bit	Field	Value	Description
15−5	Reserved		These read-only reserved bits always return 0s.
4	DAT		DAT3 level
		0	The signal level on the DAT3 pin is low.
		1	The signal level on the DAT3 pin is high.
3	DRFUL		Data receive full
		0	A data receive full condition has not been detected. The data receive shift register is not full.
		1	A data receive full condition has been detected. The data receive shift register is full. No new bits can be shifted in from the memory card.
2	DXEMP		Data transmit empty
		0	A data transmit empty condition has not been detected. The data transmit shift register is not empty.
		1	A data transmit empty condition has been detected. The data transmit shift register is empty. No bits are available to be shifted out to the memory card.

Table 5−6. Status Register 1 (MMCST1) Field Descriptions (Continued)

Bit	Field	Value	Description
1	CLKSTP		Clock stopped
		0	The CLK pin is active. The memory clock signal is being driven on the pin.
		1	The CLK pin is held low. Possible reasons are a manual stop (CLKEN = 0), a data receive full condition, or a data transmit empty condition.
0	BUSY		Busy
		0	A busy signal has not been detected.
		1	A busy signal has been detected (the memory card is busy).

5.7 Interrupt Enable Register (MMCIE)

This register is used to enable or disable status interrupts. To disable an interrupt, clear the corresponding bit in MMCIE; to enable it, set the bit. Figure 5−6 and Table 5−7 summarize MMCIE.

Figure 5−6. Interrupt Enable Register (MMCIE)

15				12	11	10	9	8
Reserved					DATEG	DRRDY	DXRDY	Reserved†
R−0					R/W−0	R/W−0	R/W−0	R/W−0
7	6	5	4		3	2	1	0
CRCRS	CRCRD	CRCWR	TOUTRS		TOUTRD	RSPDNE	BSYDNE	DATDNE
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 = Read; W = Write; -*n* = Value after hardware reset

† Keep the default value (0) in this reserved bit.

Table 5−7. Interrupt Enable Register (MMCIE) Field Descriptions

Bit	Field	Value	Description
15−12	Reserved		These read-only reserved bits always return 0s.
11	DATEG		DAT3 edge interrupt enable
		0	The DAT3 edge detect interrupt is disabled.
		1	The DAT3 edge detect interrupt is enabled.
10	DRRDY		Data receive ready interrupt enable
		0	The data receive ready interrupt is disabled.
		1	The data receive ready interrupt is enabled.
9	DXRDY		Data transmit ready interrupt enable
		0	The data transmit ready interrupt is disabled.
		1	The data transmit ready interrupt is enabled.
8	Reserved	0	Keep the default value (0) in this reserved bit.
7	CRCRS		Response CRC error interrupt enable
		0	The response CRC error interrupt is disabled.
		1	The response CRC error interrupt is enabled.

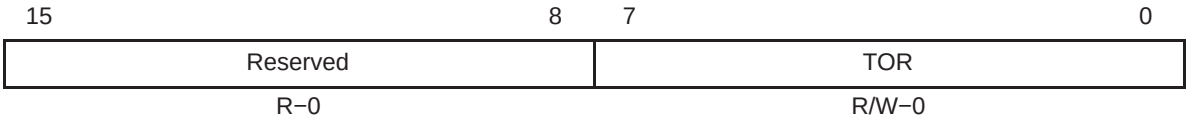
Table 5−7. Interrupt Enable Register (MMCIE) Field Descriptions (Continued)

Bit	Field	Value	Description
6	CRCRD		Read-data CRC error interrupt enable
		0	The read-data CRC error interrupt is disabled.
		1	The read-data CRC error interrupt is enabled.
5	CRCWR		Write-data CRC error interrupt enable
		0	The write-data CRC error interrupt is disabled.
		1	The write-data CRC error interrupt is enabled.
4	TOUTRS		Response time-out interrupt enable
		0	The response time-out interrupt is disabled.
		1	The response time-out interrupt is enabled.
3	TOUTRD		Read-data time-out interrupt enable
		0	The read-data time-out interrupt is disabled.
		1	The read-data time-out interrupt is enabled.
2	RSPDNE		Response/command done interrupt enable
		0	The response/command done interrupt is disabled.
		1	The response/command done interrupt is enabled.
1	BSYDNE		Busy done interrupt enable
		0	The busy done interrupt is disabled.
		1	The busy done interrupt is enabled.
0	DATDNE		Data transfer done interrupt enable
		0	The data transfer done interrupt is disabled.
		1	The data transfer done interrupt is enabled.

