ARM PrimeCell™

MultiMedia Card Interface (PL181)

Technical Reference Manual

ARM

ARM DDI 0205B

ARM PrimeCell™
Technical Reference Manual

Release information

Change history

Date	Issue	Change
December 2000	A	First release
January 2001	B	Minor changes to testing of primary outputs in Chapter 4,

Proprietary notice

Document confidentiality status

This document is Open Access. This means there is no restriction on the distribution of the information.

Product status

The information in this document is Final (information on a developed product).

ARM web address

`http://www.arm.com`

Contents
ARM PrimeCell™ MultiMedia Card Interface (PL181) Technical Reference Manual

Preface

This preface introduces the ARM PrimeCell MultiMedia Card Interface (PL181) and its reference documentation. It contains the following sections:

About this document

This document is the technical reference manual for the ARM PrimeCell *MultiMedia Card Interface* (MMCI).

Intended audience

This document has been written for implementation engineers and architects, and provides a description of an optimal PrimeCell MMCI architecture. The PrimeCell MMCI provides an interface between the *Advanced Peripheral Bus* (APB) system bus and multimedia cards.

Using this manual

This document is organized into the following chapters:

Chapter 1 *Introduction*

Read this chapter for an introduction to the PrimeCell MMCI.

Chapter 2 *Functional Overview*

Read this chapter for an overview of the PrimeCell MMCI in a multimedia card system.

Chapter 3 *Programmer's Model*

Read this chapter for a description of the registers and for details of system initialization.

Chapter 4 *Programmer's Model for Test*

Read this chapter for a description of the additional logic for functional verification and production testing.

Appendix A *ARM PrimeCell MMCI (PL181) Signal Descriptions*

Read this appendix for a description of the PrimeCell MMCI signals.

Typographical conventions

The following typographical conventions are used in this document:

bold Highlights ARM processor signal names, and interface elements such as menu names. Also used for terms in descriptive lists, where appropriate.

italic Highlights special terminology, cross-references and citations.

`typewriter` Denotes text that may be entered at the keyboard, such as commands, file names and program names, and source code.

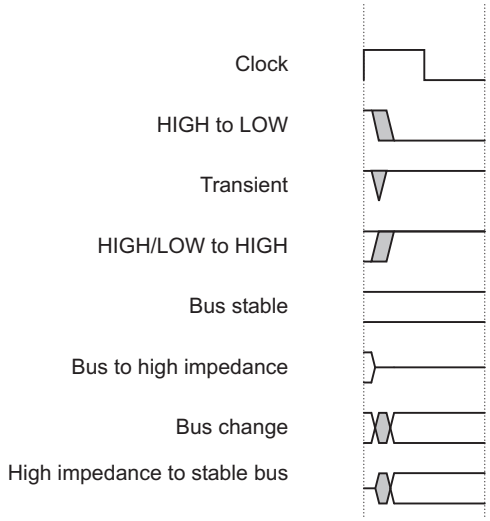
`type`writer Denotes a permitted abbreviation for a command or option. The underlined text may be entered instead of the full command or option name.

`typewriter italic`
Denotes arguments to commands or functions where the argument is to be replaced by a specific value.

`typewriter bold`
Denotes language keywords when used outside example code.

Timing diagram conventions

This manual contains one or more timing diagrams. The following key explains the components used in these diagrams. Any variations are clearly labeled when they occur. Therefore, no additional meaning should be attached unless specifically stated.



Key to timing diagram conventions

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.

Further reading

This section lists publications by ARM Limited, and by third parties.

ARM periodically provides updates and corrections to its documentation. See `http://www.arm.com` for current errata sheets and addenda.

See also the ARM Frequently Asked Questions list at:
`http://www.arm.com/DevSupp/Sales+Support/faq.html`

ARM publications

This document contains information that is specific to the ARM PrimeCell MMCI. Refer to the following documents for other relevant information:

- *AMBA Specification Rev 2.0* (ARM IHI 0011A)
- *MultiMedia Card Integration Manual* (PL181_INTM_0000)
- *MultiMedia Card Controller Design Manual* (PL181_DDES_0000).

Other publications

This section lists relevant documents published by third parties.

- *Multimedia Card System Specification v2.11*.

Feedback

ARM Limited welcomes feedback both on the ARM PrimeCell MultiMedia Card Interface, and on the documentation.

Feedback on the ARM PrimeCell MultiMedia Card Interface

If you have any comments or suggestions about this product, please contact your supplier giving:

- the product name
- a concise explanation of your comments.

Feedback on this document

If you have any comments on about this document, please send email to `errata@arm.com` giving:

- the document title
- the document number
- the page number(s) to which your comments refer
- a concise explanation of your comments.

General suggestions for additions and improvements are also welcome.

Chapter 1
Introduction

This chapter describes the ARM PrimeCell MultiMedia Card Interface (PL181) and contains the following section:

- *About the ARM PrimeCell MultiMedia Card Interface (PL181)* on page 1-2.

1.1 About the ARM PrimeCell MultiMedia Card Interface (PL181)

The PrimeCell *MultiMedia Card Interface* (MMCI) is an *Advanced Microcontroller Bus Architecture* (AMBA) compliant, *System-on-a-Chip* (SoC) peripheral that is developed, tested, and licensed by ARM.

See Figure 1-1 for a simplified block diagram of the PrimeCell MMCI.

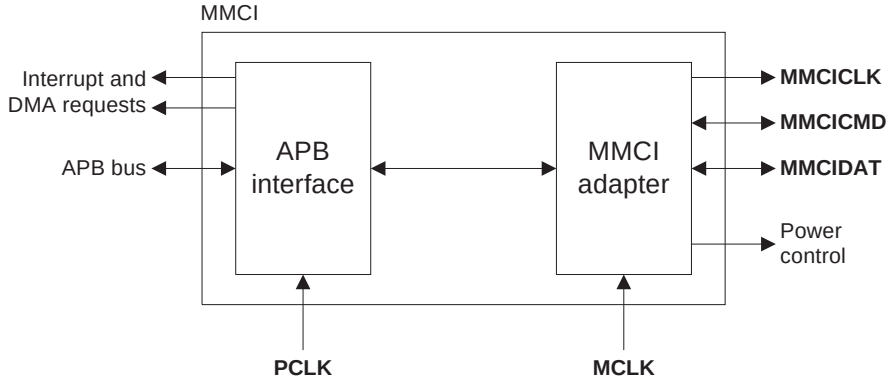


Figure 1-1 PrimeCell MMCI block diagram

The PrimeCell MMCI is an interface between the Advanced Peripheral Bus (APB) system bus and multimedia cards. It consists of two parts:

- The PrimeCell MMCI adapter block provides all functions specific to the multimedia card, such as the clock generation unit, power management control, command and data transfer.
- The APB interface accesses the PrimeCell MMCI adapter registers, and generates interrupt and DMA request signals.

1.1.1 Features of the PrimeCell MMCI

The following features are provided by the PrimeCell MMCI:

- Conformance to *Multimedia Card Specification v2.11*.
- Use as a multimedia card bus host. It can be connected to up to 30 cards as a multimedia card bus.

Chapter 2
Functional Overview

This chapter provides an overview of the PrimeCell *MultiMedia Card Interface* (MMCI) and its interface with the multimedia card system. It contains the following sections:

2.1 About the ARM PrimeCell MMCI (PL181)

The PrimeCell MMCI provides an interface between the APB system bus and multimedia cards. It consists of two parts:

- The PrimeCell MMCI adapter block provides all functions specific to the multimedia card. These include the clock generation unit, power management control, command and data transfer (see *PrimeCell MMCI adapter* on page 2-4 for more information).
- The APB interface provides access to the PrimeCell MMCI adapter registers, and generates interrupt and DMA request signals (see *APB interface* on page 2-19 for more information).

You can connect up to 30 cards as a multimedia card bus.

The PrimeCell MMCI uses two clock signals, each with a maximum frequency of 100 MHz. One or both clocks can be switched off for power saving:

- PrimeCell MMCI adapter clock (**MCLK**)
- APB bus clock (**PCLK**).

The relationship between **MCLK** and **PCLK** is defined below:

$$fp \geq \frac{3}{8} fm$$

where p = **PCLK** and m = **MCLK**.

Figure 2-1 shows the multimedia card system.

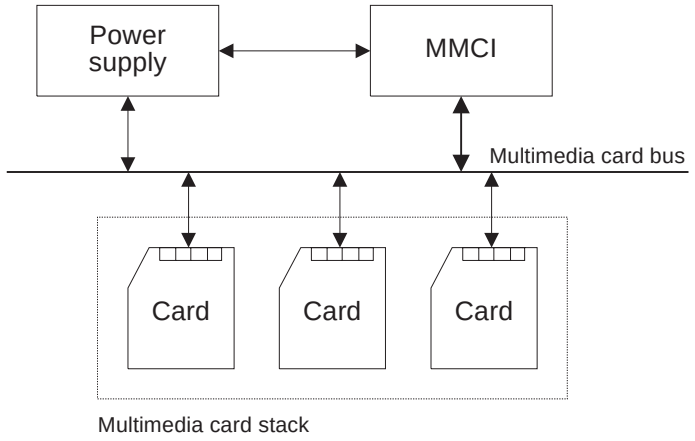


Figure 2-1 Multimedia card system

The multimedia card system provides communication and data storage, and consists of:

- A multimedia card stack. This can consist of up to 30 cards on a single physical bus.
- A multimedia card controller: This is the multimedia card master, and provides an interface between a system bus and the multimedia card bus.

Multimedia cards are grouped into three types according to their function:

- *Read Only Memory* (ROM) cards, containing preprogrammed data
- *Read/Write* (R/W) cards, used for mass storage
- *Input/Output* (I/O) cards, used for communication.

The multimedia card system transfers commands and data using three signal lines:

CLK One bit is transferred on both command and data lines with each clock cycle. The clock frequency varies between 0MHz and 20MHz.

CMD Bidirectional command channel that initializes a card and transfers commands. **CMD** has two operational modes:

- Open-drain for initialization
- Push-pull for command transfer.

DAT Bidirectional data channel, operating in push-pull mode.

2.2 PrimeCell MMCI adapter

Figure 2-2 shows a simplified block diagram of the PrimeCell MMCI adapter.

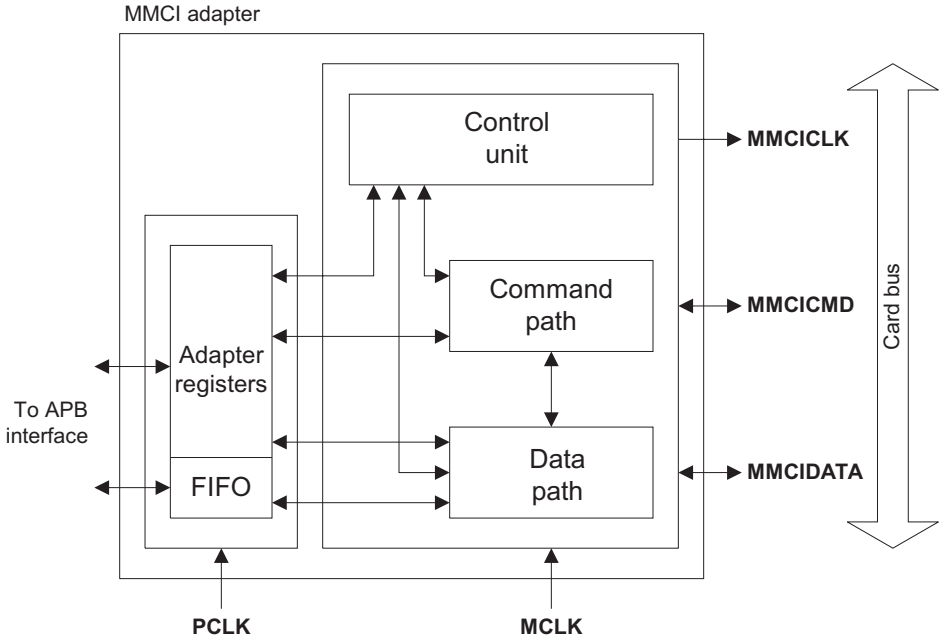


Figure 2-2 PrimeCell MMCI Adapter

The PrimeCell MMCI adapter is a multimedia card bus master that provides an interface to the multimedia card stack. It consists of five subunits:

- *Adapter register block* on page 2-5
- *Control unit* on page 2-5
- *Command path* on page 2-6
- *Data path* on page 2-11
- *Data FIFO* on page 2-16.

Note

The adapter registers and FIFO use the APB bus clock domain. The control unit, command path, and data path use the PrimeCell MMCI adapter clock domain.

