SD Specification

Part A2

SD Host Controller Standard Simplified Specification

Version 1.00

April 3, 2006

Technical Committee
SD Association

Revision History

Date	Version	Explanation and changes compared to previous issue
April 3, 2006	1.00	Simplified version initial release.

Release of SD Simplified Specification

The following conditions apply to the release of the SD simplified specification ("Simplified Specification") by the SD Card Association. The Simplified Specification is a subset of the complete SD Specification which is owned by the SD Card Association.

Publisher:

SD Card Association
2400 Camino Ramon, Suite 375
San Ramon, CA 94583 USA
Telephone: +1 (925) 275-6615,
Fax: +1 (925) 886-4870
E-mail: office@sdcard.org

Copyright Holder:

The SD Card Association

Notes:

This Simplified Specification is provided on a non-confidential basis subject to the disclaimers below. Any implementation of the Simplified Specification may require a license from the SD Card Association or other third parties.

Disclaimers:

The information contained in the Simplified Specification is presented only as a standard specification for SD Cards and SD Host/Ancillary products and is provided "AS-IS" without any representations or warranties of any kind. No responsibility is assumed by the SD Card Association for any damages, any infringements of patents or other right of the SD Card Association or any third parties, which may result from its use. No license is granted by implication, estoppel or otherwise under any patent or other rights of the SD Card Association or any third party. Nothing herein shall be construed as an obligation by the SD Card Association to disclose or distribute any technical information, know-how or other confidential information to any third party.

Conventions Used in This Document

Naming Conventions

- Register names are shown in italic text such as *Present State*.
- Names of bits or fields within registers are in bold text such as **Buffer Write Enable**.
- Signal names are capitalized, bold and italic, followed by '#' if low active such as ***SDCD#.***
- Some terms are capitalized to distinguish their definition from their common English meaning. Words not capitalized have their common English meaning.
- Register names and the names of fields and bits in registers and headers are presented with the first letter capitalized and the remainder in lower case.

Numbers and Number Bases

- Hexadecimal numbers are written with a lower case "h" suffix, e.g., FFFFh and 80h.
- Binary numbers are written with a lower case "b" suffix (e.g., 10b).
- Binary numbers larger than four digits are written with a space dividing each group of four digits, as in 1000 0101 0010b.
- All other numbers are decimal.

Key Words

- May: Indicates flexibility of choice with no implied recommendation or requirement.
- Shall: Indicates a mandatory requirement. Designers shall implement such mandatory requirements to ensure interchangeability and to claim conformance with the specification.
- Should: Indicates a strong recommendation but not a mandatory requirement. Designers should give strong consideration to such recommendations, but there is still a choice in implementation.

Special Terms

In this document, the following terms shall have special meaning:

- Host Controller Refers to an SD Host Controller that complies with this Specification.
- Host Driver Refers to the OS-specific driver for a Host Controller
- Card Driver Refers to a driver for an SD/SDIO card or card function
- Host System (or System) Refers to the entire system, such as a cellular phone, containing the Host Controller

Implementation Notes

- Some sections of this document provide guidance to Host Controller or Host Driver implementers. To distinguish non-mandatory guidance from other parts of the SD Host Specification, it will be shown as follows:

Implementation Note: This is an example of an implementation note.

Table of Contents

Table of Figures

Table of Tables

1. Overview of the SD Standard Host

The Secure Digital (SD) Host Standard Specification is the SD Association's (SDA) guideline for designing SD Host Controllers and related vendor products. Within the scope of the SD Associations adherence to this specification is not mandatory. It is the Host Controller vendor's responsibility to design products that comply with the SD Specification and where possible to use standard Host Drivers. OS vendor, IHVs and OEMs may require compliance according to their own policies so adherence is recommended.

1.1 Scope of the Standard SD Host

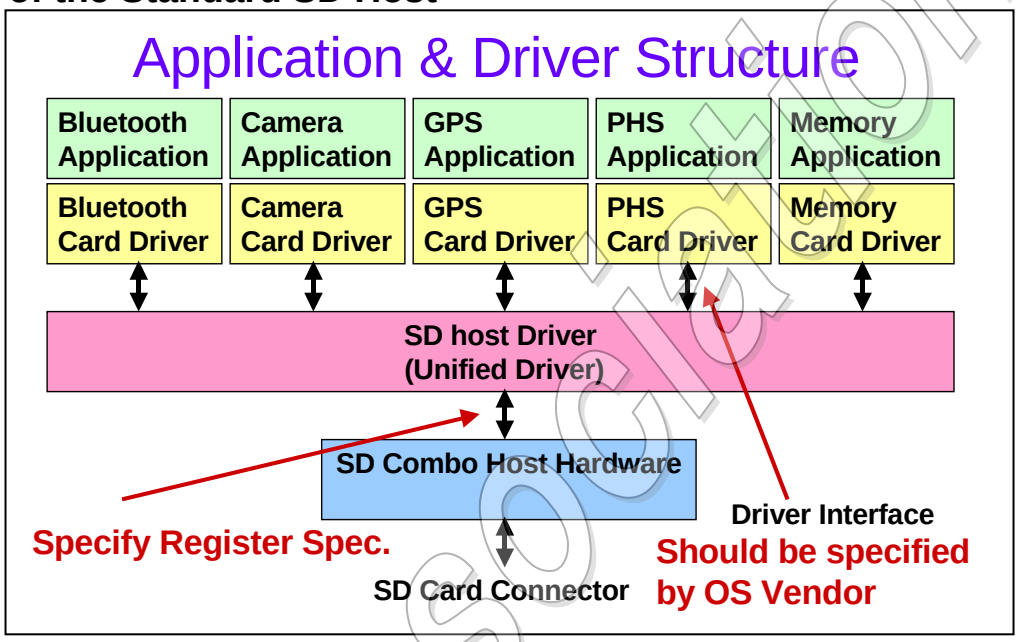


Figure 1-1 : Host Hardware and Driver Architecture

It is the intention of this specification that by defining a standard SD Host Controller, Operating System (OS) Vendors can develop SD Host Drivers that will work with Host Controllers from any vendor. Applications may in addition require the Card Drivers. The Card Drivers will communicate with the Host Driver using a driver interface specified by the OS.

Implementation Note: This specification can be applied to any system bus interface. The interface between the Host Driver and its parent system driver (if any) is not defined by this specification.

1.2 Register Map

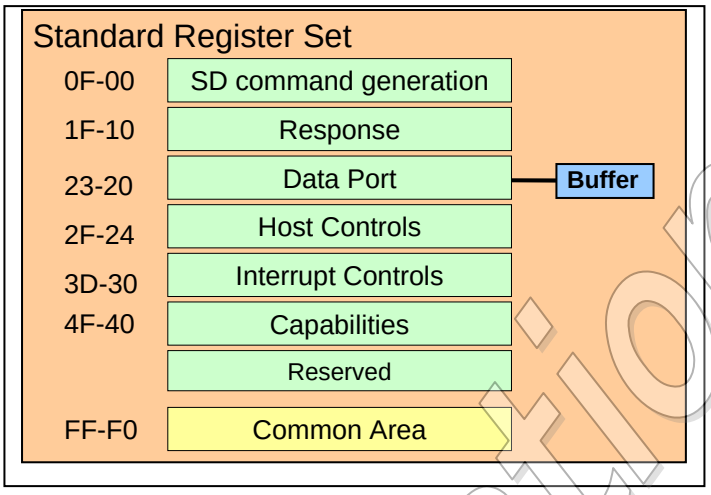


Figure 1-2 : Classification of the Standard Register Map

The standard register map is classified in 7 parts listed below. The Host Controller shall support byte, word and double word accesses to these registers. (The Available Byte Enable Pattern determines the method of buffer access. Refer to Table 1-3.) Reserved bits in all registers shall be fixed to zero. The Host Controller shall ignore writes to reserved bits; however, the Host Driver should write them as zero to ensure compatibility with possible future revisions to this Specification.

(1) SD command generation : Parameters to generate SD commands
(2) Response : Response value from the card
(3) Data port : Data access port to the internal buffer
(4) Host controls : Present State, controls for the SD Bus, Host reset and so on.
(5) Interrupt controls : Interrupt statuses and enables.
(6) Capabilities : Vendor specific host controller information.
(7) Common area : Common information area.

1.3 Multiple Slot support

One Standard Register Set is defined for each slot. If the Host Controller has two slots, two register sets shall be required. Each slot is controlled independently. This enables support for combinations of Full speed and Low speed cards in regards to SD Clock frequencies.

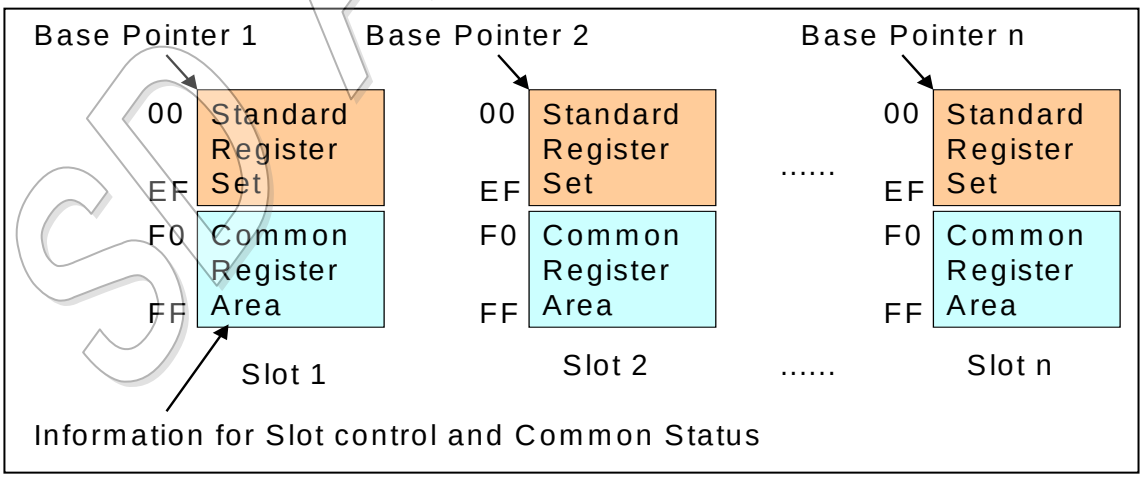


Figure 1-3 : Register Map for Multiple Slot Controller

Figure 1-3 shows the register map for a multiple slot Host Controller. The Host Driver must determine the number of slots and base pointers to each slot's Standard Register Set using system bus or vendor specific methods. Offsets from 0F0h to 0FFh are reserved for the Common register area that defines information for slot control and common status. The common register area is accessible from any slot's register set. This allows software to independently control each slot, since it has access to the *Slot Interrupt Status* register and the *Host Controller Version* register from each register set.

1.4 Supporting DMA

The Host Controller provides a "programmed I/O" method for the Host Driver to transfer data using the *Buffer Data Port* register. Optionally, Host Controller implementers may support data transfer using DMA. Host Controller support for DMA can be determined by checking the **DMA Support** in the *Capabilities* register. Prior to using DMA, the Host Driver must confirm that both the Host Controller and the system bus support it.

DMA allows a peripheral to read and write memory without intervention from the CPU. Only one SD command transaction can be executed by DMA. Host Controllers that support DMA shall support both single block and multiple block transfers. The *System Address* register points to the first data address, and data is then accessed sequentially from that address. Host Controller registers shall remain accessible for issuing non-DAT line commands during a DMA transfer. The result of a DMA transfer shall be the same regardless of the system bus transaction method used. DMA shall not support infinite transfers.
DMA transfers can be stopped and restarted using control bits in the *Block Gap Control* register. When the **Stop At Block Gap Request** is set, DMA transfers shall be suspended. When the **Continue Request** is set or a Resume Command is issued, DMA shall continue to execute transfers. Refer to the *Block Gap Control* register for details. If SD Bus errors occur, SD Bus transfers shall be stopped and DMA transfers shall be stopped. Setting the **Software Reset For DAT Line** in the *Software Reset* register shall abort DMA transfers.

1.5 SD Command generation

	DMA command	CPU data transfer	Non-DAT transfer
System Address	Yes	No	No
Block Size	Yes	Yes	No (Protected)
Block Count	Yes	Yes	No (Protected)
Argument	Yes	Yes	Yes
Transfer Mode	Yes	Yes	No (Protected)
Command	Yes	Yes	Yes

Table 1-1 : Registers to Generate SD Command

Table 1-1 shows register settings (at offsets from 000h to 00Fh in the register set) necessary for three types of transactions: DMA generated transfers, CPU data transfers (using "programmed I/O") and non-DAT transfers. When initiating a transaction, the Host Driver should program these registers sequentially from 000h to 00Fh. The beginning register offset may be calculated based on the type of transaction. The last written offset shall be always 00Fh because writing to the upper byte of the *Command* register shall trigger issuance of an SD command.

The Host Driver should not read the *System Address*, *Block Size* and *Block Count* registers during a data transaction unless the transfer is stopped or suspended because the value is changing and not stable. To prevent destruction of registers using data transfer when issuing command, the *Block Size*, *Block Count* and *Transfer Mode* registers shall be write protected by the Host Controller while **Command Inhibit (DAT)** is set to 1 in the *Present State* register. (The System Address cannot be protected by this signal.) The Host Driver shall not write the *Argument* and *Command* registers while **Command Inhibit (CMD)** is set to 1. The Host Driver shall protect the *System Address*, *Argument* and *Command* registers.

1.6 Suspend and Resume mechanism

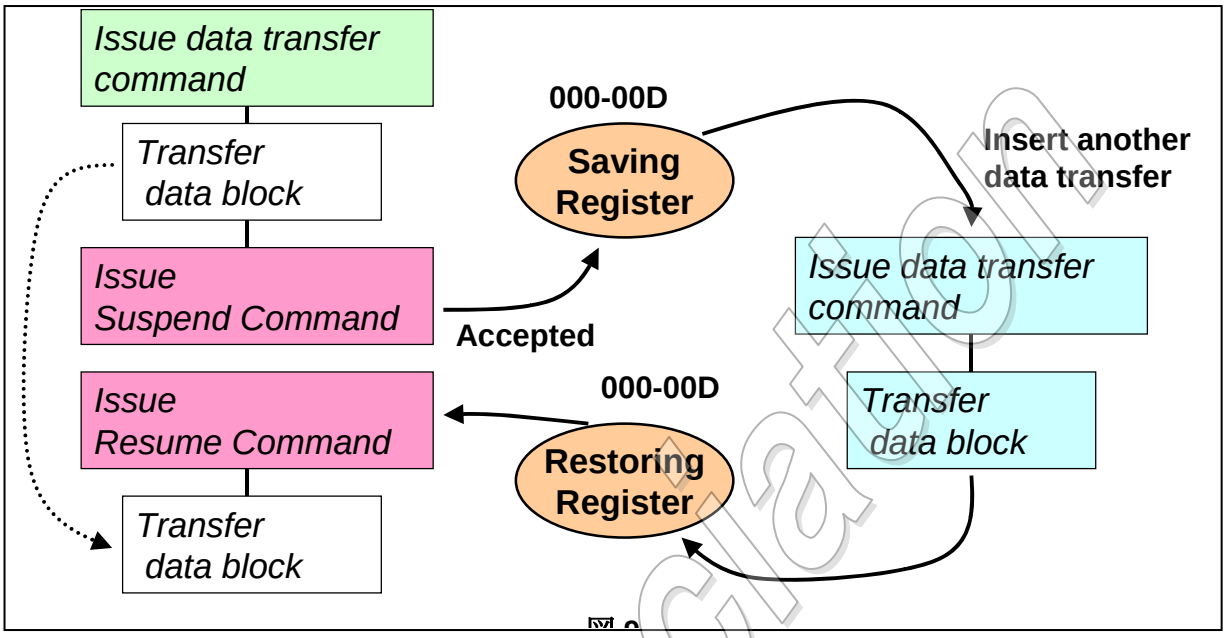


Figure 1-4 : Suspend and Resume mechanism

Support for Suspend/Resume can be determined by checking **Suspend/Resume Support** in the *Capabilities* register. When the SD card accepts a Suspend request, the Host Driver shall save its state information in the first 14bytes (that is, offsets 000h-00dh) of the Host Controller Register Map. The Host Driver shall save these register values before issuing another SD command. On resume, the Host Driver shall restore these registers and then issue the Resume command so that the transfer can continue.

The SD card sets the **DF** (Resume Data Flag) in the response to the Resume command. (Since the Suspend and Resume commands are CMD52 operations, the response data is actually the Function Select Register in the CCCR.) If **DF** is set to 0, it means the SDIO card cannot continue data transfer while suspended. This bit can be used to control data transfers and interrupt cycles. If the Resume Data Flag is set to 0, no more data is transferred and an interrupt cycle is started if the transaction being resumed is in 4-bit mode. If **DF** is set to 1, data transfers continue. The Suspend/Resume protocol is described in the SDIO Specification (SD card Specification part E1).

Note: Card support for the Suspend and Resume commands and for Read Wait are necessary for the Host Controller to be able to support Suspend and Resume.

1.7 Buffer Control

In this section, the terms BE[] and A[] refer to internal signals the Host Controller may use to sequence buffer data access correctly. BE[n] specifies the byte position within a 32-bit double-word of buffer data. A[n] specifies a bit of the address line signal. The Host System asserts BE[03:00] and A[01:00] to selectively access one or more bytes in a 32-bit register

1.7.1 Policy of the Host Buffer Access

The Host Driver accesses the Host Controller's data buffer through the 32-bit *Buffer Data Port* register. Internally, the Host Controller maintains a pointer into the data buffer, and accesses to the *Buffer Data Port* register increment the pointer. The Host Controller's pointer value is not directly accessible by the Host Driver.

In order to accommodate a variety of system busses, the Host Controller implementation of this pointer cannot restrict system bus width. The Host Controller data buffer interface shall have the following characteristics:

(1) Only sequential contiguous access is allowed to this port
Sequential and contiguous access is necessary to increment the pointer correctly. The order is little endian format. For example, BE[1] is accessed, next access has to start form BE[2]. Random or skipped access is not allowed.

(2) Byte enable and address restriction
8-bit, 16-bit or 32-bit system busses are supported. To specify byte position for Buffer Data Port (4 bytes), Byte Enable (BE[]) or Lower Address (A[]) is used. Table 1-2 shows the relation between lower address and byte enable depending on system bus width.

System Bus	A[01]	A[00]	BE[03:02] D[31:16]	BE[01] D[15:08]	BE[00] D[07:00]
32-bit	No	No	Yes	Yes	Yes
16-bit	Yes	No	No	Yes	Yes
8-bit	Yes	Yes	No	No	Yes *

Yes : Use for control
No : Not used

* : BE[00] for 8-bit bus is always 1 therefore it may not be defined.

Table 1-2 : Relations Between Address and Byte Enable

Table 1-3 shows the possible byte enable pattern for 32-bit system bus. Byte enable has to be managed to fit the sequential address position. Skipping Byte enable is not allowed. Therefore there are some restrictions for byte enable sequence pattern. For example:

OK BE[3:0]=0011b (2-byte) => BE[3:0]=0100b (1-byte) => BE[3:0]=1000b (1-byte)
OK BE[3:0]=0001b (1-byte) => BE[3:0]=0010b (1-byte) => BE[3:0]=1100b (2-byte)
Not OK BE[3:0]=0001b (2-byte) => BE[3:0]=1100b (2-byte) (Cannot skip BE[1])

Byte Enable		BE[3]	BE[2]	BE[1]	BE[0]
Data Bus		D[31:24]	D[23:16]	D[15:08]	D[07:00]
Access Type	4-byte	1	1	1	1
	2-byte	0	0	1	1
	2-byte	1	1	0	0
	1-byte	0	0	0	1
	1-byte	0	0	1	0
	1-byte	0	1	0	0
	1-byte	1	0	0	0

1 : Data Valid
0 : Data not valid

Table 1-3 : Available Byte Enable Pattern for 32-bit System Bus

Implementation Note: Table 1-3 implies that the Host Driver should align register accesses on address boundaries matching the number of bytes in the access. That is, single byte accesses may be aligned on any offset within the register set; word (double byte) accesses should be aligned on two-byte offsets; and double-word (quad byte) accesses should be aligned on four-byte offsets.

(3) Border control of buffer block size

The buffer accumulates the data written through the data port and flushes when the data count reaches the block size limit specified in the *Block Size* register. The Host Controller shall not start data transmission until a full block of data is written to the data port. Block size may be specified, by the Host Driver, as any value from 1 byte to the maximum block size. Excess data shall be ignored. For write operations, if 2 bytes are left to reach the block count and a 32-bit (4-byte) block of data is written to the *Buffer Data Port* register, then the lower 2 bytes of data shall be written and the upper 2 bytes shall be ignored. The next block starts from BE[00]. For read operations, if just 2 bytes of data are in the buffer and a 32-bit read is performed, then the lower 2 bytes shall be valid but the upper 2 bytes shall be undefined. The next block data shall start from BE[00]. This ensures that every block starts from BE[00] (the least significant byte of the *Buffer Data Port* register) for multiple block data transfers. As a result, the Host Driver may repeatedly access the *Buffer Data Port* register with the maximum 32-bit width for any data length.

1.7.2 Determining Buffer block length

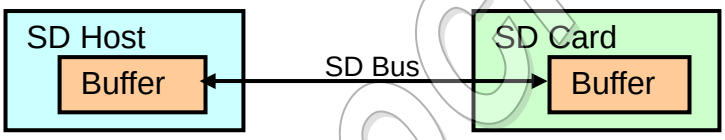


Figure 1-5 : Buffer size relation between the SD Host and the SD Card

To be able to transfer blocks of data at a burst, the relationship between Host Controller and SD card buffer sizes is important. The Host Driver shall use the same data block length for both the Host Controller and the card. If the controller and card buffer sizes are different, the Host Driver shall use the smaller value. The maximum Host Controller buffer size is defined by the **Max Block Length** field in the *Capabilities* register.

Implementation Note: The card buffer size is described in the Card Specific Data (CSD) register for memory cards (for cards compliant with the SD Physical Specification, READ_BL_LEN and WRITE_BL_LEN shall be the same) and in the Card Information Structure (CIS) for SDIO cards. If the card has multiple buffers, the block size in CIS indicates the size of the smallest buffer; this is the value the Host Driver should use when programming block size. Buffer information (for example, buffer port address) in the SD card is function specific.

1.7.3 Dividing large data transfer

The SDIO command CMD53 definition limits the maximum data size of data transfers according to the following formula:

Max data size = Block size x Block count

For example, Block size is specified by the buffer size as described in 1.7.2 and the block count can be a maximum of 512 (9-bit count) as specified in the command argument for CMD53. In the worst case, if the card has only a 1 byte buffer, up to 512 bytes of data can be transferred using CMD53 (Block Size =1, Block Count = 512). If the card does not support multiple block mode, only one byte can be transferred in this case. If an application or Card Driver wants to transfer larger sizes of data, the Host Driver shall divide large data into multiple CMD53 blocks.

1.7.4 Data lengths which cannot be divided by block size

The length of a multiple block transfer needs to be in block size units. If the total data length cannot be divided evenly by the block size, there are two ways to transfer the data depending on the function and card design:

Option 1 is for the Card Driver to split the transaction. The remainder of block size data is transferred by using a single block command at the end.
Option 2 is to add dummy data in the last block to fill the block size. This is efficient but the card must manage or remove the dummy data.

Implementation Note: The Host Driver interface to the Card Drivers should accept any size data. The Host Driver can always rely on Option 1 if the data size is not a multiple of the block size. The Host Driver may expose Option 2 to the Card Drivers, or may expect the Card Drivers to implement Option 2 themselves if they need the additional optimization. This document does not specify the interface between the Host Driver and the Card Drivers.

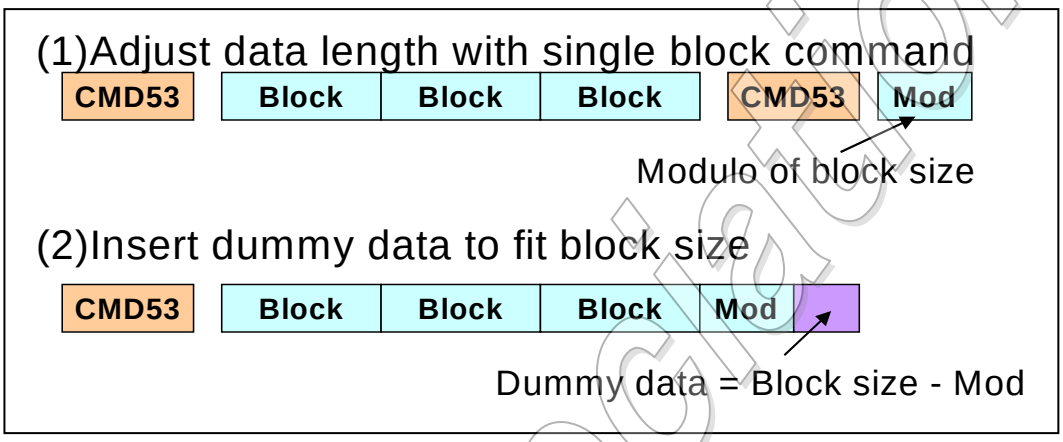


Figure 1-6 : Two cases to handle a transfer of data which can not be divided by block size

1.8 Relationship between Interrupt Control Registers

The Host Controller implements a number of interrupt sources. Interrupt sources can be enabled as interrupts or as system wakeup signals as shown in Figure 1-7. If the interrupt source's corresponding bit in the *Normal Interrupt Status Enable* or *Error Interrupt Status Enable* register is 1 and the interrupt becomes active its active state is latched and made available to the Host Driver in the *Normal Interrupt Status* register or the *Error Interrupt Status* register. *Interrupt Status* shall be cleared when *Interrupt Status Enable* is cleared. (This is not expressed in the Figure 1-7.)

An interrupt source with its bit set in an interrupt status register shall assert a system interrupt signal if its corresponding bit is also set in the *Normal Interrupt Signal Enable* register or the *Error Interrupt Signal Enable* register. Once signaled, most interrupts are cleared by writing a 1 to the associated bit in the interrupt status register. Card interrupts, however, must be cleared by the Card Driver. If the **Card Interrupt** is generated, the Host Driver shall clear **Card interrupt Status Enable** before the Card Driver clears interrupt sources. After all interrupt sources are cleared, the Host Driver shall set it again to enable another card interrupt.

The *Wakeup Control* register enables **Card Interrupt**, **Card Insertion,** or **Card Removal** status changes to be configured to generate a system wakeup signal. These interrupts are enabled or masked independently of the *Normal Interrupt Signal Enable* register. The kind of wakeup event can be read from *Normal Interrupt Status* register.

The interrupt signal and wakeup signal are logical ORed and shall be read from the *Slot Interrupt Status* register.

Implementation Note: The Host Driver is responsible for enabling wakeup signals and disabling interrupt signals when the Host System goes into its sleep mode, and for disabling wakeup signals and enabling interrupt signals when the Host System goes into run mode. The Host Driver should not enable both at the same time.

Implementation Note: The Host Systems may implement interrupt and wakeup signals in various ways. For example, the PCI bus supports ***PME#***, which can be asserted without PCI clock, then interrupts use ***INTx#*** and wakeups use ***PME#***. Alternatively, the system may use an ORed signal of interrupt and wakeup if the system bus supports one interrupt line to the Host Controller.

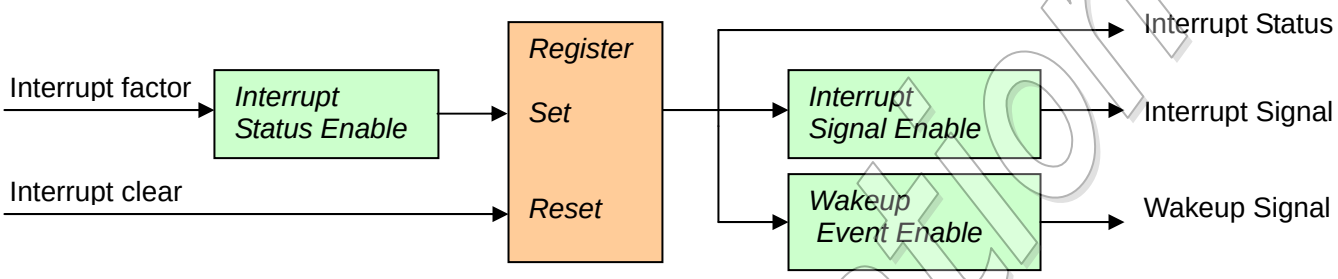


Figure 1-7 : Logical relation for interrupt registers

Interrupt Status Enable	Interrupt Signal Enable	Wakeup Event Enable	Interrupt Status	Interrupt Signal
0 (Mask)	x (don't care)	x (don't care)	0 (Not exist)	0 (De-assert)
1 (Enable)	0 (Mask)	x (don't care)	x (don't care)	0 (De-assert)
1 (Enable)	1 (Enable)	x (don't care)	0 (Not exist)	0 (De-assert)
1 (Enable)	1 (Enable)	x (don't care)	1 (Exist)	1 (Assert)

Table 1-4 : Interrupt Signal Table

Interrupt Status Enable	Interrupt Signal Enable	Wakeup Event Enable	Interrupt Status	Wakeup Signal
0 (Mask)	x (don't care)	x (don't care)	0 (Not exist)	0 (De-assert)
1 (Enable)	x (don't care)	0 (Mask)	x (don't care)	0 (De-assert)
1 (Enable)	x (don't care)	1 (Enable)	0 (Not exist)	0 (De-assert)
1 (Enable)	x (don't care)	1 (Enable)	1 (Exist)	1 (Assert)

Table 1-5 : Wakeup Signal Table

Implementation Note: The Host Controller may implement asserted wakeup or interrupt signals as active high or active low.

