



**SD Specifications**  
**Part A2**  
**SD Host Controller**  
**Simplified Specification**

**Version 2.00**

**February 8, 2007**

**Technical Committee**  
**SD Association**

## Revision History

Date	Version	Changes compared to previous issue
April 3, 2004	1.00	The first release
February 8, 2007	2.00	Support Advanced DMA2, Test Registers and 64-bit system bus. Apply changes in the Supplementary Notes Ver1.00 Draft. Some typos are fixed.

## **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 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

<b>1. Overview of the SD Standard Host .....</b>	<b>1</b>
<b>1.1. Scope of the Standard SD Host .....</b>	<b>1</b>
<b>1.2. Register Map .....</b>	<b>2</b>
<b>1.3. Multiple Slot support .....</b>	<b>2</b>
<b>1.4. Supporting DMA.....</b>	<b>3</b>
<b>1.5. SD Command generation .....</b>	<b>3</b>
<b>1.6. Suspend and Resume mechanism.....</b>	<b>4</b>
<b>1.7. Buffer Control.....</b>	<b>5</b>
1.7.1. Control of Buffer Pointer.....	5
1.7.2. Determining Buffer block length .....	7
1.7.3. Dividing large data transfer.....	7
1.7.4. Data lengths which cannot be divided by block size.....	7
<b>1.8. Relationship between Interrupt Control Registers .....</b>	<b>9</b>
<b>1.9. HW Block Diagram and timing part .....</b>	<b>11</b>
<b>1.10. Power state definition of SD Host Controller .....</b>	<b>12</b>
<b>1.11. Auto CMD12.....</b>	<b>13</b>
<b>1.12. Controlling SDCLK .....</b>	<b>14</b>
<b>1.13. Advanced DMA.....</b>	<b>15</b>
1.13.1. Block Diagram of ADMA2.....	15
1.13.2. An Example of ADMA2 Programming.....	16
1.13.3. Data Address and Data Length Requirements .....	16
1.13.4. Descriptor Table .....	17
1.13.5. ADMA2 States .....	18
<b>1.14. Test Registers .....</b>	<b>19</b>
<b>2. SD Host Standard Register.....</b>	<b>20</b>
<b>2.1. Summary of register set.....</b>	<b>20</b>
2.1.1. SD Host Control Register Map .....	20
2.1.2. Configuration Register Types .....	21
2.1.3. Register Initial values .....	21
2.1.4. Reserved bits of register.....	21
<b>2.2. SD Host Standard Register .....</b>	<b>22</b>
2.2.1. SDMA System Address Register (Offset 000h) .....	22
2.2.2. Block Size Register (Offset 004h) .....	23
2.2.3. Block Count Register (Offset 006h).....	25
2.2.4. Argument Register (Offset 008h).....	26
2.2.5. Transfer Mode Register (Offset 00Ch) .....	27
2.2.6. Command Register (Offset 00Eh) .....	29
2.2.7. Response Register (Offset 010h).....	31
2.2.8. Buffer Data Port Register (Offset 020h).....	32
2.2.9. Present State Register (Offset 024h).....	33
2.2.10. Host Control Register (Offset 028h) .....	40

**SD Host Controller Simplified Specification Version 2.00**

2.2.11. Power Control Register (Offset 029h).....	42
2.2.12. Block Gap Control Register (Offset 02Ah).....	43
2.2.13. Wakeup Control Register (Offset 02Bh) .....	45
2.2.14. Clock Control Register (Offset 02Ch) .....	46
2.2.15. Timeout Control Register (Offset 02Eh) .....	48
2.2.16. Software Reset Register (Offset 02Fh).....	49
2.2.17. Normal Interrupt Status Register (Offset 030h) .....	51
2.2.18. Error Interrupt Status Register (Offset 032h) .....	55
2.2.19. Normal Interrupt Status Enable Register (Offset 034h) .....	58
2.2.20. Error Interrupt Status Enable Register (Offset 036h) .....	60
2.2.21. Normal Interrupt Signal Enable Register (Offset 038h) .....	62
2.2.22. Error Interrupt Signal Enable Register (Offset 03Ah) .....	64
2.2.23. Auto CMD12 Error Status Register (Offset 03Ch) .....	66
2.2.24. Capabilities Register (Offset 040h).....	68
2.2.25. Maximum Current Capabilities Register (Offset 048h) .....	71
2.2.26. Slot Interrupt Status Register (Offset 0FCh) .....	72
2.2.27. Host Controller Version Register (Offset 0FEh).....	73
2.2.28. Force Event Register for Error Interrupt Status (Offset 052h).....	73
2.2.29. Force Event Register for Auto CMD12 Error Status (Offset 050h).....	75
2.2.30. ADMA Error Status Register (Offset 054h) .....	76
2.2.31. ADMA System Address Register (Offset 058h) .....	77
<b>3. SEQUENCE .....</b>	<b>78</b>
<b>3.1. SD Card Detection .....</b>	<b>78</b>
<b>3.2. SD Clock Control .....</b>	<b>80</b>
3.2.1. SD Clock Supply Sequence .....	80
3.2.2. SD Clock Stop Sequence .....	81
3.2.3. SD Clock Frequency Change Sequence.....	81
<b>3.3. SD Bus Power Control.....</b>	<b>82</b>
<b>3.4. Changing Bus Width.....</b>	<b>84</b>
<b>3.5. Timeout Setting on DAT Line .....</b>	<b>85</b>
<b>3.6. Card Initialization and Identification .....</b>	<b>86</b>
<b>3.7. SD Transaction Generation .....</b>	<b>90</b>
3.7.1. Transaction Control without Data Transfer Using DAT Line.....	91
3.7.1.1. The sequence to issue a SD Command .....	91
3.7.1.2. The sequence to finalize a command .....	93
3.7.2. Transaction Control with Data Transfer Using DAT Line.....	94
3.7.2.1. Not using DMA.....	95
3.7.2.2. Using SDMA .....	97
3.7.2.3. Using ADMA .....	99
<b>3.8. Abort Transaction .....</b>	<b>101</b>
3.8.1. Asynchronous Abort .....	101
3.8.2. Synchronous Abort.....	102
<b>3.9. Changing Bus Speed Mode Setting .....</b>	<b>103</b>
<b>3.10. Error Recovery .....</b>	<b>105</b>
3.10.1. Error Interrupt Recovery .....	107
3.10.2. Auto CMD12 Error Recovery .....	110
<b>3.11. Wakeup Control (Optional).....</b>	<b>112</b>

**SD Host Controller Simplified Specification Version 2.00**

<b>3.12. Suspend/Resume (Optional)</b> .....	<b>114</b>
3.12.1. Suspend Sequence .....	114
3.12.2. Resume Sequence .....	116
3.12.3. Read transaction wait / continue timing .....	117
3.12.4. Write transaction wait / continue timing .....	119
<b>Appendix A : PCI Configuration Register</b> .....	<b>121</b>
<b>A.1 Register Maps</b> .....	<b>121</b>
<b>A.2 SD Controller Configuration Register MAP</b> .....	<b>122</b>
<b>A.3 PCI Configuration Register</b> .....	<b>123</b>
A.3.1 Class Code Register (Offset 09h) .....	123
A.3.2 Base Address Register (Offset 10h).....	124
A.3.3 Slot Information Register (Offset 40h).....	125
<b>A.4 The relation between Device State, Power and Clock</b> .....	<b>126</b>
<b>A.5 Generate PME interrupt by the Wakeup Events</b> .....	<b>126</b>
<b>Appendix B</b> .....	<b>127</b>
<b>B.1 Related Documents</b> .....	<b>127</b>
<b>B.2 Abbreviations and terms</b> .....	<b>128</b>
<b>Appendix C</b> .....	<b>129</b>
<b>C.1 Advanced DMA1</b> .....	<b>129</b>
<b>C.2 Block Diagram of ADMA</b> .....	<b>129</b>
<b>C.3 An Example of ADMA1 Programming</b> .....	<b>130</b>
<b>C.4 Transfer Length Requirement</b> .....	<b>130</b>
<b>C.5 32-bit Address Descriptor Table</b> .....	<b>131</b>
<b>C.6 ADMA States</b> .....	<b>132</b>
<b>C.7 ADMA1 Control Registers</b> .....	<b>133</b>

# Table of Figures

Figure 1-1 : Host Hardware and Driver Architecture .....	1
Figure 1-2 : Classification of the Standard Register Map.....	2
Figure 1-3 : Register Map for Multiple Slots Controller .....	2
Figure 1-4 : Suspend and Resume mechanism .....	4
Figure 1-5 : Buffer size relation between the SD Host and the SD Card .....	7
Figure 1-6 : Two cases to handle a transfer of data that cannot be divided by block size .....	8
Figure 1-7 : Logical relation for interrupt registers .....	9
Figure 1-8 : Block Diagram of Host Controller .....	11
Figure 1-9 : Block Diagram of ADMA2.....	15
Figure 1-10 : An Example of ADMA2 Data Transfer .....	16
Figure 1-11 : 32-bit Address Descriptor Table.....	17
Figure 1-12 : State Diagram of the ADMA2 .....	18
Figure 2-1 : SDMA System Address Register.....	22
Figure 2-2 : Block Size Register .....	23
Figure 2-3 : Block Count Register .....	25
Figure 2-4 : Argument Register .....	26
Figure 2-5 : Transfer Mode Register.....	27
Figure 2-6 : Command Register .....	29
Figure 2-7 : Response Register.....	31
Figure 2-8 : Buffer Data Port Register .....	32
Figure 2-9 : Present State Register .....	33
Figure 2-10 : Card Detect State.....	34
Figure 2-11 : Timing of Command Inhibit (DAT) and Command Inhibit (CMD) with data transfer.....	39
Figure 2-12 : Timing of Command Inhibit (DAT) for the case of response with busy .....	39
Figure 2-13 : Timing of Command Inhibit (CMD) for the case of no response command .....	39
Figure 2-14 : Host Control Register.....	40
Figure 2-15 : Power Control Register .....	42
Figure 2-16 : Block Gap Control Register.....	43
Figure 2-17 : Wakeup Control Register .....	45
Figure 2-18 : Clock Control Register .....	46
Figure 2-19 : Timeout Control Register.....	48
Figure 2-20 : Software Reset Register .....	49
Figure 2-21 : Normal Interrupt Status Register .....	51
Figure 2-22 : Error Interrupt Status Register.....	55
Figure 2-23 : Normal Interrupt Status Enable Register.....	58
Figure 2-24 : Error Interrupt Status Enable Register .....	60
Figure 2-25 : Normal Interrupt Signal Enable Register.....	62
Figure 2-26 : Error Interrupt Signal Enable Register .....	64
Figure 2-27 : Auto CMD12 Error Status Register.....	66
Figure 2-28 : Capabilities Register .....	68
Figure 2-29 : Maximum Current Capabilities Register.....	71
Figure 2-30 : Slot Interrupt Status Register .....	72
Figure 2-31 : Host Controller Version Register.....	73
Figure 2-32 : Force Event Register for Error Interrupt Status .....	73
Figure 2-33 : Force Event Register for Auto CMD12 Error Status .....	75
Figure 2-34 : ADMA Error Status Register.....	76
Figure 2-35 : ADMA System Address Register .....	77
Figure 3-1 : Double Box Notation .....	78
Figure 3-2 : SD Card Detect Sequence.....	78
Figure 3-3 : SD Clock Supply Sequence .....	80
Figure 3-4 : SD Clock Stop Sequence.....	81



**SD Host Controller Simplified Specification Version 2.00**

Figure 3-5: SD Clock Change Sequence.....	81
Figure 3-6: SD Bus Power Control Sequence .....	82
Figure 3-7: Change Bus Width Sequence .....	84
Figure 3-8: Timeout Setting Sequence .....	85
Figure 3-9 : Card Initialization and Identification.....	87
Figure 3-10: SD Command Issue Sequence .....	91
Figure 3-11: Command Complete Sequence .....	93
Figure 3-12: Transaction Control with Data Transfer Using DAT Line Sequence (Not using DMA) .....	95
Figure 3-13: Transaction Control with Data Transfer Using DAT Line Sequence (Using SDMA).....	97
Figure 3-14: Transaction Control with Data Transfer Using DAT Line Sequence (Using ADMA).....	99
Figure 3-15: Asynchronous Abort Sequence .....	101
Figure 3-16: Synchronous Abort Sequence.....	102
Figure 3-17 : High Speed Mode Setting for Combo Card .....	103
Figure 3-18 : Error Report and Recovery .....	105
Figure 3-19: Return Status of Auto CMD12 Error Recovery .....	106
Figure 3-20: Error Interrupt Recovery Sequence.....	108
Figure 3-21 : Auto CMD12 Error Recovery Sequence.....	110
Figure 3-22: Wakeup Control before Standby Mode.....	112
Figure 3-23: Wakeup from Standby .....	113
Figure 3-24 : The sequence for suspend.....	114
Figure 3-25 : The sequence for resume .....	116
Figure 3-26 : Wait read transfer by Stop At Block Gap Request.....	117
Figure 3-27 : Stop At Block Gap Request is not accepted at the last block of the read transfer .....	118
Figure 3-28 : Continue read transfer by Continue Request .....	118
Figure 3-29 : Wait write transfer by Stop At Block Gap Request .....	119
Figure 3-30 : Stop At Block Gap Request is not accepted at the last block of the write transfer .....	120
Figure 3-31 : Continue write transfer by Continue Request.....	120
Figure A - 1 : Register Set for PCI Device (Example for 2 slots).....	121
Figure A - 2 : PCI Config. Class Code Register .....	123
Figure A - 3 : PCI Config. Base Address Register .....	124
Figure A - 4 : PCI Config. Slot Information Register .....	125
Figure A - 5 : Condition to generate PME Interrupt.....	126
Figure C - 1 : Block Diagram of ADMA .....	129
Figure C - 2 : Example of ADMA Transfer.....	130
Figure C - 3 : Definition of the 32-bit Descriptor Table for ADMA1.....	131
Figure C - 4 : State Diagram of the ADMA1 .....	132