In Figure 2-2, the power connections are not shown for clarity.

2.2.1 Adapter register block

The adapter register block contains all system registers. This block also generates the signals that clear the static flags in the MMCIStatus register (see Table 3-15 on page 3-13). The clear signals are generated when 1 is written into the corresponding bit location of the MMCIClear register (see Table 3-16 on page 3-14). The clear signal for flags generated in the **MCLK** domain is synchronized to that domain.

2.2.2 Control unit

The control unit contains the power management functions and the card bus clock divider. Figure 2-3 shows a block diagram of the control unit.

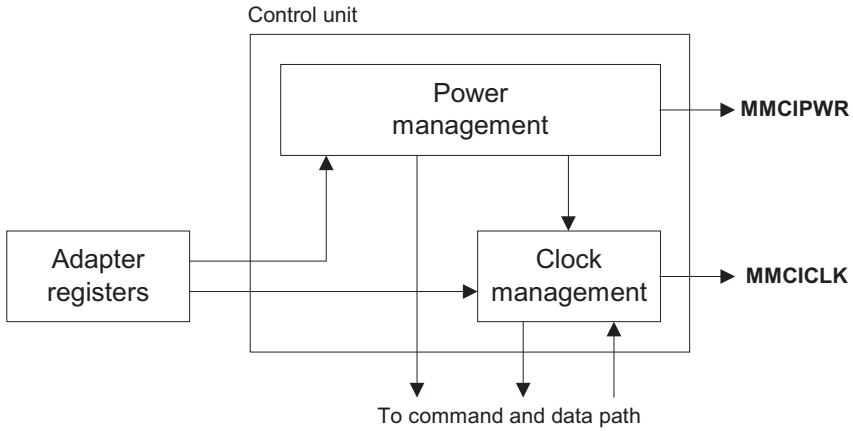


Figure 2-3 Control unit

There are three power phases:

- power-off
- power-up
- power-on.

The power management logic controls an external power supply unit, and disables the card bus output signals during the power-off or power-up phases. The power-up phase is a transition phase between the power-off and power-on phases, and allows an external power supply to reach the card bus operating voltage. A device driver is used to ensure that the PrimeCell MMCI remains in the power-up phase until the external power supply reaches the operating voltage.

The clock management logic generates and controls the **MMCICLK** signal. The **MMCICLK** output can use either a clock divide or clock bypass mode. The clock output is inactive:

- after the PrimeCell MMCI is reset
- during the power-off or power-up phases
- if the power saving mode is enabled and the card bus is in the IDLE state (eight clock periods after both the command and data path subunits enter the IDLE phase).

2.2.3 Command path

The command path subunit sends commands to and receives responses from the cards. Figure 2-4 shows a block diagram of the command path.

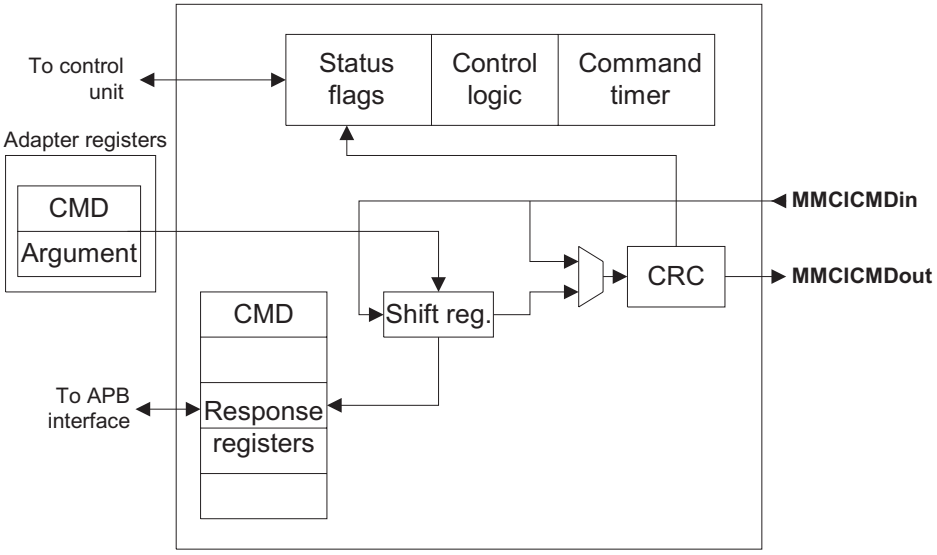


Figure 2-4 Command path

Command path state machine

When the command register is written to and the enable bit is set, command transfer starts. When the command has been sent, the *Command Path State Machine* (CPSM) sets the status flags and enters the IDLE state if a response is not required. If a response is required, it waits for the response (see Figure 2-5 on page 2-7). When the response is received, the received CRC code and the internally generated code are compared, and the appropriate status flags are set.

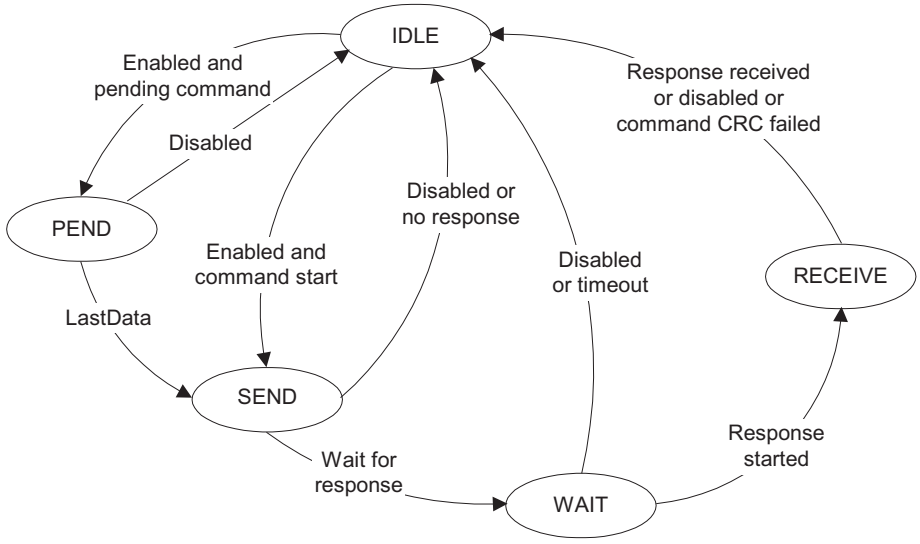


Figure 2-5 Command path state machine

When the WAIT state is entered, the command timer starts running. If the timeout is reached before the CPSM moves to the RECEIVE state, the timeout flag is set and the IDLE state is entered.

Note

The timeout period has a fixed value of 64 **MMCICLK** clock periods.

If the interrupt bit is set in the command register, the timer is disabled and the CPSM waits for an interrupt request from one of the cards. If a pending bit is set in the command register, the CPSM enters the PEND state, and waits for a **CmdPend** signal from the data path subunit. When **CmdPend** is detected, the CPSM moves to the SEND state. This enables the data counter to trigger the stop command transmission.

Note

The CPSM remains in the IDLE state for at least eight **MMCICLK** periods to meet Ncc and Nrc timing constraints.

Figure 2-6 on page 2-8 shows the PrimeCell MMCI command transfer.

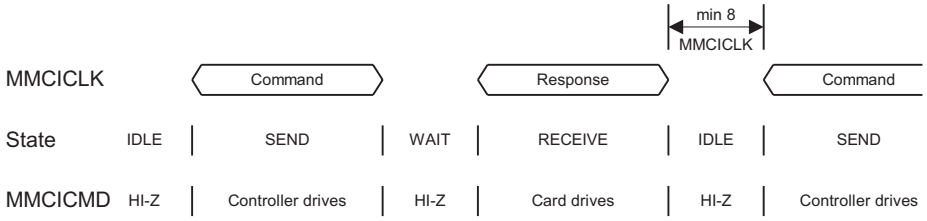


Figure 2-6 PrimeCell MMCI command transfer

Command format

The command path operates in a half-duplex mode, so that commands and responses can either be sent or received. If the CPSM is not in the SEND state, the **MMCICMD** output is in HI-Z state, as shown in Figure 2-6. Data on **MMCICMD** is synchronous to the rising **MMCICLK** edge. All commands have a fixed length of 48 bits. Table 2-1 shows the command format.

Table 2-1 Command format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	1	Transmission bit
[45:40]	6	-	Command index
[39:8]	32	-	Argument
[7:1]	7	-	CRC7
0	1	1	End bit

The PrimeCell MMCI adapter supports two response types. Both use CRC error checking:

- 48 bit short response (see Table 2-2 on page 2-9)
- 136 bit long response (see Table 2-3 on page 2-9).

Note

If the response does not contain CRC (CMD1 response), the device driver must ignore the CRC failed status.

Table 2-2 Short response format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	-	Command index
[39:8]	32	-	Argument
[7:1]	7	-	CRC7 (or 1111111)
0	1	1	End bit

Table 2-3 Long response format

Bit position	Width	Value	Description
135	1	0	Start bit
134	1	1	Transmission bit
[133:128]	6	111111	Reserved
[127:1]	127	-	CID or CSD (including internal CRC7)
0	1	1	End bit

The command register contains the command index (six bits sent to a card) and the command type. These determine whether the command requires a response, and whether the response is 48 or 136 bits long (see *Command register, MMCICommand* on page 3-8 for more information). The command path implements the status flags shown in Table 2-4 on page 2-10 (see *Status register, MMCIStatus* on page 3-13 for more information).

Table 2-4 Command path status flags

Flag	Description
CmdRespEnd	Set if response CRC is OK
CmdCrcFail	Set if response CRC fails
CmdSent	Set when command (that does not require response) is sent
CmdTimeOut	Response timeout
CmdActive	Command transfer in progress

The CRC generator calculates the CRC checksum for all bits before the CRC code. This includes the start bit, transmitter bit, command index, and command argument (or card status). The CRC checksum is calculated for the first 120 bits of CID or CSD for the long response format. Note that the start bit, transmitter bit and the six reserved bits are not used in the CRC calculation. The CRC checksum is a 7-bit value:

$CRC[6:0] = Remainder [(M(x) * x^7) / G(x)]$

$G(x) = x^7 + x^3 + 1$

$M(x) = (start\ bit) * x^{39} + ... + (last\ bit\ before\ CRC) * x^0$, or

$M(x) = (MSB) * x^{119} + ... + (last\ bit\ before\ CRC) * x^0$

2.2.4 Data path

The data path subunit transfers data to and from cards. Figure 2-7 shows a block diagram of the data path.

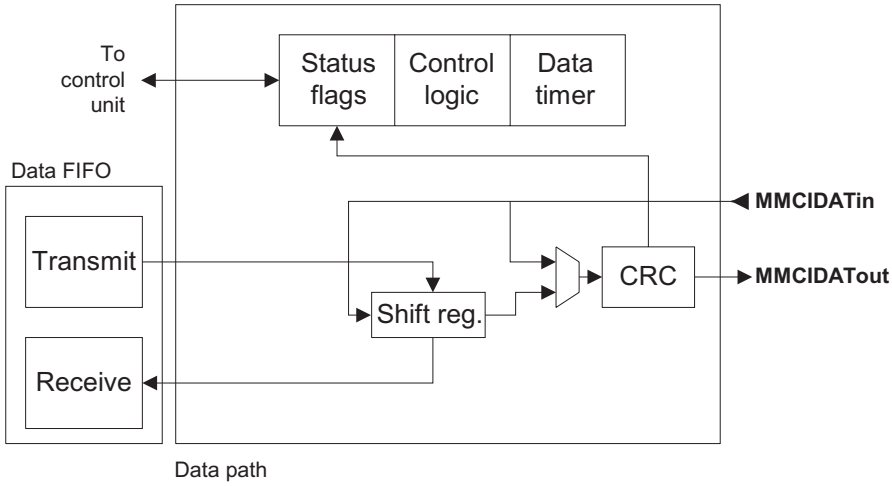


Figure 2-7 Data path

Depending on the transfer direction (send or receive), the *Data Path State Machine* (DPSM) moves to the WAIT_S or WAIT_R state when it is enabled:

Send The DPSM moves to the WAIT_S state. If there is data in the send FIFO, the DPSM moves to the SEND state, and the data path subunit starts sending data to a card.

Receive The DPSM moves to the WAIT_R state and waits for a start bit. When it receives a start bit, the DPSM moves to the RECEIVE state, and the data path subunit starts receiving data from a card.

