MultiMediaCard™

User's Manual

HITACHI

ADE-603-002A
Rev. 2.0
9/6/01
Hitachi, Ltd.

Hitachi semiconductor

Cautions

1. Hitachi neither warrants nor grants licenses of any rights of Hitachi's or any third party's patent, copyright, trademark, or other intellectual property rights for information contained in this document. Hitachi bears no responsibility for problems that may arise with third party's rights, including intellectual property rights, in connection with use of the information contained in this document.
2. Products and product specifications may be subject to change without notice. Confirm that you have received the latest product standards or specifications before final design, purchase or use.
3. Hitachi makes every attempt to ensure that its products are of high quality and reliability. However, contact Hitachi's sales office before using the product in an application that demands especially high quality and reliability or where its failure or malfunction may directly threaten human life or cause risk of bodily injury, such as aerospace, aeronautics, nuclear power, combustion control, transportation, traffic, safety equipment or medical equipment for life support.
4. Design your application so that the product is used within the ranges guaranteed by Hitachi particularly for maximum rating, operating supply voltage range, heat radiation characteristics, installation conditions and other characteristics. Hitachi bears no responsibility for failure or damage when used beyond the guaranteed ranges. Even within the guaranteed ranges, consider normally foreseeable failure rates or failure modes in semiconductor devices and employ systemic measures such as fail-safes, so that the equipment incorporating Hitachi product does not cause bodily injury, fire or other consequential damage due to operation of the Hitachi product.
5. This product is not designed to be radiation resistant.
6. No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without written approval from Hitachi.
7. Contact Hitachi's sales office for any questions regarding this document or Hitachi semiconductor products.

Contents

MultiMediaCard™
The industry's smallest flash/ROM card standard, developed by Infineon Technologies AG of Germany and SanDisk Corporation of the USA. MultiMediaCard™ is a trademark of Infineon Technologies AG of Germany, and is licensed to the MultiMediaCard™ Association (MMCA).
The MMCA is the standardization body for small Secure MultiMediaCard™ and MultiMediaCard™—ranking among the smallest and lightest of such devices in the world—and currently counts some 80 member companies.

Section 1 History and Features of MultiMediaCard™

1.1 History of Flash Cards

The history of cards using flash memory begins with the PC memory card whose standard was decided by the PCMCIA in 1990. This implementation of a PC memory card by means of flash memory is now generally called a linear flash memory card, and is mainly used as expansion memory in small portable devices such as notebook personal computers.

Then, with the revision of the PC card standard in 1992, the PC-ATA card standard (PC Card ATA specification) was born as an IDE/ATA-compatible I/O card using the same kind of interface as a hard disk. This card has become so widely used that most notebook PCs currently on the market include a slot for such cards.

From this point the trend was toward smaller flash cards, and 1995 saw the release of the CompactFlash™ card offering the same functions as the PC-ATA card in a smaller package. Production of CompactFlash™ cards grew at a rapid rate thanks to their use in products such as digital cameras and handheld PCs. This same period saw the advent of Smart Media as cards containing flash memory only, and together with CompactFlash™ cards, these have also come into widespread use as a storage medium for digital cameras and the like. Then, in 1996, came the announcement of the specifications of a new miniature card designed as a general-purpose memory card.

In 1998, the MultiMediaCard™ (MMC) and Memory Stick were released, offering drastically smaller size for use in small devices such as portable music players. The MultiMediaCard™ featured an amazing 80% reduction in cubic capacity compared with CompactFlash™, and has brought a constant stream of new products such as MP3 players and video cameras designed for its use.

As with CompactFlash™, the target for MultiMediaCard™ is to achieve industry standardization by taking an open standard approach, and the MultiMediaCard™ Association (MMCA)*[1] was established in January 1998 to promote such standardization.

In 1999, cards equipped with copyright protection functions made their appearance in rapid succession—the Magic Gate Memory Stick, SD Memory Card, and Secure MultiMediaCard™. The Secure MultiMediaCard™—the smallest of the three—is a MultiMediaCard™ with copyright protection functions added, and maintains MultiMediaCard™ compatibility.

The brief history of flash cards given above is summarized in figure 1.1.

Hitachi has to date released PC-ATA cards, CompactFlash™*[2] cards, and MultiMediaCard™, and henceforth plans to undertake active marketing of Secure MultiMediaCard™.

Notes: 1. MMCA Home Page: http://www.mmca.org/
2. CompactFlash is a trademark of SanDisk Corporation of the USA, and is licensed to the CFA (CompactFlash™ Association).

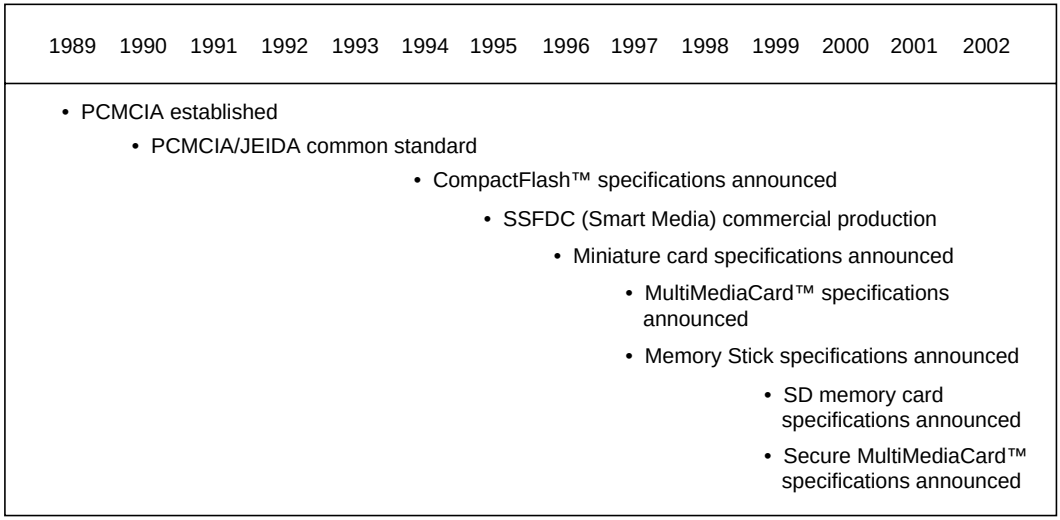


Figure 1.1 History of Flash Cards

1.2 Features of MultiMediaCard™

The important factors for future small flash cards will be size and weight, since major considerations in selecting component parts for portable devices such as mobile phones are small size and light weight. As shown in figure 1.2 and table 1.1, at 1.5 g the MultiMediaCard™ is the lightest of the small flash cards, making it ideal for such products. And its postage-stamp size of 32 mm × 24 mm, and thickness of only 1.4 mm, allow it to be installed virtually anywhere.

With personal products such as audio players, ease of use is a major factor in achieving popularity. A MultiMediaCard™ connects to the system via only seven pins and includes a mechanism to prevent incorrect insertion, ensuring trouble-free use by the average user with no specialized knowledge of such devices.

There are currently two kinds of MultiMediaCard™—flash cards and ROM cards. In the future, we can expect to see the release of I/O cards as well.

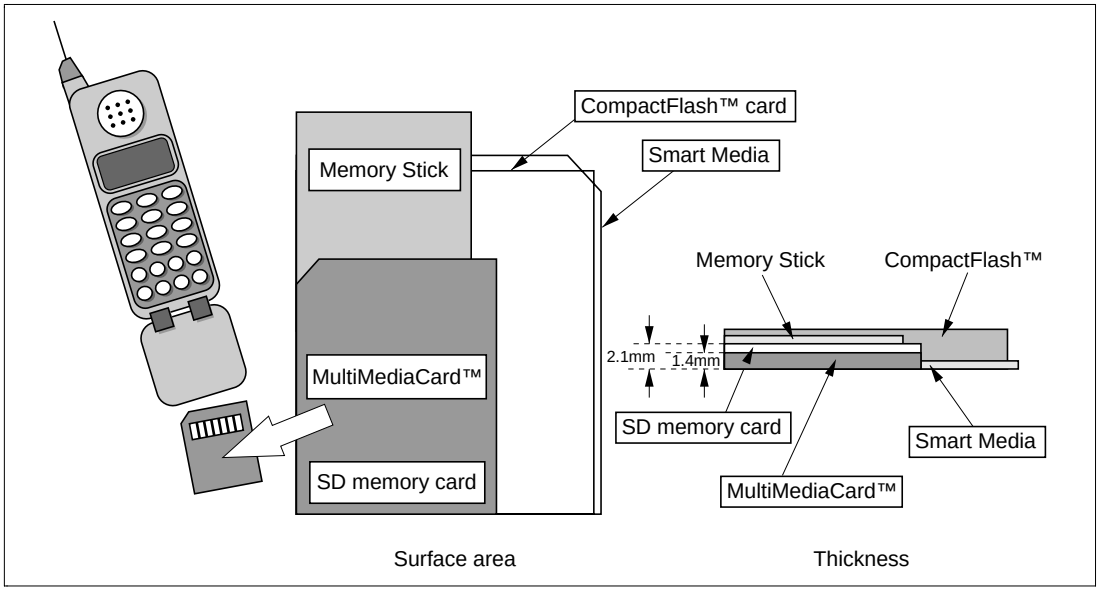


Figure 1.2 Comparative Sizes of Small Flash Cards

Table 1.1 Small Flash Card Specifications

	Card Type						
Item	**CompactFlash™ Card**	**Smart Media**	**MultiMediaCard™**	**Secure MultiMediaCard™**	**Memory Stick**	**Magic Gate Memory Stick**	**Secure Digital**
Size	42.8 × 36.4 × 3.3 mm	45.0 × 37.0 × 0.76 mm	32.0 × 24.0 × 1.4 mm	32.0 × 24.0 × 1.4 mm	50.0 × 21.5 × 2.8 mm	50.0 × 21.5 × 2.8 mm	32.0 × 24.0 × 2.1 mm
Weight	10 g	2 g	1.5 g	1.5 g	4 g	4 g	Approx. 2 g
Pins	50	22	7	7	10	10	9
Data pins	16	8	1	1	1	1	4
Copyright protection functions	No	No	No	Yes	No	Yes	Yes

1.3 MMCA Standard Ver. 3.1

Ver. 2.11 was used until June 2001, when Ver. 3.1 was released. The main differences between these versions are summarized in table 1.2.

Table 1.2 Differences between MMCA Standard Ver. 2.11 and Ver. 3.1

No.	Item	v2.11	v3.1
1	Low voltage	2.0 V to 3.6 V	1.6 V to 3.6 V (Not fully downward-compatible)
2	Forced erasure of locked card in MMC mode	Undefined	• When TMP_WRITE_PROTECT in CSD is 1, forced erasure is performed and TMP_WRITE_PROTECT is cleared to 0. • When PERM_WRITE_PROTECT in CSD is 1, erasure fails and the LOCK_UNLOCK_FAILED bit is set to 1. (The card remains locked.) (The same applies to SPI mode.)
3	Multiple Read/Write in SPI mode	Not supported	The same transfer modes are supported as for Multiple Read/Write in MMC mode.
Applicable Hitachi MultiMedia Cards		HB28xxxxMM1 Series*	HB28xxxxMM2 Series*

Note: * See the respective Data Sheets for detailed specifications.

Section 2 Overview of Hitachi MultiMediaCard™

2.1 MultiMediaCard™ System Concept

As MultiMediaCard™ are intended for information and content storage in small information devices such as mobile phones and portable audio players, their system concept is to offer ease of use in the host system while reducing the load on the system.

Examples of system configurations using a MultiMediaCard™ are shown in figure 2.1.

1. Software control

   With this method, MultiMediaCard™ signals are directly connected to a microcontroller port. The MultiMediaCard™ protocol is emulated by means of software control of port signals. The host system is extremely simple, but data transfer speed is low.

2. Adapter control

   With this method, an adapter (hardware) is inserted between the microprocessor and the MultiMediaCard™ to control the MultiMediaCard™.

3. Adapter control via simple bus

   With this method, a MultiMediaCard™ adapter is connected to a simple bus via an application adapter.

4. Adapter control via PC bus

   With this method, a MultiMediaCard™ adapter is connected to the PC bus via a bus bridge. The system is complex and expensive, but data transfer speed is increased.

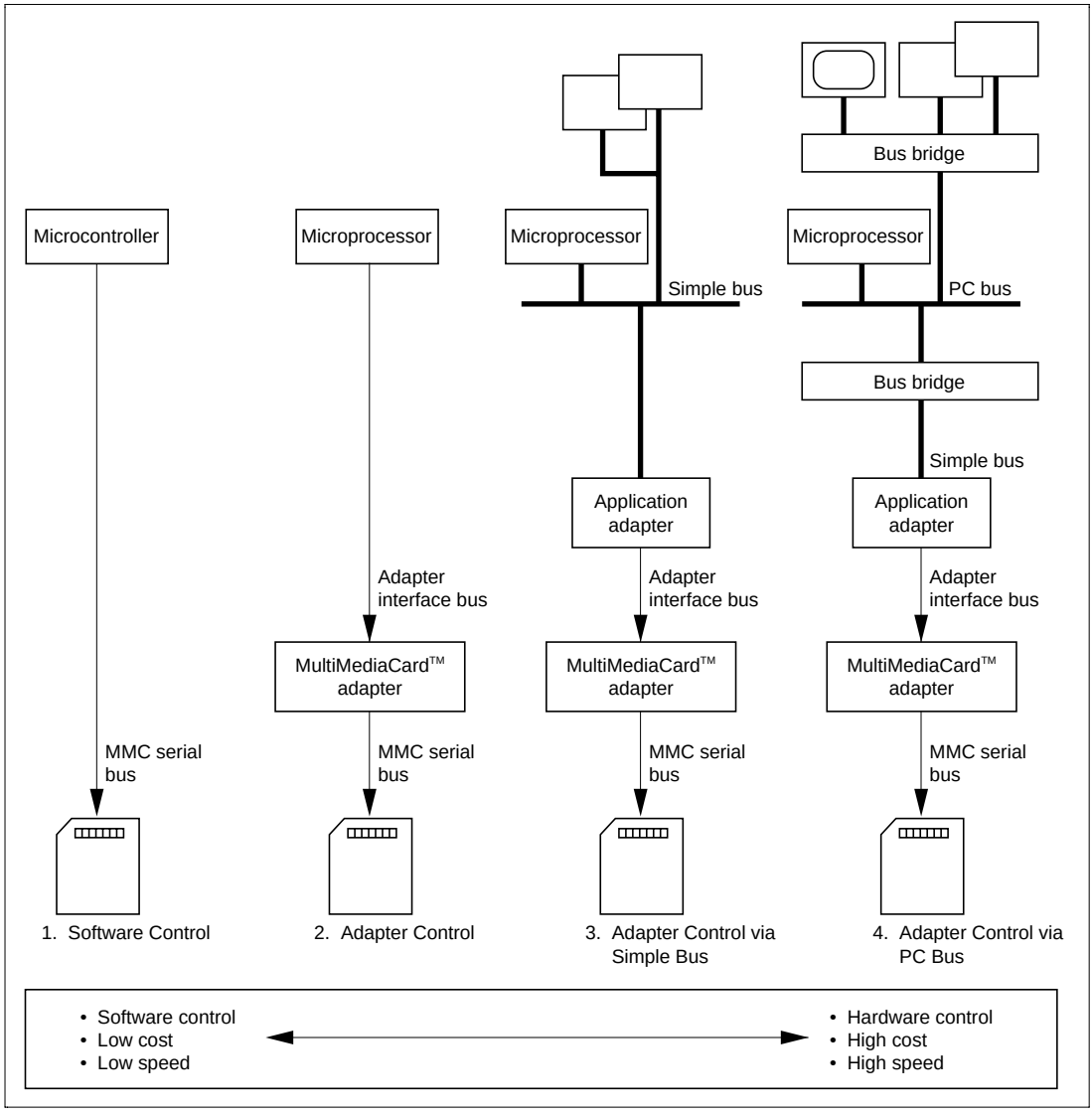


Figure 2.1 Sample MultiMediaCard™ System Configurations

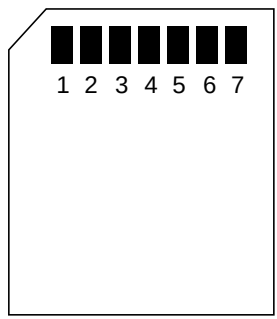
2.2 Bus Interface

2.2.1 Interface

As shown in figure 2.2, the MultiMediaCard™ interface is a 7-pin serial interface, and there are two communication protocol modes—MMC (MultiMediaCard™) mode and SPI mode. The maximum clock operating frequency in both modes is 20 MHz. Although the interface method differs in the two modes, data written in either mode can be read by a host in either mode. Hitachi's MultiMediaCard™ also allow operation with either an MMC or SPI host. Details of the mode setting method are given later, but basically, SPI mode is entered if the CS pin is driven low when command 0 is issued during the MultiMediaCard™ identification procedure.

(A) MMC Mode

Pin No.	Name	Type	Pin Function
1	RSV	NC	Reserved pin
2	CMD	I/O, PP, OD	Command/response
3	V_{SS1}	S	GND
4	V_{DD1}	S	V_{CC}
5	CLK	I	Clock
6	V_{SS2}	S	GND
7	DAT	I/O, PP	Data input/output



(B) SPI Mode

Pin No.	Name	Type	Pin Function
1	CS	I	Chip select
2	DI	I	Data input
3	V_{SS}	S	GND
4	V_{DD}	S	V_{CC}
5	SCLK	I	Clock
6	V_{SS2}	S	GND
7	DO	O, PP	Data output

Note: S = Power supply
I = Input
O = Output
PP = Push pull
OD = Open drain
NC = Not connected

Figure 2.2 MultiMediaCard™ Pin Assignments

2.2.2 Identification of Multiple Cards

Up to 30 MultiMediaCard™ can be used on the same bus. The method of selecting the card to be accessed from among a number of cards is as follows for each mode.

1. MMC mode

   In MMC mode, in order to select a particular card for access from among a number of cards, a relative address RCA (relative card address: see section 2.4, Registers) corresponding to a unique name is assigned to each card during the identification procedure, and a card is selected by specifying this relative address.

2. SPI mode

   In SPI mode, a chip select signal (CS) is connected to each card, and the card to be accessed is selected by asserting the chip select signal for that card to the low level.

2.2.3 Bus Protocol

Figure 2.2 shows the MultiMediaCard™ pin assignments, and figure 2.3 shows schematic diagrams of the protocol in each mode. The MultiMediaCard™ operates when a command corresponding to a particular function is issued to the card from the host.

1. MMC mode

   In MMC mode, three signals are used to interface to the host system: clock, command, and data. The clock signal is used to maintain synchronization between the system and the card. The command signal is used to issue commands from the host to the card and to return responses from the card to the host. The data signal is used to write and read data to and from the card. The command and data signals are bidirectional bus signals.

2. SPI mode

   In SPI mode, four signals are used to interface to the host system: clock, data in, data out, and chip select. The clock signal is used to maintain synchronization between the system and the card, as in MMC mode. The data in signal is used to issue commands from the host to the card and to write data to the card, while the data out signal is used to return responses from the card to the host and to read data from the card. The data in and data out signals are unidirectional bus signals. The chip select signal is used to select a particular card to be accessed from among any number of cards.

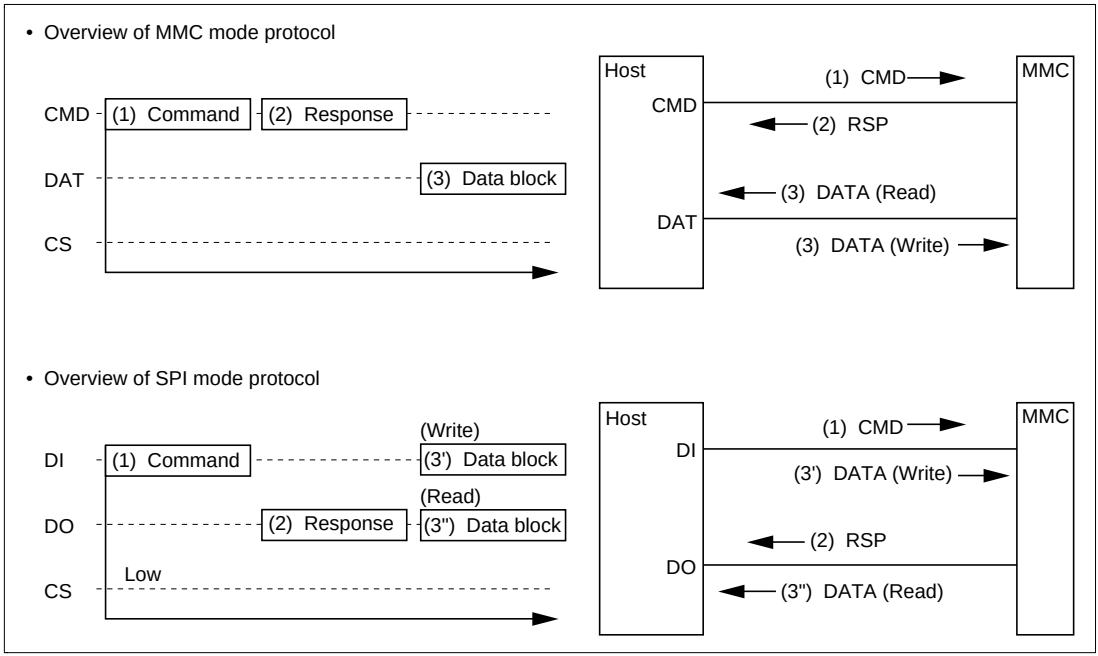


Figure 2.3 Overview of MultiMediaCard™ Protocols

2.2.4 Transfer Modes

Figure 2.4 shows the MultiMediaCard™ transfer modes. There are three transfer modes: single block transfer, multiple block transfer, and stream transfer.

1. MMC mode

   MMC mode supports all three transfer methods: single block, multiple block, and stream. A default block size is stipulated according to the card; one block can be transferred in response to one command in single block transfer, and multiple blocks in multiple block transfer. The default block size for Hitachi MultiMediaCard™ is 512 bytes, but this can be changed to a size from 1 byte to 2048 bytes with the block size change command. In stream transfer there is no concept of a block: any number of bytes of data can be transferred in byte units from any address. However, since CRCs are not used, this method is not suitable for applications requiring data reliability, and since transfer is not performed sector by sector, it is also unsuitable for systems using a DOS-compatible FAT file system.

2. SPI mode

   Starting with MMCA Standard Ver. 3.1 (released June 2001), SPI mode supports single block transfer mode and multiple block transfer mode. The block size can be changed in the same way as in MMC mode, with a setting range of 1 byte to 2048 bytes in the case of Hitachi MultiMediaCard™.

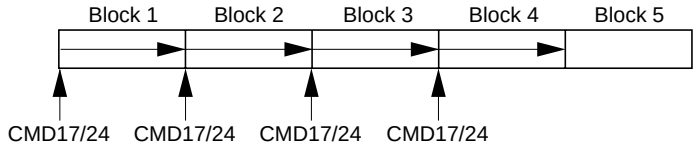
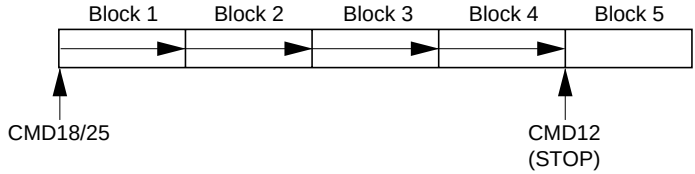
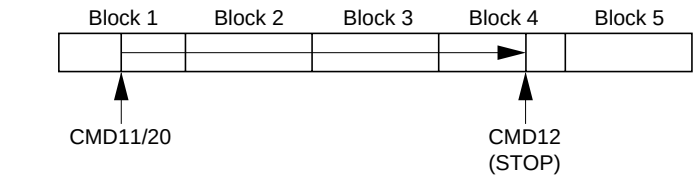
Transfer Mode	Applicable Mode	Transfer Image Diagram
(1) Single block Read/Write CMD17/CMD24	MMC/ SPI	
(2) Multiple block Read/Write CMD18/CMD25	MMC	
(3) Stream Read/Write CMD11/CMD20	MMC	

Figure 2.4 Overview of Transfer Modes

2.3 CRC

CRCs (cycle redundancy codes) are added to MultiMediaCard™ commands, responses, and data before they are transferred.

Checking the CRC enables erroneous transfer of data to be detected, and allows retransfer processing to be carried out on the system side when an error is detected, for example, resulting in a higher level of system reliability.

In MMC mode, CRCs are mandatory in transfer modes other than stream transfer. A 7-bit CRC is added to each command, and a 16-bit CRC to every 512 bytes of data.

CRCs are optional in SPI mode.

2.4 Registers

Table 2.1 lists the MultiMediaCard™ internal registers.

Table 2.1 MultiMediaCard™ Registers

Register	Name	Length	Description
OCR	Operation conditions register	32 bits	Defines the MMC operating voltage. 1 bit = 100 mV. 1: operable range; 0: non-operable range
CID	Card identification register	128 bits	Card-specific identification information. Manufacturer, OEM, individual card number, etc.
RCA	Relative card address register	16 bits	Used to set name for card identification by host. 0001h to FFFFh (MMC mode only)
CSD	Card specific data register	128 bits	Stores card-specific information. Corresponds to CIS of PCMCIA.
DSR	Driver stage register	16 bits	Used to set MMC bus drive capability. Optional. Not used in Hitachi MMCs.
	Status register	32 bits	Indicates MMC status and error occurrence conditions.

2.4.1 OCR (Operation Conditions Register)

OCR is a 32-bit register that defines the operable range of the card. Bits corresponding to the operable range are set to 1, with each bit representing a 100 mV step. This register also contains an R/B bit for polling whether or not card internal processing has finished in the card identification procedure. The contents of OCR are shown in table 2.2.

Table 2.2 OCR (Operation Conditions Register) Contents

OCR Bit Position	V_{DD} Voltage Window	Hitachi MMC	
31	Card power up status (R/B)	0: Busy 1: Ready	0 or 8
30 to 24	Reserved	0	
23	3.5 to 3.6	1*[1]	F
22	3.4 to 3.5	1	
21	3.3 to 3.4	1	
20	3.2 to 3.3	1	
19	3.1 to 3.2	1	F
18	3.0 to 3.1	1	
17	2.9 to 3.0	1	
16	2.8 to 2.9	1	
15	2.7 to 2.8	1	8
14	2.6 to 2.7	0*[2]	
13	2.5 to 2.6	0	
12	2.4 to 2.5	0	
11	2.3 to 2.4	0	0
10	2.2 to 2.3	0	
9	2.1 to 2.2	0	
8	2.0 to 2.1	0	
7 to 0	Reserved	0	—

Notes: 1. 1: Support voltage
2. 0: Not support voltage

2.4.2 CID (Card Identification Number: Card ID Register)

CID is a 128-bit register in which the serial number of the card is written by the card manufacturer. The CID number is unique to each MultiMediaCard™, and in MMC mode is used to assign a relative address for selecting a card.

2.4.3 RCA (Relative Card Address: Relative Card Address Register) —Used in MMC mode only—

RCA is a 16-bit register that is used to assign the address corresponding to a card. An address from 0001h to FFFFh can be assigned. After a card is selected using the address assigned by this register, reading, writing, erasing, or other processing is performed on that card.

2.4.4 CSD (Card Specific Data: Card Specific Data Register)

CSD is a 128-bit register that holds various kinds of card-related information, including the corresponding MMCA spec version, supported command classes, card capacity, access time, and transfer unit block length.

2.4.5 DSR (Driver Specific Data: Driver Stage Register)

DSR is a 16-bit register that allows the bus drive capability of the card to be set. This is an optional register, and is not supported by Hitachi cards.

