Technical Note

Booting from Embedded MMC

Introduction

MultiMediaCard (MMC) technology provides low-cost data storage media for many mobile applications. MMC benefits include:

- Royalty-free design
- Industry-standard interface
- High-performance, 8-bit, 52 MHz I/O interface

MMC devices handle NAND Flash chores such as wear leveling and error correction, freeing the host system from these tasks.

This technical note describes booting from an embedded MMC (eMMC) device—such as Micron's Managed NAND—and provides supporting documentation for designers incorporating MMCs in their applications. It draws on the MMC specification published by the MultiMediaCard Association and includes details from that specification (the complete specification is available at: www.mmca.org/home).

Booting from Managed NAND

To implement Managed NAND as the single nonvolatile memory device in an embedded system, the system must boot up from the Managed NAND device.

Managed NAND memory devices do not inherently support execute in place (XIP). Operating system (OS) code and boot code can be stored in the Managed NAND device, but the code must be copied (or shadowed) to RAM before it can be executed. Because of this, system designers must modify the boot process to support code shadowing from a Managed NAND device. The benefits of this modification are the lower cost of Managed NAND storage media and the high performance of RAM as the XIP memory.

This document focuses on booting from a Managed NAND device in applications using a 32-bit ARM® processor. Each stage in the boot sequence process is described in detail.

Terms and abbreviations used in this technical note are defined in Table 1.

Table 1: Terms and Abbreviations

Abbreviation	Description	Abbreviation	Description
CID	MMC device ID	DAT	MMC data signal
CMD	MMC command signal	eMMC	Embedded MMC
CRC	Cyclic redundancy code	Managed NAND	Micron's eMMC device
CSD	Card-specific data	MMC	MultiMediaCard

Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2006 Micron Technology, Inc. All rights reserved.

Products and specifications discussed herein are for evaluation and reference purposes only and are subject to change by Micron without notice. Products are only warranted by Micron to meet Micron's production data sheet specifications. All information discussed herein is provided on an "as is" basis, without warranties of any kind.

MMC Bus Description

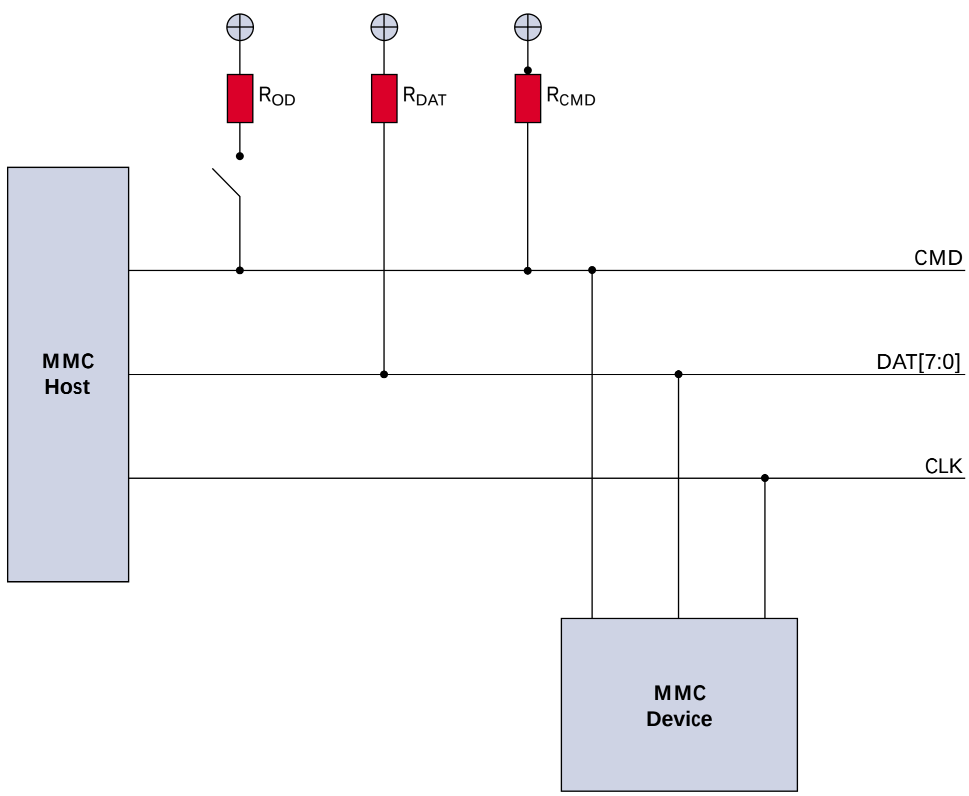
The CLK, CMD, and DAT[7:0] pins are used for all MMC bus communication (see Figure 1).