Data path state machine

The DPSM operates at **MMCICLK** frequency. Data on the card bus signals is synchronous to the rising edge of **MMCICLK**. The DPSM has six states, as shown in Figure 2-8.

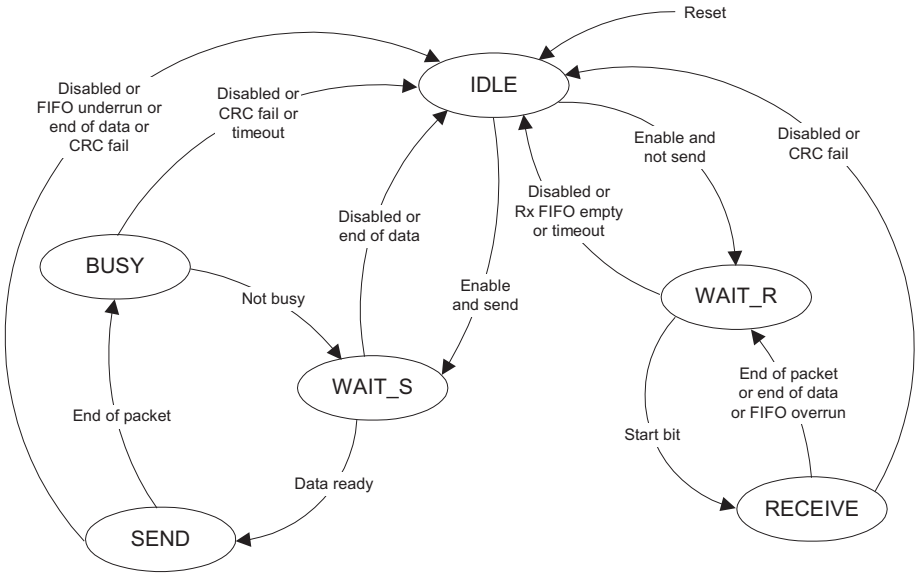


Figure 2-8 Data path state machine

IDLE The data path is inactive, and the **MMCIDAT** outputs are in HI-Z. When the data control register is written and the enable bit is set, the DPSM loads the data counter with a new value and, depending on the data direction bit, moves to either the WAIT_S or WAIT_R state.

WAIT_R If the data counter equals zero, the DPSM moves to the IDLE state when the receive FIFO is empty. If the data counter is not zero, the DPSM waits for a start bit on **MMCIDAT**.

The DPSM moves to the RECEIVE state if it receives a start bit before a timeout, and loads the data block counter. If it reaches a timeout before it detects a start bit, it moves to the IDLE state and sets the timeout status flag.

RECEIVE Serial data received from a card is packed in bytes and written to the data FIFO. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block or stream:

- In *block mode*, when the data block counter reaches zero, the DPSM waits until it receives the CRC code. If the received code matches the internally generated CRC code, the DPSM moves to the WAIT_R state. If not, the CRC fail status flag is set and the DPSM moves to the IDLE state.
- In *stream mode*, the DPSM receives data while the data counter is not zero. When the counter is zero, the remaining data in the shift register is written to the data FIFO, and the DPSM moves to the WAIT-R state.

If a FIFO overrun error occurs, the DPSM sets the FIFO error flag and moves to the WAIT_R state.

WAIT_S The DPSM moves to the IDLE state if the data counter is zero. If not, it waits until the data FIFO empty flag is deasserted, and moves to the SEND state.

Note

The DPSM remains in the WAIT_S state for at least two clock periods to meet Nwr timing constraints.

SEND The DPSM starts sending data to a card. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block or stream:

- In *block mode*, when the data block counter reaches zero, the DPSM sends an internally generated CRC code and end bit, and moves to the BUSY state.
- In *stream mode*, the DPSM sends data to a card while the enable bit is HIGH and the data counter is not zero. It then moves to the IDLE state.

If a FIFO underrun error occurs, the DPSM sets the FIFO error flag and moves to the IDLE state.

BUSY The DPSM waits for the CRC status flag:

- If it does not receive a positive CRC status, it moves to the IDLE state and sets the CRC fail status flag.
- If it receives a positive CRC status, it moves to the WAIT_S state if **MMCIDAT** is not LOW (the card is not busy).

If a timeout occurs while the DPSM is in the BUSY state, it sets the data timeout flag and moves to the IDLE state.

The data timer is enabled when the DPSM is in the WAIT_R or BUSY state, and generates the data timeout error:

- When transmitting data, the timeout occurs if the DPSM stays in the BUSY state for longer than the programmed timeout period.
- When receiving data, the timeout occurs if the end of the data is not true, and if the DPSM stays in the WAIT_R state for longer than the programmed timeout period.

Data counter

The data counter has two functions:

- To stop a data transfer when it reaches zero. This is the end of the data condition.
- To start transferring a pending command (see Figure 2-9). This is used to send the stop command for a stream data transfer.

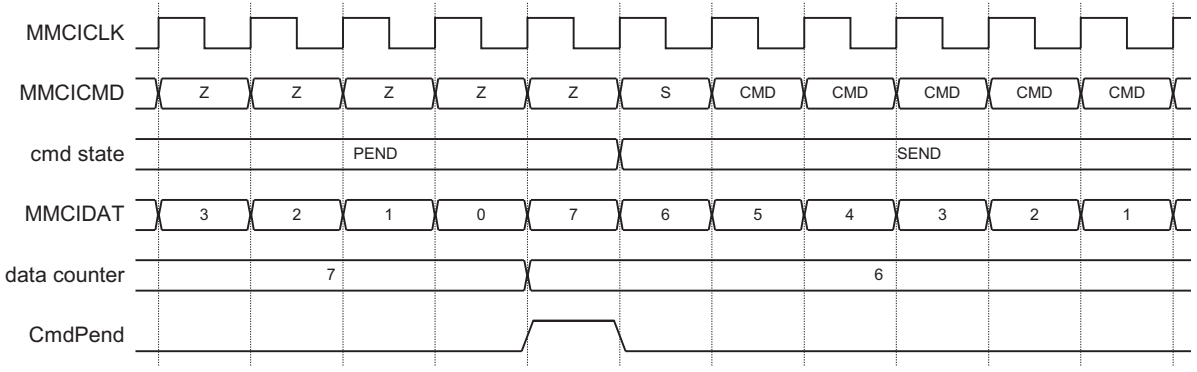


Figure 2-9 Pending command start

The data block counter determines the end of a data block. If the counter is zero, the end-of-data condition is TRUE (see *Data control register*, *MMCIDataCtrl* on page 3-11 for more information).

Bus mode

The data path also operates in half-duplex mode, where data is either sent to a card or received from a card. While not being transferred, **MMCIDAT** is in the HI-Z state. Data on this signals is synchronous to the rising edge of the clock period.

CRC token status

The CRC token status follows each write data block, and determines whether a card has received the data block correctly. When the token has been received, the card asserts a busy signal by driving **MMCIDAT** LOW. Table 2-5 shows the CRC token status values.

Table 2-5 CRC token status

Token	Description
010	Card has received error-free data block
101	Card has detected a CRC error

Status flags

Table 2-6 lists the data path status flags (see *Status register, MMCIStatus* on page 3-13 for more information).

Table 2-6 Data path status flags

Flag	Description
TxFifoFull	Transmit FIFO is full
TxFifoEmpty	Transmit FIFO is empty
TxFifoHalfEmpty	Transmit FIFO is half full
TxDataAvlbl	Transmit FIFO data available
TxUnderrun	Transmit FIFO underrun error
RxFifoFull	Receive FIFO is full
RxFifoEmpty	Receive FIFO is empty
RxFifoHalfFull	Receive FIFO is half full
RxDataAvlbl	Receive FIFO data available
RxOverrun	Receive FIFO overrun error
DataBlockEnd	Data block sent/received
DataCrcFail	Data packet CRC failed
DataEnd	Data end (data counter is zero)
DataTimeOut	Data timeout
TxActive	Data transmission in progress
RxActive	Data reception in progress

CRC generator

The CRC generator calculates the CRC checksum only for the data bits in a single block, and is bypassed in data stream mode. The checksum is a 16-bit value:

$CRC[15:0] = Remainder [(M(x) * x^{15}) / G(x)]$

$G(x) = x^{16} + x^{12} + x^5 + 1$

$M(x) - (first\ data\ bit) * x^n + ... + (last\ data\ bit) * x^0$

2.2.5 Data FIFO

The data FIFO (first-in-first-out) subunit is a data buffer with transmit and receive logic. Figure 2-10 shows a block diagram of the FIFO.

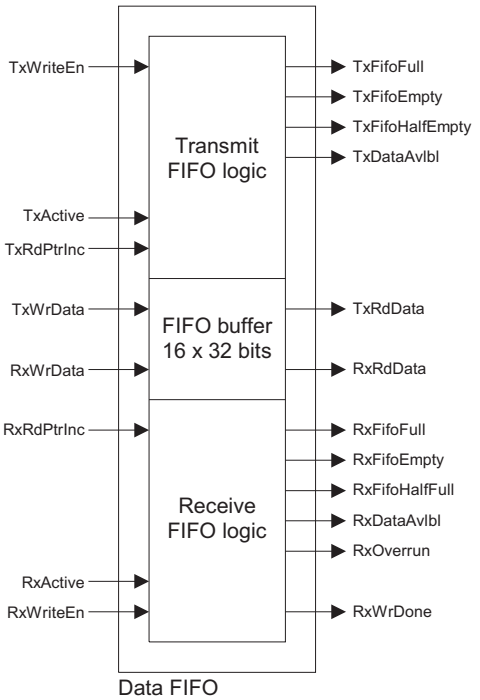


Figure 2-10 Data FIFO

The FIFO contains a 32-bit wide, 16-word deep data buffer, and transmit and receive logic. Because the data FIFO operates in the APB clock domain (**PCLK**), all signals from the subunits in the PrimeCell MMCI clock domain (**MCLK**) are resynchronized.

Depending on **TxActive** and **RxActive**, the FIFO can be disabled, transmit enabled, or receive enabled. **TxActive** and **RxActive** are driven by the data path subunit and are mutually exclusive:

- The transmit FIFO refers to the transmit logic and data buffer when **TxActive** is asserted (see *Transmit FIFO* on page 2-17)
- The receive FIFO refers to the receive logic and data buffer when **RxActive** is asserted (see *Receive FIFO* on page 2-18).

Transmit FIFO

Data is written to the transmit FIFO through the APB interface once the MMCI is enabled for transmission. When the write signal (**TxWriteEn**) is asserted, data can be written (on the rising edge of **PCLK**) into the FIFO location specified by the current value of the data pointer. The pointer is incremented after every FIFO write.

The transmit FIFO contains a data output register. This holds the data word pointed to by the read pointer. When the data path subunit has loaded its shift register, the data path logic toggles **TxRdPtrInc**. This signal is synchronized with **PCLK**, and increments the read pointer and drives new data on the **TxRdData** output.

If the transmit FIFO is disabled, all status flags are deasserted, and the read and write pointers are reset. The data path subunit asserts **TxActive** when it transmits data. Table 2-7 lists the transmit FIFO status flags.

Table 2-7 Transmit FIFO status flags

Flag	Description
TxFifoFull	Set to HIGH when all 16 transmit FIFO words contain valid data.
TxFifoEmpty	Set to HIGH when the transmit FIFO does not contain valid data.
TxHalfEmpty	Set to HIGH when 8 or more transmit FIFO words are empty. This flag can be used as a DMA request.
TxDataAvlbl	Set to HIGH when the transmit FIFO contains valid data. This flag is the inverse of the **TxFifoEmpty** flag.
TxUnderrun	Set to HIGH when an underrun error occurs. This flag is cleared by writing to the MMCIClear register.

Receive FIFO

When the data path subunit receives a word of data, it drives data on the write data bus and asserts the write enable signal. This signal is synchronized to the **PCLK** domain. The write pointer is incremented after the write is completed, and the receive FIFO control logic toggles **RxWrDone**, that then deasserts the write enable signal.

On the read side, the content of the FIFO word pointed to by the current value of the read pointer is driven on the read data bus. The read pointer is incremented when the APB bus interface asserts **RxRdPrtInc**.

If the receive FIFO is disabled, all status flags are deasserted, and the read and write pointers are reset. The data path subunit asserts **RxActive** when:

- it receives data
- data is received, but the FIFO is not empty.

Table 2-8 lists the receive FIFO status flags.