1.9 HW Block Diagram and timing part

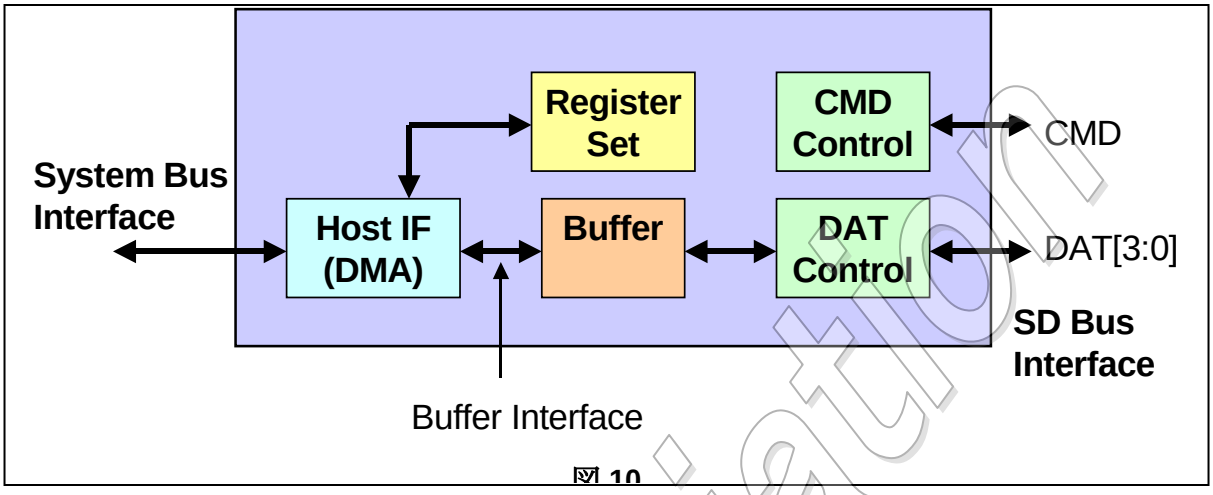


Figure 1-8 : Block Diagram of Host Controller

The Host Controller has two bus interfaces, the System Bus Interface and the SD Bus Interface. The Host Controller assumes that these interfaces are asynchronous (that is, are working on different clock frequencies). The Host Driver is on system bus time (because it is software executed by the Host Controller CPU, on its system clock). The SD card is on SD Bus time (that is, its operation is synchronized by ***SDCLK***). The Host Controller shall synchronize signals to communicate between these interfaces. Blocks of data shall be synchronized at the buffer module. All status registers shall be synchronized by the system clock and maintain synchronization during output to the system interface (except, optionally, statuses for testing). Control registers, which trigger SD Bus transactions, shall be synchronized by ***SDCLK***. Therefore, there will be a timing delay when propagating signals between the two interfaces. This means the Host Driver cannot do real time control of the SD Bus and must rely on the Host Controller to control the SD Bus according to register settings.

The Buffer Interface enables internal read and write buffers (refer to use of the **Buffer Read Enable** and **Buffer Write Enable** in the *Present State* register *as described in section* 1.7 "Buffer Control").
The **Transfer Complete** interrupt status indicates completion of the read / write transfer for both DMA and non DMA transfers. However, the timing is different between reads and writes. Read transfers shall complete after all valid data has been transferred to the Host System and are ready for the Host Driver to access. Write transfers shall complete after all valid data has been transferred to the SD card and the busy state is over.

Table 1-6 shows the relation between statuses and interrupts for data transfer.

Type of Data transfer	Buffer Status	Buffer Ready	Complete Status
Write Transfer (Non DMA)	Buffer Write Enable	Buffer Write Ready	Transfer Complete
Write Transfer (DMA)	(Driver ignores)	(Driver ignores)	Transfer Complete
Read Transfer (Non DMA)	Buffer Read Enable	Buffer Read Ready	Transfer Complete
Read Transfer (DMA)	(Driver ignores)	(Driver ignores)	Transfer Complete

Table 1-6 : Summary of Register Status for data transfer

1.10 Power state definition of SD Host Controller

Implementation Note: Table 1-7 defines controller power states, which are listed in increasing order of power consumption. The Host Controller should reduce the power consumption by using these conditions.

SD Card	Internal Clock *1	SD Power	SD Clock	SD Bus	Power State *2	Comment
No exist	Stop	OFF	Stop	-	P00	Host not used
	Oscillate	OFF	Stop	-	P01	No card
	Oscillate	ON	Stop	-	P02	Short transition state *3
	Oscillate	ON	Oscillate	-	P03	Short transition state *3
Exist	Stop	OFF	Stop	-	P10	Host not used
	Oscillate	OFF	Stop	-	P11	Low power mode
	Oscillate	ON	Stop	-	P12	Wakeup
	Oscillate	ON	Oscillate	Wait	P13	Ready to issue command
	Oscillate	ON	Oscillate	Access	P14	During transaction

Table 1-7 : Power State Definition

Implementation Note:
*1: Internal clock should be stopped when the Host System does not use the Host Controller.
*2: Power states are not actually implemented in Host Controller. This label is for reference.
*3: Short transition state: Temporary power states. The Host Controller automatically goes to P01 when it detects No Card.

The SD Clock shall not oscillate when card power is OFF.
States described in Table 1-7 can be determined by reading the corresponding register bits:

Internal clock oscillate/stop	**Internal Clock Enable** in the *Clock Control* register
SD Card : Exist/Not exist	**Card Inserted** in the *Present State* register
SD Power : ON/OFF	**SD Bus Power** in the *Power Control* register
SD Clock : oscillate/stop	**SD Clock Enable** in the *Clock Control* register
SD Bus : access/wait (idle)	**Command Inhibit (CMD)** and **Command Inhibit (DAT)** in the *Present State* register

1.11 Auto CMD12

Multiple block transfers for SD memory require CMD12 to stop the transactions. The Host Controller automatically issues CMD12 when the last block transfer is completed. This feature of the Host Controller is called Auto CMD12. The Host Driver should set **Auto CMD12 Enable** in the *Transfer Mode* register when issuing a multiple block transfer command. Auto CMD12 timing synchronization with the last data block shall be done by hardware in the Host Controller. Commands that don't use the DAT line can be issued during multiple block transfers. These commands are referred to using the notation CMD_wo_DAT.

In order to prevent DAT line commands and CMD_wo_DAT commands from conflicting, the Host Controller shall arbitrate the timing by which each command is issued on the SD Bus. Therefore, a command might not immediately be issued after the Host Driver writes to the *Command* register. The command may be issued before or after Auto CMD12, depending on the timing. To be able to distinguish the responses of DAT line and CMD_wo_DAT commands, the Auto CMD12 response can be determined from the upper four bytes of the *Response* register (at offset 01Ch in the standard register set).

If errors are detected related to Auto CMD12, the Host Controller shall issue an **Auto CMD12 Error** interrupt. The Host Driver can check the Auto CMD12 error status (Command Index/End bit/CRC/Timeout Error) by reading the *Auto CMD12 Error Status* register.

The Table 1-8 illustrates the relationship between Auto CMD12 errors and any CMD_wo_DAT commands that have been issued by the Host Driver.

Relation of the commands	Error Status	Comments
Auto CMD12 only	CMD_wo_DAT : Unrelated Auto CMD12 : Error	Only Auto CMD12 is issued, therefore Auto CMD12 is failed.
CMD_wo_DAT before Auto CMD12	CMD_wo_DAT : No Error Auto CMD12 : Error	CMD_wo_DAT successful, but Auto CMD12 failed.
CMD_wo_DAT before Auto CMD12	CMD_wo_DAT : Error Auto CMD12 : Not executed	CMD_wo_DAT failed, therefore Auto CMD12 could not be issued.
Auto CMD12 before CMD_wo_DAT	CMD_wo_DAT : Not executed Auto CMD12 : Error	Auto CMD12 failed, therefore CMD_wo_DAT could not be issued.

Table 1-8 : Relation between Auto CMD12 and CMD_wo_DAT

The Host Driver may determine which of these error cases has occurred by checking the *Auto CMD12 Error Status* register when an **Auto CMD12 Error** interrupt occurs. If the Auto CMD12 was not executed, the Host Driver needs to recover from the CMD_wo_DAT error and issue CMD12 to stop the multiple block transfer. If the CMD_wo_DAT was not executed, the Host Driver can issue it again after recovering from the Auto CMD12 error. The procedures for recovering from error interrupts and from Auto CMD12 errors are described in sections 3.8.1 and 3.8.2.

2. SD Host Standard Register

2.1 Summary of register set

2.1.1 SD Host Control Register Map

Table 2-1 summarizes the standard SD Host Controller register set. The Host Driver needs to determine the base address of the register set by a Host System specific method. The register set is 256 bytes in size. For multiple slot controllers, one register set is assigned per each slot, but the registers at offsets 0F0h-0FFh are assigned as a common area. These registers contain the same values for each slot's register set.

Offset	15-08 bit	07-00 bit	Offset	15-08 bit	07-00 bit
002h	System Address (High)		000h	System Address (Low)	
006h	Block Count		004h	Block Size	
00Ah	Argument1		008h	Argument0	
00Eh	Command		00Ch	Transfer Mode	
012h	Response1		010h	Response0	
016h	Response3		014h	Response2	
01Ah	Response5		018h	Response4	
01Eh	Response7		01Ch	Response6	
022h	Buffer Data Port1		020h	Buffer Data Port0	
026h	Present State		024h	Present State	
02Ah	Wakeup Control	Block Gap Control	028h	Power Control	Host Control
02Eh	Software Reset	Timeout Control	02Ch	Clock Control	
032h	Error Interrupt Status		030h	Normal Interrupt Status	
036h	Error Interrupt Status Enable		034h	Normal Interrupt Status Enable	
03Ah	Error Interrupt Signal Enable		038h	Normal Interrupt Signal Enable	
03Eh	---		03Ch	Auto CMD12 Error Status	
042h	Capabilities		040h	Capabilities	
046h	Capabilities (Reserved)		044h	Capabilities (Reserved)	
04Ah	Maximum Current Capabilities		048h	Maximum Current Capabilities	
04Eh	Maximum Current Capabilities (Reserved)		04Ch	Maximum Current Capabilities (Reserved)	
---	---		---	---	
0F2h	---		0F0h	---	
---	---		---	---	
0FEh	Host Controller Version		0FCh	Slot Interrupt Status	

Table 2-1 : SD Host Controller Register Map

2.1.2 Configuration Register Types

Configuration register fields are assigned one of the attributes described below:

Register Attribute	Description
RO	Read-only register: Register bits are read-only and cannot be altered by software or any reset operation. Writes to these bits are ignored.
ROC	Read-only status: These bits are initialized to zero at reset. Writes to these bits are ignored.
RW	Read-Write register: Register bits are read-write and may be either set or cleared by software to the desired state.
RW1C	Read-only status, Write-1-to-clear status: Register bits indicate status when read, a set bit indicating a status event may be cleared by writing a 1. Writing a 0 to RW1C bits has no effect.
RWAC	Read-Write, automatic clear register: The Host Driver requests a Host Controller operation by setting the bit. The Host Controllers shall clear the bit automatically when the operation of complete. Writing a 0 to RWAC bits has no effect.
HwInit	Hardware Initialized: Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. Bits are read-only after initialization, and writes to these bits are ignored.
Rsvd	Reserved. These bits are initialized to zero, and writes to them are ignored.

Table 2-2 : Register (and Register Bit-Field) Types

Implementation Note: If the Host Driver writes to RO, ROC, HwInit and Rsvd bits, the Host Driver should write these bits as zero to avoid possible compatibility problems with future versions of this specification.

2.1.3 Register Initial values

The Host Controller shall set all registers to their initial values at power-on reset. Initial values of the register are defined as follows. All other registers' default value shall be all bits set to zero.

(1) Value of the *Capabilities* register and *Maximum Current Capabilities* register depends on the Host Controller.
(2) Value of the *Host Controller Version* register depends on the Host Controller.

2.1.4 Reserved bits of register

"Reserved" means the bit can be defined for future use and is currently set to 0. These bits should be written as zero.

2.2 SD Host Standard Register

2.2.1 System Address Register (Offset 000h)

This register contains the physical system memory address used for DMA transfers.

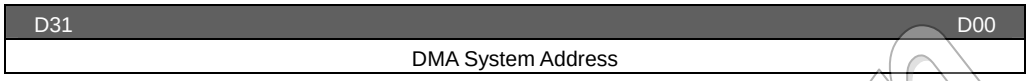


Figure 2-1 : System Address Register

Location	Attrib	
31-00	RW	**DMA System Address** This register contains the system memory address for a DMA transfer. When the Host Controller stops a DMA transfer, this register shall point to the system address of the next contiguous data position. It can be accessed only if no transaction is executing (i.e., after a transaction has stopped). Read operations during transfers may return an invalid value. The Host Driver shall initialize this register before starting a DMA transaction. After DMA has stopped, the next system address of the next contiguous data position can be read from this register. The DMA transfer waits at the every boundary specified by the **Host DMA Buffer Boundary** in the *Block Size* register. The Host Controller generates **DMA Interrupt** to request the Host Driver to update this register. The Host Driver set the next system address of the next data position to this register. When the most upper byte of this register (003h) is written, the Host Controller restart the DMA transfer. When restarting DMA by the Resume command or by setting **Continue Request** in the *Block Gap Control* register, the Host Controller shall start at the next contiguous address stored here in the *System Address* register.

Table 2-3 : System Address Register

2.2.2 Block Size Register (Offset 004h)

This register is used to configure the number of bytes in a data block.

D15	D14 D12	D11 D00
Rsvd	Host DMA Buffer Boundary	Transfer Block Size

Figure 2-2 : Block Size Register

Location	Attrib	
15	Rsvd	Reserved
14-12	RW	**Host DMA Buffer Boundary** The large contiguous memory space may not be available in the virtual memory system. To perform long DMA transfer, *System Address* register shall be updated at every system memory boundary during DMA transfer. These bits specify the size of contiguous buffer in the system memory. The DMA transfer shall wait at the every boundary specified by these fields and the Host Controller generates the **DMA Interrupt** to request the Host Driver to update the *System Address* register. In case of this register is set to 0 (buffer size = 4K bytes), lower 12-bit of byte address points data in the contiguous buffer and the upper 20-bit points the location of the buffer in the system memory. The DMA transfer stops when the Host Controller detects carry out of the address from bit 11 to 12. These bits shall be supported when the **DMA Support** in the *Capabilities* register is set to 1 and this function is active when the **DMA Enable** in the *Transfer Mode* register is set to 1. 000b 4K bytes (Detects A11 carry out) 001b 8K bytes (Detects A12 carry out) 010b 16K Bytes (Detects A13 carry out) 011b 32K Bytes (Detects A14 carry out) 100b 64K bytes (Detects A15 carry out) 101b 128K Bytes (Detects A16 carry out) 110b 256K Bytes (Detects A17 carry out) 111b 512K Bytes (Detects A18 carry out)
11-00	RW	**Transfer Block Size** This register specifies the block size for block data transfers for CMD17, CMD18, CMD24, CMD25, and CMD53. Values ranging from 1 up to the maximum buffer size can be set (Refer to 1.7.2). It can be accessed only if no transaction is executing (i.e., after a transaction has stopped). Read operations during transfers may return an invalid value, and write operations shall be ignored. 0800h 2048 Bytes … … 0200h 512 Bytes 01FFh 511 Bytes … … 0004h 4 Bytes 0003h 3 Bytes 0002h 2 Bytes 0001h 1 Byte 0000h No data transfer

Table 2-4 : Block Size Register

2.2.3 Block Count Register (Offset 006h)

This register is used to configure the number of data blocks.

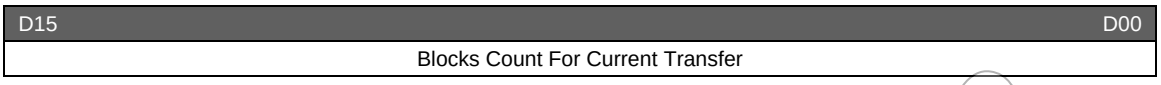


Figure 2-3 : Block Count Register

Location	Attrib	
15-00	RW	**Blocks Count For Current Transfer** This register is enabled when **Block Count Enable** in the *Transfer Mode* register is set to 1 and is valid only for multiple block transfers. The Host Driver shall set this register to a value between 1 and the maximum block count. The Host Controller decrements the block count after each block transfer and stops when the count reaches zero. Setting the block count to 0 results in no data blocks being transferred. This register should be accessed only when no transaction is executing (i.e., after transactions are stopped). During data transfer, read operations on this register may return an invalid value and write operations are ignored. When saving transfer context as a result of a Suspend command, the number of blocks yet to be transferred can be determined by reading this register. When restoring transfer context prior to issuing a Resume command, the Host Driver shall restore the previously saved block count. FFFFh: 65535 blocks …: … 0002h: 2 blocks 0001h: 1 block 0000h: Stop Count

Table 2-5 : Block Count Register

2.2.4 Argument Register (Offset 008h)

This register contains the SD Command Argument.

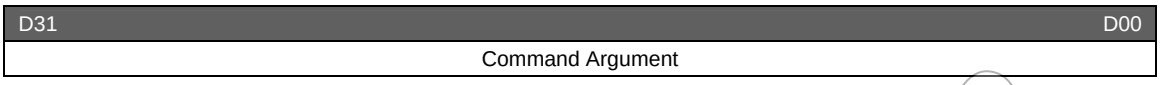


Figure 2-4 : Argument Register

Location	Attrib	
31-00	RW	**Command Argument** The SD Command Argument is specified as bit39-8 of Command-Format in the SD Memory Card Physical Layer Specification.

Table 2-6 : Argument Register

2.2.5 Transfer Mode Register (Offset 00Ch)

This register is used to control the operation of data transfers. The Host Driver shall set this register before issuing a command which transfers data (see **Data Present Select** in the *Command* register), or before issuing a Resume command. The Host Driver shall save the value of this register when the data transfer is suspended (as a result of a Suspend command) and restore it before issuing a Resume command. To prevent data loss, the Host Controller shall implement write protection for this register during data transactions. Writes to this register shall be ignored when the **Command Inhibit (DAT)** in the *Present State* register is 1.

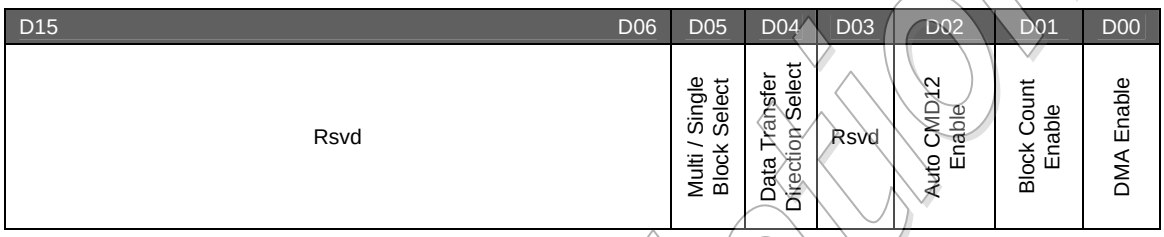


Figure 2-5 : Transfer Mode Register

Location	Attrib	
15-06	Rsvd	**Reserved**
05	RW	**Multi / Single Block Select** This bit enables multiple block DAT line data transfers. For any other commands, this bit shall be set to 0. If this bit is 0, it is not necessary to set the *Block Count* register. (Refer to Table 2-8) 1 – Multiple Block 0 – Single Block
04	RW	**Data Transfer Direction Select** This bit defines the direction of DAT line data transfers. The bit is set to 1 by the Host Driver to transfer data from the SD card to the SD Host Controller and it is set to 0 for all other commands. 1 – Read (Card to Host) 0 – Write (Host to Card)
03	Rsvd	**Reserved**
02	RW	**Auto CMD12 Enable** Multiple block transfers for memory require CMD12 to stop the transaction. When this bit is set to 1, the Host Controller shall issue CMD12 automatically when last block transfer is completed. The Host Driver shall not set this bit to issue commands that do not require CMD12 to stop data transfer. In particular, secure commands defined in the Part 3 File Security specification do not require CMD12. 1 – Enable 0 – Disable
01	RW	**Block Count Enable** This bit is used to enable the *Block Count* register, which is only relevant for multiple block transfers. When this bit is 0, the *Block Count* register is disabled, which is useful in executing an infinite transfer. (Refer to Table 2-8) 1 – Enable 0 – Disable

00	RW	**DMA Enable** This bit enables DMA functionality as described in section 1.4. DMA can be enabled only if it is supported as indicated in the **DMA Support** in the *Capabilities* register. If DMA is not supported, this bit is meaningless and shall always read 0. If this bit is set to 1, a DMA operation shall begin when the Host Driver writes to the upper byte of *Command* register (00Fh). 1 Enable 0 Disable

Table 2-7 : Transfer Mode Register

Table 2-8 shows the summary of how register settings determine types of data transfer.

Multi/Single Block Select	Block Count Enable	*Block Count*	Function
0	Don't care	Don't care	Single Transfer
1	0	Don't care	Infinite Transfer
1	1	Not Zero	Multiple Transfer
1	1	Zero	Stop Multiple Transfer

Table 2-8 : Determination of Transfer Type

2.2.6 Command Register (Offset 00Eh)

The Host Driver shall check the **Command Inhibit (DAT)** bit and **Command Inhibit (CMD)** bit in the *Present State* register before writing to this register. Writing to the upper byte of this register triggers SD command generation. The Host Driver has the responsibility to write this register because the Host Controller does not protect for writing when **Command Inhibit (CMD)** is set.

D15 D14	D13 D08	D07 D06	D05	D04	D03	D02	D01 D00
Rsvd	Command Index	Command Type	Data Present Select	Command Index Check Enable	Command CRC Check Enable	Rsvd	Response Type Select

Figure 2-6 : Command Register

Location	Attrib	
15-14	Rsvd	**Reserved**
13-08	RW	**Command Index** These bits shall be set to the command number (CMD0-63, ACMD0-63) that is specified in bits 45-40 of the Command-Format in the SD Memory Card Physical Layer Specification and SDIO Card Specification.
07-06	RW	**Command Type** There are three types of special commands: Suspend, Resume and Abort. These bits **shall** be set to 00b for all other commands. • Suspend Command If the Suspend command succeeds, the Host Controller shall assume the SD Bus has been released and that it is possible to issue the next command which uses the ***DAT*** line. The Host Controller shall de-assert Read Wait for read transactions and stop checking busy for write transactions. The interrupt cycle shall start, in 4-bit mode. If the Suspend command fails, the Host Controller shall maintain its current state, and the Host Driver shall restart the transfer by setting **Continue Request** in the *Block Gap Control* register. (Refer to 3.10.1 Suspend Sequence) • Resume Command The Host Driver re-starts the data transfer by restoring the registers in the range of 000-00Dh. (Refer to Figure 1-4 in section 1.6 for the register map.) The Host Controller shall check for busy before starting write transfers. • Abort Command If this command is set when executing a read transfer, the Host Controller shall stop reads to the buffer. If this command is set when executing a write transfer, the Host Controller shall stop driving the ***DAT*** line. After issuing the Abort command, the Host Driver should issue a software reset. (Refer to 3.7 Abort Transaction)

11b	Abort	CMD12, CMD52 for writing "I/O Abort" in CCCR
10b	Resume	CMD52 for writing "Function Select" in CCCR
01b	Suspend	CMD52 for writing "Bus Suspend" in CCCR
00b	Normal	Other commands

<table>
<tr><td>05</td><td>RW</td><td>Data Present Select
This bit is set to 1 to indicate that data is present and shall be transferred using the DAT line. It is set to 0 for the following:
(1) Commands using only CMD line (ex. CMD52).
(2) Commands with no data transfer but using busy signal on DAT[0] line (R1b or R5b ex. CMD38)
(3) Resume command

<table><tr><td>1</td><td>Data Present</td></tr><tr><td>0</td><td>No Data Present</td></tr></table></td></tr>
<tr><td>04</td><td>RW</td><td>Command Index Check Enable
If this bit is set to 1, the Host Controller shall check the Index field in the response to see if it has the same value as the command index. If it is not, it is reported as a Command Index Error. If this bit is set to 0, the Index field is not checked.

<table><tr><td>1</td><td>Enable</td></tr><tr><td>0</td><td>Disable</td></tr></table></td></tr>
<tr><td>03</td><td>RW</td><td>Command CRC Check Enable
If this bit is set to 1, the Host Controller shall check the CRC field in the response. If an error is detected, it is reported as a Command CRC Error. If this bit is set to 0, the CRC field is not checked. The number of bits checked by the CRC field value changes according to the length of the response. (Refer to D01-00 and Table 2-10 below.)

<table><tr><td>1</td><td>Enable</td></tr><tr><td>0</td><td>Disable</td></tr></table></td></tr>
<tr><td>02</td><td>Rsvd</td><td>Reserved</td></tr>
<tr><td>01-00</td><td>RW</td><td>Response Type Select

<table><tr><td>00</td><td>No Response</td></tr><tr><td>01</td><td>Response Length 136</td></tr><tr><td>10</td><td>Response Length 48</td></tr><tr><td>11</td><td>Response Length 48 check Busy after response</td></tr></table></td></tr>
</table>

Table 2-9 : Command Register

Response Type	Index Check Enable	CRC Check Enable	Name of Response Type
00	0	0	No Response
01	0	1	R2
10	0	0	R3, R4
10	1	1	R1, R6, R5
11	1	1	R1b, R5b

Table 2-10 : Relation Between Parameters and the Name of Response Type

These bits determine Response types.

Note : In the SDIO specification, response type notation of R5b is not defined. R5 includes R5b in the SDIO specification. But R5b is defined in this specification to specify the Host Controller shall check busy after receiving response. For example, usually CMD52 is used as R5 but I/O abort command shall be used as R5b.

Implementation Note: the CRC field for R3 and R4 is expected to be all "1" bits. The CRC check should be disabled for these response types.

2.2.7 Response Register (Offset 010h)

This register is used to store responses from SD cards.

Offset 010h	D31	D00
	Command Response 0 - 31	
Offset 014h	D31	D00
	Command Response 32 - 63	
Offset 018h	D31	D00
	Command Response 64 - 95	
Offset 01Ch	D31	D00
	Command Response 96 - 127	

Figure 2-7 : Response Register

Location	Attrib	
127-00	ROC	**Command Response** The Table 2-12 describes the mapping of command responses from the SD Bus to this register for each response type. In the table, R[] refers to a bit range within the response data as transmitted on the SD Bus, REP[] refers to a bit range within the *Response* register.

Table 2-11 : Response Register

Kind of Response	Meaning of Response	Response Field	Response Register
R1, R1b (normal response)	Card Status	R [39:8]	REP [31:0]
R1b (Auto CMD12 response)	Card Status for Auto CMD12	R [39:8]	REP [127:96]
R2 (CID, CSD register)	CID or CSD reg. incl.	R [127:8]	REP [119:0]
R3 (OCR register)	OCR register for memory	R [39:8]	REP [31:0]
R4 (OCR register)	OCR register for I/O etc	R [39:8]	REP [31:0]
R5,R5b	SDIO response	R [39:8]	REP [31:0]
R6 (Published RCA response)	New published RCA[31:16] etc	R [39:8]	REP [31:0]

Table 2-12 : Response Bit Definition for Each Response Type.

The Response Field indicates bit positions of "Responses" defined in the PHYSICAL LAYER SPECIFICATION Version 1.01.

The Table 2-12 shows that most responses with a length of 48 (R[47:0]) have 32 bits of the response data (R[39:8]) stored in the *Response* register at REP[31:0]. Responses of type R1b (Auto CMD12 responses) have response data bits R[39:8] stored in the *Response* register at REP[127:96]. Responses with length 136 (R[135:0]) have 120 bits of the response data (R[127:8]) stored in the *Response* register at REP[119:0].