5.8 Response Time-Out Register (MMCTOR)

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 MMCST0. If the corresponding bit is set in MMCIE, an interrupt is generated when TOUTRS is set. MMCTOR is summarized in Figure 5−7 and Table 5−8. If a memory card should require a longer time-out period than MMCTOR can provide, a software time-out mechanism can be implemented.

Figure 5−7. Response Time-Out Register (MMCTOR)



Legend: R = Read; W = Write; -*n* = Value after hardware reset

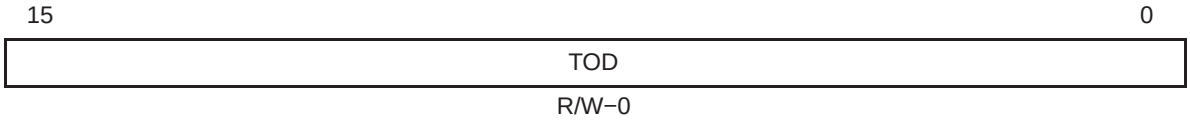
Table 5−8. Response Time-Out Register (MMCTOR) Field Descriptions

Bit	Field	Value	Description
15−8	Reserved		These read-only reserved bits always return 0s.
7−0	TOR		Time-out period for response
		0	No time-out
		01h−FFh	1 CLK clock cycle to 255 CLK clock cycles

5.9 Data Read Time-Out Register (MMCTOD)

When the MMC controller has requested data from a memory card, MMCTOD defines how long the MMC controller waits for the data before recording a time-out condition in the TOUTRD bit of MMCST0. If the corresponding bit is set in MMCIE, an interrupt is generated when TOUTRD is set. MMCTOD is summarized in Figure 5−8 and Table 5−9. If a memory card should require a longer time-out period than MMCTOD can provide, a software time-out mechanism can be implemented.

Figure 5−8. Data Read Time-Out Register (MMCTOD)



Legend: R = Read; W = Write; -*n* = Value after hardware reset

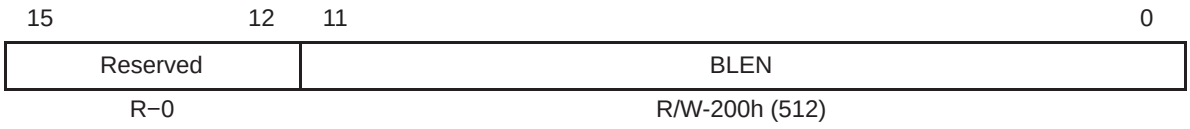
Table 5−9. Data Read Time-Out Register (MMCTOD) Field Description

Bit	Field	Value	Description
15−0	TOD		Time-out period for data read
		0	No time-out
		0001h−FFFFh	1 CLK clock cycles to 65535 CLK clock cycles

5.10 Block Length Register (MMCBLEN)

MMCBLEN specifies the data block length in bytes. This value must match the block length setting in the memory card. The default value in this register after a hardware reset is 512. Figure 5−9 and Table 5−10 summarize MMCBLEN.

Figure 5−9. Block Length Register (MMCBLEN)



Legend: R = Read; W = Write; -*n* = Value after hardware reset

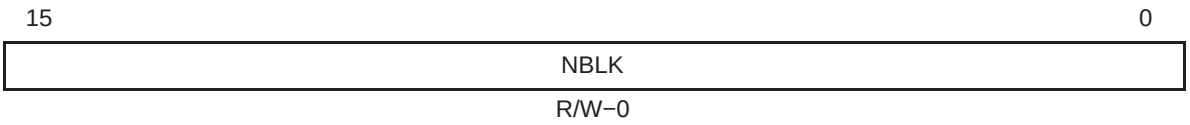
Table 5−10. Block Length Register (MMCBLEN) Field Descriptions

Bit	Field	Value	Description
15−12	Reserved		These read-only reserved bits always return 0s.
11−0	BLEN	1−512	Use this field to set the block length, which is the byte count of a data block. The value 0 is prohibited.