# Table of Tables

Table 1-1 : Registers to Generate SD Command .....	3
Table 1-2 : Relations Between Address and Byte Enable.....	5
Table 1-3 : Available Byte Enable Pattern for Buffer Data Port Register.....	6
Table 1-4 : Interrupt Signal Table .....	10
Table 1-5 : Wakeup Signal Table .....	10
Table 1-6 : Summary of Register Status for data transfer .....	11
Table 1-7 : Power State Definition .....	12
Table 1-8 : Relation between Auto CMD12 and CMD_wo_DAT .....	13
Table 1-9 : Controlling SDCLK by the SD Bus Power and SD Clock Enable.....	14
Table 1-10 : ADMA2 Length Field.....	17
Table 1-11 : 64-bit Address Descriptor Table .....	17
Table 1-12 : ADMA2 States.....	18
Table 2-1 : SD Host Controller Register Map.....	20
Table 2-2 : Register (and Register Bit-Field) Types .....	21
Table 2-3 : SDMA System Address Register .....	22
Table 2-4 : Block Size Register.....	24
Table 2-5 : Block Count Register .....	25
Table 2-6 : Argument Register .....	26
Table 2-7 : Transfer Mode Register .....	28
Table 2-8 : Determination of Transfer Type.....	28
Table 2-9 : Command Register.....	30
Table 2-10 : Relation Between Parameters and the Name of Response Type .....	30
Table 2-11 : Response Register .....	31
Table 2-12 : Response Bit Definition for Each Response Type. ....	31
Table 2-13 : Buffer Data Port Register.....	32
Table 2-14 : Present State Register (Part 1) .....	34
Table 2-15 : Present State Register (Part 2) .....	37
Table 2-16 : Host Control Register .....	41
Table 2-17 : Power Control Register.....	42
Table 2-18 : Block Gap Control Register .....	44
Table 2-19 : Wakeup Control Register.....	45
Table 2-20 : Clock Control Register.....	47
Table 2-21 : Timeout Control Register .....	48
Table 2-22 : Software Reset Register.....	50
Table 2-23 : Normal Interrupt Status Register.....	54
Table 2-24 : Error Interrupt Status Register .....	57
Table 2-25 : The relation between Command CRC Error and Command Timeout Error .....	57
Table 2-26 : Normal Interrupt Status Enable Register .....	59
Table 2-27 : Error Interrupt Status Enable Register .....	61
Table 2-28 : Normal Interrupt Signal Enable Register .....	63
Table 2-29 : Error Interrupt Signal Enable Register .....	65
Table 2-30 : Auto CMD12 Error Status Register .....	67
Table 2-31 : The relation between CRC Error and Timeout Error for Auto CMD12.....	67
Table 2-32 : Capabilities Register (Part 1).....	68
Table 2-33 : Capabilities Register (Part 2).....	70
Table 2-34 : Maximum Current Capabilities Register.....	71
Table 2-35 : Maximum Current Value Definition .....	71
Table 2-36 : Slot Interrupt Status Register.....	72
Table 2-37 : Host Controller Version .....	73
Table 2-38 : Force Event for Error Interrupt Status Register.....	74
Table 2-39 : Force Event Register for Auto CMD12 Error Status.....	75

**SD Host Controller Simplified Specification Version 2.00**

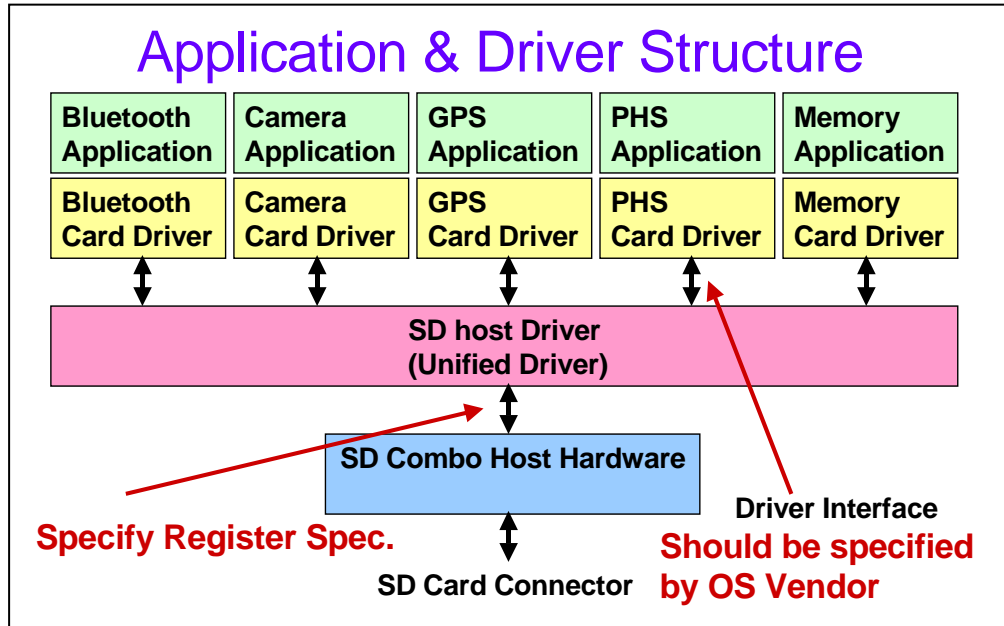
---

Table 2-40 : ADMA Error Status Register.....	76
Table 2-41 : ADMA System Address Register.....	77
Table 3-1 Suspend / Resume Condition .....	115
Table A - 1 : PCI Configuration Register for Standard SD Host Controller.....	122
Table A - 2 : PCI Config. Class Code Register.....	123
Table A - 3 : PCI Config. Base Address Register .....	124
Table A - 4 : PCI Config. Slot Information Register .....	125
Table A - 5 : The relation between Device State, Power and Clock .....	126
Table C - 1 : Symbol Definition used in Descriptor Table of ADMA1 .....	131
Table C - 2 : ADMA1 States .....	132
Table C - 3 : ADMA1 Select in Host Control Register .....	133
Table C - 4 : ADMA1 Support Bit in Capabilities register .....	133

# 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



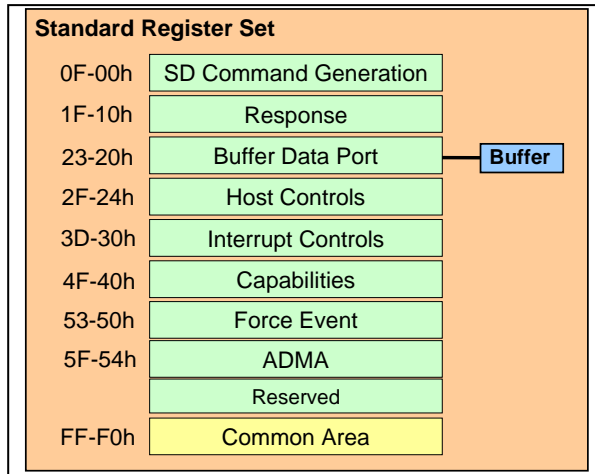
**Figure 1-1 : Host Hardware and Driver Architecture**

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



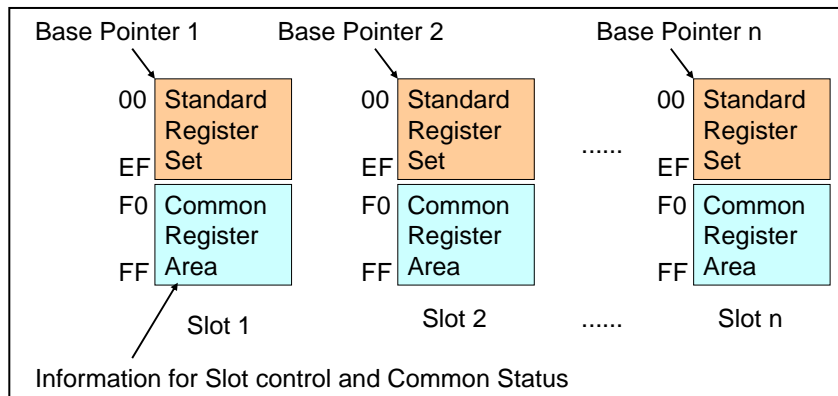
**Figure 1-2 : Classification of the Standard Register Map**

The standard register map is classified in 9 parts listed below. The Host Controller shall support byte, word and double word accesses to these registers. 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) Buffer 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) Force Event : Test register to generate events by software.
- (8) ADMA : Advanced DMA registers.
- (9) 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 Slots 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 control each slot independently, 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. The DMA algorithm defined in the SD Host Controller Standard Specification Version 1.00 is called SDMA (Single Operation DMA). Only one SD command transaction can be executed per a SDMA operation. Support of SDMA can be checked by the **SDMA Support** in the *Capabilities* register.

This specification defines a new DMA transfer algorithm called ADMA (Advanced DMA). ADMA provides data transfer between system memory and SD card without interruption of CPU execution. Support of ADMA can be checked by the *Capabilities* register. Refer to Section 1.13 for more details about ADMA. When the term "DMA" is used in this document, it applies to both SDMA and ADMA.

Prior to using DMA, the Host Driver shall confirm that both the Host Controller and the system bus support it (PCI bus can support DMA). DMA shall support both single block and multiple-block transfers but does not support infinite transfers. Host Controller registers shall remain accessible for issuing non-DAT line commands during a DMA transfer execution. The result of a DMA transfer shall be the same regardless of the system bus data transfer method.

The Host Driver can stop and restart a DMA operation by the control bits in the *Block Gap Control* register. By setting **Stop At Block Gap Request**, a DMA operation can be stopped at block gap. By setting **Continue Request**, DMA operation can be restarted. Refer to the *Block Gap Control* register for more details. If an error occurs, DMA operation shall be stopped. To abort DMA transfer, Host driver shall reset the Host Controller by the **Software Reset For DAT Line** in the *Software Reset* register and issue CMD12 if multiple-block read / write command is executing.

## 1.5. SD Command generation

	DMA command	CPU data transfer	Non-DAT transfer
<b>SDMA System Address</b>	Yes	No	No
<b>Block Size</b>	Yes	Yes	No (Protected)
<b>Block Count</b>	Yes	Yes	No (Protected)
<b>Argument</b>	Yes	Yes	Yes
<b>Transfer Mode</b>	Yes	Yes	No (Protected)
<b>Command</b>	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 *SDMA 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 *SDMA 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 *SDMA System Address*, *Argument* and *Command* registers.

## 1.6. Suspend and Resume mechanism

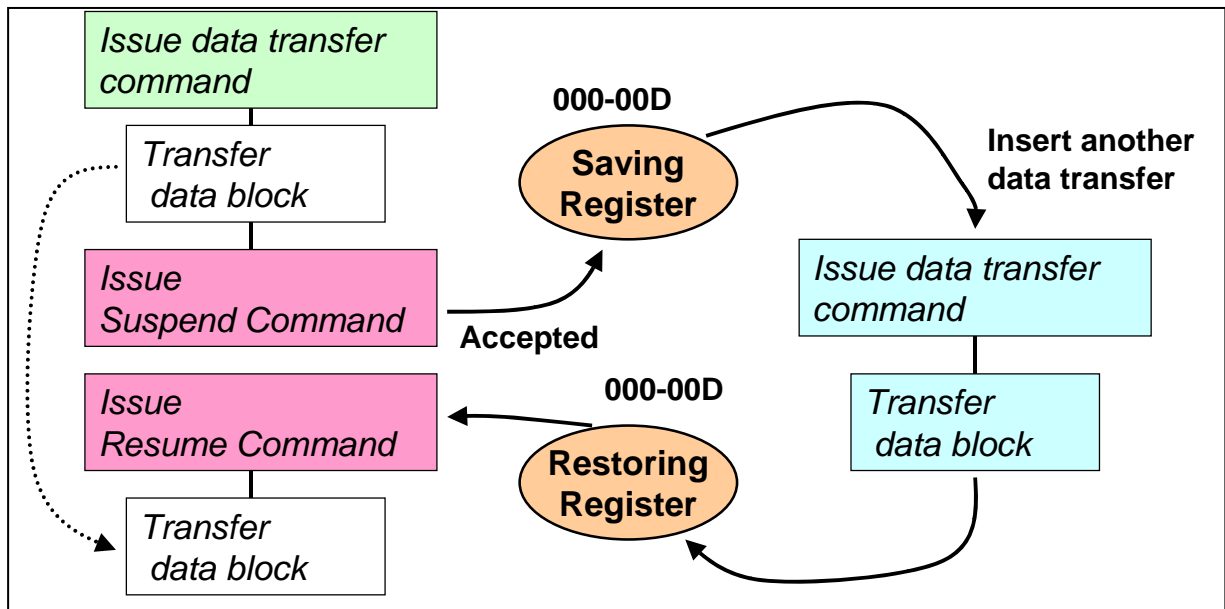


Figure 1-4 : Suspend and Resume mechanism

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 saves information in the first 14bytes registers (that is, offsets 000h-00dh) before issuing other SD commands. On resuming, the Host Driver restores these registers and then issues the Resume command to continue suspended operation.

The SDIO 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: To use Suspend and Resume function, it is necessary that SDIO Card supports the Suspend and Resume commands and Read Wait control.

## 1.7. Buffer Control

The Host Controller has a data buffer for data transfer. The Host Driver accesses internal buffer through the 32-bit *Buffer Data Port* register. Followings show some rules to access the buffer.