2.4.6 Status Register

This is a 32-bit register that indicate the card status and error occurrence conditions.

2.5 Commands

2.5.1 Overview of Commands

All processing, including card identification, reading, writing, and erasing, is initiated by commands issued from the host to the card. The commands are listed in table 2.3.

Table 2.3 Commands Supported by MultiMediaCard™

CMD Index	Abbreviation	Function	MMC Mode			SPI Mode	
			Type	Argument	Resp.	Argument	Resp.
CMD0	GO_IDLE_STATE	MMC reset	bc	[31:0] Stuff bits	—	None	R1
CMD1	SEND_OP_COND	MMC R/B polling Operating voltage setting (MMC mode only)	bcr	[31:0] OCR without busy	R2	None	R1
CMD2	ALL_SEND_CID	CID transmission request	bcr	[31:0] Stuff bits	R3		
CMD3	SET_RELATIVE_ ADDR	RCA setting	ac	[31:16] RCA [15:0] Stuff bits	R1		
CMD4	SET_DSR	DSR setting	bc	[31:16] DSR [15:0] Stuff bits	—		
CMD7	SELECT/ DESELECT_CARD	Selection of MMC to be accessed	ac	[31:16] RCA [15:0] Stuff bits	R1b (only from the selected card)		
CMD9	SEND_CSD	CSD transmission request	ac	[31:16] RCA [15:0] Stuff bits	R2	None	R1
CMD10	SEND_CID	CID transmission request	ac	[31:16] RCA [15:0] Stuff bits	R2	None	R1
CMD11	READ_DAT_UNTIL_ STOP	Stream read	adtc	[31:0] Data address	R1		
CMD12	STOP_ TRANSMISSION	Read/write stop command	ac	[31:0] Stuff bits	R1b		
CMD13	SEND_STATUS	Card status request	ac	[31:16] RCA [15:0] Stuff bits	R1	None	R2
CMD15	GO_INACTIVE_ STATE	Places card in inactive state	ac	[31:16] RCA [15:0] Stuff bits	—		
CMD16	SET_BLOCKLEN	Read/write block length change	ac	[31:0] Block length	R1	[31:0] Block length	R1
CMD17	READ_SINGLE_ BLOCK	Single block read	adtc	[31:0] Data address	R1	[31:0] Data address	R1

CMD Index	Abbreviation	Function	MMC Mode			SPI Mode	
			Type	Argument	Resp.	Argument	Resp.
CMD18	READ_MULTIPLE_ BLOCK	Multiple block read	adtc	[31:0] Data address	R1		
CMD20	WRITE_DAT_UNTIL_ STOP	Stream write	adtc	[31:0] Data address	R1		
CMD24	WRITE_BLOCK	Single block write	adtc	[31:0] Data address	R1	[31:0] Data address	R1
CMD25	WRITE_MULTIPLE_ BLOCK	Multiple block write	adtc	[31:0] Data address	R1		
CMD26	PROGRAM_CID	CID programming	adtc	[31:0] Stuff bits	R1		
CMD27	PROGRAM_CSD	CSD programming	adtc	[31:0] Stuff bits	R1	None	R1
CMD28	SET_WRITE_PROT	Protection setting	ac	[31:0] Data address	R1b	[31:0] Data address	R1b
CMD29	CLR_WRITE_PROT	Protection clearing	ac	[31:0] Data address	R1b	[31:0] Data address	R1b
CMD30	SEND_WRITE_PROT	Protect bit status transmission	adtc	[31:0] Write protect data address	R1	[31:0] Write protect data address	R1
CMD32	TAG_SECTOR_ START	Sets first sector of area to be erased	ac	[31:0] Data address	R1	[31:0] Data address	R1
CMD33	TAG_SECTOR_END	Sets last sector of area to be erased	ac	[31:0] Data address	R1	[31:0] Data address	R1
CMD34	UNTAG_SECTOR_	Sets sector not to be erased within erase area	ac	[31:0] Data address	R1	[31:0] Data address	R1
CMD35	TAG_ERASE_ GROUP_START	Sets first group of area to be erased	ac	[31:0] Data address	R1	[31:0] Data address	R1
CMD36	TAG_ERASE_ GROUP_END	Sets last group of area to be erased	ac	[31:0] Data address	R1	[31:0] Data address	R1
CMD37	UNTAG_ERASE_ GROUP	Sets group not to be erased within erase area	ac	[31:0] Data address	R1	[31:0] Data address	R1
CMD38	ERASE	Erases selected area	ac	[31:0] Stuff bits	R1b	[31:0] Stuff bits	R1b
CMD42	LOCK_UNLOCK	Password setting	adtc	[31:0] Stuff bits	R1b	[31:0] Stuff bits	R1b
CMD58	READ_OCR	OCR (operating voltage) information read				None	R3
CMD59	CRC_ON_OFF	CRC specification on/off control				[31:1] Stuff bits [0:0] CRC option	R1

In MMC mode, the following four command types are defined.

1. Broadcast command (bc)
   Command issued from the CMD line to all cards on the bus. There is no response from the cards.

2. Broadcast command with response (bcr)
   Command issued from the CMD line to all cards on the bus. All the cards on the bus simultaneously return a response from the CMD line.
3. Addressed (point-to-point) command (ac)
   Command issued from the CMD line to a designated specific card on the bus. The designated specific command returns a response from the CMD line.

4. Addressed (point-to-point) data transfer command (adtc)
   Command issued from the CMD line to a designated specific card on the bus. After the designated specific command returns a response from the CMD line, data transfer is performed via the DAT line.

(1) Command Format

Table 2.4 shows the command format. A command consists of 48 bits (6 bytes), comprising a start bit (always 0), a transfer bit (always 1), a 6-bit command field, a 4-byte (32-bit) argument field, a 7-bit CRC field, and an end bit (always 1). The argument field contains the necessary information (card relative address, read address, write address, etc.) for issuing that command.

Table 2.4 Command Format

Serial data →

[47]	Command	[0]

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

(2) Card Command Classes

Commands are divided into a number of classes, such as basic commands, read commands, write commands, and erase commands. The command classes are listed in table 2.5. Of course, write and erase class commands cannot be used with ROM version MCCs since these cards do not permit write and erase operations. The CSD register described above contains the supported classes to enable the host to identify the kind of card, and must be read during the identification sequence to obtain this information.

Table 2.5 Card Command Classes

Command Class	Class Definition	Supported Commands	
		MMC Mode	SPI Mode
Class 0	Basic	0, 1, 2, 3, 4, 7, 9, 10, 12, 13, 15	0, 1, 9, 10, 13, 58, 59
Class 1	Stream read	11	Not supported
Class 2	Block read	16, 17, 18	16, 17
Class 3	Stream write	20	Not supported
Class 4	Block write	16, 24, 25, 26, 27	24, 27
Class 5	Erase	32, 33, 34, 35, 36, 37, 38	32, 33, 34, 35, 36, 37, 38
Class 6	Write protection	28, 29, 30	28, 29, 30
Class 7	Lock card	42	42
Class 8	Application specific	55, 56	55, 56
Class 9	I/O mode	39, 40	Not supported
Class 10, 11	Reserved		

Next, the method of use of each command will be described, together with points for attention, taking each command class in turn.

2.5.2 MMC Mode Commands

(1) Basic Commands (Class 0)

Table 2.6 shows the commands in command class 0. This class contains basic commands used for the identification sequence and for acquiring internal information.

Table 2.6 Basic Commands (Class 0)

Command Index	Type	Argument	Resp	Abbrevlation	Command Description
CMD0	bc	[31:0] Stuff bits	None	GO_IDLE_STATE	Command for resetting all cards except cards in the Ina state to the Idle state. If CMD0 is issued while the CS signal is low, the selected card enters SPI mode.
CMD1	bcr	[31:0] OCR without busy	R3	SEND_OP_COND	Used to check the operable voltage range of the card used, and to check whether card internal processing has finished. When CMD1 is issued to a card, the card returns an R3 type response that includes the value of the operation conditions register (OCR). Individual bits in OCR correspond to a 100 mV range (e.g. 3.2 V to 3.3 V), and when set to 1 indicate that the card can operate in that voltage range. In addition, the most significant bit is used for Ready/Busy polling to determine whether card internal processing has finished.
CMD2	bcr	[31:0] Stuff bits	R2	ALL_SEND_CID	When CMD2 is issued, all cards in the Ready state simultaneously start transmitting one bit of CID information each to the CMD line. The CMD line has an open-drain architecture, and each card compares the CMD line bus state with its own CID value every transfer bit, and if they differ halts CID transmission at that point and returns to the Ready state. As a result, eventually the card with the lowest CID value completes transmission of its own CID value to the end, and that single card is selected and goes to the Identification state.

Command Index	Type	Argument	Resp	Abbrevlation	Command Description
CMD3	ac	[31:16] RCA [15:0] Stuff bits	R1	SET_RELATIVE_ ADDR	Sets the relative card address (RCA) for the card that has been placed in the Identification state by means of CMD2. The card for which RCA is set goes to the Standby state, and makes no response to subsequent CMD2 or CMD3 commands.
CMD4	bc	[31:16] DSR [15:0] Stuff bits	None	SET_DSR	Used to program the driver state register (DSR). Hitachi MMCs do not support DSR.
CMD7	ac	[31:16] RCA [15:0] Stuff bits	R1 (select card only)	SELECT/ DESELECT_ CARD	Selects a card and switches it from the Standby state to the Transfer state. Only one card can be placed in the Transfer state, and that card alone responds to subsequent read, write, erase, or other commands. If a card is selected for the Transfer state when another card has already been selected, the newly selected card goes to the Transfer state and the previously selected card returns to the Standby state. Issuing CMD7 when RCA = 0000h is reserved for returning all cards selected for the Transfer state to the Standby state.
CMD9	ac	[31:16] RCA [15:0] Stuff bits	R2	SEND_CSD	Used to read CSD information for the card specified by RCA.
CMD10	ac	[31:16] RCA [15:0] Stuff bits	R2	SEND_CID	Used to read CID information for the card specified by RCA.
CMD12	ac	[31:0] Stuff bits	R1b	STOP_ TRANSMISSION	Command for forcibly stopping stream read, multiple read, stream write, and multiple write command processing.

Command Index	Type	Argument	Resp	Abbrevlation	Command Description
CMD13	ac	[31:16] RCA [15:0] Stuff bits	R1	SEND_STATUS	Command that requests transmission of the status register contents to the card whose address is specified. The status register holds details of errors that have occurred in the card, the card status, and so forth.
CMD15	ac	[31:16] RCA [15:0] Stuff bits	None	GO_INACTIVE_ STATE	Command for placing the card in the Inactive state.

(2) Stream Read Command (Class 1)

Table 2.7 shows the command in command class 1. This class contains the stream read command.

Table 2.7 Stream Read Command (Class 1)

Command Index	Type	Argument	Resp	Abbrevlation	Command Description
CMD11	adtc	[31:0] Data address	R1	READ_DAT_ UNTIL_STOP	Command for reading a data stream starting from the given address from the card until a STOP_TRANSMISSION command is transmitted.

(3) Block Read Commands (Class 2)

Table 2.8 shows the commands in command class 2. This class contains commands relating to block reads.

Table 2.8 Block Read Commands (Class 2)

Command Index	Type	Argument	Resp	Abbrevlation	Command Description
CMD16	ac	[31:0] Block length	R1	SET_BLOCKLEN	Command used to change the transfer block length in subsequent CMD17 (single block read), CMD18 (multiple block read), CMD24 (single block write), and CMD25 (multiple block write) commands.
CMD17	adtc	[31:0] Data address	R1	READ_SINGLE_ BLOCK	Command for reading data from the card, starting at the address specified by the argument and with the block length set by CMD16 (or default length of 512 bytes if no setting is made).
CMD18	adtc	[31:0] Data address	R1	READ_MULTIPLE_ BLOCK	Command for consecutively reading multiple blocks of data from the card, starting at the address specified by the argument and with the block length set by CMD16 (or default length of 512 bytes if no setting is made), until a stop command (CMD12) is input.

- CMD16

  With Hitachi MMCs, the transfer block size can be changed in bytes units within a range of 1 to 2048 bytes, using CMD16.

  Example: Argument = 00 00 00 20h → 32-byte unit transfer

  Once the block size has been changed, the new block size remains valid until changed again with CMD16 or reset with CMD0. When this command is not used, data transfer is performed using the default block size of 512 bytes.

  As READ_BLK_PARTIAL = 1 is set as a CSD register value in Hitachi MMCs, if the transfer block size is changed using CMD16, when a CMD17 or CMD18 block read command is used, data can be read using the new block size.

  However, as WRITE_BLK_PARTIAL = 0 is set as a CSD register value in Hitachi MMCs, even if the transfer block size is changed using CMD16, when a CMD24 or CMD25 block write command is used, data is written using a fixed 512-byte block size, not the new block size.

- CMD17

  With Hitachi MMCs, READ_BLK_MISALIGN = 0 is set in CSD, and so a read data block cannot exceed a physical block boundary line. Start address and block size changes must be carried out so that the read data range does not cross a physical boundary. Physical block boundaries are located every 0800h addresses (every 2048 bytes) starting from address 0000h. If an attempt is made to read block data that runs over a physical block boundary, an error bit will be set in the response. Figure 2.5 shows sample start address settings.

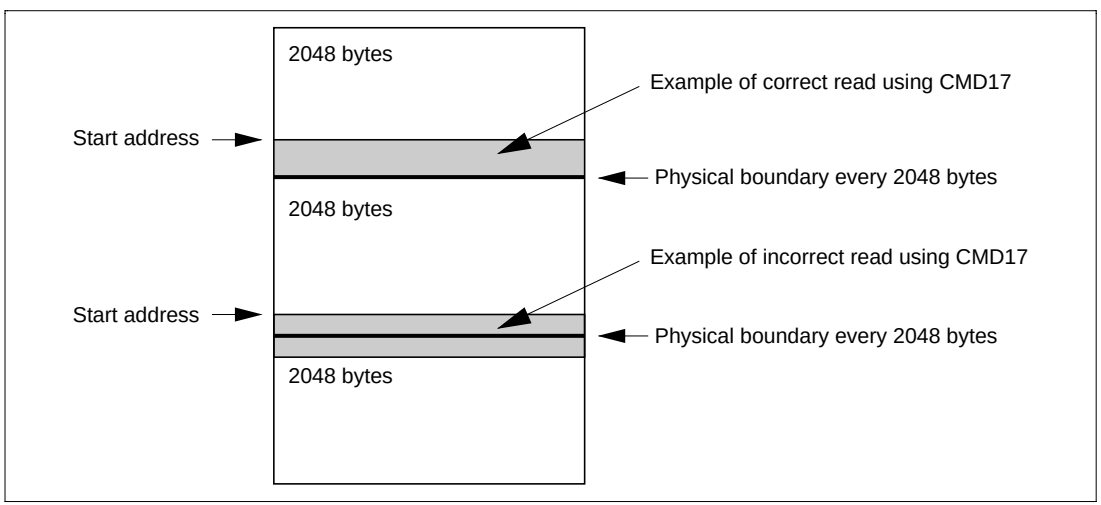


Figure 2.5 Sample Start Address Settings When Using CMD17

- CMD18

  As with CMD17, since READ_BLK_MISALIGN = 0 is set in CSD in Hitachi MMCs, an individual data block in a multiple block read cannot exceed a physical block boundary line. Start address and block size changes must be carried out so that individual data blocks do not cross a physical boundary. Physical block boundaries are located every 0800h addresses (every 2048 bytes) starting from address 0000h. If an attempt is made to read block data that runs over a physical block boundary, an error bit will be set in the response. Figure 2.6 shows sample start address settings.

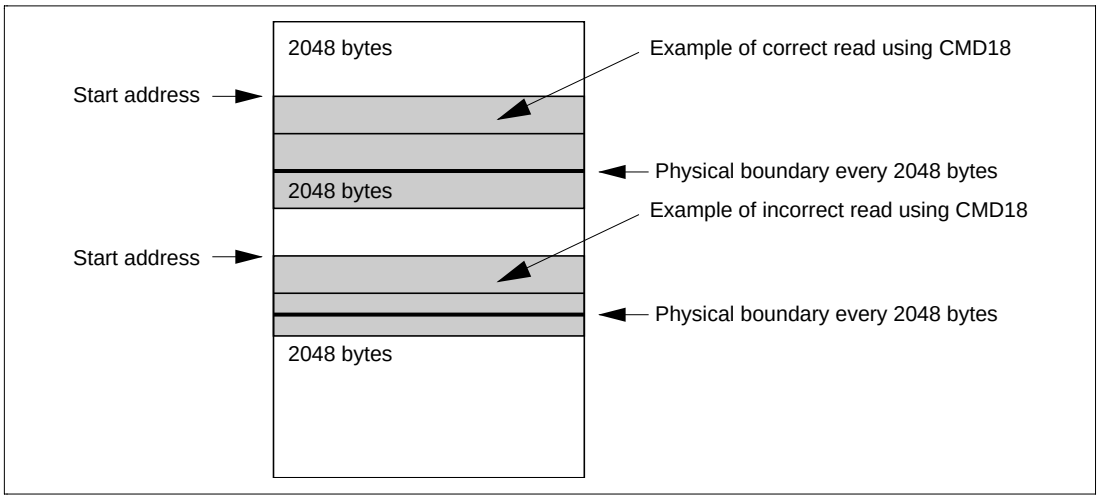


Figure 2.6 Sample Start Address Settings When Using CMD18

(4) Stream Write Command (Class 3)

Table 2.9 shows the command in command class 3. This class contains the stream write command.

Table 2.9 Stream Write Command (Class 3)

Command Index	Type	Argument	Resp	Abbrevlation	Command Description
CMD20	adtc	[31:0] Data address	R1	WRITE_DAT_ UNTIL_STOP	Writes a data stream starting from the given address to the card until a STOP_TRANSMISSION command (CMD12) is issued.

(5) Block Write Commands (Class 4)

Table 2.10 shows the commands in command class 4. This class contains commands relating to block writes.

Table 2.10 Block Write Commands (Class 4)

Command Index	Type	Argument	Resp	Abbrevlation	Command Description
CMD24	adtc	[31:0] Data address	R1	WRITE_BLOCK	Command for writing data to the card, starting at the address specified by the argument and with the block length set by CMD16 (or default length of 512 bytes if no setting is made). Hitachi MMCs use a fixed block length of 512 bytes.
CMD25	adtc	[31:0] Data address	R1	WRITE_ MULTIPLE_BLOCK	Command for consecutively writing multiple blocks of data to the card, starting at the address specified by the argument and with the block length set by CMD16 (or default length of 512 bytes if no setting is made), until a stop command (CMD12) is input. Hitachi MMCs use a fixed block length of 512 bytes.
CMD26	adtc	[31:0] Stuff bits	R1	PROGRAM_CID	Command used for CID register programming. This command can only be used once, to write the CID prior to shipment by the manufacturer, and cannot be used by the MMC host.
CMD27	adtc	[31:0] Stuff bits	R1	PROGRAM_CSD	Command used for programmable bit programming in the CSD register.

- CMD24

  WRITE_BLK_PARTIAL = 0 is set in CSD in Hitachi MMCs, and so the block length is fixed at 512 bytes.

  With Hitachi MMCs, WRITE_BLK_MISALIGN = 0 is set in CSD, and so a write data block cannot exceed a physical block boundary line. The start address setting must be made so that the write data range does not cross a physical boundary. Physical block boundaries are located every 0800h addresses (every 2048 bytes) starting from address 0000h. If an attempt is made to write block data that runs over a physical block boundary, an error bit will be set in the response. Figure 2.7 shows sample start address settings.

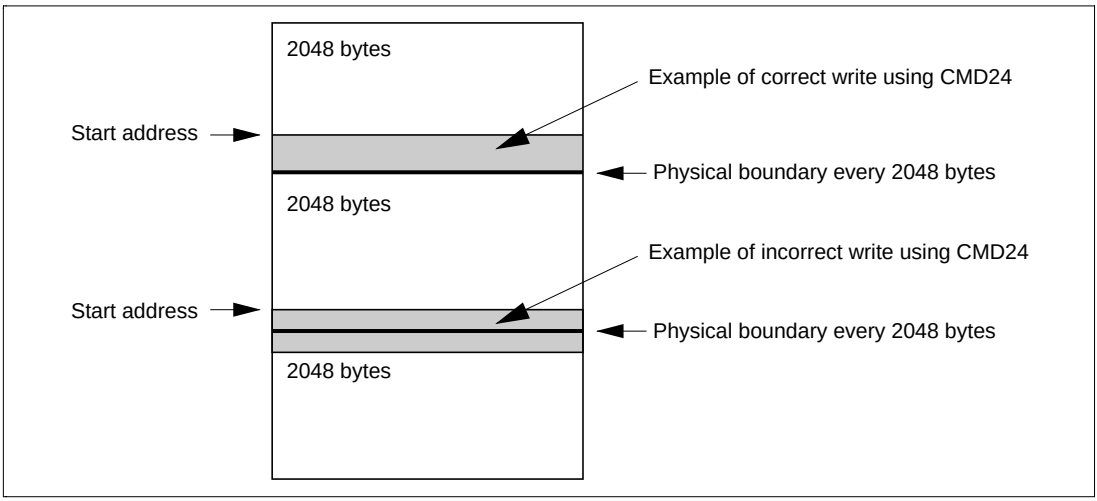


Figure 2.7 Sample Start Address Settings When Using CMD24

- CMD25

  As with CMD24, since WRITE_BLK_MISALIGN = 0 is set in CSD in Hitachi MMCs, an individual data block in a multiple block write cannot exceed a physical block boundary line. The start address setting must be made so that individual data blocks do not cross a physical boundary. Physical block boundaries are located every 0800h addresses (every 2048 bytes) starting from address 0000h. If an attempt is made to write block data that runs over a physical block boundary, an error bit will be set in the response. Figure 2.8 shows sample start address settings.

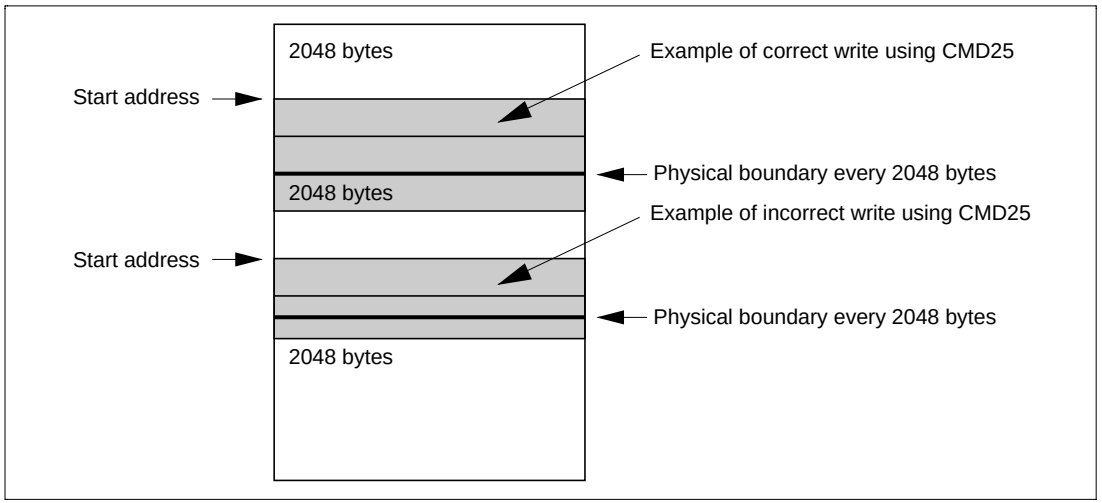


Figure 2.8 Sample Start Address Settings When Using CMD25

In the internal structure of a Hitachi MMC, a 512-byte × 4-stage buffer and 2-kbyte flash comprise a single unit. Actual writes to the flash are performed in 2-kbyte units, and data up to that point is temporarily stored in the internal buffer. Therefore, when using multiple blocks, the write time (Busy period) for each block is short when performing a buffer write, and the Busy period becomes longer when writes to flash occurs every 2 kbytes. Figure 2.9 shows guidelines for the Busy time for each transfer block.

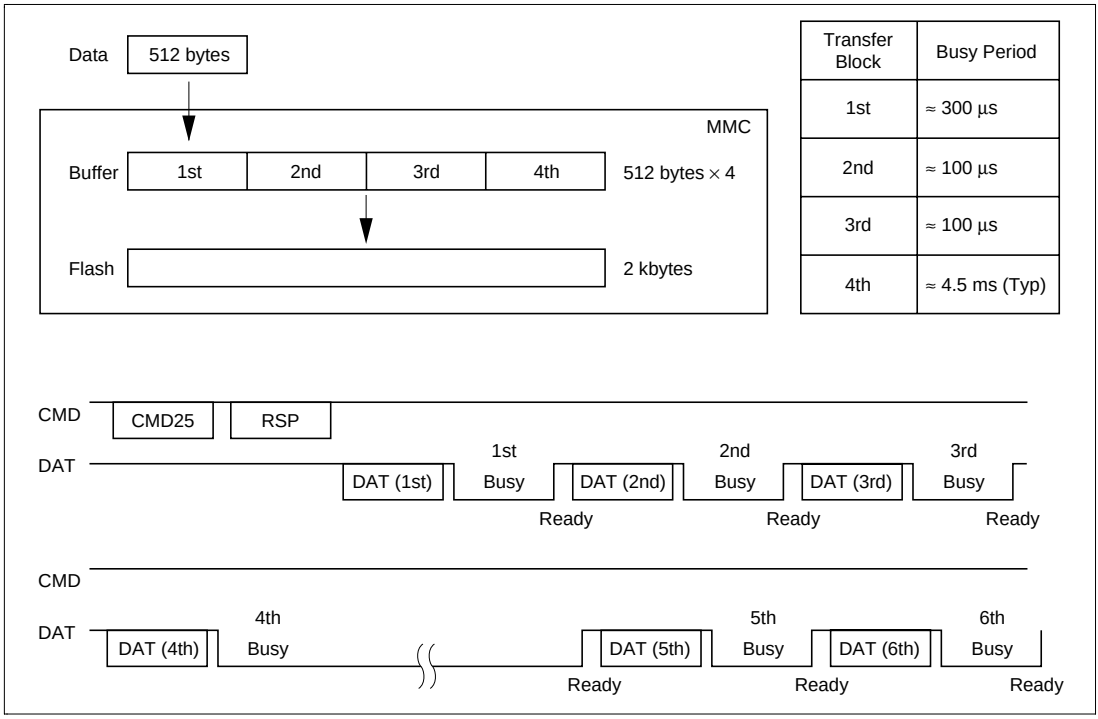


Figure 2.9 Concept of Busy Time for Each Transfer Block

In Hitachi MMCs, there are flash write physical boundaries at 2048-byte intervals (addresses 0000h, 0800h, 1000h, 1800h ...), and writes are performed in these units. Therefore, when writing 2048 bytes of data comprising four 512-byte units using a multiple block write (CMD25) as shown in figure 2.10, processing for writing to flash from the internal buffer is performed once when writing from an address (0000h) whereby the data is within a 2048-byte physical boundary, as in (A), and twice when the write runs over a physical boundary, as in (B). To achieve higher speed, therefore, data writes should be executed from addresses that take account of 2048-byte physical boundaries (0000h, 0800h, 1000h, 1800h ...).