The CLK signal synchronizes data between the MMC device and the host (system processor) on the MMC bus. With each CLK LOW-to-HIGH cycle, a bit transfer occurs on the CMD and DAT lines. The CLK frequency can vary from 0 to 52 MHz depending on the current state of the MMC device.

The bidirectional CMD channel transfers commands from the host to the MMC device and transfers responses from the device back to the host. The CMD channel operates in 400 kHz open-drain mode during initialization of the MMC device. The CMD channel switches to a higher-frequency push-pull mode for fast READ, WRITE, and ERASE command transfers. Push-pull mode supports frequencies of up to 52 MHz. The host uses the R_{OD} pull-up resistor on the CMD line to control transitions between open-drain and push-pull mode (see Figure 1).

The DAT channels support only push-pull mode and are bidirectional. Most data is read from or written to the MMC device via the DAT channels. DAT channels also indicate the device status (ready or busy) during some MMC commands.

Figure 1: MMC Bus



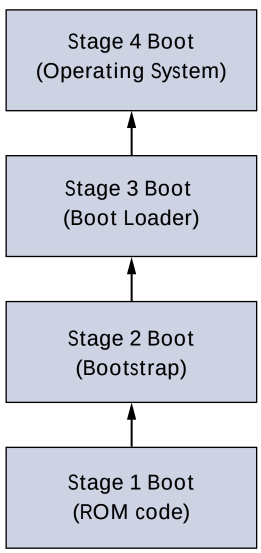
Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2006 Micron Technology, Inc. All rights reserved.

Managed NAND Boot Sequence for 32-bit ARM Processors

The boot sequence for a 32-bit ARM-processor-based system involves the four steps shown in Figure 2.

- Stage 1: ROM code initializes the Managed NAND device and copies (or shadows) the code for stage 2 from the Managed NAND device to RAM. The ROM code then jumps to the RAM memory location containing the first instruction of the stage 2 code.
- Stage 2: Code runs in RAM. It bootstraps the system by shadowing the stage 3 boot-loader code from the Managed NAND device to RAM. The stage 2 code then jumps to the RAM memory location containing the first instruction of the stage 3 code.
- Stage 3: Code also runs in RAM. It shadows the OS code from the Managed NAND device to RAM. The stage 3 code then jumps to the RAM memory location containing the first instruction in the OS.
- Stage 4: The OS takes control of the system.

Figure 2: Boot Sequence



Three Methods for Accomplishing Stages 1–4

System designers have several methods available to them for initializing the Managed NAND device and starting the transfer of boot code from the device. These methods are dependent on the specification and commands supported by the Managed NAND device. They include:

- Boot method for Managed NAND devices supporting MMCA Specification v4.2.
- Boot method for Managed NAND devices supporting MMCA Specification v4.3, MMC Boot Sequence Option 1.
- Boot method for Managed NAND devices supporting MMCA Specification v4.3, MMC Boot Sequence Option 2.

Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2006 Micron Technology, Inc. All rights reserved.