### 1.7.1. Control of Buffer Pointer

Internally, the Host Controller maintains a pointer to control the data buffer. The pointer is not directly accessible by the Host Driver. Every time the *Buffer Data Port* register is accessed, the pointer is incremented depending on amount of data written to the buffer. In order to accommodate a variety of system busses, this pointer shall be implemented regardless of system bus width (8-bit, 16-bit, 32-bit or 64-bit system bus width can be supported). To specify control of the pointer, the Host Controller data buffer interface shall have the following characteristics:

(1) System Bus Width and Byte Enable Address

8-bit, 16-bit, 32-bit or 64-bit system bus is supported. To specify byte position for *Buffer Data Port* register (4 bytes), Byte Enable (**BE**[*i*]) or Lower Address (**A**[*i*]) is used. Table 1-2 shows the relation between lower address and byte enable depending on system bus width. The *Buffer Data Port* register can be accessed by **BE**[3:0] for 64-bit system bus which has **BE**[7:0].

System Bus	A[02]	A[01]	A[00]	BE[3] D[31:24]	BE[2] D[23:16]	BE[1] D[15:08]	BE[0] D[07:00]
64-bit	No	No	No	Yes	Yes	Yes	Yes
32-bit	Yes	No	No	Yes	Yes	Yes	Yes
16-bit	Yes	Yes	No	No	No	Yes	Yes
8-bit	Yes	Yes	Yes	No	No	No	Yes <sup>*2</sup>

\*1 "Yes" means the signal is used for control and "No" means the signal is not used.

\*2 : BE[00] for 8-bit bus is always 1 therefore it may not be defined.

**Table 1-2 : Relations Between Address and Byte Enable**

(2) Sequential and continuous access

The *Buffer Data Port* register shall be accessed by sequential and continuous manner. The buffer pointer is controlled by the Byte Enable patterns when accessing to the Buffer Data Port register. So Byte Enable patterns shall be sequential and continuous as well. The order of Byte Enable is according to little endian format. For example, **BE**[1] is accessed, next access shall start from **BE**[2]. Random or skipped access is not allowed.

Table 1-3 shows possible byte enable patterns that the Host Controller shall be supported. However, if the system controller supports write merge, it may generate the other byte enable patterns. To avoid generating unsupported byte enable patterns for the 32-bit or 64-bit bus system, the Host Driver is allowed to use word or double word access to the *Buffer Data Port* register except for the last access to every block data.

OK	BE[3:0]=0011b (2-byte) => BE[3:0]=1100b (2-byte) => BE[3:0]=0011b (2-byte)
OK	BE[3:0]=1100b (2-byte) => BE[3:0]=1111b (4-byte) => BE[3:0]=0011b (2-byte)
OK	BE[3:0]=1111b (4-byte) => BE[3:0]=1111b (4-byte) => BE[3:0]=1111b (4-byte)
Not OK	BE[3:0]=0011b (2-byte) => BE[3:0]=0011b (2-byte) (Cannot skip BE[2],BE[3])
Not OK	BE[3:0]=0011b (2-byte) => BE[3:0]=1111b (4-byte) (Cannot skip BE[2],BE[3])



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 means BE is valid and 0 means BE is not valid.

**Table 1-3 : Available Byte Enable Pattern for Buffer Data Port Register**

(3) Buffer Control with Block Size

The buffer preserves data up to the block size specified by the *Block Size* register. Following definitions of controlling buffer enable the Host Driver to access the *Buffer Data Port* register repeatedly with 32-bit width regardless of block size.

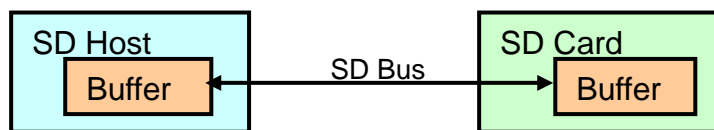
In case of write operation, the buffer accumulates the data written through the *Buffer Data Port* register. When the buffer pointer reaches the block size, **Buffer Write Enable** in the *Present State* register changes 1 to 0. It means no more data can be written to the buffer. Excess data of the last write is ignored. For example, if just lower 2 bytes data can be written to the buffer and a 32-bit (4-byte) block of data is written to the *Buffer Data Port* register, the lower 2 bytes of data is written to the buffer and the upper 2 bytes is ignored. Every time **Buffer Write Enable** changes 0 to 1, it means a next block of data can be written to the buffer. A new blocks write shall always start from BE[00] position. After that, a block of data can be written to the buffer without checking **Buffer Write Enable**.

In case of read operation, every time **Buffer Read Enable** in the *Present State* register changes 0 to 1, a block of data can be read through the *Buffer Data Port* register. A new block read shall always start from BE[00] position. After that a block of data can be read from the buffer without checking **Buffer Read Enable**. Excess data of the last read is ignored. For example, if just lower 2 bytes of data are left in the buffer and a 32-bit (4-byte) read is performed, the lower 2 bytes is valid but the upper 2 bytes is undefined. When the buffer pointer reaches block size, **Buffer Read Enable** changes 1 to 0. It means no more data can be read from the buffer.

**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. According to the feature (3), the Host Driver can always access *Buffer Data Port* register with double-word access.

### 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 as maximum block length in the Card Specific Data (CSD) register for memory cards (for cards compliant with the Physical Layer Specification, READ\_BL\_LEN and WRITE\_BL\_LEN shall be the same) and in the Card Information Structure (CIS) for SDIO cards. Physical Layer Specification re-defines that maximum block length is only used to calculate capacity of memory card. Even though it indicates larger than 512 bytes, block length shall be set to 512 byte for data transfer. This is because 512 bytes block length is required to keep compatibility with 512 bytes data boundary. If the SDIO 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 SDIO 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.

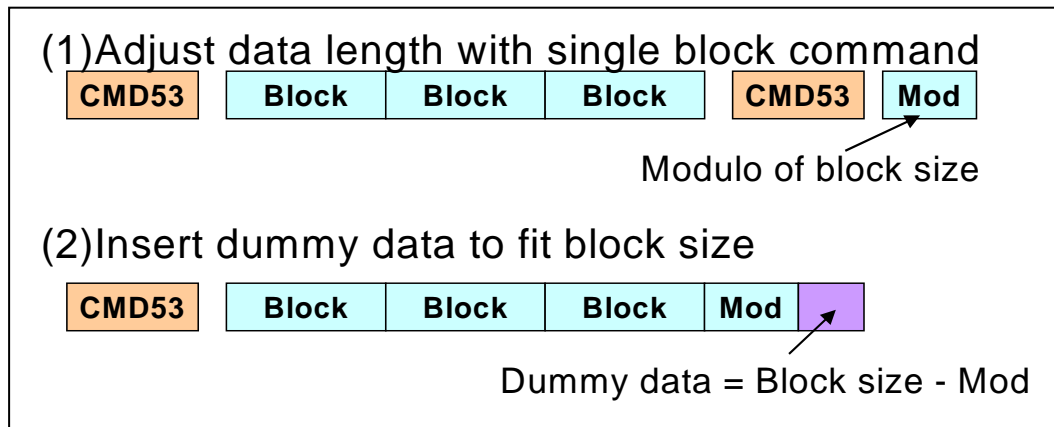


Figure 1-6 : Two cases to handle a transfer of data that cannot be divided by block size

## 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 the *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.

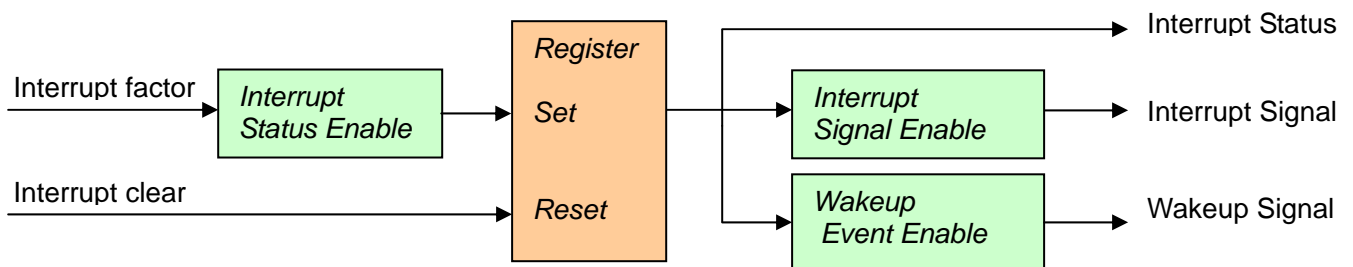


Figure 1-7 : Logical relation for interrupt registers

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

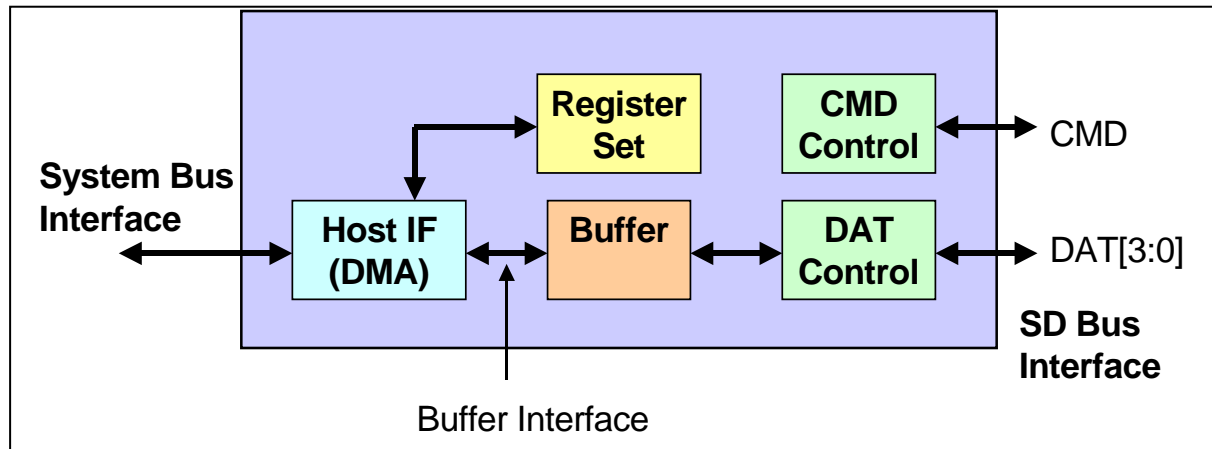


Figure 1-8 : Block Diagram of Host Controller

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 / writes transfer for both DMA and non DMA transfers. However, the timing is different between reads and writes. Read transfers shall be completed after all valid data have been transferred to the Host System and are ready for the Host Driver to access. Write transfers shall be completed after all valid data have 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 Interrupt	Complete Interrupt
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 be supplied when card power is OFF.

States described in Table 1-7 can be determined by reading the corresponding register bits:

Internal clock oscillate/stop	<b>Internal Clock Enable</b> in the <i>Clock Control</i> register
SD Card : Exist/Not exist	<b>Card Inserted</b> in the <i>Present State</i> register
SD Power : ON/OFF	<b>SD Bus Power</b> in the <i>Power Control</i> register
SD Clock : oscillate/stop	<b>SD Clock Enable</b> in the <i>Clock Control</i> register
SD Bus : access/wait (idle)	<b>Command Inhibit (CMD)</b> and <b>Command Inhibit (DAT)</b> in the <i>Present State</i> 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 do not 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 is failed, therefore Auto CMD12 could not be issued.
Auto CMD12 before CMD_wo_DAT	CMD_wo_DAT : Not executed Auto CMD12 : Error	Auto CMD12 is 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 3.10.1 and 3.10.2.



## 1.12. Controlling SDCLK

Table 1-9 shows how **SDCLK** is controlled by the **SD Bus Power** in the *Power Control* register and the **SD Clock Enable** in the *Clock Control* register.

The Clock Period of **SDCLK** is specified by the **SDCLK Frequency Select** in the *Clock Control* register and the **Base Clock Frequency For SD Clock** in the *Capabilities* register. Because of the SD card may use both clock edges, the duty of SD clock should be average 50% (scattering within 45-55%) and the Period of High should be half of the Clock Period. The oscillation of **SDCLK** starts from driving specified Period of High. When **SDCLK** is stopped by the **SD Clock Enable**, the Host Controller shall stop **SDCLK** after driving Period of High to maintain clock duty. When **SDCLK** is stopped by the **SD Bus Power**, the Host Controller shall stop **SDCLK** immediately (drive Low).

SD Bus Power (Note 1)	SD Clock Enable (Note 2)	State of SDCLK
Change 0 to 1	0	Drive Low
	1	Start Clock with specified Period of High
Change 1 to 0	0	Drive Low
	1	Drive Low immediately
0	Don't Care	Drive Low
1	Change 0 to 1	Start Clock with specified Period of High
	Change 1 to 0	Maintains Period of High and then stops Clock and drive Low

**Table 1-9 : Controlling SDCLK by the SD Bus Power and SD Clock Enable**

Note 1: When the card state is changed from Debouncing to No Card, the Host Controller shall clear the **SD Bus Power**.

Note 2: When the the card state is changed from Card Inserted to Debouncing, the Host Controller shall clear the **SD Clock Enable** immediately.

## 1.13. Advanced DMA

In the SD Host Controller Standard Specification Version 2.00, new DMA transfer algorithm called ADMA (Advanced DMA) is defined. The DMA algorithm defined in the SD Host Controller Standard Specification Version 1.00 is called SDMA (Single Operation DMA). SDMA had disadvantage that **DMA Interrupt** generated at every page boundary disturbs CPU to reprogram the new system address. This SDMA algorithm forms a performance bottleneck by interruption at every page boundary. ADMA adopts scatter gather DMA algorithm so that higher data transfer speed is available. The Host Driver can program a list of data transfers between system memory and SD card to the Descriptor Table before executing ADMA. It enables ADMA to operate without interrupting the Host Driver. Furthermore, ADMA can support not only 32-bit system memory addressing but also 64-bit system memory addressing. The 32-bit system memory addressing uses lower 32-bit field of 64-bit address registers. Support of SDMA and ADMA are optional for the Host Controller.