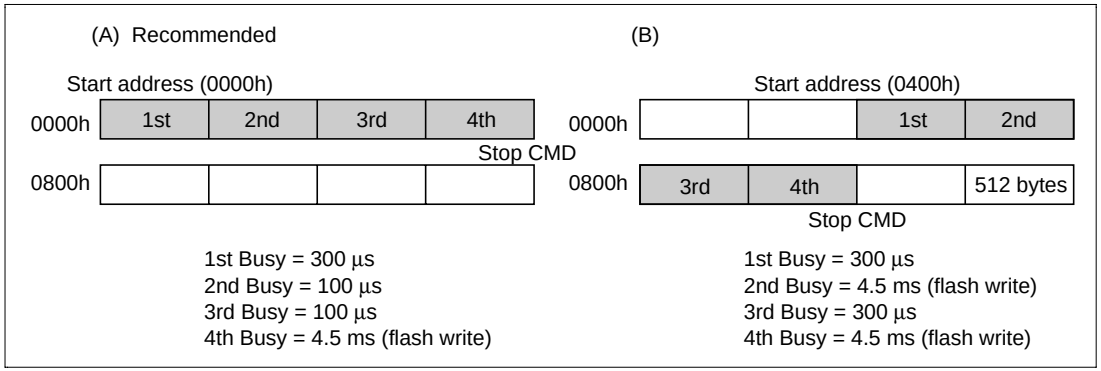


Figure 2.10 Example of Recommended Multiple Block Start Address Setting

When using a multiple block write, as mentioned earlier, the Busy time differs greatly according to whether only buffer write processing is performed or flash writes are also executed. After data transfer the DAT line goes to the Busy state, and the host carries out polling until DAT goes to the Ready state again. If this polling interval is coordinated with the Busy period when flash writes occur, there will be wasted time. Polling should therefore be performed at short intervals coordinated with the Busy period when internal buffer writes are executed.

(6) Erase Commands (Class 5)

Table 2.11 shows the commands in command class 5. This class contains commands relating to erasing.

Table 2.11 Erase Commands (Class 5)

Command Index	Type	Argument	Resp	Abbrevlation	Command Description
CMD32	ac	[31:0] Data address	R1	TAG_SECTOR_ START	Sets the address of the first sector in the range within which erasing is to be performed consecutively within the erase group.
CMD33	ac	[31:0] Data address	R1	TAG_SECTOR_ END	Sets the address of the last sector in the range within which erasing is to be performed consecutively within the erase group. Can be the same sector as set by CMD32.
CMD34	ac	[31:0] Data address	R1	UNTAG_SECTOR	Clears erasing of any sector in the range within which erasing is to be performed consecutively within the erase group.
CMD35	ac	[31:0] Data address	R1	TAG_ERASE_ GROUP_START	Sets the address of the first erase group in the range within which erasing is to be performed consecutively.
CMD36	ac	[31:0] Data address	R1	TAG_ERASE_ GROUP_END	Sets the address of the last erase group in the range within which erasing is to be performed consecutively. Can be the same erase group as set by CMD35.
CMD37	ac	[31:0] Data address	R1	UNTAG_ERASE_ GROUP	Clears erasing of one of the erase groups on which erasing is to be performed consecutively.
CMD38	ac	[31:0] Stuff bits	R1b	ERASE	Executes erasing within the set range.

To execute erasing in sector units, commands are issued in the sequence shown in figure 2.11. The commands must be issued in the order shown. With Hitachi MMCs, an erase group size of 8 kbytes is set in CSD. Sector erases cannot run across erase groups. When executing erasing by erase group unit, commands are issued in the sequence shown in figure 2.12. As with sector erasing, the commands must be issued in the order shown.

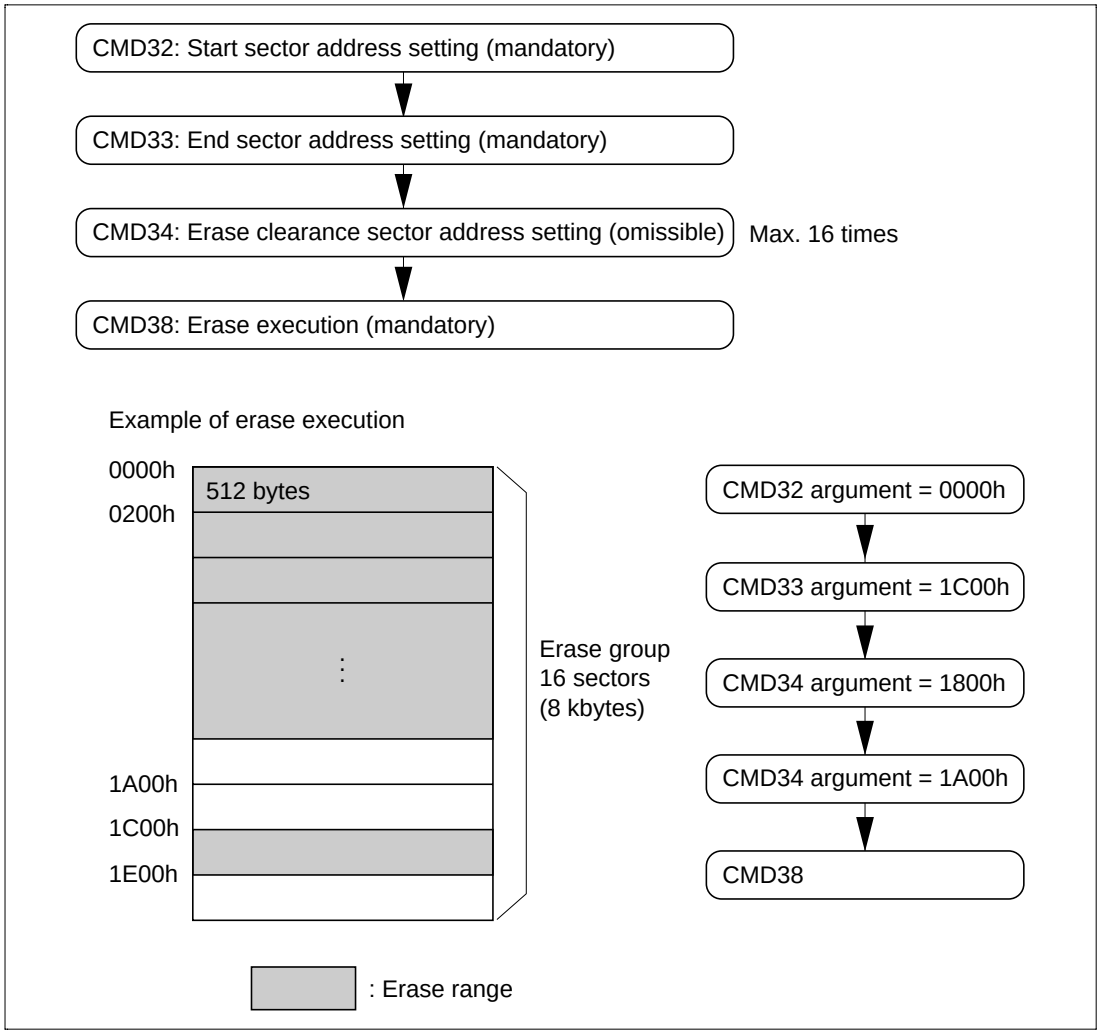


Figure 2.11 Sector-Unit Erase Procedure

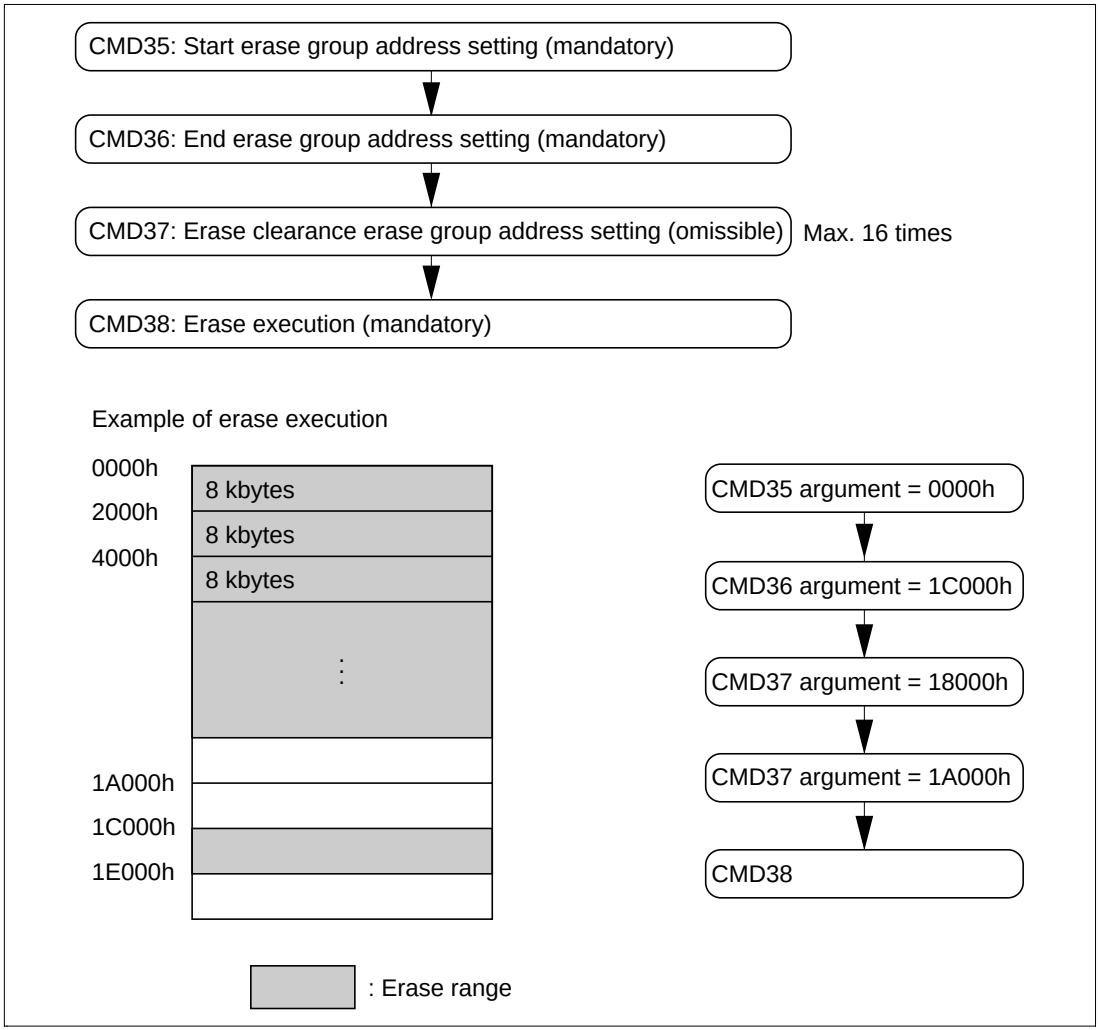


Figure 2.12 Block-Unit Erase Procedure

(7) Write Protect Commands (Class 6)

Table 2.12 shows the commands in command class 6. This class contains commands relating to write protection

Table 2.12 Write Protect Commands (Class 6)

Command Index	Type	Argument	Resp	Abbrevlation	Command Description
CMD28	ac	[31:0] Data address	R1b	SET_WRITE_ PROT	Sets the write protect bit of the group whose address is specified when the card has a write protect function.
CMD29	ac	[31:0] Data address	R1	CLR_WRITE_ PROT	Clears the write protect bit of the group whose address is specified when the card has a write protect function.
CMD30	adtc	[31:0] Write protect data address	R1	SEND_WRITE_ PROT	Requests transmission of the write protect bit status to the card when the card has a write protect function.

In Hitachi MMCs, WP_GRP_SIZE = 1 is set in CSD. The write protect group size is stipulated by

$$(1 + \text{WP\_GRP\_SIZE}) \times \text{ERASE\_GRP\_SIZE}$$

and so is 16 kbytes.

Therefore, write protection can be set using CMD28 and cleared using CMD29 in 16-kbyte units. Figure 2.13 shows an example of a write protect block. When CMD28 is issued with argument = 4000h, the area shown in the figure is write-protected.

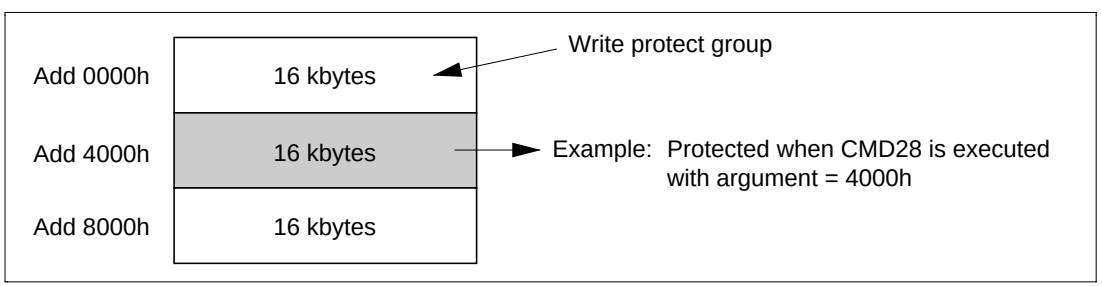


Figure 2.13 Example of Write Protect Group

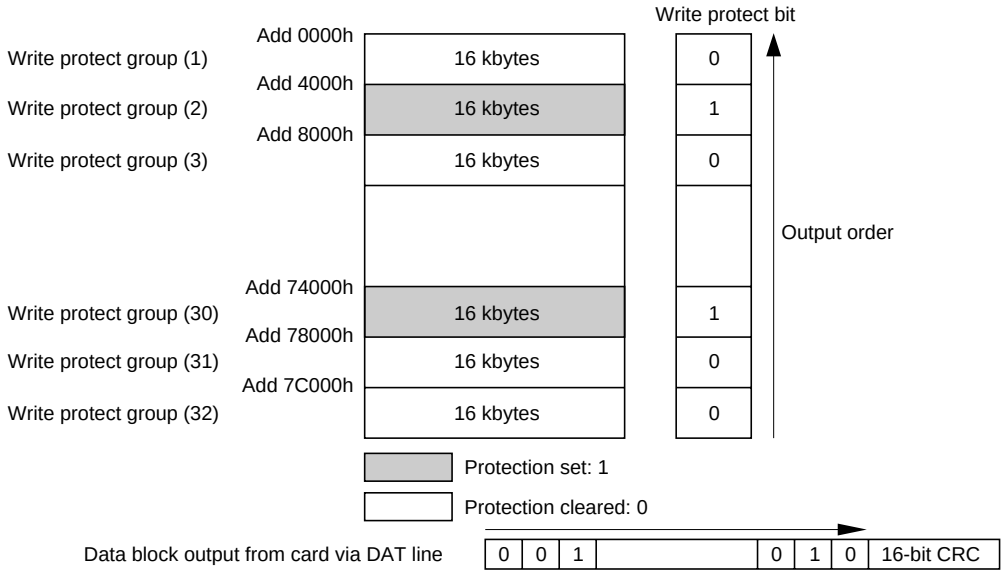
Erase and write protection can be applied to the entire card by setting either PERM_WRITE_PROTECT or TMP_WRITE_PROTECT to 1 in the CSD register. (Both are cleared to 0 by default.)

PERM_WRITE_PROTECT can only be set once by the customer, and once set, the card is permanently protected against erasing and writing. TMP_WRITE_PROTECT, on the other hand, enables protection to be applied temporarily, and can be set and cleared any number of times.

The status of the write protect bit can be read using CMD30 (send write protection). Figure 2.14 shows an example of transmission of the write protect bit status. Normally, the readable range is 32 protect groups from the specified address, and the status of the protect group at the specified address is output as the final data, using the same method as for a single read.

If there are not 32 write protect groups after the write protect group corresponding to the specified address, 0 is output for the write protect bits of nonexistent write protect groups.

- When there are 32 write protect groups
  Example: CMD30 issued with 0000h specified as write protect data address.
  → Write protect bit statuses for 32 groups are output from write protect group 0000h.



- When there are not 32 write protect groups
  → Write protect bits for nonexistent write protect groups are output as 0.

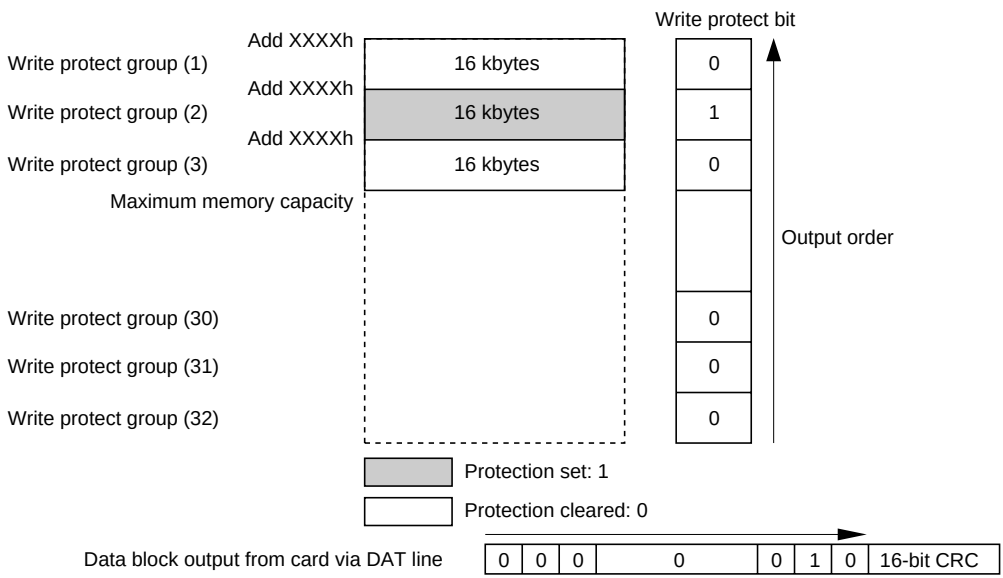


Figure 2.14 Examples of Write Protect Bit Status Transmission

(8) Lock Card Command (Class 7)

Table 2.13 shows the command in command class 7. This class contains a command for locking the card by means of a password.

Table 2.13 Lock Card Command (Class 7)

Command Index	Type	Argument	Resp	Abbrevlation	Command Description
CMD42	adtc	[31:0] Stuff bits	R1b	LOCK_UNLOCK	Used for password setting/ erasing or card locking/ unlocking. When a card has been locked by means of a password, data cannot be read or written until the card is unlocked. A password of up to 128 bits can be set.

A card can be locked by using a protection function implemented with a password. A locked card returns a response only to basic command class (class 0) and lock card command class (class 7) commands. That is, only initialization, identification, selection, status inquiry, and lock-related commands can be used, and card data cannot be read, written, or erased. Password information and lock information is nonvolatile, and is not lost when power is turned on again.

Table 2.14 shows the register used by the card lock function.

Table 2.14 Register Used for Card Locking

Byte#	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Reserved				ERASE	LOCK_ UNLOCK	CLR_ PWD	SET_ PWD
1	PWD_LEN							
… PWD_LEN + 1	Password data (Max 128 bits)							

Notes: ERASE: Set when all data in the card is to be forcibly erased.
1: Forcible erase executed

LOCK_UNLOCK: Set when locking or unlocking the card.
1: Card is locked
0: Card is unlocked

CLR_PWD: Set when clearing the password.
1: Password cleared

SET_PWD: Set when setting a new password.
1: New password is set in PWD register

PWD_LEN: Password length (in bytes)

PWD: Password data

Setting a New Password:

1. If the card for which the password is to be set is deselected, perform card selection using CMD7.
2. Decide the length of the password to be set (between 1 and 16 bytes, in byte units).
3. Change the block length using CMD16 to allow data transfer of (decided password byte length + 2 bytes).
4. Transmit the card lock/unlock command (CMD42) and a data block.

   If the LOCK_UNLOCK bit of the data block is set to 1 when transmission is performed, the card is locked immediately after this CMD42 command is executed. Even if the LOCK_UNLOCK bit is cleared to 0, the card is locked by cutting the card power supply and then turning it on again. This is because the contents of PWD_LEN indicate whether or not a password is currently set, and if the PWD_LEN value is not 0, the card is automatically locked after a power-on operation.
5. The card saves the password length (PWD_LEN) and the password (PWD).

Table 2.15 Data Block Sent to Card When Setting New Password

Byte#	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Reserved				0	LOCK_ UNLOCK	0	1
1	Byte length of password to be set (PWD_LEN)							
… PWD_LEN + 1	New password to be set (PWD)							

Changing the Password:

1. If the card for which the password is to be set is deselected, perform card selection using CMD7.
2. Decide the length of the new password (between 1 and 16 bytes, in byte units).
3. Change the block length using CMD16 to allow data transfer of (old password byte length + new password byte length + 2 bytes).
4. Transmit the card lock/unlock command (CMD42) and a data block.

   If the LOCK_UNLOCK bit of the data block is set to 1 when transmission is performed, the card is locked immediately after this CMD42 command is executed. Even if the LOCK_UNLOCK bit is cleared to 0, the card is locked by cutting the card power supply and then turning it on again. This is because the contents of PWD_LEN indicate whether or not a password is currently set, and if the PWD_LEN value is not 0, the card is automatically locked after a power-on operation.

Table 2.16 Data Block Sent to Card When Changing Password

Byte#	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Reserved				0	LOCK_ UNLOCK	0	1
1	Byte length of new password (PWD_LEN)							
… PWD_LEN + 1	Old password + New password							

Clearing the Password:

1. If the card for which the password is to be cleared is deselected, perform card selection using CMD7.
2. Change the block length using CMD16 to allow data transfer of (set password byte length + 2 bytes).
3. Transmit the card lock/unlock command (CMD42) and a data block.
4. If the transmitted password byte length and password match the values stored in the register, the password is erased and the password length (PWD_LEN) is cleared to 0. The password length of 0 in this case indicates that a password is not set.

Table 2.17 Data Block Transferred When Clearing Password

Byte#	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Reserved				0	X	1	0
1	Set password byte length (PWD_LEN)							
… PWD_LEN + 1	Set password (PWD)							

Locking a Card:

1. If the card to be locked is deselected, perform card selection using CMD7.
2. Change the block length using CMD16 to allow data transfer of (set password byte length + 2 bytes).
3. Transmit the card lock/unlock command (CMD42) and a data block.
4. If the transmitted password byte length and password match the values stored in the register, the card is locked and CARD_IS_LOCKED = 1 is returned as the response to the command.

Table 2.18 Data Block Transferred When Locking Card

Byte#	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Reserved				0	1	0	0
1	Set password byte length (PWD_LEN)							
… PWD_LEN + 1	Set password (PWD)							

Unlocking a Card:

1. If the card to be unlocked is deselected, perform card selection using CMD7.
2. Change the block length using CMD16 to allow data transfer of (set password byte length + 2 bytes).
3. Transmit the card lock/unlock command (CMD42) and a data block.
4. If the transmitted password byte length and password match the values stored in the register, the card is unlocked and CARD_IS_LOCKED = 0 is returned as the response to the command.

Note: The card unlocked state is maintained only as long as power to the card is on. If the card power supply is cut and then turned on again, the card will again be locked automatically. To permanently unlock a card, the password must be erased.

Table 2.19 Data Block Transferred When Unlocking Card

Byte#	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Reserved				0	0	0	0
1	Set password byte length (PWD_LEN)							
… PWD_LEN + 1	Set password (PWD)							

Forcibly Erasing Data: If the user forgets the card password information, the password contents can be erased together with all the data in the card.

1. If the card to be forcibly erased is deselected, perform card selection using CMD7.
2. Change the transfer block length to 1 byte using CMD16 to allow data transfer of 1 byte.
3. Transmit the card lock/unlock command (CMD42) and a data block.
4. All the card data areas, including the password length register and password register, are erased, and if the card was locked it is unlocked.

Table 2.20 Data Block Transferred When Forcibly Erasing Data

Byte#	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Reserved				1	0	0	0

2.5.3 SPI Mode Commands

SPI commands are listed in table 2.21. The main differences from MMC mode commands are that, since the OCR register value cannot be read with CMD1, CMD58 is assigned for OCR reads, and since CRC is off by default in SPI mode, CMD59 is assigned for turning CRC on. With other commands, although the response format differs, command functions and issuance conditions are the same as in MMC mode.

Table 2.21 SPI Mode Commands

Command Index	SPI Mode	Argument	Resp	Abbrevlation	Command Description
CMD0	Yes	None	R1	GO_IDLE_STATE	Command for a card reset. If CMD0 is issued while the CS signal is low, the selected card enters SPI mode. After issuing CMD0, the only commands that can be issued are CMD1 and CMD58.
CMD1	Yes	None	R1	SEND_OP_COND	Used to monitor completion of card initialization. The host should continue polling with CMD1 until response R1 changes from 01h (Busy) to 00h (Ready). As there is no power supply voltage range exchange such as that by CMD1 in MMC mode, CMD58 (Read_OCR) should be used to check the power supply voltage range, if wished.
CMD2	No				
CMD3	No				
CMD4	No				
CMD5	Reserved				
CMD6	Reserved				
CMD7	No				
CMD8	Reserved				
CMD9	Yes	None	R1	SEND_CSD	Used to read CSD information for the card specified by RCA.
CMD10	Yes	None	R1	SEND_CID	Used to read CID information for the card specified by RCA.
CMD11	No				
CMD12	No				

Command Index	SPI Mode	Argument	Resp	Abbrevlation	Command Description
CMD13	Yes	None	R2	SEND_STATUS	Command for ordering transmission to the selected card of the card's status register.
CMD14	Reserved				
CMD15	No				
CMD16	Yes	[31:0] Block length	R1	SET_BLOCKLEN	Command used to change the transfer block length in subsequent CMD17 (single block read) and CMD24 (single block write) commands. Hitachi MMCs use a fixed write transfer block size of 512 bytes. See the description of CMD16 in MMC mode for a sample setting when issuing this command.
CMD17	Yes	[31:0] Data address	R1	READ_SINGLE_ BLOCK	Command for reading data from the card, starting at the address specified by the argument and with the block length set by CMD16 (or default length of 512 bytes if no setting is made).
CMD18	No	[31:0] Data address	R1	READ_MULTIPLE_ BLOCK	Command for consecutively reading multiple blocks of data from the card, starting at the address specified by the argument and with the block length set by CMD16 (or default length of 512 bytes if no setting is made), until a stop command (CMD12) is input.
CMD19	Reserved				
CMD20	No				
CMD21 to CMD23	Reserved				
CMD24	Yes	[31:0] Data address	R1	WRITE_BLOCK	Command for writing data to the card, starting at the address specified by the argument and with the block length set by CMD16 (or default length of 512 bytes if no setting is made). Hitachi MMCs use a fixed block length of 512 bytes.

Command Index	SPI Mode	Argument	Resp	Abbrevlation	Command Description
CMD25	No	[31:0] Data address	R1	WRITE_ MULTIPLE_BLOCK	Command for consecutively writing multiple blocks of data to the card, starting at the address specified by the argument and with the block length set by CMD16 (or default length of 512 bytes if no setting is made), until a stop command (CMD12) is input. Hitachi MMCs use a fixed block length of 512 bytes. (Applies from the MMCA Standard Ver. 3.1 MM2 Series onward.)
CMD26	No				
CMD27	Yes	None	R1b	PROGRAM_CSD	Command used for programmable bit programming in the CSD register.
CMD28	Yes	[31:0] Data address	R1b	SET_WRITE_ PROT	Sets the write protect bit of the group whose address is specified when the card has a write protect function.
CMD29	Yes	[31:0] Data address	R1b	CLR_WRITE_ PROT	Clears the write protect bit of the group whose address is specified when the card has a write protect function.
CMD30	Yes	[31:0] Write protect data address	R1	SEND_WRITE_ PROT	Requests transmission of the write protect bit status to the card when the card has a write protect function.
CMD31	Reserved				
CMD32	Yes	[31:0] Data address	R1	TAG_SECTOR_ START	Sets the address of the first sector in the range within which erasing is to be performed consecutively within the erase group.
CMD33	Yes	[31:0] Data address	R1	TAG_SECTOR_ END	Sets the address of the last sector in the range within which erasing is to be performed consecutively within the erase group. Can be the same sector as set by CMD32.
CMD34	Yes	[31:0] Data address	R1	UNTAG_SECTOR	Clears erasing of any sector in the range within which erasing is to be performed consecutively within the erase group.