To be able to read the response status efficiently, the Host Controller only stores part of the response data in the *Response* register. This enables the Host Driver to efficiently read 32 bits of response data in one read cycle on a 32-bit bus system. Parts of the response, the Index field and the CRC, are checked by the Host Controller (as specified by the **Command Index Check Enable** and the **Command CRC Check Enable** bits in the *Command* register) and generate an error interrupt if an error is detected. The bit range for the CRC check depends on the response length. If the response length is 48, the Host Controller shall check R[47:1], and if the response length is 136 the Host Controller shall check R[119:1].

Since the Host Controller may have a multiple block data DAT line transfer executing concurrently with a CMD_wo_DAT command, the Host Controller stores the Auto CMD12 response in the upper bits (REP[127:96]) of the *Response* register. The CMD_wo_DAT response is stored in REP[31:0]. This allows the Host Controller to avoid overwriting the Auto CMD12 response with the CMD_wo_DAT and vice versa.

When the Host Controller modifies part of the *Response* register, as shown in the Table 2-12, it shall preserve the unmodified bits.

2.2.8 Buffer Data Port Register (Offset 020h)

32-bit data port register to access internal buffer.

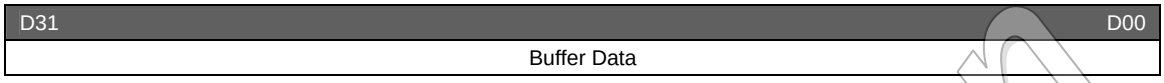


Figure 2-8 : Buffer Data Port Register

Buffer can be accessed through 32-bit *Data Port* register. Refer to 1.7

Location	Attrib	
31-00	RW	**Buffer Data** The Host Controller buffer can be accessed through this 32-bit *Data Port* register. Refer to 1.7

Table 2-13 : Buffer Data Port Register

2.2.9 Present State Register (Offset 024h)

The Host Driver can get status of the Host Controller from this 32-bit read only register.

D31	D25	D24	D23	D20	D19	D18	D17	D16
Rsvd		CMD Line Signal Level	DAT[3:0] Line Signal Level		Write Protect Switch Pin Level	Card Detect Pin Level	Card State Stable	Card Inserted

D15	D12	D11	D10	D09	D08	D07	D03	D02	D01	D00
Rsvd		Buffer Read Enable	Buffer Write Enable	Read Transfer Active	Write Transfer Active	Rsvd		DAT Line Active	Command Inhibit (DAT)	Command Inhibit (CMD)

Figure 2-9 : Present State Register

Location	Attrib	
31-25	Rsvd	**Reserved**
24	RO	**CMD Line Signal Level** This status is used to check the ***CMD*** line level to recover from errors, and for debugging.
23-20	RO	**DAT[3:0] Line Signal Level** This status is used to check the ***DAT*** line level to recover from errors, and for debugging. This is especially useful in detecting the busy signal level from ***DAT***[0]. D23 DAT[3] D22 DAT[2] D21 DAT[1] D20 DAT[0]
19	RO	**Write Protect Switch Pin Level** The Write Protect Switch is supported for memory and combo cards. This bit reflects the ***SDWP#*** pin. 1 Write enabled (***SDWP#***=1) 0 Write protected (***SDWP#***=0)
18	RO	**Card Detect Pin Level** This bit reflects the inverse value of the ***SDCD#*** pin. Debouncing is not performed on this bit. This bit may be valid when **Card State Stable** is set to 1, but it is not guaranteed because of propagation delay. Use of this bit is limited to testing since it must be debounced by software. 1 Card present (***SDCD#***=0) 0 No card present (***SDCD#***=1)

17	RO	**Card State Stable** This bit is used for testing. If it is 0, the **Card Detect Pin Level** is not stable. If this bit is set to 1, it means the **Card Detect Pin Level** is stable. No Card state can be detected by this bit is set to 1 and **Card Inserted** is set to 0. The **Software Reset For All** in the *Software Reset* register shall not affect this bit. 1 – No Card or Inserted 0 – Reset or Debouncing
16	RO	**Card Inserted** This bit indicates whether a card has been inserted. The Host Controller shall debounce this signal so that the Host Driver will not need to wait for it to stabilize. Changing from 0 to 1 generates a **Card Insertion** interrupt in the *Normal Interrupt Status* register and changing from 1 to 0 generates a **Card Removal** interrupt in the *Normal Interrupt Status* register. The **Software Reset For All** in the *Software Reset* register shall not affect this bit. If a card is removed while its power is on and its clock is oscillating, the Host Controller shall clear **SD Bus Power** in the *Power Control* register (refer to 2.2.11) and **SD Clock Enable** in the *Clock Control* register (refer to 2.2.14). In addition, the Host Driver should clear the Host Controller by the **Software Reset For All** in *Software Reset* register. The card detect is active regardless of the **SD Bus Power**. 1 – Card Inserted 0 – Reset or Debouncing or No Card

Table 2-14 : Present State Register (Part 1)

Figure 2-10 shows the state definitions of hardware that handles "Debouncing".

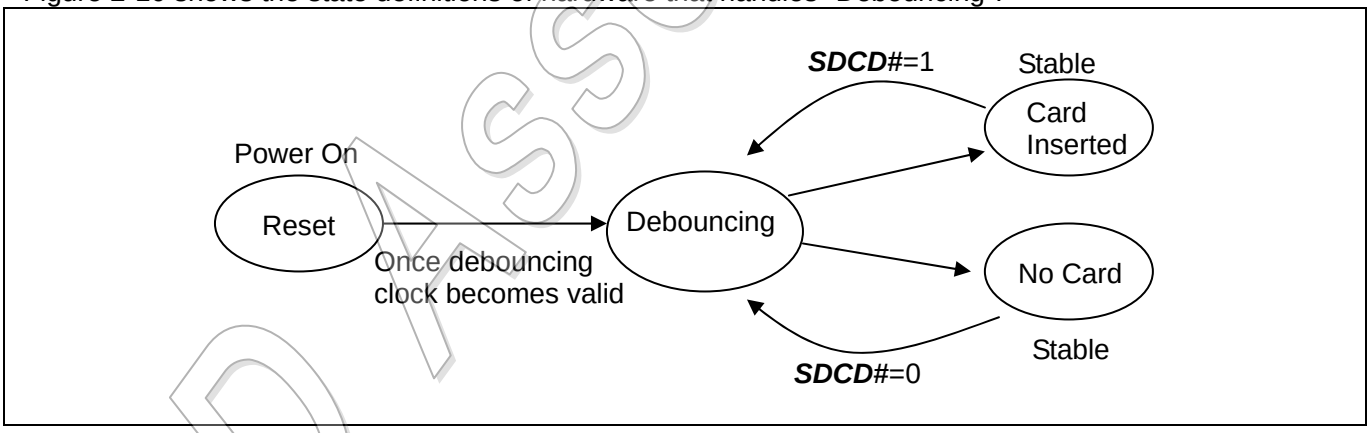


Figure 2-10 : Card Detect State

Implementation Note: The Host Controller starts in "Reset" state at power on and changes to the "Debouncing" state once the debouncing clock is valid. In the "Debouncing" state, if the Host Controller detects that the signal (***SDCD#***) is stable during the debounce period, the state shall change to "Card Inserted" or "No Card". If the card is removed while in the "Card Inserted" state, it will immediately change to the "Debouncing" state. Since the card detect signal is then not stable, the Host Controller will change to the "Debouncing" state.

Location	Attrib	
15-12	Rsvd	**Reserved**

<table>
<tr><td>11</td><td>ROC</td><td>Buffer Read Enable
This status is used for non-DMA read transfers.
The Host Controller may implement multiple buffers to transfer data efficiently. This read only flag indicates that valid data exists in the host side buffer status. If this bit is 1, readable data exists in the buffer. A change of this bit from 1 to 0 occurs when all the block data is read from the buffer. A change of this bit from 0 to 1 occurs when block data is ready in the buffer and generates the Buffer Read Ready interrupt.
<table><tr><td>1</td><td>Read enable</td></tr><tr><td>0</td><td>Read disable</td></tr></table></td></tr>
<tr><td>10</td><td>ROC</td><td>Buffer Write Enable
This status is used for non-DMA write transfers.
The Host Controller can implement multiple buffers to transfer data efficiently. This read only flag indicates if space is available for write data. If this bit is 1, data can be written to the buffer. A change of this bit from 1 to 0 occurs when all the block data is written to the buffer. A change of this bit from 0 to 1 occurs when top of block data can be written to the buffer and generates the Buffer Write Ready interrupt.

<table><tr><td>1</td><td>Write enable</td></tr><tr><td>0</td><td>Write disable</td></tr></table></td></tr>
<tr><td>09</td><td>ROC</td><td>Read Transfer Active
This status is used for detecting completion of a read transfer. Refer to Section 3.10.3 for sequence details.
This bit is set to 1 for either of the following conditions:
(1) After the end bit of the read command.
(2) When writing a 1 to Continue Request in the Block Gap Control register to restart a read transfer.
This bit is cleared to 0 for either of the following conditions::
(1) When the last data block as specified by block length is transferred to the System.
(2) When all valid data blocks have been transferred to the System and no current block transfers are being sent as a result of the Stop At Block Gap Request being set to 1.
A Transfer Complete interrupt is generated when this bit changes to 0.

<table><tr><td>1</td><td>Transferring data</td></tr><tr><td>0</td><td>No valid data</td></tr></table></td></tr>
</table>

<table>
<tr><td>08</td><td>ROC</td><td>Write Transfer Active
This status indicates a write transfer is active. If this bit is 0, it means no valid write data exists in the Host Controller. Refer to Section 3.10.4 for more details on the sequence of events.

This bit is set in either of the following cases:
(1) After the end bit of the write command.
(2) When writing a 1 to Continue Request in the Block Gap Control register to restart a write transfer.
This bit is cleared in either of the following cases:
(1) After getting the CRC status of the last data block as specified by the transfer count (Single and Multiple)
(2) After getting the CRC status of any block where data transmission is about to be stopped by a Stop At Block Gap Request.

During a write transaction, a Block Gap Event interrupt is generated when this bit is changed to 0, as result of the Stop At Block Gap Request being set. This status is useful for the Host Driver in determining when to issue commands during write busy.

<table><tr><td>1</td><td>Transferring data</td></tr><tr><td>0</td><td>No valid data</td></tr></table></td></tr>
<tr><td>07-03</td><td>Rsvd</td><td>Reserved</td></tr>
<tr><td>02</td><td>ROC</td><td>DAT Line Active
This bit indicates whether one of the DAT line on SD Bus is in use.

(a) In the case of read transactions
This status indicates if a read transfer is executing on the SD Bus. Changes in this value from 1 to 0 between data blocks generates a Block Gap Event interrupt in the Normal Interrupt Status register. Refer to Section 3.10.3 for details on timing.
This bit shall be set in either of the following cases:
(1) After the end bit of the read command.
(2) When writing a 1 to Continue Request in the Block Gap Control register to restart a read transfer.
This bit shall be cleared in either of the following cases:
(1) When the end bit of the last data block is sent from the SD Bus to the Host Controller.
(2) When beginning a wait read transfer at a stop at the block gap initiated by a Stop At Block Gap Request.
The Host Controller shall wait at the next block gap by driving Read Wait at the start of the interrupt cycle. If the Read Wait signal is already driven (data buffer cannot receive data), the Host Controller can wait for current block gap by continuing to drive the Read Wait signal. It is necessary to support Read Wait in order to use the suspend / resume function.

(b) In the case of write transactions
This status indicates that a write transfer is executing on the SD Bus. Changes in this value from 1 to 0 generate a Transfer Complete interrupt in the Normal Interrupt Status register. Refer to Section 3.10.4 for sequence details.

This bit shall be set in either of the following cases:
(1) After the end bit of the write command.
(2) When writing to 1 to Continue Request in the Block Gap Control register to continue a write transfer.
This bit shall be cleared in either of the following cases:</td></tr>
</table>

<table>
<tr><td></td><td></td><td>(1) When the SD card releases write busy of the last data block the Host Controller shall also detect if output is not busy. If SD card does not drive busy signal for 8 SD Clocks, the Host Controller shall consider the card drive "Not Busy".
(2) When the SD card releases write busy prior to waiting for write transfer as a result of a Stop At Block Gap Request.

<table><tr><td>1</td><td>DAT Line Active</td></tr><tr><td>0</td><td>DAT Line Inactive</td></tr></table></td></tr>
<tr><td>01</td><td>ROC</td><td>Command Inhibit (DAT)
This status bit is generated if either the DAT Line Active or the Read Transfer Active is set to 1. If this bit is 0, it indicates the Host Controller can issue the next SD Command. Commands with busy signal belong to Command Inhibit (DAT) (ex. R1b, R5b type). Changing from 1 to 0 generates a Transfer Complete interrupt in the Normal Interrupt Status register.
Note: The SD Host Driver can save registers in the range of 000-00Dh for a suspend transaction after this bit has changed from 1 to 0.

<table><tr><td>1</td><td>Cannot issue command which uses the DAT line</td></tr><tr><td>0</td><td>Can issue command which uses the DAT line</td></tr></table></td></tr>
<tr><td>00</td><td>ROC</td><td>Command Inhibit (CMD)
If this bit is 0, it indicates the CMD line is not in use and the Host Controller can issue a SD Command using the CMD line.
This bit is set immediately after the Command register (00Fh) is written. This bit is cleared when the command response is received. Even if the Command Inhibit (DAT) is set to 1, Commands using only the CMD line can be issued if this bit is 0. Changing from 1 to 0 generates a Command Complete interrupt in the Normal Interrupt Status register. If the Host Controller cannot issue the command because of a command conflict error (Refer to Command CRC Error in 2.2.18) or because of Command Not Issued By Auto CMD12 Error (Refer to 2.2.23), this bit shall remain 1 and the Command Complete is not set. Status issuing Auto CMD12 is not read from this bit.

<table><tr><td>1</td><td>Cannot issue command</td></tr><tr><td>0</td><td>Can issue command using only CMD line</td></tr></table></td></tr>
</table>

Table 2-15 : Present State Register (Part 2)

Implementation Note:
The Host Driver can issue CMD0, CMD12, CMD13 (for memory) and CMD52 (for SDIO) when the **DAT** lines are busy during data transfer. These commands can be issued when **Command Inhibit (CMD)** is set to zero. Other commands shall be issued when **Command Inhibit (DAT)** is set to zero. Possible changes to the SD Physical Specification may add other commands to this list in the future.

Implementation Note:
Some fields defined in the Present State Register change values asynchronous to the system clock. The System reads these statuses through the System Bus Interface and it may require data stable period during bus cycle. The Host Controller should sample and hold values during reads from this register according to the timing required by the System Bus Interface specification.

Figure 2-11 to Figure 2-13 shows the timing of setting and clearing the **Command Inhibit (DAT)** and the **Command Inhibit (CMD)**.

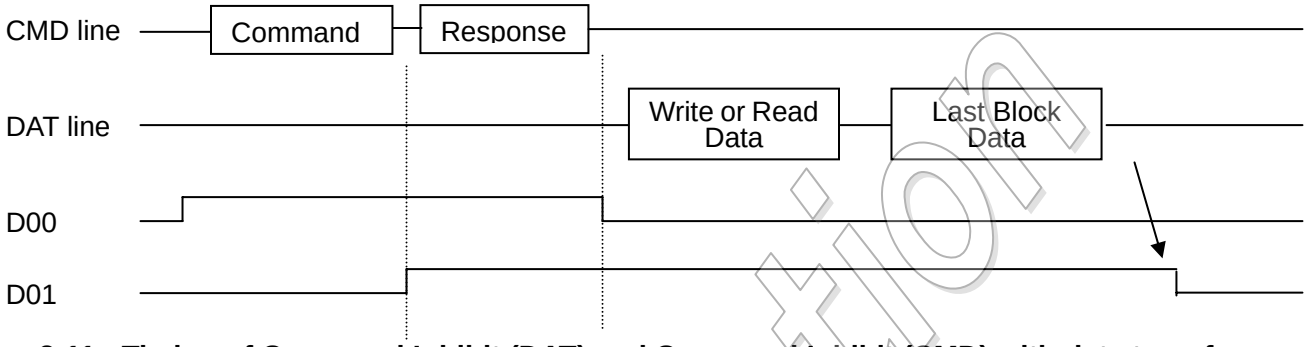


Figure 2-11 : Timing of Command Inhibit (DAT) and Command Inhibit (CMD) with data transfer

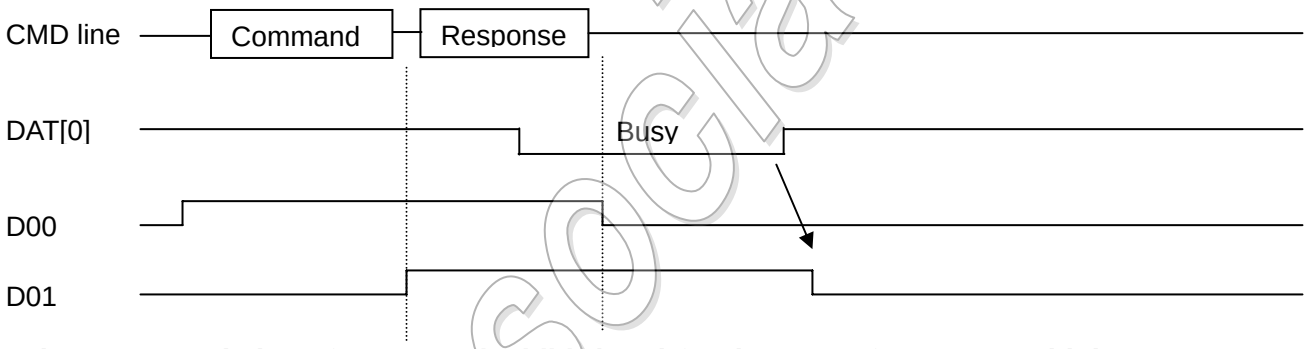


Figure 2-12 : Timing of Command Inhibit (DAT) for the case of response with busy

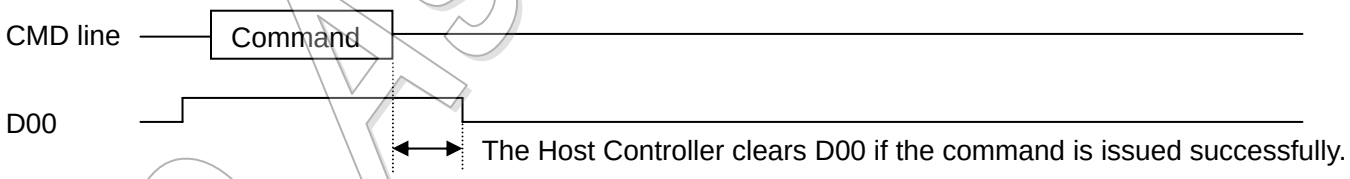


Figure 2-13 : Timing of Command Inhibit (CMD) for the case of no response command

2.2.10 Host Control Register (Offset 028h)

D07 D03	D02	D01	D00
Rsvd	High Speed Enable	Data Transfer Width	LED Control

Figure 2-14 : Host Control Register

Location	Attrib	
07-03	Rsvd	**Reserved**
02	RW	**High Speed Enable** This bit is optional. Before setting this bit, the Host Driver shall check the **High Speed Support** in the *Capabilities* register. If this bit is set to 0 (default), the Host Controller outputs ***CMD*** line and ***DAT*** lines at the falling edge of the SD Clock (up to 25MHz). If this bit is set to 1, the Host Controller outputs ***CMD*** line and ***DAT*** lines at the rising edge of the SD Clock (up to 50MHz). 1 High Speed mode 0 Normal Speed mode
01	RW	**Data Transfer Width** This bit selects the data width of the Host Controller. The Host Driver shall set it to match the data width of the SD card. 1 4-bit mode 0 1-bit mode
00	RW	**LED Control** This bit is used to caution the user not to remove the card while the SD card is being accessed. If the software is going to issue multiple SD commands, this bit can be set during all these transactions. It is not necessary to change for each transaction. 1 LED on 0 LED off

Table 2-16 : Host Control Register

2.2.11 Power Control Register (Offset 029h)

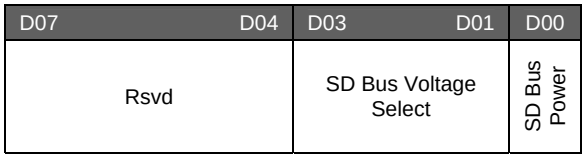


Figure 2-15 : Power Control Register

Location	Attrib	
07-04	Rsvd	**Reserved**
03-01	RW	**SD Bus Voltage Select** By setting these bits, the Host Driver selects the voltage level for the SD card. Before setting this register, the Host Driver shall check the **Voltage Support** bits in the *Capabilities* register. If an unsupported voltage is selected, the Host System shall not supply SD Bus voltage. 111b: 3.3V (Typ.) 110b: 3.0V (Typ.) 101b: 1.8V (Typ.) 100b – 000b: Reserved
00	RW	**SD Bus Power** Before setting this bit, the SD Host Driver shall set **SD Bus Voltage Select**. If the Host Controller detects the No Card state, this bit shall be cleared. 1: Power on 0: Power off

Table 2-17 : Power Control Register

Implementation Note:
Basically, the Host Driver has responsibility to supply SD Bus voltage by **SD Bus Power**, according to SD card OCR and supply voltage capabilities depend on the Host System.
If the Host Driver selects an unsupported voltage in the **SD Bus Voltage Select** field, the Host Controller may ignore writes to SD Bus Power and keep its value at zero.

Implementation Note:
The Host System shall not supply SD Bus power when **SD Bus Power** is set to 0 and can supply SD Bus power when **SD Bus Power** is set to 1 depending on the system conditions (ex. Left of the battery).

2.2.12 Block Gap Control Register (Offset 02Ah)

D07 D04	D03	D02	D01	D00
Rsvd	Interrupt At Block Gap	Read Wait Control	Continue Request	Stop At Block Gap Request

Figure 2-16 : Block Gap Control Register

Location	Attrib	
07-04	Rsvd	**Reserved**
03	RW	**Interrupt At Block Gap** This bit is valid only in 4-bit mode of the SDIO card and selects a sample point in the interrupt cycle. Setting to 1 enables interrupt detection at the block gap for a multiple block transfer. Setting to 0 disables interrupt detection during a multiple block transfer. If the SD card cannot signal an interrupt during a multiple block transfer, this bit should be set to 0. When the Host Driver detects an SD card insertion, it shall set this bit according to the CCCR of the SDIO card. 1 Enabled 0 Disabled
02	RW	**Read Wait Control** The read wait function is optional for SDIO cards. If the card supports read wait, set this bit to enable use of the read wait protocol to stop read data using the ***DAT***[2] line. Otherwise the Host Controller has to stop the SD Clock to hold read data, which restricts commands generation. When the Host Driver detects an SD card insertion, it shall set this bit according to the CCCR of the SDIO card. If the card does not support read wait, this bit shall never be set to 1 otherwise ***DAT*** line conflict may occur. If this bit is set to 0, Suspend/Resume cannot be supported. 1 Enable Read Wait Control 0 Disable Read Wait Control
01	RWAC	**Continue Request** This bit is used to restart a transaction which was stopped using the **Stop At Block Gap Request**. To cancel stop at the block gap, set **Stop At Block Gap Request** to 0 and set this bit 1 to restart the transfer. The Host Controller automatically clears this bit in either of the following cases: (1) In the case of a read transaction, the **DAT Line Active** changes from 0 to 1 as a read transaction restarts. (2) In the case of a write transaction, the **Write Transfer Active** changes from 0 to 1 as the write transaction restarts. Therefore it is not necessary for Host Driver to set this bit to 0. If **Stop At Block Gap Request** is set to 1, any write to this bit is ignored. 1 Restart 0 Not affect
00	RW	**Stop At Block Gap Request** This bit is used to stop executing a transaction at the next block gap for both DMA and non-DMA transfers. Until the **Transfer Complete** is set to 1, indicating a transfer completion the Host Driver shall leave this bit set to 1. Clearing both the

		Stop At Block Gap Request and **Continue Request** shall not cause the transaction to restart. Read Wait is used to stop the read transaction at the block gap. The Host Controller shall honor **Stop At Block Gap Request** for write transfers, but for read transfers it requires that the SD card support Read Wait. Therefore the Host Driver shall not set this bit during read transfers unless the SD card supports Read Wait and has set **Read Wait Control** to 1. In the case of write transfers in which the Host Driver writes data to the *Buffer Data Port* register, the Host Driver shall set this bit after all block data is written. If this bit is set to 1, the Host Driver shall not write data to *Buffer Data Port* register. This bit affects **Read Transfer Active**, **Write Transfer Active**, **DAT Line Active** and **Command Inhibit (DAT)** in the *Present State* register. Regarding detailed control of bits D01 and D00, refer to Section 3.7 and 3.10. 1 Stop 0 Transfer

Table 2-18 : Block Gap Control Register

There are three cases to restart the transfer after stop at the block gap. Which case is appropriate depends on whether the Host Controller issues a Suspend command or the SD card accepts the Suspend command.

(1) If the Host Driver does not issue a Suspend command, the **Continue Request** shall be used to restart the transfer.
(2) If the Host Driver issues a Suspend command and the SD card accepts it, a Resume command shall be used to restart the transfer.
(3) If the Host Driver issues a Suspend command and the SD card does not accept it, the **Continue Request** shall be used to restart the transfer.

Any time **Stop At Block Gap Request** stops the data transfer, the Host Driver shall wait for **Transfer Complete** (in the *Normal Interrupt Status* register) before attempting to restart the transfer. When restarting the data transfer by **Continue Request**, the Host Driver shall clear **Stop At Block Gap Request** before or simultaneously.

2.2.13 Wakeup Control Register (Offset 02Bh)

This register is mandatory for the Host Controller, but wakeup functionality depends on the Host Controller system hardware and software. The Host Driver shall maintain voltage on the SD Bus, by setting **SD Bus Power** to 1 in the *Power Control* register, when wakeup event via Card Interrupt is desired.

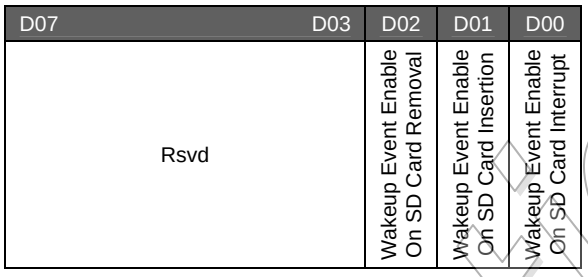


Figure 2-17 : Wakeup Control Register

Location	Attrib	
07-03	Rsvd	**Reserved**
02	RW	**Wakeup Event Enable On SD Card Removal** This bit enables wakeup event via **Card Removal** assertion in the *Normal Interrupt Status* register. **FN_WUS** (Wake Up Support) in CIS does not affect this bit. 1 Enable 0 Disable
01	RW	**Wakeup Event Enable On SD Card Insertion** This bit enables wakeup event via **Card Insertion** assertion in the *Normal Interrupt Status* register. **FN_WUS** (Wake Up Support) in CIS does not affect this bit. 1 Enable 0 Disable
00	RW	**Wakeup Event Enable On Card Interrupt** This bit enables wakeup event via **Card Interrupt** assertion in the *Normal Interrupt Status* register. This bit can be set to 1 if **FN_WUS** (Wake Up Support) in CIS is set to 1. 1 Enable 0 Disable

Table 2-19 : Wakeup Control Register

2.2.14 Clock Control Register (Offset 02Ch)

At the initialization of the Host Controller, the Host Driver shall set the **SDCLK Frequency Select** according to the *Capabilities* register.

D15 D08	D07 D03	D02	D01	D00
SDCLK Frequency Select	Rsvd	SD Clock Enable	Internal Clock Stable	Internal Clock Enable

Figure 2-18 : Clock Control Register

Location	Attrib	
15-08	RW	**SDCLK Frequency Select** This register is used to select the frequency of ***SDCLK*** pin. The frequency is not programmed directly; rather this register holds the divisor of the **Base Clock Frequency For SD Clock** in the *Capabilities* register. Only the following settings are allowed. 80h base clock divided by 256 40h base clock divided by 128 20h base clock divided by 64 10h base clock divided by 32 08h base clock divided by 16 04h base clock divided by 8 02h base clock divided by 4 01h base clock divided by 2 00h Base clock (10MHz-63MHz) Setting 00h specifies the highest frequency of the SD Clock. When setting multiple bits, the most significant bit is used as the divisor. But multiple bits should not be set. The two default divider values can be calculated by the frequency that is defined by the **Base Clock Frequency For SD Clock** in the *Capabilities* register. (1) 25MHz divider value (2) 400KHz divider value According to the SD Physical Specification Version 1.01 and the SDIO Card Specification Version 1.0, maximum SD Clock frequency is 25MHz, and shall never exceed this limit. The frequency of SDCLK is set by the following formula: Clock Frequency = (Base Clock) / divisor Thus, choose the smallest possible divisor which results in a clock frequency that is less than or equal to the target frequency. For example, if the **Base Clock Frequency For SD Clock** in the *Capabilities* register has the value 33MHz, and the target frequency is 25MHz, then choosing the divisor value of 01h will yield 16.5MHz, which is the nearest frequency less than or equal to the target. Similarly, to approach a clock value of 400KHz, the divisor value of 40h yields the optimal clock value of 258KHz.
07-03	Rsvd	**Reserved**

<table>
<tr><td>02</td><td>RW</td><td>SD Clock Enable
The Host Controller shall stop SDCLK when writing this bit to 0. SDCLK Frequency Select can be changed when this bit is 0. Then, the Host Controller shall maintain the same clock frequency until SDCLK is stopped (Stop at SDCLK=0). If the Host Controller detects the No Card state, this bit shall be cleared.