Managed NAND Devices Supporting MMCA Specification v4.2

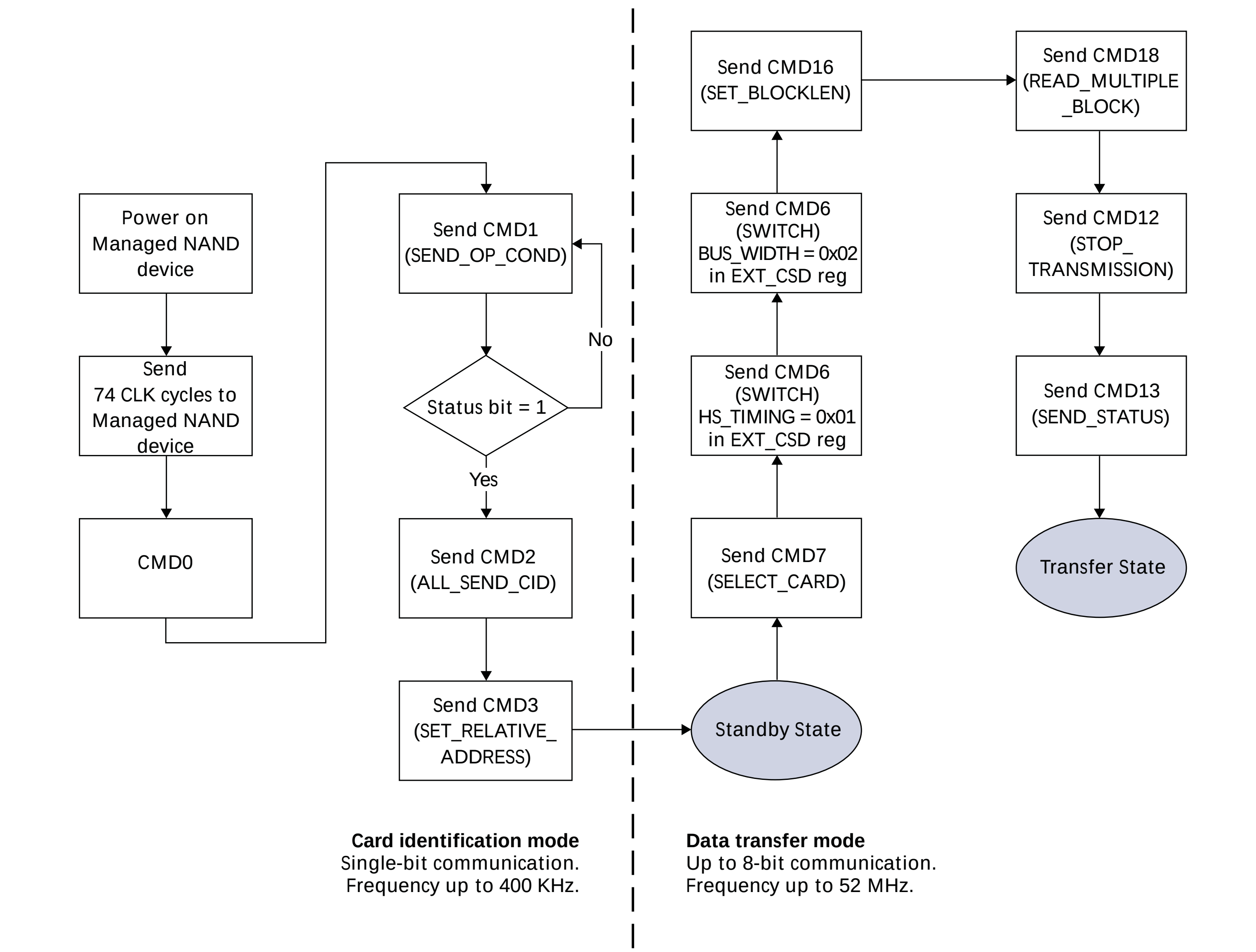
MMCA v4.2 Managed NAND devices include support for 52 MHz operation and support densities greater than 2GB. While these devices do not have a specific boot mode they can support boot operations via the method described below.

Managed NAND Initialization and Data Transfer

Before the boot code can be read from the device and copied into RAM for execution, the host system must initialize the Managed NAND device with a series of commands. Figure 3 shows device initialization and data transfer steps for the Managed NAND device. The commands are described in Table 2 on page 9 and Table 3 on page 10.

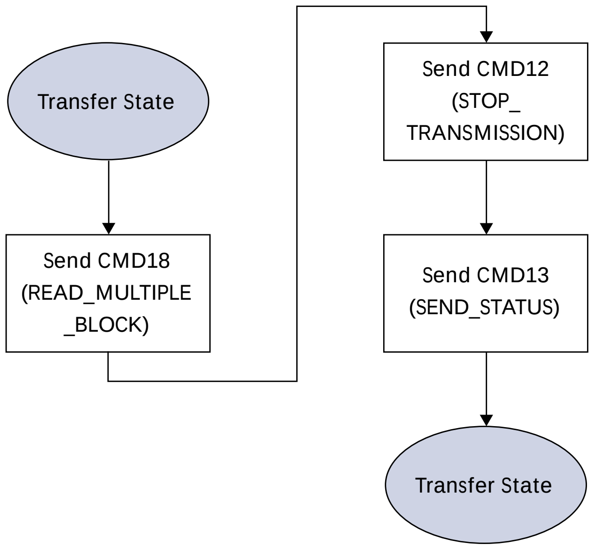
After the Managed NAND device has been initialized, boot data is read from the device as shown in Figure 4 on page 5.

Figure 3: ROM Code Flowchart for MMC Initialization and Data Transfer



Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2006 Micron Technology, Inc. All rights reserved.