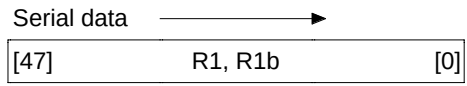
Command Index	SPI Mode	Argument	Resp	Abbrevlation	Command Description
CMD35	Yes	[31:0] Data address	R1	TAG_ERASE_ GROUP_START	Sets the address of the first erase group in the range within which erasing is to be performed consecutively.
CMD36	Yes	[31:0] Data address	R1	TAG_ERASE_ GROUP_END	Sets the address of the last erase group in the range within which erasing is to be performed consecutively. Can be the same erase group as set by CMD35.
CMD37	Yes	[31:0] Data address	R1	UNTAG_ERASE_ GROUP	Clears erasing of one of the erase groups on which erasing is to be performed consecutively.
CMD38	Yes	[31:0] Stuff bits	R1b	ERASE	Executes erasing within the set range.
CMD39	No				
CMD40	No				
CMD41	Reserved				
CMD42	Yes	[31:0] Stuff bits	R1b	LOCK_UNLOCK	Used for password setting/ erasing or card locking/ unlocking. When a card has been locked by means of a password, data cannot be read or written until the card is unlocked. A password of up to 128 bits can be set.
CMD43 to CMD57	Reserved				
CMD58	Yes	None	R3	READ_OCR	Command for reading the contents of the OCR register.
CMD59	Yes	[31:1] Stuff bits [0:0] CRC option	R1	CRC_ON_OFF	Command for turning the CRC option on and off. In SPI mode the CRC option is off by default, and is turned on using this command. • CRC option bit = 1 → CRC on • CRC option bit = 0 → CRC off
CMD60	No				

2.6 Responses

When a command is issued from the host to a card, on receiving the command the card returns a response to the host in a format stipulated for each command. These formats differ according to whether MMC mode or SPI mode is used for the interface. The response formats in MMC mode are shown in table 2.22, and those in SPI mode in table 2.23.

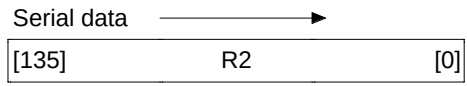
Table 2.22 Response Formats (MMC Mode)

R1, R1b (normal response commands: 48 bits)



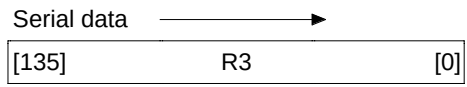
Bit position	47	46	[45:40]	[39:8]	[7:1]	0
Width (bit)	1	1	6	32	7	1
Value	0	0	X	X	X	1
Description	Start bit	Transmission bit	Command index	Card status	CRC7	End bit

R2 (CID and CSD register response commands: 136 bits)



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

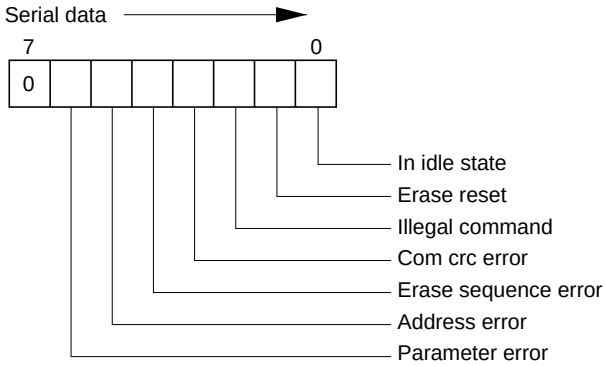
R3 (OCR register response commands: 136 bits)



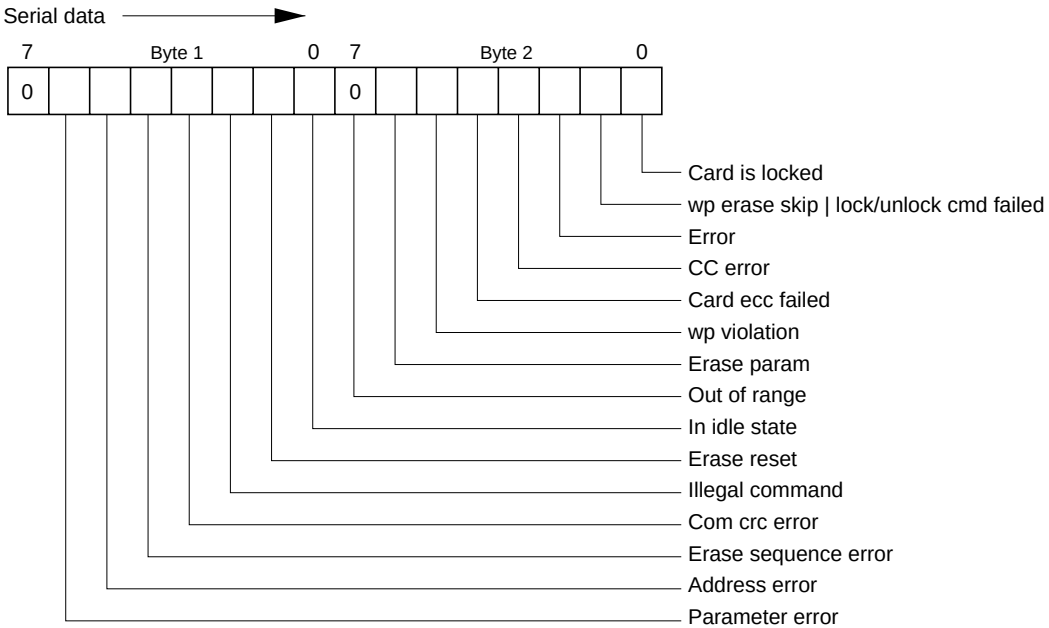
Bit position	47	46	[45:40]	[39:8]	[7:1]	0
Width (bit)	1	1	6	32	7	1
Value	0	0	111111	X	111111	1
Description	Start bit	Transmission bit	Reserved	OCR register	Reserved	End bit

Table 2.23 Response Formats (SPI Mode)

R1, R1b (response to all commands except SEND_STATUS command)



R2 (response to SEND_STATUS command: CMD13)



Note: SEND_STATUS command = CMD13

1. MMC mode

   In MMC mode there are five response types: R1, R2, R3, R4, and R5. The type of response returned for the various commands is as shown in table 2.16. The main command response is R1, which returns card status information. R2 returns a 136-bit response containing the CSD register value, CID register value, etc., when command 2, 9, or 10 is received, and R3 returns a 48-bit response containing the OCR register value when command 1 is received

2. SPI mode

   In SPI mode there are two response types: R1 and R2. An R2 type 16-bit response is returned when command 13 is received, and an R1 type 8-bit response is returned when any other command is received. The lower 8 bits of the R2 type 16-bit response are exactly the same as the R1 type.

2.7 Read/Write Protocols

Figures 2.15 and 2.16 show examples of the read and write protocols in MMC mode and SPI mode.

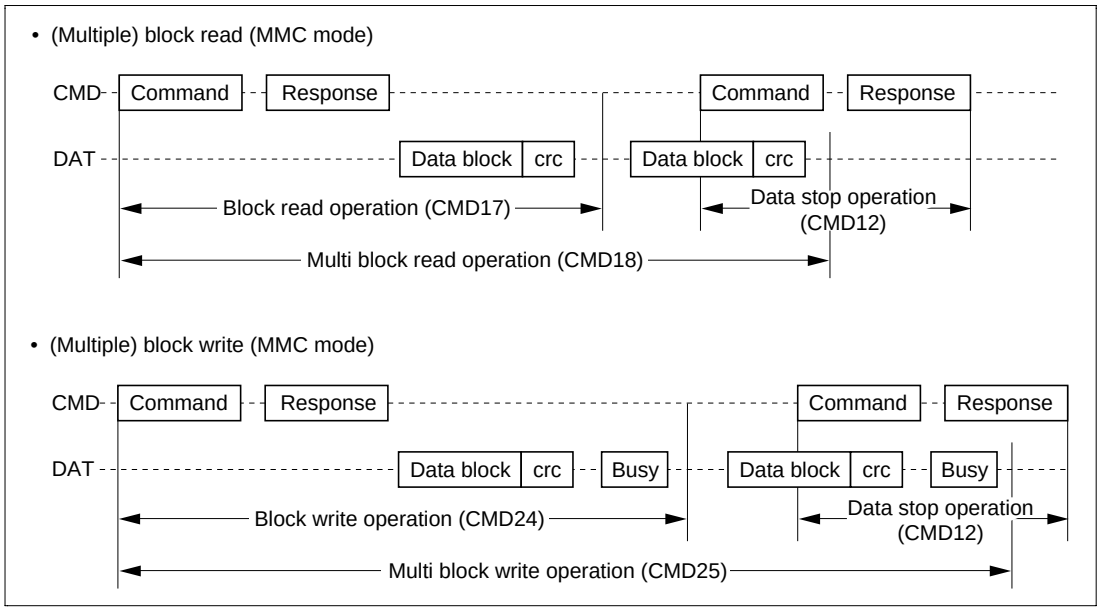


Figure 2.15 Examples of MMC Mode Read/Write Transfer

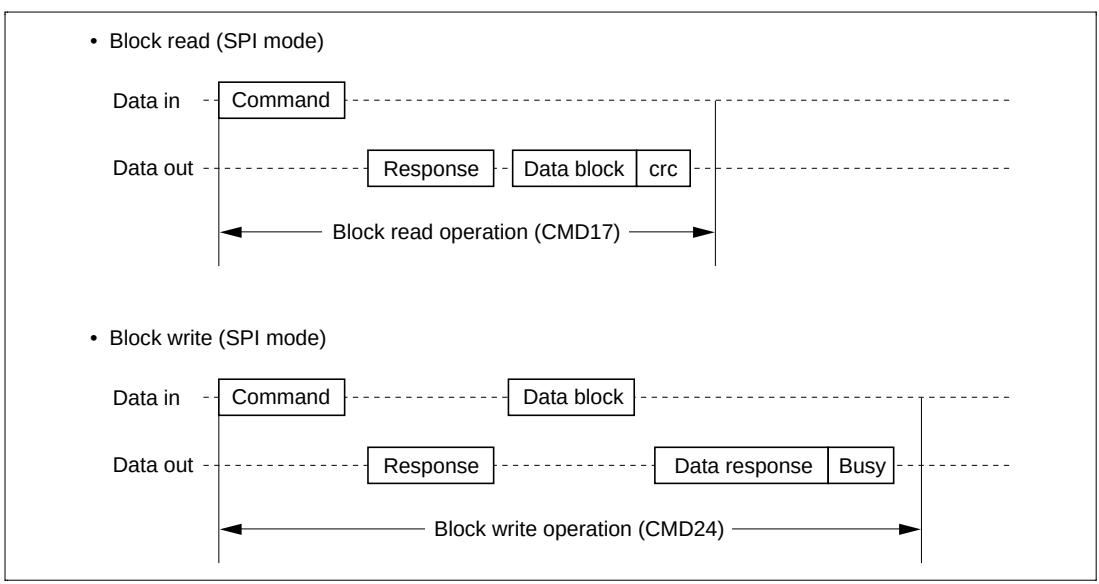


Figure 2.16 Examples of SPI Mode Read/Write Transfer

Section 3 Total System Support for Application Product Development

3.1 MultiMediaCard™ System Development

An MMC is not simply a card containing flash memory, but, with use in multimedia products in mind, also includes a controller for increasing the level of intelligence (generally through the use of a microcomputer) and a command interface for all requests to the card. These features provide a high level of user program independence in interfacing to the MMC, and simplify program development.

3.1.1 Support Policy

1. Easy MMC evaluation and examination

   When an MMC is used in an application product, a development platform is provided that facilitates evaluation and examination of the card's characteristics and functions. Evaluation can be carried out simply by connecting this platform to a personal computer, without use of the user system.

2. Easy construction of PC-compatible files on an MMC

   Image, voice, and character data is generally stored in a card in the form of files, and a file system is required for this purpose. Hitachi offers a PC-compatible file system, as well as MMC driver software that handles communication with the MMC via commands, etc., simplifying the construction of a file system.

3. MMC total system development support

   When developing a system that supports MMCs, it is important to be able to check the interface between the system and the card under various conditions. A protocol analyzer offering the following features is available to simplify this work.

   a. Monitoring of all signals between MMC and system
   b. Detection of various trigger events such as commands and data patterns during monitoring
   c. Chameleon modes for easier debugging
      - Normal mode

        Monitoring between system and card
      - Pseudo-card mode

        Pseudo system-side operation, debugging of interface on card side
      - Pseudo-host mode

        Pseudo card-side operation, debugging of interface on host side

d. Provision of interface LSI in adapter control system

The adapter control system includes MMC mode and SPI mode, with LSIs available in both modes to simplify system construction.

3.1.2 System Development Sequence

Figure 3.1 shows the support situation for development of a system using an MMC.

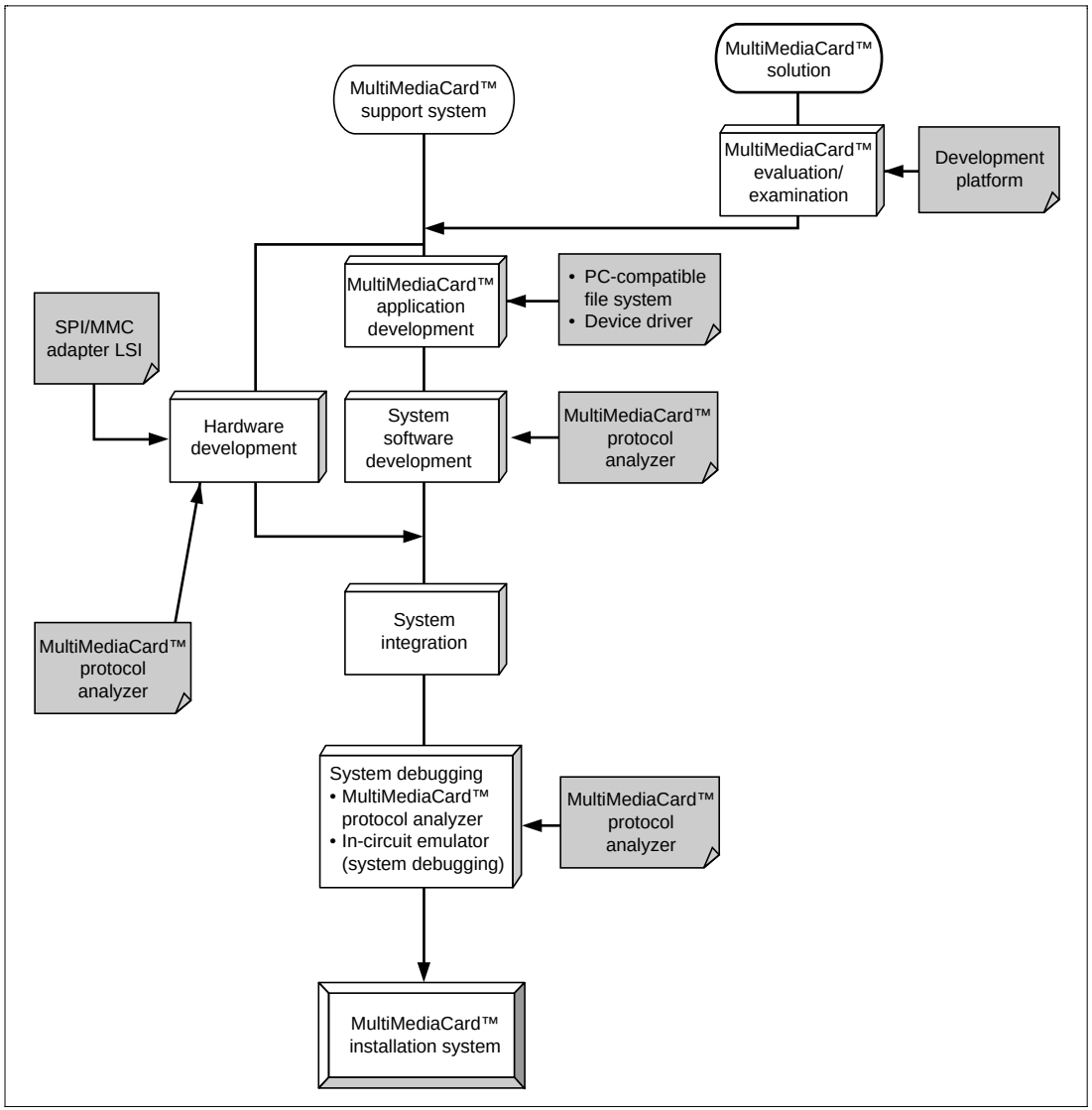


Figure 3.1 Overview of System Development

First, MMC evaluation and examination is carried out using the development platform in the MMC solution stage. When this evaluation work is completed, development of the MMC support system is begun. The development sequence comprises software development, hardware development, and then system integration, after which system debugging is carried out and the MMC installation system is completed.

1. Software development

   Development of the installation system control program and user application program is carried out.

   In MMC application development, files in the MMC can be supported by using the PC-compatible file system and device driver.

   In system software development, use of a multimedia protocol analyzer enables almost all system software checking to be carried out before the application product is completed.

2. Hardware development

   Hardware development involves MMC interface development in addition to the usual development tasks. Use of adapter control system SPI mode and MMC mode LSIs and an MMC socket simplifies MMC implementation. In addition, the provision of SuperH™* microcomputers, FPAGs, ASICs, CBICs, etc., as MMC interface adapter logic components is planned, to further simplify embedding and shorten development times. Hardware-based system debugging of the MMC interface of the completed hardware can be carried out using a multimedia protocol analyzer.

   Note: * SuperH is a trademark of Hitachi, Ltd.

3. System debugging

   When development of both software and hardware has been completed, the two are integrated in the system integration stage, and system debugging is begun. Conventionally, hardware/software debugging is executed and system confirmation work carried out using an in-circuit emulator, but with an intelligent device such as an MMC, a logic analyzer or similar means is used, making it impossible to shorten the system development time and also imposing a heavy load on the developers. MMC-related bug analysis can be carried out using the MultiMediaCard™ protocol analyzer provided for the purpose of simplifying MMC installation. Also, if the developers want to conduct testing with specific data provided in the MMC, it is possible to carry out formatting and analysis of data recorded in system testing with the development platform.

The development of an MMC installation system is carried out in this way.

An overview of the development sequence in developing an MMC installation system has been given above. The construction of the installation system will now be considered in greater detail as shown in figure 3.2.

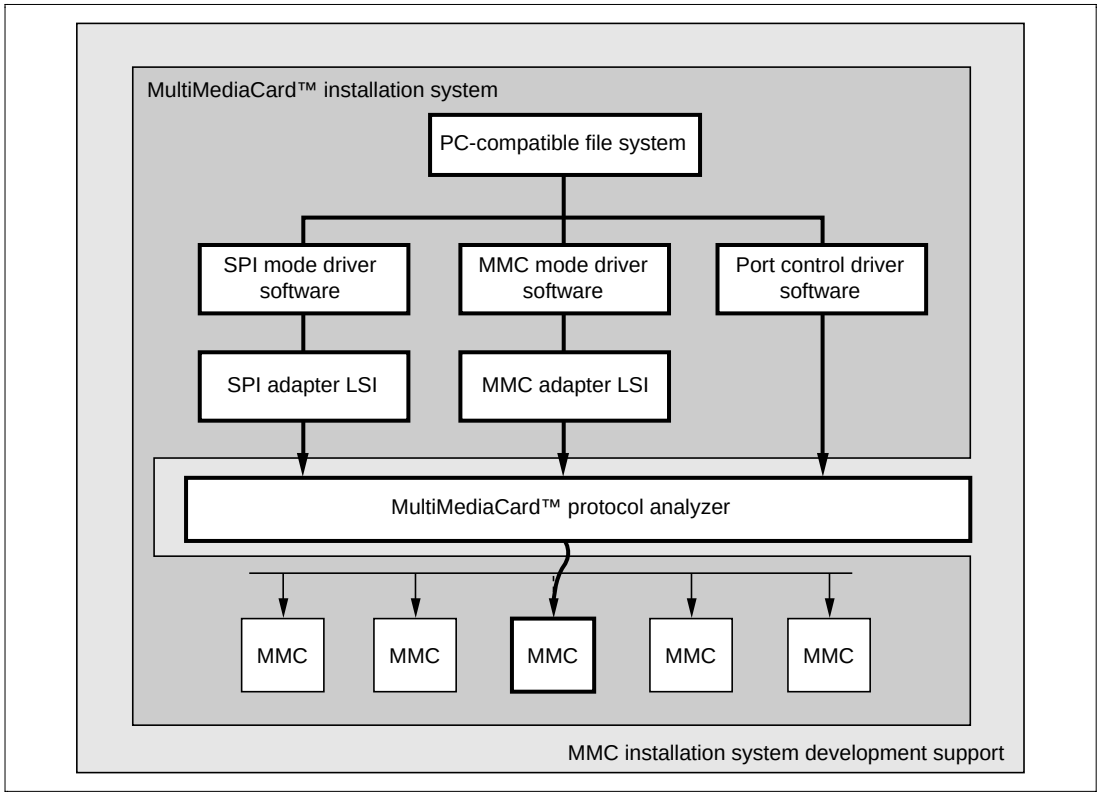


Figure 3.2 Installation System Configuration Diagram

An MMC installation system includes port control and adapter control by means of an MMC control system , with a PC-compatible file system as the main element. The adapter control system SPI mode and MMC mode are described here.

As shown in figure 3.2, an installation system is constructed by selecting a number of systems, in all of which MMC access is performed via SPI mode driver software and an SPI adapter LSI or MMC mode driver software and an MMC adapter LSI, and data writes, reads, and erases are performed in file units. MMC processing is executed in parallel on the system side, while on the card side commands, data, and responses from the card are processed serially. In intelligent card debugging, the ability to collect and analyze various statuses and other kinds of information more effectively enables the developer to make more efficient use of manpower. Development time can be shortened by using an MMC protocol analyzer.

Figure 3.3 summarizes the development support available for products using MMCs. Development support can be broadly divided into three categories.

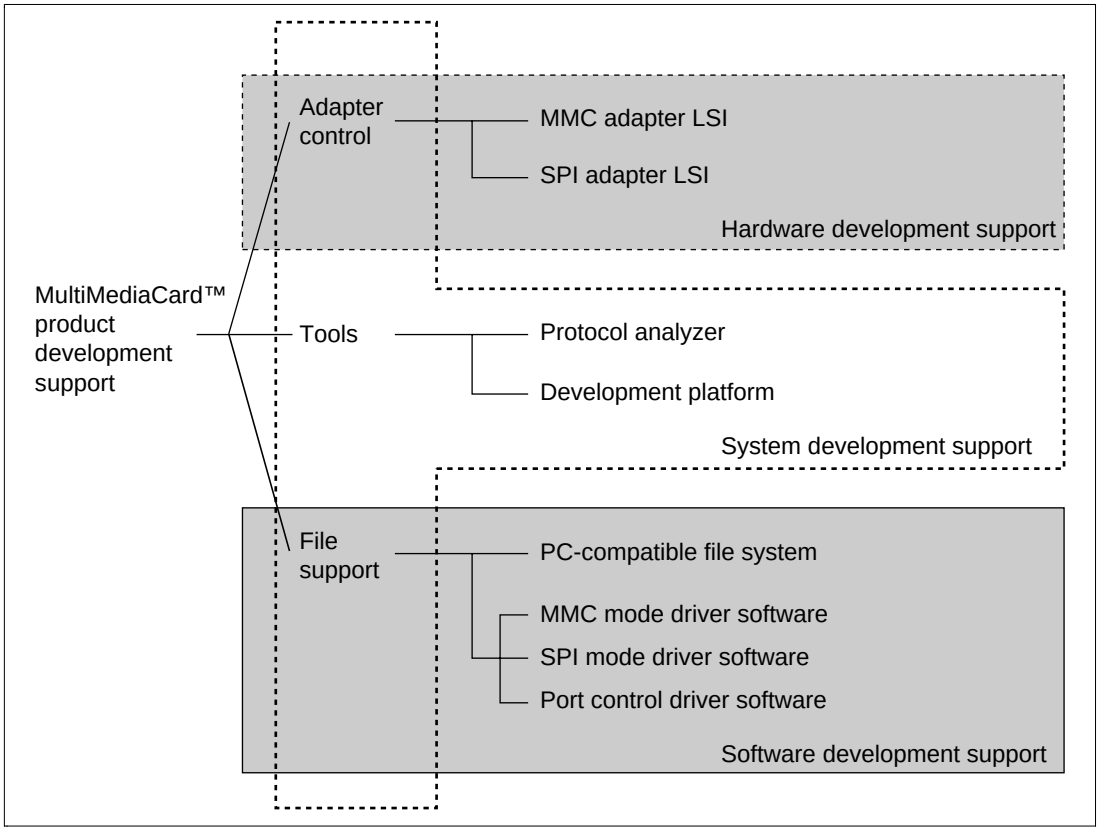


Figure 3.3 Product Development Support

1. Hardware development support

   In order to install an MMC in hardware, an adapter control system must be implemented. For this purpose, an MMC adapter LSI and SPI adapter LSI are provided, allowing easy incorporation of both systems in the user system.

2. Software development support

   To support easy implementation of a file system in an MMC, a PC-compatible file system is provided, together with two kinds of driver software for implementing an adapter control system. This allows the user to develop application software without any special knowledge of MMCs. Similarly, port control system driver software is also provided.

3. System development support

   A protocol analyzer and development platform are provided in order to carry out comprehensive debugging of software and hardware during system development. The development platform can be used for the following purposes in addition to evaluation and examination.

a. Although MMC file writes do not function in system debugging, files (image, voice, text, etc.) can be created if it is wished to test read processing.
b. Checking whether a file written in system debugging has been correctly created.
c. Correction of data in the MMC.

Use of such development support tools simplifies the development of an MMC system. Details of software support and tools available are given in the following sections.

3.2 Adapter Logic

An MMC is connected to the host by serial interface lines called CLK, CMD, and DAT. To enable high-speed data transfer at up to 20 Mbps to be executed efficiently between the card and the host, hardware called an adapter is required between the general-purpose microcomputer in the host and the MMC. Figure 3.4 shows a sample configuration of a system incorporating this hardware.

Two kinds of adapter—an MMC adapter and an SPI adapter—are stipulated in the MultiMediaCard™ System Specification*. The amount of circuitry incorporated in the adapter is determined by the required system speed. Here, port control system, MMC adapter, and SPI adapter hardware circuits implemented using commercially available FPGAs will be described.