<table><tr><td>1</td><td>Enable</td></tr><tr><td>0</td><td>Disable</td></tr></table></td></tr>
<tr><td>01</td><td>ROC</td><td>Internal Clock Stable
This bit is set to 1 when SD Clock is stable after writing to Internal Clock Enable in this register to 1. The SD Host Driver shall wait to set SD Clock Enable until this bit is set to 1.
Note: This is useful when using PLL for a clock oscillator that requires setup time.

<table><tr><td>1</td><td>Ready</td></tr><tr><td>0</td><td>Not Ready</td></tr></table></td></tr>
<tr><td>00</td><td>RW</td><td>Internal Clock Enable
This bit is set to 0 when the Host Driver is not using the Host Controller or the Host Controller awaits a wakeup interrupt. The Host Controller should stop its internal clock to go very low power state. Still, registers shall be able to be read and written. Clock starts to oscillate when this bit is set to 1. When clock oscillation is stable, the Host Controller shall set Internal Clock Stable in this register to 1. This bit shall not affect card detection.

<table><tr><td>1</td><td>Oscillate</td></tr><tr><td>0</td><td>Stop</td></tr></table></td></tr>
</table>

Table 2-20 : Clock Control Register

2.2.15 Timeout Control Register (Offset 02Eh)

At the initialization of the Host Controller, the Host Driver shall set the **Data Timeout Counter Value** according to the *Capabilities* register.

D07 D04	D03 D00
Rsvd	Data Timeout Counter Value

Figure 2-19 : Timeout Control Register

Location	Attrib	
07-04	Rsvd	**Reserved**
03-00	RW	**Data Timeout Counter Value** This value determines the interval by which DAT line timeouts are detected. Refer to the **Data Timeout Error** in the *Error Interrupt Status* register for information on factors that dictate timeout generation. Timeout clock frequency will be generated by dividing the base clock TMCLK value by this value. When setting this register, prevent inadvertent timeout events by clearing the **Data Timeout Error Status Enable** (in the *Error Interrupt Status Enable* register) 1111b: Reserved 1110b: TMCLK x 2^{27} ……..: …… 0001b: TMCLK x 2^{14} 0000b: TMCLK x 2^{13}

Table 2-21 : Timeout Control Register

2.2.16 Software Reset Register (Offset 02Fh)

A reset pulse is generated when writing 1 to each bit of this register. After completing the reset, the Host Controller shall clear each bit. Because it takes some time to complete software reset, the SD Host Driver shall confirm that these bits are 0.

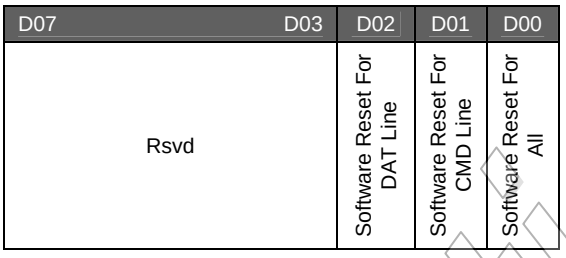


Figure 2-20 : Software Reset Register

Location	Attrib	
07-03	Rsvd	**Reserved**
02	RWAC	**Software Reset For DAT Line** Only part of data circuit is reset. DMA circuit is also reset. The following registers and bits are cleared by this bit: *Buffer Data Port* register Buffer is cleared and initialized. *Present State* register **Buffer Read Enable** **Buffer Write Enable** **Read Transfer Active** **Write Transfer Active** **DAT Line Active** **Command Inhibit (DAT)** *Block Gap Control* register **Continue Request** **Stop At Block Gap Request** *Normal Interrupt Status* register **Buffer Read Ready** **Buffer Write Ready** **DMA Interrupt** **Block Gap Event** **Transfer Complete** 1 Reset 0 Work

<table>
<tr><td>01</td><td>RWAC</td><td>Software Reset For CMD Line
Only part of command circuit is reset.

The following registers and bits are cleared by this bit:
Present State register
Command Inhibit (CMD)
Normal Interrupt Status register
Command Complete

<table><tr><td>1</td><td>Reset</td></tr><tr><td>0</td><td>Work</td></tr></table></td></tr>
<tr><td>00</td><td>RWAC</td><td>Software Reset For All
This reset affects the entire Host Controller except for the card detection circuit. Register bits of type ROC, RW, RW1C, RWAC are cleared to 0. During its initialization, the Host Driver shall set this bit to 1 to reset the Host Controller. The Host Controller shall reset this bit to 0 when capabilities registers are valid and the Host Driver can read them. Additional use of Software Reset For All may not affect the value of the Capabilities registers. If this bit is set to 1, the SD card shall reset itself and must be reinitialized by the Host Driver.

<table><tr><td>1</td><td>Reset</td></tr><tr><td>0</td><td>Work</td></tr></table></td></tr>
</table>

Table 2-22 : Software Reset Register

2.2.17 Normal Interrupt Status Register (Offset 030h)

The *Normal Interrupt Status Enable* affects reads of this register, but *Normal Interrupt Signal Enable* does not affect these reads. An interrupt is generated when the Normal Interrupt Signal Enable is enabled and at least one of the status bits is set to 1. For all bits except **Card Interrupt** and **Error Interrupt**, writing 1 to a bit clears it; writing to 0 keeps the bit unchanged. More than one status can be cleared with a single register write. The **Card Interrupt** is cleared when the card stops asserting the interrupt; that is, when the Card Driver services the interrupt condition.

D15	D14 D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
Error Interrupt	Rsvd	Card Interrupt	Card Removal	Card Insertion	Buffer Read Ready	Buffer Write Ready	DMA Interrupt	Block Gap Event	Transfer Complete	Command Complete

Figure 2-21 : Normal Interrupt Status Register

Location	Attrib	
15	ROC	**Error Interrupt** If any of the bits in the *Error Interrupt Status* register are set, then this bit is set. Therefore the Host Driver can efficiently test for an error by checking this bit first. This bit is read only. 1 Error 0 No Error
14-09	Rsvd	**Reserved**
08	ROC	**Card Interrupt** Writing this bit to 1 does not clear this bit. It is cleared by resetting the SD card interrupt factor. In 1-bit mode, the Host Controller shall detect the **Card Interrupt** without SD Clock to support wakeup. In 4-bit mode, the card interrupt signal is sampled during the interrupt cycle, so there are some sample delays between the interrupt signal from the SD card and the interrupt to the Host System. It is necessary to define how to handle this delay. When this status has been set and the Host Driver needs to start this interrupt service, **Card Interrupt Status Enable** in the *Normal Interrupt Status Enable* register shall be set to 0 in order to clear the card interrupt statuses latched in the Host Controller and to stop driving the interrupt signal to the Host System. After completion of the card interrupt service (It should reset interrupt factors in the SD card and the interrupt signal may not be asserted), set **Card Interrupt Status Enable** to 1 and start sampling the interrupt signal again. 1 Generate Card Interrupt 0 No Card Interrupt

<table>
<tr><td>07</td><td>RW1C</td><td>Card Removal
This status is set if the Card Inserted in the Present State register changes from 1 to 0.
When the Host Driver writes this bit to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card detect state may possibly be changed when the Host Driver clear this bit and interrupt event may not be generated.

<table><tr><td>1</td><td>Card removed</td></tr><tr><td>0</td><td>Card state stable or Debouncing</td></tr></table></td></tr>
<tr><td>06</td><td>RW1C</td><td>Card Insertion
This status is set if the Card Inserted in the Present State register changes from 0 to 1.
When the Host Driver writes this bit to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card detect state may possibly be changed when the Host Driver clear this bit and interrupt event may not be generated.

<table><tr><td>1</td><td>Card inserted</td></tr><tr><td>0</td><td>Card state stable or Debouncing</td></tr></table></td></tr>
<tr><td>05</td><td>RW1C</td><td>Buffer Read Ready
This status is set if the Buffer Read Enable changes from 0 to 1. Refer to the Buffer Read Enable in the Present State register.

<table><tr><td>1</td><td>Ready to read buffer</td></tr><tr><td>0</td><td>Not ready to read buffer</td></tr></table></td></tr>
<tr><td>04</td><td>RW1C</td><td>Buffer Write Ready
This status is set if the Buffer Write Enable changes from 0 to 1. Refer to the Buffer Write Enable in the Present State register.

<table><tr><td>1</td><td>Ready to write buffer</td></tr><tr><td>0</td><td>Not ready to write buffer</td></tr></table></td></tr>
<tr><td>03</td><td>RW1C</td><td>DMA Interrupt
This status is set if the Host Controller detects the Host DMA Buffer boundary during transfer. Refer to the Host DMA Buffer Boundary in the Block Size register.
Other DMA interrupt factors may be added in the future. This interrupt shall not be generated after the Transfer Complete.

<table><tr><td>1</td><td>DMA Interrupt is generated</td></tr><tr><td>0</td><td>No DMA Interrupt</td></tr></table></td></tr>
</table>

<table>
<tr><td>02</td><td>RW1C</td><td>Block Gap Event
If the Stop At Block Gap Request in the Block Gap Control register is set, this bit is set when both a read / write transaction is stopped at a block gap. If Stop At Block Gap Request is not set to 1, this bit is not set to 1.
(1) In the case of a Read Transaction
This bit is set at the falling edge of the DAT Line Active Status (When the transaction is stopped at SD Bus timing. The Read Wait must be supported in order to use this function. Refer to Section 3.10.3 about the detail timing.
(2) Case of Write Transaction
This bit is set at the falling edge of Write Transfer Active Status (After getting CRC status at SD Bus timing). Refer to Section 3.10.4 for more details on the sequence of events.

<table><tr><td>1</td><td>Transaction stopped at block gap</td></tr><tr><td>0</td><td>No Block Gap Event</td></tr></table></td></tr>
<tr><td>01</td><td>RW1C</td><td>Transfer Complete
This bit is set when a read / write transfer is completed.
(1) In the case of a Read Transaction
This bit is set at the falling edge of Read Transfer Active Status. There are two cases in which this interrupt is generated. The first is when a data transfer is completed as specified by data length (After the last data has been read to the Host System). The second is when data has stopped at the block gap and completed the data transfer by setting the Stop At Block Gap Request in the Block Gap Control register (After valid data has been read to the Host System). Refer to Section 3.10.3 for more details on the sequence of events.
(2) In the case of a Write Transaction
This bit is set at the falling edge of the DAT Line Active Status. There are two cases in which this interrupt is generated. The first is when the last data is written to the SD card as specified by data length and the busy signal released. The second is when data transfers are stopped at the block gap by setting Stop At Block Gap Request in the Block Gap Control register and data transfers completed. (After valid data is written to the SD card and the busy signal released). Refer to Section 3.10.4 for more details on the sequence of events.

The table below shows that Transfer Complete has higher priority than Data Timeout Error. If both bits are set to 1, the data transfer can be considered complete.

Relation between Transfer Complete and Data Timeout Error
<table><tr><th>Transfer Complete</th><th>Data Timeout Error</th><th>Meaning of the status</th></tr><tr><td>0</td><td>0</td><td>Interrupted by another factor</td></tr><tr><td>0</td><td>1</td><td>Timeout occur during transfer</td></tr><tr><td>1</td><td>Don't Care</td><td>Data transfer complete</td></tr></table>
<table><tr><td>1</td><td>Data transfer complete</td></tr><tr><td>0</td><td>No transfer complete</td></tr></table></td></tr>
</table>

<table>
<tr>
<td>00</td>
<td>RW1C</td>
<td>
Command Complete
This bit is set when get the end bit of the command response. (Except Auto CMD12)
Refer to Command Inhibit (CMD) in the Present State register.

The table below shows that Command Timeout Error has higher priority than Command Complete. If both bits are set to 1, it can be considered that the response was not received correctly.
<table>
<tr><th>Command Complete</th><th>Command Timeout Error</th><th>Meaning of the status</th></tr>
<tr><td>0</td><td>0</td><td>Interrupted by another factor</td></tr>
<tr><td>Don't Care</td><td>1</td><td>Response not received within 64 SDCLK cycles.</td></tr>
<tr><td>1</td><td>0</td><td>Response received</td></tr>
</table>
<table>
<tr><td>1</td><td>Command complete</td></tr>
<tr><td>0</td><td>No command complete</td></tr>
</table>
</td>
</tr>
</table>

Table 2-23 : Normal Interrupt Status Register

2.2.18 Error Interrupt Status Register (Offset 032h)

Signals defined in this register can be enabled by the *Error Interrupt Status Enable* register, but not by the *Error Interrupt Signal Enable* register. The interrupt is generated when the *Error Interrupt Signal Enable* is enabled and at least one of the statuses is set to 1. Writing to 1 clears the bit and writing to 0 keeps the bit unchanged. More than one status can be cleared at the one register write.

D15 D12	D11 D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
Vendor Specific Error Status	Rsvd	Auto CMD12 Error	Current limit Error	Data End Bit Error	Data CRC Error	Data Timeout Error	Command Index Error	Command End Bit Error	Command CRC Error	Command Timeout Error

Figure 2-22 : Error Interrupt Status Register

Location	Attrib	
15-12	RW1C	**Vendor Specific Error Status** Additional status bits can be defined in this register by the vendor.
11-09	Rsvd	**Reserved**
08	RW1C	**Auto CMD12 Error** Occurs when detecting that one of the bits in *Auto CMD12 Error Status* register has changed from 0 to 1. This bit is set to 1,not only when the errors in Auto CMD12 occur but also when Auto CMD12 is not executed due to the previous command error. 1 Error 0 No Error
07	RW1C	**Current Limit Error** By setting the **SD Bus Power** bit in the *Power Control* register, the Host Controller is requested to supply power for the SD Bus. If the Host Controller supports the Current Limit function, it can be protected from an illegal card by stopping power supply to the card in which case this bit indicates a failure status. Reading 1 means the Host Controller is not supplying power to SD card due to some failure. Reading 0 means that the Host Controller is supplying power and no error has occurred. The Host Controller may require some sampling time to detect the current limit. If the Host Controller does not support this function, this bit shall always be set to 0. 1 Power fail 0 No Error
06	RW1C	**Data End Bit Error** Occurs either when detecting 0 at the end bit position of read data which uses the ***DAT*** line or at the end bit position of the CRC Status. 1 Error 0 No Error
05	RW1C	**Data CRC Error** Occurs when detecting CRC error when transferring read data which uses the ***DAT*** line or when detecting the Write CRC status having a value of other than "010". 1 Error 0 No Error

04	RW1C	**Data Timeout Error** Occurs when detecting one of following timeout conditions. (1) Busy timeout for R1b,R5b type (2) Busy timeout after Write CRC status (3) Write CRC Status timeout (4) Read Data timeout. 1 Time out 0 No Error
03	RW1C	**Command Index Error** Occurs if a Command Index error occurs in the command response. 1 Error 0 No Error
02	RW1C	**Command End Bit Error** Occurs when detecting that the end bit of a command response is 0. 1 End Bit Error Generated 0 No Error
01	RW1C	**Command CRC Error** Command CRC Error is generated in two cases. (1) If a response is returned and the **Command Timeout Error** is set to 0 (indicating no timeout), this bit is set to 1 when detecting a CRC error in the command response. (2) The Host Controller detects a ***CMD*** line conflict by monitoring the ***CMD*** line when a command is issued. If the Host Controller drives the ***CMD*** line to 1 level, but detects 0 level on the ***CMD*** line at the next SDCLK edge, then the Host Controller shall abort the command (Stop driving ***CMD*** line) and set this bit to 1. The Command Timeout Error shall also be set to 1 to distinguish ***CMD*** line conflict (Refer to Table 2-25). 1 CRC Error Generated. 0 No Error
00	RW1C	**Command Timeout Error** Occurs only if no response is returned within 64 ***SDCLK*** cycles from the end bit of the command. If the Host Controller detects a ***CMD*** line conflict, in which case **Command CRC Error shall** also be set as shown in Table 2-25, this bit **shall** be set without waiting for 64 ***SDCLK*** cycles because the command will be aborted by the Host Controller. 1 Time out 0 No Error

Table 2-24 : Error Interrupt Status Register

The relation between **Command CRC Error** and **Command Timeout Error** is shown in Table 2-25.

Command CRC Error	Command Timeout Error	Kinds of error
0	0	No Error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	***CMD*** line conflict

Table 2-25 : The relation between Command CRC Error and Command Timeout Error

2.2.19 Normal Interrupt Status Enable Register (Offset 034h)

Setting to 1 enables Interrupt Status.

D15	D14 D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
Fixed to 0	Rsvd	Card Interrupt Status Enable	Card Removal Status Enable	Card Insertion Status Enable	Buffer Read Ready Status Enable	Buffer Write Ready Status Enable	DMA Interrupt Status Enable	Block Gap Event Status Enable	Transfer Complete Status Enable	Command Complete Status Enable

Figure 2-23 : Normal Interrupt Status Enable Register

Location	Attrib	
15	RO	**Fixed to 0** The Host Driver shall control error interrupts using the *Error Interrupt Status Enable* register.
14-09	Rsvd	**Reserved**
08	RW	**Card Interrupt Status Enable** If this bit is set to 0, the Host Controller shall clear interrupt request to the System. The **Card Interrupt** detection is stopped when this bit is cleared and restarted when this bit is set to 1. The Host Driver should clear the **Card Interrupt Status Enable** before servicing the **Card Interrupt** and should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts. 1 Enabled 0 Masked
07	RW	**Card Removal Status Enable** 1 Enabled 0 Masked
06	RW	**Card Insertion Status Enable** 1 Enabled 0 Masked
05	RW	**Buffer Read Ready Status Enable** 1 Enabled 0 Masked
04	RW	**Buffer Write Ready Status Enable** 1 Enabled 0 Masked
03	RW	**DMA Interrupt Status Enable** 1 Enabled 0 Masked
02	RW	**Block Gap Event Status Enable** 1 Enabled 0 Masked

01	RW	**Transfer Complete Status Enable** 1 Enabled 0 Masked
00	RW	**Command Complete Status Enable** 1 Enabled 0 Masked

Table 2-26 : Normal Interrupt Status Enable Register

Implementation Note:
The Host Controller may sample the card interrupt signal during interrupt period and may hold its value in the flip-flop. If the **Card Interrupt Status Enable** is set to 0, the Host Controller shall clear all internal signals regarding **Card Interrupt**.

2.2.20 Error Interrupt Status Enable Register (Offset 036h)

Setting to 1 enables Interrupt Status.

D15 D12	D11 D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
Vendor Specific Error Status Enable	Rsvd	Auto CMD12 Error Status Enable	Current Limit Error Status Enable	Data End Bit Error Status Enable	Data CRC Error Status Enable	Data Timeout Error Status Enable	Command Index Error Status Enable	Command End Bit Error Status Enable	Command CRC Error Status Enable	Command Timeout Error Status Enable

Figure 2-24 : Error Interrupt Status Enable Register

Location	Attrib	
15-12	RW	**Vendor Specific Error Status Enable** 1 Enabled 0 Masked
11-09	Rsvd	**Reserved**
08	RW	**Auto CMD12 Error Status Enable** 1 Enabled 0 Masked
07	RW	**Current Limit Error Status Enable** 1 Enabled 0 Masked
06	RW	**Data End Bit Error Status Enable** 1 Enabled 0 Masked
05	RW	**Data CRC Error Status Enable** 1 Enabled 0 Masked
04	RW	**Data Timeout Error Status Enable** 1 Enabled 0 Masked
03	RW	**Command Index Error Status Enable** 1 Enabled 0 Masked
02	RW	**Command End Bit Error Status Enable** 1 Enabled 0 Masked
01	RW	**Command CRC Error Status Enable** 1 Enabled 0 Masked

00	RW	**Command Timeout Error Status Enable** 1 Enabled 0 Masked

Table 2-27 : Error Interrupt Status Enable Register

Implementation Note: To detect CMD line conflict, the Host Driver must set both **Command Timeout Error Status Enable** and **Command CRC Error Status Enable** to 1.

2.2.21 Normal Interrupt Signal Enable Register (Offset 038h)

This register is used to select which interrupt status is indicated to the Host System as the interrupt. These status bits all share the same1 bit interrupt line. Setting any of these bits to 1 enables interrupt generation.

D15	D14 D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
Fixed to 0	Rsvd	Card Interrupt Signal Enable	Card Removal Signal Enable	Card Insertion Signal Enable	Buffer Read Ready Signal Enable	Buffer Write Ready Signal Enable	DMA Interrupt Signal Enable	Block Gap Event Signal Enable	Transfer Complete Signal Enable	Command Complete Signal Enable

Figure 2-25 : Normal Interrupt Signal Enable Register

Location	Attrib	
15	RO	**Fixed to 0** The Host Driver shall control error interrupts using the *Error Interrupt Signal Enable* register.
14-09	Rsvd	**Reserved**
08	RW	**Card Interrupt Signal Enable** 1 Enabled 0 Masked
07	RW	**Card Removal Signal Enable** 1 Enabled 0 Masked
06	RW	**Card Insertion Signal Enable** 1 Enabled 0 Masked
05	RW	**Buffer Read Ready Signal Enable** 1 Enabled 0 Masked
04	RW	**Buffer Write Ready Signal Enable** 1 Enabled 0 Masked
03	RW	**DMA Interrupt Signal Enable** 1 Enabled 0 Masked
02	RW	**Block Gap Event Signal Enable** 1 Enabled 0 Masked
01	RW	**Transfer Complete Signal Enable** 1 Enabled 0 Masked

00	RW	**Command Complete Signal Enable** 1 Enabled 0 Masked

Table 2-28 : Normal Interrupt Signal Enable Register

2.2.22 Error Interrupt Signal Enable Register (Offset 03Ah)

This register is used to select which interrupt status is notified to the Host System as the interrupt. These status bits all share the same 1 bit interrupt line. Setting any of these bits to 1 enables interrupt generation.

D15 D12	D11 D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
Vendor Specific Error Signal	Rsvd	Auto CMD12 Error Signal Enable	Current Limit Error Signal Enable	Data End Bit Error Signal Enable	Data CRC Error Signal Enable	Data Timeout Error Signal Enable	Command Index Error Signal Enable	Command End Bit Error Signal Enable	Command CRC Error Signal Enable	Command Timeout Error Signal Enable

Figure 2-26 : Error Interrupt Signal Enable Register

Location	Attrib	
15-12	RW	**Vendor Specific Error Signal Enable** 1 Enabled 0 Masked
11-09	Rsvd	**Reserved**
08	RW	**Auto CMD12 Error Signal Enable** 1 Enabled 0 Masked
07	RW	**Current Limit Error Signal Enable** 1 Enabled 0 Masked
06	RW	**Data End Bit Error Signal Enable** 1 Enabled 0 Masked
05	RW	**Data CRC Error Signal Enable** 1 Enabled 0 Masked
04	RW	**Data Timeout Error Signal Enable** 1 Enabled 0 Masked
03	RW	**Command Index Error Signal Enable** 1 Enabled 0 Masked
02	RW	**Command End Bit Error Signal Enable** 1 Enabled 0 Masked
01	RW	**Command CRC Error Signal Enable** 1 Enabled 0 Masked

00	RW	**Command Timeout Error Signal Enable** 1 Enabled 0 Masked

Table 2-29 : Error Interrupt Signal Enable Register

2.2.23 Auto CMD12 Error Status Register (Offset 03Ch)

When *Auto CMD12 Error Status* is set, the Host Driver shall check this register to identify what kind of error Auto CMD12 indicated. This register is valid only when the **Auto CMD12 Error** is set.

D15 D08	D07	D06 D05	D04	D03	D02	D01	D00
Rsvd	Command Not Issued by Auto CMD12 Error	Rsvd	Auto CMD12 Index Error	Auto CMD12 End Bit Error	Auto CMD12 CRC Error	Auto CMD12 Timeout Error	Auto CMD12 not executed

Figure 2-27 : Auto CMD12 Error Status Register

Location	Attrib	
15-08	Rsvd	**Reserved**
07	ROC	**Command Not Issued By Auto CMD12 Error** Setting this bit to 1 means CMD_wo_DAT is not executed due to an Auto CMD12 Error (D04-D01) in this register. 1 Not Issued 0 No error
06-05	Rsvd	**Reserved**
04	ROC	**Auto CMD12 Index Error** Occurs if the Command Index error occurs in response to a command. 1 Error 0 No error
03	ROC	**Auto CMD12 End Bit Error** Occurs when detecting that the end bit of command response is 0. 1 End Bit Error Generated 0 No error
02	ROC	**Auto CMD12 CRC Error** Occurs when detecting a CRC error in the command response. 1 CRC Error Generated 0 No error
01	ROC	**Auto CMD12 Timeout Error** Occurs if no response is returned within 64 ***SDCLK*** cycles from the end bit of command. If this bit is set to1, the other error status bits (D04-D02) are meaningless. 1 Time out 0 No error

00	ROC	**Auto CMD12 Not Executed** If memory multiple block data transfer is not started due to command error, this bit is not set because it is not necessary to issue Auto CMD12. Setting this bit to 1 means the Host Controller cannot issue Auto CMD12 to stop memory multiple block data transfer due to some error. If this bit is set to 1, other error status bits (D04-D01) are meaningless. 1 Not executed 0 Executed

Table 2-30 : Auto CMD12 Error Status Register

The relation between **Auto CMD12 CRC Error** and **Auto CMD12 Timeout Error** is shown in Table 2-31 :.

Auto CMD12 CRC Error	Auto CMD12 Timeout Error	Kinds of error
0	0	No Error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD line conflict

Table 2-31 : The relation between Command CRC Error and Command Timeout Error for Auto CMD12

The timing of changing *Auto CMD12 Error Status* can be classified in three scenarios:

(1) When the Host Controller is going to issue Auto CMD12
Set D00 to 1 if Auto CMD12 cannot be issued due to an error in the previous command.
Set D00 to 0 if Auto CMD12 is issued.

(2) At the end bit of an Auto CMD12 response
Check received responses by checking the error bits D01, D02, D03 and D04.
Set to 1 if error is detected.
Set to 0 if error is not detected.

(3) Before reading the Auto CMD12 Error Status bit D07
Set D07 to 1 if there is a command cannot be issued
Set D07 to 0 if there is no command to issue

Timing of generating the **Auto CMD12 Error** and writing to the *Command* register are asynchronous. Then D07 shall be sampled when driver never writing to the *Command* register. So just before reading the *Auto CMD12 Error Status* register is good timing to set the D07 status bit.

An Auto CMD12 Error Interrupt is generated when one of the error bits D00 to D04 is set to 1. The **Command Not Issued By Auto CMD12 Error** does not generate an interrupt.

2.2.24 Capabilities Register (Offset 040h)

This register provides the Host Driver with information specific to the Host Controller implementation. The Host Controller may implement these values as fixed or loaded from flash memory during power on initialization. Refer to **Software Reset For All** in the *Software Reset* register for loading from flash memory and completion timing control.

D63														D32
Rsvd														
D31 D27	D26	D25	D24	D23	D22	D21	D20 D18	D17-16	D15-14	D13 D08	D07	D06	D05 D00	
Rsvd	Voltage Support 1.8V	Voltage Support 3.0V	Voltage Support 3.3V	Suspend/Resume Support	DMA Support	High Speed Support	Rsvd	Max Block Length	Rsvd	Base Clock Frequency For SD Clock	Timeout Clock Unit	Rsvd	Timeout Clock Frequency	

Figure 2-28 : Capabilities Register

Location	Attrib	
63-32	Rsvd	**Reserved**
31-27	Rsvd	**Reserved** Reserved for voltage support
26	HwInit	**Voltage Support 1.8V** 1 – 1.8V Supported 0 – 1.8V Not Supported
25	HwInit	**Voltage Support 3.0V** 1 – 3.0V Supported 0 – 3.0V Not Supported
24	HwInit	**Voltage Support 3.3V** 1 – 3.3V Supported 0 – 3.3V Not Supported

Table 2-32 : Capabilities Register (Part 1)

Implementation Note: The Host System shall support at least one of these voltages above. The Host Driver sets the **SD Bus Voltage Select** in *Power Control* register according to these support bits. If multiple voltages are supported, select the usable lower voltage by comparing with the OCR value from the SD card.