There are two types of ADMA; ADMA1 and ADMA2. ADMA1 can support data transfer of only 4KB aligned data in system memory. ADMA2 improves the restriction so that data of any location and any size can be transferred in system memory. The format of Descriptor Table is different between them. The Host Controller Specification Ver2.00 defines ADMA2 as standard ADMA and recommends supporting ADMA2 rather than ADMA1. This is because the ADMA1 specification is described in Appendix C. When the term "ADMA" is used in this document, it applies to both ADMA1 and ADMA2.

### 1.13.1. Block Diagram of ADMA2

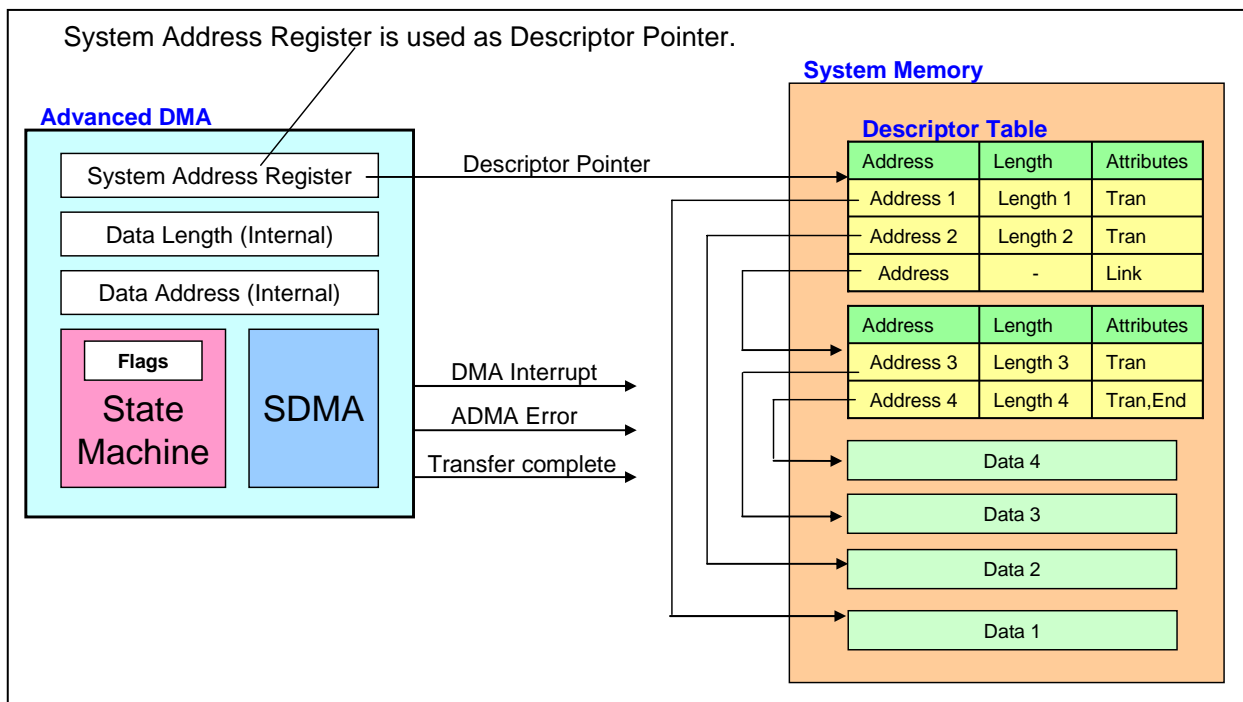
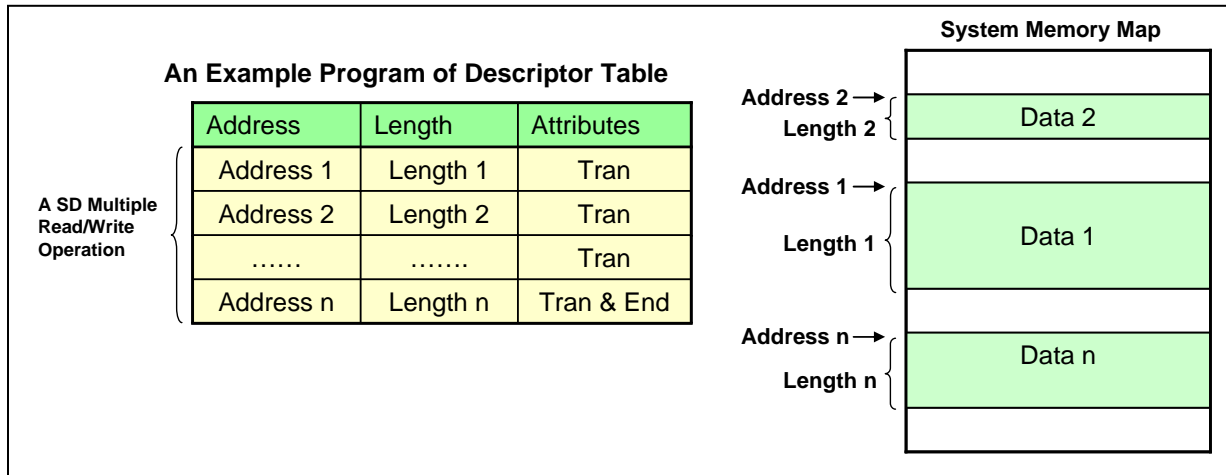


Figure 1-9 : Block Diagram of ADMA2

Figure 1-9 shows block diagram of ADMA2. The Descriptor Table is created in system memory by the Host Driver. 32-bit Address Descriptor Table is used for the system with 32-bit addressing and 64-bit Address Descriptor Table is used for the system with 64-bit addressing. Each descriptor line (one executable unit) consists with address, length and attribute field. The attribute specifies operation of the descriptor line. ADMA2 includes SDMA, State Machine and Registers circuits. ADMA2 does not use 32-bit *SDMA System Address Register* (offset 0) but uses the 64-bit *Advanced DMA System Address*

**SD Host Controller Simplified Specification Version 2.00**

register (offset 058h) for descriptor pointer. Writing Command register triggers off ADMA2 transfer. ADMA2 fetches one descriptor line and execute it. This procedure is repeated until end of descriptor is found (End=1 in attribute).

**1.13.2. An Example of ADMA2 Programming**

**Figure 1-10 : An Example of ADMA2 Data Transfer**

Figure 1-10 shows a typical ADMA2 descriptor program. The data area is sliced in various lengths and each slice is placed somewhere in system memory. The Host Driver describes the Descriptor Table with set of address, length and attributes. Each sliced data is transferred in turns as programmed in descriptor.

**1.13.3. Data Address and Data Length Requirements**

There are 3 requirements to program descriptor.

The minimum unit of address is 4byte.

The maximum data length of each descriptor line is less than 64KB.

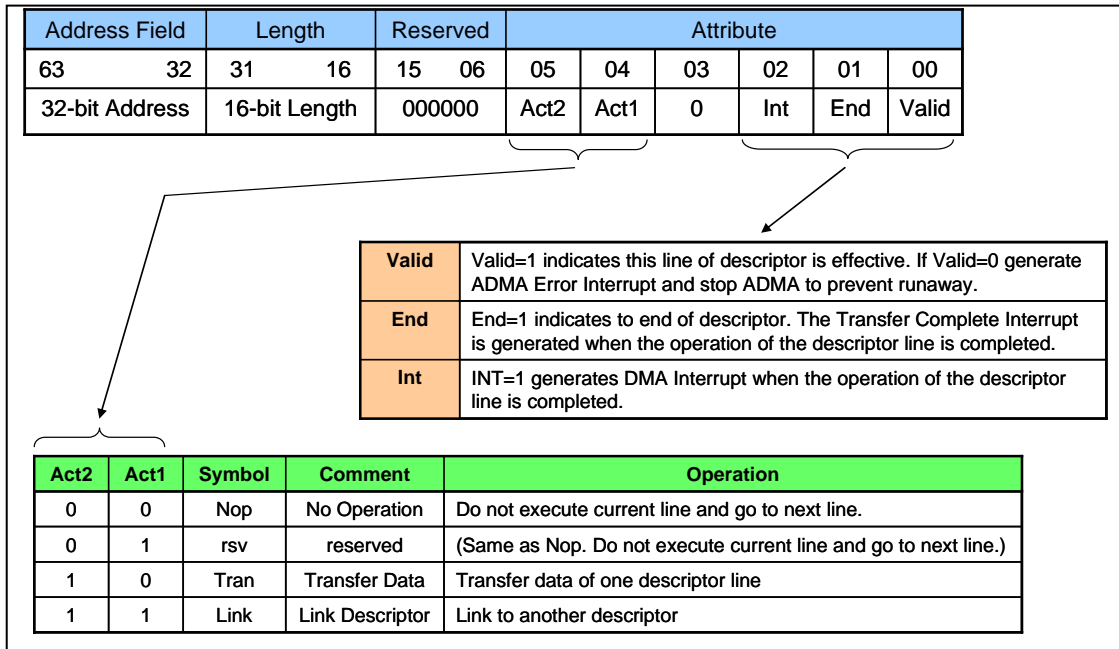
Total Length = Length 1 + Length 2 + Length 3 + ... + Length n  
= multiple of Block Size

If total length of a descriptor were not multiple of block size, ADMA2 transfer might not be terminated. In this case, the transfer should be aborted by data timeout.

*Block Count* register is defined as 16-bit register and it limits the maximum of 65535 blocks transfer. If ADMA2 operation is less than or equal 65535 blocks transfer, *Block Count* register can be used. In this case, total length of Descriptor Table shall be equivalent to multiply block size and block count. If ADMA2 operation is more than 65535 blocks transfer, *Block Count* register shall be disabled by setting 0 to **Block Count Enable** in the *Transfer Mode Register*. In this case, length of data transfer is not designated by block count but Descriptor Table. Therefore, the timing of detecting the last block on SD bus may be different and it affects the control of **Read Transfer Active**, **Write Transfer Active** and **DAT line Active** in the *Present State* register. In case of read operation, several blocks may be read more than required. The Host Driver shall ignore out of range error if the read operation is for the last block of memory area.

**SD Host Controller Simplified Specification Version 2.00**

**1.13.4. Descriptor Table**



**Figure 1-11 : 32-bit Address Descriptor Table**

Figure 1-11 shows the definition of 32-bit Address Descriptor Table. One descriptor line consumes 64-bit (8-byte) memory space. Attribute is used to control descriptor. 3 action symbols are specified. "Nop" operation skips current descriptor line and fetches next one. "Tran" operation transfers data designated by address and length field. "Link" operation is used to connect separated two descriptors. The address field of link points to next Descriptor Table. The combination of Act2=0 and Act1=1 is reserved and defined the same operation as Nop. A future version of controller may use this field and redefine a new operation. 32-bit address is stored in the lower 32-bit of 64-bit address registers. Address field shall be set on 32-bit boundary (Lower 2-bit is always set to 0) for 32-bit address descriptor table. Table 1-10 shows the definition of length field in the Descriptor Table.

Length Field	Value of Length
0000h	65536 bytes
0001h	1 byte
0002h	2 bytes
.....	.....
FFFFh	65535 bytes

**Table 1-10 : ADMA2 Length Field**

Address Field		Length		Reserved		Attribute					
95	32	31	16	15	06	05	04	03	02	01	00
64-bit Address		16-bit Length		000000		Act2	Act1	0	Int	End	Valid

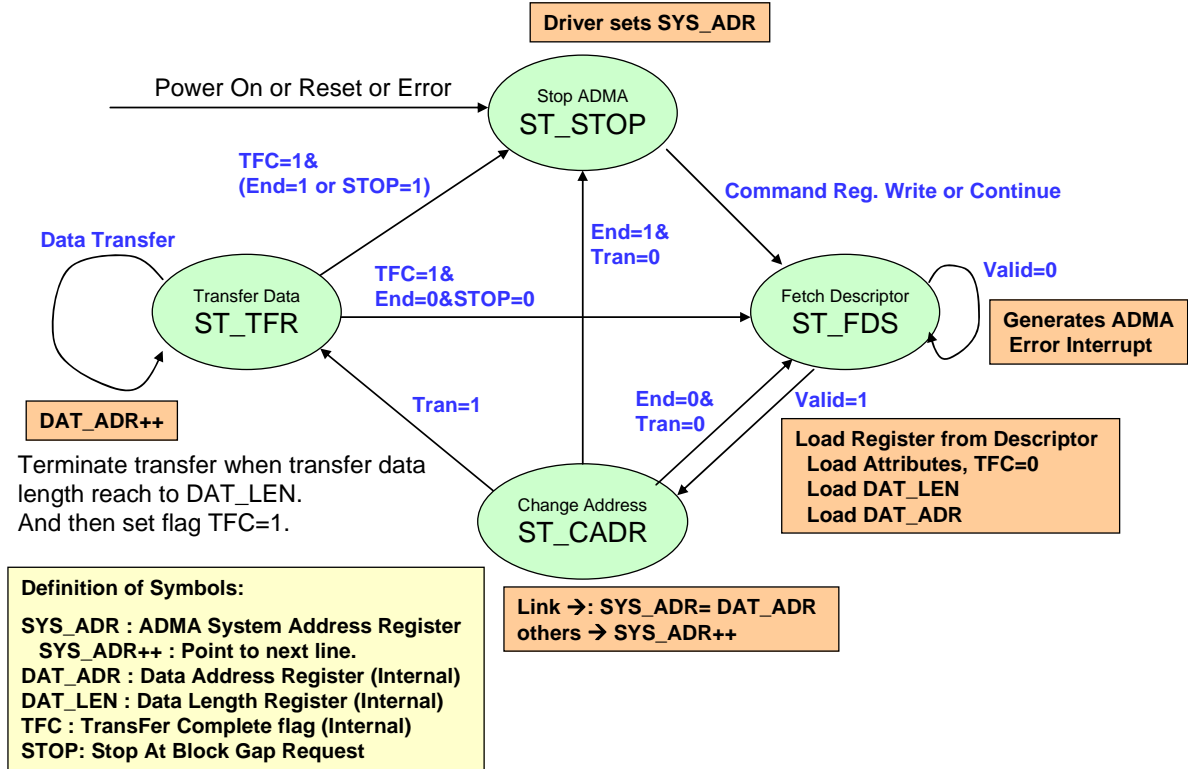
**Table 1-11 : 64-bit Address Descriptor Table**

Table 1-11 shows the 64-bit Address Descriptor Table. Address field is extended to 64-bit. One descriptor line consumes 96-bit (12-byte) memory space. Other functions are equivalent to 32-bit Address Descriptor. Address field shall be set on 64-bit boundary (Lower 3-bit is always set to 0) for 64-bit address descriptor table.

