1 MultiMediaCard/Secure Digital Controller

1.1 Overview

The MultiMediaCard (MMC) is a universal low cost data storage and communication media that is designed to cover a wide area of applications such as electronic toys, organizers, PDAs, smart phones, and so on. The MMC communication is base on an advanced 7 pin serial bus designed to operate in a low voltage range, at medium speed (20 Mbps).

The Secure Digital (SD) card is an evolution of MMC, with an additional 2 pins and the same form factors. It is specifically designed to meet the security, capacity, performance, and environmental requirements inherent in newly emerging audio and video consumer electronic devices. The physical form factor, pin assignment, and data transfer protocol are forward compatible with the MultiMediaCard with some additions. An SD card can be categorized as SD memory or SD I/O card, commonly known as SDIO. A memory card invokes a copyright protection mechanism that complies with the security of the SDMI standard and is faster and capable of higher memory capacity. The SDIO card provides high-speed data I/O with low-power consumption for mobile electronic devices.

Features of the MultimediaCard/Secure Digital Controller include the following:

- Fully compatible with the *MMC System Specification version 3.3*
- Fully compatible with the *SD Memory Card Specification 1.01* and *SD I/O Specification 1.0* with 1 command channel and 4 data channels
- 20-80 Mbps maximum data rate
- Built-in programmable frequency divider for MMC/SD bus
- Maskable hardware interrupt for SDIO interrupt, internal status and FIFO status
- 16-entry x 32-bit built-in data FIFO
- Password protection of cards
- Multi-SD function support including multiple I/O and combined I/O and memory
- Up to 7 I/O functions plus one memory supported on single SDIO card
- IRQ supported enable card to interrupt MMC/SD controller
- Single or multi block access to the card including erase operation
- Stream access to the card
- Supports SDIO read wait, interrupt detection during 1-bit or 4-bit access
- The maximum block length is 2048 bytes

1.2 Block Diagram

MMC/SD Controller Block Diagram

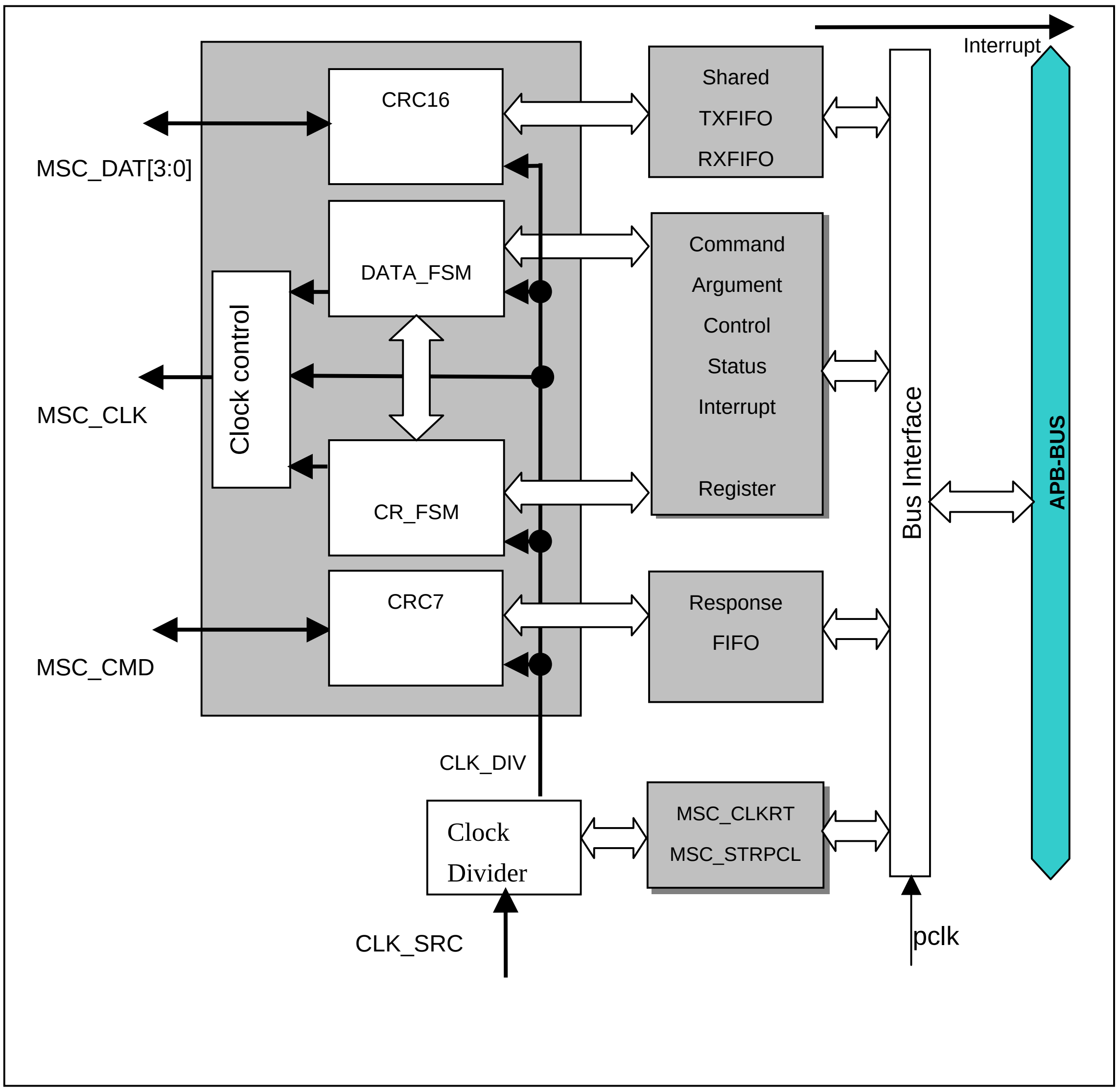
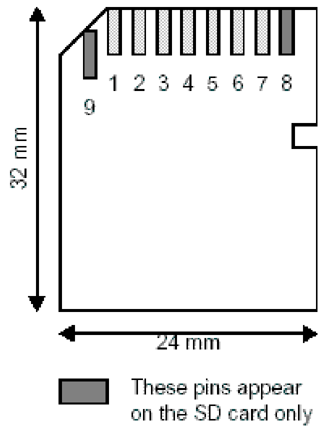


Figure 1-1 MMC/SD Controller Block Diagram

1.3 MMC/SD Controller Signal I/O Description

The MMC and SD cards are 7- or 9- pin cards that operate as external memory storage. The pin assignment and form factor are shown in Table 1-1.

Table 1-1 MMC/SD Controller Signal Description

Form factor and pinout	Pin number	MMC card	SD card	
			1-bit mode	**4-bit mode**
 1 2 3 4 5 6 7 8 9 32 mm 24 mm These pins appear on the SD card only	1	Reserved	Not Used	Data Line [Bit 3]
	2	Command/Response (CMD)		
	3	Supply Voltage Ground (Vss1)		
	4	Supply Voltage (Vdd)		
	5	Clock (CLK)		
	6	Supply Voltage Ground (Vss2)		
	7	Data Line [Bit 0]		
	8	▨	Interrupt (IRQ)	Data Line [Bit 1] or Interrupt (IRQ)
	9	▨	ReadWait(RW)	Data Line [Bit 2] or ReadWait (RW)

MSC and the card communication over the CMD and DATA line is base on command and data bit streams which are initiated by a start bit and terminated by a stop bit.

- **Command**: a command is a token, which starts an operation. A command is sent from MSC either to a single card (addressed command) or to all connected cards (broadcast command). A command is transferred serially on the CMD line. Each command token is preceded by a start bit ('0') and succeeded by an end bit ('1'). The total length is 48 bits and protected by CRC bits.

Table 1-2 Command Token Format

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

- **Response**: a response is a token which is sent from an addressed card, or (synchronously) from all connected cards, to MSC as an answer to a previously received command. A response is transferred serially on the CMD line. Response tokens have varies coding schemes depending on their content.

- **Data:** data can be transferred from the card to MSC or vice versa. Data is transferred via the data line. Data transfers to/from the SD Memory Card are done in blocks. Data blocks always succeeded by CRC bits. Single and multiple block operations are defined. Note that the Multiple Block operation mode is better for faster write operation. A multiple block transmission is terminated when a stop command follows on the CMD line. Data transfer can be configured by the MSC to use single or multiple data lines.

Table 1-3 MMC/SD Data Token Format

Description	Start bit	Data	CRC16	End bit
Stream Data	0	X	no CRC	1
Block Data	0	X	X	1

1.4 Register Description

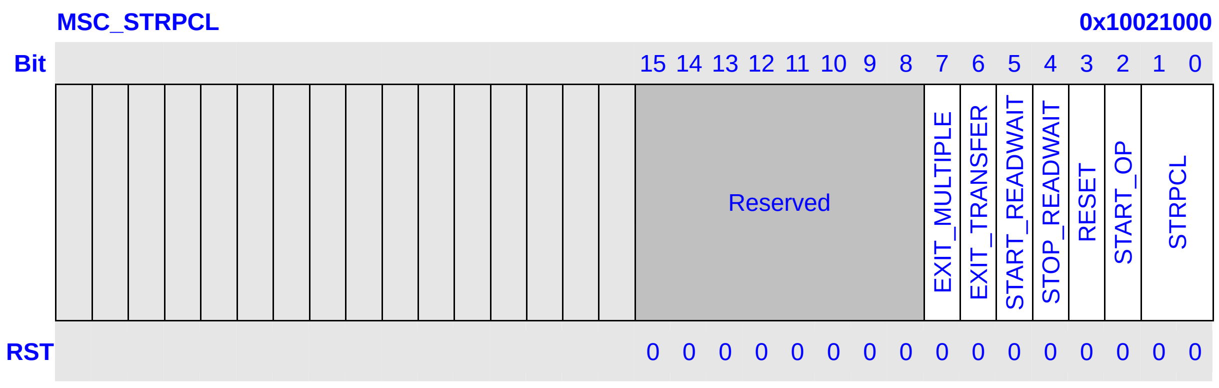
The MMC/SD controller is controlled by a set of registers that the application configures before every operation. The Table 1-4 lists all the MSC registers.