Table 2-8 Receive FIFO status flags

Flag	Description
RxFifoFull	Set to HIGH when all 16 receive FIFO words contain valid data.
RxFifoEmpty	Set to HIGH when the receive FIFO does not contain valid data.
RxHalfFull	Set to HIGH when 8 or more receive FIFO words contain valid data. This flag can be used as a DMA request.
RxDataAvlbl	Set to HIGH when the receive FIFO is not empty. This flag is the inverse of the **RxFifoEmpty** flag.
RxOverrun	Set to HIGH when an overrun error occurs. This flag is cleared by writing to the MMCIClear register.

2.3 APB interface

Figure 2-11 shows a block diagram of the APB interface.

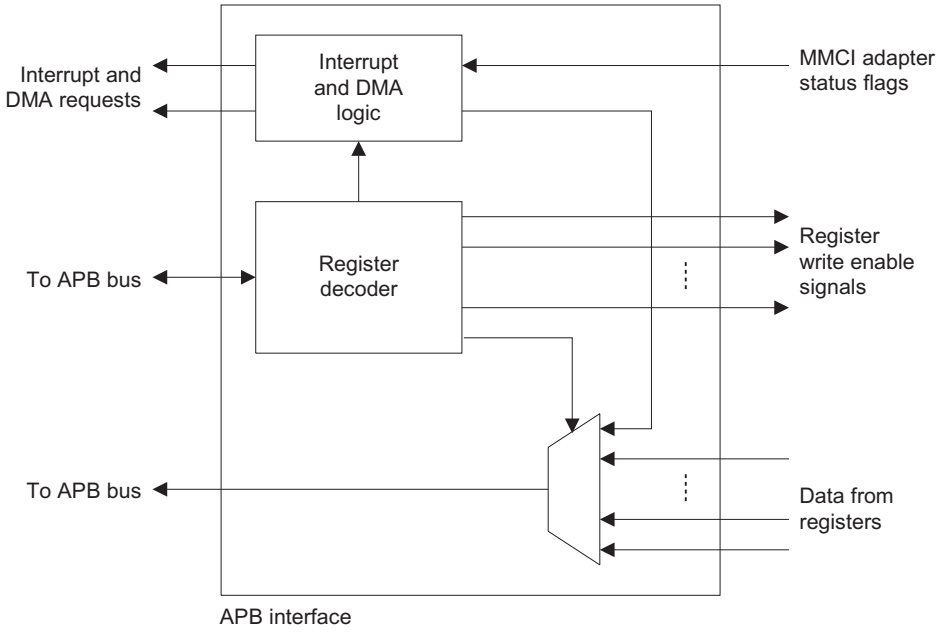


Figure 2-11 APB interface

The APB interface generates the interrupt and DMA requests, and accesses the PrimeCell MMCI adapter registers and the data FIFO. It consists of a data path, register decoder, and interrupt/DMA logic.

2.3.1 Interrupt logic

The interrupt logic (see Figure 2-12) generates two interrupt request signals, that are asserted when at least one of the selected status flags is HIGH. A status flag generates the interrupt request if a corresponding mask flag is set. You can assert the interrupt request even if **PCLK** is disabled.

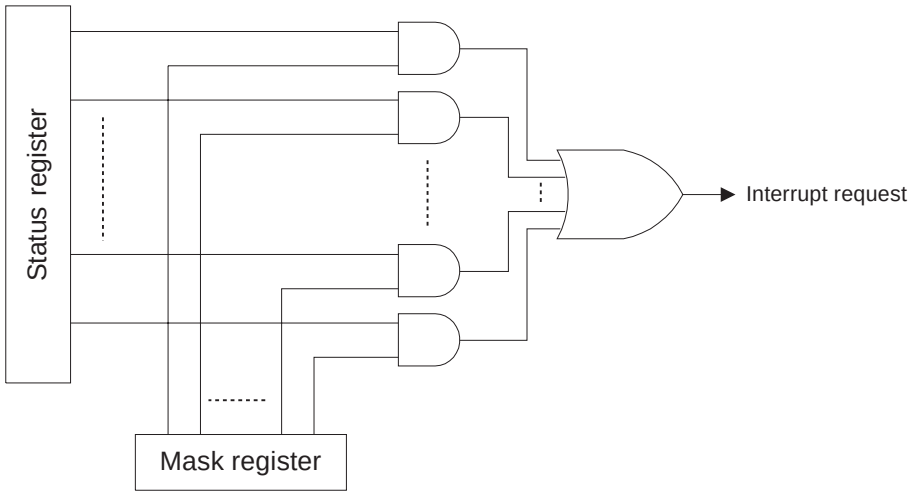


Figure 2-12 Interrupt request logic

Note

A separate mask register is provided for each interrupt request signal (see *Interrupt mask registers, MMCIMask0-1* on page 3-15 for more information).

2.3.2 DMA

The interface to the DMA controller includes the signals described in Table 2-9.

Table 2-9 DMA controller interface signals

Signal	Type	Description
DMASREQ	Single word DMA transfer request, asserted by PrimeCell MMCI	For receive: Asserted if data counter is zero and receive FIFO contains more than one and fewer than eight words. For transmit: Asserted if fewer than eight and more than one word remain for transfer to FIFO.
DMABREQ	Burst DMA transfer request, asserted by PrimeCell MMCI	For receive: Asserted if FIFO contains eight words and data counter is not zero, or if FIFO contains more than eight words. For transmit: Asserted if more than eight words remain for transfer to FIFO.
DMALSREQ	Last single word DMA transfer request, asserted by PrimeCell MMCI	For receive: Asserted if data counter is zero and FIFO contains only one word. For transmit: Asserted if only one word remains for transfer to FIFO.
DMALBREQ	Last burst DMA transfer request, asserted by PrimeCell MMCI	For receive: Asserted if data counter is zero and FIFO contains eight words. For transmit: Asserted if only eight words remain for transfer to FIFO.
DMACLR	DMA request clear, asserted by DMA controller to clear request signals	Asserted during transfer of last data in burst if DMA burst transfer is requested.

Because the four request signals are mutually exclusive, only one signal is asserted at a time. The signal remains asserted until **DMACLR** is asserted. After this, a request signal can be active again, depending on the conditions described in Table 2-9. When the Enable bit in the Data Control register is cleared, the data path is disabled and all request signals are de-asserted.

The DMA signals are synchronous with **PCLK**. Figure 2-13 shows the DMA transfer of the last three words.

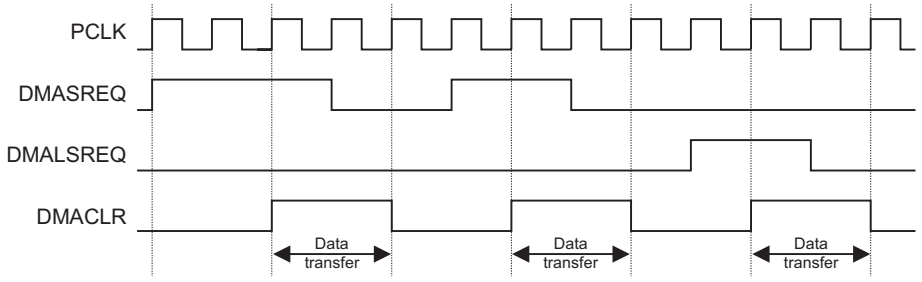


Figure 2-13 DMA interface

2.4 Timing requirements

The clock output is routed back to the PrimeCell MMCI and is used to clock the output registers, to meet the hold time requirements of **MMCICMD** and **MMCIDAT**. Figure 2-14 shows a block diagram of the clock output routing.

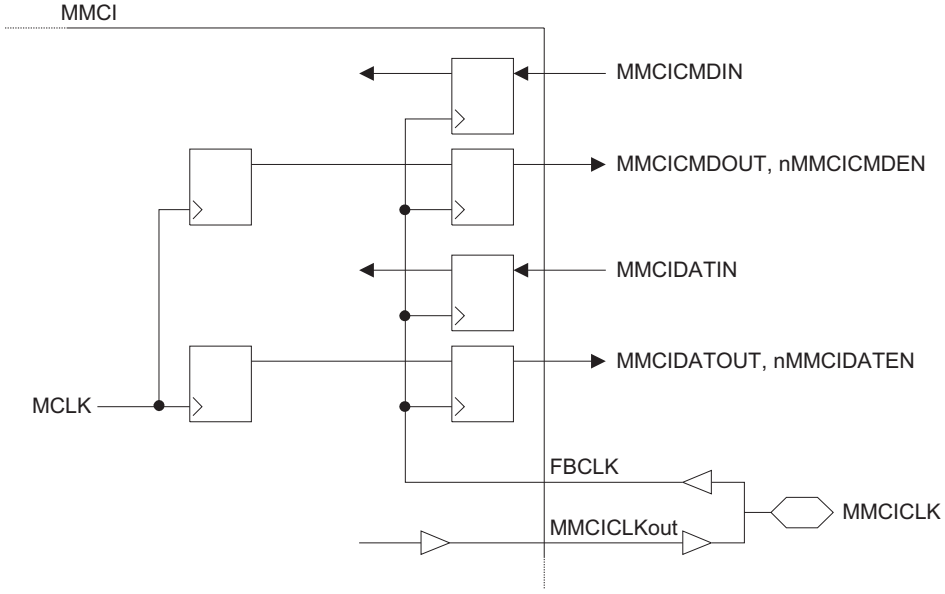


Figure 2-14 Clock output retiming logic

Ensure that you meet the input and timing requirements when integrating the PrimeCell MMCI in a system. Figure 2-15 shows the signal relationship.

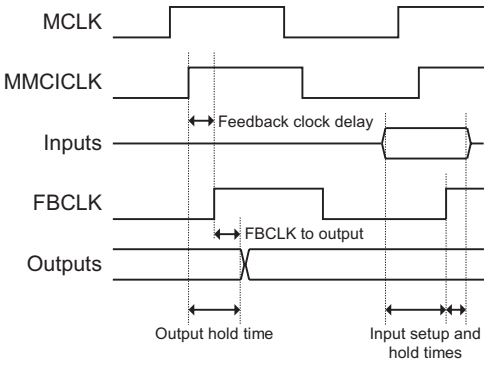


Figure 2-15 MMCICMD and MMCIDAT timing

Chapter 3
Programmer's Model

This chapter describes the ARM PrimeCell MMCI (PL181) registers, and provides details needed when programming the microcontroller. It contains the following sections:

3.1 About the programmer's model

The base address of the PrimeCell MMCI is not fixed, and can be different for each particular system implementation.

3.2 Summary of PrimeCell MMCI registers

The PrimeCell MMCI registers are shown in Table 3-1.

Table 3-1 PrimeCell MMCI register summary

Address	Type	Width	Reset value	Name	Description
MMCI Base + 0x000	Read/write	8	0x00	MMCIPower	Power control register.
MMCI Base + 0x004	Read/write	11	0x000	MMCIClock	Clock control register.
MMCI Base + 0x008	Read/write	32	0x00000000	MMCIArgument	Argument register.
MMCI Base + 0x00C	Read/write	11	0x000	MMCICommand	Command register.
MMCI Base + 0x010	Read only	6	0x00	MMCIRespCmd	Response command register.
MMCI Base + 0x014	Read only	32	0x00000000	MMCIResponse0	Response register.
MMCI Base + 0x018	Read only	32	0x00000000	MMCIResponse1	Response register.
MMCI Base + 0x01C	Read only	32	0x00000000	MMCIResponse2	Response register.
MMCI Base + 0x020	Read only	31	0x00000000	MMCIResponse3	Response register.
MMCI Base + 0x024	Read/write	32	0x00000000	MMCIDataTimer	Data timer.
MMCI Base + 0x028	Read/write	16	0x0000	MMCIDataLength	Data length register.
MMCI Base + 0x02C	Read/write	8	0x00	MMCIDataCtrl	Data control register.
MMCI Base + 0x030	Read only	16	0x0000	MMCIDataCnt	Data counter.
MMCI Base + 0x034	Read only	22	0x000000	MMCIStatus	Status register.
MMCI Base + 0x038	Write only	11	-	MMCIClear	Clear register.
MMCI Base + 0x03C	Read/write	22	0x000000	MMCIMask0	Interrupt 0 mask register.
MMCI Base + 0x040	Read/write	22	0x000000	MMCIMask1	Interrupt 1 mask register.
MMCI Base + 0x044	-	-	-	Reserved	-
MMCI Base + 0x048	Read only	15	0x0000	MMCIFifoCnt	FIFO counter.
MMCI Base + 0x04C - 0x07C	-	-	-	Reserved	-
MMCI Base + 0x080 - 0x0BC	Read/write	32	0x00000000	MMCIFIFO	Data FIFO register.