5.11 Number of Blocks Register (MMCNBLK)

MMCNBLK is used for specifying the number of blocks for a multiple-block transfer. Figure 5−10 and Table 5−11 summarize MMCNBLK.

Figure 5−10. Number of Blocks Register (MMCNBLK)



Legend: R = Read; W = Write; -*n* = Value after hardware reset

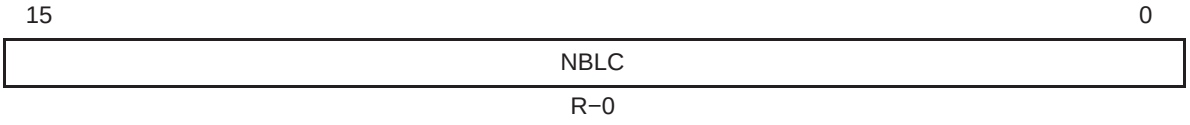
Table 5−11. Number of Blocks Register (MMCNBLK) Field Description

Bit	Field	Value	Description
15−0	NBLK		Use this field 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 MMCCMD.
		n = 1−65535	n blocks. The MMC controller reads/writes only n blocks of data, even if the STOP_TRANSMISSION command has not been written to MMCCMD yet.

5.12 Number of Blocks Counter Register (MMCNBLC)

MMCNBLC is a down counter for tracking the number of blocks left to be transferred during a multiple-block transfer. Figure 5−11 and Table 5−12 summarize MMCNBLC.

Figure 5−11. Number of Blocks Counter Register (MMCNBLC)



Legend: R = Read; -*n* = Value after hardware reset

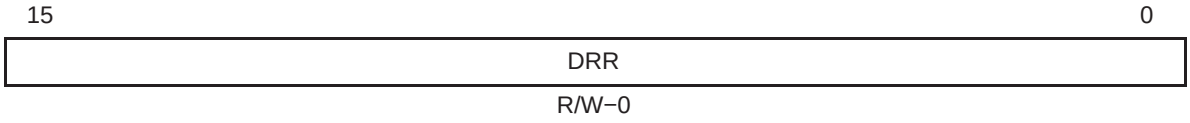
Table 5−12. Number of Blocks Counter Register (MMCNBLC) Field Description

Bit	Field	Value	Description
15−0	NBLC	0−65535	Read this field to determine the number of blocks left to be transferred.

5.13 Data Receive Register (MMCDRR)

Data comes into the MMC controller via MMCDRR (see Figure 5−12 and Table 5−13). The CPU or the DMA controller can read data from this register.

Figure 5−12. Data Receive Register (MMCDRR)



Legend: R = Read; W = Write; -*n* = Value after hardware reset

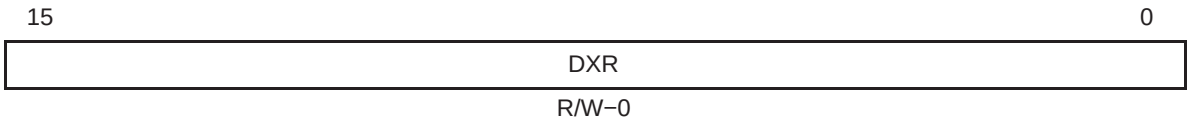
Table 5−13. Data Receive Register (MMCDRR) Field Description

Bit	Field	Value	Description
15−0	DRR	0000h−FFFFh	This field holds data received by the MMC controller.

5.14 Data Transmit Register (MMCDXR)

Data exits the MMC controller via MMCDXR (see Figure 5−13 and Table 5−14). The CPU or the DMA controller can write data to this register.

Figure 5−13. Data Transmit Register (MMCDXR)



Legend: R = Read; W = Write; -*n* = Value after hardware reset

Table 5−14. Data Transmit Register (MMCDXR) Field Description

Bit	Field	Value	Description
15−0	DXR	0000h−FFFFh	Data to be transmitted by the MMC controller must be written to this field.

5.15 Command Register (MMCCMD)

Note:

Writing to MMCCMD causes the MMC controller to send the programmed command. Therefore, the argument registers (MMCARGH and MMCARGL) have to be loaded properly before a write to MMCCMD.