Table 1-4 MMC/SD Controller Registers Description

Name	RW	Reset Value	Address	Access Size
MSC_STRPCL	W	0x0000	0x10021000	16
MSC_STAT	R	0x00000040	0x10021004	32
MSC_CLKRT	RW	0x0000	0x10021008	16
MSC_CMDAT	RW	0x00000000	0x1002100C	32
MSC_RESTO	RW	0x40	0x10021010	16
MSC_RDTO	RW	0xFFFF	0x10021014	16
MSC_BLKLEN	RW	0x0000	0x10021018	16
MSC_NOB	RW	0x0000	0x1002101C	16
MSC_SNOB	R	0x????	0x10021020	16
MSC_IMASK	RW	0x00FF	0x10021024	16
MSC_IREG*1	RW	0x0000	0x10021028	16
MSC_CMD	RW	0x00	0x1002102C	8
MSC_ARG	RW	0x00000000	0x10021030	32
MSC_RES	R	0x????	0x10021034	16
MSC_RXFIFO	R	0x????????	0x10021038	32
MSC_TXFIFO	W	0x????????	0x1002103C	32

Note: *1: Writing MSC_IREG is used to clear interrupt source and only three interrupt sources can be cleared by this way.

1.4.1 Start/stop MMC/SD clock Register (MSC_STRPCL)



Bits	Name	Description	RW
15:8	Reserved		R
7	EXIT_MULTIPLE	If CMD12 or CMD52 (I/O abort) is to be sent to terminate multiple block read/write in advance, set this bit to 1. 0 – No effect.	W

		1 – Exit from multiple block read/write.	
6	EXIT_TRANSFER	**Only used for SDIO suspend/resume and MMC stream read.** For SDIO, after suspend is accepted, set this bit with 1. For MMC, after the expected number of data are received, set this bit with 1. 0 – No effect. 1 – Exit from multiple block read/write after suspend is accepted, or exit from stream read.	W
5	START_READWAIT	Only used for SDIO ReadWait. Start the ReadWait cycle. 0 – No effect. 1 – Start ReadWait.	W
4	STOP_READWAIT	Only used for SDIO ReadWait. Stop the ReadWait cycle. 0 – No effect. 1 – Start ReadWait.	W
3	RESET	Resets the MMC/SD controller. 0 – No effect. 0 – Reset the MMC/SD controller.	W
2	START_OP	This bit is used to start the new operation. When starting the clock, this bit can be 1. When stopping the clock, this bit can only be 0. 0 – Do nothing. 1 – Start the new operation.	W
1:0	CLOCK_CONTROL	These bits are used to start or stop clock. 00 – Do nothing. 01 – Stop MMC/SD clock 10 – Start MMC/SD clock 11 – Reserved	W

1.4.2 MSC Status Register (MSC_STAT)

MSC_STAT **0x10021004**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																IS_RESETTING	SDIO_INT_ACTIVE	PRG_DONE	DATA_TRAN_DONE	END_CMD_RES	DATA_FIFO_AFULL	IS_READWAIT	CLK_EN	DATA_FIFO_FULL	DATA_FIFO_EMPTY	CRC_RES_ERR	CRC_READ_ERROR	CRC_WRITE_ERROR		TIME_OUT_RES	TIME_OUTREAD
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bits	Name	Description	RW
31:16	Reserved		R
15	IS_RESETTING	MSC is resetting after power up or MSC_STRPCL[RESET] is written with 1 0 – Reset has been finished. 1 – Reset has not been finished.	R
14	SDIO_INT_ACTIVE	Indicates whether an interrupt is detected at the SD I/O card. A separate acknowledge command to the card is required to clear this interrupt. 0 – No interrupt detected. 1 – The interrupt from SDIO is detected.	R
13	PRG_DONE	Indicates whether card has finished programming. 0 – Card has not finished programming and is busy. 1 – Card has finished programming and is not busy.	R
12	DATA_TRAN_DONE	Indicates whether data transmission to card has completed. 0 – Data transmission to card has not completed. 1 – Data transmission to card has completed.	R
11	END_CMD_RES	End command-response sequence or command sequence. 0 – Command and response/no-response sequence has not completed. 1 – Command and response/no-response sequence has completed.	R
10	DATA_FIFO_AFULL	Indicates whether data FIFO is almost full (The number of words >= 15). For reading data from card, use this bit. 0 – Data FIFO is not full. 1 – Data FIFO is full.	R
9	IS_READWAIT	Indicates whether SDIO card has entered ReadWait State 0 – Card has not enter ReadWait. 1 – Card has enter ReadWait.	R
8	CLK_EN	Clock enabled. 0 – Clock is off. 1 – Clock is on.	R
7	DATA_FIFO_FULL	Indicates whether data FIFO is full. For reading data from card, do not use this bit, because it almost keeps to be 0. 0 – Data FIFO is not full. 1– Data FIFO is full.	R
6	DATA_FIFO_EMPTY	Indicates whether data FIFO is empty. 0 – Data FIFO is not empty. 1– Data FIFO is empty.	R
5	CRC_RES_ERR	Response CRC error. 0 – No error on the response CRC.	R

		1– CRC error occurred on the response.	
4	CRC_READ_ERROR	CRC read error. 0 – No error on received data. 1– CRC error occurred on received data	R
3:2	CRC_WRITE_ERROR	CRC write error. 00 – No error on transmission of data. 01 – Card observed erroneous transmission of data. 10 – No CRC status is sent back. 11 – Reserved	R
1	TIME_OUT_RES	Response time out. 0 – Card response has not timed out. 1 – Card response has time out.	R
0	TIME_OUT_READ	Read time out. 0 – Card read data has not timed out. 1 – Card read data has timed out.	R

1.4.3 MSC Clock Rate Register (MSC_CLKRT)

The MSC_CLKRT register specifies the frequency division of the MMC/SD bus clock. The software is responsible for setting this register.

MSC_CLKRT **0x10021008**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved													CLK_RATE		
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:3	Reserved		R
2:0	CLK_RATE	Clock rate. 000 – CLK_SRC. 001 – 1/2 of CLK_SRC. 010 – 1/4 of CLK_SRC. 011 – 1/8 of CLK_SRC. 100 – 1/16 of CLK_SRC. 101 – 1/32 of CLK_SRC. 110 – 1/64 of CLK_SRC. 111 – 1/128 of CLK_SRC.	WR

1.4.4 MMC/SD Command and Data Control Register (MSC_CMDAT)

MSC_CMDAT **0x1002100C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																Reserved				IO_ABORT	BUS_WIDTH		DMA_EN	INIT	BUSY	STREAM_BLOCK	WRITE_READ	DATA_EN	RESPONSE_FORMAT		
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:12	Reserved		R
11	IO_ABORT	Specifies the current command is used to abort data transfer. **Only used for SDIO.** 0 – Nothing. 1 – The current command is used to abort transfer.	WR
10:9	BUS_WIDTH	Specifies the width of the data bus. 00 – 1-bit. 01 – Reserved. 10 – 4-bit. 11 – Reserved.	WR
8	DMA_EN	DMA mode enables. When DMA mode is used, this bit is also a mask on RXFIFO_RD_REQ and TXFIFO_WR_REQ interrupts. 0 – Program I/O. 1 – DMA mode.	WR
7	INIT	80 initialization clocks 0 – Do not precede command sequence with 80 clocks. 1 – Precede command sequence with 80 clocks.	W
6	BUSY	Specifies whether a busy signal is expected after the current command. This bit is for no data command/response transactions only. 0 – Not expect a busy signal. 0 – Expects a busy signal. If the response is R1b, then set it.	WR
5	STREAM_BLOCK	Stream mode 0 – Data transfer of the current command sequence is not in stream mode. 1– Data transfer of the current command sequence is in	WR

		stream mode.	
4	WRITE_READ	Specifies that the data transfer of the current command is a read or write operation. 0 – Specifies that the data transfer of the current command is a read operation. 1 – Specifies that the data transfer of the current command is a write operation.	WR
3	DATA_EN	Specifies whether the current command includes a data transfer. It is also used to reset RX_FIFO and TX_FIFO. 0 – No data transfer with current command. 1 – Has data transfer with current command. It is also used to reset RX_FIFO and TX_FIFO.	WR
2:0	RESPONSE_FORMAT	These bit specify the response format for the current command. 000 – No response. 001 – Format R1 and R1b. 010 – Format R2. 011 – Format R3. 100 – Format R4. 101 – Format R5. 110 – Format R6. 111 – Reserved.	WR

1.4.5 MMC/SD Response Time Out Register (MSC_RESTO)

MSC_RESTO **0x10021010**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								RES_TO							
RST	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bits	Name	Description	RW
15:8	Reserved		R
7:0	RES_TO	Specifies the number of MSC_CLK clock counts between the command and when the MMC/SD controller turns on the time-out error for the received response. The default value is 64.	WR

1.4.6 MMC/SD Read Time Out Register (MSC_RDTO)

MSC_RDTO **0x10021014**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	READ_TO															
RST	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description	RW
15:0	READ_TO	Specifies the number of clocks between the command and when the MMC/SD host controller turns on the time-out error for the received data. The unit is CLK_SRC / 256.	WR

1.4.7 MMC/SD Block Length Register (MSC_BLKLEN)

MSC_BLKLEN **0x10021018**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved				BLK_LEN											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:12	Reserved		R
11:0	BLK_LEN	Specifies the number of bytes in a block, and is normally set to 0x200 for MMC/SD data transactions. The value Specified in the cards CSD.	WR

1.4.8 MSC/SD Number of Block Register (MSC_NOB)

MSC_NOB **0x1002101C**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NOB															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:0	NOB	Specifies the number of blocks in a data transfer. One block is a possibility.	WR

1.4.9 MMC/SD Number of Successfully-transferred Blocks Register (MSC_SNOB)

In block mode, the MSC_SNOB register records the number of successfully transferred blocks. If the last block has CRC error, this register also summaries it. It is used to query blocks for multiple block transfer.