Location	Attrib	

<table>
<tr><td>23</td><td>HwInit</td><td>Suspend/Resume Support
This bit indicates whether the Host Controller supports Suspend / Resume functionality. If this bit is 0, the Suspend and Resume mechanism (Refer to 1.6) are not supported and the Host Driver shall not issue either Suspend or Resume commands.

<table><tr><td>1</td><td>Supported</td></tr><tr><td>0</td><td>Not supported</td></tr></table></td></tr>
<tr><td>22</td><td>HwInit</td><td>DMA Support
This bit indicates whether the Host Controller is capable of using DMA to transfer data between system memory and the Host Controller directly.

<table><tr><td>1</td><td>DMA Supported</td></tr><tr><td>0</td><td>DMA not supported</td></tr></table></td></tr>
<tr><td>21</td><td>HwInit</td><td>High Speed Support
This bit indicates whether the Host Controller and the Host System support High Speed mode and they can supply SD Clock frequency from 25MHz to 50MHz.

<table><tr><td>1</td><td>High Speed Supported</td></tr><tr><td>0</td><td>High Speed not supported</td></tr></table></td></tr>
<tr><td>20-18</td><td>Rsvd</td><td>Reserved</td></tr>
<tr><td>17-16</td><td>HwInit</td><td>Max Block Length
This value indicates the maximum block size that the Host Driver can read and write to the buffer in the Host Controller. The buffer shall transfer this block size without wait cycles. Three sizes can be defined as indicated below.

<table><tr><td>00</td><td>512(byte)</td></tr><tr><td>01</td><td>1024</td></tr><tr><td>10</td><td>2048</td></tr><tr><td>11</td><td>Reserved</td></tr></table></td></tr>
<tr><td>15-14</td><td>Rsvd</td><td>Reserved</td></tr>
<tr><td>13-08</td><td>HwInit</td><td>Base Clock Frequency For SD Clock
This value indicates the base (maximum) clock frequency for the SD Clock. Unit values are 1MHz. If the real frequency is 16.5MHz, the lager value shall be set 01 0001b (17MHz) because the Host Driver use this value to calculate the clock divider value (Refer to the SDCLK Frequency Select in the Clock Control register.) and it shall not exceed upper limit of the SD Clock frequency. The supported clock range is 10MHz to 63MHz. If these bits are all 0, the Host System has to get information via another method.

<table><tr><td>Not 0</td><td>1MHz to 63MHz</td></tr><tr><td>00 0000b</td><td>Get information via another method</td></tr></table></td></tr>
<tr><td>07</td><td>HwInit</td><td>Timeout Clock Unit
This bit shows the unit of base clock frequency used to detect Data Timeout Error.

<table><tr><td>0</td><td>KHz</td></tr><tr><td>1</td><td>MHz</td></tr></table></td></tr>
<tr><td>06</td><td>Rsvd</td><td>Reserved</td></tr>
</table>

<table>
<tr>
<td>05-00</td>
<td>HwInit</td>
<td>Timeout Clock Frequency
This bit shows the base clock frequency used to detect Data Timeout Error.
The Timeout Clock Unit defines the unit of this fields value.
Timeout Clock Unit =0 [KHz] unit: 1KHz to 63KHz
Timeout Clock Unit =1 [MHz] unit: 1MHz to 63MHz

<table>
<tr><td>Not 0</td><td>1KHz to 63KHz or 1MHz to 63MHz</td></tr>
<tr><td>00 0000b</td><td>Get information via another method</td></tr>
</table>
</td>
</tr>
</table>

Table 2-33 : Capabilities Register (Part 2)

2.2.25 Maximum Current Capabilities Register (Offset 048h)

These registers indicate maximum current capability for each voltage. The value is meaningful if **Voltage Support** is set in the *Capabilities* register. If this information is supplied by the Host System via another method, all *Maximum Current Capabilities* register shall be 0.

D63 D32			
Rsvd			
D31 D24	D23 D16	D15 D08	D07 D00
Rsvd	Maximum Current for 1.8V	Maximum Current for 3.0V	Maximum Current for 3.3V

Figure 2-29 : Maximum Current Capabilities Register

Location	Attrib	
63-56	Rsvd	**Reserved**
55-48	Rsvd	**Reserved**
47-40	Rsvd	**Reserved**
39-32	Rsvd	**Reserved**
31-24	Rsvd	**Reserved**
23-16	HwInit	**Maximum Current for 1.8V**
15-08	HwInit	**Maximum Current for 3.0V**
07-00	HwInit	**Maximum Current for 3.3V**

Table 2-34 : Maximum Current Capabilities Register

This register measures current in 4mA steps. Each voltage level's current support is described using the Table 2-35.

Register Value	Current Value
0	Get information via another method
1	4mA
2	8mA
3	12mA
…….	…….
255	1020mA

Table 2-35 : Maximum Current Value Definition

2.2.26 Slot Interrupt Status Register (Offset 0FCh)

D15 D08	D07 D00
Rsvd	Interrupt Signal For Each Slot

Figure 2-30 : Slot Interrupt Status Register

Location	Attrib	
15-08	Rsvd	**Reserved**
07-00	ROC	**Interrupt Signal For Each Slot** These status bits indicate the logical OR of Interrupt Signal and Wakeup Signal for each slot. A maximum of 8 slots can be defined. If one interrupt signal is associated with multiple slots, the Host Driver can know which interrupt is generated by reading these status bits. By a power on reset or by setting **Software Reset For All**, the interrupt signal shall be de-asserted and this status shall read 00h. Bit 00 Slot 1 Bit 01 Slot 2 Bit 02 Slot 3 .... Bit 07 Slot 8

Table 2-36 : Slot Interrupt Status Register

2.2.27 Host Controller Version Register (Offset 0FEh)

D15 D08	D07 D00
Vendor Version Number	Specification Version Number

Figure 2-31 : Host Controller Version Register

Location	Attrib	
15-08	HwInit	**Vendor Version Number** This status is reserved for the vendor version number. The Host Driver should not use this status.
07-00	HwInit	**Specification Version Number** This status indicates the Host Controller Spec. Version. The upper and lower 4-bits indicate the version. 00 SD Host Specification Version 1.0 others Reserved

Table 2-37 : Host Controller Version

3. SEQUENCE

This section defines basic sequence flow chart divided into several sub sequences.

"Wait for interrupts" is used in the flow chart. This means the Host Driver waits until specified interrupts are asserted. If already asserted, then fall through that step in the flow chart. Timeout checking shall be always required to detect no interrupt generated but this is not described in the flow chart.

This specification uses the double box like Figure 3-1, (the step (1) in Figure 3-5 and the step (5) in Figure 3-21), It means that the other flows, which already are shown, shall be performed. Therefore, the interrupt may be included in the other flows.

Figure 3-1 : Double Box Notation

3.1 SD Card Detection

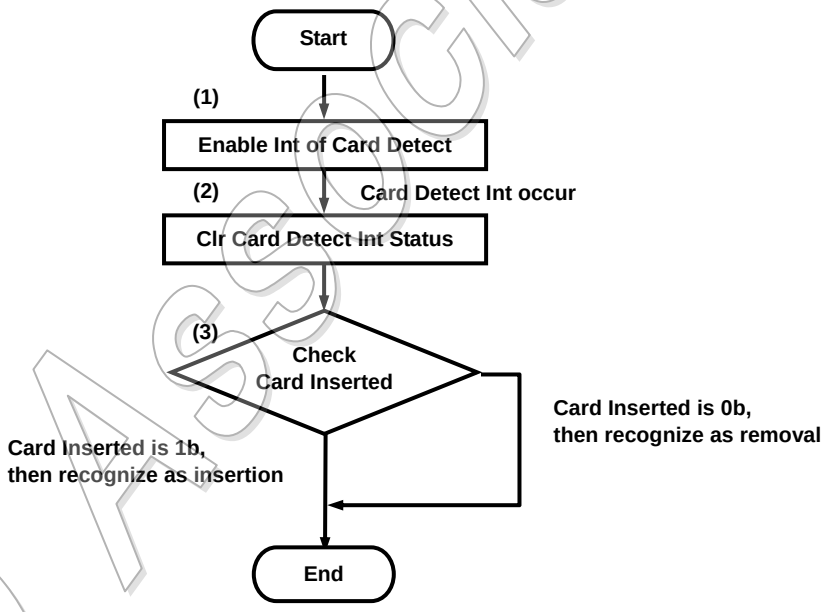


Figure 3-2: SD Card Detect Sequence

The flow chart for detecting a SD card is shown in Figure 3-2. Each step is executed as follows:

(1) To enable interrupt for card detection, write 1 to the following bits:
Card Insertion Status Enable in the *Normal Interrupt Status Enable* register
Card Insertion Signal Enable in the *Normal Interrupt Signal Enable* register
Card Removal Status Enable in the *Normal Interrupt Status Enable* register
Card Removal Signal Enable in the *Normal Interrupt Signal Enable* register

(2) When the Host Driver detects the card insertion or removal, clear its interrupt statuses. If **Card Insertion** interrupt is generated, write 1 to **Card Insertion** in the *Normal Interrupt Status* register. If **Card Removal** interrupt is generated, write 1 to **Card Removal** in the *Normal Interrupt Status* register.

(3) Check **Card Inserted** in the *Present State* register. In the case where **Card Inserted** is 1, the Host Driver can supply the power and the clock to the SD card. In the case where **Card Inserted** is 0, the other executing processes of the Host Driver shall be immediately closed.

3.2 SD Clock Control

3.2.1 SD Clock Supply Sequence

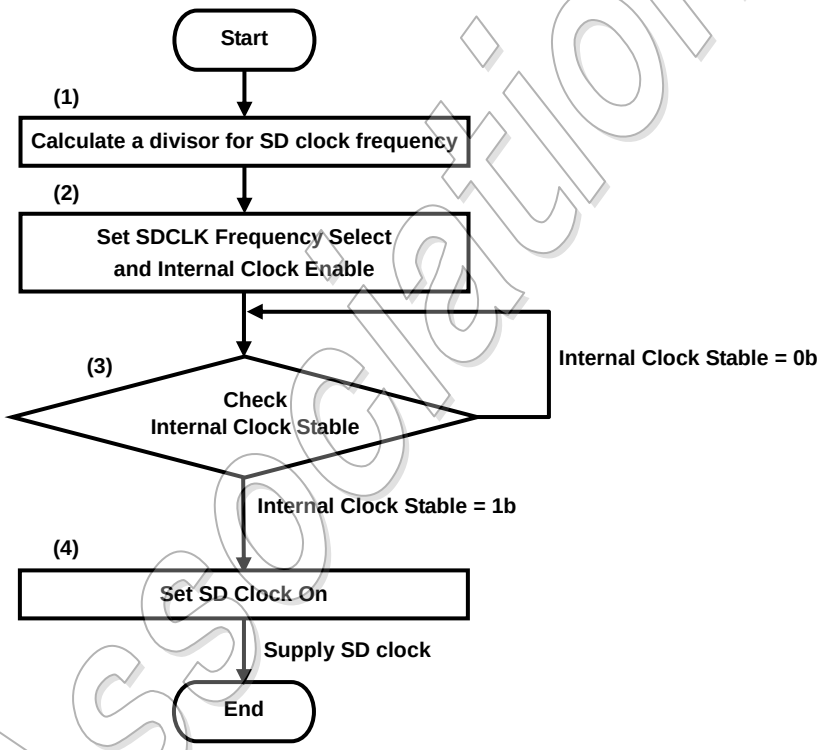


Figure 3-3: SD Clock Supply Sequence

The sequence for supplying SD Clock to a SD card is described in Figure 3-3. The clock shall be supplied to the card before either of the following actions is taken.

a) Issuing a SD command
b) Detect an interrupt from a SD card in 4-bit mode.

(1) Calculate a divisor to determine SD Clock frequency by reading **Base Clock Frequency For SD Clock** in the *Capabilities* register. If **Base Clock Frequency For SD Clock** is 00 0000b, the Host System shall provide this information to the Host Driver by another method.

(2) Set **Internal Clock Enable** and **SDCLK Frequency Select** in the *Clock Control* register in accordance with the calculated result of step (1).

(3) Check **Internal Clock Stable** in the *Clock Control* register. Repeat this step until Clock Stable is 1.

(4) Set **SD Clock Enable** in the *Clock Control* register to 1. Then, the Host Controller starts to supply the SD Clock.

3.2.2 SD Clock Stop Sequence

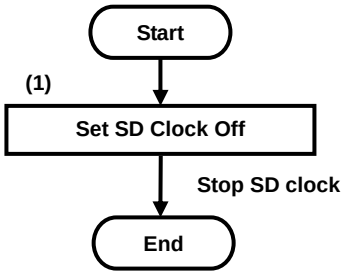


Figure 3-4: SD Clock Stop Sequence

The flow chart for stopping the SD Clock is shown in Figure 3-4. The Host Driver shall not stop the SD Clock when a SD transaction is occurring on the SD Bus -- namely, when either **Command Inhibit (DAT)** or **Command Inhibit (CMD)** in the *Present State* register is set to 1.

(1) Set **SD Clock Enable** in the *Clock Control* register to 0. Then, the Host Controller stops supplying the SD Clock.

3.2.3 SD Clock Frequency Change Sequence

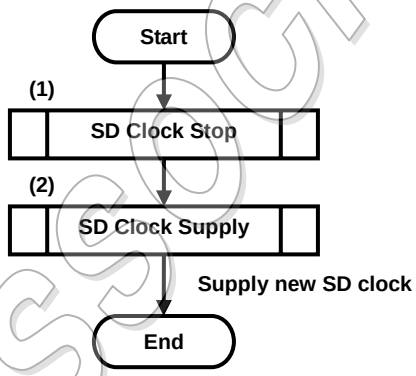


Figure 3-5: SD Clock Change Sequence

The sequence for changing SD Clock frequency is shown in Figure 3-5. When SD Clock is still off, step (1) is omitted. Please refer to Section 3.2.2 for details regarding step (1) and Section 3.2.1 for step (2).

3.3 SD Bus Power Control

The sequence for controlling the SD Bus Power is described in Figure 3-6.

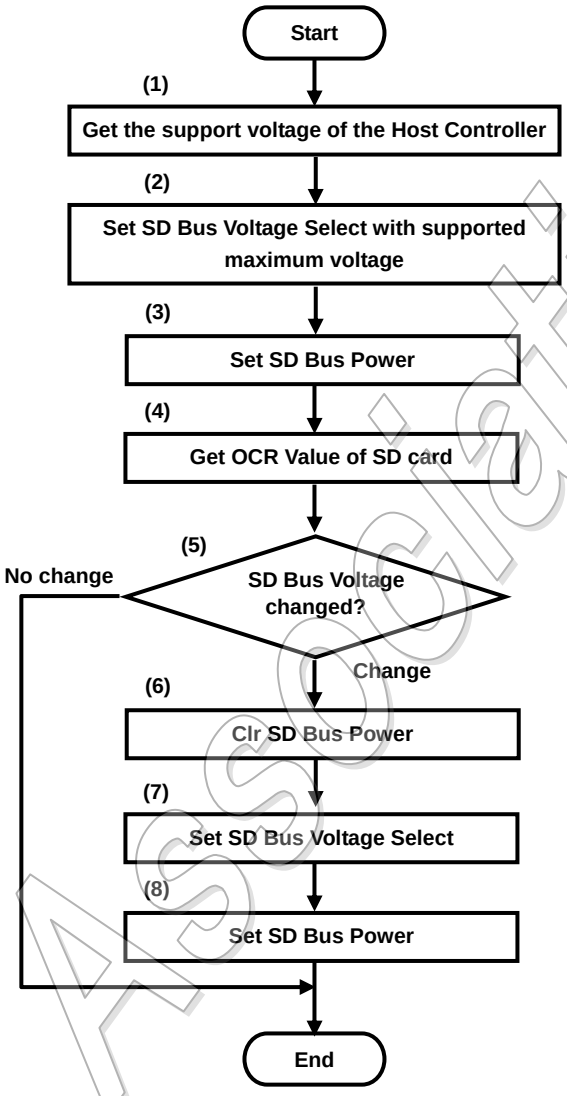


Figure 3-6: SD Bus Power Control Sequence

(1) By reading the *Capabilities* register, get the support voltage of the Host Controller.

(2) Set **SD Bus Voltage Select** in the *Power Control* register with maximum voltage that the Host Controller supports.

(3) Set **SD Bus Power** in the *Power Control* register to 1.

(4) Get the OCR value of all function internal of SD card.

(5) Judge whether SD Bus voltage needs to be changed or not. In case where SD Bus voltage needs to be changed, go to step (6). In case where SD Bus voltage does not need to be changed, go to 'End'.

(6) Set **SD Bus Power** in the *Power Control* register to 0 for clearing this bit. The card requires voltage rising from 0 volt to detect it correctly. The Host Driver shall clear **SD Bus Power** before changing voltage by setting **SD Bus Voltage Select**.

(7) Set **SD Bus Voltage Select** in the *Power Control* register.

(8) Set **SD Bus Power** in the *Power Control* register to 1.

Note:
Step (2) and step (3) can be executed at same time. And also, step (7) and step (8) can be executed at same time.

3.4 Changing Bus Width

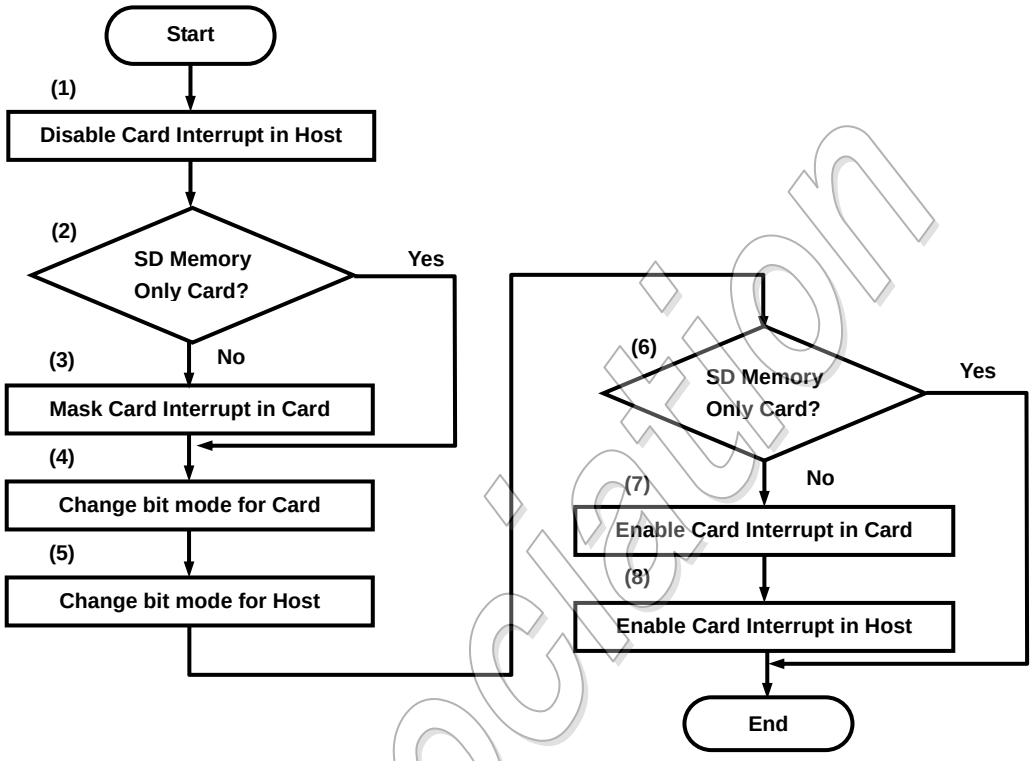


Figure 3-7: Change Bus Width Sequence

The sequence for changing bit mode on SD Bus is shown in Figure 3-7.

(1) Set **Card Interrupt Status Enable** in the *Normal Interrupt Status Enable* register to 0 for masking incorrect interrupts that may occur while changing the bus width.

(2) In case of SD memory only card, go to step (4). In case of other card, go to step (3).

(3) Set "**IENM**" of the CCCR in a SDIO or SD combo card to 0 by CMD52. Please refer to Section 3.6.1 for how to generate CMD52.

(4) Change the bit mode for a SD card. Changing SD memory card bus width by ACMD6 and changing SDIO card bus width by setting **Bus Width** of *Bus Interface Control* register in CCCR.

(5) In case of changing to 4-bit mode, set **Data Transfer Width** in the *Host Control* register to 1. In another case (1-bit mode), set this bit to 0.

(6) In case of SD memory only card, go to the 'End'. In case of other card, go to step (7).

(7) Set "**IENM**" of the CCCR in a SDIO or SD combo card to 1 by CMD52.

(8) Set Card Interrupt Status Enable in the Normal Interrupt Status Enable register to 1.

3.5 Timeout Setting on DAT Line

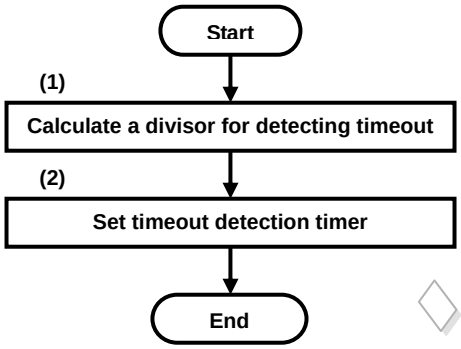


Figure 3-8: Timeout Setting Sequence

In order to detect timeout errors on DAT line, the Host Driver shall execute the following two steps before any SD transaction. For more information regarding SD transactions, refer to Section 3.6.2

(1) Calculate a divisor to detect timeout errors by reading **Timeout Clock Frequency** and **Timeout Clock Unit** in the *Capabilities* register. If **Timeout Clock Frequency** is 00 0000b, the Host System shall provide this information to the Host Driver by another method.

(2) Set **Data Timeout Counter Value** in the *Timeout Control* register in accordance with the value from step (1) above.

3.6 SD Transaction Generation

This section describes the sequences how to generate and control various kinds of SD transactions. SD transactions are classified into three cases:

(1) Transactions that do not use the DAT line.
(2) Transactions that use the DAT line only for the busy signal.
(3) Transactions that use the DAT line for transferring data.

In this specification the first and the second case's transactions are classified as "Transaction Control without Data Transfer using DAT Line", the third case's transaction is classified as "Transaction Control with Data Transfer using DAT Line".
Please refer to the specifications below for the detailed specifications on the SD Command itself:

- SD Memory Card Specification Part 1
  PHYSICAL LAYER SPECIFICATION Version 1.01
- SD Card Specification PART E1
  Secure Digital Input/Output (SDIO) Specification Version 1.00

3.6.1 Transaction Control without Data Transfer Using DAT Line

In this section, the sequence for issuing and completing the SD Command is explained. The sequence for issuing a SD Command is shown in Figure 3-9 and for completing a SD Command is shown in Figure 3-10.

3.6.1.1 The sequence for issue of the SD Command

The sequence for issuing the SD Command is detailed below.

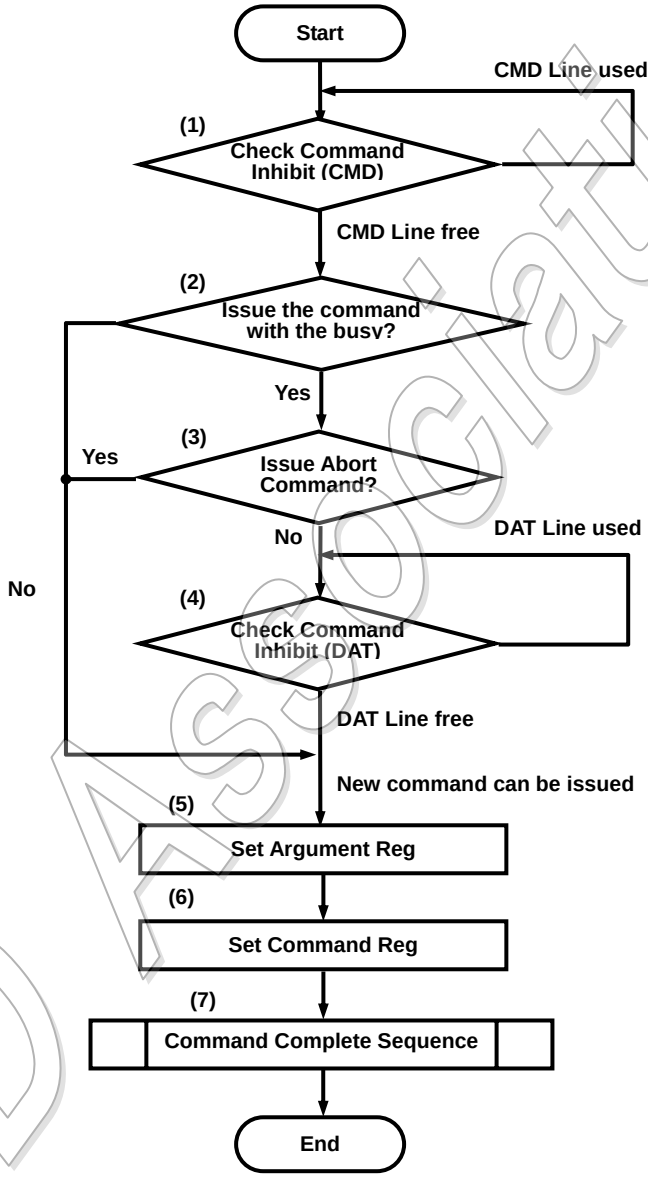


Figure 3-9: SD Command Issue Sequence

(1) Check **Command Inhibit (CMD)** in the *Present State* register. Repeat this step until **Command Inhibit (CMD)** is 0. That is, when **Command Inhibit (CMD)** is 1, the Host Driver shall not issue a SD Command.

(2) If the Host Driver issues a SD Command with busy signal, go to step (3). If without busy signal, go to step (5).

(3) If the Host Driver issues an abort command, go to step (5). In the case of no abort command, go to step (4).

(4) Check **Command Inhibit (DAT)** in the *Present State* register. Repeat this step until **Command Inhibit (DAT)** is 0.

(5) Set the value corresponding to the issued command in the *Argument* register.

(6) Set the value corresponding to the issued command in the *Command* register.
Note: Writing the upper byte in the *Command* register causes a SD command to be issued.

(7) Perform Command Complete Sequence in accordance with 3.6.1.2.

3.6.1.2 The sequence for complete command

The sequence for completing the SD Command is shown in Figure 3-10. There is a possibility that the errors (Command Index/End bit/CRC/Timeout Error) occur during this sequence.

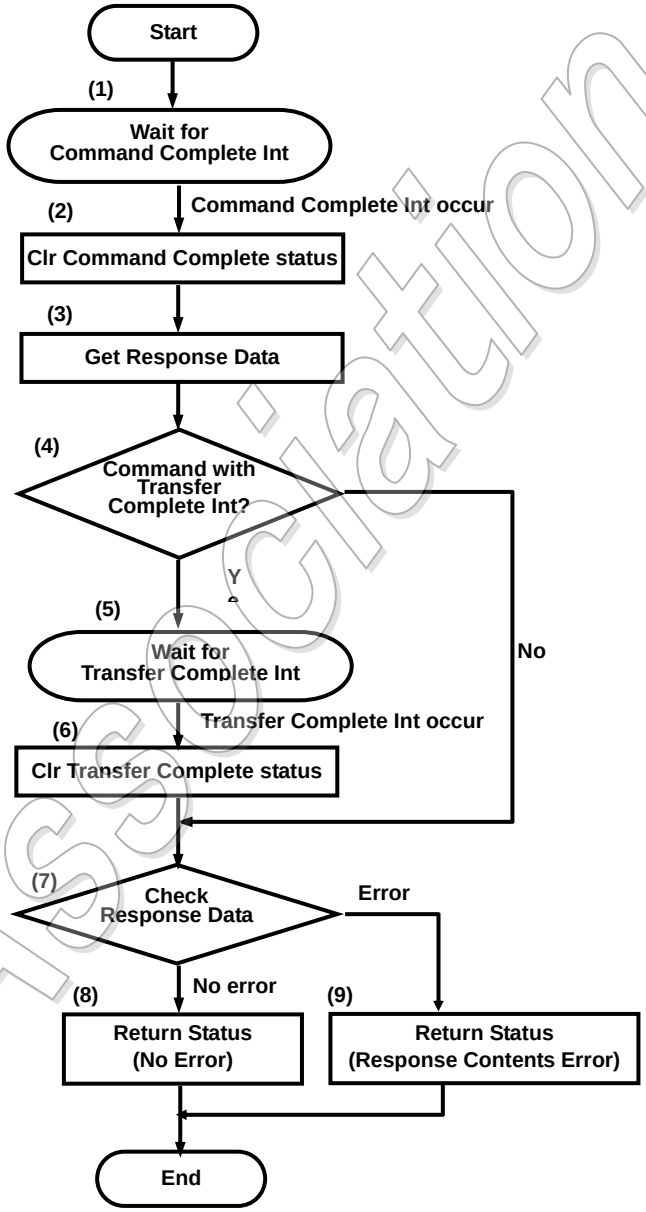


Figure 3-10: Command Complete Sequence

(1) Wait for the **Command Complete** Interrupt. If the **Command Complete** Interrupt has occurred, go to step (2).