Figure 4: Flow to Read Boot-Loader and OS Code from MMC Device

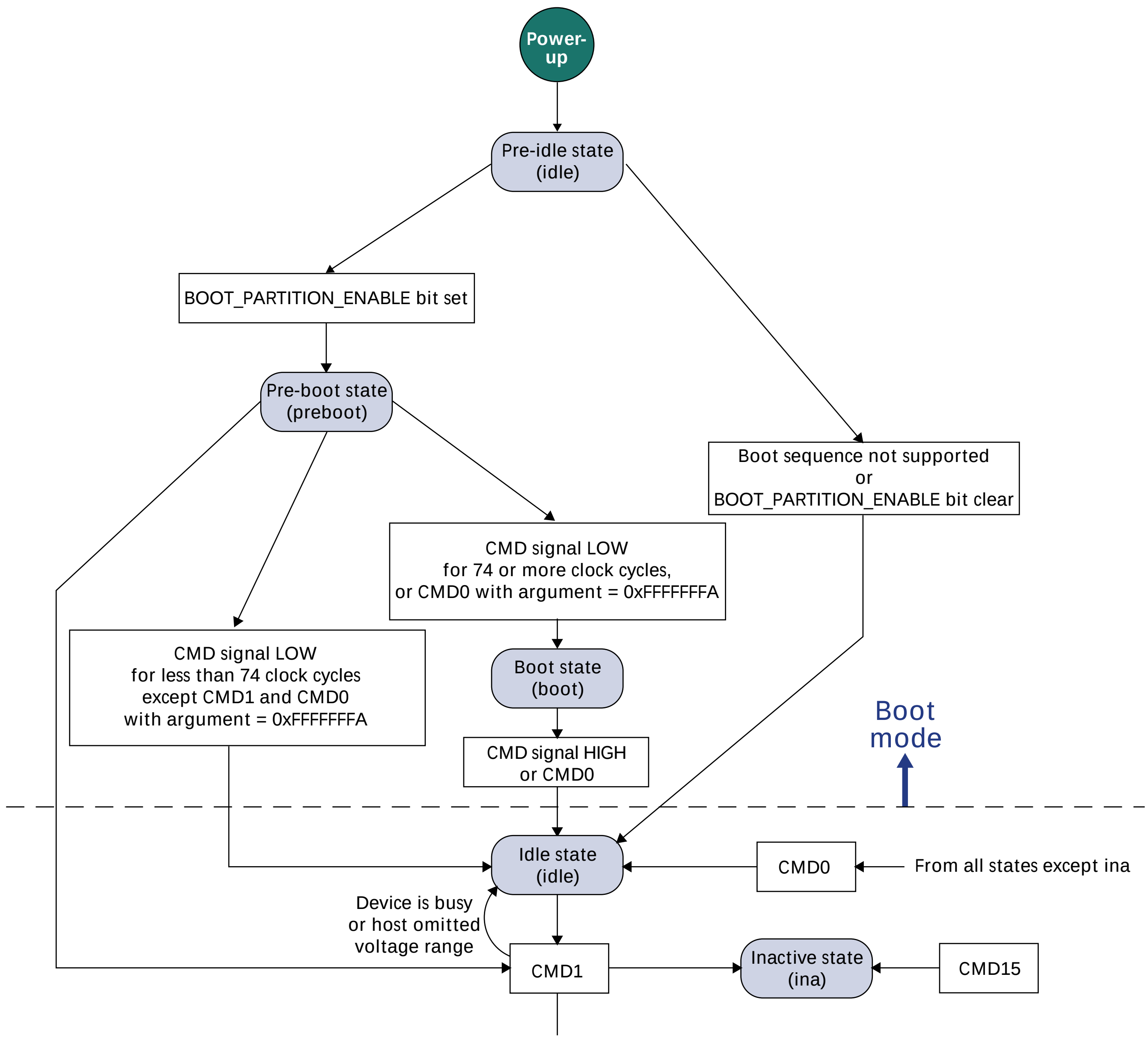


Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2006 Micron Technology, Inc. All rights reserved.

Managed NAND Devices Supporting MMCA Specification v4.3

The MMCA Specification 4.3-compliant devices are backward compatible and support MMCA Specification v4.2 boot methods described previously. They also offer a simpler approach to reading boot code from the device, and either of two options can be used.