Table 3-1 PrimeCell MMCI register summary (continued)

Address	Type	Width	Reset value	Name	Description
MMCI Base + 0xFE0	Read only	8	0x81	MMCIPeriphID0	Peripheral identification register bits 7:0.
MMCI Base + 0xFE4	Read only	8	0x11	MMCIPeriphID1	Peripheral identification register bits 15:8.
MMCI Base + 0xFE8	Read only	8	0x04	MMCIPeriphID2	Peripheral identification register bits 23:16.
MMCI Base + 0xFEC	Read only	8	0x00	MMCIPeriphID3	Peripheral identification register bits 31:24.
MMCI Base + 0xFF0	Read only	8	0x0D	MMCIPCellID0	PrimeCell identification register bits 7:0.
MMCI Base + 0xFF4	Read only	8	0xF0	MMCIPCellID1	PrimeCell identification register bits 15:8.
MMCI Base + 0xFF8	Read only	8	0x05	MMCIPCellID2	PrimeCell identification register bits 23:16.
MMCI Base + 0xFFC	Read only	8	0xB1	MMCIPCellID3	PrimeCell identification register bits 31:24.

3.3 Register descriptions

The following PrimeCell MMCI registers are described in this section:

3.3.1 Power control register, MMCIPower

The MMCIPower register controls an external power supply. You can switch the power on and off, and adjust the output voltage. Table 3-2 shows the bit assignment of the MMCIPower register.

Table 3-2 MMCIPower register

Bit	Name	Type	Function
1:0	Ctrl	Read/write	00 = Power-off 01 = Reserved 10 = Power-up 11 = Power-on
5:2	Voltage	Read/write	Output voltage
6	OpenDrain	Read/write	MMCICMD output control
7	Rod	Read/write	Rod control
31:8	Reserved	-	-

When you switch the external power supply on, the software first enters the power-up phase, and waits until the supply output is stable before moving to the power-on phase. During the power-up phase, **MMCIPWR** is set HIGH. The card bus outlets are disabled during both phases.

You can set the supply output voltage using the voltage value on the **MMCIVDD** outputs. Because the operating voltage range can be any value between 2.0 and 3.6 volts, the encoding of the voltage bits in the power control register is application-specific.

Note

After a data write, data cannot be written to this register for three MCLK clock periods plus two PCLK clock periods.

3.3.2 Clock control register, MMCIClock

The MMCIClock register controls the **MMCICLK** output. Table 3-3 shows the bit assignment of the clock control register.

Table 3-3 MMCIClock register

Bit	Name	Type	Function
7:0	ClkDiv	Read/write	PrimeCell MMCI bus clock period: **MMCICLK** frequency = **MCLK** / [2x(ClkDiv+1)].
8	Enable	Read/write	Enable PrimeCell MMCI bus clock: 0 = Clock disabled 1 = Clock enabled.
9	PwrSave	Read/write	Disable PrimeCell MMCI clock output when bus is idle: 0 = Always enabled 1 = Clock enabled when bus is active.
10	Bypass	Read/write	Enable bypass of clock divide logic: 0 = Disable bypass 1 = Enable bypass (**MCLK** driven to card bus output (**MMCICLK**)).
31:11	Reserved	-	-

While the PrimeCell MMCI is in identification mode, the **MMCICLK** frequency must be less than 400 kHz. You can change the clock frequency to the maximum card bus frequency when relative card addresses are assigned to all cards.

Note

After a data write, data cannot be written to this register for three MCLK clock periods plus two PCLK clock periods.

3.3.3 Argument register, MMCIArgument

The MMCIArgument register contains a 32-bit command argument, which is sent to a card as part of a command message. Table 3-4 shows the bit assignment of the MMCIArgument register.

Table 3-4 MMCIArgument register

Bit	Name	Type	Function
31:0	CmdArg	Read/write	Command argument

If a command contains an argument, it must be loaded into the argument register before writing a command to the command register.

3.3.4 Command register, MMCICommand

The MMCICommand register contains the command index and command type bits:

- The command index is sent to a card as part of a command message
- The command type bits control the *Command Path State Machine* (CPSM). Writing 1 to the enable bit starts the command send operation, while clearing the bit disables the CPSM.

Table 3-5 shows the bit assignment of the MMCICommand register.

Table 3-5 MMCICommand register

Bit	Name	Type	Function
5:0	CmdIndex	Read/write	Command index
6	Response	Read/write	If set, CPSM waits for a response
7	LongRsp	Read/write	If set, CPSM receives a 136-bit long response
8	Interrupt	Read/write	If set, CPSM disables command timer and waits for interrupt request
9	Pending	Read/write	If set, CPSM waits for CmdPend before it starts sending a command
10	Enable	Read/write	If set, CPSM is enabled
31:11	Reserved	-	-

Note

After a data write, data cannot be written to this register for three MCLK clock periods plus two PCLK clock periods.

Table 3-6 shows the response types.

Table 3-6 Command response types

Response	LongRsp.	Description
0	0	No response, expect CmdSent flag
0	1	No response, expect CmdSent flag
1	0	Short response, expect CmdRespEnd or CmdCrcFail flag
1	1	Long response, expect CmdRespEnd or CmdCrcFail flag

3.3.5 Command response register, MMCIRespCommand

The MMCIRespCommand register contains the command index field of the last command response received. Table 3-7 shows the bit assignment of the MMCIRespCommand register.

Table 3-7 MMCIRespCommand register

Bit	Name	Type	Function
5:0	RespCmd	Read	Response command index
31:6	Reserved	-	-

If the command response transmission does not contain the command index field (long response), the RespCmd field is unknown, although it must contain 111111 (the value of the reserved field from the response).

3.3.6 Response registers, MMCIResponse0-3

The MMCIResponse0-3 registers contain the status of a card, which is part of the received response. Table 3-8 shows the bit assignment of the MMCIResponse0-3 registers.

Table 3-8 MMCIResponse0-3 registers

Bit	Name	Type	Function
31:0	Status	Read	Card status

The card status size can be 32 or 127 bits, depending on the response type (see Table 3-9).

Table 3-9 Response register type

Description	Short response	Long response
MMCIResponse0	Card status [31:0]	Card status [127:96]
MMCIResponse1	Unused	Card status [95:64]
MMCIResponse2	Unused	Card status [63:32]
MMCIResponse3	Unused	Card status [31:1]

The most significant bit of the card status is received first. The MMCIResponse3 register LSBit is always 0.

3.3.7 Data timer register, MMCIDataTimer

The MMCIDataTimer register contains the data timeout period, in card bus clock periods. Table 3-10 shows the bit assignment of the MMCIDataTimer register.

Table 3-10 MMCIDataTimer register

Bit	Name	Type	Function
31:0	DataTime	Read/write	Data timeout period

A counter loads the value from the data timer register, and starts decrementing when the *Data Path State Machine* (DPSM) enters the WAIT_R or BUSY state. If the timer reaches 0 while the DPSM is in either of these states, the timeout status flag is set.

A data transfer must be written to the data timer register and the data length register before being written to the data control register.

3.3.8 Data length register, MMCIDataLength

The MMCIDataLength register contains the number of data bytes to be transferred. The value is loaded into the data counter when data transfer starts. Table 3-11 shows the bit assignment of the MMCIDataLength register.

Table 3-11 MMCIDataLength register

Bit	Name	Type	Function
15:0	DataLength	Read/write	Data length value
31:16	Reserved	-	-

For a block data transfer, the value in the data length register must be a multiple of the block size (see *Data control register, MMCIDataCtrl*).

A data transfer must be written to the data timer register and the data length register before being written to the data control register.

3.3.9 Data control register, MMCIDataCtrl

The MMCIDataCtrl register controls the DPSM. Table 3-12 shows the bit assignment of the MMCIDataCtrl register.

Table 3-12 MMCIDataCtrl register

Bit	Name	Type	Function
0	Enable	Read/write	Data transfer enabled
1	Direction	Read/write	Data transfer direction: 0 = From controller to card 1 = From card to controller
2	Mode	Read/write	Data transfer mode: 0 = Block data transfer 1 = Stream data transfer
3	DMAEnable	Read/write	Enable DMA: 0 = DMA disabled 1 = DMA enabled.
7:4	BlockSize	Read/write	Data block length
31:8	Reserved	-	-

Note

After a data write, data cannot be written to this register for three MCLK clock periods plus two PCLK clock periods.

Data transfer starts if 1 is written to the enable bit. Depending on the direction bit, the DPSM moves to the WAIT_S or WAIT_R state. You do not need to clear the enable bit after data transfer. Table 3-13 shows the data block length if block data transfer mode is selected.

Table 3-13 Data block length

Block size	Block length
0	2^0 = 1 byte
1	2^1 = 2 bytes
...	-
11	2^{11} = 2048 bytes
12:15	Reserved

3.3.10 Data counter register, MMCIDataCnt

The MMCIDataCnt register loads the value from the data length register (see *Data length register, MMCIDataLength* on page 3-11) when the DPSM moves from the IDLE state to the WAIT_R or WAIT_S state. As data is transferred, the counter decrements the value until it reaches 0. The DPSM then moves to the IDLE state and the data status end flag is set. Table 3-14 shows the bit assignment of the MMCIDataCnt register.

Table 3-14 MMCIDataCnt register

Bit	Name	Type	Function
15:0	DataCount	Read	Remaining data
31:16	Reserved	-	-

Note

This register should be read only when the data transfer is complete.

3.3.11 Status register, MMCIStatus

The MMCIStatus register is a read-only register. It contains two types of flag:

Static [10:0] These remain asserted until they are cleared by writing to the Clear register (see *Clear register, MMCIClear* on page 3-14).

Dynamic [21:11] These change state depending on the state of the underlying logic (for example, FIFO full and empty flags are asserted and deasserted as data while written to the FIFO).

Table 3-15 shows the bit assignment of the MMCIStatus register.

Table 3-15 MMCIStatus register

Bit	Name	Type	Function
0	CmdCrcFail	Read	Command response received (CRC check failed)
1	DataCrcFail	Read	Data block sent/received (CRC check failed)
2	CmdTimeOut	Read	Command response timeout
3	DataTimeOut	Read	Data timeout
4	TxUnderrun	Read	Transmit FIFO underrun error
5	RxOverrun	Read	Receive FIFO overrun error
6	CmdRespEnd	Read	Command response received (CRC check passed)
7	CmdSent	Read	Command sent (no response required)
8	DataEnd	Read	Data end (data counter is zero)
9	Reserved	-	-
10	DataBlockEnd	Read	Data block sent/received (CRC check passed)
11	CmdActive	Read	Command transfer in progress
12	TxActive	Read	Data transmit in progress
13	RxActive	Read	Data receive in progress
14	TxFifoHalfEmpty	Read	Transmit FIFO half empty

Table 3-15 MMCIStatus register (continued)

Bit	Name	Type	Function
15	RxFifoHalfFull	Read	Receive FIFO half full
16	TxFifoFull	Read	Transmit FIFO full
17	RxFifoFull	Read	Receive FIFO full
18	TxFifoEmpty	Read	Transmit FIFO empty
19	RxFifoEmpty	Read	Receive FIFO empty
20	TxDataAvlbl	Read	Data available in transmit FIFO
21	RxDataAvlbl	Read	Data available in receive FIFO
31:22	Reserved	-	-

3.3.12 Clear register, MMCIClear

The MMCIClear register is a write-only register. The corresponding static status flags can be cleared by writing a 1 to the corresponding bit in the register. Table 3-16 shows the bit assignment of the MMCIClear register.

Table 3-16 MMCIClear register

Bit	Name	Type	Function
0	CmdCrcFailClr	Write	Clears CmdCrcFail flag
1	DataCrcFailClr	Write	Clears DataCrcFail flag
2	CmdTimeOutClr	Write	Clears CmdTimeOut flag
3	DataTimeOutClr	Write	Clears DataTimeOut flag
4	TxUnderrunClr	Write	Clears TxUnderrun flag
5	RxOverrunClr	Write	Clears RxOverrun flag
6	CmdRespEndClr	Write	Clears CmdRespEnd flag
7	CmdSentClr	Write	Clears CmdSent flag
8	DataEndClr	Write	Clears DataEnd flag
9	Reserved	-	-
10	DataBlockEndClr	Write	Clears DataBlockEnd flag
31:11	Reserved	-	-

3.3.13 Interrupt mask registers, MMCIMask0-1

There are two interrupt mask registers, MMCIMask0-1, one for each interrupt request signal. Table 3-17 shows the bit assignment of the MMCIMask0-1 registers.