When the DSP writes to MMCCMD, the MMC controller sends the programmed command, including any arguments in the argument registers (MMCARGH and MMCARGL). For the format of a command (index, arguments, and other bits), see the description for the argument registers (section 5.16 on page 5-20).

Figure 5−14 and Table 5−15 summarize MMCCMD. The CMD field of MMCCMD specifies the type of command to be sent. The other fields define the operation (command, response, additional activity) for the MMC controller.

The content of MMCCMD is kept after the transfer to the transmit shift register.

Figure 5−14. Command Register (MMCCMD)

15	14	13	12	11	10	9	8
DCLR	INIT	DATA	STREAM	WRITE	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
Reserved		CMD	
R−0		R/W−0	

Legend: R = Read; W = Write; -*n* = Value after hardware reset

Table 5−15. Command Register (MMCCMD) Field Descriptions

Bit	Field	Value	Description
15	DCLR		Data receive/transmit clear. Use this bit to clear the data receive ready (DRRDY) and data transmit ready (DXRDY) bits before a new read or write sequence. This clears any previous status.
		0	Do not clear DRRDY and DXRDY.
		1	Clear DRRDY and DXRDY.
14	INIT		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 field. These dummy clock cycles are required for resetting a card after power on.

Table 5−15. Command Register (MMCCMD) Field Descriptions (Continued)

Bit	Field	Value	Description
13	DATA		Data transfer indicator
		0	There is to be no data transfer.
		1	There is a data transfer associated with the command.
12	STREAM		Stream enable
		0	If DATA = 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 DATA = 1, the data transfer is a stream transfer. Once the data transfer is started, it does not stop until the MMC controller issues a stop command to the memory card.
11	WRITE		Write enable
		0	If DATA = 1, the data transfer is a read operation.
		1	If DATA = 1, the data transfer is a write operation.
10−9	RSPFMT		Response format (expected type of response to the command)
		00b	No response
		01b	R1, R4, R5, or R6. 48 bits with CRC.
		10b	R2. 136 bits with CRC.
		11b	R3. 48 bits with no CRC.
8	BSYEXP		Busy expected. If an R1b (R1 with busy) response is expected, set RSPFMT = 01 and BSYEXP = 1.
		0	A busy signal is not expected.
		1	A busy signal is expected.
7−6	Reserved		These read-only reserved bits always return 0s.
5−0	CMD	xxxxxxb	Command index. This field must contain the command index for the command to be sent to the memory card.

5.16 Argument Registers (MMCARGH and MMCARGL)

MMCARGH and MMCARGL are used for specifying the arguments to be sent with the command specified in MMCCMD. Writing to MMCCMD causes the MMC controller to send a command; load MMCARGH and MMCARGL before writing to MMCCMD. The contents of the argument registers are kept after the transfer to the shift register.

Note:

Do not modify the argument registers while they are being used for an operation.

Figure 5−15, Table 5−16, and Table 5−17 summarize the argument registers. Table 5−18 shows the format for a command.

Figure 5−15. Argument Registers (MMCARGH and MMCARGL)

MMCARGH

15															0
ARGH															
R/W−0															

MMCARGL

15															0
ARGL															
R/W−0															

Legend: R = Read; W = Write; -*n* = Value after hardware reset

Table 5−16. Argument Register, High (MMCARGH) Field Description

Bit	Field	Value	Description
15−0	ARGH	0000h−FFFFh	High part (upper 16 bits) of argument

Table 5−17. Argument Register, Low (MMCARGL) Field Description

Bit	Field	Value	Description
15−0	ARGL	0000h−FFFFh	Low part (lower 16 bits) of argument

Table 5−18. Command Format

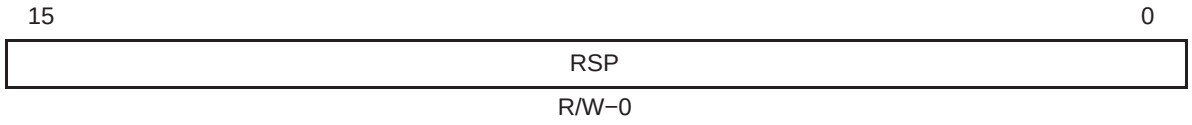
Register	Bit Position	Description
−	47	Start bit
−	46	Transmission bit
MMCCMD(5−0)	45−40	Command index
MMCARGH	39−24	Argument, high part
MMCARGL	23−8	Argument, low part
−	7−1	CRC7
−	0	End bit

5.17 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 eight response registers (MMCRSP7−MMCRSP0). The response registers are updated as the responses arrive, even if the CPU has not read the previous contents.