Figure 5: MMC Specification 4.3 Boot Sequence Options 1 and 2 Flowchart



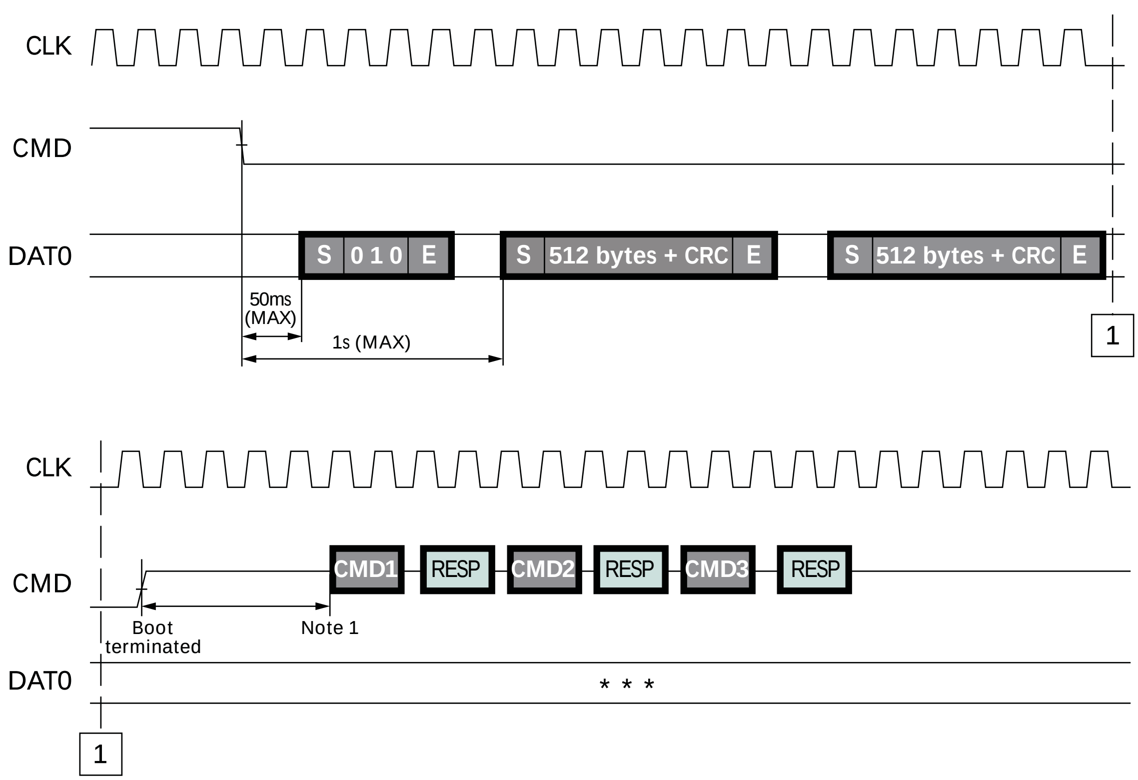
Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2006 Micron Technology, Inc. All rights reserved.

Boot Sequence Option 1

Details for implementing boot sequence option 1 include:

- The initialization described in Figure 3 on page 4 is not required.
- If the host sends any normal MMC commands before initiating boot, the device will be locked out of boot mode until the next power cycle.
- Immediately after power-up, the host must hold the MMC CMD line LOW for 74 clock cycles. This tells the Managed NAND device to access its boot partition.
- The host does not need to provide an address for the boot partition.
- If boot acknowledge is enabled, the Managed NAND device responds to the CMD line LOW with a boot acknowledge pattern of "010" if it is bootable.
- There is a maximum of 50ms (typically 25ms) from CMD line LOW to boot acknowledge on DAT[0].
- The host can now read boot code in boot partition by clocking the Managed NAND device.
- Boot operation is terminated by CMD line HIGH or all contents of enabled boot partition are sent to host.
- A minimum of 56 clocks are required after terminating boot.
- The device returns to open-drain mode at the end of boot (CMD line driver only).
- See Figure 6 on page 7 for additional details.

Figure 6: MMC Specification v4.3 Boot Sequence Option 1



Notes: 1. MIN: 8 clocks + 48 clocks = 56 clocks required from CMD line HIGH to next MMC command.