Table 3-17 MMCIMask0-1 registers

Bit	Name	Type	Function
0	Mask0	Read/write	Mask CmdCrcFail flag
1	Mask1	Read/write	Mask DataCrcFail flag
2	Mask2	Read/write	Mask CmdTimeOut flag
3	Mask3	Read/write	Mask DataTimeOut flag
4	Mask4	Read/write	Mask TxUnderrun flag
5	Mask5	Read/write	Mask RxOverrun flag
6	Mask6	Read/write	Mask CmdRespEnd flag
7	Mask7	Read/write	Mask CmdSent flag
8	Mask8	Read/write	Mask DataEnd flag
9	Reserved	-	-
10	Mask10	Read/write	Mask DataBlockEnd flag
11	Mask11	Read/write	Mask CmdActive flag
12	Mask12	Read/write	Mask TxActive flag
13	Mask13	Read/write	Mask RxActive flag
14	Mask14	Read/write	Mask TxFifoHalfEmpty flag
15	Mask15	Read/write	Mask RxFifoHalfFull flag
16	Mask16	Read/write	Mask TxFifoFull flag
17	Mask17	Read/write	Mask RxFifoFull flag
18	Mask18	Read/write	Mask TxFifoEmpty flag
19	Mask19	Read/write	Mask RxFifoEmpty flag
20	Mask20	Read/write	Mask TxDataAvlbl flag
21	Mask21	Read/write	Mask RxDataAvlbl flag
31:22	Reserved	-	-

The interrupt mask registers determine which status flags generate an interrupt request by setting the corresponding bit to 1.

3.3.14 FIFO counter register, MMCIFifoCnt

The MMCIFifoCnt register contains the remaining number of words to be written to or read from the FIFO. The FIFO counter loads the value from the data length register (see *Data length register, MMCIDataLength* on page 3-11) when the Enable bit is set in the data control register. If the data length is not word aligned (multiple of 4), the remaining 1 to 3 bytes are regarded as a word. Table 3-18 shows the bit assignment of the MMCIFifoCnt register.

Table 3-18 MMCIFifoCnt register

Bit	Name	Type	Function
14:0	DataCount	Read	Remaining data
31:15	Reserved	-	-

3.3.15 Data FIFO register, MMCIFIFO

The receive and transmit FIFOs can be read or written as 32-bit wide registers. The FIFOs contain 16 entries on 16 sequential addresses. This allows the microprocessor to use its load and store multiple operands to read/write to the FIFO. Table 3-19 shows the bit assignment of the MMCIFIFO register.

Table 3-19 MMCIFIFO register

Bit	Name	Type	Function
31:0	Data	Read/write	FIFO data

3.3.16 Peripheral identification registers, MMCIPeriphID0-3

The MMCIPeriphID0-3 registers are four 8-bit registers, that span address locations `0xFE0-0xFEC`. The registers can conceptually be treated as a single 32-bit register. The read-only registers provide the following options of the peripheral:

Part number [11:0] This is used to identify the peripheral. The three digit product code 181 is used for the PrimeCell MMCI.

Designer [19:12] This is the identification of the designer. ARM Ltd. is `0x41` (ASCII A).

Revision number [23:20]
This is the revision number of the peripheral. The revision number starts from 0.

Configuration [31:24]
This is the configuration option of the peripheral. The configuration value is 0.

Figure 3-1 shows the bit assignment for the MMCIPeriphID0-3 registers.

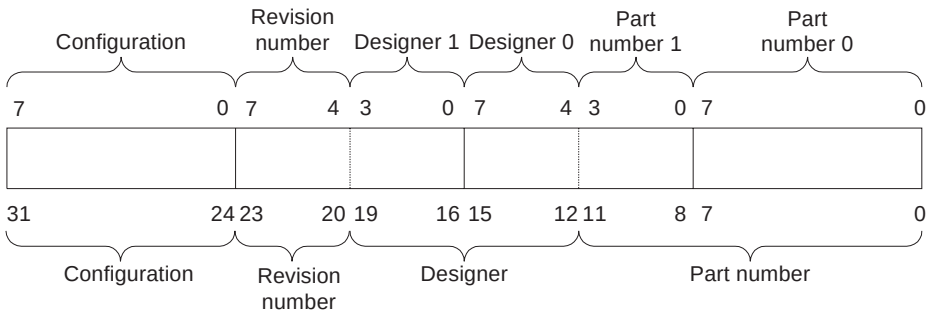


Figure 3-1 Peripheral identification register bit assignment

Note

When you design a systems memory map you must remember that the register has a 4KB-memory footprint.

The 4-bit revision number is implemented by instantiating a component called *RevisionAnd* four times with its inputs tied off as appropriate, and the output sent to the read multiplexor.

All memory accesses to the peripheral identification registers must be 32-bit, using the LDR and STR instructions.

The four, 8-bit peripheral identification registers are described in the following subsections:

- *MMCIPeriphID0 register*
- *MMCIPeriphID1 register* on page 3-19
- *MMCIPeriphID2 register* on page 3-19
- *MMCIPeriphID3 register* on page 3-19.

MMCIPeriphID0 register

The MMCIPeriphID0 register is hard coded and the fields within the registers determine the reset value. Table 3-20 shows the bit assignment of the MMCIPeriphID0 register.

Table 3-20 MMCIPeriphID0 register

Bits	Name	Type	Function
31:8	-	-	Reserved, read undefined, must be written as zeros.
7:0	Partnumber0	Read	These bits read back as `0x81`.

MMCIPeriphID1 register

The MMCIPeriphID1 register is hard coded and the fields within the registers determine the reset value. Table 3-21 shows the bit assignment of the MMCIPeriphID1 register.

Table 3-21 MMCIPeriphID1 register

Bits	Name	Type	Function
31:8	-	-	Reserved, read undefined, must be written as zeros.
7:4	Designer0	Read	These bits read back as `0x1`.
3:0	Partnumber1	Read	These bits read back as `0x1`.

MMCIPeriphID2 register

The MMCIPeriphID2 register is hard coded and the fields within the registers determine the reset value. Table 3-22 shows the bit assignment of the MMCIPeriphID2 register.

Table 3-22 MMCIPeriphID2 register

Bits	Name	Type	Function
31:8	-	-	Reserved, read undefined, must be written as zeros.
7:4	Revision	Read	These bits read back as `0x0`.
3:0	Designer1	Read	These bits read back as `0x4`.

MMCIPeriphID3 register

The MMCIPeriphID3 register is hard coded and the fields within the registers determine the reset value. Table 3-23 shows the bit assignment of the MMCIPeriphID3 register.

Table 3-23 MMCIPeriphID3 register

Bits	Name	Type	Function
31:8	-	-	Reserved, read undefined, must be written as zeros.
7:0	Configuration	Read	These bits read back as `0x0`.

3.3.17 PrimeCell identification registers, MMCIPCellID0-3

The MMCIPCellID0-3 registers are four 8-bit registers, that span address locations `0xFF0-0xFFC`. The read-only registers can conceptually be treated as a single 32-bit register. The register is used as a standard cross-peripheral identification system. Figure 3-2 shows the bit assignment for the MMCIPCellID0-3 registers.

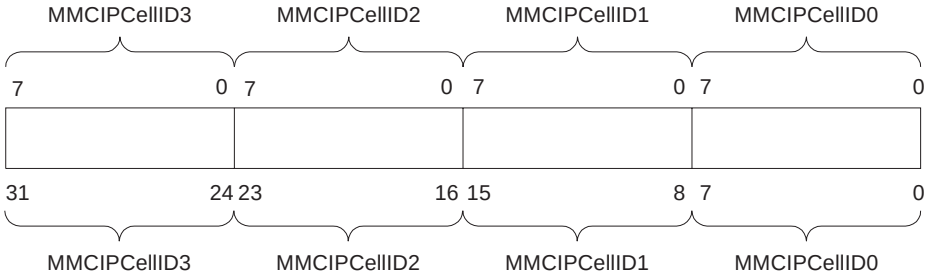


Figure 3-2 PrimeCell identification register bit assignment

The four, 8-bit registers are described in the following subsections:

- *MMCIPCellID0 register*
- *MMCIPCellID1 register* on page 3-21
- *MMCIPCellID2 register* on page 3-21
- *MMCIPCellID3 register* on page 3-21.

MMCIPCellID0 register

The MMCIPCellID0 register is hard coded and the fields within the registers determine the reset value. Table 3-24 shows the bit assignment of the MMCIPCellID0 register.

Table 3-24 MMCIPCellID0 register

Bits	Name	Type	Function
31:8	-	-	Reserved, read undefined, must be written as zeros.
7:0	MMCIPCellID0	Read	These bits read back as `0x0D`.

MMCIPCellID1 register

The MMCIPCellID1 register is hard coded and the fields within the registers determine the reset value. Table 3-25 shows the bit assignment of the MMCIPCellID1 register.

Table 3-25 MMCIPCellID1 register

Bits	Name	Type	Function
31:8	-	-	Reserved, read undefined, must be written as zeros.
7:0	MMCIPCellID1	Read	These bits read back as `0xF0`.

MMCIPCellID2 register

The MMCIPCellID2 register is hard coded and the fields within the registers determine the reset value. Table 3-26 shows the bit assignment of the MMCIPCellID2 register.

Table 3-26 MMCIPCellID2 register

Bits	Name	Type	Function
31:8	-	-	Reserved, read undefined, must be written as zeros.
7:0	MMCIPCellID2	Read	These bits read back as `0x05`.

MMCIPCellID3 register

The MMCIPCellID3 register is hard coded and the fields within the registers determine the reset value. Table 3-27 shows the bit assignment of the MMCIPCellID3 register.

Table 3-27 MMCIPCellID3 register

Bits	Name	Type	Function
31:8	-	-	Reserved, read undefined, must be written as zeros.
7:0	MMCIPCellID3	Read	These bits read back as `0xB1`.

Chapter 4
Programmer's Model for Test

This chapter describes the additional logic for functional verification and provisions made for production testing. It contains the following sections:

4.1 PrimeCell MMCI test harness overview

The additional logic for functional verification and integration vectors allows:

- capture of input signals to the block
- stimulation of the output signals.

The integration vectors provide a way of verifying that the PrimeCell MMCI is correctly wired into a system. This is done by separately testing three groups of signals:

AMBA signals These are tested by checking the connections of all the address and data bits.

Primary input/output signals

These are tested using a simple trickbox and dummy pad block that can demonstrate the correct connection of the input/output signals to external pads. Figure 4-1 on page 4-3 shows the test harness connectivity.

Intra-chip signals (such as interrupt sources)

The tests for these signals are system-specific, and enable you to write the necessary tests. Additional logic is implemented allowing you to read and write to each intra-chip input/output signal.

These test features are controlled by test registers. This allows you to test the PrimeCell MMCI in isolation from the rest of the system using only transfers from the AMBA APB.

Off-chip test vectors are supplied using a 32-bit parallel *External Bus Interface* (EBI) and converted to internal AMBA bus transfers. The application of test vectors is controlled through the *Test Interface Controller* (TIC) AMBA bus master module.

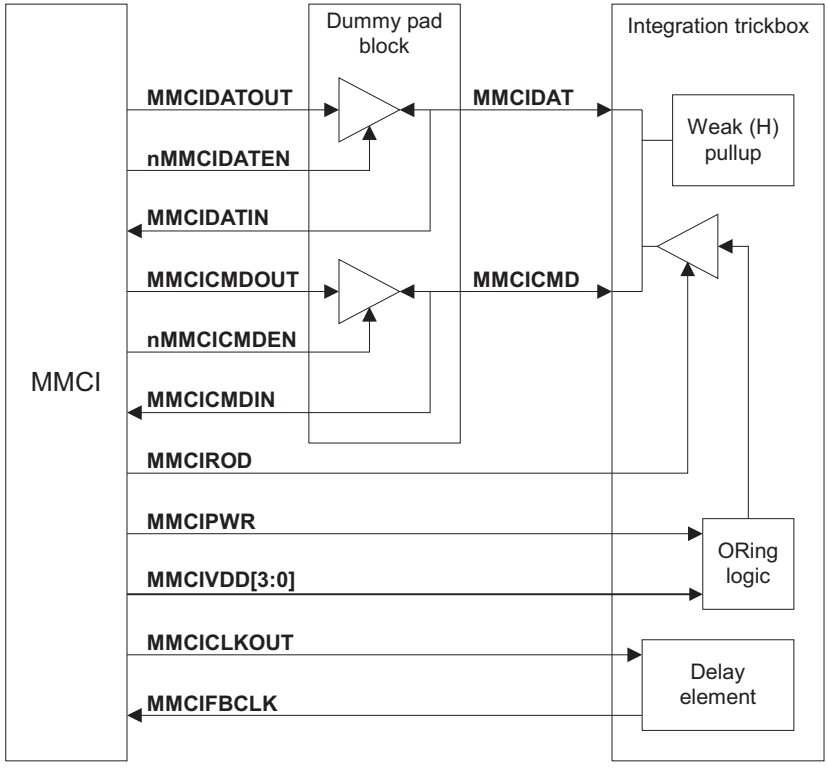


Figure 4-1 Primary input/output signal test harness

4.2 Scan testing

The PrimeCell MMCI has been designed to simplify:

- insertion of scan test cells
- use of *Automatic Test Pattern Generation* (ATPG).