(2) Write 1 to Command Complete in the Normal Interrupt Status register to clear this bit.

(3) Read the *Response* register and get necessary information in accordance with the issued command.

(4) Judge whether the command uses the **Transfer Complete** Interrupt or not. If it uses **Transfer Complete**, go to step (5). If not, go to step (7).

(5) Wait for the **Transfer Complete** Interrupt. If the **Transfer Complete** Interrupt has occurred, go to step (6).

(6) Write 1 to **Transfer Complete** in the *Normal Interrupt Status* register to clear this bit.

(7) Check for errors in Response Data. If there is no error, go to step (8). If there is an error, go to step (9).

(8) Return Status of "No Error".

(9) Return Status of "Response Contents Error".

Note1: While waiting for the **Transfer Complete** interrupt, the Host Driver shall only issue commands that do not use the busy signal.
Note2: The Host Driver shall judge the Auto CMD12 complete by monitoring Transfer Complete.
Note3: When the last block of un-protected area is read using memory multiple block read command (CMD18), OUT_OF_RANGE error may occur even if the sequence is correct. The Host Driver should ignore it. This error will appear in the response of Auto CMD12 or in the response of the next memory command.

3.6.2 Transaction Control with Data Transfer Using DAT Line

Depending on whether DMA (optional) is used or not, there are two execution methods. The sequence not using DMA is shown in Figure 3-11 and the sequence using DMA is shown in Figure 3-12.

In addition, the sequences for SD transfers are basically classified into following three kinds according to how the number of blocks is specified:

1) Single Block Transfer:
   The number of blocks is specified to the Host Controller before the transfer. The number of blocks specified is always one.
2) Multiple Block Transfer:
   The number of blocks is specified to the Host Controller before the transfer. The number of blocks specified shall be one or more.
3) Infinite Block Transfer:
   The number of blocks is not specified to the Host Controller before the transfer. This transfer is continued until an abort transaction is executed. This abort transaction is performed by CMD12 in the case of a SD memory card, and by CMD52 in the case of a SDIO card.

3.6.2.1 Not using DMA

The sequence for not using DMA is shown below.

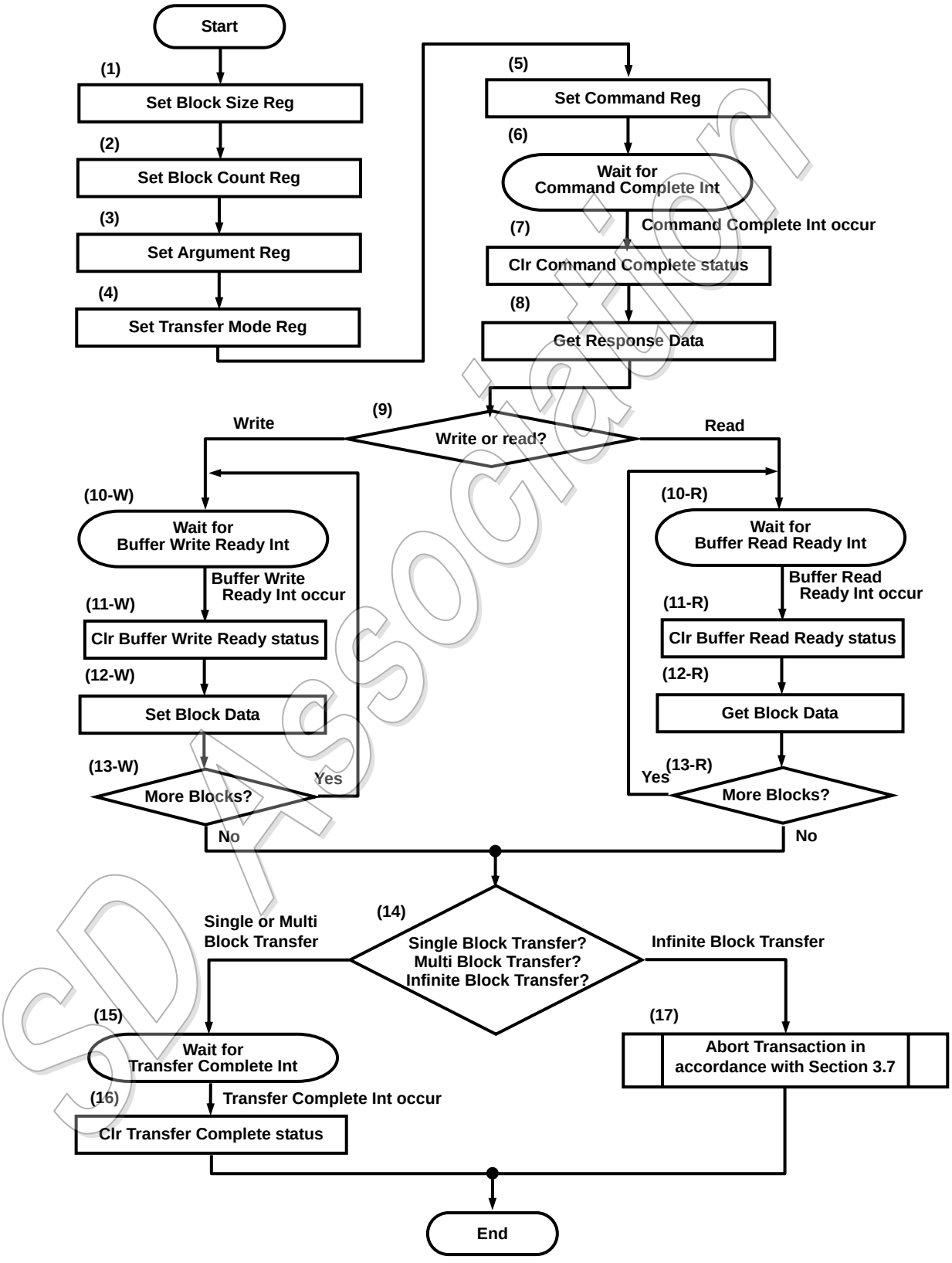


Figure 3-11: Transaction Control with Data Transfer Using DAT Line Sequence (Not using DMA)

(1) Set the value corresponding to the executed data byte length of one block to *Block Size* register.

(2) Set the value corresponding to the executed data block count to *Block Count* register in accordance with Table 2-8.

(3) Set the value corresponding to the issued command to *Argument* register.

(4) Set the value to **Multi / Single Block Select** and **Block Count Enable** in accordance with Table 2-8. And at this time, set the value corresponding to the issued command to **Data Transfer Direction**, **Auto CMD12 Enable** and **DMA Enable**.

(5) Set the value corresponding to the issued command to *Command* register.
Note: When writing the upper byte of *Command* register, SD command is issued.

(6) And then, wait for the **Command Complete** Interrupt.

(7) Write 1 to the **Command Complete** in the *Normal Interrupt Status* register for clearing this bit.

(8) Read *Response* register and get necessary information in accordance with the issued command.

(9) In the case where this sequence is for write to a card, go to step (10-W). In case of read from a card, go to step (10-R).

(10-W) And then wait for **Buffer Write Ready** Interrupt.

(11-W) Write 1 to the **Buffer Write Ready** in the *Normal Interrupt Status* register for clearing this bit.

(12-W) Write block data (in according to the number of bytes specified at the step (1)) to *Buffer Data Port* register.

(13-W) Repeat until all blocks are sent and then go to step (14).

(10-R) And then wait for the **Buffer Read Ready** Interrupt.

(11-R) Write 1 to the **Buffer Read Ready** in the *Normal Interrupt Status* register for clearing this bit.

(12-R) Read block data (in according to the number of bytes specified at the step (1)) from the *Buffer Data Port* register.

(13-R) Repeat until all blocks are received and then go to step (14).

(14) If this sequence is for Single or Multiple Block Transfer, go to step (15). In case of Infinite Block Transfer, go to step (17).

(15) Wait for **Transfer Complete** Interrupt.

(16) Write 1 to the **Transfer Complete** in the *Normal Interrupt Status* register for clearing this bit.

(17) Perform the sequence for Abort Transaction in accordance with Section 3.7.

Note: Step (1) and Step (2) can be executed at same time. Step (4) and Step (5) can be executed at same time.

3.6.2.2 Using DMA

The sequence for using DMA is shown below.

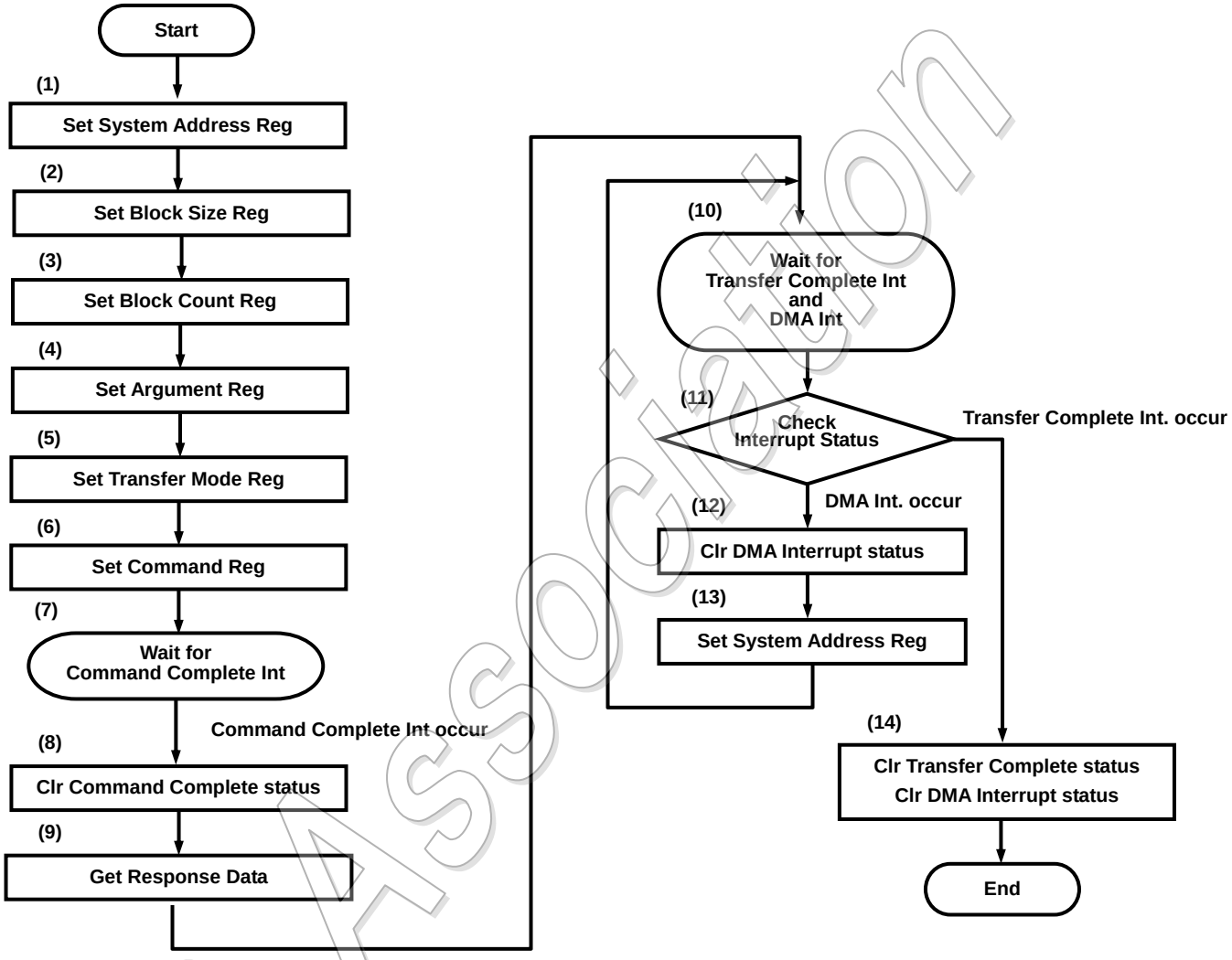


Figure 3-12: Transaction Control with Data Transfer Using DAT Line Sequence (Using DMA)

(1) Set the system address for DMA in the *System Address* register.

(2) Set the value corresponding to the executed data byte length of one block in the Block Size register.

(3) Set the value corresponding to the executed data block count in the Block Count register in accordance with Table 2-8.

(4) Set the value corresponding to the issued command in the *Argument* register.

(5) Set the values for **Multi / Single Block Select** and **Block Count Enable** in accordance with Table 2-8. And at this time, set the value corresponding to the issued command for **Data Transfer Direction**, **Auto CMD12 Enable** and **DMA Enable**.

(6) Set the value corresponding to the issued command in the *Command* register.
Note: When writing to the upper byte of the *Command* register, the SD command is issued and DMA is started.

(7) And then wait for the **Command Complete** Interrupt.

(8) Write 1 to the **Command Complete** in the *Normal Interrupt Status* register to clear this bit.

(9) Read *Response* register and get necessary information in accordance with the issued command.

(10) Wait for the **Transfer Complete** Interrupt and **DMA Interrupt**.

(11) If **Transfer Complete** is set 1, go to Step (14) else if **DMA Interrupt** is set to 1, go to Step (12).
Transfer Complete is higher priority than **DMA Interrupt**.

(12) Write 1 to the **DMA Interrupt** in the *Normal Interrupt Status* register to clear this bit.

(13) Set the next system address of the next data position to the *System Address* register and go to Step (10).

(14) Write 1 to the **Transfer Complete** and **DMA Interrupt** in the *Normal Interrupt Status* register to clear this bit.

Note: Step (2) and Step (3) can be executed simultaneously. Step (5) and Step (6) can also be executed simultaneously.

3.7 Abort Transaction

An abort transaction is performed by issuing CMD12 for a SD memory card and by issuing CMD52 for a SDIO card. There are two cases where the Host Driver needs to do an Abort Transaction. The first case is when the Host Driver stops Infinite Block Transfers. The second case is when the Host Driver stops transfers while a Multiple Block Transfer is executing.

There are two ways to issue an Abort Command. The first is an asynchronous abort. The second is a synchronous abort. In an asynchronous abort sequence, the Host Driver can issue an Abort Command at anytime unless **Command Inhibit (CMD)** in the *Present State* register is set to 1. In a synchronous abort, the Host Driver shall issue an Abort Command after the data transfer stopped by using **Stop At Block Gap Request** in the *Block Gap Control* register.

3.7.1 Asynchronous Abort

The sequence for Asynchronous Abort is shown in Figure 3-13.

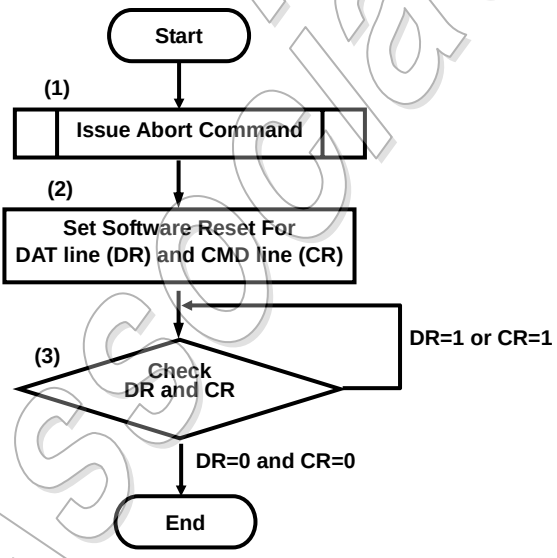


Figure 3-13: Asynchronous Abort Sequence

(1) Issue Abort Command in accordance with Section 3.6.1

(2) Set both **Software Reset For DAT Line** and **Software Reset For CMD Line** to 1 in the *Software Reset* register to do software reset.

(3) Check **Software Reset For DAT Line** and **Software Reset For CMD Line** in the *Software Reset* register. If both **Software Reset For DAT Line** and **Software Reset For CMD Line** are 0, go to "End". If either **Software Reset For DAT Line** or **Software Reset For CMD Line** is 1, go to step (3).

3.7.2 Synchronous Abort

The sequence for Synchronous Abort is shown in Figure 3-14

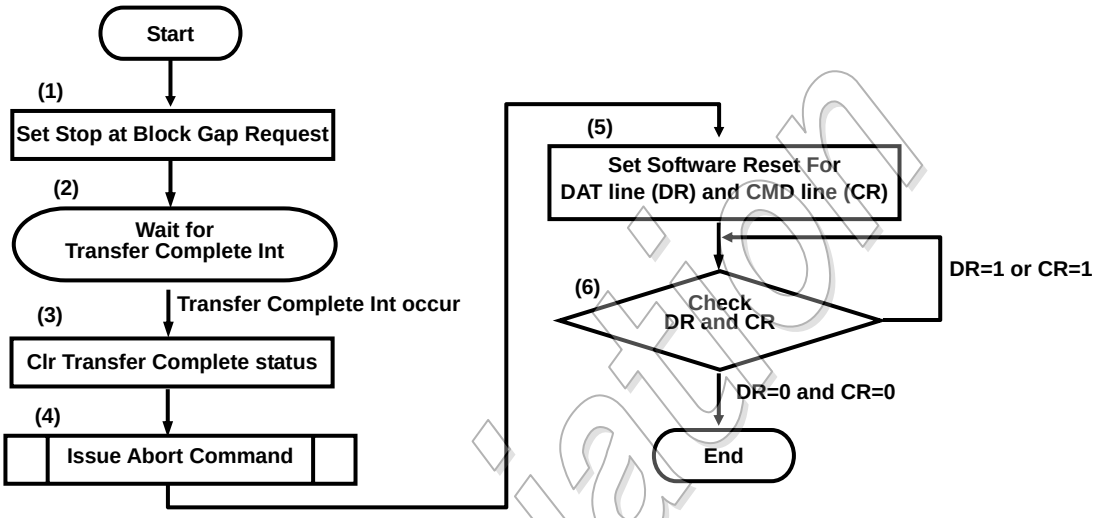


Figure 3-14: Synchronous Abort Sequence

(1) Set the **Stop At Block Gap Request** in the *Block Gap Control* register to 1 to stop SD transactions.

(2) Wait for the Transfer Complete Interrupt.

(3) Set the **Transfer Complete** to 1 in the *Normal Interrupt Status* register to clear this bit.

(4) Issue the Abort Command in accordance with Section 3.6.1

(5) Set both **Software Reset For DAT Line** and **Software Reset For CMD Line** to 1 in the *Software Reset* register to do software reset.

(6) Check both **Software Reset For DAT Line** and **Software Reset For CMD Line** in the *Software Reset* register. If both **Software Reset For DAT Line** and **Software Reset For CMD Line** are 0, go to 'End'. If either **Software Reset For DAT Line** or **Software Reset For CMD Line** is 1, go to step (6).

3.8 Error Recovery

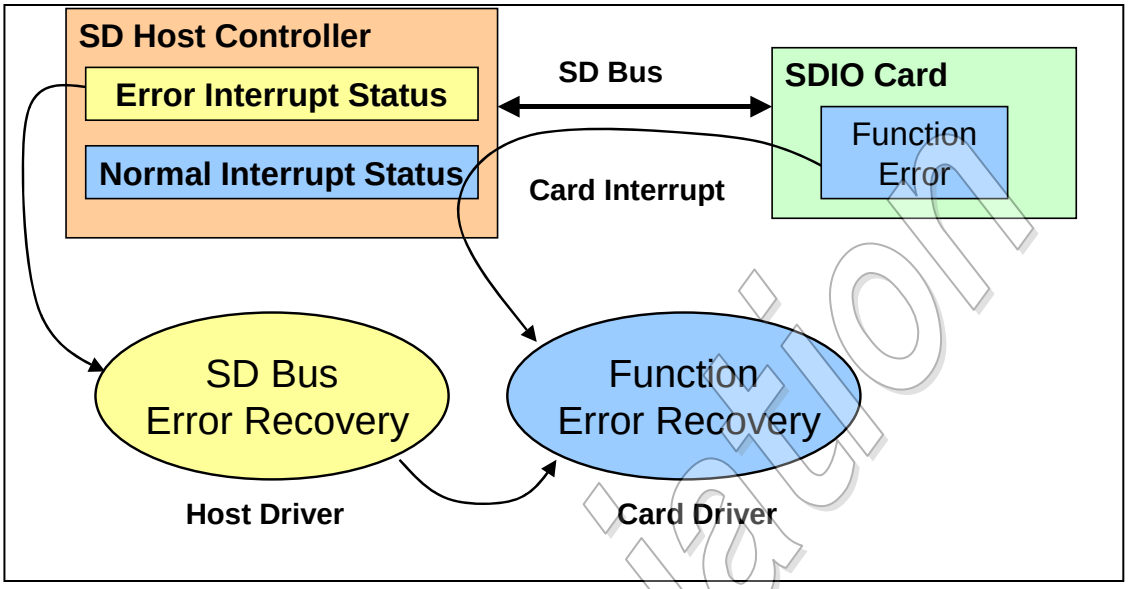


Figure 3-15 : Error Report and Recovery

Figure 3-15 shows concept of error report and its recovery. The Host Controller has 2 interrupt status registers. If error occur in the SD Bus transaction, one of the bits is set in the *Error Interrupt Status* register. If the function errors occur in the SDIO card, the card interrupt informs these function errors and the **Card interrupt** is set in the *Normal Interrupt Status* register. (The **Card Interrupt** is used for not only error usage but also normal usage. For example to inform function ready.) The Card Driver shall do function error recovery because the Host Driver doesn't know how to control the function. In the case that function error occurs due to SD Bus error, SD Bus error recovery is required before function error recovery. Abort command is used to recover SD Bus, then the Host Driver should save error statuses related to SD Bus errors before issuing abort command and transfer these statuses to the Card Driver. These statuses may be used to recover function error. Following explanations are related to SD Bus error recovery. This specification does not specify the function error recovery.

Implementation Note:
If the Card Driver cannot recover the function errors, the Host Driver should try following methods.

(1) Using **IOEx** for SDIO card
IOEx may be used as the reset per function basis. Sequence is as follows:
Clear **IOEx**=0 and wait until **IORx**=0 and then set **IOEx**=1 again. SDIO may be recovered when IORx=1.

(2) Using reset command for memory and SDIO card
Re-initialization sequence is required.

(3) Off and on power supply for the SD Bus
The card may be recovered by the power on reset. Re-initialization sequence is required.

The two cases where the Host Driver needs the "Error Recovery" sequence are classified as follows:

(1) Error Interrupt Recovery:
If error interrupt is indicated by the *Error Interrupt Status* register, the Host Driver shall apply this sequence.

(2) Auto CMD12 Error Recovery:
If there are errors in Auto CMD12, the Host Driver shall apply this sequence. In terms of Return Status, Auto CMD12 Error Recovery is classified into 4 cases. It is shown in Figure 3-16. If error occurs during memory write transfer, strongly recommend using ACMD22 and then in the following recovery sequence, retry to send remaining blocks not written.

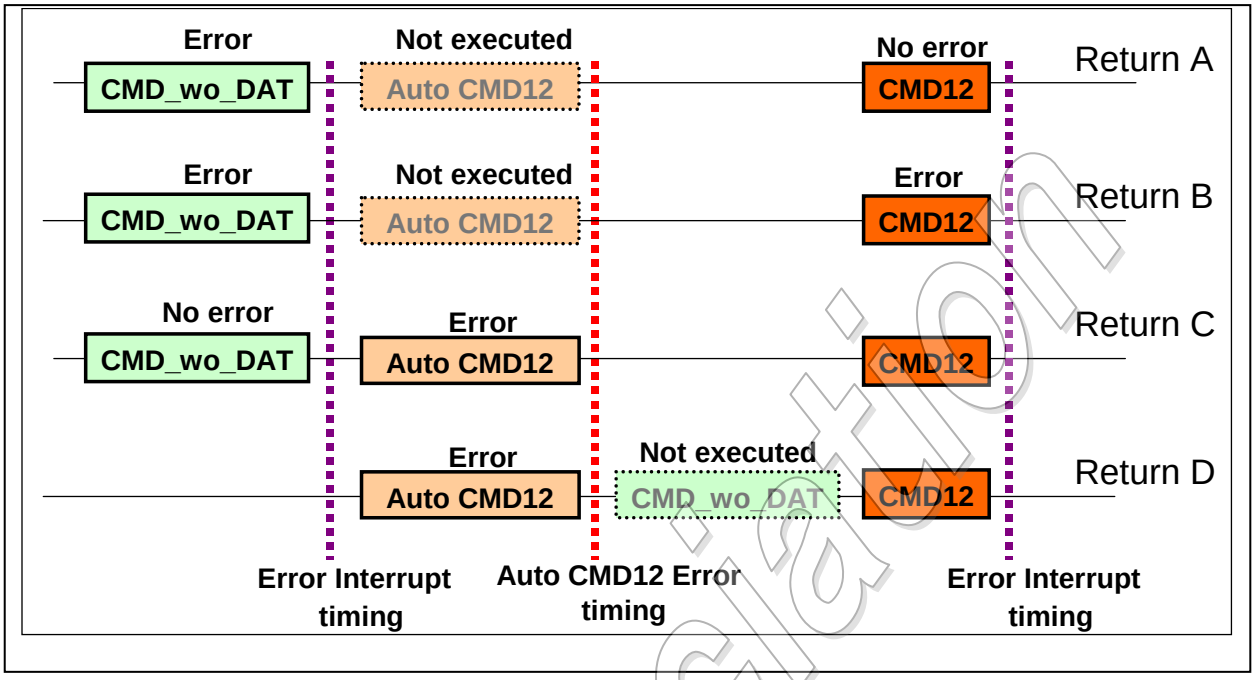


Figure 3-16: Return Status of Auto CMD12 Error Recovery

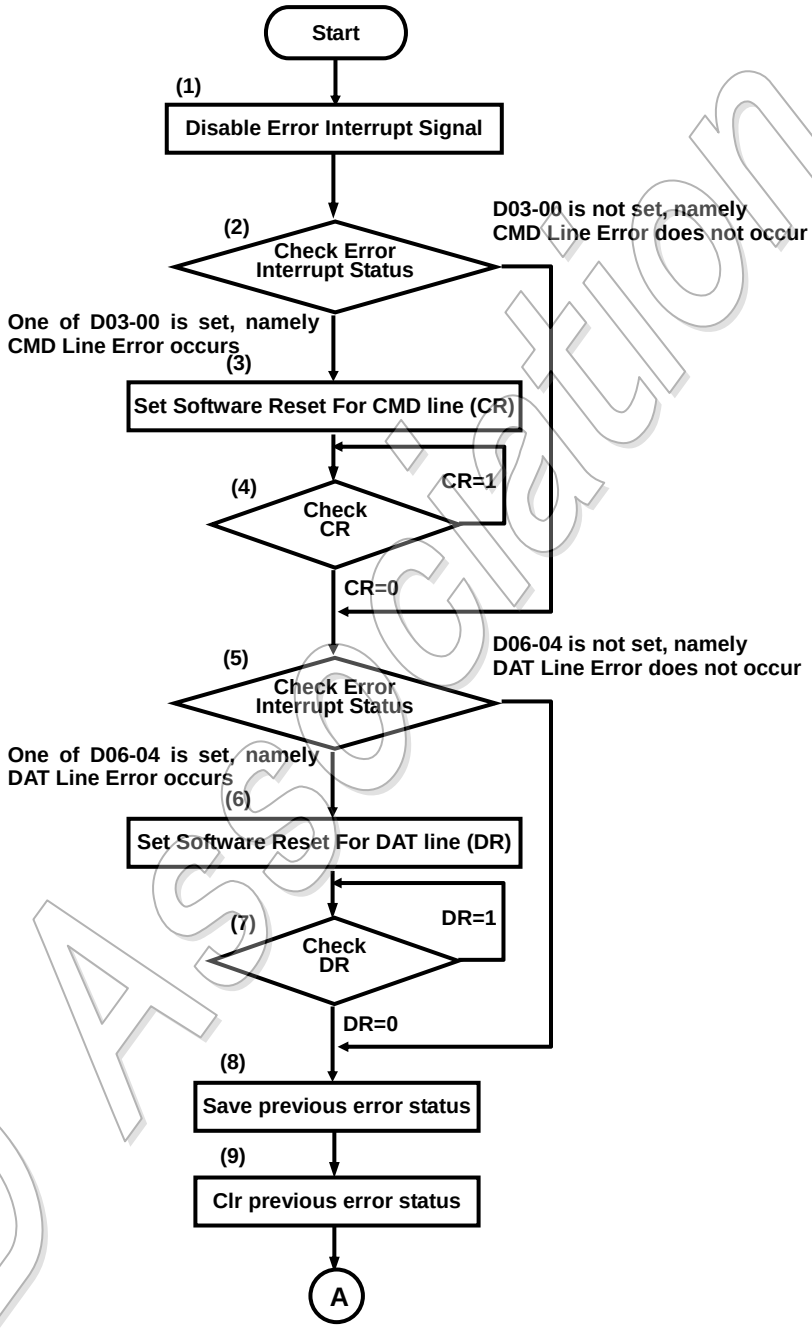
Implementation Note:
Abort command is used to recover from SD Bus error. SDIO transaction abort using CMD52 returns response but in the case of memory transaction abort using CMD12, response returns depending on the memory card state. If no response returns after issue CMD12, the Host Driver should check card state using CMD13. If the state is "tran" in the CURRENT_STATE, consider CMD12 is successful.