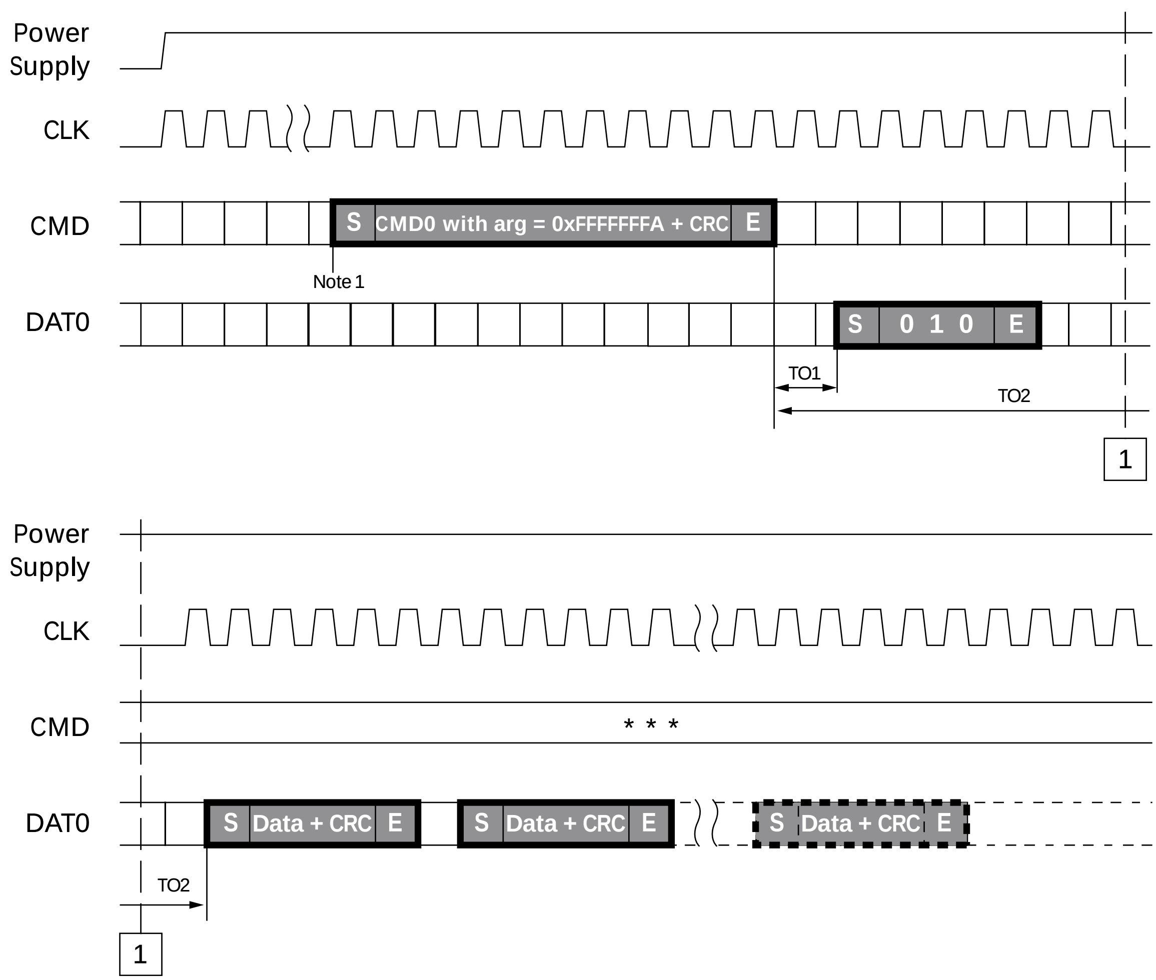
Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2006 Micron Technology, Inc. All rights reserved.

Boot Sequence Option 2

Details for implementing boot sequence option 2 include:

- The initialization described in Figure 2 on page 3 is not required.
- If the host sends any normal MMC commands before initiating boot, the device will be locked out of boot mode until the next power cycle.
- Immediately after power-up, the host sends CMD 0 with argument 0xFFFFFFFA to tell the Managed NAND device to access the boot partition.
- If boot acknowledge is enabled, the Managed NAND device responds to the CMD line LOW with a boot acknowledge pattern of “010” if it is bootable.
- The host can now read boot code in the boot partition by clocking the Managed NAND device.
- There is a maximum of 1 second from the end bit of CMD0 to the start of boot data on the DAT line(s).
- Boot operation is terminated only by the host sending CMD0 RESET or all contents of the enabled boot partition sent to host.
- See Figure 7 on page 8 for additional details.

Figure 7: MMC Specification 4.3 Boot Sequence Option 2



Notes: 1. A minimum of 74 clocks are required after power is stable to the start bit of boot command.

Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2006 Micron Technology, Inc. All rights reserved.