As shown in Figure 5−16, each of the response registers holds up to 16 bits. The tables that follow the figure show which registers are used for each type of response. Table 5−19 and Table 5−20 show response formats. The first byte of the response is a command index byte and is stored in the command index register (MMCCIDX; see section 5.19 on page 5-24).

Figure 5−16. Format of a Response Register (MMCRSPn)



Legend: R = Read; W = Write; -*n* = Value after hardware reset

Table 5−19. R1, R3, R4, R5, or R6 Response (48 Bits)

Register	Bits of the Response
MMCCIDX	47−40
MMCRSP7	39−24
MMCRSP6	23−8
MMCRSP5†	7−0
MMCRSP4−0	−

† Bits 7−0 of the response are stored to the lower half (bits 7−0) of MMCRSP5.

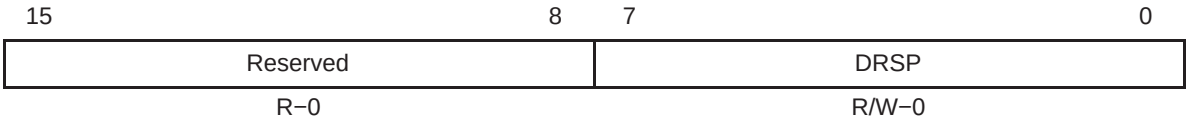
Table 5−20. R2 Response (136 Bits)

Register	Bits of the Response
MMCCIDX	135−128
MMCRSP7	127−112
MMCRSP6	111−96
MMCRSP5	95−80
MMCRSP4	79−64
MMCRSP3	63−48
MMCRSP2	47−32
MMCRSP1	31−16
MMCRSP0	15−0

5.18 Data Response Register (MMCDRSP)

After the MMC controller sends a data block to a memory card, the return byte from the memory card is stored in MMCDRSP. Figure 5−17 and Table 5−21 summarize this register.

Figure 5−17. Data Response Register (MMCDRSP)



Legend: R = Read; W = Write; -*n* = Value after hardware reset

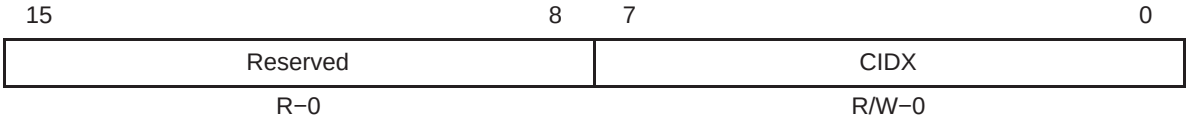
Table 5−21. Data Response Register (MMCDRSP) Field Descriptions

Bit	Field	Value	Description
15−8	Reserved		These read-only reserved bits always return 0s.
7−0	DRSP	xxh	During a write operation (see section 1.3.1 on page 1-5), the CRC status token is stored in this register .

5.19 Command Index Register (MMCCIDX)

The first byte of a response from a memory card is a command index byte. The MMC controller stores this byte in MMCCIDX (see Figure 5−18 and Table 5−22).

Figure 5−18. Command Index Register (MMCCIDX)



Legend: R = Read; W = Write; -*n* = Value after hardware reset

Table 5−22. Command Index Register (MMCCIDX) Field Descriptions

Bit	Field	Value	Description
15−8	Reserved		These read-only reserved bits always return 0s.
7−0	CIDX	xxh	When a command index byte is received, the byte is stored in this field. The byte consists of a start bit, a transmission bit, and a command index.

Index

D

E

F

I

M

N

O

P

R

S

T

W

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
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
Low Power Wireless	www.ti.com/lpw	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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