Implementation Note:
The following sequence is one possible error recovery flow. There may be another methods, sometimes using interrupts or polling. It can be possible to use another flows, based on Host System requirements.

In these error recovery sequences, return statuses for the next sequence. When the Host Controller cannot issue the next command due to SD Bus error, the error recovery sequences return "Non-recoverable" status. In this case, the Host System may cut off power to the SD Bus, and then power on SD Bus and initialize both the Host Controller and the SD card again.

3.8.1 Error Interrupt Recovery

The sequence for Error Interrupt Recovery is shown in Figure 3-17.



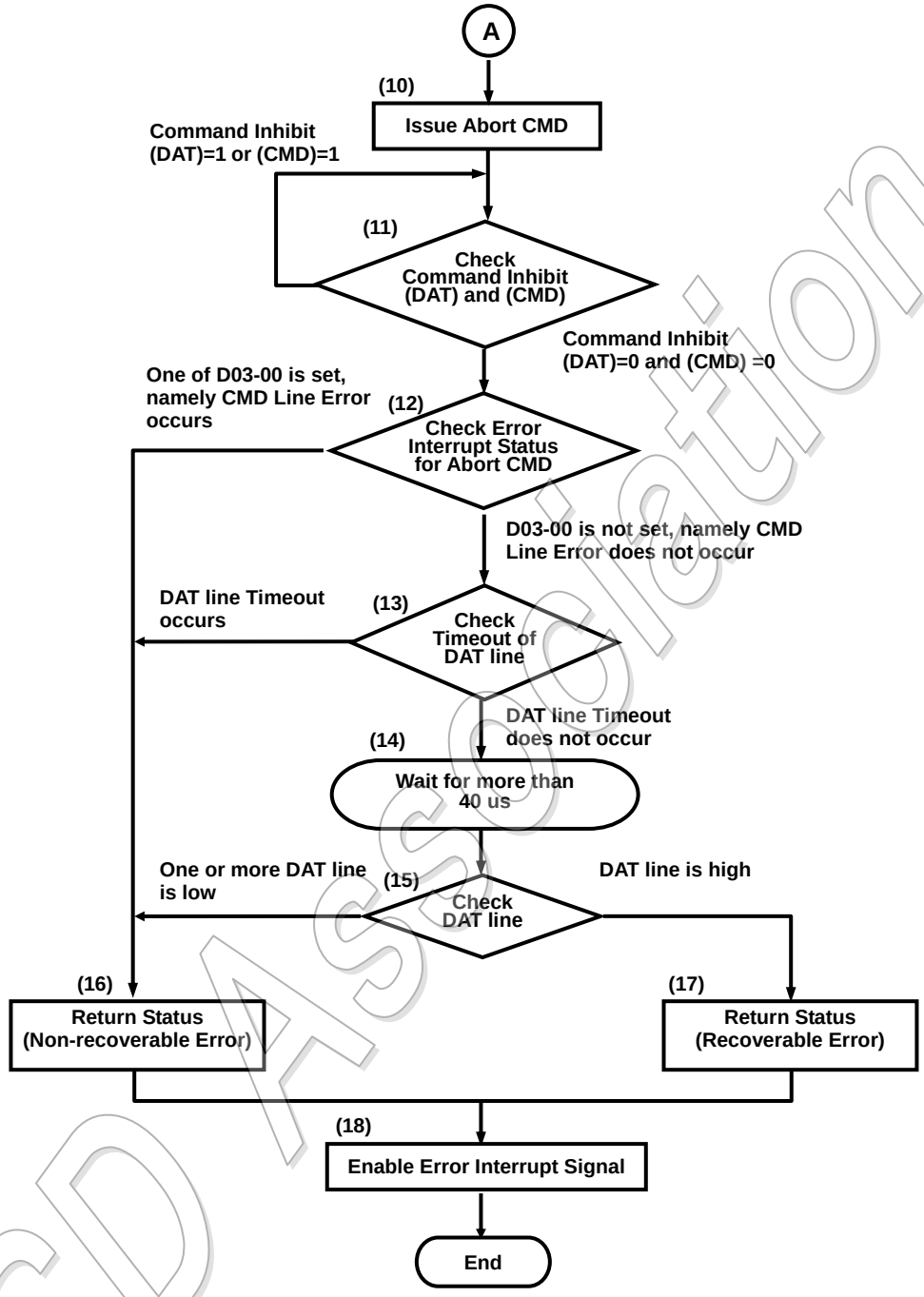


Figure 3-17: Error Interrupt Recovery Sequence

(1) Disable the Error Interrupt Signal.

(2) Check bits D03-00 in the *Error Interrupt Status* register. If one of these bits (D03-00) is set to 1, go to step (3). If none are set to 1 (all are 0), go to step (5).

(3) Set **Software Reset For CMD Line** to 1 in the *Software Reset* register for software reset of the CMD line.

(4) Check **Software Reset For CMD Line** in the *Software Reset* register. If **Software Reset For CMD Line** is 0, go to step (5). If it is 1, go to step (4).

(5) Check bits D06-04 in the *Error Interrupt Status* register. If one of these bits (D06-04) is set to 1, go to step (6). If none are set to 1 (all are 0), go to step (8).

(6) Set **Software Reset For DAT Line** to 1 in the *Software Reset* register for software reset of the DAT line.

(7) Check **Software Reset For DAT Line** in the *Software Reset* register. If **Software Reset For DAT Line** is 0, go to step (8). If it is 1, go to step (7).

(8) Save previous error status.

(9) Clear previous error status with setting them to 1.

(10) Issue Abort Command.

(11) Check **Command Inhibit (DAT)** and **Command Inhibit (CMD)** in the *Present State* register. Repeat this step until both **Command Inhibit (DAT)** and **Command Inhibit (CMD)** are set to 0.

(12) Check bits D03-00 in the *Error Interrupt Status* register for Abort Command. If one of these bits is set to 1, go to step (16). If none of these bits are set to 1 (all are 0), go to step (13).

(13) Check **Data Timeout Error** in the *Error Interrupt Status* register. If this bit is set to 1, go to step (16). If it is 0, go to step (14).

(14) Wait for more than 40 us.

(15) By monitoring the **DAT [3:0] Line Signal Level** in the *Present State* register, judge whether the level of the DAT line is low or not. If one or more DAT lines are low, go to step (16). If the DAT lines are high, go to step (17).

(16) Return Status of "Non-recoverable Error".

(17) Return Status of "Recoverable Error".

(18) Enable the Error Interrupt Signal.

3.8.2 Auto CMD12 Error Recovery

The sequence for Auto CMD12 Error Recovery is shown in Figure 3-18.

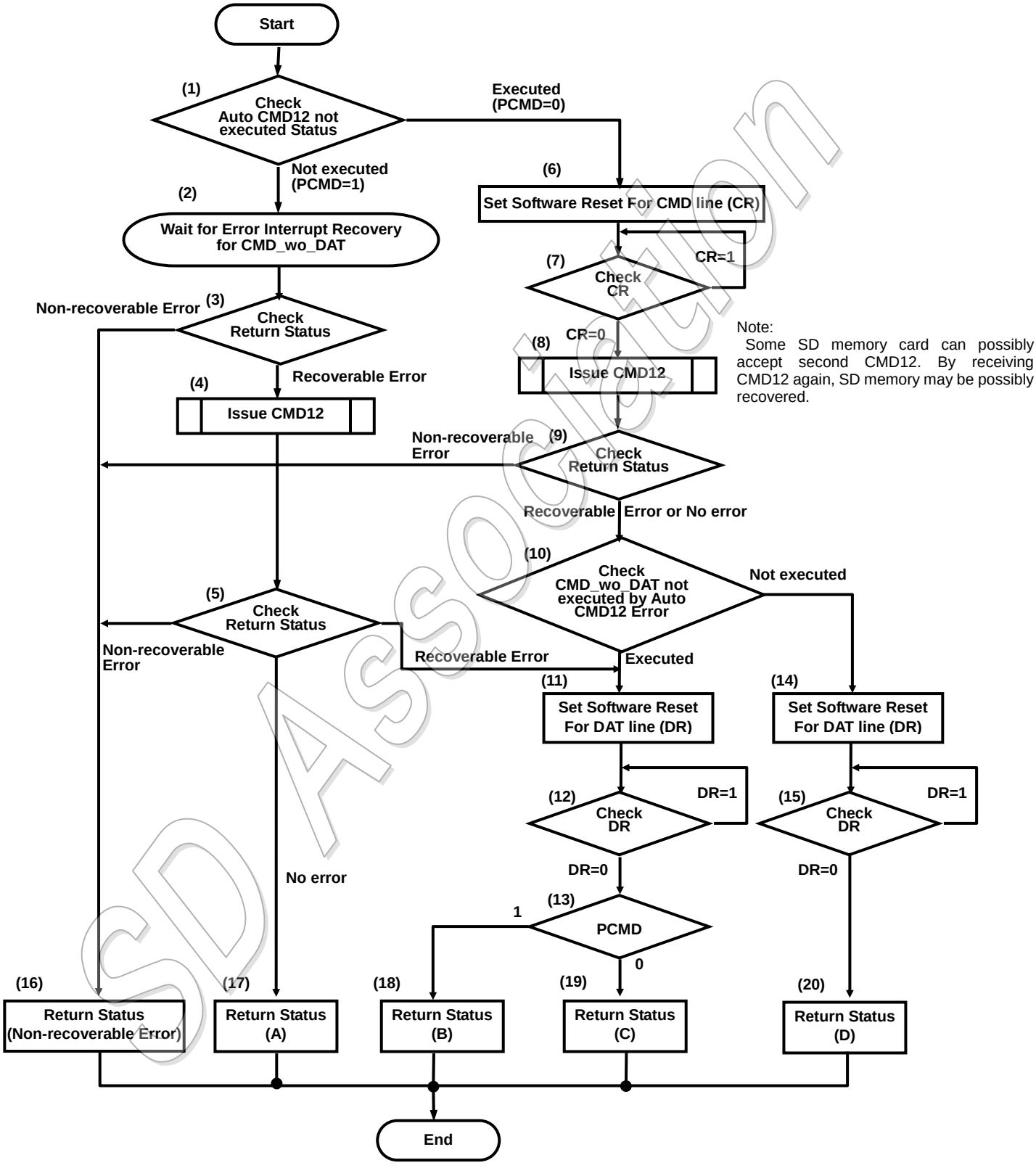


Figure 3-18: Auto CMD12 Error Recovery Sequence

The sequence for Auto CMD12 Error Recovery covers the following four cases:

A: An error occurred in CMD_wo_DAT, but not in the SD memory transfer.
B: An error occurred in CMD_wo_DAT, and also occurred in the SD memory transfer.
C: An error did not occur in CMD_wo_DAT, but an error occurred in the SD memory transfer.
D: CMD_wo_DAT was not issued, and an error occurred in the SD memory transfer.

(1) Check **Auto CMD12 Not Executed** in the *Auto CMD12 Error Status* register. If this bit is set to 1, go to step (2). If this bit is set to 0, go to step (6). In addition, the Host Driver shall define **PCMD** flag, which changes to 1 if **Auto CMD12 Not Executed** is set to 1.

(2) Wait for Error Interrupt Recovery for CMD_wo_DAT.

(3) Check "Return Status". In the case of "Non-recoverable Error", go to step (16). In the case of "Recoverable Error", go to step (4).

(4) Issue CMD12 in accordance with Section 3.6.1

(5) If the CMD line errors occur for the CMD12 (One of D03-00 is set in the Error Interrupt Status register), "Return Status" is "Non-recoverable Error" and go to step (16). If not CMD line error and busy timeout error occur (D04 is set in the Error Interrupt Status register), "Return Status" is "Recoverable Error" and go to step (11). Otherwise, "Return Status" is "No error" and go to step (17).

(6) Set **Software Reset For CMD Line** to 1 in the *Software Reset* register for software reset of the CMD line.

(7) Check **Software Reset For CMD Line** in the *Software Reset* register. If **Software Reset For CMD Line** is 0, go to step (8). If it is 1, go to step (7).

(8) Issue CMD12 in accordance with Section 3.6.1

(9) Check "Return Status" for CMD12. If "Return Status" returns "Non-recoverable Error", go to step (16). In the case of "Recoverable Error" or "No error", go to step (10).

(10) Check the **Command Not Issued By Auto CMD12 Error** in the *Auto CMD12 Error Status* register. If this bit is 0, go to step (11). If it is 1, go to step (14).

(11) Set **Software Reset For DAT Line** to 1 in the *Software Reset* register for software reset of the DAT line.

(12) Check **Software Reset For DAT Line** in the *Software Reset* register. If **Software Reset For DAT Line** is 0, go to step (13). If it is 1, go to step (12).

(13) Check the **PCMD** flag. If **PCMD** is 1, go to step (18). If it is 0, go to step (19).

(14) Set **Software Reset For DAT Line** to 1 in the *Software Reset* register for software reset of the DAT line.

(15) Check **Software Reset For DAT Line** in the *Software Reset* register. If **Software Reset For DAT Line** is 0, go to step (20). If it is 1, go to step (15).

(16) Return Status of "Non-recoverable Error".

(17) Return Status that an error has occurred in CMD_wo_DAT, but not in the SD memory transfer.

(18) Return Status that an error has occurred in both CMD_wo_DAT, and the SD memory transfer.

(19) Return Status that an error has not occurred in CMD_wo_DAT, but has occurred in the SD memory transfer.

(20) Return Status that CMD_wo_DAT has not been issued, and an error has occurred in the SD memory transfer.

3.9 Wakeup Control (Optional)

After the Host System goes into standby mode, the Host System can resume from standby via a wakeup event initiated by one of the following three events:

(1) Interrupt from a SD card:

If an SD card interrupt occurs, the Host System can resume from standby mode. If the Host System uses this wakeup factor, SD Bus power shall be kept on.

(2) Insertion of SD card:

If a SD card is inserted, the Host System can resume from standby mode.

(3) Removal of SD card:

If a SD card is removed, the Host System can resume from standby mode.

The sequence for preparing wakeup before the Host System goes into standby mode is shown in Figure 3-19.

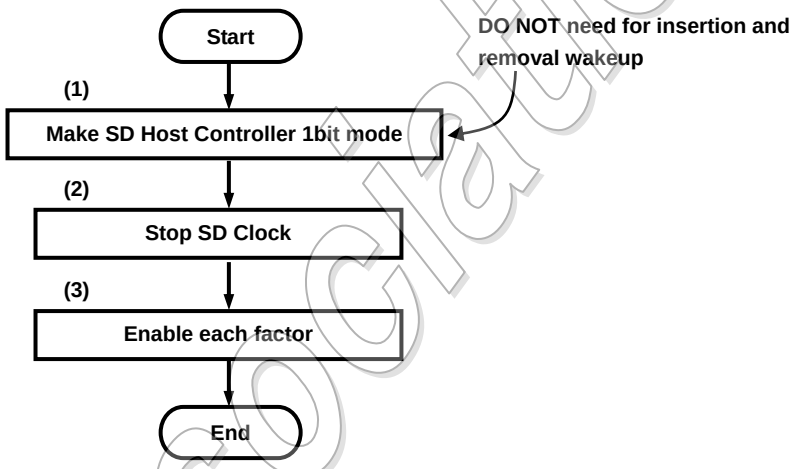


Figure 3-19: Wakeup Control before Standby Mode

(1) Set Data Transfer Width to 0 in the Host Control register.

(2) Set SD Clock Enable to 0 in the Clock Control register.

(3) Clear the *Normal Interrupt Status* register and the *Normal Interrupt Signal Enable* register, and then set the enable bits of each wakeup event factor to 1 in the *Wakeup Control* register and set the bits of *Normal Interrupt Status Enable* register to use wakeup.

The sequence for wakeup once in standby mode is shown in Figure 3-20.

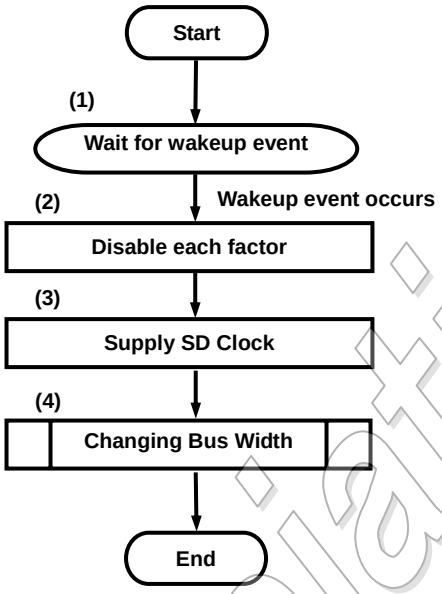


Figure 3-20: Wakeup from Standby

(1) Wait for wakeup event.

(2) Set the enable bits of each wakeup event factor to 0 in the *Wakeup Control* register and then clear event statuses in the *Normal Interrupt Status* register. If necessary, set the *Normal Interrupt Signal Enable* register.

(3) Set **SD Clock Enable** to 1 in the *Clock Control* register.

(4) Set the SD Bus width in accordance with Section 3.4.

3.10 Suspend/Resume (Optional)

If a SD card supports suspend and resume functionality, then the Host Controller can initiate suspend and resume. It is necessary for both the Host Controller and the SD card to support the function of "**Read Wait**".

3.10.1 Suspend Sequence

The sequence for suspend is shown in Figure 3-21.

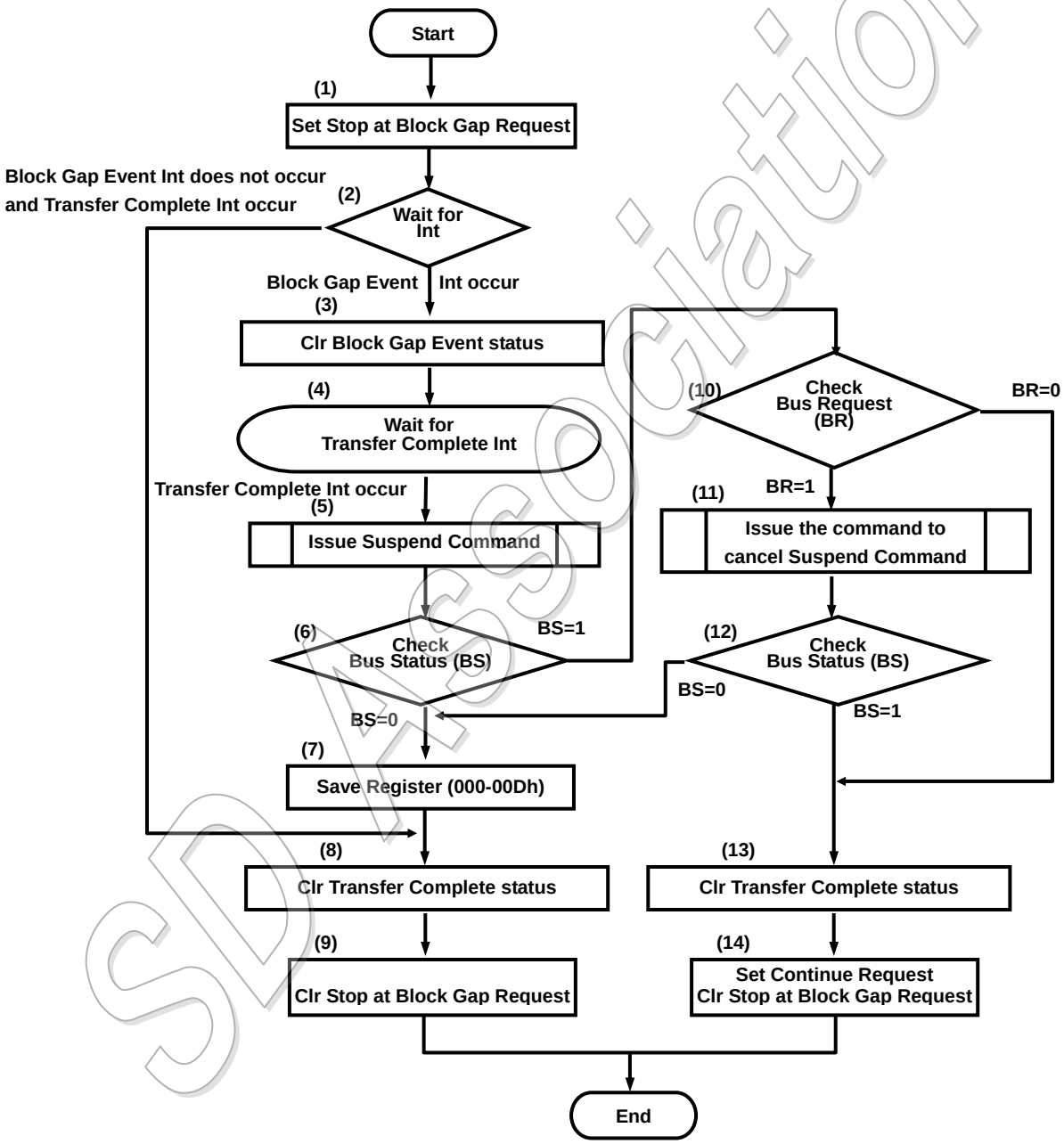


Figure 3-21 : The sequence for suspend

(1) Set **Stop At Block Gap Request** to 1 in the *Block Gap Control* register to stop the SD transaction.

(2) Wait for an Interrupt. If **Block Gap Event** is set to 0 and **Transfer Complete** is set to 1 in the *Normal Interrupt Status* register, go to step (8). If **Block Gap Event** is set to 1, go to step (3).

(3) Set **Block Gap Event** to 1 in the *Normal Interrupt Status* register to clear this bit.

(4) Wait for the **Transfer Complete** Interrupt.

(5) Issue the Suspend Command in accordance with Section 3.6.1.

(6) Check the **BS** value of the response data. If **BS** is 0, go to step (7). If **BS** is 1, go to step (10).

(7) Save the register (000h-00Dh).

(8) Set **Transfer Complete** to 1 in the *Normal Interrupt Status* register to clear this bit.

(9) Set **Stop At Block Gap Request** to 0 in the *Block Gap Control* register to clear this bit.

(10) Check the **BR** value of the response data. If **BR** is 1, go to step (11). If **BR** is 0, go to step (13).

(11) Issues the command to cancel the previous suspend command in accordance with Section 3.6.1 Transaction Control without Data Transfer Using DAT Line.

(12) Check the **BS** value of the response data. If **BS** is 0, go to step (7). If **BS** is 1, go to step (13).

(13) Set **Transfer Complete** to 1 in the *Normal Interrupt Status* register to clear this bit.

(14) Set **Continue Request** to 1 in the *Block Gap Control* register to continue the transaction. At the same time, write 0 to **Stop At Block Gap Request** to clear this bit.

The Table 3-1 shows conditions to be able to use Suspend / Resume function.

Conditions			**Suspend/Resume Function**	
Host Suspend/Resume Support	**Card Suspend/Resume Support**	**Card Read Wait Support**	**Write Suspend/Resume**	**Read Suspend/Resume**
Not supported	Don't care	Don't care	Cannot be used	Cannot be used
Supported	Not supported	Don't care	Cannot be used	Cannot be used
Supported	Supported	Not supported	Can be used	Cannot be used
Supported	Supported	Supported	Can be used	Can be used

Table 3-1 Suspend / Resume Condition

3.10.2 Resume Sequence

The sequence for resume is shown in Figure 3-22.

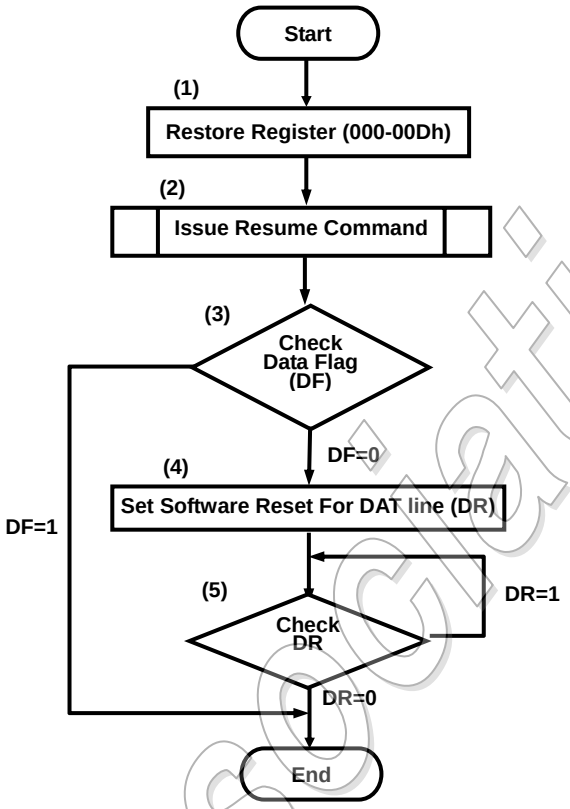


Figure 3-22 : The sequence for resume

(1) Restore the register (000h-00Dh).

(2) Issue the Resume Command in accordance with Section 3.6.1.

(3) Check the **DF** value of the response data. If **DF** is 0, go to step (4). If **DF** is 1, go to 'End'.

(4) Set **Software Reset For DAT Line** to 1 in the *Software Reset* register for software reset of the DAT line.

(5) Check **Software Reset For DAT Line** in the *Software Reset* register. If **Software Reset For DAT Line** is 0, go to 'End'. If it is 1, go to step (5).

3.10.3 Read transaction wait / continue timing

Implementation Note: Read Wait, **DAT Line Active** and **Read Transfer Active** shall be set and cleared by the Host Controller. **Stop At Block Gap Request** shall be set and cleared by the Host Driver. **Continue Request** shall be set by the Host Driver and be cleared by the Host Controller. **Block Gap Event** and **Transfer Complete** shall be set by the Host Controller and be cleared by the Host Driver.

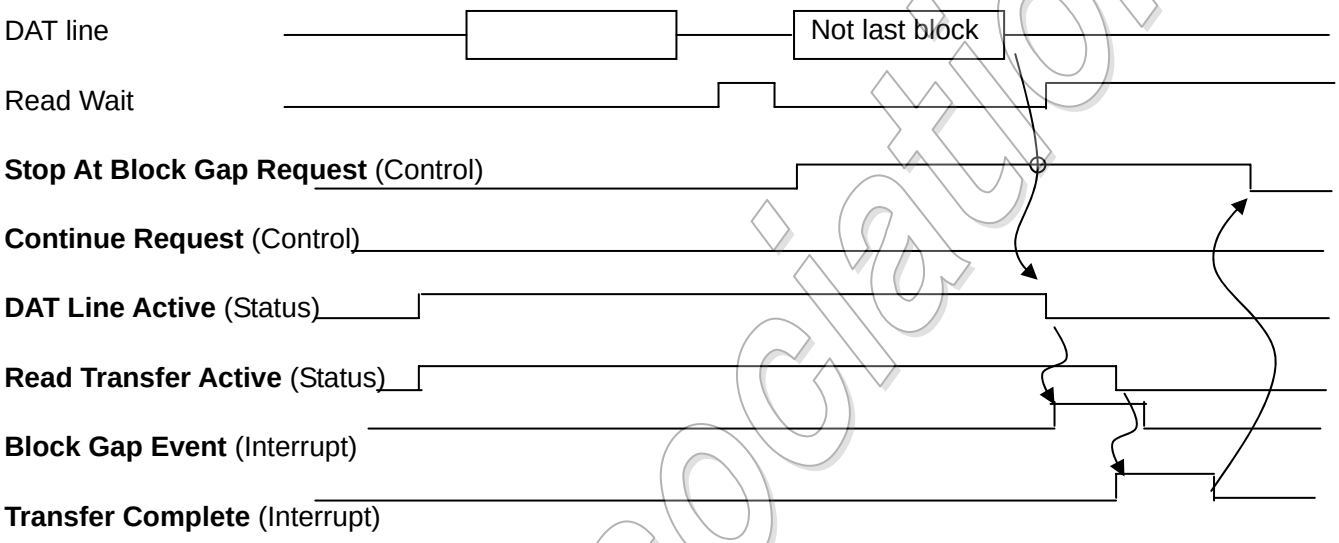


Figure 3-23 : Wait read transfer by Stop At Block Gap Request

The Host Controller can accept a **Stop At Block Gap Request** when all the following conditions are met.