**SD Host Controller Simplified Specification Version 2.00**

**1.13.5. ADMA2 States**

Figure 1-12 shows state diagram of ADMA2. 4 states are defined; Fetch Descriptor state, Change Address state, Transfer Data state, and Stop ADMA state. Operation of each state is explained in Table 1-12.



**Figure 1-12 : State Diagram of the ADMA2**

State Name	Operation
ST_FDS (Fetch Descriptor)	ADMA2 fetches a descriptor line and set parameters in internal registers. Next go to ST_CADR state.
ST_CADR (Change Address)	Link operation loads another Descriptor address to ADMA System Address register. In other operations, ADMA System Address register is incremented to point next descriptor line. If End=0, go to ST_TFR state. ADMA2 shall not be stopped at this state even if some errors occur.
ST_TFR (Transfer Data)	Data transfer of one descriptor line is executed between system memory and SD card. If data transfer continues (End=0) go to ST_FDS state. If data transfer completes, go to ST_STOP state.
ST_STOP (Stop DMA)	ADMA2 stays in this state in following cases: (1) After Power on reset or software reset. (2) All descriptor data transfers are completed If a new ADMA2 operation is started by writing Command register, go to ST_FDS state.

**Table 1-12 : ADMA2 States**

ADMA2 does not support suspend / resume function but stop and continue are available. When the **Stop At Block Gap Request** in the *Block Gap Control* register is set during the ADMA2 operation, the **Block Gap Event Interrupt** is generated when ADMA2 is stopped at block gap. The Host Controller shall stop ADMA2 read operation by using Read Wait or stopping SD Clock. While stopping ADMA2, any SD commands cannot be issued. (In case of Host Controller version 1.00, the **Stop At Block Gap Request** can be set only when the card supports the Read Wait.)

Error occurrence during ADMA2 transfer may stop ADMA2 operation and generates **ADMA Error Interrupt**. The **ADMA Error State** field in the *ADMA Error Status* register holds state of ADMA2 stopped. The host driver can identify error descriptor location by following method. If ADMA stopped at ST\_FDS state, the *ADMA System Address Register* points the error descriptor line. If ADMA stopped at ST\_TFR or ST\_STOP state, the *ADMA System Address Register* points the next location of error descriptor line. By this reason, ADMA2 shall not stop at ST\_CADR state.

## 1.14. Test Registers

New test registers is defined for the test purpose. If it would be difficult to generate some interrupts intentionally, this feature would be useful to generate these interrupts manually for driver debugging. The *Force Event* register to control the *Error Interrupt Status* and *Auto CMD12 Error Status* are defined for this purpose. Intentional control of card insertion and removal are also difficult. The **Card Detect Signal Selection** and **Card Detect Test Level** in the *Host Control* register enable manual control of **Card Inserted** in the *Present State* register and generating interrupt of **Card Insertion** and **Card Removal** in the *Normal Interrupt Status* register. Support of the test registers is mandatory.

## 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	SDMA System Address (High)		000h	SDMA 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)	
052h	Force Event for Error Interrupt Status		050h	Force Event for Auto CMD12 Error Status	
	---		054h	---	ADMA Error Status
05Ah	ADMA System Address [31:16]		058h	ADMA System Address [15:00]	
05Eh	ADMA System Address [63:48]		05Ch	ADMA System Address [47:32]	
---	---		---	---	
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.
Hwlnit	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.
WO	Write-only register. It is not physically implemented register. Rather, it is an address at which registers can be written.

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

Implementation Note: If the Host Driver writes to RO, ROC, Hwlnit 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.

Value of the Capabilities register and Maximum Current Capabilities register depends on the Host Controller. Value of the Host Controller Version register depends on the Host Controller.

### 2.1.4. Reserved bits of register

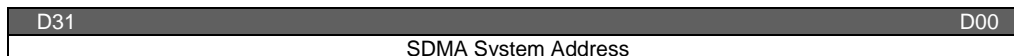
"Reserved" means the bit can be defined for future use and is currently set to 0. These bits should be written as zero.



## 2.2. SD Host Standard Register

### 2.2.1. SDMA System Address Register (Offset 000h)

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



**Figure 2-1 : SDMA System Address Register**

Location	Attrib	Register Field Explanation
31-00	RW	<p><b>SDMA System Address</b></p> <p>This register contains the system memory address for a SDMA transfer. When the Host Controller stops a SDMA 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.</p> <p>The Host Driver shall initialize this register before starting a SDMA transaction. After SDMA has stopped, the next system address of the next contiguous data position can be read from this register.</p> <p>The SDMA transfer waits at the every boundary specified by the <b>Host SDMA Buffer Boundary</b> in the <i>Block Size</i> register. The Host Controller generates <b>DMA Interrupt</b> to request the Host Driver to update this register. The Host Driver sets 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 restarts the SDMA transfer.</p> <p>When restarting SDMA by the Resume command or by setting <b>Continue Request</b> in the <i>Block Gap Control</i> register, the Host Controller shall start at the next contiguous address stored here in the <i>SDMA System Address</i> register.</p> <p>ADMA does not use this register.</p>

**Table 2-3 : SDMA 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 SDMA Buffer Boundary		Transfer Block Size	

**Figure 2-2 : Block Size Register**

Location	Attrib	Register Field Explanation																								
15	Rsvd	<b>Reserved</b>																								
14-12	RW	<p><b>Host SDMA Buffer Boundary</b></p> <p>The large contiguous memory space may not be available in the virtual memory system. To perform long SDMA transfer, <i>SDMA System Address</i> register shall be updated at every system memory boundary during SDMA transfer. These bits specify the size of contiguous buffer in the system memory. The SDMA transfer shall wait at the every boundary specified by these fields and the Host Controller generates the <b>DMA Interrupt</b> to request the Host Driver to update the <i>SDMA System Address</i> register. At the end of transfer, the Host Controller may issue or may not issue <b>DMA Interrupt</b>. In particular, <b>DMA Interrupt</b> shall not be issued after <b>Transfer Complete Interrupt</b> is issued. 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 SDMA transfer stops when the Host Controller detects carry out of the address from bit 11 to 12. These bits shall be supported when the <b>SDMA Support</b> in the <i>Capabilities</i> register is set to 1 and this function is active when the <b>DMA Enable</b> in the <i>Transfer Mode</i> register is set to 1. ADMA does not use this register.</p> <table border="1"> <tr> <td>000b</td> <td>4K bytes</td> <td>(Detects A11 carry out)</td> </tr> <tr> <td>001b</td> <td>8K bytes</td> <td>(Detects A12 carry out)</td> </tr> <tr> <td>010b</td> <td>16K Bytes</td> <td>(Detects A13 carry out)</td> </tr> <tr> <td>011b</td> <td>32K Bytes</td> <td>(Detects A14 carry out)</td> </tr> <tr> <td>100b</td> <td>64K bytes</td> <td>(Detects A15 carry out)</td> </tr> <tr> <td>101b</td> <td>128K Bytes</td> <td>(Detects A16 carry out)</td> </tr> <tr> <td>110b</td> <td>256K Bytes</td> <td>(Detects A17 carry out)</td> </tr> <tr> <td>111b</td> <td>512K Bytes</td> <td>(Detects A18 carry out)</td> </tr> </table>	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)
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)																								

**SD Host Controller Simplified Specification Version 2.00**

11-00	RW	<p><b>Transfer Block Size</b></p> <p>This register specifies the block size of data transfers for CMD17, CMD18, CMD24, CMD25, and CMD53. Values ranging from 1 up to the maximum buffer size can be set. In case of memory, it shall be set up to 512 bytes (Refer to Implementation Note in Section 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.</p> <table border="1" data-bbox="456 443 813 779"> <tr> <td>0800h</td> <td>2048 Bytes</td> </tr> <tr> <td>...</td> <td>...</td> </tr> <tr> <td>0200h</td> <td>512 Bytes</td> </tr> <tr> <td>01FFh</td> <td>511 Bytes</td> </tr> <tr> <td>...</td> <td>...</td> </tr> <tr> <td>0004h</td> <td>4 Bytes</td> </tr> <tr> <td>0003h</td> <td>3 Bytes</td> </tr> <tr> <td>0002h</td> <td>2 Bytes</td> </tr> <tr> <td>0001h</td> <td>1 Byte</td> </tr> <tr> <td>0000h</td> <td>No data transfer</td> </tr> </table>	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
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	Register Field Explanation										
15-00	RW	<p><b>Blocks Count For Current Transfer</b></p> <p>This register is enabled when <b>Block Count Enable</b> in the <i>Transfer Mode</i> 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 is transferred.</p> <p>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.</p> <p>When a suspend command is completed, the number of blocks yet to be transferred can be determined by reading this register. Before issuing a resume command, the Host Driver shall restore the previously saved block count.</p> <table border="1" style="margin-left: 20px;"> <tr> <td>FFFFh</td> <td>65535 blocks</td> </tr> <tr> <td>...</td> <td>...</td> </tr> <tr> <td>0002h</td> <td>2 blocks</td> </tr> <tr> <td>0001h</td> <td>1 block</td> </tr> <tr> <td>0000h</td> <td>Stop Count</td> </tr> </table>	FFFFh	65535 blocks	...	...	0002h	2 blocks	0001h	1 block	0000h	Stop 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	Register Field Explanation
31-00	RW	<b>Command Argument</b> The SD Command Argument is specified as bit39-8 of Command-Format in the 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.

D15	D06	D05	D04	D03	D02	D01	D00
Rsvd		Multi / Single Block Select	Data Transfer Direction Select	Rsvd	Auto CMD12 Enable	Block Count Enable	DMA Enable

**Figure 2-5 : Transfer Mode Register**

Location	Attrib	Register Field Explanation				
15-06	Rsvd	<b>Reserved</b>				
05	RW	<p><b>Multi / Single Block Select</b> This bit is set when issuing multiple-block transfer commands using DAT line. For any other commands, this bit shall be set to 0. If this bit is 0, it is not necessary to set the <i>Block Count</i> register. (Refer to Table 2-8)</p> <table border="1"> <tr> <td>1</td> <td>Multiple Block</td> </tr> <tr> <td>0</td> <td>Single Block</td> </tr> </table>	1	Multiple Block	0	Single Block
1	Multiple Block					
0	Single Block					
04	RW	<p><b>Data Transfer Direction Select</b> 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.</p> <table border="1"> <tr> <td>1</td> <td>Read (Card to Host)</td> </tr> <tr> <td>0</td> <td>Write (Host to Card)</td> </tr> </table>	1	Read (Card to Host)	0	Write (Host to Card)
1	Read (Card to Host)					
0	Write (Host to Card)					
03	Rsvd	<b>Reserved</b>				
02	RW	<p><b>Auto CMD12 Enable</b> 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.</p> <table border="1"> <tr> <td>1</td> <td>Enable</td> </tr> <tr> <td>0</td> <td>Disable</td> </tr> </table>	1	Enable	0	Disable
1	Enable					
0	Disable					

**SD Host Controller Simplified Specification Version 2.00**

01	RW	<p><b>Block Count Enable</b></p> <p>This bit is used to enable the <i>Block Count</i> register, which is only relevant for multiple block transfers. When this bit is 0, the <i>Block Count</i> register is disabled, which is useful in executing an infinite transfer. (Refer to Table 2-8)</p> <p>If ADMA2 data transfer is more than 65535 blocks, this bit shall be set to 0. In this case, data transfer length is designated by Descriptor Table.</p> <table border="1"> <tr> <td>1</td> <td>Enable</td> </tr> <tr> <td>0</td> <td>Disable</td> </tr> </table>	1	Enable	0	Disable
1	Enable					
0	Disable					
00	RW	<p><b>DMA Enable</b></p> <p>This bit enables DMA functionality as described in section 1.4. DMA can be enabled only if it is supported as indicated in the <i>Capabilities</i> register. One of the DMA modes can be selected by <b>DMA Select</b> in the <i>Host Control</i> 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 <i>Command</i> register (00Fh).</p> <table border="1"> <tr> <td>1</td> <td>DMA Data transfer</td> </tr> <tr> <td>0</td> <td>No data transfer or Non DMA data transfer</td> </tr> </table>	1	DMA Data transfer	0	No data transfer or Non DMA data transfer
1	DMA Data transfer					
0	No data transfer or Non DMA data transfer					

**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**

**SD Host Controller Simplified Specification Version 2.00****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	Register Field Explanation												
15-14	Rsvd	<b>Reserved</b>												
13-08	RW	<p><b>Command Index</b> 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 Physical Layer Specification and SDIO Card Specification.</p>												
07-06	RW	<p><b>Command Type</b> There are three types of special commands: Suspend, Resume and Abort. These bits <b>shall</b> be set to 00b for all other commands.</p> <p>(1) 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 <b>DAT</b> 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 <b>Continue Request</b> in the <i>Block Gap Control</i> register. (Refer to 3.12.1 Suspend Sequence)</p> <p>(2) 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.</p> <p>(3) 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 <b>DAT</b> line. After issuing the Abort command, the Host Driver should issue a software reset. (Refer to 3.8 Abort Transaction)</p> <table border="1"> <tbody> <tr> <td>11b</td> <td>Abort</td> <td>CMD12, CMD52 for writing "I/O Abort" in CCCR</td> </tr> <tr> <td>10b</td> <td>Resume</td> <td>CMD52 for writing "Function Select" in CCCR</td> </tr> <tr> <td>01b</td> <td>Suspend</td> <td>CMD52 for writing "Bus Suspend" in CCCR</td> </tr> <tr> <td>00b</td> <td>Normal</td> <td>Other commands</td> </tr> </tbody> </table>	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
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												