MMCA Specification v4.3 Alternative Method for Accessing Boot Code

Assuming that the Managed NAND has already been initialized, the host can read the boot partition data using the following process:

- The host sends a SELECT command to put the device into transfer state.
- The host sends a SWITCH command to set BOOT_PARTITION_ACCESS[2:0] bits in extended CSD byte[179].
- The host reads boot data using a single- or multiple-block READ command with the specific addressing mode supported by the device; i.e., byte addressing or sector addressing.
- The host sends a SWITCH command to clear BOOT_PARTITION_ACCESS[2:0] bits in extended CSD byte[179].

Command Definitions and Registers

Tables 2 through 4 provide the command definitions and registers designers will need to correctly implement an embedded MMC device in their designs.

Table 2: MMC Basic Commands

CMD INDEX	Argument	Response	Abbreviation	Command Description
CMD0	[31:0] Stuff bits	–	GO_IDLE_STATE	Resets the card to idle state.
CMD1	[31:0] OCR without busy	R3	SEND_OP_COND	Asks the card, in idle state, to send its operating conditions register contents in the response on the CMD line.
CMD2	[31:0] Stuff bits	R2	ALL_SEND_CID	Asks the card to send its CID number on the CMD line.
CMD3	[31:16] RCA [15:0] Stuff bits	R1	SET_RELATIVE_ADDR	Assigns relative address to the card.
CMD4	[31:16] DSR [15:0] Stuff bits	–	SET_DSR	Programs the driver stage register of the card.
CMD5	[31:16] RCA [15] Sleep/awake [14:0] Stuff bits	R1b	Sleep	Toggles device between the standby and the sleep state. Set “1” for the device to enter the sleep state and “0” for it to exit the sleep state.
CMD6	[31:26] Set to 0 [25:24] Access [23:16] Index [15:8] Value [7:3] Set to 0 [2:0] Command set	R1b	SWITCH	Switches the mode of operation of the selected card or modifies the EXT_CSD registers (see Table 4 on page 10).
CMD7	[31:16] RCA [15:0] Stuff bits	R1b	SELECT/DESELECT_CARD	Command toggles a card between the standby and transfer states or between the programming and disconnect states. In both cases the card is selected by its own relative address and gets deselected by any other address; address 0 deselects the card.
CMD8	[31:0] Stuff bits	R1	SEND_EXT_CSD	The card sends its EXT_CSD register as a block of data.
CMD9	[31:16] RCA [15:0] Stuff bits	R2	SEND_CSD	Addressed card sends its card-specific data (CSD) on the CMD line.
CMD10	[31:16] RCA [15:0] Stuff bits	R2	SEND_CID	Addressed card sends its card identification (CID) on the CMD line.

Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2006 Micron Technology, Inc. All rights reserved.

Table 2: MMC Basic Commands (continued)

CMD INDEX	Argument	Response	Abbreviation	Command Description
CMD11	[31:0] Data address	R1	READ_DAT_UNTIL_STOP	Reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows.
CMD12	[31:0] Stuff bits	R1b	STOP_TRANSMISSION	Forces the card to stop transmission.
CMD13	[31:16] RCA [15:0] Stuff bits	R1	SEND_STATUS	Addressed card sends its status register.
CMD14	[31:0] Stuff bits	R1	BUSTEST_R	A host reads the reversed bus testing data pattern from a card.
CMD15	[31:16] RCA [15:0] Stuff bits	–	GO_INACTIVE_STATE	Sets the card to inactive state.
CMD19	[31:0] Stuff bits	R1	BUSTEST_W	A host sends the bus test data pattern to a card.

Table 3: MMC Block-Oriented Read Commands

CMD INDEX	Argument	Response	Abbreviation	Command Description
CMD16	[31:0] Block length	R1	SET_BLOCKLEN	Sets the block length (in bytes) for all following block commands (READ and WRITE). Default block length is specified in the CSD.
CMD17	[31:0] Data address	R1	READ_SINGLE_BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.
CMD18	[31:0] Data address	R1	READ_MULTIPLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by a stop command, or until the requested number of data blocks is transmitted.