Note: * Specification drawn up by the MMCA (MultiMediaCard™ Association)

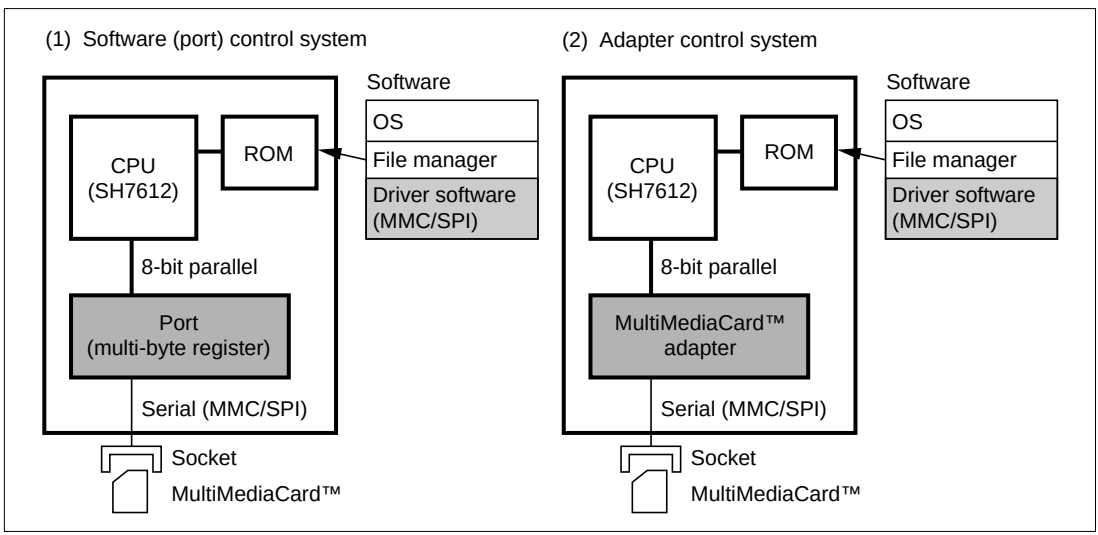


Figure 3.4 Example of MultiMediaCard™ System Implementation

3.2.1 Port Control System

Figure 3.5 shows a block diagram of the port control system constructed this time.

A one-byte register port is constructed in the external memory area of the host microcomputer using an FPGA, and control of CLK, CMD, DAT, and VDD is executed by means of bits in this register. Power supply on/off control, clock high/low level control, individual bit output for commands, and bit-unit input/output of data are all handled by host microcomputer driver software processing. This results in a lower transfer speed but simplifies the hardware configuration.

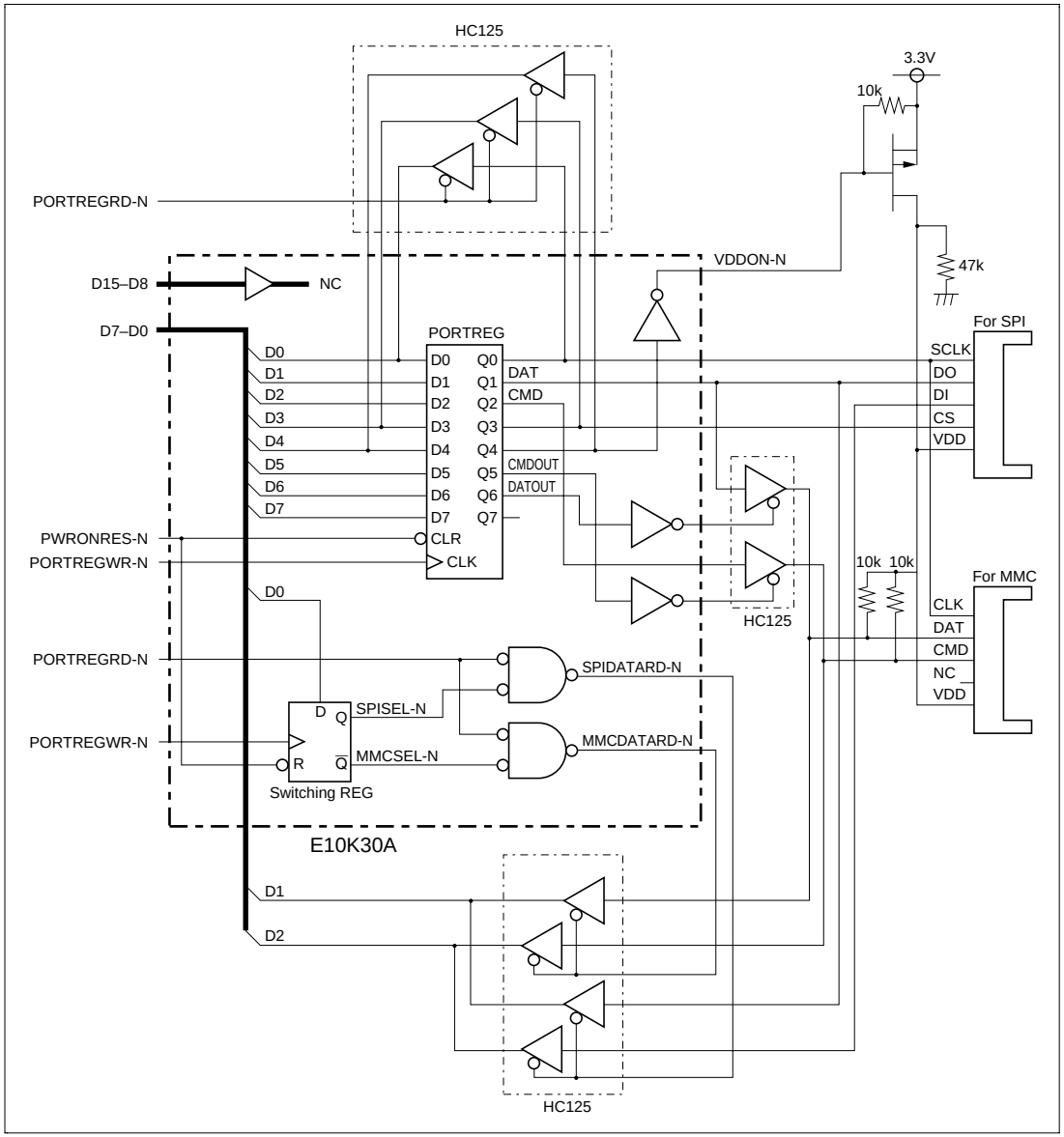


Figure 3.5 Block Diagram of Port Control System

3.2.2 MMC Adapter

Figure 3.6 shows a block diagram of the MMC adapter constructed this time.

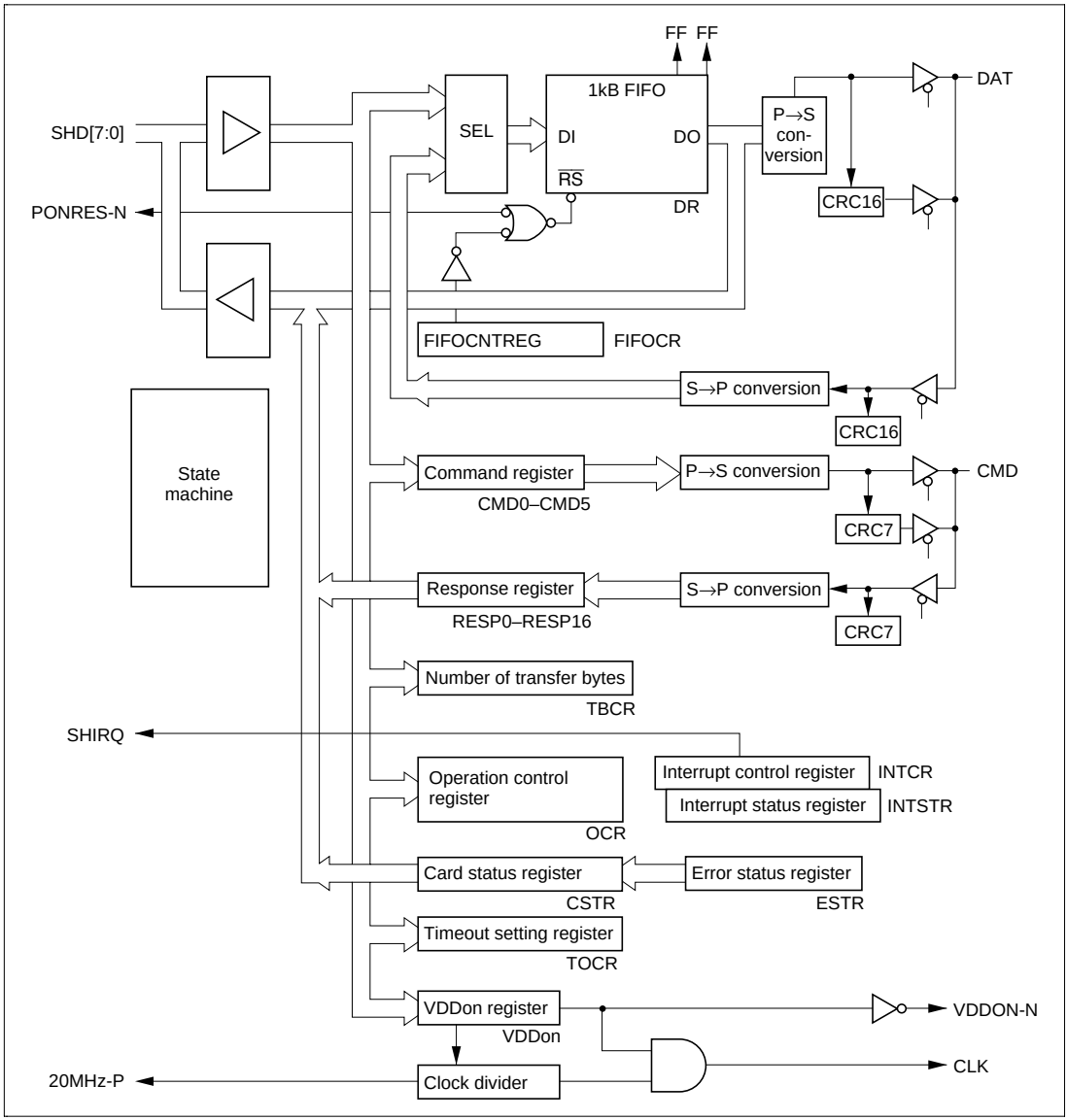


Figure 3.6 Block Diagram of MMC Adapter

In order to reduce the software overhead of the host microcomputer, FPGAs are used to configure circuits such as a circuit to convert the microcomputer bus interface to the MMC serial interface, and a command register (6 bytes), response register (17 bytes), and data register (1-kbyte FIFO).

The MMC transfer protocol requires a CRC check to ensure data transfer quality in command transfer, response transfer, and data transfer, so a CRC code is automatically generated by hardware and added to the bit pattern when a command or data is transmitted. When a response or data is received, a CRC check is carried out automatically.

The data register incorporates a FIFO that can be written to and read independently in the FPGA. In data transfer to the MMC, data is written to the FIFO from the microcomputer beforehand, a command is set in the register, and when the command is started the command pattern is automatically transmitted together with the CRC, and the response from the card is shown in the response register. The microcomputer checks the contents of this response, and if valid, set the data transfer enable register to transfer enable status. The data held in the FIFO is then read by hardware, byte by byte, converted to a serial bit stream, and sent from the DAT line on the MMC interface. An error during data transfer, lack of a response, or a FIFO empty or FIFO full condition, is reported by an interrupt to the microcomputer. In addition to the above registers required as MMC adapter registers, a register that turns the MMC card power supply on and off, and a function that selects a 400 kHz, 5 MHz, 10 MHz, or 20 MHz clock for supply to the MMC, are also provided.

3.2.3 SPI Adapter

The hardware configuration of the SPI adapter is the same as that of the MMC adapter described above. Differences in the MMC bus interface are that the response length is a maximum of 2 bytes, the data transfer block size is a maximum of 512 bytes, and multiple block data transfer in response to a single command is not permitted. As with the MMC adapter, a parallel/serial conversion circuit, command register (6 bytes), response register (2 bytes), data register (512-byte FIFO), etc., are configured by FPGAs. The CRC circuit, interrupt generation circuit, clock generation circuit, and power supply on/off circuit are the same as in the MMC adapter. The CS line specific to the SPI interface is controlled from the microcomputer by means of a newly added register.

3.2.4 Host Microcomputer

For the above adapter functions, operation has been checked with each MMC adapter incorporated in the daughter board holding the FPGA. The read performance has been obtained as an initial aim with the port control system, SPI adapter, and MMC adapter. In the case of an MMC card, there is a strong demand for the use of an 8-bit class microcomputer as the host, and the development platform has been created with an H8S microcomputer as its base, as described below.

3.3 Development Platform

The MMC development platform was created in order to simplify the development of a system using MMC cards, and because there is a need for a system enabling easy evaluation of MMC card reading and write performance.

Figure 3.7 shows the configuration of the system created this time, and figure 3.8 shows a block diagram of the system.

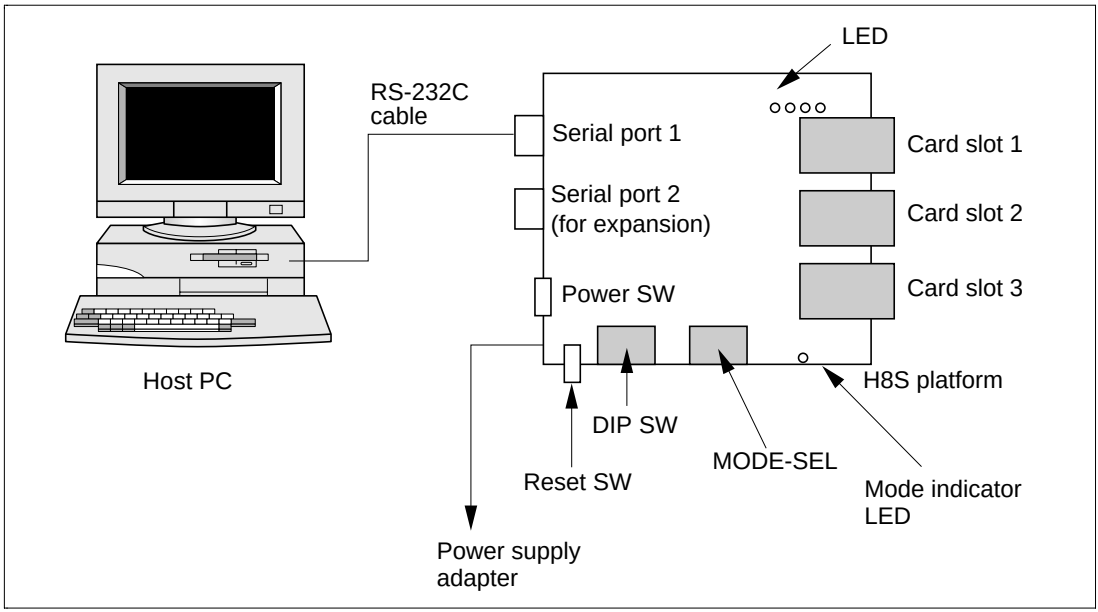


Figure 3.7 System Configuration

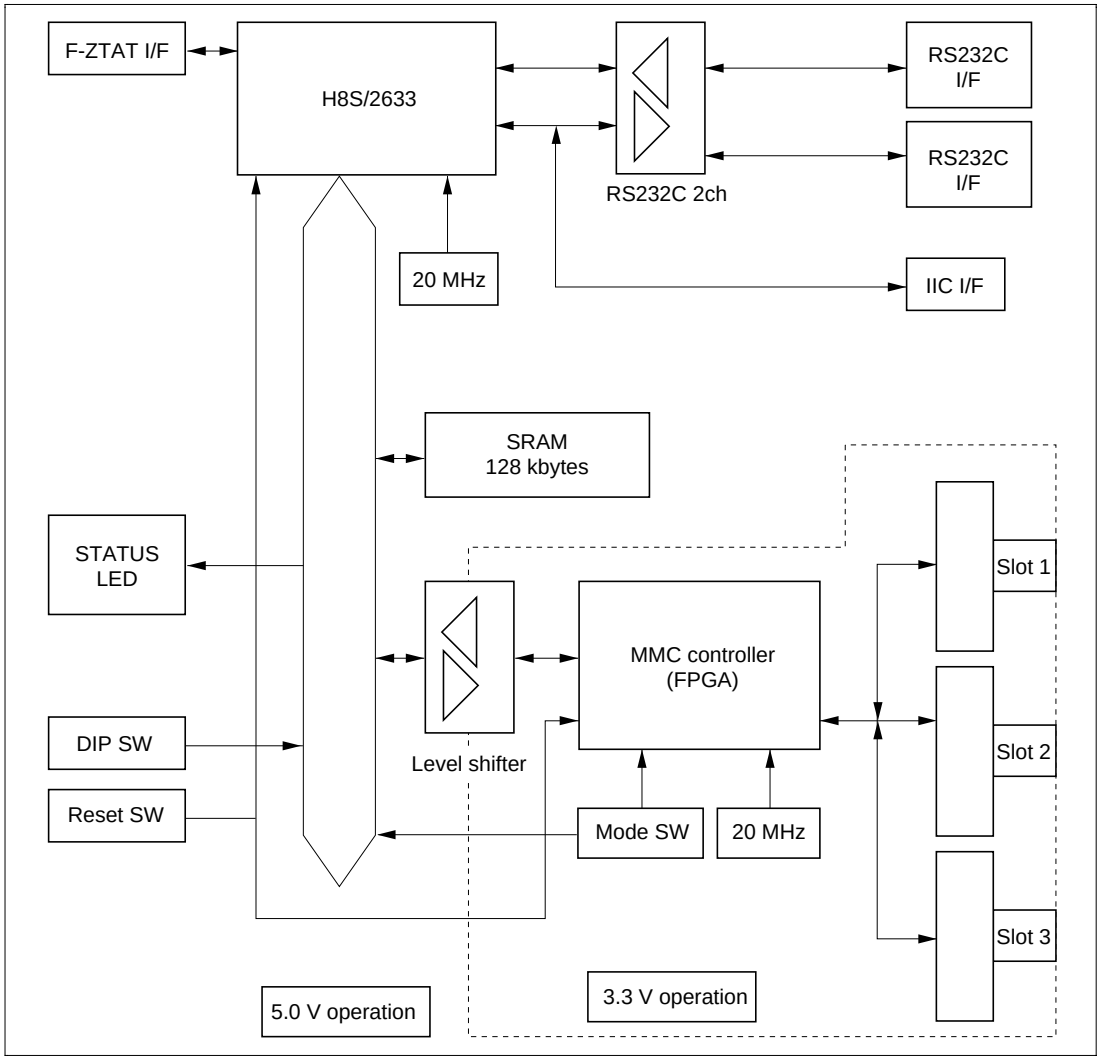


Figure 3.8 Block Diagram of Development Platform

3.3.1 Microcomputer Selection

The H8S/2633 was selected as a microcomputer for low-end consumer applications, featuring on-chip flash memory, an external memory space of 128 kbyte or more, and two or more serial interface channels to a personal computer, and capable of operating at 20 MHz or above.

As the maximum MMC bus interface clock speed is 20 MHz, 20 MHz has been set for the CPU clock.

3.3.2 Control Adapter Circuit

Either the SPI adapter circuit or MMC adapter circuit mentioned earlier is configured in the FPGA automatically at power-on by means of a switch mounted on the board. The value of this switch can also be read by the H8S microcomputer, enabling monitor software incorporated in the platform to recognize automatically which adapter is mounted.

An 8-bit standard interface is used as the interface between the adapter and the H8S microcomputer, and data (8-bit), address (5-bit), RD, WR, CS, IRQ, and CLOCK signal lines are used. The adapter is seen from the H8S microcomputer as a group of registers comprising approximately 30 bytes.

3.3.3 MMC Bus Interface

The control adapter has an interface to the microcomputer and an interface to the cards. The MultiMediaCard™ System Specification stipulates a card stack of up to 30 cards for a single MMC bus, but the platform developed this time is provided with 3 card slots. When a single control adapter handles multiple card slots, this is provided for in the SPI adapter by increasing the number of CS lines (CS0, CS1, CS2, and so on). With the MMC adapter, multiple cards are managed automatically on the card side, without the need for additional signals. Three card slots are provided to facilitate operation verification and evaluation when there are a number of memory cards in the system.

3.3.4 Monitor Commands

The MultiMediaCard™ development platform is used with a personal computer connected as a terminal via a serial cable.

A monitor program and driver software are written in the flash memory on the platform, and the following functions are supported. Table 3.1 lists the monitor commands.

- MMC card register information (CID and CSD) can be displayed.
- Specified MMC card addresses (specified by sector number) can be read into the edit buffer, edited, and written.
- MMC card file operations can be performed, including file list display, file copying, and file contents display.
- File transfer can be executed between the host system and an MMC card on the H8S platform.
- MMC and SPI protocols are supported for MMC card access.

Table 3.1 Monitor Commands

Function	Command	Processing	Notes
CID, CSD display	CID	CID register information display	
	CSD	CSD register information display	
MMC specified address read/write	R	Reads data in a specified MMC card sector into the edit buffer.	Single-sector unit (fixed at 512 bytes)
	W	Writes data in a specified MMC card sector from the edit buffer.	Single-sector unit (fixed at 512 bytes)
Buffer editing	D	Edit buffer contents display	
	M	Edit buffer contents change	
	F	Data embedding in edit buffer	
Formatting	FORMAT	MMC card formatting	
Directory operations	DIR	File list display	
	MD	Directory creation	
	RD	Directory deletion	
File operations	COPY	File copy	
	DISKCOPY	Disk copy	
	DUMP	File dump	
File transfer to/from host system	PUT	Transfer of one file from MMC card to host PC	
	GET	Transfer of one file from host PC to MMC card	

3.4 Software for Products Incorporating MMCs

In order for data written in an MMC to be recognized as a file by a personal computer, etc., a PC-compatible file system and driver software must be installed on the embedded system side. Here, the software required for development of an embedded product incorporating an MMC is described, taking the example of driver software created for use with “USFiles” available commercially as a PC-compatible file system for embedded applications.

As software for implementing requests to the MMC such as “data write to MMC” and “data read from MMC” in an embedded system, software called a “driver” is needed that performs accesses to the MMC based on a procedure whereby an instruction called a “command” is sent to the MMC and a reply called a “response” is received from the MMC by the software. The main processing of the driver consists of MMC initialization, data reading, data writing, area erasing, and MMC status checking.

3.4.1 Software Configuration

Figure 3.9 shows an example of the software configuration in an embedded system.

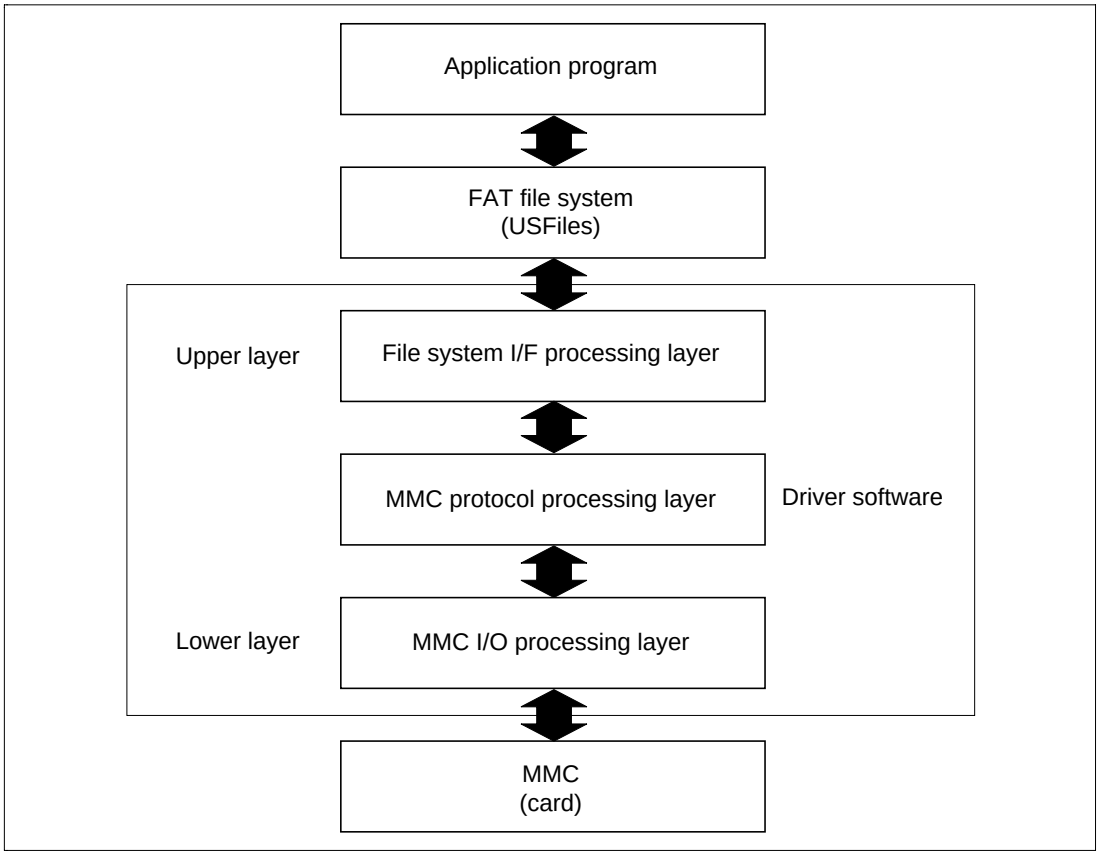


Figure 3.9 MMC Control Software Configuration

The area from the file system/driver I/F processing layer to the MMC I/O processing layer corresponds to the driver software. The MMC has two modes—SPI mode and MMC mode—which differ as follows from the viewpoint of software.

1. MMC mode
   - There is an Identification sequence in which an address is assigned to an MMC (details given below).
   - There is a Broadcast command that issues instructions to all MMCs simultaneously.
   - In order to select a specific MMC from among multiple MMCs, select and deselect processing is required, using MMC addresses. (Uses the card select command.)
   - There are more types of response than in SPI mode.

2. SPI mode
   - The Identification sequence is simple.
   - When multiple MMCs are used, a Chip Select signal (CS) is used to select a specific MMC.
   - The response system is simple, with fewer types of response from the MMC than in MMC mode.

The software that performs processing in accordance with the MMC or SPI mode protocol comprises the protocol processing layer. Processing such as MMC initialization, reading, writing, and erasing is performed in this layer.

There are two control methods, distinguished by the nature of the trade-off between hardware and driver software functions: the adapter control system in which MMC bus protocol processing is handled by hardware, and the port control system in which processing is performed by software. These are handled in the I/O processing layer.

1. Port control system

   In this system, everything from command generation to port input/output processing is controlled by software. Specifically, all processing including clock transmission, data transmission and reception in synchronization with the clock, and CRC addition to transmit data and CRC checking in receive data to detect errors on the bus, is controlled by software.

3.4.2 Interface to File System

The interface between the file system and driver software comprises the 8 functions shown in table 3.2. Providing these functions on the driver side enables access to be performed without awareness of the recording medium (MMC, CF card, etc.).

Table 3.2 Interface between File System and Driver

init( )	Initializes device
format( )	Performs physical formatting of sectors
raw_read( )	Reads sector (cylinder, head, sector specified)
raw_write( )	Writes sector (cylinder, head, sector specified)
read( )	Reads sector (logical sector number specified)
write( )	Writes sector (logical sector number specified)
timestamp( )	Returns time and date
diskchange( )	Returns notification that disk has been changed

3.4.3 MMC Hot Insertion/Removal

MMCs support hot insertion and removal. The process for hot insertion and removal is described below taking the example of file opening (figure 3.10).

When a file open request is sent from the application program to the file system, the file system first checks if the medium has been changed. This is done by calling mmc_diskchange() on the driver software side. If a medium change is not detected, the procedure moves on to file open processing. If a medium change is detected, on the other hand, MMC initialization is performed by means of mmc_init() on the driver software side. When initialization is completed normally, the procedure moves on to file open processing.

A sample program for the diskchange() function of an MMC driver created for USFiles is shown in figure 3.11.