This is the recommended method of manufacturing test.

4.3 Test registers

The PrimeCell MMCI test registers are memory-mapped as shown in Table 4-1.

Table 4-1 Test registers memory map

Address	Type	Width	Reset value	Name	Description
MMCI Base + `0x100`	Read/write	6	`110000`	MMCITCR	Test control register
MMCI Base + `0x104`	Read/write	6	`000000`	MMCIITIP	Integration test input register
MMCI Base + `0x108`	Read/write	12	`0x000`	MMCIITOP	Integration test output register

Each register shown in Table 4-1 is described below.

4.3.1 Test control register, MMCITCR

MMCITCR is a six-bit test control register. The ITEN bit in this register forces the multiplexers on the inputs and outputs to use the test values. Table 4-2 shows the bit assignment of the MMCITCR register.

Table 4-2 MMCITCR register

Bit	Name	Type	Function
31:6	Reserved	-	-
5	nMMCIDATEN	Read/write	**nMMCIDATEN** signal control bit.
4	nMMCICMDEN	Read/write	**nMMCICMDEN** signal control bit.
3	REGTEST	Read/write	Register Test Bit: 0 = default, normal mode. Accesses to the registers are controlled by the hardware protection circuitry. 1 = test mode. The hardware protection circuitry is bypassed. Normal Write/Read/Write/Read tests can be performed independently of the link side.

Table 4-2 MMCITCR register (continued)

Bit	Name	Type	Function
2:1	FIFOTEST	Read/write	FIFO Test: 00 = default, normal mode. Reads to the DataRegister return data from the read port of the FIFO if the direction control bit is configured for receive. If the direction control bit is configured for transmit, this read has no effect. Writes to the DataRegister write data into the write port of the FIFO if the direction control bit is configured for transmit. If the direction control bit is configured for receive, this write has no effect. 01 = test mode. Reads to the DataRegister return data from the read port of the FIFO irrespective of the setting of the direction bit, or whether the PrimeCell MMCI is data enabled. Writes to the DataRegister write data into the write Port of the FIFO irrespective of the setting of the direction bit, or whether the PrimeCell MMCI is data enabled. 10 = reserved. 11 = test mode. Reads to the DataRegister return data from the read port of the FIFO irrespective of the setting of the direction bit, or whether the PrimeCell MMCI is data enabled. Additionally, this read access automatically generates a Write Access to the write port of the FIFO (the write data pattern is pre-defined in this case). Writes to the DataRegister write data into the write port of the FIFO irrespective of the setting of the direction bit, or whether the PrimeCell MMCI is data enabled (the test code will not perform writes when FIFOTEST = 11).
0	ITEN	Read/write	Integration Test Enable: 0 = PrimeCell MMCI placed in normal mode. 1 = PrimeCell MMCI placed in Integration test mode.

4.3.2 Integration test input read/set register, MMCIITIP

MMCIITIP is 3-bit register. Out of these three bits, only the MMCIDMACLR bit is writable. The other bits are read-only.

The ITEN bit is the control bit for the multiplexor. This is used in the read path of the **MMCIDMACLR** intra-chip input. If the ITEN control bit is de-asserted, the **MMCIDMACLR** intra-chip input is routed as the internal **MMCIDMACLR** input. If not, the stored register value is driven on the internal line. All other read-only bits in the MMCIITIP register are directly connected to the primary input pins.

Table 4-3 shows the bit assignment of the MMCIITIP register.

Table 4-3 MMCIITIP register

Bit	Name	Type	Function
31:6	Reserved	-	-
5	MMCICMDIN	Read-only	Reads return the value on the MMCICMDIN primary input.
4:2	Reserved	-	-
1	MMCIDATIN	Read-only	Reads return the value on the MMCIDATIN primary inputs.
0	MMCIDMACLR	Read/write	Writes specify the value to be driven on the intra-chip MMCIDMACLR input in the Integration Test Mode. Reads return the value on MMCIDMACLR at the output of the test multiplexor.

4.3.3 Integration test output read/set register, MMCIITOP

MMCIITOP is a 9-bit register. MMCIITOP[5:0] are connected to intra-chip outputs and MMCIITOP[11:10] and MMCITOP[6] are connected to the primary outputs. The read path is different for the intra-chip and the primary output pins. If the ITEN bit is set for intra-chip output, MMCIITOP[5:0] is connected with the actual outputs. If not, these pins are connected with the stored register values.

Table 4-4 shows the bit assignment for the MMCIITOP register.

Table 4-4 MMCIITOP register

Bit	Name	Type	Function
31:12	Reserved	-	-
11	MMCIPOWER	Read/write	Primary output. Writes specify the value to be driven on the MMCIPOWER primary output in the integration test mode. Reads return the value written into this field.
10	MMCICMDOUT	Read/write	Primary output. Writes specify the value to be driven on the MMCICMDOUT primary output in the integration test mode. Reads return the value written into this field.
9:7	Reserved	-	-
6	MMCIDATOUT	Read/write	Primary output. Writes specify the value to be driven on the MMCIDATOUT primary output in the integration test mode. Reads return the value written into this field.
5	MMCIDMALBREQ	Read/write	Intra-chip output. Writes specify the value to be driven on the intra-chip MMCIDMALBREQ output in the integration test mode. Reads return the value on MMCIDMALBREQ at the output of the test multiplexor.
4	MMCIDMALSREQ	Read/write	Intra-chip output. Writes specify the value to be driven on the intra-chip MMCIDMALSREQ output in the integration test mode. Reads return the value on MMCIDMALSREQ at the output of the test multiplexor.

Table 4-4 MMCIITOP register (continued)

Bit	Name	Type	Function
3	MMCIDMABREQ	Read/write	Intra-chip output. Writes specify the value to be driven on the intra-chip MMCIDMABREQ output in the integration test mode. Reads return the value on MMCIDMABREQ at the output of the test multiplexor.
2	MMCIDMASREQ	Read/write	Intra-chip output. Writes specify the value to be driven on the intra-chip MMCIDMASREQ output in the integration test mode. Reads return the value on MMCIDMASREQ at the output of the test multiplexor.
1	MMCIINTR1	Read/write	Intra-chip output. Writes specify the value to be driven on the intra-chip MMCIINTR1 output in the integration test mode. Reads return the value on MMCIINTR1 at the output of the test multiplexor.
0	MMCIINTR0	Read/write	Intra-chip output. Writes specify the value to be driven on the intra-chip MMCIINTR0 output in the integration test mode. Reads return the value on MMCIINTR0 at the output of the test multiplexor.

Note

MMCIINTR1 and MMCIINTR0 are asynchronous.

4.4 Integration testing of block inputs

The following sections describe the integration testing for the block inputs:

- *Intra-chip inputs*
- *Primary inputs*.

4.4.1 Intra-chip inputs

When you run integration tests with the PrimeCell MMCI in a standalone test setup:

- Write a 1 to the ITEN bit in the control register. This selects the test path from the MMCIITIP[0] register bit to the internal **MMCIDMACLR** signal.
- Write a 1 and then a 0 to the MMCIITIP[0] register bit and read the same register bit to ensure that the value written is read out.

When you run integration tests with the PrimeCell MMCI as part of an integrated system:

- Write a 0 to the ITEN bit in the control register. This selects the normal path from the external MMCIDMACLR pin to the internal **MMCIDMACLR** signal.
- Write a 1 and then a 0 to the internal test registers of the DMA controller to toggle the MMCIDMACLR signal connection between the DMA controller and the PrimeCell MMCI. Read from the MMCIITIP[0] register bit to verify that the value written into the DMA controller is read out through the PrimeCell MMCI.

Figure 4-2 on page 4-11 shows details of the implementation of the input integration test harness.

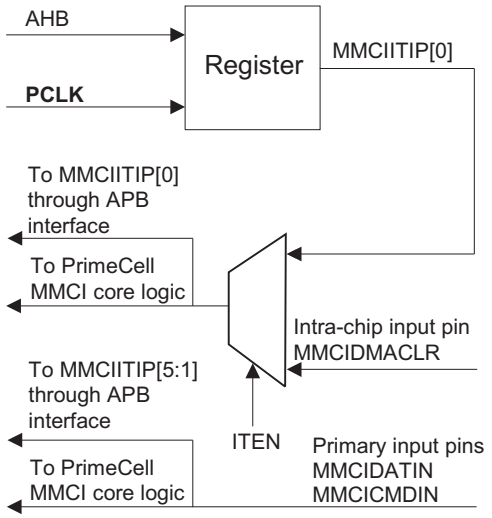


Figure 4-2 Input integration test harness

4.4.2 Primary inputs

Use this test for the following inputs:

- **MMCIDATIN**
- **MMCICMDIN**
- **MMCIFBCLK**.

Test **MMCIDATIN** using the integration vector trickbox and dummy pad block by looping back the **MMCICMDOUT** line. The exact test sequence to verify connectivity of **MMCIDATIN** is detailed in *Primary outputs* on page 4-13.

The sequence for testing **MMCICMDIN** is detailed in *Primary outputs* on page 4-13.

Test **MMCIFBCLK** using the integration vector trickbox by looping back the **MMCICLKOUT** line.

4.5 Integration testing of block outputs

The following sections describe the integration testing for the block outputs:

- *Intra-chip outputs*
- *Primary outputs* on page 4-13.

4.5.1 Intra-chip outputs

Use this test for the following outputs:

- **MMCIINTR0**
- **MMCIINTR1**
- **MMCIDMASREQ**
- **MMCIDMABREQ**
- **MMCIDMALSREQ**
- **MMCIDMALBREQ**.

When you run integration tests with the PrimeCell MMCI in a standalone test setup:

- Write a 1 to the ITEN bit in the control register. This selects the test path from the MMCIITOP[5:0] register bits to the intra-chip output signals.
- Write a 1 and then a 0 to the MMCIITOP[5:0] register bits and read the same register bits to ensure that the value written is read out.

When you run integration tests with the PrimeCell MMCI as part of an integrated system:

- Write a 1 to the ITEN bit in the control register. This selects the test path from the MMCIITOP[5:0] register bits to the intra-chip output signals.
- Write a 1 and then a 0 to the MMCIITOP[5:0] register bits to toggle the signal connections between the DMA controller/interrupt controller and the PrimeCell MMCI. Read from the internal test registers of the DMA controller/interrupt controller to ensure that the value written into the MMCIITOP[5:0] register bits is read out through the PrimeCell MMCI.

Figure 4-3 shows details of the implementation of the output integration test harness in the case of intra-chip outputs.

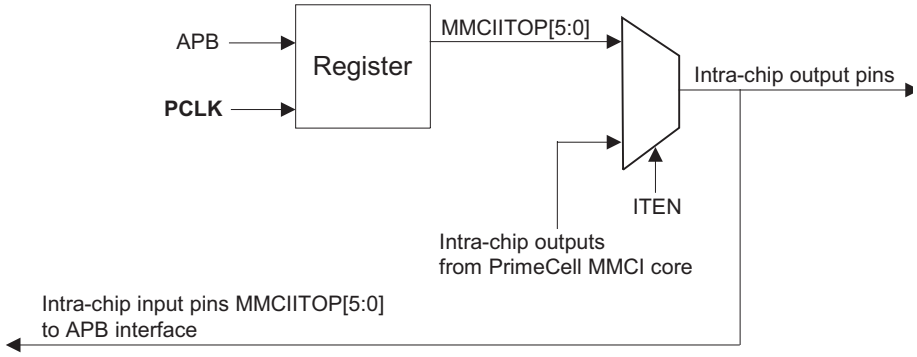


Figure 4-3 Output integration test harness, intra-chip outputs

4.5.2 Primary outputs

Integration testing of primary outputs and primary inputs is carried out using the integration vector trickbox and dummy pad block.