**SD Host Controller Simplified Specification Version 2.00**

05	RW	<p><b>Data Present Select</b> This bit is set to 1 to indicate that data is present and shall be transferred using the <b>DAT</b> line. It is set to 0 for the following:</p> <ul style="list-style-type: none"> <li>(1) Commands using only <b>CMD</b> line (ex. CMD52).</li> <li>(2) Commands with no data transfer but using busy signal on <b>DAT[0]</b> line (R1b or R5b ex. CMD38)</li> <li>(3) Resume command</li> </ul> <table border="1"> <tr> <td>1</td> <td>Data Present</td> </tr> <tr> <td>0</td> <td>No Data Present</td> </tr> </table>	1	Data Present	0	No Data Present				
1	Data Present									
0	No Data Present									
04	RW	<p><b>Command Index Check Enable</b> 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.</p> <table border="1"> <tr> <td>1</td> <td>Enable</td> </tr> <tr> <td>0</td> <td>Disable</td> </tr> </table>	1	Enable	0	Disable				
1	Enable									
0	Disable									
03	RW	<p><b>Command CRC Check Enable</b> 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 position of CRC field is determined according to the length of the response. (Refer to definition in D01-00 and Table 2-10 below.)</p> <table border="1"> <tr> <td>1</td> <td>Enable</td> </tr> <tr> <td>0</td> <td>Disable</td> </tr> </table>	1	Enable	0	Disable				
1	Enable									
0	Disable									
02	Rsvd	<b>Reserved</b>								
01-00	RW	<p><b>Response Type Select</b></p> <table border="1"> <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>	00	No Response	01	Response Length 136	10	Response Length 48	11	Response Length 48 check Busy after response
00	No Response									
01	Response Length 136									
10	Response Length 48									
11	Response Length 48 check Busy after response									

**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, R5, R6, R7
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.

**SD Host Controller Simplified Specification Version 2.00**

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	Register Field Explanation
127-00	ROC	<b>Command Response</b> 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 <i>Response</i> 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.

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].

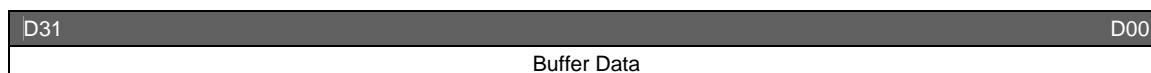
To be able to read the response status efficiently, the Host Controller only stores part of the response data in the *Response* register. This enables the Host Driver to read 32 bits of response data efficiently in one read cycle on a 32-bit bus system. Parts of the response, the Index field and the 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	Register Field Explanation
31-00	RW	<b>Buffer Data</b> The Host Controller buffer can be accessed through this 32-bit <i>Data Port</i> register. Refer to 1.7

**Table 2-13 : Buffer Data Port Register**

**SD Host Controller Simplified Specification Version 2.00****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**

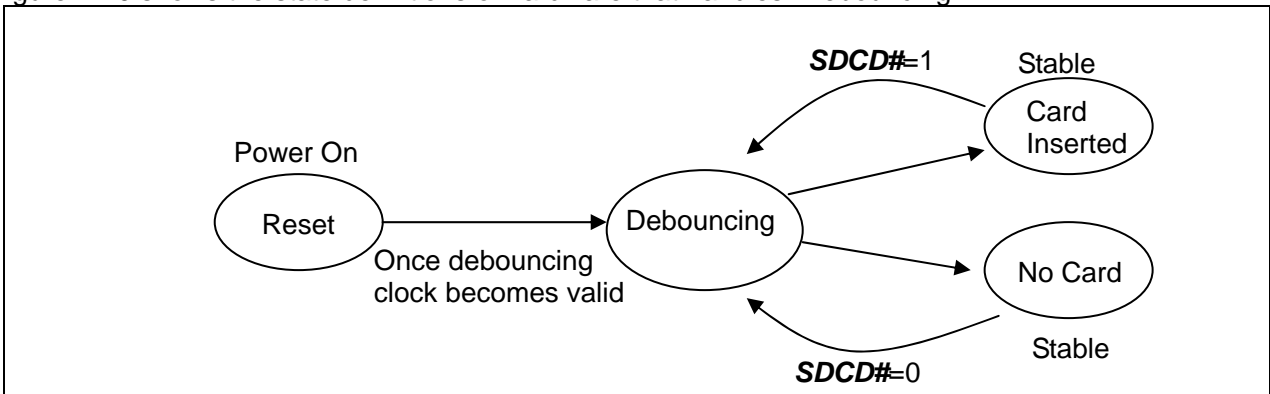
Location	Attrib	Register Field Explanation								
31-25	Rsvd	<b>Reserved</b>								
24	RO	<b>CMD Line Signal Level</b> This status is used to check the <b>CMD</b> line level to recover from errors, and for debugging.								
23-20	RO	<b>DAT[3:0] Line Signal Level</b> This status is used to check the <b>DAT</b> line level to recover from errors, and for debugging. This is especially useful in detecting the busy signal level from <b>DAT[0]</b> . <table border="1" data-bbox="527 1165 787 1291"> <tr> <td>D23</td> <td>DAT[3]</td> </tr> <tr> <td>D22</td> <td>DAT[2]</td> </tr> <tr> <td>D21</td> <td>DAT[1]</td> </tr> <tr> <td>D20</td> <td>DAT[0]</td> </tr> </table>	D23	DAT[3]	D22	DAT[2]	D21	DAT[1]	D20	DAT[0]
D23	DAT[3]									
D22	DAT[2]									
D21	DAT[1]									
D20	DAT[0]									
19	RO	<b>Write Protect Switch Pin Level</b> The Write Protect Switch is supported for memory and combo cards. This bit reflects the <b>SDWP#</b> pin. <table border="1" data-bbox="527 1417 1047 1491"> <tr> <td>1</td> <td>Write enabled (<b>SDWP#=1</b>)</td> </tr> <tr> <td>0</td> <td>Write protected (<b>SDWP#=0</b>)</td> </tr> </table>	1	Write enabled ( <b>SDWP#=1</b> )	0	Write protected ( <b>SDWP#=0</b> )				
1	Write enabled ( <b>SDWP#=1</b> )									
0	Write protected ( <b>SDWP#=0</b> )									
18	RO	<b>Card Detect Pin Level</b> This bit reflects the inverse value of the <b>SDCD#</b> pin. Debouncing is not performed on this bit. This bit may be valid when <b>Card State Stable</b> 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. <table border="1" data-bbox="527 1680 1047 1753"> <tr> <td>1</td> <td>Card present (<b>SDCD#=0</b>)</td> </tr> <tr> <td>0</td> <td>No card present (<b>SDCD#=1</b>)</td> </tr> </table>	1	Card present ( <b>SDCD#=0</b> )	0	No card present ( <b>SDCD#=1</b> )				
1	Card present ( <b>SDCD#=0</b> )									
0	No card present ( <b>SDCD#=1</b> )									

**SD Host Controller Simplified Specification Version 2.00**

17	RO	<p><b>Card State Stable</b>                  This bit is used for testing. If it is 0, the <b>Card Detect Pin Level</b> is not stable. If this bit is set to 1, it means the <b>Card Detect Pin Level</b> is stable. No Card state can be detected by this bit is set to 1 and <b>Card Inserted</b> is set to 0. The <b>Software Reset For All</b> in the <i>Software Reset</i> register shall not affect this bit.</p> <table border="1" data-bbox="524 415 964 478"> <tr> <td>1</td> <td>No Card or Inserted</td> </tr> <tr> <td>0</td> <td>Reset or Debouncing</td> </tr> </table>	1	No Card or Inserted	0	Reset or Debouncing
1	No Card or Inserted					
0	Reset or Debouncing					
16	RO	<p><b>Card Inserted</b>                  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 <b>Card Insertion</b> interrupt in the <i>Normal Interrupt Status</i> register and changing from 1 to 0 generates a <b>Card Removal</b> interrupt in the <i>Normal Interrupt Status</i> register. The <b>Software Reset For All</b> in the <i>Software Reset</i> register shall not affect this bit.</p> <p>If a card is removed while its power is on and its clock is oscillating, the Host Controller shall clear <b>SD Bus Power</b> in the <i>Power Control</i> register (refer to 2.2.11) and <b>SD Clock Enable</b> in the <i>Clock Control</i> register (refer to 2.2.14). When this bit is changed from 1 to 0, the Host Controller shall immediately stop driving <b>CMD</b> and <b>DAT[3:0]</b> (tri-state). In addition, the Host Driver should clear the Host Controller by the <b>Software Reset For All</b> in <i>Software Reset</i> register. The card detect is active regardless of the <b>SD Bus Power</b>.</p> <table border="1" data-bbox="524 1024 1127 1089"> <tr> <td>1</td> <td>Card Inserted</td> </tr> <tr> <td>0</td> <td>Reset or Debouncing or No Card</td> </tr> </table>	1	Card Inserted	0	Reset or Debouncing or No Card
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.

**SD Host Controller Simplified Specification Version 2.00**

Location	Attrib					
15-12	Rsvd	<b>Reserved</b>				
11	ROC	<p><b>Buffer Read Enable</b></p> <p>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. 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 <b>Buffer Read Ready</b> interrupt.</p> <table border="1"> <tr> <td>1</td> <td>Read enable</td> </tr> <tr> <td>0</td> <td>Read disable</td> </tr> </table>	1	Read enable	0	Read disable
1	Read enable					
0	Read disable					
10	ROC	<p><b>Buffer Write Enable</b></p> <p>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 <b>Buffer Write Ready</b> interrupt. The Host Controller should neither set Buffer Write Enable nor generate Buffer Write Ready Interrupt after the last block data is written to the Buffer Data Port Register.</p> <table border="1"> <tr> <td>1</td> <td>Write enable</td> </tr> <tr> <td>0</td> <td>Write disable</td> </tr> </table>	1	Write enable	0	Write disable
1	Write enable					
0	Write disable					
09	ROC	<p><b>Read Transfer Active</b></p> <p>This status is used for detecting completion of a read transfer. Refer to Section 3.12.3 for sequence details.</p> <p>This bit is set to 1 for either of the following conditions:</p> <ol style="list-style-type: none"> <li>(1) After the end bit of the read command.</li> <li>(2) When read operation is restarted by writing a 1 to <b>Continue Request</b> in the <i>Block Gap Control</i> register.</li> </ol> <p>This bit is cleared to 0 for either of the following conditions::</p> <ol style="list-style-type: none"> <li>(1) When the last data block as specified by block length is transferred to the System.</li> <li>(2) In case of ADMA2, end of read operation is designated by Descriptor Table.</li> <li>(3) When all valid data blocks in the Host Controller have been transferred to the System and no current block transfers are being sent as a result of the <b>Stop At Block Gap Request</b> being set to 1.</li> </ol> <p>A <b>Transfer Complete interrupt</b> is generated when this bit changes to 0.</p> <table border="1"> <tr> <td>1</td> <td>Transferring data</td> </tr> <tr> <td>0</td> <td>No valid data</td> </tr> </table>	1	Transferring data	0	No valid data
1	Transferring data					
0	No valid data					

08	ROC	<p><b>Write Transfer Active</b></p> <p>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 3.12.4 for more details on the sequence of events.</p> <p>This bit is set in either of the following cases:</p> <ol style="list-style-type: none"> <li>(1) After the end bit of the write command.</li> <li>(2) When write operation is restarted by writing a 1 to <b>Continue Request</b> in the <i>Block Gap Control</i> register.</li> </ol> <p>This bit is cleared in either of the following cases:</p> <ol style="list-style-type: none"> <li>(1) After getting the CRC status of the last data block as specified by the transfer count (Single and Multiple) In case of ADMA2, transfer count is designated by Descriptor Table.</li> <li>(2) After getting the CRC status of any block where data transmission is about to be stopped by a <b>Stop At Block Gap Request</b>.</li> </ol> <p>During a write transaction, a <b>Block Gap Event</b> interrupt is generated when this bit is changed to 0, as the result of the <b>Stop At Block Gap Request</b> being set. This status is useful for the Host Driver in determining non DAT line commands can be issued during write busy.</p> <table border="1" data-bbox="444 856 834 926"> <tr> <td>1</td> <td>Transferring data</td> </tr> <tr> <td>0</td> <td>No valid data</td> </tr> </table>	1	Transferring data	0	No valid data
1	Transferring data					
0	No valid data					
07-03	Rsvd	<b>Reserved</b>				
02	ROC	<p><b>DAT Line Active</b></p> <p>This bit indicates whether one of the <i>DAT</i> line on SD Bus is in use.</p> <p>(a) In the case of read transactions</p> <p>This status indicates whether a read transfer is executing on the SD Bus. Changing this value from 1 to 0 generates a <b>Block Gap Event interrupt</b> in the <i>Normal Interrupt Status</i> register, as the result of the <b>Stop At Block Gap Request</b> being set. Refer to Section 3.12.3 for details on timing.</p> <p>This bit shall be set in either of the following cases:</p> <ol style="list-style-type: none"> <li>(1) After the end bit of the read command.</li> <li>(2) When writing a 1 to <b>Continue Request</b> in the <i>Block Gap Control</i> register to restart a read transfer.</li> </ol> <p>This bit shall be cleared in either of the following cases:</p> <ol style="list-style-type: none"> <li>(1) When the end bit of the last data block is sent from the SD Bus to the Host Controller. In case of ADMA2, the last block is designated by the last transfer of Descriptor Table.</li> <li>(2) When a read transfer is stopped at the block gap initiated by a <b>Stop At Block Gap Request</b>.</li> </ol> <p>The Host Controller shall stop read operation at the start of the interrupt cycle of the next block gap by driving Read Wait or stopping SD clock. If the Read Wait signal is already driven (due to data buffer cannot receive data), the Host Controller can continue to stop read operation by driving the Read Wait signal. It is necessary to support Read Wait in order to use suspend / resume function.</p> <p>(b) In the case of write transactions</p> <p>This status indicates that a write transfer is executing on the SD Bus. Changing this value from 1 to 0 generate a <b>Transfer Complete</b> interrupt in the <i>Normal Interrupt</i></p>				