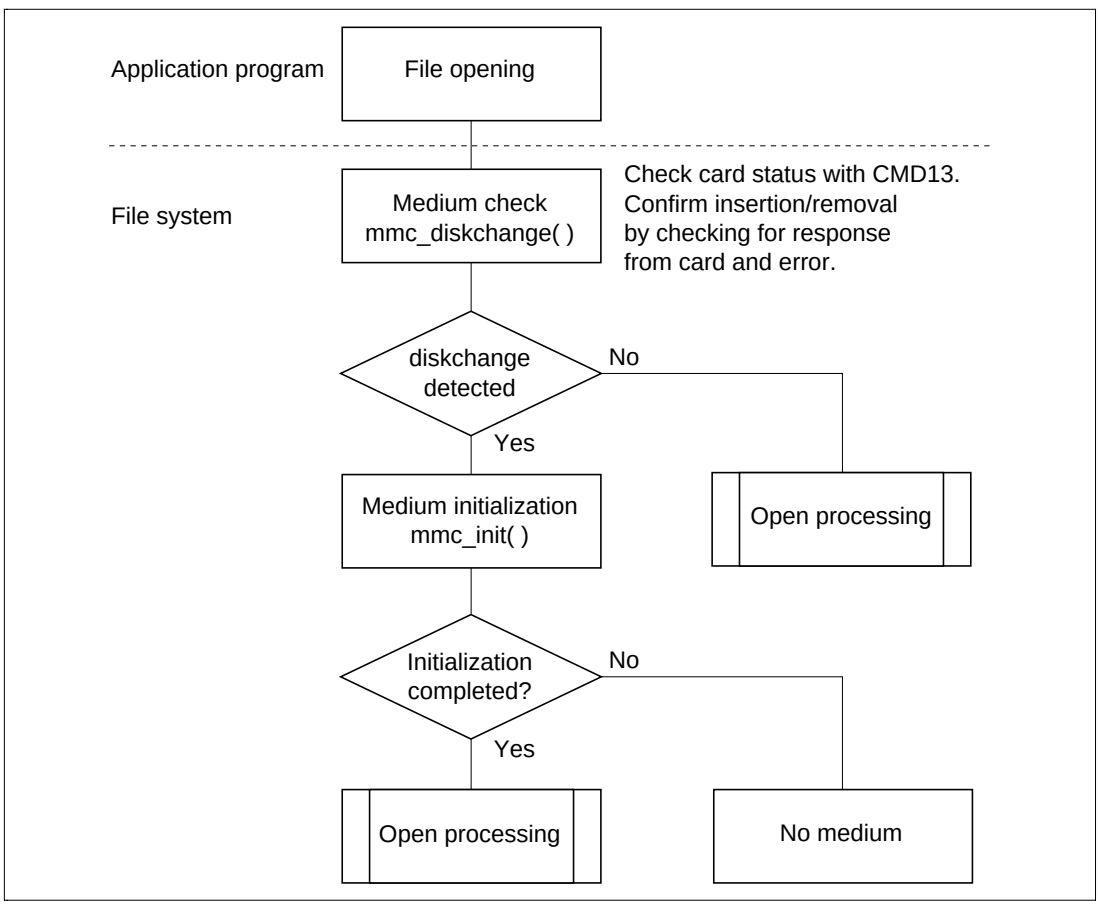


Figure 3.10 Processing for File Opening

```
int mmc_diskchange( DEVICE *devp )
{
    byte   card_no;
    uint32  ret;

    card_no = 0;
    ret = 0;
    /*-----------------------------------------------------*/
    /*  Card number acquisition                            */
    /*-----------------------------------------------------*/
    card_no = (devp->unit_no & ~UNIT_NO);
    /*-----------------------------------------------------*/
    /*  Card status check                                  */
    /*-----------------------------------------------------*/
    if ( ret = MMC_Send_Status( card_no, &Card_Info ) ){
       return( 1 );/* Medium change    */
    }
    else{
       return( 0 );/* No medium change */
    }
}
```

Figure 3.11 Sample Coding for Handling Hot Insertion/Removal

3.4.4 MMC Initialization

In order to perform data read/write access, etc., to an MMC, the MMC must first be initialized. The initialization processing is different for MMC mode and SPI mode, and the appropriate processing must be carried out for the relevant mode. MMC mode uses an open-drain control method. The initialization processing for open-drain MMC mode is described in detail below.

The commands shown in table 3.3 are provided for MMC initialization.

Table 3.3 Commands for Initialization

Command	Response	Abbreviation	Description
CMD0	—	GO_IDLE_STATE	Resets all cards and places them in the idle state.
CMD1	R3	SEND_OP_COND	Asks all cards for their operation condition.
CMD2	R2	ALL_SEND_CID	Asks all cards for CID.
CMD3	R1	SET_RELATIVE_ADDR	Assigns RCA.

1. CMD0 issuance

   In MMC initialization, the host first issues CMD0 to the MMCs. Issuing CMD0 places the MMCs in the idle state. Normally, an MMC returns a response to a command from the host, but there are number of commands for which no response is returned. CMD0 is one such command, and no response is sent to indicate whether an MMC has entered the idle state. Therefore, the host needs to issue CMD0 as many times as is considered adequate.

2. CMD1 issuance

   When the MMCs have gone to the idle state, CMD1 is issued. CMD1 enables the voltage value of the power supply to the MMCs to be set. When CMD1 is issued, if an MMC supports the set voltage value, it returns the voltage range it supports as a response. An MMC outside the supported voltage range goes to the inactive state.

3. CMD2 issuance

   If there is an MMC from which a response is obtained after issuing CMD1, the host next issues CMD2. When CMD2 is issued, the MMC returns CID (Card Identification Register) data as a response.

   If there are a number of MMCs on the bus, all the MMCs return a response on receiving CMD2 from the host. In this case, the response of the MMC that first acquires bus mastership can be received by the host.

4. CMD3 issuance

   The host must issue a CMD3 command to the MMC that has acquired the bus. CMD3 allows the RCA (Relative Card Address)—the number assigned to an MMC—to be set. The RCA can be set in a range from 1 to 65,535, and is used to select a particular MMC to be accessed from among multiple MMCs in subsequent accesses.

   If there are a number of MMCs on the bus, the host must issue the same number of CMD2 commands as there are MMCs on the bus, for the MMCs that were not able to acquire the bus when the first CMD2 was issued.

   When CMD3 ends normally, the MMC goes to data transfer mode (from card identification mode). An MMC that has entered data transfer mode does not return a response to subsequent CMD2 commands.

   This procedure prevents duplicate RCAs from being set when there are a number of MMCs on the bus, and enables all the MMCs to be placed in data transfer mode.

The above initialization procedure is illustrated in figure 3.12.

Figure 3.13 shows a sample program for the initialization function mmc_init() of an MMC driver created for USFiles.

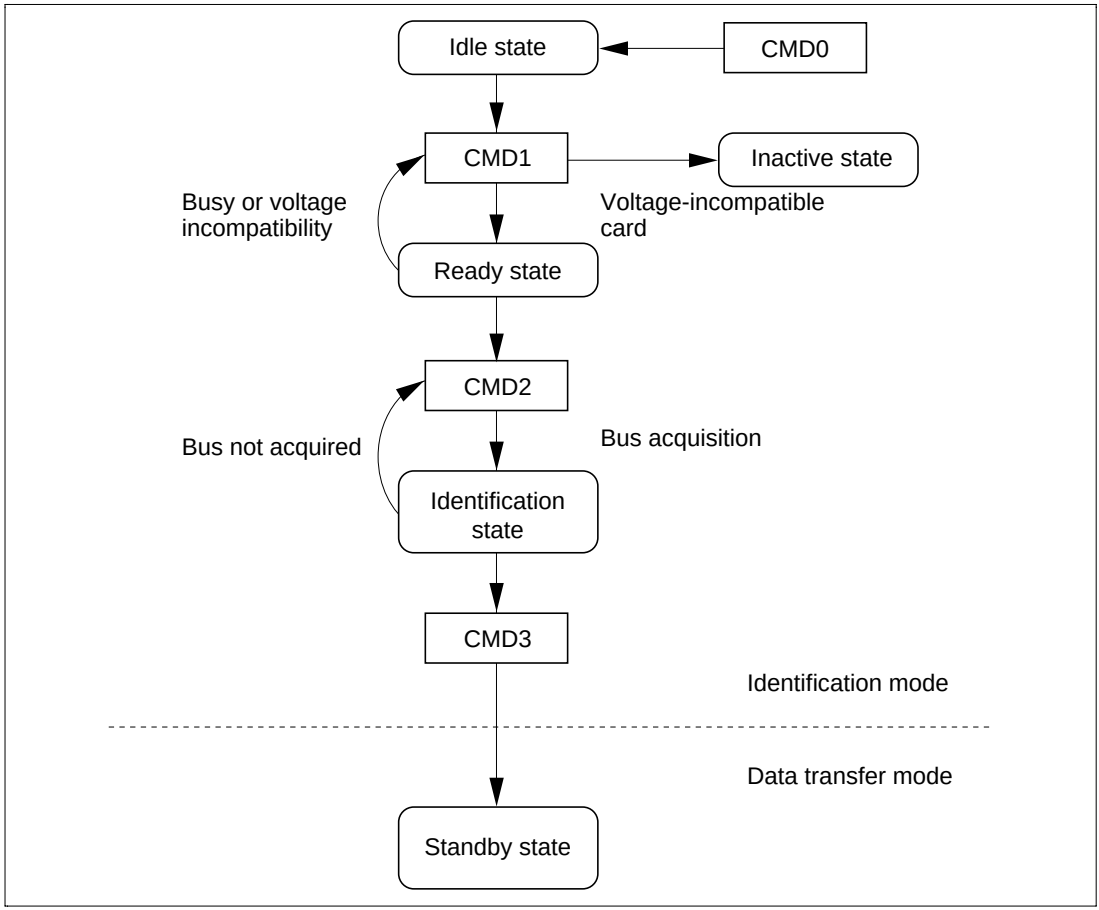


Figure 3.12 MMC States during Initialization

```
int mmc_init( DEVICE *devp )
{
    uint32  ret;/* Error presence/absence during initialization */

    ret = 0;
    /*-------------------------------------------------------------*/
    /*  Card initialization (SPI mode: processing up to CMD0, CMD1 */
    /*                       MMC mode: processing up to CMD0 to CMD3) */
    /*-------------------------------------------------------------*/
    if ( ret = MMC_InitCard( &Card_Info ) ){
        return( 1 );
    }
    /*-------------------------------------------------------------*/
    /*  CSD, CID acquisition                                       */
    /*-------------------------------------------------------------*/
    if ( ret = MMC_GetInfo( &Card_Info ) ){
        return( 1 );
    }
    /*-------------------------------------------------------------*/
    /*  Set number of bytes per block                              */
    /*-------------------------------------------------------------*/
    if ( ret = MMC_SetSectLen( &Card_Info ) ){
        return( 1 );
    }
    return( 0 );
}
```

Figure 3.13 Sample Initialization Coding

3.5 MultiMediaCard™ Protocol Analyzer

A MultiMediaCard™ has a 7-pin external interface, over which data is transferred to and from the application product using three serial lines (CLK, CMD, and DAT). While the hardware interface is simple, data transfer between the card and the application product is determined by a protocol comprising commands and responses. There are 64 kinds of command. When developing a system using cards, the system development time can be shortened if a means is available for simplifying analysis of this protocol. With this in mind, an MMC protocol analyzer was developed and released on the market. (“Protocol analyzer” may be abbreviated to “analyzer” in the following text.)

Photo 3.1 and table 3.4 show the appearance and physical specifications of the protocol analyzer. This protocol analyzer system consists of the main analyzer unit and software running on a host personal computer for performing analyzer operations via a GUI. Figure 3.14 shows the protocol analyzer system configuration.

The protocol analyzer includes a variety of functions that are helpful in MMC system development, as described later, all of which can be handled via the GUI.

Photo 3.1 Protocol Analyzer

Table 3.4 Physical Specifications of Analyzer

Dimensions of Main Unit	235(W) × 170(L) × 50(D) mm
Weight	1.45 kg
Power Supply	AC 90 V to 132 V or AC 200 V to 264 V
Accessories	AC cord (for AC 90 V to 132 V or for AC 200 V to 264 V)
	RS232C cable (2m, DSUB 9-pin type)
	Parallel cable (2m, DSUB 25-pin type)
	Analyzer probe (50 cm, coaxial cable)
	Pseudo-card cable (50 cm, coaxial cable)
	System software (two 3.5" FDs)
	Operation manual (Japanese and English versions)

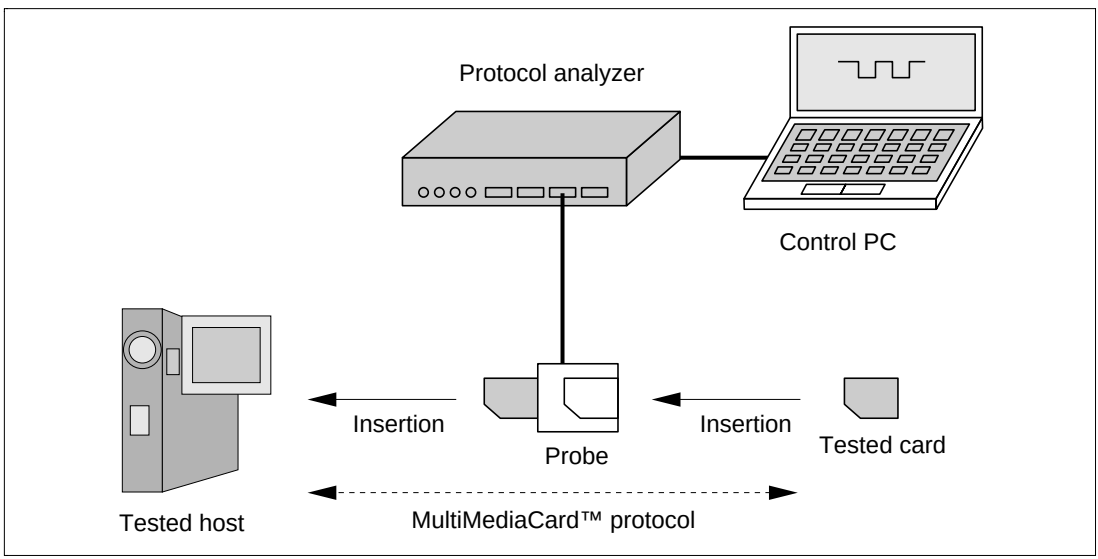


Figure 3.14 Protocol Analyzer System Configuration

Table 3.5 lists the functional specifications of the protocol analyzer.

Table 3.5 Functional Specifications

No.	Item	Specifications
1	Trace functions	• Traces MMC bus states (CMD, DAT, CLK, V_{DD} voltage, V_{DD} current, Time Stamp information) • Supports four trace modes (1 of 4 selectable) — Free trace mode — Trace start condition specification mode — Trace stop condition specification mode — Trace monitor mode • Command/response-only trace mode (MMC mode only) • Trace capacity: 64K cycles — (Overwrite and buffer stop functions included) • Time stamp (real time) display — Resolution: 50 ns units — CLK, COMMAND, RESPONSE, and DATA intervals on MMC bus displayed as real time or number of clocks
2	Event detection	• Command pattern, response 1, 2, 3 pattern, or data pattern trace trigger (start or stop condition) can be specified • Forced trace start/stop by means of GUI buttons
3	Operating status monitoring	• MMC bus CLK status, V_{DD} voltage value, V_{DD} current value constantly displayed on GUI screen regardless of whether trace is on or off
4	Pseudo modes	• Normal mode, pseudo-host mode, and pseudo-MMC mode supported (1 of 3 selectable) — Normal mode used with analyzer probe inserted between MMC adapter and MMC. — Pseudo-host used with MMC inserted in MMC connector on main unit. Commands and write data can be sent from protocol analyzer. — Pseudo-MMC used with host connected via pseudo-MMC cable. Responses and write data can be returned from protocol analyzer. • Traces also valid in pseudo modes (simple protocol checks and timing checks possible) • V_{DD} on/off control and selection of various clocks possible in pseudo-host mode

No.	Item	Specifications
5	Host interface	• RS232C asynchronous serial or bidirectional parallel connection • RS232C serial: 9600 bps to 38400 bps • Bidirectional parallel: 150 kbytes/sec
6	LED display	• Power On, Trace On, MMC-VDD, CMD, and DAT status display
7	Switches	• Power ON/OFF switch (back) • Reset switch (front) • DIP switch (front)
8	DIAG functions	• Execution/non-execution of detailed diagnostic test at power-on specifiable according to DIP switch state
9	SPI mode support	• Functions in items 1, 2, 3, 5, 6, 7, and 8 are also valid in SPI mode (Either use in SPI mode or use in MMC adapter mode can be selected)

1. Trace function

   It is possible to obtain a trace of the signal states between a MultiMediaCard™ and the system, and to display commands, responses, data, etc., in an easily understood format on the host personal computer. (Voltage and current values are also displayed.)

2. Event detection function

   It is possible to specify various trigger conditions such as commands, responses, and data patterns, and to start and stop traces based on these conditions.

3. Pseudo-operation

   Two pseudo-operation modes are available: pseudo-host mode in which the analyzer performs pseudo-host operation and card-side interface debugging can be carried out, and pseudo-card mode in which the analyzer performs pseudo-card operation and host-side interface debugging can be carried out.

   These functions enable MMC system development to go ahead without a host system or card.

4. Performance measurement

   Data that can be obtained with the trace function includes data on the time for which the MMC bus is used, in addition to MMC signals. This data makes it possible to indicate the data transfer performance between an MMC and the host.

5. Bus state display

   The MMC bus power supply state and clock state are constantly displayed.

6. State saving and restoration

   Emulation states (set conditions and trace information) can be saved to disk and restored.

7. Hard copy

   Contents displayed on the screen can be output as hard copy on a printer.

Figure 3.15 shows an overview of the trace acquisition modes.

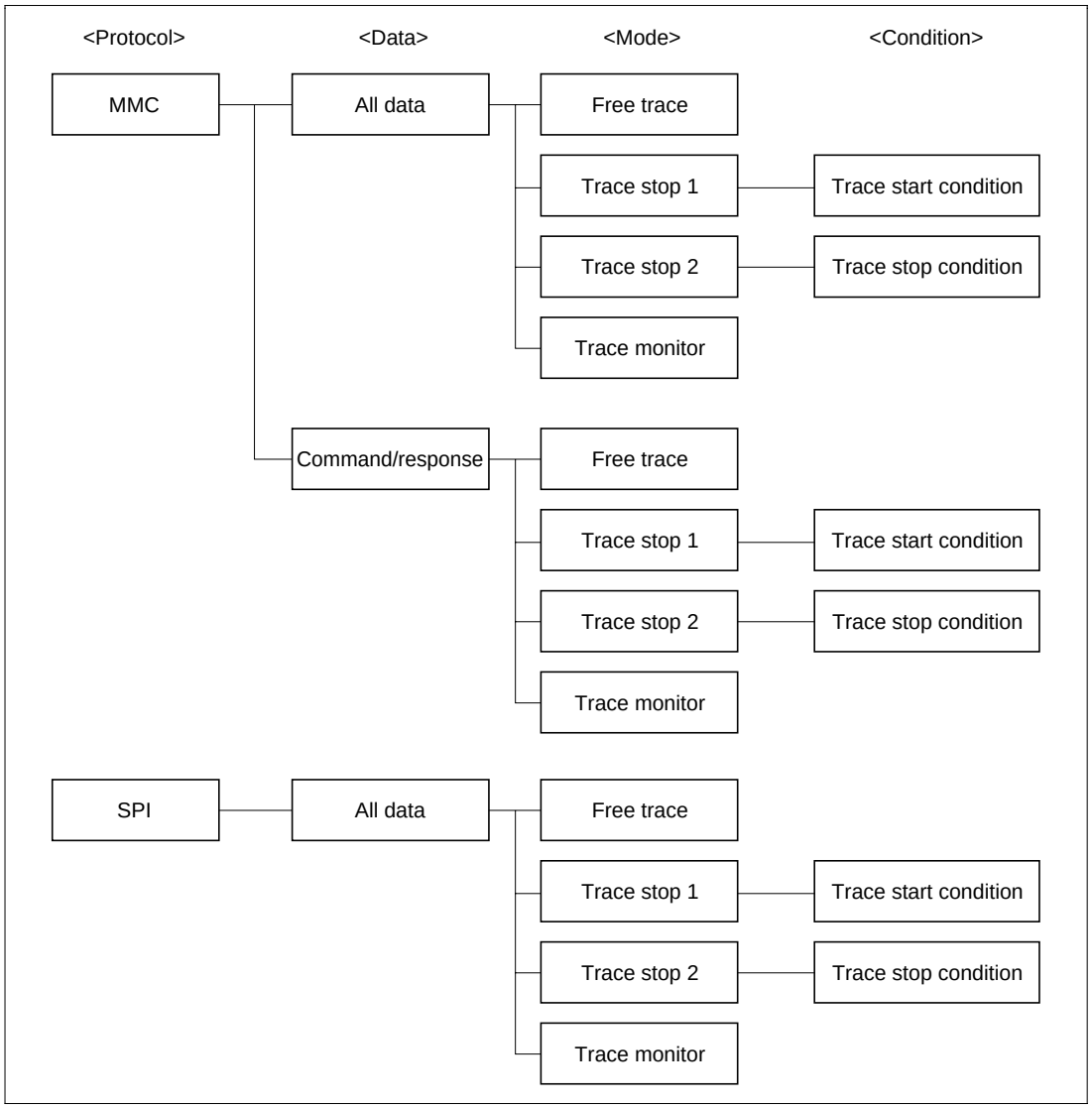


Figure 3.15 Overview of Trace Acquisition Modes

For <Protocol>, either MMC or SPI can be chosen as the protocol used for trace analysis.

For <Data>, there are two functions: an all-data trace function that acquires commands, responses, and data, and a command/response trace function that acquires commands and responses only. Use of this function enables only commands and responses to be traced, simplifying MMC protocol analysis.

For <Mode> and <Condition>, the trace mode and trace start and stop conditions can be specified. In a free trace, the latest bus states are constantly written to trace memory holding information for 64k cycles, and older information is overwritten.

Figure 3.16 shows the trace data outline window and figure 3.17 shows the trace data detail window.

After trace acquisition, these trace display functions can be used to give a clear indication on the personal computer screen of command types, response types, their intervals, clock-by-clock bit changes, and so on.

Trace data can also be output to the screen in list format.

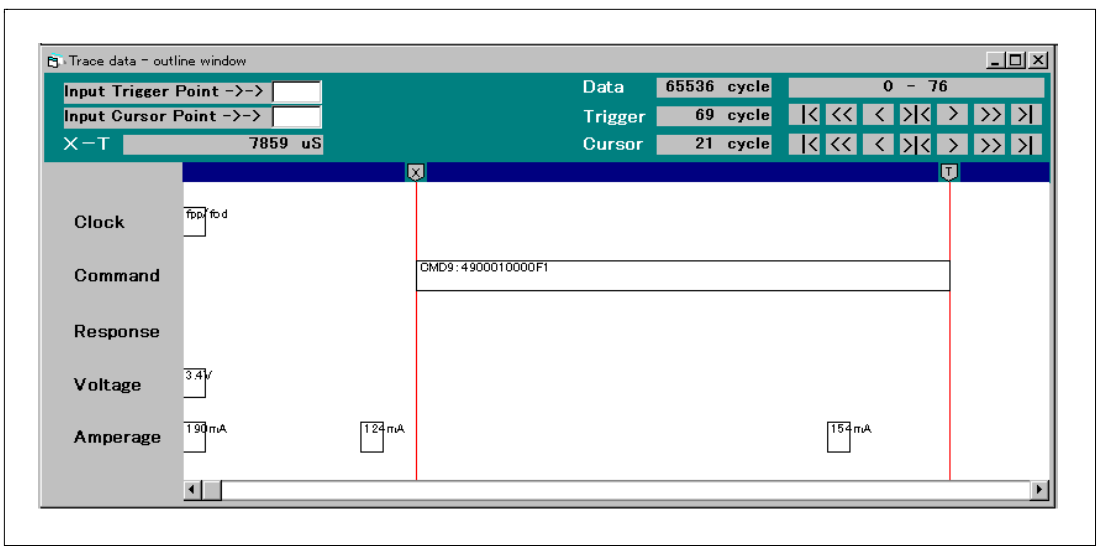


Figure 3.16 Trace Data Outline Window

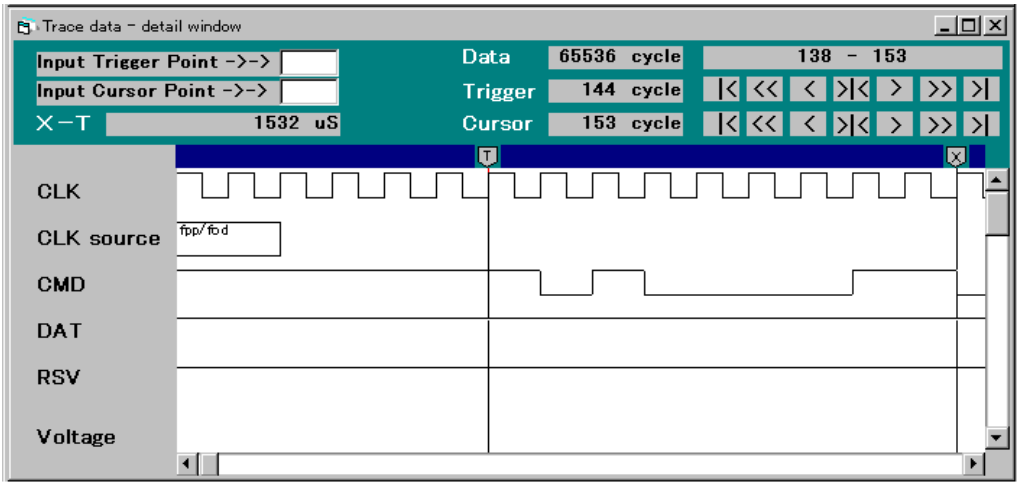


Figure 3.17 Trace Data Detail Window

Figure 3.18 shows the pseudo-host mode window.

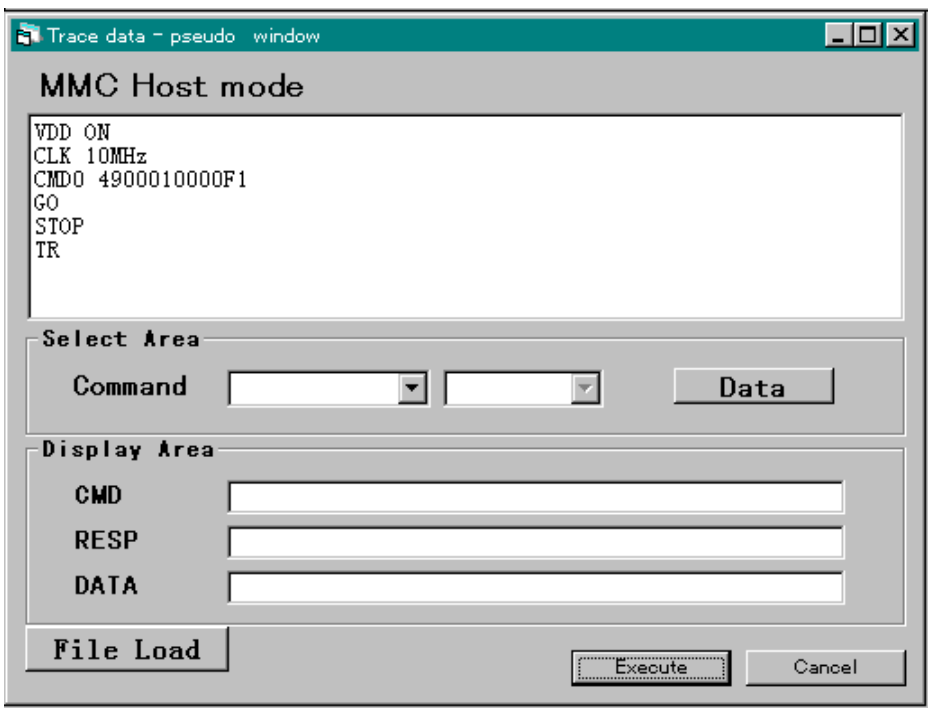


Figure 3.18 Pseudo-Host Mode Window

The MMC card protocol can be checked by using pseudo-host mode commands, issuing a command to an MMC card and checking the response. In the example in figure 3.18, after the protocol analyzer turns on V_{DD} for the card, it supplies a 10 MHz clock and then issues a CMD0 command. This sequence and the responses from the card are traced, allowing easy analysis. The pseudo-modes comprise the pseudo-host mode described here, and a pseudo-card mode in which the analyzer returns a response to the host in place of the MMC card.