MSC_SNOB **0x10021020**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MSC_SNOB															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
15:0	MSC_SNOB	Specify the number of successfully transferred blocks for a multiple block transfer.	R

1.4.10 MMC/SD Interrupt Mask Register (MSC_IMASK)

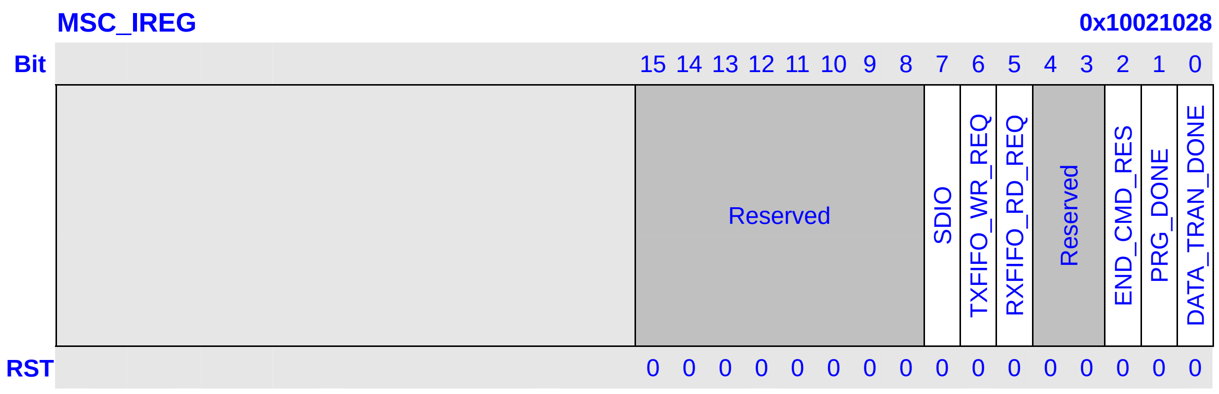
MSC_IMASK **0x10021024**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								SDIO	TXFIFO_WR_REQ	RXFIFO_RD_REQ	Reserved		END_CMD_RES	PRG_DONE	DATA_TRAN_DONE
RST	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Bits	Name	Description	RW
15:8	Reserved		R
7	SDIO	Mask the interrupt from the SD I/O card. 0 – Not masked. 1 – Masked.	WR
6	TXFIFO_WR_REQ	Mask the Transmit FIFO write request interrupt. 0 – Not masked. 1 – Masked.	WR
5	RXFIFO_RD_REQ	Mask the Receive FIFO read request interrupt. 0 – Not masked. 1 – Masked.	WR
4:3	Reserved		R
2	END_CMD_RES	Mask the End command response interrupt. 0 – Not masked. 1 – Masked.	WR
1	PRG_DONE	Mask the Programming done interrupt. 0 – Not masked. 1 – Masked.	WR
0	DATA_TRAN_DONE	Mask the Data transfer done interrupt. 0 – Not masked. 1 – Masked.	WR

1.4.11 MMC/SD Interrupt Register (MSC_IREG)

The MSC_IREG register shows the currently requested interrupt. The FIFO request interrupts, TXFIFO_WR_REQ, and RXFIFO_RD_REQ are masked off with the DMA_EN bit in the MSC_CMDAT register. The software is responsible for monitoring these bit in program I/O mode.



Bits	Name	Description	RW
15:8	Reserved		R
7	SDIO	Indicates whether the interrupt from SDIO is detected. 0 – The interrupt from SDIO is not detected. 1 – The interrupt from SDIO is detected.	R
6	TXFIFO_WR_REQ	Transmit FIFO write request. Set if data FIFO becomes half empty (the number of words is < 8). 0 – No Request for data Write to MSC_TXFIFO. 1 – Request for data write to MSC_TXFIFO.	R
5	RXFIFO_RD_REQ	Receive FIFO read request. Set if data FIFO becomes half full (the number of words is >= 8) or the entries in data FIFO are the last read data. 0 – No Request for data read from MSC_RXFIFO. 1 – Request for data read from MSC_RXFIFO.	R
4:3	Reserved		R
2	END_CMD_RES	Indicates whether the command/response sequence has been finished. 0 – The command/response sequence has not been finished. 1 – The command/response sequence has bee finished. Write 1 to clear.	WR
1	PRG_DONE	Indicates whether card has finished programming. 0 – Card has not finished programming and is busy. 1 – Card has finished programming and is no longer busy. Write 1 to clear.	WR
0	DATA_TRAN_DONE	Indicates whether data transfer is done. Note that for stream read/write, only when CMD12 (STOP_TRANS) has been sent, is this bit set.	WR

		0 – Data transfer is not complete. 1 – Data transfer has completed or an error has occurred. Write 1 to clear.	

1.4.12 MMC/SD Command Index Register (MSC_CMD)

MSC_CMD **0x1002102C**

Bit	7	6	5	4	3	2	1	0
	Reserved		CMD_INDEX					
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:6	Reserved		R
5:0	CMD_INDEX	Specifies the command index to be executed.	WR

1.4.13 MMC/SD Command Argument Register (MSC_ARG)

MSC_ARG **0x10021030**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ARG																															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:0	ARG	Specifies the argument for the current command.	WR

1.4.14 MMC/SD Response FIFO Register (MSC_RES)

The read-only MMC/SD Response FIFO register (RES_FIFO) holds the response sent back to the MMC/SD controller after every command. The size of this FIFO is 8 x 16-bit. The RES FIFO does not contain the 7-bit CRC for the response. The Status for CRC checking and response time-out status is in the status register, MSC_STAT.

The first halt-word read from the response FIFO is the most significant halt-word of the received response.

MSC_RES **0x10021034**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
15:0	DATA	Contains the response to every command that is sent by the MMC/SD controller. The size of this FIFO register is 8 x 16-bit.	R

1.4.15 MMC/SD Receive Data FIFO Register (MSC_RXFIFO)

The MSC_RXFIFO is used to read the data from a card. It is read-only to the software, and is read on 32-bit boundary. The size of this FIFO is 16 x 32-bit.

MSC_RXFIFO **0x10021038**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA																															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
31:0	DATA	One word of read data. The size of this FIFO is 16 x 32-bit.	R

1.4.16 MMC/SD Transmit Data FIFO Register (MSC_TXFIFO)

The MSC_TXFIFO is used to write the data to a card. It is write-only to the software, and is written on 32-bit boundary. The size of this FIFO is 16 x 32-bit.

MSC_TXFIFO **0x1002103C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA																															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
31:0	DATA	One word of write data. The size of this FIFO is 16 x 32-bit.	W

1.5 MMC/SD Functional Description

All communication between system and cards is controlled by the MSC. The MSC sends commands of two type: broadcast and addressed (point-to-point) commands.

Broadcast commands are intended for all cards, command like “Go_Idle_State”, “Send_Op_Cond”, “All_send_CID” and “Set_relative_Addr” are using way of broadcasting. During Broadcast mode, all cards are in open-drain mode, to avoid bus contention.

After Broadcast commands “Set_relative_Addr” issue, cards are enter standby mode, and Addressed command will be used from now on, in this mode, CMD/DAT will return to push-pull mode, to have maximum driving for maximum operation frequency.

The MMC and the SD are similar product. Besides the 4x bandwidth and the built-in encryption, they are being programmed similarly.

The MMC/SD controller (MSC) is the interface between the software and the MMC/SD bus. It is responsible for the timing and protocol between the software and the MMC/SD bus. It consists of control and status registers, a 16-bit response FIFO that is 8 entries deep, and one 32-bit receive/transmit data FIFOs that are 16 entries deep. The registers and FIFOs are accessible by the software.

MSC also enable minimal data latency by buffering data and generating and checking CRCs.

1.5.1 MSC Reset

The MMC/SD controller (MSC) can be reset by a hardware reset or software reset. All registers and FIFO controls are set to their default values after any reset.

1.5.2 MSC Card Reset

The command Go_Idle_State, CMD0 is the software reset command for MMC and SD Memory Card, and sets each card into Idle State regardless of the current card state; while in SDIO card, CMD52 is used to write IO reset in CCCR. The cards are initialized with a default relative card address (RCA=0x0000) and with a default driver stage register setting (lowest speed, highest driving current capability).

1.5.3 Voltage Validation

All cards shall be able to establish communication with the host using any operation voltage in the maximal allowed voltage range specified in this standard. However, the support minimum and maximum values for Vdd are defined in Operation Conditions register (OCR) and many not cover the whole range. Cards that store the CID and CSD data in the payload memory would be able to communicate these information only under data transfer Vdd conditions. That means if host and card have non compatible Vdd ranges, the card will not be able to complete the identification cycle, nor to send CSD data.

Therefore, a special command Send_Op_cont (CMD1 for MMC), SD_Send_Op_Cont (CMD41 for SD Memory) and IO_Send_Op_Cont (CMD5 for SDIO) are designed to provide a mechanism to identify and reject cards which do not match the Vdd range desired by the host. This is accomplished by the host sending the required Vdd voltage window as the operand of this command. Cards which can not perform data transfer in the specified range must discard themselves from further bus operations and go into Inactive State. By omitting the voltage range in the command, the host can query each card and determine the common voltage range before sending out-of-range cards into the Inactive State. This query should be used if the host is able to select a common voltage range or if a notification to the application of non usable cards in the stack is desired.

1.5.4 Card Registry

Card registry on MCC and SD card are different.

For SD card, Identification process start at clock rate Fod, while CMD line output drives are push-pull drivers instead of open-drain. After the bus is activated the host will request the cards to send their valid operation conditions. The response to ACMD41 is the operation condition register of the card. The same command shall be send to all of the new cards in the system. Incompatible cards are sent into Inactive State. The host then issue the command All_Send_CID (CMD2) to each card and get its unique card indentification (CID) number. Card that is unidentified, that is, which is in Ready State, send its CID number as the response. After the CID was sent by the card it goes into Identification State. Thereafter, the host issues Send_Relative_Addr (CMD3) asks the card to publish a new relative card address (RCA), which is shorter that CID and which will be used to address the card in the future data transfer mode. Once the RCA is received the card state changes to the Stand-by State. At this point, if the host wants that the card will have another RCA number, it may ask the card to publish a new number by sending another Send_Relative_Addr command to the card. The last published RCA is the actual RCA of the card. The host repeats the identification process, that is, the cycles with CMD2 and CMD3 for each card in the system.