Table 4: Extended CSD

Name		Field	Size (Bytes)	Cell Type	CSD Slice	Extended CSD Value
Properties segment						
Reserved[1]			7	–	[511:505]	0h
Supported command sets		S_CMD_SET	1	R	[504]	0h
Reserved[1]			275	TBD	[503:229]	0h
Boot information		BOOT_INFO	1	R	[228]	1h
Reserved[1]			1	TBD	[227]	0h
Boot partition size		BOOT_SIZE_MULT	1	R	[226]	1h
Access size		ACC_SIZE	1	R	[225]	
High-capacity erase unit size		HC_ERASE_GP_SIZE	1	R	[224]	
High-capacity erase timeout		ERAE_TIMEOUT_MULT	1	R	[223]	
Reliable write-sector count		REL_WR_SEC_C	1	R	[222]	01h
High-capacity write protect group size		HC_WP_GRP_SIZE	1	R	[221]	
Sleep current (VCC)		S_C_VCC	1	R	[220]	01h
Sleep current (VCCQ)		S_C_VCCQ	1	R	[219]	06h
Reserved[1]			1	TBD	[218]	00h
Sleep/awake timeout		S_A_TIMEOUT	1	R	[217]	0Fh
Reserved[1]			1	TBD	[216]	0h

Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2006 Micron Technology, Inc. All rights reserved.

Table 4: Extended CSD (continued)

Name		Field	Size (Bytes)	Cell Type	CSD Slice	Extended CSD Value
Sector count	1G	SEC_COUNT	4	R	[215:212]	FA800h
	2G					3CE000h
	4G					778000h
	8G					
Reserved[1]			1			
Minimum write performance for 8-bit at 52 MHz		MIN_PERF_W_8_52	1	R	[210]	
Minimum read performance for 8-bit at 52 MHz		MIN_PERF_R_8_52	1	R	[209]	
Minimum write performance for 8-bit at 26 MHz and 4-bit at 52 MHz		MIN_PERF_W_8_26_4_52	1	R	[208]	
Minimum read performance for 8-bit at 26 MHz and 4-bit at 52 MHz		MIN_PERF_R_8_26_4_52	1	R	[207]	
Minimum write performance for 4-bit at 26 MHz		MIN_PERF_W_4_26	1	R	[206]	
Minimum read performance for 4-bit at 26 MHz		MIN_PERF_R_4_26	1	R	[205]	
Reserved			1	TBD	[204]	
Power class for 26 MHz at 3.6V		PWR_CL_26_360	1	R	[203]	00h
Power class for 52 MHz at 3.6V		PWR_CL_52_360	1	R	[202]	00h
Power class for 26 MHz at 1.95V		PWR_CL_26_195	1	R	[201]	00h
Power class for 52 MHz at 1.95V		PWR_CL_52_195	1	R	[200]	00h
Reserved[1]			3		[199:197]	0h
Card type		CARD_TYPE	1	R	[196]	3h
Reserved[1]			1	TBD	[195]	0h
CSD structure version		CSD_STRUCTURE	1	R	[194]	2h
Reserved[1]			1	TBD	[193]	0h
Extended CSD revision		EXT_CSD_REV	1	R	[192]	3h
Modes Segment						
Command set		CMD_SET	1	R/W	[191]	0h
Reserved[1]			1		[190]	0h
Command set revision		CMD_SET_REV	1	RO	[189]	0h
Reserved[1]			1		[188]	0h
Power class		POWER_CLASS	1	R/W	[187]	0h
Reserved[1]			1		[186]	0h
High-speed interface timing		HS_TIMING	1	R/W	[185]	0h
Reserved[1]			1		[184]	0h
Bus width mode		BUS_WIDTH	1	WO	[183]	0h
Reserved[1]			1		[182]	0h
Erased memory content		ERASED_MEM_CONT	1	RO	[181]	0h
Reserved[1]			1		[180]	0h
Boot configuration		BOOT_CONFIG	1	R/W	[179]	0h
Reserved[1]			1		[178]	0h
Boot bus width[1]		BOOT_BUS_WIDTH	1	R/W	[177]	0h
Reserved[1]			1		[176]	0h
High-density erase group definition		ERASE_GROUP_DEF	1	R/W	[175]	0h
Reserved[1]			175		[174:0]	0h

Notes: 1. Reserved bits should be read as “0.”

Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2006 Micron Technology, Inc. All rights reserved.

Summary

MMC devices provide low-cost, high-performance, industry-standard data storage media for many mobile applications. These devices free the host processor from routine tasks such as wear leveling and error correction. By enabling systems to boot from MMC, designers can use an MMC device as a single, nonvolatile memory source, thus reducing costs and achieving higher performance than they can obtain from other data storage options.

8000 S. Federal Way, P.O. Box 6, Boise, ID 83707-0006, Tel: 208-368-3900
prodmktg@micron.com www.micron.com Customer Comment Line: 800-932-4992
Micron, the M logo, and the Micron logo are trademarks of Micron Technology, Inc.All other trademarks are the property of their respective owners.

Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2006 Micron Technology, Inc. All rights reserved.