Verify the **MMCICLKOUT**, **MMCIFBCLK**, **MMCICMDOUT**, and **MMCIDATIN** primary input/output pin connections as follows:

- Primary output **MMCICLKOUT** and primary input **MMCIFBCLK** are connected inside the integration vector trickbox with a delay element between the lines.
- In the test register, set:
  - **nMMCICMDEN** to 0
  - **nMMCIDATEN** to 1
  - **MMCIROD** to 0 (active HIGH).
- This routes primary output **MMCICMDOUT** through the primary input pin **MMCIDATIN**.
- Clear the MMCITCR register to switch to Normal mode. Set bits [1:0] in the MMCIPower register to 11, to switch to Power On mode.
- Enable the **MCICLK** through the MMCIClock register.
- Load the command to be transmitted into the MMCIArgument register.

- Set the MCIDataLength register to receive 6 bytes.
- Set the MMCIDataTimer register to the maximum timeout length.
- Enable the DPSM for a stream data transfer from card to controller.
- Load an appropriate Command Index into the MMCICommand register, ensuring that the Response bit is cleared.
- Enable the CPSM transmit.

Note

To receive a 32-bit word, the DPSM receives a frame of 48 bits, the bytes of which must be reordered to retrieve the transmitted command:

- Bytes[4:1] are contained within the first FIFO register. Bytes[6:5] are the last two bytes [right justified] of the second FIFO register.
- The first bit transmitted by the CPSM is the start bit. This bit is not taken as a data bit by the DPSM, is not shifted into the Rx shift register, and is not written into the FIFO.
- The DPSM interprets received data as bytes. Because of this, after one word of data is received, a swapping of bytes occurs for data write into the FIFO. For example:
  - the first byte received, which is in [31:24] of the shift register, is written into [7:0] of the FIFO
  - the second byte received, which is in [23:16] of the shift register, is written into [15:8] of the FIFO.
- Bits[40:9] of the received frame contain the received command argument.

The first FIFO word contains the argument value in the following order:

- [31:24] contains Argument[14:7]
- [23:16] contains Argument[22:15]
- [15:8] contains Argument[30:23]
- [7:0] contains DontCare[6:0], and Argument[31].

The second FIFO word contains the argument value in the following order:

- [7:0] contains DontCare[6:0], and Argument[0].

- If the received argument is the same as the transmitted argument, the connectivity of **MMCICLKOUT**, **MMCIFBCLK**, **MMCICMDOUT**, and **MMCIDATIN** is proven.

Verify the **MMCIVDD**, **MMCIROD**, **MMCIPWR**, **MMCIDATOUT** and **MMCICMDIN** (or **MMCIDATIN**) pin connections as follows:

- Test **MMCICMDIN** by looping back **MMCIDATOUT**.
- Primary output **MMCIDATOUT** and primary input **MMCICMDIN** are connected by writing 1 to the ITEN bit and 0 to the nMMCIDATEN bit of the MMCITCR register. Tristate the **MMCICMDOUT** buffer by writing 1 to the nMMCICMDEN bit of the MMCITCR register. Write 0 to the MMCIROD bit in the MMCIPower register to tristate the **MMCIORMUX** buffer. Write 1 and then 0 to the MMCIDATOUT bit in the MMCIITOP register, and read the value of the MMCICMDIN bit in the MMCIITIP register.
- Primary output pins **MMCIPWR** and **MMCIVDD[3:0]** are ORed together and routed back to the primary input pins **MMCICMDIN** (or **MMCIDATIN**) through the integration vector trickbox, using **MMCIROD** as an enable signal. Disable the **MMCIDATOUT** and **MMCICMDOUT** buffers by writing 1 to the nMMCIDATEN and nMMCICMDEN bits of the MMCITCR register.
- You can strobe primary outputs **MMCIVDD[3:0]** and **MMCIROD** through the MMCIPower register, and access all other primary outputs through the MMCIITOP register. Different data patterns are written to the output pins using the MMCIPower and MMCIITOP registers. Read back the ORed data from the **MMCICMDIN** pin (or **MMCIDATIN** pin) through the MMCIITIP register.

Figure 4-4 shows details of the implementation of the output integration test harness in the case of primary outputs.

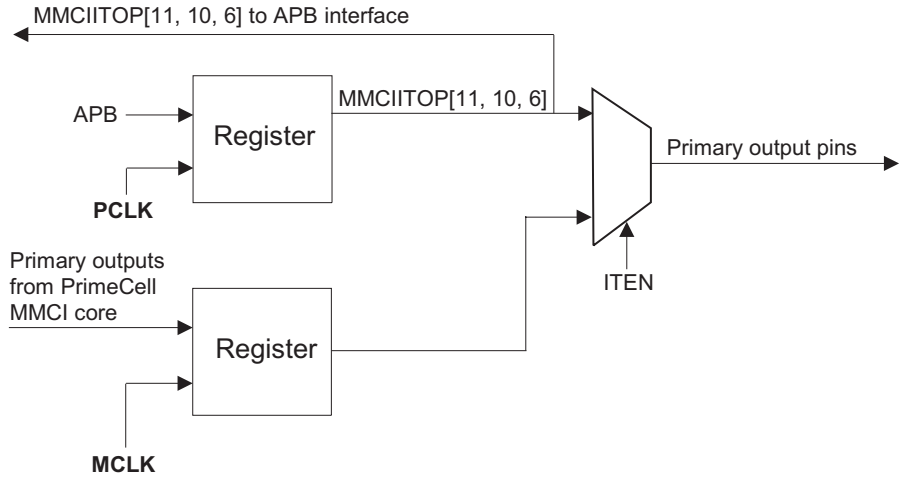


Figure 4-4 Output integration test harness, primary outputs

4.6 Integration test summary

Table 4-5 summarizes the integration test strategy for all PrimeCell MMCI pins.

Table 4-5 PrimeCell MMCI integration test strategy

Name	Type	Source/ destination	Test strategy
PCLK	In	APB	Register read/write.
PRESETn	In	APB	Register read/write.
PADDR[11:2]	In	APB	Register read/write.
PSEL	In	APB	Register read/write.
PENABLE	In	APB	Register read/write.
PWRITE	In	APB	Register read/write.
PWDATA[31:0]	In	APB	Register read/write.
PRDATA[31:0]	Out	APB	Register read/write.
MMCIINTR0	Out	Intra-chip	Use MMCIITOP register.
MMCIINTR1	Out	Intra-chip	Use MMCIITOP register.
MMCIDMASREQ	Out	Intra-chip	Use MMCIITOP register.
MMCIDMABREQ	Out	Intra-chip	Use MMCIITOP register.
MMCIDMALSREQ	Out	Intra-chip	Use MMCIITOP register.
MMCIDMALBREQ	Out	Intra-chip	Use MMCIITOP register.
MMCIDMACLR	In	Intra-chip	Use MMCIITIP register.
MCLK	In	Primary	Indirectly tested by performing a command-to-data looped back data transfer.
nMCLK	In	Primary	Not tested using integration test vectors.
nMMCIRST	In	Primary	Indirectly tested by performing a command-to-data looped back data transfer.
MMCICLKOUT	Out	Primary	Use integration vector trickbox and perform a command-to-data looped back data transfer.
MMCIFBCLK	In	Primary	Use integration vector trickbox and perform a command-to-data looped back data transfer.
MMCICMDIN	In	Primary	Use integration vector trickbox, and MMCIITIP, MMCIITOP, and MMCIPower registers.

Table 4-5 PrimeCell MMCI integration test strategy (continued)

Name	Type	Source/ destination	Test strategy
MMCICMDOUT	Out	Primary	Use integration vector trickbox and perform a command-to-data looped back data transfer.
nMMCICMDEN	Out	Primary	Indirectly tested during a data transfer through MMCICMD pad.
MMCIDATIN	In	Primary	Use integration vector trickbox and perform a command-to-data looped back data transfer.
MMCIDATOUT	Out	Primary	Use integration vector trickbox, and MMCIITIP, MMCIITOP, and MMCIPower registers.
nMMCIDATEN	Out	Primary	Indirectly tested during a data transfer through MMCIDAT pad.
MMCIPWR	Out	Primary	Use integration vector trickbox, and MMCIITIP, MMCIITOP, and MMCIPower registers.
MMCIVDD[3:0]	Out	Primary	Use integration vector trickbox, and MMCIITIP, MMCIITOP, and MMCIPower registers.
MMCIROD	Out	Primary	Use integration vector trickbox, and MMCIITIP, MMCIITOP, and MMCIPower registers.
SCANENABLE	In	Test controller	Not tested using integration test vectors.
SCANINPCLK	In	Test controller	Not tested using integration test vectors.
SCANINMCLK	In	Test controller	Not tested using integration test vectors.
SCANINMMCIFBCLK	In	Test controller	Not tested using integration test vectors.
SCANOUTPCLK	Out	Test controller	Not tested using integration test vectors.
SCANOUTMCLK	Out	Test controller	Not tested using integration test vectors.
SCANOUTMMCIFBCLK	Out	Test controller	Not tested using integration test vectors.

Appendix A
ARM PrimeCell MMCI (PL181) Signal Descriptions

This appendix describes the signals that interface with the ARM PrimeCell *MultiMedia Card Interface* (MMCI). It contains the following sections:

A.1 AMBA APB signals

The PrimeCell MMCI module is connected to the AMBA APB as a bus slave. With the exception of the **BnRES** signal, the AMBA APB signals have a **P** prefix and are active HIGH. Active LOW signals contain a lower case **n**. The AMBA APB signals are described in Table A-1.

Table A-1 AMBA APB signals

Name	Type	Source/ destination	Description
PCLK	Input	Clock source	APB bus clock
PRESETn	Input	Reset controller	APB bus reset
PADDR[11:2]	Input	Master	APB bus address
PSEL	Input	Master	APB bus peripheral select
PENABLE	Input	Master	APB bus enable
PWRITE	Input	Master	APB bus write signal
PWDATA[31:0]	Input	Master	APB bus write data
PRDATA[31:0]	Output	Master	APB bus read data

A.2 Miscellaneous internal signals

Refer to Table A-2 for an overview of the miscellaneous internal signals.

Table A-2 Miscellaneous internal signals

Name	Type	Source/ destination	Description
MMCIINTR0	Output	Interrupt controller	Interrupt request 0
MMCIINTR1	Output	Interrupt controller	Interrupt request 1
MMCIDMASREQ	Output	DMA controller	DMA single word request
MMCIDMABREQ	Output	DMA controller	DMA burst request
MMCIDMALSREQ	Output	DMA controller	DMA last word request
MMCIDMALBREQ	Output	DMA controller	DMA last burst request
MMCIDMACLR	Input	DMA controller	DMA request clear
MCLK	Input	Clock source	Multimedia card adapter clock
nMCLK	Input	Clock source	Inverted MCLK
nMMCIRST	Input	Reset controller	Reset signal for **MCLK** domain

A.3 Scan test control signals

The internal scan test control signals are shown in Table A-3.

Table A-3 Scan test control signals

Name	Type	Source/ destination	Description
SCANENABLE	Input	Scan controller	Scan enable
SCANINPCLK	Input	Scan controller	Scan data input for **PCLK** domain
SCANINMCLK	Input	Scan controller	Scan data input for **MCLK** domain
SCANINnMCLK	Input	Scan controller	Scan data input for **nMCLK** domain
SCANINMMCIFBCLK	Input	Scan controller	Scan data input for **FBCLK** domain
SCANOUTPCLK	Output	Scan controller	Scan data output for **PCLK** domain
SCANOUTMCLK	Output	Scan controller	Scan data output for **MCLK** domain
SCANOUTnMCLK	Output	Scan controller	Scan data output for **nMCLK** domain
SCANOUTMMCIFBCLK	Output	Scan controller	Scan data output for **FBCLK** domain

A.4 MMCI signals

Refer to Table A-4 for an overview of the off-chip PrimeCell MMCI signals.

Table A-4 PrimeCell MMCI signals

Name	Type	Source/ destination	Description
MMCICLKOUT	Output	Pad	Multimedia card clock output
MMCICMDIN	Input	Pad	Multimedia card command input
MMCICMDOUT	Output	Pad	Multimedia card command output
nMMCICMDEN	Output	Pad	Multimedia card command output enable
MMCIDATIN	Input	Pad	Multimedia card data input
MMCIDATOUT	Output	Pad	Multimedia card data output
nMMCIDATEN	Output	Pad	Multimedia card data enable
MMCIPWR	Output	Pad	Power supply enable
MMCIVDD[3:0]	Output	Pad	Power supply output voltage
MMCIROD	Output	Pad	Open-drain resistor enable
MMCIFBCLK	Input	Pad	Fed-back clock from multimedia card

Index

The items in this index are listed in alphabetic order, with symbols and numerics appearing at the end. The references given are to page numbers.