In MMC, the host starts the card identification process in open-drain mode with the identification clock rate Fod. The open drain driver stages on the CMD line allow parallel card operation during card identification. After the bus is actived the host will request the cards to send their valid operation conditions (CMD1). The response to CMD1 is the ‘wired or’ operation on the condition restrictions of all cards in the system. Incompatible cards are sent into Inactive State. The host then issues the broadcast command All_Send_CID (CMD2), asking all cards for their unique card identification (CID) number. All unidentified cards, that is, those which are in Ready State, simultaneously start sending their CID numbers serially, while bit-wise monitoring their outgoing bitstream. Those cards, whose outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods stop sending their CID immediately and must wait for the next identification cycle. Since CID is unique for each card, only one card can be successfully send its full CID to the host. This card then goes into Identification State. Thereafter, the host issues Set_Relative_Addr (CMD3) to assign to this card a relative card address (RCA). Once the RCA is received the card state changes to the Stand-by State, and the card does not react to further identification cycles, and its output switches from open-drain to push-pull. The host repeat the process, that is CM2 and CMD3, until the host receive time-out condition to recognize completion of the identification process.

1.5.5 Card Access

1.5.5.1 Block Access, Block Write and Block Read

During block write (CMD24-27) one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. A card supporting block write shall always be able to accept a block of data defined by WRITE_BL_LEN. If the CRC fails, the card shall indicate the failure on the DAT line; the transferred data will be discarded and not written, and all further transmitted blocks (in multiple block write mode) will be ignored.

Programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card will report an error and not change any register contents. Some cards may require long and unpredictable times to write a block of data. After receiving a block of data and completing the CRC check, the card will begin writing and hold the DAT line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) at any time, and the card will respond with its status. The status bit READY_FOR_DATA indicates whether the card can accept new data or whether the write process is still in progress). The host may deselect the card by issuing CMD7 (to select a different card) which will displace the card into the Disconnect State and release the DAT line without interrupting the write operation. When reselecting the card, it will reactivate busy indication by pulling DAT to low if programming is still in progress and the writte buffer is unavailable.

Block read is similar to stream read, except the basic unit of data transfer is a block whose maximizes is defined in the CSD (READ_BL_LEN). If READ_BL_PARTIAL is set, smaller blocks whose starting and ending address are entirely contained within one physical block (as defined by READ_BL_LEN) may also be transmitted. Unlike stream read, a CRC is appended to the end of each block ensuring data transfer integrity. CMD17 (READ_SINGLE_BLOCK) initiates a block read and after completing the transfer, the card returns to the Transfer state. CMD18 (READ_MULTIPLE_BLOCK) starts a transfer of several consecutive blocks. Blocks will be continuously transferred until a stop command is issued. If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed, the card shall detect a block misalignment at the beginning of the first mis-aligned block, set the ADDRESS_ERROR error bit in the status register, abort transmission and wait in the Data State for a stop command.l

1.5.5.2 Stream Access, Stream Write and Stream Read (MMC Only)

Stream write (CMD20) starts the data transfer from the host to the card beginning from the starting address until the host issues a stop command. Since the amount of data to be transferred is not determined in advance, CRC can not be used. If the end of the memory range is reached while sending data and no stop command has been sent by the host, all further transferred data is discarded.

There is a stream oriented data transfer controlled by READ_DAT_UNTIL_STOP (CMD11). This command instructs the card to send its payload, starting at a specified address, until the host sends a STOP_TRANSMISSION command (CMD12). The stop command has execution delay due to the serial command transmission. The data transfer stops after the end bit of the stop command. If the end of the memory range is reached while sending data and no stop command has been sent yet by the host, the contents of the further transferred payload is undefined.

1.5.5.3 Erase, Group Erase and Sector Erase (MMC Only)

It is desirable to erase many sectors simultaneously in order to enhance the data throughput. Identification of these sectors is accomplished with the TAG_* commands. Either an arbitrary set of sectors within a single erase group, or an arbitrary selection of erase groups may be erase at one time, but not both together. That is, the unit of measure for determining an erase is either a sector or an erase group. If a set of sectors must be erased, all selected sectors must lie within the same erase group. To facilitate selection, a first command with the starting address is followed by a second command with the final address, and all sectors (or groups) within this range will be selected for erase.

1.5.5.4 Wide Bus Selection/Deselection

Wide Bus (4 bit bus width) operation mode may be selected / deselected using ACMD6. The default bus width after power up or GO_IDLE (CMD0) is 1 bit bus width. ACMD6 command is valid in 'trans state' only. That means the bus width may be changed only after a card was selected (CMD7).

1.5.6 Protection Management

Three write protect methods are supported in the host for Cards, Card internal write protect (Card's responsibility), Mechanical write protect switch (Host responsibility only) and Password protection card lock operation.

1.5.6.1 Card Internal Write Protection

Card data may be protected against either erase or write. The entire card may be permanently write protected by the manufacturer or content provider by setting the permanent or temporary write protect bits in the CSD. For cards which support write protection of groups of sectors by setting the WP_GRP_SIZE sectors as specified in the CSD), and the write protection may be changed by the application. The SET_WRITE_PROT command sets the write protection of the addressed write-protect group, and the CLR_WRITE_PROT command clears the write protection of the addressed write-protect group.

The SEND_WRITE_PROT command is similar to a single block read command. The card shall send a data block containing 32 write protection bits (representing 32 write protect groups starting at the specified address) followed by 16 CRC bits. The address field in the write protect commands is a group address in byte units. The card will ignore all LSB's below the group size.

1.5.6.2 Mechanical write protect switch

A mechanical sliding tablet on the side of the card will be used by the user to indicate that a given card is write protected or not. If the sliding tablet is positioned in such a way that the window is open that means the card is write protected. If the window is close the card is not write protected.

A proper, matched, switch on the socket side will indicated to the host that the card is write protected or not. It is the responsibility of the host to protect the card. The position of the write protect switch is un-known to the internal circuitry of the card.

1.5.6.3 Password Protect

The password protection feature enables the host to lock a card while providing a password, which later will be used for unlocking the card. The password and its size is kept in an 128-bit PWD and 8-bit PWD_LEN registers, respectively. These registers are non-volatile so that a power cycle will not erase them.

Locked cards respond to (and execute) all commands in the basic command class (class 0) and "lock card" command class. Thus the host is allowed to reset, initialize, select, query for status, etc., but not to access data on the card. If the password was previously set (the value of PWD_LEN is not 0) will be locked automatically after power on. Similar to the existing CSD and CID register write commands the lock/unlock command is available in "trans_state" only. This means that it does not include an address argument and the card must be selected before using it. The card lock/unlock command has the structure and bus transaction type of a regular single block write command. The transferred data block includes all the required information of the command (password setting mode, PWD itself, card lock/unlock etc.). The following table describes the structure of the command data block.

Table 1-5 Command Data Block Structure

Byte #	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Rsv	Rsv	Rsv	Rsv	ERASE	LOCK_UNLOCK	CLR_PWD	SET_PWD
1	PWDS_LEN							
2	Password Data							
…								
PWDS_LEN + 1								

- **ERASE** – 1 Defines Forced Erase Operation (all other bits shall be 0) and only the command byte is sent.
- **LOCK/UNLOCK** – 1=Locks the card. 0=Unlock the card (note that it is valid to set this bit together with SET_PWD but it is not allowed to set it together with CLR_PWD).
- **CLR_PWD** – 1=Clears PWD.
- **SET_PWD** – 1=Set new password to PWD.
- **PWD_LEN** – Defines the following password length (in bytes).
- **PWD** – The password (new or currently used depending on the command).

The data block size shall be defined by the host before it send the card lock/unlock command. This will allow different password sizes.

The following paragraphs define the various lock/unlock command sequences:

- Setting the Password:
  - Select a card (CMD7), if not previously selected already
  - Define the block length (CMD16), given by the 8bit card lock/unlock mode, the 8 bits password size (in bytes), and the number of bytes of the new password. In case that a password replacement is done, then the block size shall consider that both passwords, the old and the new one, are sent with the command.
  - Send Card Lock/Unlock command with the appropriate data block size on the data line including 16-bit CRC. The data block shall indicate the mode (SET_PWD), the length (PWD_LEN) and the password itself. In case that a password replacement is done, then the length value (PWD_LEN) shall include both passwords, the old and the new one, and the PWD field shall include the old password (currently used) followed by the new password.
  - In case that the sent old password is not correct (not equal in size and content) then LOCK_UNLOCK_FAILED error bit will be set in the status register and the old password does not change. In case that PWD matches the sent old password then the given new password and its size will be saved in the PWD and PWD_LEN fields, respectively.

  Note that the password length register (PWD_LEN) indicates if a password is currently set. When it equals 0 there is no password set. If the value of PWD_LEN is not equal to zero the card will lock itself after power up. It is possible to lock the card immediately in the current power session by setting the LOCK/UNLOCK bit (while setting the password) or sending additional command for card lock.

- Reset the password:
  - Select a card (CMD7), if not previously selected already
  - Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of the currently used password.
  - Send the card lock/unlock command with the appropriate data block size on the data line including 16-bit CRC. The data block shall indicate the mode CLR_PWD, the length (PWD_LEN) and the password (PWD) itself (LOCK/UNLOCK bit is don't care). If the PWD and PWD_LEN is set to 0. If the password is not correct then the LOCK_UNLOCK_FAILED error bit will be set in the status register.