**SD Host Controller Simplified Specification Version 2.00**

		<p>Status register. Refer to Section 3.12.4 for sequence details.</p> <p>This bit shall be set in either of the following cases:</p> <ol style="list-style-type: none"> <li>(1) After the end bit of the write command.</li> <li>(2) When writing to 1 to <b>Continue Request</b> in the <i>Block Gap Control</i> register to continue a write transfer.</li> </ol> <p>This bit shall be cleared in either of the following cases:</p> <ol style="list-style-type: none"> <li>(1) When the SD card releases write busy of the last data block. If SD card does not drive busy signal for 8 SD Clocks, the Host Controller shall consider the card drive "Not Busy". In case of ADMA2, the last block is designated by the last transfer of Descriptor Table.</li> <li>(2) When the SD card releases write busy prior to waiting for write transfer as a result of a <b>Stop At Block Gap Request</b>.</li> </ol> <p>(c) Command with busy This status indicates whether a command indicates busy (ex. erase command for memory) is executing on the SD Bus. This bit is set after the end bit of the command with busy and cleared when busy is de-asserted. Changing this bit from 1 to 0 generate a <b>Transfer Complete</b> interrupt in the <i>Normal Interrupt Status</i> register. Refer to Figure 2-11 to Figure 2-13.</p> <table border="1"> <tr> <td>1</td> <td>DAT Line Active</td> </tr> <tr> <td>0</td> <td>DAT Line Inactive</td> </tr> </table>	1	DAT Line Active	0	DAT Line Inactive
1	DAT Line Active					
0	DAT Line Inactive					
01	ROC	<p><b>Command Inhibit (DAT)</b></p> <p>This status bit is generated if either the <b>DAT Line Active</b> or the <b>Read Transfer Active</b> 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 <b>Command Inhibit (DAT)</b> (ex. R1b, R5b type). Changing from 1 to 0 generates a <b>Transfer Complete</b> interrupt in the <i>Normal Interrupt Status</i> register.</p> <p>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.</p> <table border="1"> <tr> <td>1</td> <td>Cannot issue command which uses the <b>DAT</b> line</td> </tr> <tr> <td>0</td> <td>Can issue command which uses the <b>DAT</b> line</td> </tr> </table>	1	Cannot issue command which uses the <b>DAT</b> line	0	Can issue command which uses the <b>DAT</b> line
1	Cannot issue command which uses the <b>DAT</b> line					
0	Can issue command which uses the <b>DAT</b> line					
00	ROC	<p><b>Command Inhibit (CMD)</b></p> <p>If this bit is 0, it indicates the <b>CMD</b> line is not in use and the Host Controller can issue a SD Command using the <b>CMD</b> line.</p> <p>This bit is set immediately after the <i>Command</i> register (00Fh) is written. This bit is cleared when the command response is received. Even if the <b>Command Inhibit (DAT)</b> is set to 1, Commands using only the <b>CMD</b> line can be issued if this bit is 0. Changing from 1 to 0 generates a <b>Command Complete interrupt</b> in the <i>Normal Interrupt Status</i> register. If the Host Controller cannot issue the command because of a command conflict error (Refer to <b>Command CRC Error</b> in 2.2.18) or because of <b>Command Not Issued By Auto CMD12 Error</b> (Refer to 2.2.23), this bit shall remain 1 and the <b>Command Complete</b> is not set. Status issuing Auto CMD12 is not read from this bit.</p> <table border="1"> <tr> <td>1</td> <td>Cannot issue command</td> </tr> <tr> <td>0</td> <td>Can issue command using only <b>CMD</b> line</td> </tr> </table>	1	Cannot issue command	0	Can issue command using only <b>CMD</b> line
1	Cannot issue command					
0	Can issue command using only <b>CMD</b> line					

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



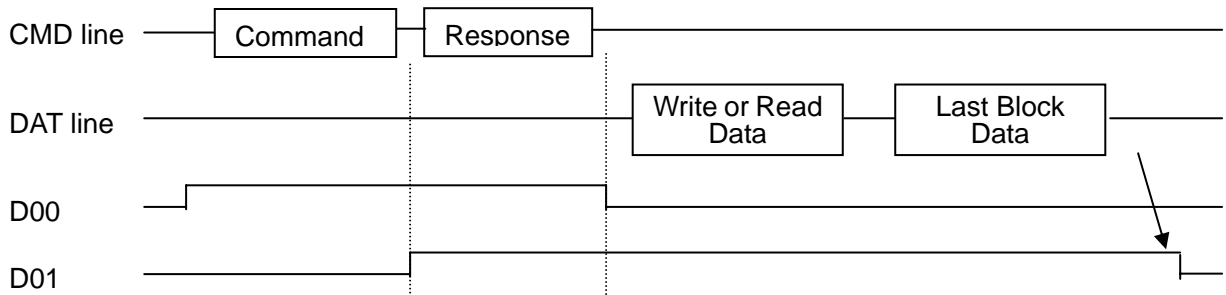
**Implementation Note:**

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 Physical Layer Specification may add other commands to this list in the future.

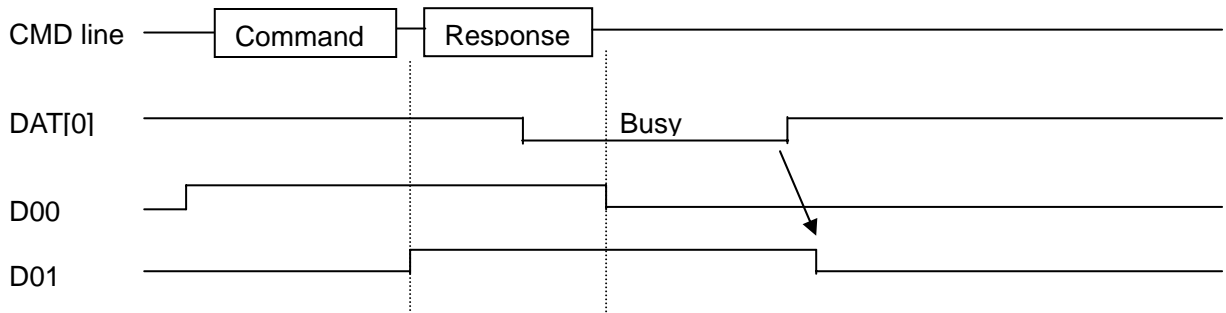
**Implementation Note:**

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.

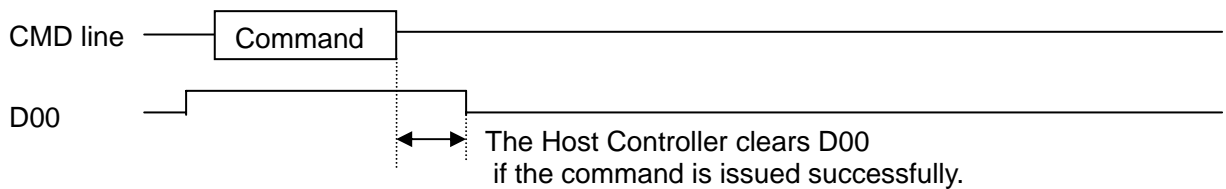
Figure 2-11 to Figure 2-13 shows the timing of setting and clearing the **Command Inhibit (DAT)** and the **Command Inhibit (CMD)**.



**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**

**SD Host Controller Simplified Specification Version 2.00****2.2.10. Host Control Register (Offset 028h)**

D07	D06	D05	D04	D03	D02	D01	D00
Card Detect Signal Selection	Card Detect Test Level	Extended Data Transfer Width		DMA Select	High Speed Enable	Data Transfer Width	LED Control

**Figure 2-14 : Host Control Register**

Location	Attrib	Register Field Explanation								
07	RW	<p><b>Card Detect Signal Selection</b> This bit selects source for the card detection.</p> <table border="1"> <tr> <td>1</td> <td>The Card Detect Test Level is selected (for test purpose)</td> </tr> <tr> <td>0</td> <td>SDCD# is selected (for normal use)</td> </tr> </table> <p>When the source for the card detection is switched, the interrupt should be disabled during the switching period by clearing the <i>Interrupt Status/Signal Enable</i> register in order to mask unexpected interrupt being caused by the glitch. The <i>Interrupt Status/Signal Enable</i> should be disabled during over the period of debouncing.</p>	1	The Card Detect Test Level is selected (for test purpose)	0	SDCD# is selected (for normal use)				
1	The Card Detect Test Level is selected (for test purpose)									
0	SDCD# is selected (for normal use)									
06	RW	<p><b>Card Detect Test Level</b> This bit is enabled while the Card Detect Signal Selection is set to 1 and it indicates card inserted or not.</p> <table border="1"> <tr> <td>1</td> <td>Card Inserted</td> </tr> <tr> <td>0</td> <td>No Card</td> </tr> </table>	1	Card Inserted	0	No Card				
1	Card Inserted									
0	No Card									
05	RW	<b>Reserved (New assignment is not allowed)</b>								
04-03	RW	<p><b>DMA Select</b> One of supported DMA modes can be selected. The host driver shall check support of DMA modes by referring the <i>Capabilities</i> register. Use of selected DMA is determined by <b>DMA Enable</b> of the <i>Transfer Mode</i> register.</p> <table border="1"> <tr> <td>00</td> <td>SDMA is selected</td> </tr> <tr> <td>01</td> <td>Reserved (Refer to Appendix C)</td> </tr> <tr> <td>10</td> <td>32-bit Address ADMA2 is selected</td> </tr> <tr> <td>11</td> <td>64-bit Address ADMA2 is selected</td> </tr> </table>	00	SDMA is selected	01	Reserved (Refer to Appendix C)	10	32-bit Address ADMA2 is selected	11	64-bit Address ADMA2 is selected
00	SDMA is selected									
01	Reserved (Refer to Appendix C)									
10	32-bit Address ADMA2 is selected									
11	64-bit Address ADMA2 is selected									
02	RW	<p><b>High Speed Enable</b> This bit is optional. Before setting this bit, the Host Driver shall check the <b>High Speed Support</b> in the <i>Capabilities</i> register. If this bit is set to 0 (default), the Host Controller outputs <b>CMD</b> line and <b>DAT</b> lines at the falling edge of the SD Clock (up to 25MHz). If this bit is set to 1, the Host Controller outputs <b>CMD</b> line and <b>DAT</b> lines at the rising edge of the SD Clock (up to 50MHz).</p> <table border="1"> <tr> <td>1</td> <td>High Speed mode</td> </tr> <tr> <td>0</td> <td>Normal Speed mode</td> </tr> </table>	1	High Speed mode	0	Normal Speed mode				
1	High Speed mode									
0	Normal Speed mode									

**SD Host Controller Simplified Specification Version 2.00**

01	RW	<p><b>Data Transfer Width</b> 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.</p> <table border="1" data-bbox="446 317 755 380"> <tr> <td data-bbox="446 317 592 348">1</td> <td data-bbox="592 317 755 348">4-bit mode</td> </tr> <tr> <td data-bbox="446 348 592 380">0</td> <td data-bbox="592 348 755 380">1-bit mode</td> </tr> </table>	1	4-bit mode	0	1-bit mode
1	4-bit mode					
0	1-bit mode					
00	RW	<p><b>LED Control</b> 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.</p> <table border="1" data-bbox="446 577 755 640"> <tr> <td data-bbox="446 577 592 609">1</td> <td data-bbox="592 577 755 609">LED on</td> </tr> <tr> <td data-bbox="446 609 592 640">0</td> <td data-bbox="592 609 755 640">LED off</td> </tr> </table>	1	LED on	0	LED off
1	LED on					
0	LED off					

**Table 2-16 : Host Control Register**

## 2.2.11. Power Control Register (Offset 029h)

D07	D04	D03	D01	D00
Rsvd		SD Bus Voltage Select		SD Bus Power

Figure 2-15 : Power Control Register

Location	Attrib	Register Field Explanation								
07-04	Rsvd	<b>Reserved</b>								
03-01	RW	<p><b>SD Bus Voltage Select</b> 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 <b>Voltage Support</b> bits in the <i>Capabilities</i> register. If an unsupported voltage is selected, the Host System shall not supply SD Bus voltage.</p> <table border="1"> <tbody> <tr> <td>111b</td> <td>3.3V (Typ.)</td> </tr> <tr> <td>110b</td> <td>3.0V (Typ.)</td> </tr> <tr> <td>101b</td> <td>1.8V (Typ.)</td> </tr> <tr> <td>100b – 000b</td> <td>Reserved</td> </tr> </tbody> </table>	111b	3.3V (Typ.)	110b	3.0V (Typ.)	101b	1.8V (Typ.)	100b – 000b	Reserved
111b	3.3V (Typ.)									
110b	3.0V (Typ.)									
101b	1.8V (Typ.)									
100b – 000b	Reserved									
00	RW	<p><b>SD Bus Power</b> Before setting this bit, the SD Host Driver shall set <b>SD Bus Voltage Select</b>. If the Host Controller detects the No Card state, this bit shall be cleared. If this bit is cleared, the Host Controller shall immediately stop driving <b>CMD</b> and <b>DAT[3:0]</b> (tri-state) and drive <b>SDCLK</b> to low level (refer to Section 2.2.14).</p> <table border="1"> <tbody> <tr> <td>1</td> <td>Power on</td> </tr> <tr> <td>0</td> <td>Power off</td> </tr> </tbody> </table>	1	Power on	0	Power off				
1	Power on									
0	Power off									