(1) It is at the block gap.
(2) The Host Controller can assert read wait or it is already asserted.
(3) **Read Wait Control** is set to 1.

After accepting the **Stop At Block Gap Request**

(1) Clear **DAT Line Active** status and generate the **Block Gap Event** Interrupt
(2) After all valid data has been read (No valid read data remains in the Host Controller), clear the **Read Transfer Active** Status and generate the **Transfer Complete** Interrupt.
(3) After accepting Transfer Complete Interrupt, clear the **Stop At Block Gap Request**

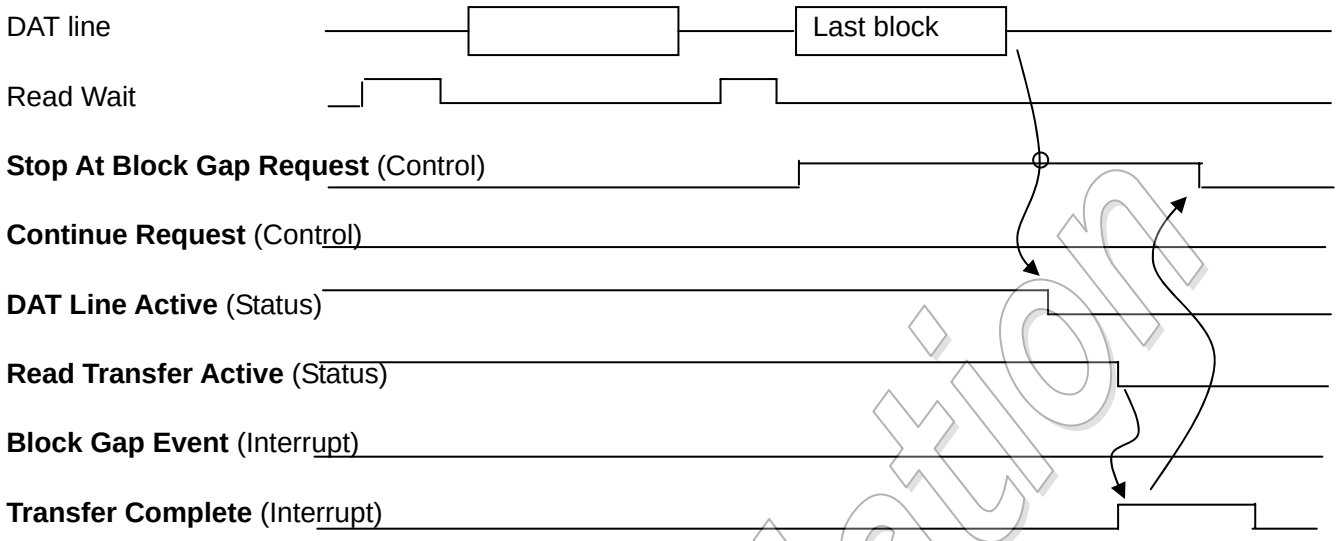


Figure 3-24 : Stop At Block Gap Request is not accepted at the last block of the read transfer

If the **Stop At Block Gap Request** is set to 1 during the last block transfer, the Host Controller shall not accept the **Stop At Block Gap Request** and stops the transaction normally. The **Block Gap Event** Interrupt is not generated. When the Transfer Complete Interrupt is generated, and if the **Block Gap Event** Status is not set to 1, the driver shall clear the **Stop At Block Gap Request**.

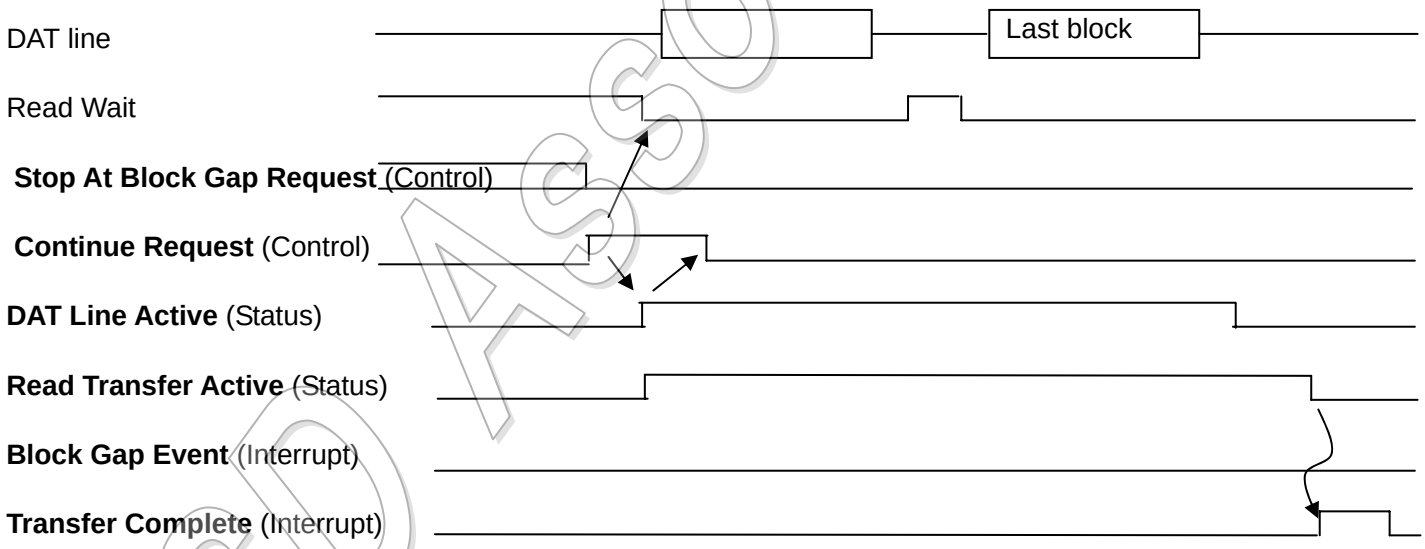


Figure 3-25 : Continue read transfer by Continue Request

To restart a stopped data transfer, set the **Continue Request** to 1. (The **Stop At Block Gap Request** shall be set to 0.)
After accepting the **Continue Request**,

(1) Release Read Wait (if the data block can accept the next data.)
(2) Set the **DAT Line Active** status and the **Read Transfer Active** Status
(3) The **Continue Request** is automatically cleared by (2).

The end of the read transfer specified by data length.

(1) Clear the **DAT Line Active** status and do not generate the **Block Gap Event** Interrupt.
(2) After all valid data has been read (No valid read data remains in the Host Controller), clear the **Read Transfer Active** Status and generate the **Transfer Complete** Interrupt.

3.10.4 Write transaction wait / continue timing

Implementation Note:
DAT Line Active and **Write Transfer Active** shall be set and cleared by the Host Controller.
Stop At Block Gap Request shall be set and cleared by the Host Driver.
Continue Request shall be set by the Host Driver and be cleared by the Host Controller.
Block Gap Event and **Transfer Complete** shall be set by the Host Controller and be cleared by the Host Driver.

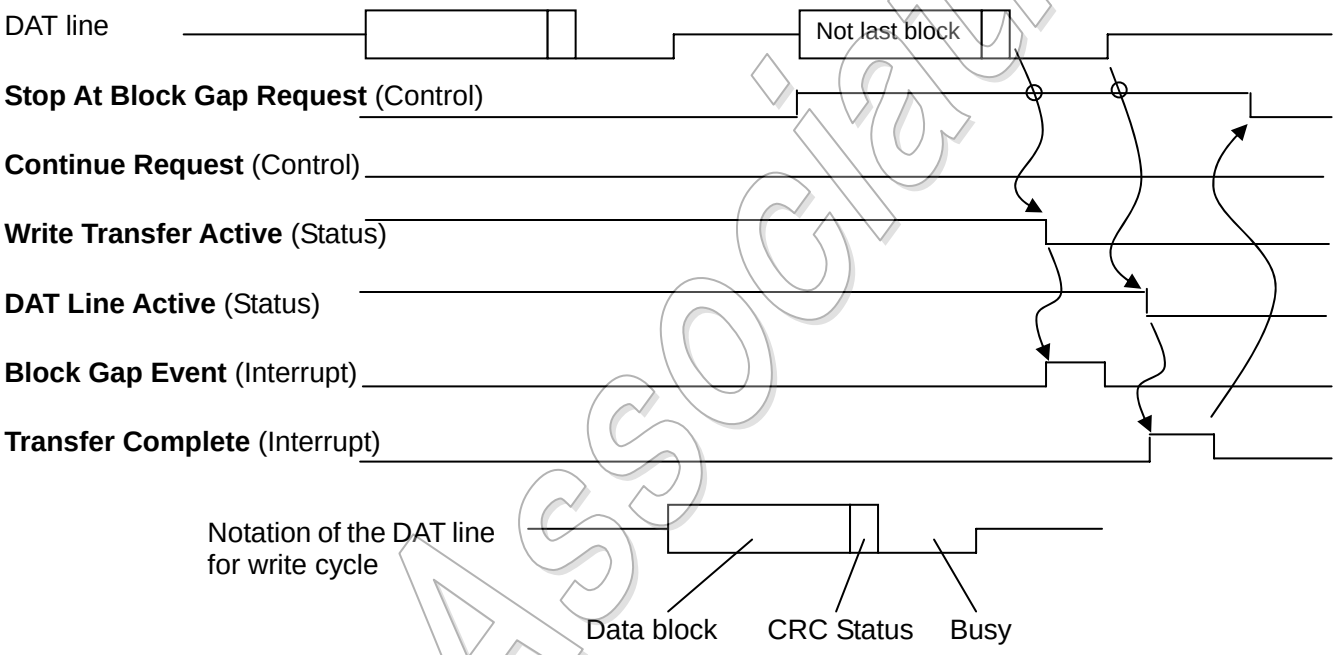


Figure 3-26 : Wait write transfer by Stop At Block Gap Request

The Host Controller can accept the **Stop At Block Gap Request** when matches all following conditions

(1) It is at the block gap.
(2) No valid write data remains in the Host Controller

After accepting the **Stop At Block Gap Request**

(1) Clear the **Write Transfer Active** Status and generate the **Block Gap Event** Interrupt
(2) After the busy signal is released, clear the **DAT Line Active** status and generate the **Transfer Complete** Interrupt.
(3) After accepting the **Transfer Complete** Interrupt, clear the **Stop At Block Gap Request**

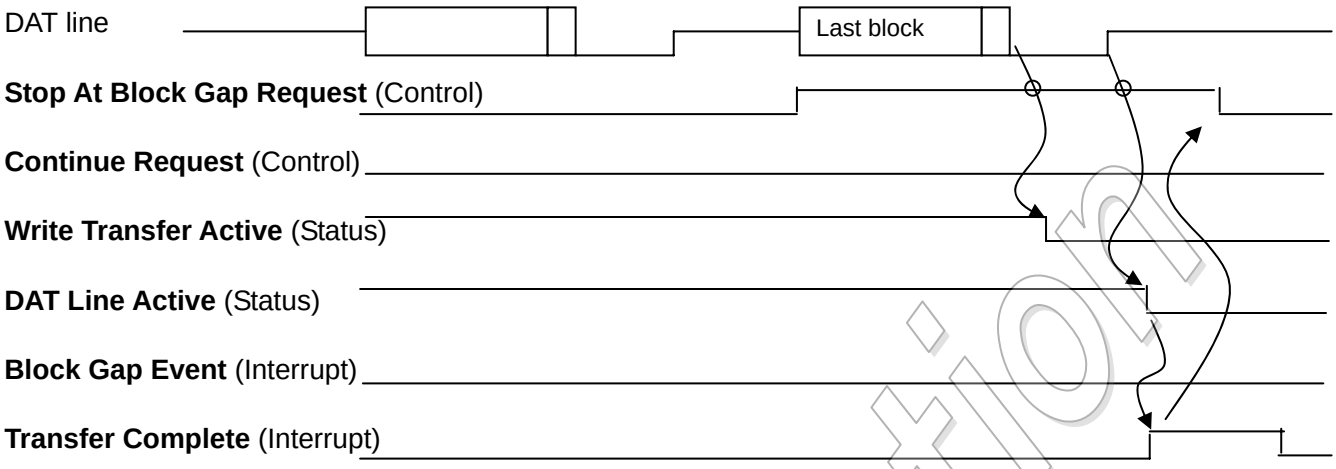


Figure 3-27 : Stop At Block Gap Request is not accepted at the last block of the write transfer

If the **Stop At Block Gap Request** is set to 1 during the last block transfer, the Host Controller shall not accept the **Stop At Block Gap Request** and terminates the transaction normally. The **Block Gap Event** Interrupt is not generated. When the **Transfer Complete** Interrupt is generated, and if the **Block Gap Event** Interrupt Status is not set to 1, the driver shall clear the **Stop At Block Gap Request**.

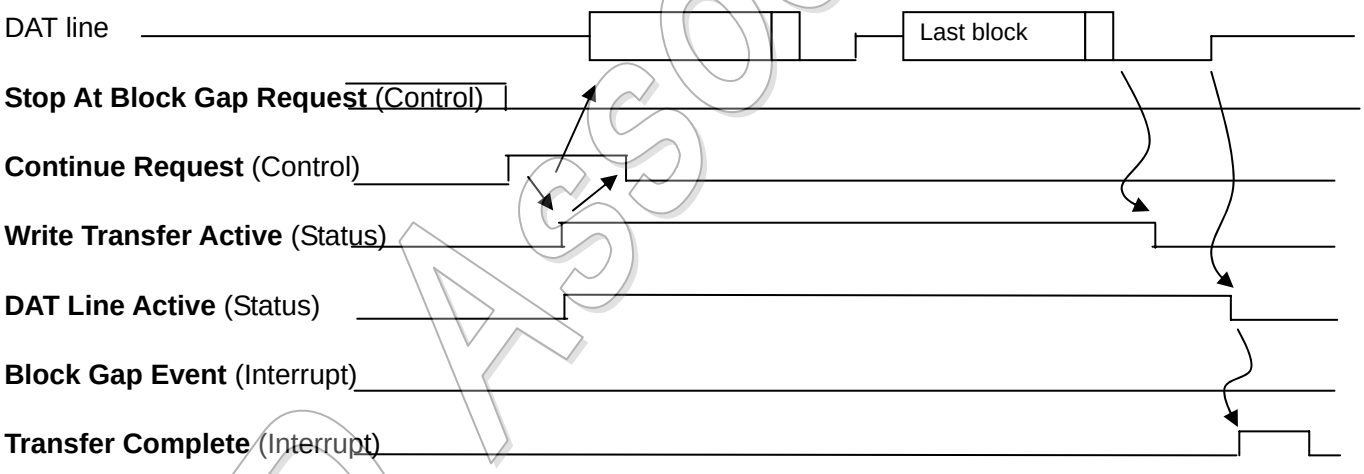


Figure 3-28 : Continue write transfer by Continue Request

To restart a stopped data transfer, set the **Continue Request** to 1. (**Stop At Block Gap Request** shall be set to 0.) After accepting the **Continue Request**:

(1) Set the **DAT Line Active** status and the **Write Transfer Active** Status
(2) The **Continue Request** is automatically cleared by (1).

The end of transfer is specified by data length.

(1) Clear the **Write Transfer Active** Status, and do not generate the **Block Gap Event** Interrupt
(2) After the busy signal is released, clear the **DAT Line Active** status and generates the **Transfer Complete** Interrupt.

Appendix A

As regards PCI bus interface, the Host Driver requires some information in the PCI Configuration registers to identify the SD Host Controller. It is specified in Appendix A of this specification.

A.1 PCI configuration register

A.1.1 Register Map

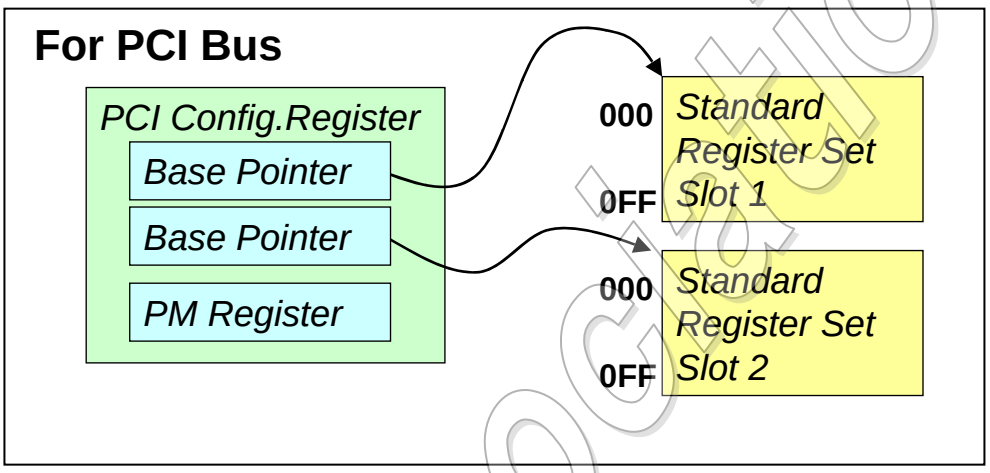


Figure A - 1 : Register Set for PCI Device (Example for 2 slots)

The PCI Configuration register is a special register to support Plug & Play and ACPI power management. The PCI Configuration registers for the PCI Based SD Host Controller is defined as appendix A.

Multiple slots can be supported through the use of multiple Base Addresses within a single PCI Function. Each of these Base Addresses is configured through the *Base Address* registers at offsets 10h to 24h in the PCI Configuration Space Header. The PCI Specification allows for a PCI Function to have up to six Base Addresses. As such, a PCI Based SD Host Controller can support up to a total of six SD Slots.

A PCI Based SD Host Controller shall configure the *Base Address* register of each supported SD Slot such that it is a memory base address with at least 256 bytes allocated. This allows for enough memory address space in each Base Address to access all of the registers defined in this specification. Each set of the SD registers shall be implemented in a separate Base Address.

The values of Power Management register specified in the PCI Configuration registers should refer to the current and consumption values for all slots combined. It shall be read the total amount of power used by the PCI Based SD Host Controller, whether it has a single slot or multiple slots.

A.1.2 SD Controller Configuration Register MAP

31 23	15	07 00	Port
Device ID	Vendor ID		00h
Status	Command		04h
Class Code		Revision ID	08h
Header Type			0Ch
Base Address(es)			10-27h
			28-2Bh
Subsystem Device ID	Subsystem Vendor ID		2Ch
			30h
		Capability Pointer	34h
			38h
	Interrupt Pin	Interrupt Line	3Ch
		Slot Information	40h
			44-7Fh
Power Management Capabilities (PMC)	Next Item Ptr	Capability ID	80h
Data / PMCSR PCI to PCI Bridge Support (PMCSR_BSE)	Power Management Control/Status (PMCSR)		84h
			88-FFh

Table A - 1 : PCI Configuration Register for Standard SD Host Controller

PCI configuration space is divided into 3 areas.

00h - 3Fh : Registers defined in the PCI Bus Interface Specification
40h - 7Fh : Register area reserved for the SD Host Specification
Slot Information assigns to 40h and 41h-7Fh is reserved for future.
80h - FFh : Register area reserved for vendor unique registers

Implementation Note:
The Host Controller should place Power Management registers anywhere in the vendor unique register area. The offset address of Power Management register is set by *Capability Pointer* register. The Table A - 1 shows the case of offset being 80h (registers from 80h to 87h).

A.1.3 PCI Configuration Register

PCI Configuration registers are defined in this section that are specific for the PCI Based SD Host Controller. Refer to PCI Specification Ver2.3 for other standard PCI Configuration registers.

A.1.3.1 Class Code Register (Offset 09h)

D31 D24	D23 D16	D15 D08
Basic Class	Sub Class	Interface Code

Figure A - 2 : PCI Config. Class Code Register

Location	Attrib	
31-24 (0Bh)	RO	**Basic Class** 08h: General Peripheral
23-16 (0Ah)	RO	**Sub Class** 05h: for SD Host Controller
15-08 (09h)	RO	**Interface Code** 00h: Standard Host not supported DMA 01h: Standard Host supported DMA 02h: Vendor unique SD Host Controller

Table A - 2 : PCI Config. Class Code Register

A.1.3.2 Base Address Register (Offset 10h)

Indicates the base address for mapping the SD Controller register set.

D31 D08	D7 D01	D00
Base Address	00 0000	Space Indicator

Figure A - 3 : PCI Config. Base Address Register

Location	Attrib	
31-08	RW	**Base Address** The SD Host Controller register set is mapped on a memory space of 256bytes starting from this base address.
07-01	RO	**Fixed to 00 0000b.**
00	RO	**Space Indicator** Set to 0 if mapped to the memory space.

Table A - 3 : PCI Config. Base Address Register

Implementation Note:
Multiple slot support Host Controller use *Base Address registers* at offsets 10h to 24h in the PCI Configuration register. Format of all *Base Address registers* are the same as this register. Not used *Base Address registers* shall be zero with RO type.

Offset 10h:	Slot1
Offset 14h:	Slot2
Offset 18h:	Slot3
Offset 1Ch:	Slot4
Offset 20h:	Slot5
Offset 24h:	Slot6

A.1.3.3 Slot Information Register (Offset 40h)

D07	D06 D04	D03	D02 D00
Reserved	Number of slots	Reserved	First Base Address Register Number

Figure A - 4 : PCI Config. Slot Information Register

Location	Attrib	
07	Rsvd	**Reserved**
06-04	RO	**Number Of Slots** These statuses indicate the number of slots the Host Controller supports. In the case of single function, maximum 6 slots can be assigned. 000b: 1 slot 001b: 2 slot 010b: 3 slot 011b: 4 slot 100b: 5 slot 101b: 6 slot
03	Rsvd	**Reserved**
02-00	RO	**First Base Address Register Number** Up to 6 Base Address can be specified in single configuration. These bits indicate first *Base Address* register number assigned for SD Host Controller register set. In the case of single function and multiple register sets, contiguous base addresses are used. **Number Of Slot** specifies number of base address. 000b: Base Address 10h (BAR0) 001b: Base Address 14h (BAR1) 010b: Base Address 18h (BAR2) 011b: Base Address 1Ch (BAR3) 100b: Base Address 20h (BAR4) 101b: Base Address 24h (BAR5)

Table A - 4 : PCI Config. Slot Information Register

A.1.4 The relation between Device State, Power and Clock

The Table A - 5 shows Power Management policies when a SD card is inserted.

State	Card Power	SD Clock	Bus Mode	SD Bus Action
D0	On	On	4 or 1 bit *	Any SD transaction or Interrupt
D1	On	On	4 or 1 bit *	Interrupt only
D2	On	Off	1 bit	Interrupt only
D3 hot	On or Off **	Off	1 bit	Interrupt only
D3 cold	On or Off **	Off	1 bit	Interrupt only

Table A - 5 : The relation between Device State, Power and Clock

* 4 bit-mode is recommended if this mode is supported.
** If PME is supported in the D3 state, card power shall be supplied.

The relations between card power supply and Device states are shown below:
In the D0 state, while a SD card is inserted or not rejected, card power shall be supplied by setting the **SD Bus Power** in the *Power Control* register. In the D1 or D2 states, the Host Driver shall keep the preceding power supply state. In the D3 state, if the Host System supports card interrupt wakeup, the card power shall keep on. In all states, when the SD card is removed after the card power is supplied, the Host Controller shall shut off the card power and clear the **SD Bus Power** in *Power Control* register automatically.

The relations between the SD Clock and Device states are shown below:
In the D0 state, while the SD card is inserted or not rejected, the SD Clock shall be supplied by setting the **SD Clock Enable** in the *Clock Control* register. In the D1 state, the Host Driver shall keep the state of the SD Clock while in the D0 state. In the D2 and D3 states, the Host Controller shall stop the SD Clock regardless of the **SD Clock Enable**. If wakeup is supported, the SD Bus mode shall be changed to 1 bit-mode just before transferring from the D0 state. In all states, when the SD card is removed after the card power has been supplied, the Host Controller shall stop the SD Clock and clear the **SD Clock Enable** automatically.

A.1.5 Generate PME interrupt by the Wakeup Events

PME interrupt is generated by rising edge of three interrupt statuses that gated by *Wakeup Event Enable (*Refer to 1.8). Writing 1 to the **PME Status** clears its status.

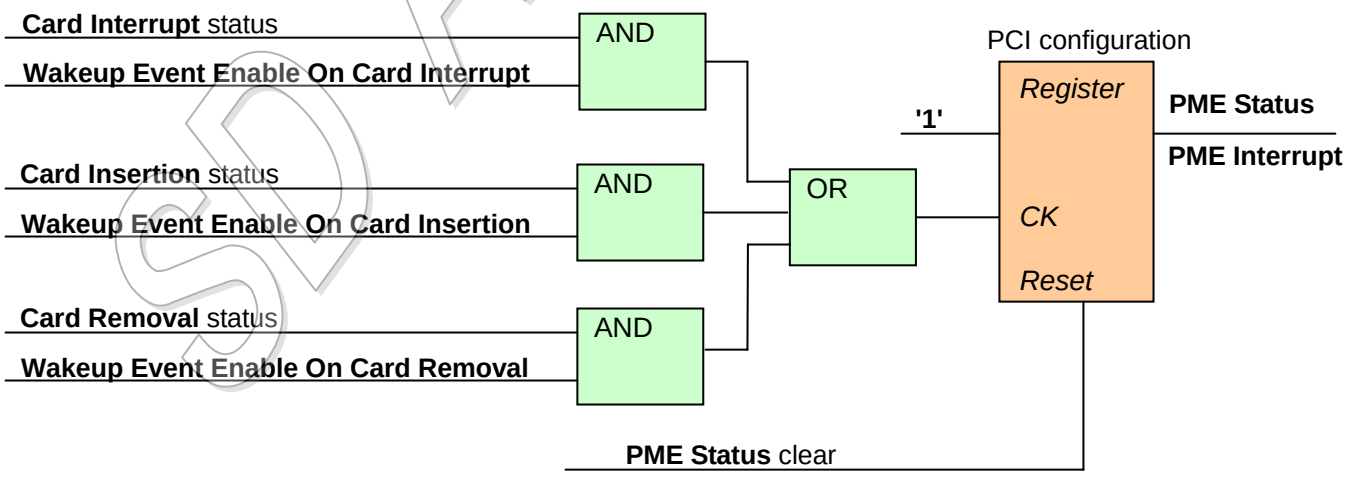


Figure A - 5 : Condition to generate PME Interrupt

Appendix B

B.1 Related Documents

This specification is based on the following documents.

SD Memory Card Specifications
Part 1 Physical Layer Specification — Version 1.01 April 2001

SD Memory Card Specifications
Part 2 File System Specification — Version 1.01 April 2001

SD Memory Card Specifications
Part 3 File Security Specification — Version 1.01 April 2001

SD Card Specifications
Part E1 Secure Digital Input/Output (SDIO) Card Specification — Version 1.0 October 2001

PCI Bus Power Management Interface Specification
Revision 1.1 December 1998

PCI Local Bus Specification
Revision 2.3 March 2002

B.2 Abbreviations and terms

ACPI	Advanced Configuration and Power Interface
API	Application Program Interface
Block Gap	Period between blocks of data
Block	a number of bytes, basic data transfer unit
Busy	Busy signal that SD card drives on DAT[0] line.
CCCR	Card Common Control Register
CDCLK	a clock for detecting SD card
CID	Card Identification number register
Clr	Clear
CMD	command line or SD Bus command (if extended CMDXX)
CMD_wo_DAT	Commands without using DAT line
CRC	Cyclic Redundancy Check
CSD	Card Specific Data register
DAT	data line
DMA	Direct Memory Access
GPS	Global Positioning System
HW	Hardware
Int	Interrupt
LED	Light Emitting Diode
OCR	Operation Conditions Register
OS	Operating System
PCI	Peripheral Component Interconnect
PHS	Personal Handyphone System
PME	Power Management Enable
Resume	Restart suspended function. It is defined in SDIO spec.

RCA	Relative Card Address register
SDCD#	a signal, which is active in a level of low, for detecting SD card
SDCLK	a clock for supplying to SD card
SDWP	a signal, which is active in a level of high, for detecting SD card to be protected writing
Suspend	Stop multiple transaction. It is defined in SDIO spec.
TMCLK	a clock for detecting a timeout on DAT line