- Locking a card:
  - Select a card (CMD7), if not previously selected already
  - Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of currently used password.
  - Send the card lock/unlock command with the appropriate data block size on the data line including 16-bit CRC. The data block shall indicate the mode LOCK, the length (PWD_LEN) and the password (PWD) itself.

If the PWD content equals to the sent password then the card will be locked and the card-locked status bit will be set in the status register. If the password is not correct then LOCK_UNLOCK_FAILED error bit will be set in the status register.

Note that it is possible to set the password and to lock the card in the same sequence. In such case the host shall perform all the required steps for setting the password (as described above) including the bit LOCK set while the new password command is sent. If the password was previously set (PWD_LEN is not 0), then the card will be locked automatically after power on reset. An attempt to lock a locked card or to lock a card that does not have a password will fail and the LOCK_UNLOCK_FAILED error bit will be set in the status register.

- Unlocking the card
  - Select a card (CMD7), if not previously selected already.
  - Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of the currently used password.
  - Send the card lock/unlock command with the appropriate data block size on the data line including 16-bit CRC. The data block shall indicate the mode UNLOCK, the length (PWD_LEN) and the password (PWD) itself.

If the PWD content equals to the sent password then the card will be unlocked and the card-locked status bit will be cleared in the status register. If the password is not correct then the LOCK_UNLOCK_FAILED error bit will be set in the status register.

Note that the unlocking is done only for the current power session. As long as the PWD is not cleared the card will be locked automatically on the next power up. The only way to unlock the card is by clearing the password. An attempt to unlock an unlocked card will fail and LOCK_UNLOCK_FAILED error bit will be set in the status register.

- Forcing Erase:

  In case that the user forgot the password (the PWD content) it is possible to erase all the card data content along with the PWD content. This operation is called Forced Erase.
  - Select a card (CMD7), if not previously selected already.
  - Define the block length (CMD16) to 1 byte (8bit card lock/unlock command). Send the card lock/unlock command with the appropriate data block of one byte on the data line including 16-bit CRC. The data block shall indicate the mode ERASE (the ERASE bit shall be the only bit set).

If the ERASE bit is not the only bit in the data field then the LCOK_UNLOCK_FAILED error bit will be set in the status register and the erase request is rejected. If the command was accepted then ALL THE CARD CONTENT WILL BE ERASED including the PWD and PWD_LEN register content and the locked card will get unlocked.

An attempt to force erase on an unlocked card will fail and LOCK_UNLOCK_FAILED error bit will be set in the status register.

1.5.7 Card Status

The response format R1 contains a 32-bit field named card status. This field is intended to transmit the card's status information (which may be stored in a local status register) to the host. If not specified otherwise, the status entries are always related to the previous issued command.

Table below defines the different entries of the status. The type and clear condition fields in the table are abbreviate as follows:

Type:

E:	Error bit.
S:	Status bit.
R:	Detected and set for the actual command response.
X:	Detected and set during command execution. The host must poll the card by issuing the status command in order to read these bits.

Clear Condition:

A:	According to the card current state.
B:	Always related to the previous command. Reception of a valid command will clear it (with a delay of one command).
C:	Clear by read.

Table 1-6 Card Status Description

Bits	Identifier	Type	Description	Clear Condition
31	OUT_OF_RAGE	E R	The commmand's argument was out of the allowed range for this card. 0 – No Error 1 – Error	C
30	ADDRESS_ERROR	E R X	A misaligned address which did not match the block length was used in the command. 0 – No Error 1 – Error	C
29	BLOCK_LEN_ERROR	E R	The transferred block length is not allowed for this, or the number of transferred bytes does not match the block length. 0 – No Error 1 – Error	C

28	ERASE_SEQ_ERROR	E R	An error in the sequence of erase commands occurred. 0 – No Error 1 – Error	C
27	ERASE_PARAM	E X	An invalid selection of sectors or groups for erase occurred. 0 – No Error 1 – Error	C
26	WP_VIOLATION	E R X	Attempt to program a write protected block. 0 – No Protected 1 – Protected	C
25	CARD_IS_LOCKED	S X	When set, signals that the card is locked by the host. 0 – Card unlocked 1 – Card locked	A
24	LOCK_UNLOCK_FAILED	E R X	Set when a sequence or password error has been detected in lock/unlock card command or if there was an attempt to access a locked card. 0 – No Error. 1 – Error.	C
23	COM_CRC_ERROR	E R	The CRC check of the previous command failed. 0 – No Error. 1 – Error.	B
22	ILLEGAL_COMMAND	E R	Command not legal for the card state. 0 – No Error. 1 – Error.	B
21	CARD_ECC_FAILED	E X	Card internal ECC was applied but failed to correct the data. 0 – normal. 1 – failure.	C
20	CC_ERROR	E R X	Internal card controller error. 0 – No Error. 1 – Error.	C
19	ERROR	E R X	A general or an unknown error occurred during the operation. 0 – No Error. 1 – Error.	C

18	UNDERRUN	E X	The card could not sustain data transfer in stream read mode. 0 – No Error. 1 – Error.	C
17	OVERRUN	E X	The card could not sustain data programming in stream write mode. 0 – No Error. 1 – Error.	C
16	CID/CSD_OVERWRITE	E R X	Can be either one of the following errors: 0 – No Error. 1 – Error.	C
15	WP_ERASE_SKIP	S X	Only partial address space was erased due to existing write protected blocks. 1 – No Protected. 1 – Protected.	C
14	CARD_ECC_DISABLED	S X	The command has been executed without using the internal ECC. 0 – enabled. 1 – disabled.	A
13	ERASE_RESET	S R	An erase sequence was cleared before executing because an out of erase sequence command was received. 0 – normal. 1 – set.	C
12:9	CURRENT_STATE	S X	The state of the card when receiving the command. If the command execution causes a state change, it will be visible to the host in the response to the next command. The four bits are interpreted as binary coded number between 0 and 15. 0 – idle 1 – ready 2 – ident 3 – stby 4 – tran 5 – data 6 – rcv 7 – prg 8 – dis (9 – 15) – rsv	B

8	READY_FOR_DATA	S X	Corresponds to buffer empty signaling on the bus. 0 – No Ready. 1 – Ready.	A
7:6	Reserved	-	-	-
5	APP_CMD	S R	The card will expect ACMD, or indication that the command has been interpreted as ACMD 0 – Disable. 1 – Enable.	C
4:0	Reserved	-	-	-

1.5.8 SD Status

The SD status contains status bits that are related to the SD card proprietary features and may be used for future application specific usage. The size of the SD status is one data block of 512bit. The content of this register is transmitted to the Host over the DAT bus along with 16-bit CRC. The SD status is sent to the host over the DAT bus if ACMD13 is sent (CMD55 followed with CMD13). ACMD13 can be sent to a card only in 'tran_state' (card is selected). SD status structure is described in below.

The same abbreviation for *type* and *clear condition* were used as for the Card Status above.

Table 1-7 SD Status Structure

Bits	Identifier	Type	Description	Clear Condition
511:510	DAT_BUS_WIDTH	S R	Shows the currently defined data bus width that was defined by SET_BUS_WIDTH command. 00 – 1 (default). 01 – Reserved. 10 – 4 bit width. 11 – Reserved.	A
509	SECURED_MODE	S R	Card is in Secured Mode of operation. 0 – Not in the Mode. 10 – In the mode.	A
508:496	Reserved			
495:480	SD_CARD_TYPE	S R	All 0, is SD Memory cards.	A
479:448	SIZE_OF_PROTECTED_AREA	S R	Size of protected area.	A
447:312	Reserved			
311:0	Reserved for manufacturer			

1.5.9 SDIO

I/O access differs from memory in that the registers can be written and read individually and directly without a FAT file structure or the concept of blocks (although block access is supported). These registers allow access to the IO data, control of the IO function, and report on status or transfer I/O data to and from the host.

Each SDIO card may have from 1 to 7 functions plus one memory function built into it. A function is a self contained I/O device. I/O functions may be identical or completely different from each other. All I/O functions are organized as a collection of registers, and there is a maximum of 131,072 registers possible for each I/O function.

1.5.9.1 SDIO Interrupts

In order to allow the SDIO card to interrupt the host, and interrupt function is added to a pin on the SD interface. Pin number 8 which is used as DAT[1] when operating in the 4 bit SD mode is used to signal the card's interrupt to the host. The use of interrupt is optional for each card or function within a card. The SDIO interrupt is "level sensitive", that is, the interrupt line must be held active (low) until it is either recognized and acted upon by the host or de-asserted due to the end of the Interrupt Period. Once the host has serviced the interrupt, it is cleared via an IO write to the appropriate bit in the CCCR. The interrupt output of all SDIO cards is active low. This host controller provides pull-up resistors on all data lines DAT[3:0].

As Pin 8 of the card is shared between the IRQ and DAT[1] use in the 4 bit SD mode, and interrupt shall only be sent by the card and recognized by the host during a specific time. The time that a low on Pin 8 will be recognized as an interrupt is defined as the Interrupt Period.

The host here will only sample the level of Pin 8 (DAT[1]/IRQ) into the interrupt detector during the Interrupt Period. At all other times, the host will ignore the level on Pin 8. Note that the Interrupt Period is applicable for both memory and IO operations. The definition of the Interrupt Period is different for operations with single block and multiple block data transfer.

1.5.9.2 SDIO Suspend/Resume

Within a multi-function SDIO or a Combo (Mix IO and Memory) card, there are multiple devices (I/O and memory) that must share access to the SD bus. In order to allow the sharing of access to the host among multiple devices, SDIO and combo cards can implement the optional concept of suspend/resume. In a card supports suspend/resume, the host may temporarily halt a data transfer operation to one function or memory (suspend) in order to free the bus for a higher priority transfer to a different function of memory. Once this higher-priority transfer is complete, the original transfer is re-started where it left off (resume). The host controller here is supported by all IO functions except zero, and the memory of a combo card, and can suspend multiple transactions and resume them in any order desired. IO function zero does not support suspend/resume.

The procedure used to perform the Suspend/Resume operation on the SD bus is:

1. The host determines which function currently used the DAT[] line(s).
2. The host requests the lower priority or slower transaction to suspend.