Use of a protocol analyzer as described above offers an improvement over the previous practice of carrying out debugging while viewing waveforms on a logic analyzer or oscilloscope, and allows more efficient development of a system employing MultiMediaCard™.

3.6 Example of Application System Development (Music Player Prototype)

Portable music players are a typical example of the kind of products that make full use of the MultiMediaCard™'s special features of small size, light weight, and large-capacity flash memory. This section covers a prototype MP2 player created as a portable product for MMC performance evaluation and demonstration purposes. The development and prototype production of a mainstream MP3 player is also planned for the future. The prototype covered here uses the MMC adapter logic FPGA, MMC driver software, PC-compatible file system, and MMC protocol analyzer described earlier in the text as development tools. Photo 3.2 illustrates the setup during development.

Photo 3.2 MP2 Player (Prototype)

Figure 3.19 shows a hardware block diagram of the player developed in this example.

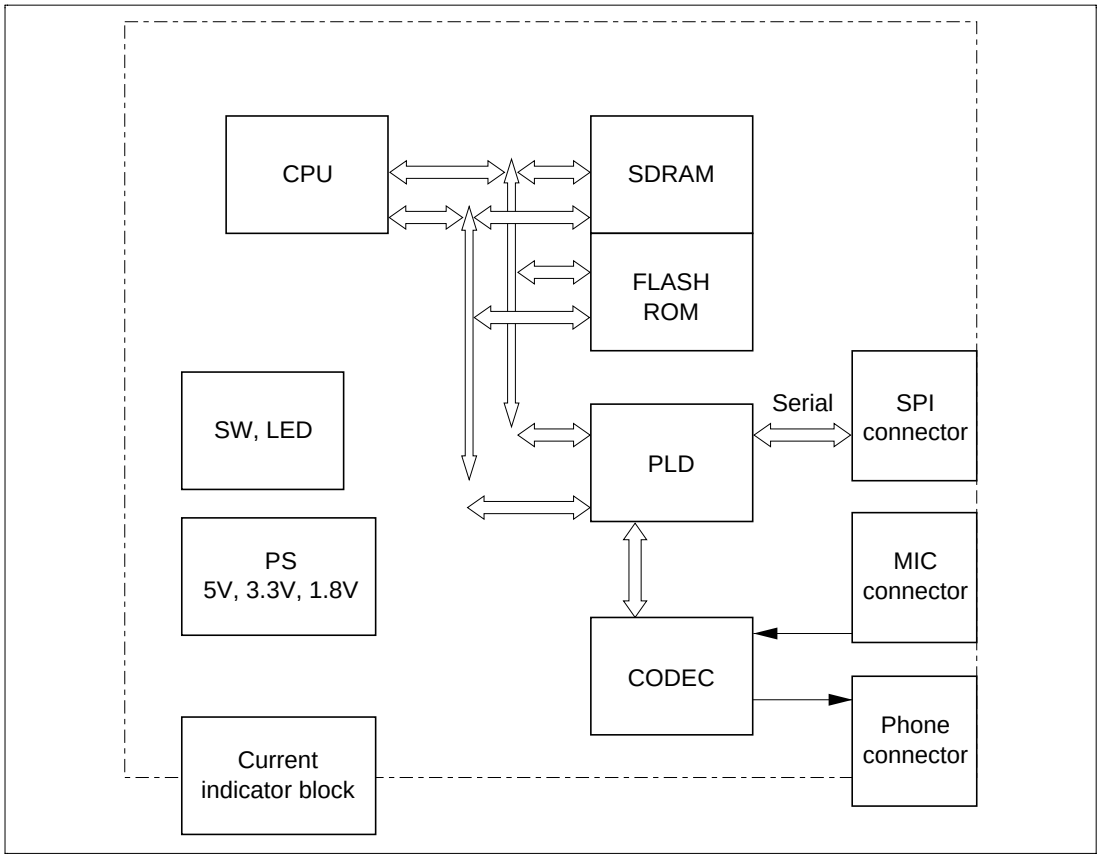


Figure 3.19 Block Diagram of MP2 Player

3.6.1 Adapter Logic between Microcomputer and MMC

In order to provide medium-class performance in terms of system speed for music recording and playback, SPI interface adapter logic circuitry is used. The SPI adapter has a 512-byte FIFO, an MMC command register, and a response register, and reduces the overhead on the driver software between the microcomputer and MMC. Figure 3.20 shows the memory map as seen from the CPU of the SPI adapter's microcomputer.

A transfer speed of 20 Mbits/second is used between the MMC and the adapter.

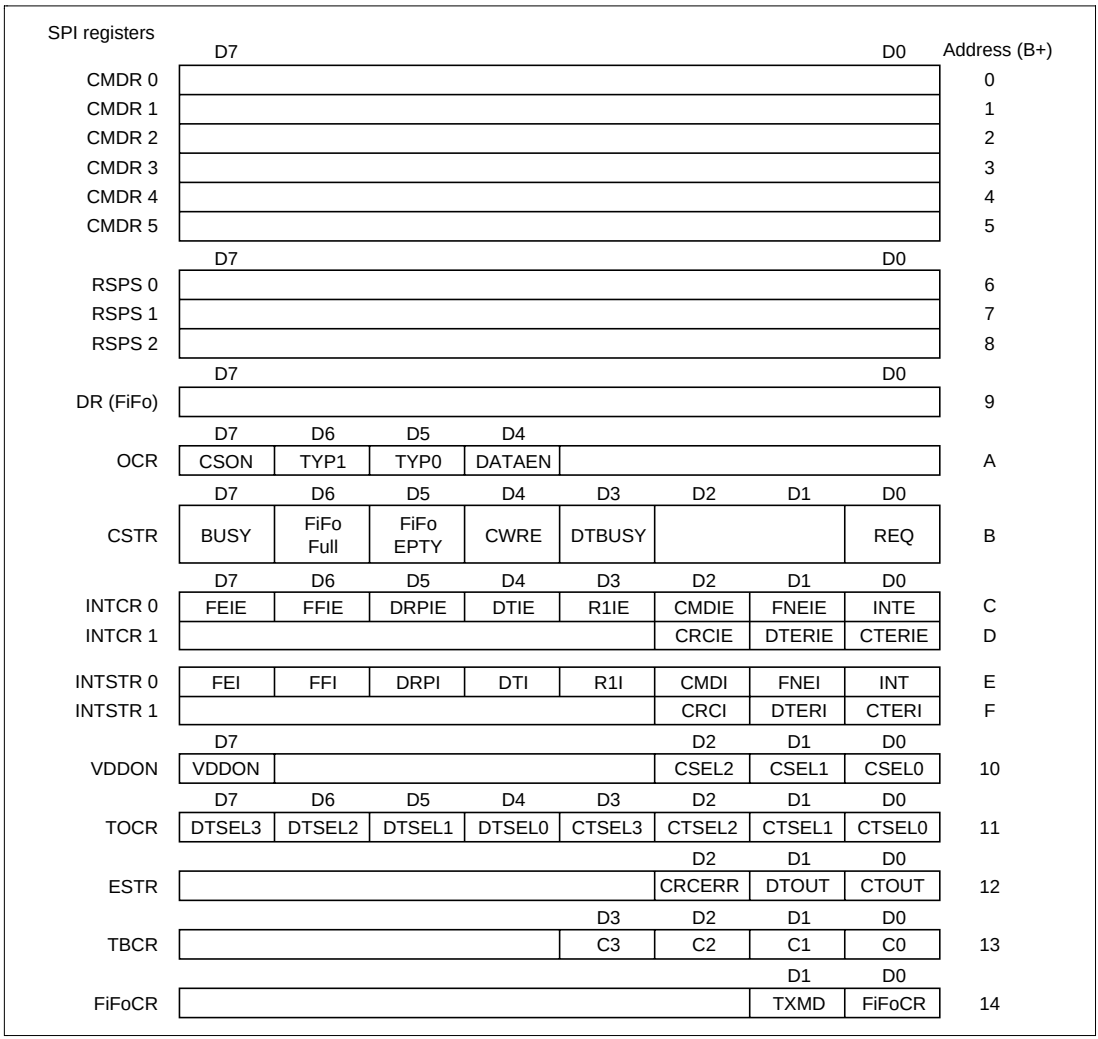


Figure 3.20 SPI Adapter Memory Map

3.6.2 Microcomputer Used

For voice compression and expansion, a microcomputer with appropriate processing power must be selected.

For this project, an SH7729—one of Hitachi's SuperH™ RISC microcomputers—was selected.

The microcomputer is operated on a 133 MHz internal clock, with a 33 MHz external bus clock and 33 MHz on-chip peripheral clock.

3.6.3 Audio Codec, etc.

An AC'97 compliant codec IC is used for the interface between audio input/output and the microcomputer, and 8-Mbyte SDRAM is provided as audio data buffer memory. A function has also been provided to monitor the system current dissipation, microcomputer current dissipation, or MMC card current dissipation, and display the value on a 7-SEG LED. This function is implemented by means of a current sensor and an H8/337 single-chip microcomputer.

(1) Software Processing Sequence

Figure 3.21 shows the software processing sequence for the newly developed player.

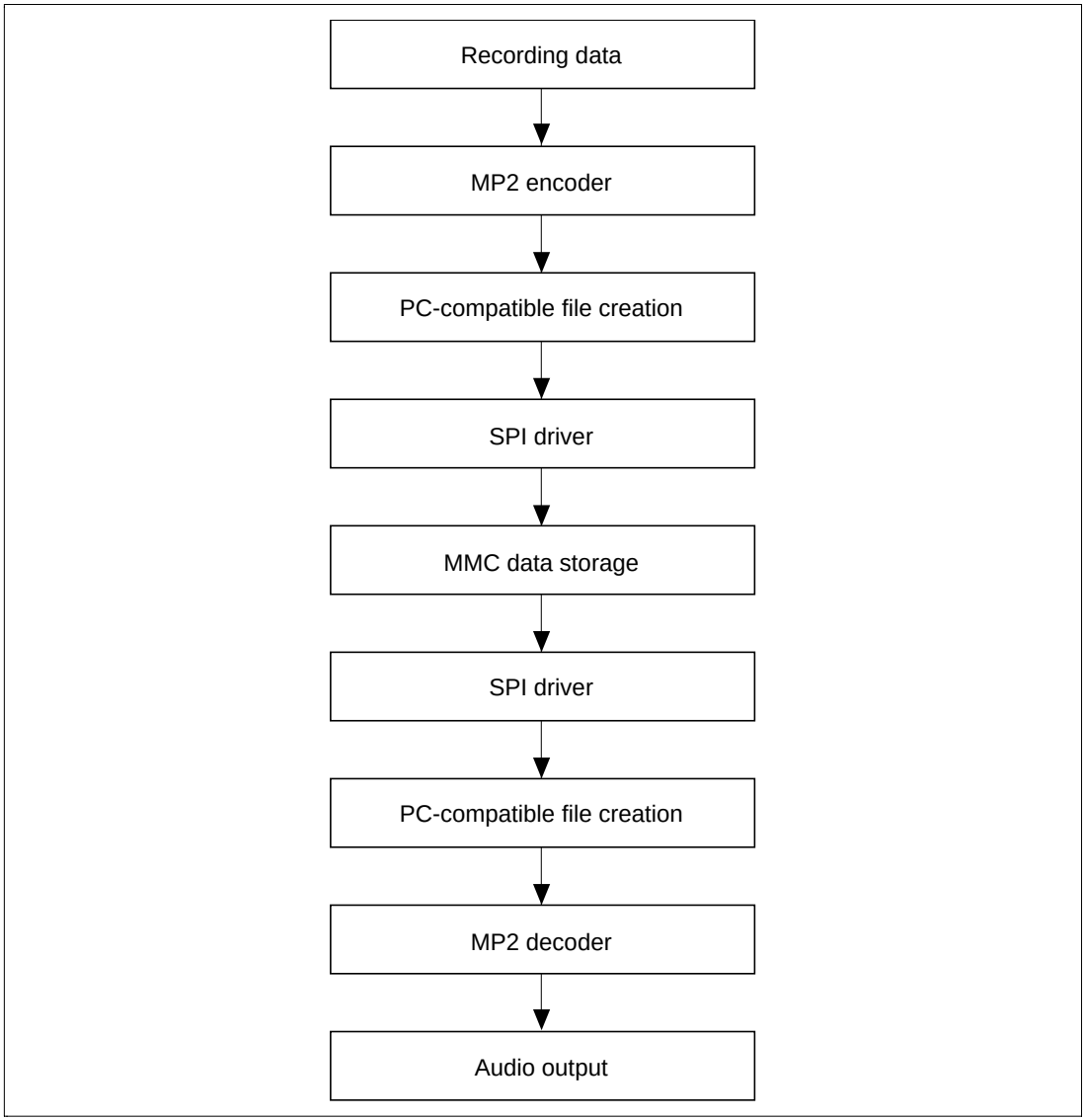


Figure 3.21 Recording/Playback Flowchart

Stereo Playback:

1. MP2 stereo audio compression is performed by host personal computer.
2. File is created in PC file format.
3. File is loaded from PC to player via serial port.
4. Audio data is stored in MMC.
5. Data stored in MMC is read and audio is output to earphone while executing stereo audio expansion.

Monaural Recording/Playback:

1. Microphone audio data is captured from codec IC.
2. Data compression is performed by means of MP2 encoder processing.
3. PC-compatible file is created, and is written to MMC by SPI driver. (Recording)
4. File in MMC is read by SPI driver. (Playback)
5. Audio is played back to earphone while performing MP2 decoding processing.

(2) Performance Evaluation

The MP2 player created in this case can store 17 minutes of stereo music data or 70 minutes of monaural voice data when using one 16-Mbyte MMC card as the storage medium.

Section 4 Notes on Design of a MultiMediaCard™ System

4.1 Introduction

An MMC is extremely small in size for a flash card. While major current flash cards such as PC cards and CompactFlash™ are PC-oriented in concept, the MMC is clearly more oriented toward consumer applications. This is shown by such features as:

- a simple 7-pin interface
- an extremely small and lightweight housing
- an architecture that allows for hot insertion and removal

The development of the MMC was thus based on excellent design concepts, but it is a new standard, and the environment is not yet one in which information and know-how for use of MMCs is becoming generally available or in which MMCs can be used by a large number of designers. Further, the fact that the MMC standard specification itself is in principle made available only to member enterprises of the MMCA (MultiMediaCard™ Association) makes it all the more difficult to obtain information. It is necessary to belong to the MMCA in order to obtain details of the standard, but an abbreviated version of the standard is available on the MMCA's Home Page, and can be downloaded free-of-charge by anyone interested.

MMCA Home Page: http://www.mmca.org/
Draft specification: http://www.mmca.org/specific.htm

This section is provided as reference material for those who have already acquired the specification and have embarked on the design process, or those currently looking into the possible use of MMCs or wanting to find out more about these cards.

4.2 Operating Modes

An MMC has the following two operating modes.

Generally, the host machine supports one or other of these modes.

1. MMC (MultiMediaCard™) mode
   Primary mode: Always supported by the MMC.

2. SPI (Serial Peripheral Interface) mode
   Optional mode: A mode designated as optional in the specification.

The MMC is placed in MMC mode immediately after powering on. SPI mode is entered from MMC mode by asserting the $\overline{CS}$ pin when CMD0 is issued. Mode switching is enabled only in the initialization procedure carried out immediately after powering on, and once operation is started in MMC mode, it is not possible to switch to SPI mode except by powering on again. The same applies to MMC mode when the card is operating in SPI mode.

The supported commands are different in each mode. MMC mode supports read/write commands for multiple sectors, while SPI mode supports only single-sector read/write commands.

The features of each mode are summarized in table 4.1. It is important to use the most appropriate mode for the system configuration under consideration.

Table 4.1 Features of Each Mode

	MMC (MultiMediaCard™) Mode	SPI (Serial Peripheral Interface) Mode
Bus system	Full-duplex 3-wire serial bus (CLK, CMD, and DAT pins)	Full-duplex 3-wire serial bus (CLK, DI, DO, and CS)
Operating frequency	Variable, 0 to 20 MHz	Variable, 0 to 20 MHz (0 to 5 MHz in MMC Specification Ver. 1.4)
Card selection method	Card selected by command Theoretically, up to 64 cards can be managed	Card selected by $\overline{CS}$ pin Theoretically, no limit on number of cards
Maximum number of cards connectable	30 (10 MHz) Up to 5 cards at 20 MHz	30 (10 MHz) Up to 5 cards at 20 MHz
Transfer commands	Single-sector/single-block unit transfer Multiple-sector/block unit transfer Sequential transfer	Single-sector/single-block unit transfer only

4.3 Bus Design

4.3.1 Bus Wiring Design

In an MMC, the MMC bus is driven in open-drain mode or push-pull mode, and pull-up resistances are essential for each bus line. Note that the bus will not function normally without resistances or if pull-down resistances are connected. Figure 4.1 shows the bus configuration. The pull-up resistance values are stipulated in the MMC specification, as shown in table 4.2.

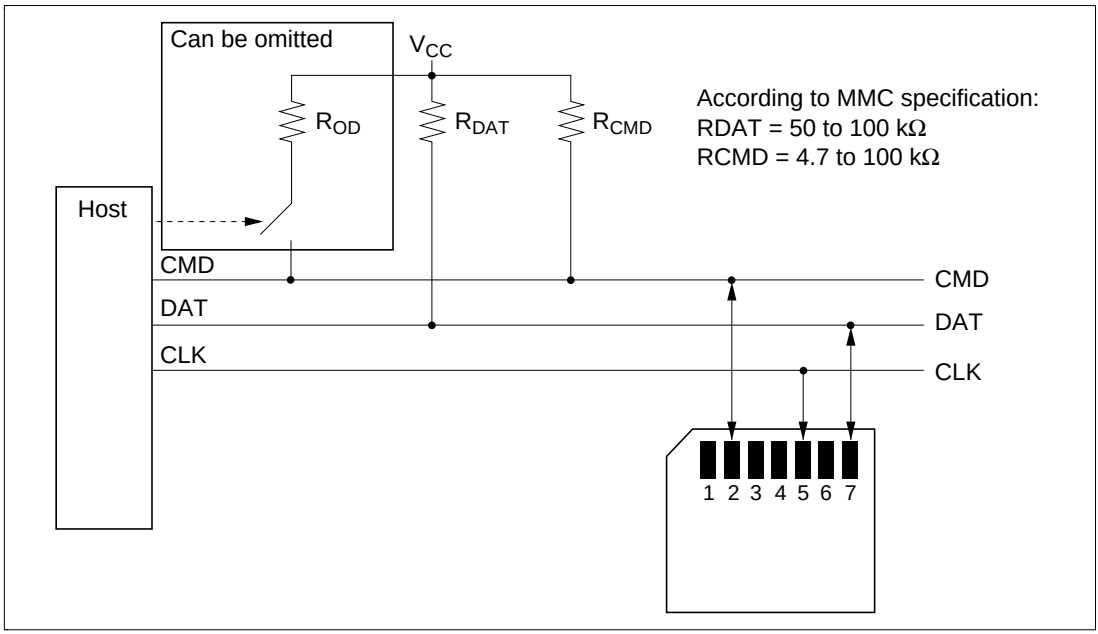


Figure 4.1 Bus Connection Diagram

Table 4.2 Pull-Up Resistance Values

Parameter	Symbol	Min	Max	Unit	Remark
CMD	R CMD	4.7	100	kΩ	To prevent bus floating
DAT	R DAT	50	100	kΩ	To prevent bus floating

The pull-up resistance values must be customized within the range given in the specification according to such conditions as the bus operating frequency, number of MMC slots, and the load capacitance of the cards. Some MMC bus conditions are shown below for reference.

Wiring capacitance CL per bus line:

- CL 250 pF {fpp ≤ 5 MHz, 30 cards connected}
- CL 100 pF {fpp ≤ 20 MHz, 10 cards connected}

Inductance L per bus line:

- Estimated at 16 nH {fpp ≤ 20 MHz}

The load capacitance C per MMC card has been set at 7 pF.

The above conditions should be observed to ensure stable operation.

4.3.2 Reducing Power Supply Noise

Generally, a bypass capacitor is used to lessen the influence of the peak current. The capacitor is located on the bus side, as shown in the figure below. The guideline value for the capacitance of the bypass capacitor, including the capacitance of the host controller (relative to V_{CC}) is: C_{buf} = 1 µF/slot.

As MMC bus lines are controlled by open-drain or push-pull operation, pull-up resistances are inserted on the V_{CC} side. It should be noted that as a result, there is comparatively more influence on power supply side V_{CC} fluctuations than GND fluctuations.

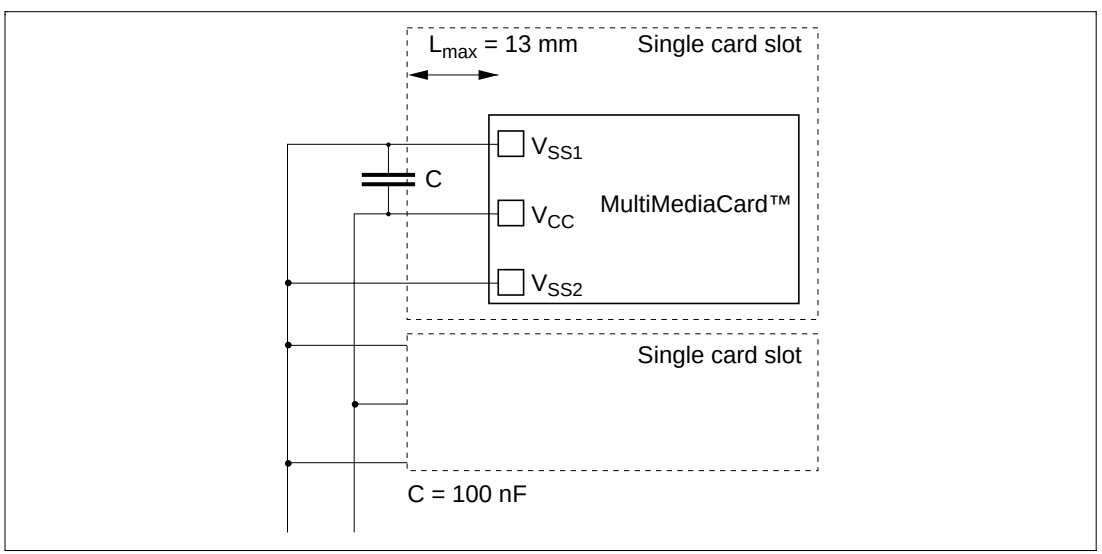


Figure 4.2 Placement of Bypass Capacitor

4.4 Cautions on Powering On, and Reset Operation

4.4.1 Powering On

Figure 4.3 shows the power-on sequence. An MMC has an internal power supply voltage detection circuit whereby, if the voltage falls below a certain level, a power-on reset is effected to prevent erroneous operation and internal initialization is performed automatically when the voltage reaches a sufficient level.

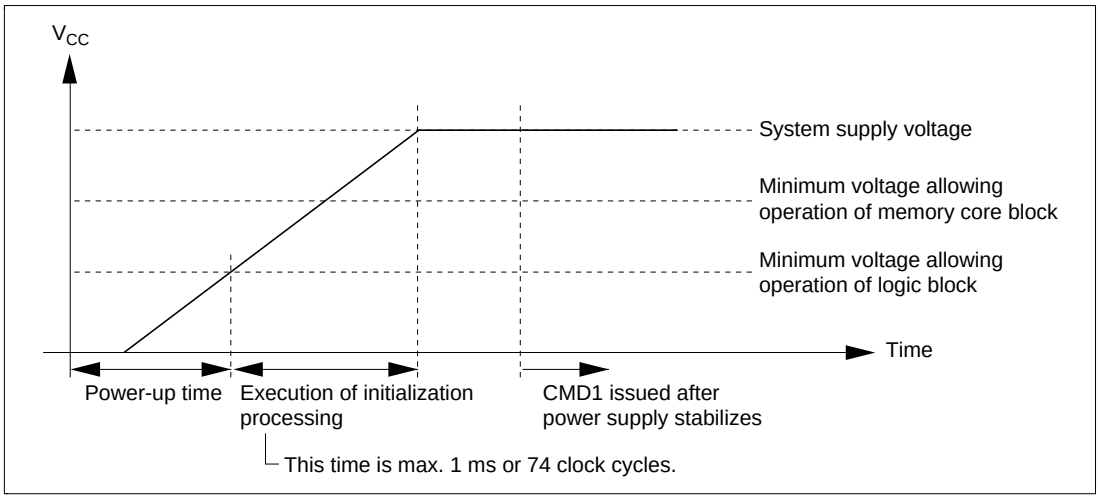


Figure 4.3 Power-On Sequence

When a sufficient voltage level has been reached after powering on, the host starts clock transmission to the MMC and issuance of the identification sequence is performed on the CMD line, but the MMCA spec stipulates that before this, the CMD line is to be held high for at least 1 ms or 74 clock cycles as a dummy clock period. When the MMC's internal automatic initialization procedure ends normally, the MMC (irrespective of hot insertion or cold insertion) goes to the Idle state in which only SEND_OP_COND (CMD1) and CMD0 are accepted.

Access to an MMC comprises two stages with respect to the power supply voltage, and is divided into a communication mode in which register access only is possible and an operation mode which also allows flash memory reads and writes. The voltage should be checked, as it varies for different products.

4.4.2 Reset Operation

An MMC reset must be carried out in accordance with the sequence defined in the specification. An MMC does not have an external reset pin; instead, a reset command is issued for this purpose. A reset command can be executed by means of the GO_IDLE_STATE (CMD0) command. When a reset command is executed, the MMC is forcibly switched to the Idle state regardless of its current state. However, an MMC in the Inactive state will ignore even this reset command. When this CMD0 command is issued, all the card's output pins go to the high-impedance state and each card's RCA register value is set to its initial value (0001h). The same applies at power-on, and after powering on a card is always in the Idle state. In this state, the only command that is accepted from a valid host is CMD1: SEND_OP_COND. However, in SPI mode, CMD1 does not have a valid operand, and therefore the host must poll the card state continuously by repeatedly sending CMD1 commands while the in-idle-state bit contained in the response is valid. When this bit becomes invalid, the relevant card terminates the initialization processing normally, indicating that preparation for acceptance of the next command has been completed. This difference in CMD1 processing in MMC mode and SPI mode must be noted.

4.5 Initial Settings up to Data Transfer in MMC Mode

Initial settings are used to carry out the following for all cards on the bus:

- acquisition and specification of operating conditions
- acquisition of card attribute information
- specification of relative addresses of cards on the bus

CMD1 is used for acquisition and specification of operating voltage conditions, CMD2 for acquisition of card attribute information, and CMD3 for specification of the cards' relative addresses. Figure 4.4 shows a state transition diagram for the initial setting procedure, and figure 4.5 shows the initial setting flowchart.