3. The host checks for the transaction suspension to complete.
4. The host begins the higher priority transaction.
5. The host waits for the completion of the higher priority transaction.
6. The host restores the suspended transaction.

1.5.9.3 SDIO Read Wait

The optional Read Wait (RW) operation is defined only for the SD 1-bit and 4-bit modes. The read wait operation allows a host to signal a card that it is doing a read multiple (CMD53) operation to temporarily stall the data transfer while allowing the host to send commands to any function within the SDIO device. To determine if a card supports the Read Wait protocol, the host must test capability bits in CCCR. The timing for Read Wait is base on the Interrupt Period.

1.5.10 Clock Control

The software should guarantee that the card identification process starts in open-drain mode with the clock rate fod (0 ~ 400khz). In addition, the software should also make the card into interrupt mode with fod (only for MMC). The commands that require fod are CMD0, CMD1, CMD2, CMD3, CMD5, CMD40 and ACMD41. In data transfer mode, the MSC controller can operate card with clock rate fpp (0 ~ 25Mhz).

1.5.11 Application Specified Command Handling

The MultiMediaCard/SD system is designed to provide a standard interface for a variety applications types. In this environment it is anticipate that there will be a need for specific customers/applications features. To enable a common way of implementing these features, two types of generic commands are defined in the standard: Application Specific Command, ACMD, and General Command, GEN_CMD.

GEN_CMD, this command, when received by the card, will cause the card to interpret the following command as an application specific command, ACMD. The ACMD has the same structure as of regular MultiMediaCard standard commands and it may have the same CMD number. The card will recognize it as ACMD by the fact that it appears after APP_CMD.

The only effect of the APP_CMD is that if the command index of the, immediately, following command has an ACMD overloading, the none standard version will used. If, as an example, a card has a definition for ACMD13 but not for ACMD7 then, if received immediately after APP_CMD command, Command 13 will be interpreted as the non standard ACMD13 but, command 7 as the standard CMD7.

In order to use one of the manufacturer specific ACMD's the host will:

- Send APP_CMD. The response will have the APP_CMD bit (new status bit) set signaling to the host that ACMD is now expected.
- Send the required ACMD. The response will have the APP_CMD bit set, indicating that the accepted command was interpreted as ACMD. If a non-ACMD is sent then it will be respected by the card as normal MultiMediaCard command and the APP_CMD bit in the Card Status

stays clear.

If a non valid command is sent (neither ACMD nor CMD) then it will be handled as a standard MultiMediaCard illegal command error.

The bus transaction of the GEN_CMD is the same as the single block read or write commands (CMD24 or CMD17). The difference is that the argument denotes the direction of the data transfer (rather than the address) and the data block is not a memory payload data but has a vendor specific format and meaning.

The card shall be selected (‘tran_state’) before sending CMD56. The data block size is the BLOCK_LEN that was defined with CMD16. The response to CMD56 will be R1b (card status + busy indication).

1.6 MMC/SD Controller Operation

1.6.1 Data FIFOs

The controller FIFOs for the response tokens, received data, and transmitted data are MSC_RES, MSC_RXFIFO, and MSC_TXFIFO, respectively. These FIFOs are accessible by the software and are described in the following paragraphs.

1.6.1.1 Response FIFO (MSC_RES)

The response FIFO, MSC_RES, contains the response received from an MMC/SD card after a command is sent from the controller. MSC_RES is a read-only, 16-bit, and 8-entry deep FIFO.

The FIFO will hold all possible response lengths. Responses that are only one byte long are located on the LSB of the 16-bit entry in the FIFO. The first half-word read from the response FIFO is the most significant half-word of the received response. For example, if the response format is R1, then the response read from RES_FIFO is bit [47:32], bit[31:16], bit[15:0] and in the third half-word only the low 8-bit is effective response [15:8] and the high 8-bit is ignored. If the response format is R2, then the response read from MSC_RES is bit [135:8] and needs reading 8 times.

The FIFO does not contain the response CRC. The status of the CRC check is in the status register, MSC_STAT.

1.6.1.2 Receive/Transmit Data FIFO (MSC_RXFIFO/MSC_TXFIFO)

The receive data FIFO and transmit data FIFO share one 16-entry x 32-bit FIFO, because at one time data are only received or are only transmitted. If it is used to receive data, it is called MSC_RXFIFO and read-only. If it is used to transmit data, it is called MSC_TXFIFO and write-only.

Data FIFO and its controls are cleared to a starting state after a system reset or at the beginning of the operations which include data transfer (MSC_CMDAT[DATA_EN] == 1).

If at any time MSC_RXFIFO becomes full and the data transmission is not complete, the controller turns the MSC_CLK off to prevent any overflows. When the clock is off, data transmission from the card stops until the clock is turned back on. After MSC_RXFIFO is not full, the controller turns the clock on to continue data transmission. The full status of the FIFO is registered in the MSC_STAT [DATA_FIFO_FULL] bit.

If at any time MSC_TXFIFO becomes empty and the data transmission is not complete, the controller turns the MSC_CLK off to prevent any underrun. When the clock is off, data transmission to the card stops until the clock is turned back on. When MSC_TXFIFO is no longer empty, the controller automatically restarts the clock. The empty status of the FIFO is registered in the MSC_STAT [DATA_FIFO_EMPTY] bit.

The FIFO is readable on word (32-bit) boundaries. The max read/written number is 16 words. The controller can correctly process big-endian an little-endian data.

Because at the beginning of the operation which include data transfer (MSC_CMDAT [DATA_EN] == 1), Data FIFO and its controls are cleared, software should guarantee data in FIFO have been read/written before beginning a new command.

1.6.2 DMA and Program I/O

Software may communicate to the MMC controller via the DMA or program I/O.

To access MSC_RXFIFO/MSC_TXFIFO with the DMA, the software must program the DMA to read or write the FIFO with source port width 32-bit, destination port width 32-bit, transfer data size 32-byte, transfer mode single. For example, to write 64 bytes of data to the MSC_TXFIFO, the software must program the DMA as follows:

```
DMA_DCTRn = 2                // Write 2 32-bytes (64 bytes)
DMA_DCCRn[SWDH] = 0          // source port width is 32-bit
DMA_DCCRn[DWDH] = 0          // destination port width is 32-bit
DMA_DCCRn[DS] = 4            // transfer data size is 32-byte
DMA_DCCRn[TM] = 4            // transfer mode is single
DMA_DCCRn[RDIL] = 0          // request detection interval length is 0
```

The number of 32-bytes should be calculated from the number of transferred bytes as follows:

The number of words = (The number of bytes + 31) / 32

If the number of transferred bytes is not the multiple of 4, the controller can correctly process endian.

The DMA trigger level is 8 words, that is to say, the DMA read trigger is when data words in MSC_RXFIFO is >= 8 and the DMA write trigger is when data words in MSC_TXFIFO is < 8. Software can also configure DMA registers based on requirements, but the above 32-byte transfer data size is most efficient.

With program I/O, the software waits for the MSC_IREG [RXFIFO_RD_REQ] or MSC_IREG [TXFIFO_WR_REQ] interrupts before reading or writing the respective FIFO.

Note:

1. The MSC_CMDAT [DMA_EN] bit must be set to a 1 to enable communication with the DMA and it must be set to a 0 to enable program I/O.
2. DMA can be enabled only after MSC_CMDAT is written, because MSC_CMDAT [DATA_EN] is used to reset TX/RXFIFO.

1.6.3 Start and Stop clock

The software stops the clock as follows:

1. Write MSC_STRPCL with 0x01 to stop the MMC/SD bus clock.
2. Wait until MSC_STAT[CLK_EN] becomes zero.

To start the clock the software writes MSC_STRPCL with 0x02.

1.6.4 Software Reset

Reset includes the MSC reset and the card reset.

The MSC reset is through MSC_STRPCL [RESET] bit.

The card reset is to make the card into idle state. CMD0 (GO_IDLE_STATE) sets the MMC and SD memory cards into idle state. CMD52 (IO_RW_DIRECT, with argument 0x88000C08) reset the SD I/O card. The MMC/SD card are initialized with a default relative card address (RCA = 0x0001 for MMC and RCA = 0x0000 for SD) and with a default driver stage register setting (lowest speed, highest driving current capability).

The following registers must be set before the clock is started:

1. Stop the clock.
2. Set MSC_STRPCL register to 0x08 to reset MSC.
3. Wait while MSC_STAT [IS_RESETTING] is 1.
4. Set MSC_CMD with CMD0.
5. Update the MSC_CMDAT register as follows:
   a) Write 0x0000 to MSC_CMDAT [RESPONSE_FORMAT]
   b) Clear the MSC_CMDAT [DATA_EN] bit.
   c) Clear the MSC_CMDAT [BUSY] bit.
   d) Clear the MSC_CMDAT [INIT] bit.
6. Start the clock.
7. Start the operation (write MSC_STRPCL with 0x04)
8. Wait for the END_CMD_RES interrupt.

9. Set MSC_CMD with CMD52.
10. Set MSC_ARG with 0x88000C08
11. Update the MSC_CMDAT register as follows:
    a) Write 0x005 to MSC_CMDAT [RESPONSE_FORMAT]
    b) Clear the MSC_CMDAT [DATA_EN] bit.
    c) Clear the MSC_CMDAT [BUSY] bit.
    d) Clear the MSC_CMDAT [INIT] bit.
12. Start the operation.
13. Wait for the END_CMD_RES interrupt.

1.6.5 Voltage Validation and Card Registry

At most 10 MMC and 1 SD (either SDMEM or SDIO) can be inserted MMC/SD bus at the same time, and their voltage validation and card registry steps are different, so the software should be programmed as follows:

1. Check whether SDIO card is inserted.
2. Check whether SDMEM card is inserted.
3. Check whether MMC cards are inserted.

1.6.5.1 Check SDIO

The commands are sent as follows:

1. (Optional) Send CMD52 (IO_RW_DIRECT) with argument 0x88000C08 to reset SDIO card.
2. Send CMD5 (IO_SEND_OP_CMD) to validate voltage.
3. If the response is correct and the number of IO functions > 0, then continue, else go to check SDMEM.
4. If C-bit in the response is ready (the initialization has finished), go to 6.
5. Send CMD5 (IO_SEND_OP_CMD) to validate voltage, then go to 4.
6. If memory-present-bit in the response is true, then it is a combo card (SDIO + Memory), else it is only a SDIO card.
7. If it is a combo card, go to check SDMEM to initialize the memory part.
8. Send CMD3 (SET_RELATIVE_ADDR) to let the card publish a RCA. The RCA is returned from the response.
9. If do not accept the new RCA, go to 8, else record the new RCA.
10. Go to check MMC, because we can assure that there is no SDMEM card.

1.6.5.2 Check SDMEM

If there is no SDIO card or there is a combo card, continue to check SDMEM.

The commands are sent as follows:

1. (Optional) Send CMD0 (GO_IDLE_STATE) to reset MMC and SDMEM card. This command has no response.
2. Send CMD55. Here the default RCA 0x0000 is used for CMD55.
3. If the response is correct (CMD55 has response), then continue, else go to check MMC.
4. Send ACMD41 (SD_SEND_OP_CMD) to validate voltage (the general OCR value is 0x00FF8000).
5. If the initialization has finished, go to 7. (The response is the OCR register and it includes a status information bit (bit [31]). This status bit is set if the card power up procedure has been finished. As long as the card is busy, the corresponding bit[31] is set to LOW.)
6. Send CMD55 and ACMD41 to validate voltage, and then go to 5.
7. Send CMD2 (ALL_SEND_CID) to get the card CID.
8. Send CMD3 (SET_RELATIVE_ADDR) to let card publish a RCA. The RCA is returned from the response.
9. If do not accept the new RCA, go to 8, else record the new RCA.
10. Go to check MMC.

1.6.5.3 Check MMC

Because there may be several MMC card, so some steps (5 ~ 8) should be repeated several times.

The commands are sent as follows:

1. Send CMD1 (SEND_OP_CMD) to validate voltage (the general OCR value is 0x00FF88000).
2. If the response is correct, then continue, else goto 9.
3. If the initialization has finished, go to 5. (The response is the OCR register and it includes a status information bit (bit [31]). This status bit is set if the card power up procedure has

been finished. As long as the card is busy, the corresponding bit[31] is set to LOW.)

4. Send CMD1 (SEND_OP_CMD) to validate voltage, and then go to 3.
5. Send CMD2 (ALL_SEND_CID) to get the card CID.
6. If the response timeout occurs, goto 9.
7. Send CMD3 (SET_RELATIVE_ADDR) to assign the card a RCA.
8. If there are other MMC cards, then go to 5.
9. Finish.

1.6.6 Single Data Block Write

In a single block write command, the following registers must be set before the operation is started:

1. Set MSC_NOB register to 0x0001.
2. Set MSC_BLKLEN to the number of bytes per block.
3. Update the MSC_CMDAT register as follows:
   a) Write 0x001 to MSC_CMDAT [RESPONSE_FORMAT]
   b) Write 0x2 to MSC_CMDAT [BUS_WIDTH] if the card is SD, else clear it.
   c) Set the MSC_CMDAT [DATA_EN] bit.
   d) Set the MSC_CMDAT [WRITE_READ] bit.
   e) Clear the MSC_CMDAT [STREAM_BLOCK] bit.
   f) Clear the MSC_CMDAT [BUSY] bit.
   g) Clear the MSC_CMDAT [INIT] bit.
4. Start the operation.
5. Write MSC_IMASK with some value to unmask the expected interrupts.

Then the software must perform the following steps:

1. Wait for the MSC_IREG [END_CMD_RES] interrupt.
2. Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
   At the same time write data to the MSC_TXFIFO and continue until all of the data have been written to the FIFO.
3. Wait for MSC_IREG [PROG_DONE] interrupt. This interrupt indicates that the card has finished programming. Certainly software may start another command sequence on a different card.
4. Read the MSC_STAT register to verify the status of the transaction (i.e. CRC error status).

To address a different card, the software sends a select command to that card by sending a basic no data command and response transaction. To address the same card, the software must wait for MSC_IREG [PROG_DONE] interrupt. This ensures that the card is not in the busy state.

In addition, CMD26 (PROGRAM_CID), CMD27 (PROGRAM_CSD), CMD42 (LOCK/UNLOCK), CMD56 (GEN_CMD: write) and CMD53 (single_block_write) operations are similar to single block write.

1.6.7 Single Block Read

In a single block read command, the following registers must be set before the operation is started:

1. Set MSC_NOB register to 0x0001.
2. Set MSC_BLKLEN register to the number of bytes per block.
3. Update the following bits in the MSC_CMDAT register:
   a) Write 0x001 to MSC_CMDAT [RESPONSE_FORMAT].
   b) Write 0x2 to MSC_CMDAT [BUS_WIDTH] if the card is SD, else clear it.
   c) Set the MSC_CMDAT [DATA_EN] bit.
   d) Clear the MSC_CMDAT [WRITE_READ] bit.
   e) Clear the MSC_CMDAT [STREAM_BLOCK] bit.
   f) Clear the MSC_CMDAT [BUSY] bit.
   g) Clear the MSC_CMDAT [INIT] bit.
4. Start the operation.
5. Write MSC_IMASK with some value to unmask the expected interrupts.

Then the software must perform the following steps:

1. Wait for the MSC_IREG [END_CMD_RES] interrupt.
2. Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
   At the same time read data from the MSC_RXFIFO as data becomes available in the FIFO, and continue reading until all data is read from the FIFO.
3. Read the MSC_STAT register to verify the status of the transaction (i.e. CRC error status).

In addition, CMD30 (SEND_WRITE_PROT), ACMD13 (SD_STATUS), CMD56 (GEN_CMD-read), ACMD51 (SEND_SCR) and CMD53 (single_block_read) are similar to single block read.

1.6.8 Multiple Block Write

The multiple block write mode is similar to the single block write mode, except that multiple blocks of data are transferred. Each block is the same length. All the registers are set as they are for the single block write, except that the MSC_NOB register is set to the number of blocks to be written.

The multiple block write mode also requires a stop transmission command, CMD12, after the data is transferred to the card. After the MSC_IREG [DATA_TRAN_DONE] interrupt occurs, the software must program the controller register to send a stop data transmission command.

If multiple block write with pre-defined block count (refer to MMC spec v-3.3) is used, CMD12 should not be sent.

For SDIO card, CMD53 (multiple_block_write) is also similar, but when IO abort (CMD52) is sent, MSC_CMDAT [IO_ABORT] should be 1.

Table 1-8 How to stop multiple block write

Operation	Stop condition	Software processing
Open-ended or SDIO infinite	After write MSC_NOB blocks	1. Wait for DATA_TARN_DONE interrupt 2. Send CMD12 or CMD52 (IO abort) 3. Wait for END_CMD_RES and PRG_DONE interrupt
Open-ended or SDIO	Stop writing in advance (not	1. Set MSC_STRPCL [EXIT_MULTIPLE]

infinite	write MSC_NOB blocks)	2. Wait for DATA_TRAN_DONE interrupt 3. Send CMD12 or CMD52 (IO abort) 4. Wait for END_CMD_RES and PRG_DONE interrupt.
Predefined block or SDIO finite	After writing MSC_NOB blocks	1. Wait for DATA_TRAN_DONE interrupt
Predefined block or SDIO finite	Stop writing in advance (not write MSC_NOB blocks)	1. Set MSC_STRPCL [EXIT_MULTIPLE] 2. Wait for DATA_TRAN_DONE interrupt 3. Send CMD12 or CMD52 (IO abort) 4. Wait for END_CMD_RES and PRG_DONE interrupt

1.6.9 Multiple Block Read

The multiple blocks read mode is similar to the single block read mode, except that multiple blocks of data are transferred. Each block is the same length. All the registers are set as they are for the single block read, except that the MSC_NOB register is set to the number of blocks to be read.

The multiple blocks read mode requires a stop transmission command, CMD12, after the data from the card is received. After the MSC_IREG [DATA_TRAN_DONE] interrupt has occurred, the software must program the controller registers to send a stop data transmission command.

If multiple block read with pre-defined block count (refer to MMC spec v-3.3) is used, CMD12 should not be sent.

For SDIO card, CMD53 (multiple_block_read) is also similar, but when IO abort (CMD52) is sent, MSC_CMDAT [IO_ABORT] should be 1.

Table 1-9 How to stop multiple block read

Operation	Stop condition	Software processing
Open-ended or SDIO infinite	After reading MSC_NOB blocks	1. Wait for DATA_TRAN_DONE interrupt 2. Send CMD12 or CMD52 (IO abort) 3. Wait for END_CMD_RES interrupt
Open-ended or SDIO infinite	Stop reading in advance (not write MSC_NOB blocks)	1. Set MSC_STRPCL [EXIT_MULTIPLE] 2. Wait for DATA_TRAN_DONE interrupt 3. Send CMD12 or CMD52 (IO abort) 4. Wait for END_CMD_RES interrupt
Predefined block or SDIO finite	After reading MSC_NOB blocks	1. Wait for DATA_TRAN_DONE interrupt
Predefined block or SDIO finite	Stop reading in advance (not write MSC_NOB blocks)	1. Set MSC_STRPCL [EXIT_MULTIPLE] 2. Wait for DATA_TRAN_DONE interrupt 3. Send CMD12 or CMD52 (IO abort) 4. Wait for END_CMD_RES interrupt

1.6.10 Stream Write (MMC)

In a stream write command, the following registers must be set before the operation is started:

1. Update MSC_CMDAT register as follows:
   a) Write 0x001 to the MSC_CMDAT [RESPONSE_FORMAT].
   b) Clear the MSC_CMDAT [BUS_WIDTH] because only MMC support stream write
   c) Set the MSC_CMDAT [DATA_EN] bit.
   d) Set the MSC_CMDAT [WRITE_READ] bit.
   e) Set the MSC_CMDAT [STREAM_BLOCK] bit.
   f) Clear the MSC_CMDAT [BUSY] bit.
   g) Clear the MSC_CMDAT [INIT] bit.
2. Start the operation.
3. Write MSC_IMASK with some value to unmask the expected interrupts.