Initial settings in SPI mode are covered in section 4.6.

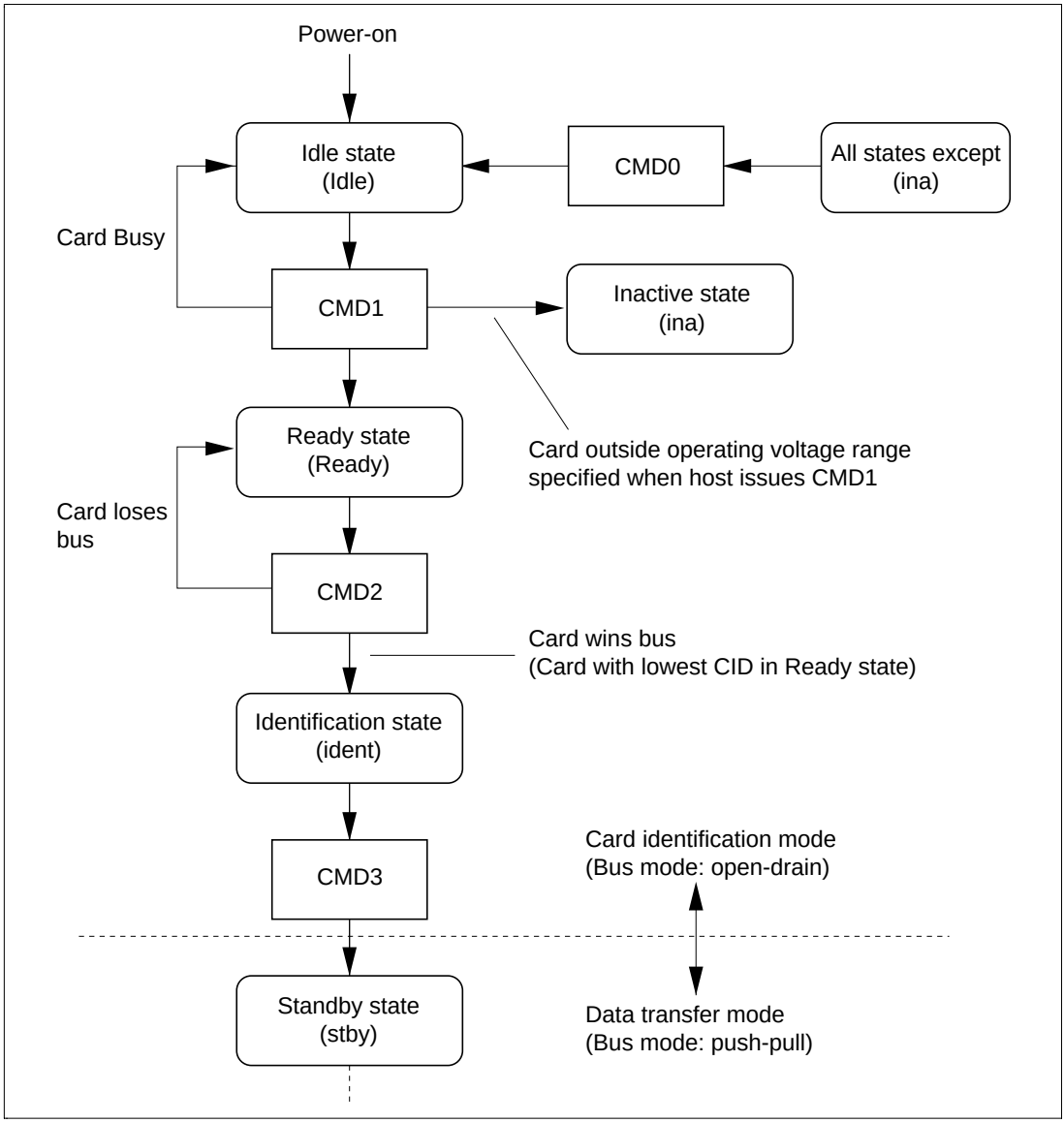


Figure 4.4 MMC State Diagram (Card Identification Mode)

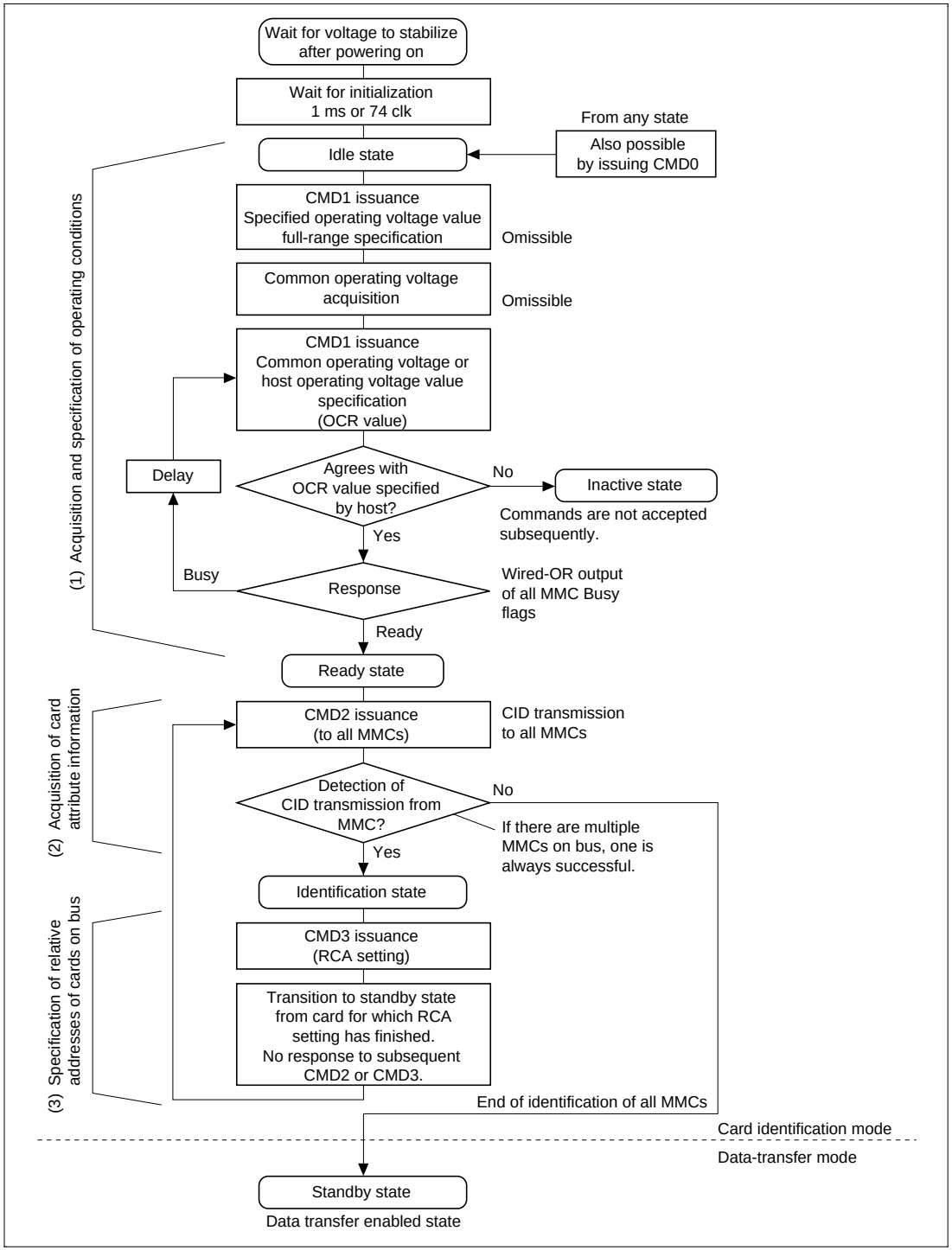


Figure 4.5 Initial Setting Flowchart

4.5.1 Acquisition and Specification of Operating Voltage Conditions

After powering on, MMCs go to the Idle state. In this state,

0. The host performs control in accordance with the frequency set during initialization,

1. The host issues SEND_OP_COND (CMD1).

2. The host must issue CMD1 commands and perform MMC polling until all the MMCs on the bus exit the power-up sequence. The operating voltage profile (OCR value) is used as the CMD1 argument. The voltage value (range) actually supplied to the MMCs should be specified as the OCR value.

   Note: Do not specify an OCR value of all 0s, as this means that a voltage is not supplied to the cards. If this is specified, the MMCs will go to the Inactive state in accordance with the specification, and will not respond to subsequent commands. If this specification is made by mistake, the power-on sequence must be repeated.

3. When CMD1 is issued to an MMC, the card's OCR value and Busy flag are returned as a response. If multiple MMCs are connected to the bus, the OCR values and Busy flag values output from each of the MMCs are wired-ORed and the result is reported to host. The Busy flag indicates that the relevant group of MMCs are in the middle of the power-up sequence, and the OCR value is the common divisor of the operable voltages for all the connected cards.

4. The host must continue issuing CMD1 commands, and must perform Busy bit polling, until all the Busy bits of the group of cards are cleared.

   Note: If CMD1 is issued to an MMC when all the MMCs' Busy bits have been cleared, the card will go to the Inactive state and may no longer operate normally.

5. When the Busy bit is cleared, it is known that the MMC (except for an MMC that has entered the Inactive state) has entered the Ready state.

Next, acquisition of card attribute information will be described.

4.5.2 Acquisition of Card Attribute Information and Specification of Relative Card Address on the Bus

The host must issue ALL_SEND_CID (CMD2) to all MMCs in the Ready state connected to the bus in order to acquire identification information (CID) for all these MMCs. The CID information is completely different for each MMC. The CMD line is used to transfer this data.

1. The host issues ALL_SEND_CID (CMD2) to MMCs in the Ready state.

2. When MMCs receive CMD2, they all begin sending CID information from the 5th clock onward.

3. As the CID information is different for each card, parts for which the MMC output and the card's bus output level are different appear during sending. A card for which this is detected returns to the Ready state and waits for a CMD2 command to be issued again.

4. As a result, for issuance of a single CMD2 command, only the single MMC with the lowest CID value succeeds in sending its CID, and goes to the Identification state.

5. There is then only one MMC in the Identification state on the bus.

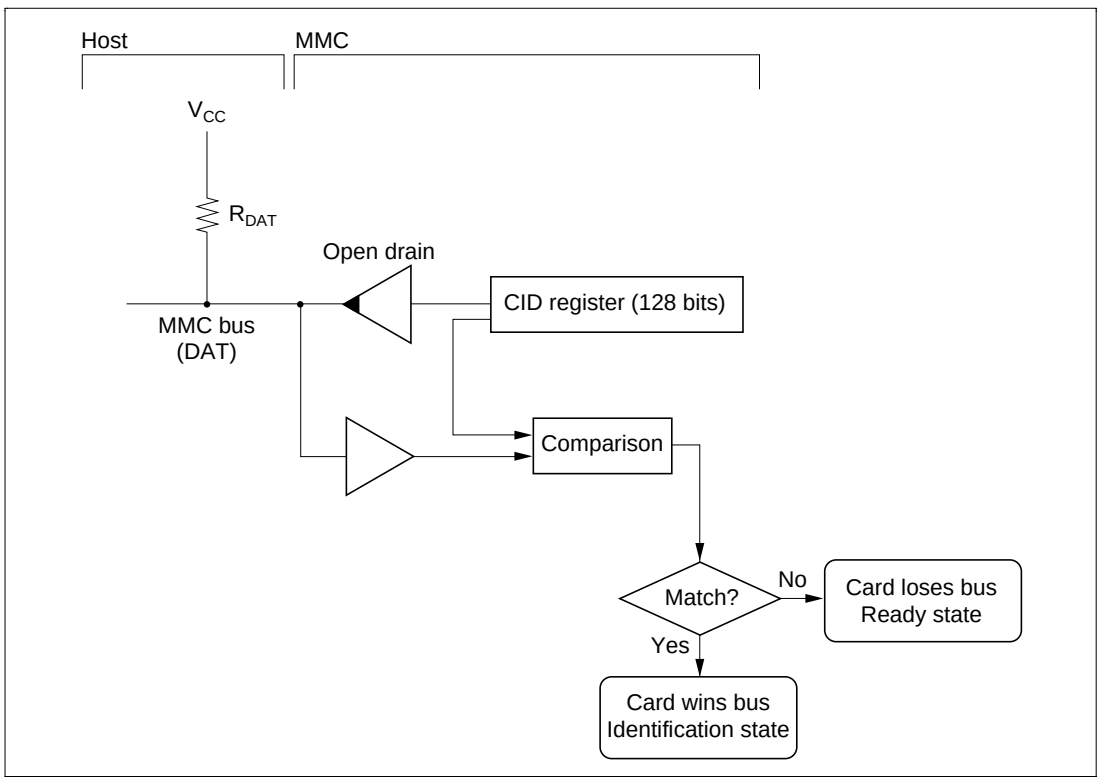


Figure 4.6 Open-Drain Architecture

6. The host now issues SET_RELATIVE_ADDR (CMD3), and sets the relative card address (RCA) for this MMC.

   Note: RCA values can be set without restriction, except that an RCA value of 0000h must not be specified. This setting is reserved in the MMC specification to enable all the MMCs on the bus to be deselected by setting RCA = 0000h when SELECT_DESELECT_ CARD (CMD7) is used in data transfer mode.
   The RCA initial value of an MMC when first powered on is 0001h.

7. The MMC for which the RCA is set switches to the standby state—one of the states in data transfer mode—and thereafter does not respond to CMD2 or CMD3 commands. The host repeats steps 1 to 7, and sets the RCA value for all the MMCs.

8. The host determines whether or not all the cards on the bus have been identified by issuing a CMD2 command and checking for no response from any MMC (i.e. no CID transmission). The timeout criterion in this case is non-detection of a start bit 5 clocks after sending CMD2.

9. The MMCs go to the data transfer state. At this point, the clock frequency can be changed to 20 MHz or below.

10. The CSD register is read.

11. Settings such as a change of read sector size are made.

12. Processing such as data reading and writing can now be carried out.

After this procedure is completed, data transfer commands such as data read and write commands can be used on the MMCs.

4.6 Initial Settings up to Data Transfer in SPI Mode

Immediately after powering on, an MMC starts up in MMC mode. Therefore, unless specific processing is executed, an MMC is normally in MMC mode, and SPI mode may be considered as a troublesome alternative. However, since SPI mode does not include the concept of “states” used in MMC mode, and card selection uses a $\overline{CS}$ pin approach similar to that used with conventional memory devices, switching to SPI mode actually allows simpler card handling than MMC mode.

Note that, although an MMC goes to the Idle state after powering on, it is still in MMC mode.

In this state, the host

1. drives $\overline{CS}$ high, making the card nonactive,
2. issues at least 76 dummy clocks for MMC initialization, and
3. drives $\overline{CS}$ low and transmits a CMD0 command (GO_IDLE_STATE).

   Note: At this point, the card is still in MMC mode, and so a CRC is necessary. The command 0 format is “40, 00, 00, 00, 00, 95”, of which “95” is the CRC.
   Also, according to the MMC specification, an MMC in the Idle state only accepts CMD0 and CMD1. If the user wishes to turn the CRC function on or off with CMD59, this must be done after issuing CMD1.

4. At this point the mode changes to SPI mode. The host waits for an R1 response from the MMC.
5. If the R1 response is 01h, the sequence proceeds to step 8; if the R1 response is a value other than 01h, an error is judged to have occurred and error handling is performed. Processing might generally be to provide an error indication or to re-execute the sequence starting from powering-on.
6. If there is no response, this is taken to indicate a timeout and error handling is performed.
7. The host issues CMD1 (SEND_OP_CMD) and polls for an R1 response from an MMC.
8. If the R1 response is 00h, the sequence proceeds to step 9 and polling is performed by issuing CMD1 commands again until a 00h response is received. If the R1 response is not 00h or 01h, an error is judged to have occurred and error handling is performed. Processing might generally be to provide an error indication or to re-execute the sequence starting from powering-on. If there is no response, this is taken to indicate a timeout and error handling is performed.

9. The mode changes to data transfer mode. At this point, the clock frequency can be changed to 20 MHz or below.

   Note: Some MMCs compliant with MMC Specification Ver. 1.4 must be operated at 0 to 5 MHz in SPI mode.

10. The CSD register is read. Unlike MMC mode, in which special commands are used to read register values, in SPI mode CSD and CID register contents are acquired by means of an ordinary simple read block transfer. In response to a request from the host, an MMC returns data by means of a standard Response token. That is, a request is answered with a 16-byte data block and an associated 16-bit CRC.

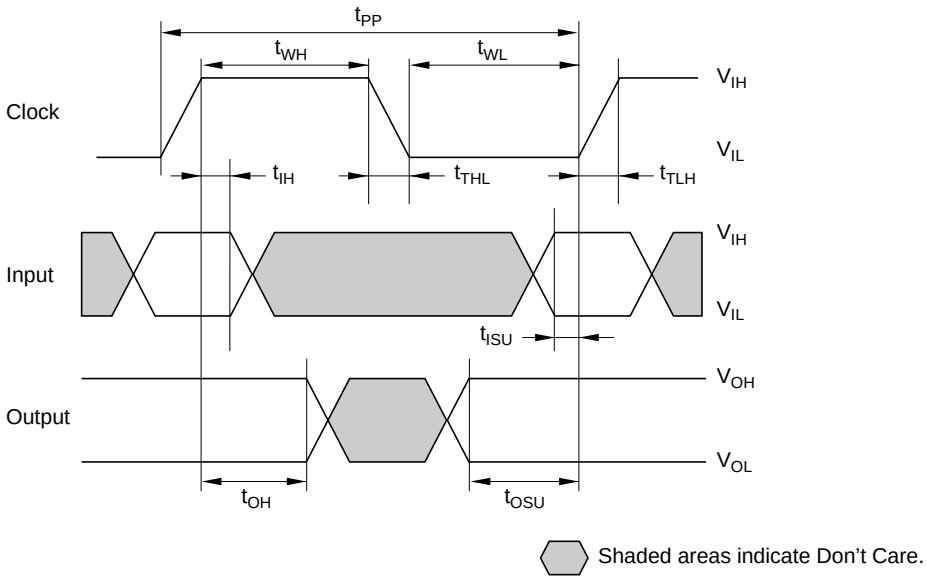
11. Settings such as a change of read sector size are made.

12. At this point read/write accesses are enabled.

13. When powering off, check that the cards are in the Ready state and drive CS high before cutting the power.

4.7 Timing Design

MMC timing specifications are shown in figure 4.7. According to the MMC specification, input and output of each signal is all stipulated with respect to the rise of the clock, and the MMC performs all signal data capture and output at the rise of the clock signal. Shaded areas in the figure are Don’t Care zones; the host should not perform data capture within these areas.



According to MMC specification

Parameter	Symbol	Min	Max	Unit	Remark
Clock CLK (All specifications are stipulated with respect to V_{IH} and V_{IL}.)					
Frequency in data transfer (10-card stack)	f_{PP}	0	20	MHz	CL ≤ 100 pF (10 cards)
Frequency in data transfer (30-card stack)	f_{PP}	0	5	MHz	CL ≤ 250 pF (30 cards)
Frequency in initialization	f_{OD}	0	400	kHz	
Clock low time	t_{WL}	10		ns	CL ≤ 100 pF (10 cards)
Clock high time	t_{WH}	10		ns	
Clock rise time	t_{TLH}		10	ns	
Clock fall time	t_{THL}		10	ns	
Clock low time	t_{WL}	50		ns	CL ≤ 250 pF (30 cards)
Clock high time	t_{WH}	50		ns	
Clock rise time	t_{TLH}		50	ns	
Clock fall time	t_{THL}		50	ns	
Input setup time	t_{ISU}	3		ns	
Input hold time	t_{IH}	3		ns	
Output setup time	t_{OSU}	5		ns	
Output hold time	t_{OH}	5		ns	

Figure 4.7 MMC Timing Chart

A. Development Support

Figure A.1 shows an outline flowchart for the development of a system incorporating MultiMediaCard™. As shown in the figure, various support tools are available for each phase of the development process.

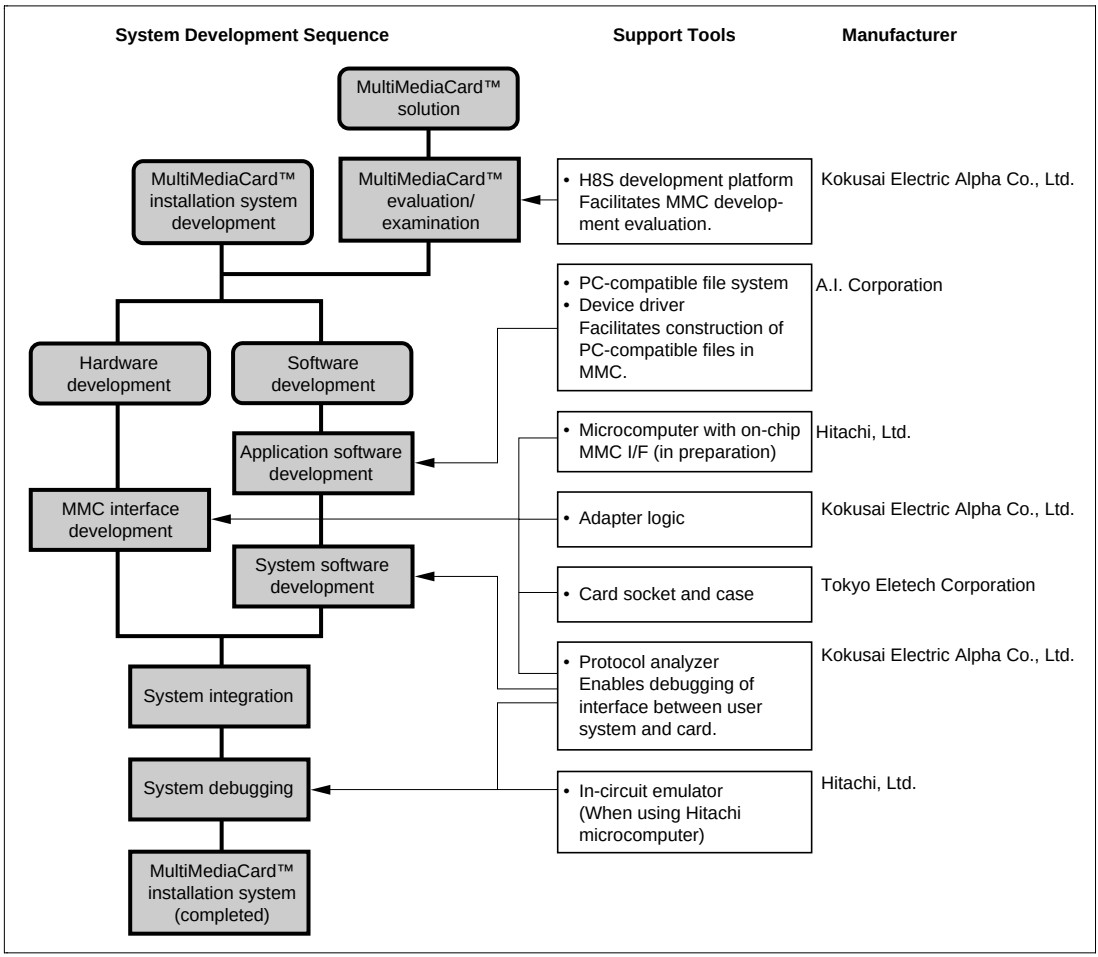


Figure A.1 MultiMediaCard™ Development Flowchart and Available Tools

Overseas Support

For inquiries concerning MultiMediaCards™, contact:

- USA
  Hitachi Semiconductor (America) Inc.
  179 East Tasman Drive San Jose CA 95134-1620
  http://www.hitachi.com/semiconductor

- Europe
  Hitachi Europe GmbH
  Dornacher Straße 3 D-85622 Feldkirchen bei München Germany
  http://www.hitachi-eu.com

- Taiwan
  Hitachi Asia Ltd.
  4th floor, No.167, Tun Hwa N. Road, Hung-Kuo Building, Taipei, Taiwan
  http://www.hitachi.com.tw

- Korea
  Hitachi Asia (Hong kong) Ltd.
  Hanji Group Bldg. 191, 2-ka, Hangang-ro, Yongsan-ku, Korea
  http://www.hitachi.com.hk/eng

- Japan
  Hitachi Ltd.
  Media Flash Memory Business Unit Product Marketing Team
  Nippon Bldg., Ohtemachi 2-chome, Chiyoda-ku, Tokyo 100-0004, Japan
  http://www.hitachi.co.jp/Sicd/index.htm

For inquiries concerning the protocol analyzer or development platform, contact:

- USA, Europe, Asia
  Nissei Sangyo Co., Ltd. and its Overseas Affiliated Companies.

For inquiries concerning adapter logic contact:

TBD

For inquiries concerning the file system or driver software, contact:

TBD

For inquiries concerning the MMC socket and case, contact:

- USA, Europe, Asia
  Nissei Sangyo Co., Ltd. and its Overseas Affiliated Companies.

- Taiwan
  Daimaru Kogyo, Ltd. Taipei Branch
  7F-3, No.128 Sec.3, Min-sheng East Rd., Taipei 105, Taiwan R.O.C.
  http://www.daimaru-kogyo.com/

- Japan
  Tokyo Eletech Corporation Head Office
  3-10 Akihabara, Taito-ku, Tokyo 110-0006, Japan
  http://www.tetc.co.jp/

Nissei Sangyo contact:

- Europe
  Mr. M. Kleiner
  Nissei Sangyo GmbH (Deutschland)
  Dornacher Str.3e 85622 Feldkirchen, F.R. Germany
  Tel: +49(0)89-99180287
  Em: martin.kleiner@nissei-eu.com

- America
  Mr. T. Komai
  Nissei Sangyo America, Ltd. Head Office
  2850 Golf Road, Suite 200, Rolling Meadows, IL 60008, USA
  Tel: +1-847-545-2230
  Em: tak.komai@nissei.com

- Asia
  Taiwan: Mr. C.H. Wang
  Nissei Sangyo Co., Ltd. Taipei Branch Office
  Shin Kang Chung Shan Bldg., 10th fl., 44, Sec.2, Chung Shan N. Road, Taipei, Taiwan
  Tel: +886-2-2522-6921
  Em: chwang@gcn.net.tw

Singapore: Mr. J. Liew
Nissei Sangyo (Singapore) Pet. Ltd.
3 Killiney Road, No.07-05/09, Winsland House, Singapore 239519, Singapore
Tel: +65-737-2011
Em: jack-liew@nissei.com.sg

Korea: Mr. S.S. Lee
Nissei Sangyo Co., Ltd. Seoul Branch Office
No.641 6th fl., The Chamber of Commerce & Industry
Industry Bldg., 45, 4-Ka, Namdaemun-Ro, Chung-Ku, Seoul, Korea
Tel: +82-2-773-5692
Em: lees-sel@gr.nisseisg.co.jp

Hong Kong: Mr. Y. Araiso
Nissei Sangyo Hong Kong Ltd.
Suite 1208-11, North Tower, World Finance Centre, Harbour City, Canton Road,
T.S.T., Kowloon, Hong Kong
Tel: +852-2737-4740
Em: araiso @nissei-hk.com.hk

MultiMediaCard™

Publication Date: 1st Edition, September 2000
2nd Edition, September 2001
Published by: Customer Service Division
Semiconductor & Integrated Circuits
Hitachi, Ltd.
Edited by: Technical Documentation Group
Hitachi Kodaira Semiconductor Co., Ltd.