Then the software must perform the following steps:

1. Wait for the MSC_IREG [END_CMD_RES] interrupt.
2. Write data to the MSC_TXFIFO and continue until all of the data is written to the Data FIFO.
3. Stop clock. Wait until MSC_STAT[CLK_EN] becomes 0. The clock must be stopped.
4. Set the command registers for a stop transaction command (CMD12) and other registers.
5. Start the clock and start the operation.
6. Wait for the MSC_IREG [END_CMD_ERS] interrupt.
7. Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
8. Wait for the MSC_IREG [PRG_DONE] interrupt. This interrupt indicates that the card has finished programming. Certainly software may start another command sequence on a different card.
9. Read the MSC_STAT register to verify the status of the transaction.

To address a different card, the software must send a select command to that card by sending a basic no data command and response transaction. To address the same card, the software must wait for MSC_IREG [PRG_DONE] interrupt. This ensures that the card is not in the busy state.

If partial blocks are allowed (if CSD parameter WRITE_BL_PARTIAL is set) the data stream can start and stop at any address within the card address space, otherwise it shall start and stop only at block boundaries. If WRITE_BL_PARTIAL is not set, 16 more stuff bytes need to be written after the useful written data, otherwise only write the useful written data.

1.6.11 Stream Read (MMC)

In a stream read command, the following registers must be set before the operation is turned on:

1. Update the MSC_CMDAT register as follows:
   a) Write 0x01 to the MSC_CMDAT [RESPONSE_FORMAT]
   b) Clear the MSC_CMDAT [BUS_WIDTH] because only MMC support stream read.
   c) Clear the MSC_CMDAT [WRITE_READ] bit.
   d) Set the MSC_CMDAT [STREAM_BLOCK] bit.
   e) Clear the MSC_CMDAT [BUSY] bit.
   f) Clear the MSC_CMDAT [INIT] bit.

2. Start the operation.
3. Write MSC_IMASK with some value to unmask the expected interrupts.

Then the software must perform the following steps:

1. Wait for the MSC_IREG [END_CMD_RES] interrupt.
2. Read data from the MSC_RXFIFO and continue until all of the expected data has been read from the FIFO.
3. Write MSC_STRPCL [EXIT_TRANSER] with 1. If MSC_STAT[DATA_FIFO_FULL] is 1, then read MSC_RXFIFO to make it not full. Because if data FIFO is full, MSC_CLK is stopped. Here, the data FIFO contains useless data.
4. Set the command registers for a stop transaction command (CMD12) and send it. There is no need to stop the clock.
5. Wait for the MSC_IREG [END_CMD_RES]
6. Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
7. Read the MSC_STAT register to verify the status of the transaction.

1.6.12 Erase, Select/Deselect and Stop

For CMD7 (SELECT/DESELECT_CARD), CMD12 (STOP_TRANSMISSION) and CMD38 (ERASE), the following registers must be set before the operation is started:

1. Update the MSC_CMDAT register as follows:
   a) Write 0x01 to the MSC_CMDAT [RESPONSE_FORMAT]
   b) Clear the MSC_CMDAT [DATA_EN] bit.
   c) Clear the MSC_CMDAT [WRITE_READ] bit.
   d) Clear the MSC_CMDAT [STREAM_BLOCK] bit.
   e) Set the MSC_CMDAT [BUSY] bit.
   f) Clear the MSC_CMDAT [INIT] bit.
2. Start the operation.
3. Write MSC_IMASK with some value to unmask the expected interrupts.

Then the software must perform the following steps:

1. Wait for the MSC_IREG [END_CMD_RES] interrupt.
2. Wait for the MSC_IREG [PRG_DONE] interrupt. If CMD12 is sent to terminate data read operation, then there is no need to wait for MSC_IREG [PRG_DONE] interrupt. This interrupt indicates that the card has finished programming. Certainly software may start another command sequence on a different card.

1.6.13 SDIO Suspend/Resume

The actual suspend/resume steps are as follows:

1. During data transfer, send CMD52 to require suspend. BR and RAW flag should be 1.
2. If BS flag in the response is 0, then suspend has been accepted and goto 4
3. Send CMD52 to query card status. R flag should be 1. Go to 2.
4. Write MSC_STRPCL [EXIT_TRANSFER] with 1.
5. Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.

6. Read MSC_NOB, MSC_SNOB and etc, save them into variables.
7. Set registers for high priority transfer and start it.
8. Wait until high priority transfer is finished.
9. Restore registers from variables, but MSC_NOB should be (MSC_NOB – MSC_SNOB).
10. Send CMD52 to require resume. FSx should be resumed function number.

1.6.14 SDIO ReadWait

The actual ReadWait steps are as follows

1. During multiple block read, read MSC_SNOB. If MSC_SNOB is nearby or equal to MSC_NOB, no need to use ReadWait.
2. Write MSC_STRPCL [START_READWAIT] with 1.
3. Wait until MSC_STAT [IS_READWAIT] becomes 1.
4. Send CMD52 to query card status.
5. Write MSC_STRPCL [STOP_READWAIT] with 1.

1.6.15 Operation and Interrupt

The software can use polling-status method to operate the MMC/SD card, but this is not the proposed method, because its performance is very low. The proposed method is to use interrupt. Generally there are fixed necessary steps to finish each command. The steps are as follows.

1. (Optional) Stop clock. Poll CLK_EN.
2. Fill the registers (MSC_CMD, MSC_CMDAT, MSC_ARG, MSC_CLKRT, and etc).
3. (Optional) Start clock.
4. Start the operation. Wait for the MSC_IREG [END_CMD_RES] interrupt.
5. Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
6. Send STOP_TRANS (CMD12) or I/O abort (CMD52). Wait for the MSC_IREG [END_CMD_ERS] interrupt.
7. Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt
8. Wait for the MSC_IREG [PRG_DONE] interrupt.

Table 1-10 The mapping between Commands and Steps

Index	Abbreviation	1	2	3	4	5	6	7	8	Comments
CMD0	GO_IDLE_STATE	Y	Y	Y	Y					
CMD1	SEND_OP_COND	Y	Y	Y	Y					
CMD2	ALL_SEND_CID	Y	Y	Y	Y					
CMD3	SET_RELATIVE_ADDR	Y	Y	Y	Y					
CMD4	SET_DSR	Y	Y	Y	Y					
CMD7	SELECT/DSELECT_CARD	Y	Y	Y	Y				Y	
CMD9	SEND_CID	Y	Y	Y	Y					
CMD10	SEND_CSD	Y	Y	Y	Y					
CMD11	READ_DAT_UNTIL_STOP	Y	Y	Y	Y		Y	Y		
CMD12	STOP_TRANSMISSION	Y	Y	Y	Y				Y	
CMD13	SEND_STATUS	Y	Y	Y	Y					
CMD15	GO_INACTIVE_STATE	Y	Y	Y	Y					
CMD16	SET_BLOCKLEN	Y	Y	Y	Y					

CMD17	READ_SINGLE_BLOCK	Y	Y	Y	Y	Y				
CMD18	READ_MULTIPLE_BLOCK	Y	Y	Y	Y	Y	Y			Open-ended
CMD18	READ_MULTIPLE_BLOCK	Y	Y	Y	Y	Y				Predefine blocks
CMD20	WRITE_DAT_UNTIL_STOP	Y	Y	Y	Y		Y	Y	Y	
CMD23	SET_BLOCK_COUNT	Y	Y	Y	Y					
CMD24	WRITE_SINGLE_BLOCK	Y	Y	Y	Y	Y			Y	
CMD25	WRITE_MULTIPLE_BLOCK	Y	Y	Y	Y	Y	Y		Y	Open-ended
CMD25	WRITE_MULTIPLE_BLOCK	Y	Y	Y	Y	Y			Y	Predefine blocks
CMD26	PROGRAM_CID	Y	Y	Y	Y	Y			Y	
CMD27	PROGRAM_CSD	Y	Y	Y	Y	Y			Y	
CMD28	SET_WRITE_PROT	Y	Y	Y	Y				Y	
CMD29	CLR_WRITE_PROT	Y	Y	Y	Y				Y	
CMD30	SEND_WRITE_PROT	Y	Y	Y	Y	Y				
CMD32	ERASE_WR_BLOCK_START	Y	Y	Y	Y					
CMD33	ERASE_WR_BLOCK_END	Y	Y	Y	Y					
CMD35	ERASE_GROUP_START	Y	Y	Y	Y					
CMD36	ERASE_GROUP_END	Y	Y	Y	Y					
CMD38	ERASE	Y	Y	Y	Y				Y	
CMD39	FAST_IO	Y	Y	Y	Y					
CMD40	GO_IRQ_STATE	Y	Y	Y	Y					
CMD42	LOCK/UNLOCK	Y	Y	Y	Y	Y			Y	
CMD55	APP_CMD	Y	Y	Y	Y					
CMD56	GEN_CMD	Y	Y	Y	Y	Y				Read
CMD56	GEN_CMD	Y	Y	Y	Y	Y			Y	Write
ACMD6	SET_BUS_WIDTH	Y	Y	Y	Y					
ACMD13	SD_STATUS	Y	Y	Y	Y	Y				
ACMD22	SEND_NUM_WR_BLOCKS	Y	Y	Y	Y					
ACMD23	SET_WR_BLOCK_COUNT	Y	Y	Y	Y					
ACMD41	SD_SEND_OP_COND	Y	Y	Y	Y					
ACMD42	SET_CLR_CARD_DETECT	Y	Y	Y	Y					
ACMD51	SEND_SCR	Y	Y	Y	Y	Y				

Note: For stream read/write, STOP_CMD is sent after finishing data transfer. For write, STOP_CMD is with the last six bytes. For read, STOP_CMD is sent after receiving data and card sends some data which MSC ignores.