Table 2-17 : Power Control Register

**Implementation Note:**

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	Register Field Explanation				
07-04	Rsvd	<b>Reserved</b>				
03	RW	<p><b>Interrupt At Block Gap</b> 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.</p> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Disabled</td> </tr> </table>	1	Enabled	0	Disabled
1	Enabled					
0	Disabled					
02	RW	<p><b>Read Wait Control</b> 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 <b>DAT[2]</b> 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 <b>DAT</b> line conflict may occur. If this bit is set to 0, Suspend/Resume cannot be supported.</p> <table border="1"> <tr> <td>1</td> <td>Enable Read Wait Control</td> </tr> <tr> <td>0</td> <td>Disable Read Wait Control</td> </tr> </table>	1	Enable Read Wait Control	0	Disable Read Wait Control
1	Enable Read Wait Control					
0	Disable Read Wait Control					
01	RWAC	<p><b>Continue Request</b> This bit is used to restart a transaction, which was stopped using the <b>Stop At Block Gap Request</b>. To cancel stop at the block gap, set <b>Stop At Block Gap Request</b> 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 <b>DAT Line Active</b> changes from 0 to 1 as a read transaction restarts.  (2) In the case of a write transaction, the <b>Write Transfer Active</b> 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 <b>Stop At Block Gap Request</b> is set to 1, any write to this bit is ignored.</p> <table border="1"> <tr> <td>1</td> <td>Restart</td> </tr> <tr> <td>0</td> <td>Not affect</td> </tr> </table>	1	Restart	0	Not affect
1	Restart					
0	Not affect					
00	RW	<p><b>Stop At Block Gap Request</b> This bit is used to stop executing read and write transaction at the next block gap for non-DMA, SDMA and ADMA transfers. The Host Driver shall leave this bit set</p>				

		<p>to 1 until the <b>Transfer Complete</b> is set to 1. Clearing both <b>Stop At Block Gap Request</b> and <b>Continue Request</b> shall not cause the transaction to restart. When Host Controller version is 1.00, the Host Driver can set this bit if the card supports <b>Read Wait Control</b>. When Host Controller version is 2.00 or higher, the Host Driver can set this bit regardless of the card supports <b>Read Wait Control</b>. The Host Controller shall stop read transfer by using Read Wait or stopping SD clock. In case of write transfers in which the Host Driver writes data to the <i>Buffer Data Port</i> 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 <i>Buffer Data Port</i> register. This bit affects <b>Read Transfer Active</b>, <b>Write Transfer Active</b>, <b>DAT Line Active</b> and <b>Command Inhibit (DAT)</b> in the <i>Present State</i> register. Regarding detailed control of bits D01 and D00, refer to Section 3.8 and 3.12.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">Stop</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">Transfer</td> </tr> </table>	1	Stop	0	Transfer
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.

D07	D03	D02	D01	D00
Rsvd		Wakeup Event Enable On SD Card Removal	Wakeup Event Enable On SD Card Insertion	Wakeup Event Enable On SD Card Interrupt

Figure 2-17 : Wakeup Control Register

Location	Attrib	Register Field Explanation				
07-03	Rsvd	<b>Reserved</b>				
02	RW	<p><b>Wakeup Event Enable On SD Card Removal</b> This bit enables wakeup event via <b>Card Removal</b> assertion in the <i>Normal Interrupt Status</i> register. <b>FN_WUS</b> (Wake Up Support) in CIS does not affect this bit.</p> <table border="1"> <tr> <td>1</td> <td>Enable</td> </tr> <tr> <td>0</td> <td>Disable</td> </tr> </table>	1	Enable	0	Disable
1	Enable					
0	Disable					
01	RW	<p><b>Wakeup Event Enable On SD Card Insertion</b> This bit enables wakeup event via <b>Card Insertion</b> assertion in the <i>Normal Interrupt Status</i> register. <b>FN_WUS</b> (Wake Up Support) in CIS does not affect this bit.</p> <table border="1"> <tr> <td>1</td> <td>Enable</td> </tr> <tr> <td>0</td> <td>Disable</td> </tr> </table>	1	Enable	0	Disable
1	Enable					
0	Disable					
00	RW	<p><b>Wakeup Event Enable On Card Interrupt</b> This bit enables wakeup event via <b>Card Interrupt</b> assertion in the <i>Normal Interrupt Status</i> register. This bit can be set to 1 if <b>FN_WUS</b> (Wake Up Support) in CIS is set to 1.</p> <table border="1"> <tr> <td>1</td> <td>Enable</td> </tr> <tr> <td>0</td> <td>Disable</td> </tr> </table>	1	Enable	0	Disable
1	Enable					
0	Disable					

Table 2-19 : Wakeup Control Register



**SD Host Controller Simplified Specification Version 2.00****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	Register Field Explanation																		
15-08	RW	<p><b>SDCLK Frequency Select</b> This register is used to select the frequency of <b>SDCLK</b> pin. The frequency is not programmed directly; rather this register holds the divisor of the <b>Base Clock Frequency For SD Clock</b> in the <i>Capabilities</i> register. Only the following settings are allowed.</p> <table border="1"> <tbody> <tr><td>80h</td><td>base clock divided by 256</td></tr> <tr><td>40h</td><td>base clock divided by 128</td></tr> <tr><td>20h</td><td>base clock divided by 64</td></tr> <tr><td>10h</td><td>base clock divided by 32</td></tr> <tr><td>08h</td><td>base clock divided by 16</td></tr> <tr><td>04h</td><td>base clock divided by 8</td></tr> <tr><td>02h</td><td>base clock divided by 4</td></tr> <tr><td>01h</td><td>base clock divided by 2</td></tr> <tr><td>00h</td><td>Base clock (10MHz-63MHz)</td></tr> </tbody> </table> <p>Setting 00h specifies the highest frequency of the SD Clock. When setting multiple bits, the most significant bit is used as the divisor but it should not be set. The three default divider values can be calculated by the frequency that is defined by the <b>Base Clock Frequency For SD Clock</b> in the <i>Capabilities</i> register.</p> <p>400KHz divider value 25MHz divider value 50MHz divider value</p> <p>According to the Physical Layer Specification, the maximum SD Clock frequency is 25 MHz in normal speed mode and 50MHz in high speed mode, and shall never exceed this limit.</p> <p>The frequency of SDCLK is set by the following formula:</p> $\text{Clock Frequency} = (\text{Base Clock}) / \text{divisor}$ <p>Thus, choose the smallest possible divisor which results in a clock frequency that is less than or equal to the target frequency.</p> <p>For example, if the <b>Base Clock Frequency For SD Clock</b> in the <i>Capabilities</i> 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.</p>	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)
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)																			

**SD Host Controller Simplified Specification Version 2.00**

07-03	Rsvd	<b>Reserved</b>				
02	RW	<p><b>SD Clock Enable</b></p> <p>The Host Controller shall stop <b>SDCLK</b> when writing this bit to 0. <b>SDCLK Frequency Select</b> can be changed when this bit is 0. Then, the Host Controller shall maintain the same clock frequency until <b>SDCLK</b> is stopped (Stop at <b>SDCLK=0</b>). If the <b>Card Inserted</b> in the <i>Present State register</i> is cleared, this bit shall be cleared.</p> <table border="1"> <tr> <td>1</td> <td>Enable</td> </tr> <tr> <td>0</td> <td>Disable</td> </tr> </table>	1	Enable	0	Disable
1	Enable					
0	Disable					
01	ROC	<p><b>Internal Clock Stable</b></p> <p>This bit is set to 1 when SD Clock is stable after writing to <b>Internal Clock Enable</b> in this register to 1. The SD Host Driver shall wait to set <b>SD Clock Enable</b> until this bit is set to 1.</p> <p>Note: This is useful when using PLL for a clock oscillator that requires setup time.</p> <table border="1"> <tr> <td>1</td> <td>Ready</td> </tr> <tr> <td>0</td> <td>Not Ready</td> </tr> </table>	1	Ready	0	Not Ready
1	Ready					
0	Not Ready					
00	RW	<p><b>Internal Clock Enable</b></p> <p>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 <b>Internal Clock Stable</b> in this register to 1. This bit shall not affect card detection.</p> <table border="1"> <tr> <td>1</td> <td>Oscillate</td> </tr> <tr> <td>0</td> <td>Stop</td> </tr> </table>	1	Oscillate	0	Stop
1	Oscillate					
0	Stop					

**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	Register Field Explanation										
07-04	Rsvd	<b>Reserved</b>										
03-00	RW	<p><b>Data Timeout Counter Value</b>            This value determines the interval by which DAT line timeouts are detected. For more information about timeout generation, refer to the <b>Data Timeout Error</b> in the <i>Error Interrupt Status</i> register. 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 <b>Data Timeout Error Status Enable</b> (in the <i>Error Interrupt Status Enable</i> register)</p> <table border="1"> <tr> <td>1111b</td> <td>Reserved</td> </tr> <tr> <td>1110b</td> <td>TMCLK x 2<sup>27</sup></td> </tr> <tr> <td>.....</td> <td>.....</td> </tr> <tr> <td>0001b</td> <td>TMCLK x 2<sup>14</sup></td> </tr> <tr> <td>0000b</td> <td>TMCLK x 2<sup>13</sup></td> </tr> </table>	1111b	Reserved	1110b	TMCLK x 2 <sup>27</sup>	.....	.....	0001b	TMCLK x 2 <sup>14</sup>	0000b	TMCLK x 2 <sup>13</sup>
1111b	Reserved											
1110b	TMCLK x 2 <sup>27</sup>											
.....	.....											
0001b	TMCLK x 2 <sup>14</sup>											
0000b	TMCLK x 2 <sup>13</sup>											

**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.

D07	D03	D02	D01	D00
Rsvd		Software Reset For DAT Line	Software Reset For CMD Line	Software Reset For All

**Figure 2-20 : Software Reset Register**

Location	Attrib	Register Field Explanation				
07-03	Rsvd	<b>Reserved</b>				
02	RWAC	<p><b>Software Reset For DAT Line</b> Only part of data circuit is reset. DMA circuit is also reset.</p> <p>The following registers and bits are cleared by this bit:</p> <p><i>Buffer Data Port</i> register Buffer is cleared and initialized.</p> <p><i>Present State</i> register  <b>Buffer Read Enable</b>  <b>Buffer Write Enable</b>  <b>Read Transfer Active</b>  <b>Write Transfer Active</b>  <b>DAT Line Active</b>  <b>Command Inhibit (DAT)</b></p> <p><i>Block Gap Control</i> register  <b>Continue Request</b>  <b>Stop At Block Gap Request</b></p> <p><i>Normal Interrupt Status</i> register  <b>Buffer Read Ready</b>  <b>Buffer Write Ready</b>  <b>DMA Interrupt</b>  <b>Block Gap Event</b>  <b>Transfer Complete</b></p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">Reset</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">Work</td> </tr> </table>	1	Reset	0	Work
1	Reset					
0	Work					

**SD Host Controller Simplified Specification Version 2.00**

01	RWAC	<p><b>Software Reset For CMD Line</b> Only part of command circuit is reset.</p> <p>The following registers and bits are cleared by this bit:  <i>Present State</i> register  <b>Command Inhibit (CMD)</b>  <i>Normal Interrupt Status</i> register  <b>Command Complete</b></p> <table border="1" data-bbox="459 478 711 543"> <tr> <td>1</td> <td>Reset</td> </tr> <tr> <td>0</td> <td>Work</td> </tr> </table>	1	Reset	0	Work
1	Reset					
0	Work					
00	RWAC	<p><b>Software Reset For All</b> 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 <i>Capabilities</i> registers are valid and the Host Driver can read them. Additional use of <b>Software Reset For All</b> may not affect the value of the <i>Capabilities</i> registers. If this bit is set to 1, the host driver should issue reset command and reinitialize the SD card.</p> <table border="1" data-bbox="459 835 711 896"> <tr> <td>1</td> <td>Reset</td> </tr> <tr> <td>0</td> <td>Work</td> </tr> </table>	1	Reset	0	Work
1	Reset					
0	Work					

**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	Register Field Explanation				
15	ROC	<p><b>Error Interrupt</b> If any of the bits in the <i>Error Interrupt Status</i> 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.</p> <table border="1"> <tr> <td>1</td> <td>Error</td> </tr> <tr> <td>0</td> <td>No Error</td> </tr> </table>	1	Error	0	No Error
1	Error					
0	No Error					
14-09	Rsvd	<b>Reserved</b>				
08	ROC	<p><b>Card Interrupt</b> 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, <b>Card Interrupt Status Enable</b> in the <i>Normal Interrupt Status Enable</i> 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 <b>Card Interrupt Status Enable</b> to 1 and start sampling the interrupt signal again.</p> <table border="1"> <tr> <td>1</td> <td>Generate Card Interrupt</td> </tr> <tr> <td>0</td> <td>No Card Interrupt</td> </tr> </table>	1	Generate Card Interrupt	0	No Card Interrupt
1	Generate Card Interrupt					
0	No Card Interrupt					
07	RW1C	<p><b>Card Removal</b> This status is set if the <b>Card Inserted</b> in the <i>Present State</i> 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 <i>Present State</i> 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.</p>				

## SD Host Controller Simplified Specification Version 2.00

		<table border="1"> <tr> <td>1</td> <td>Card removed</td> </tr> <tr> <td>0</td> <td>Card state stable or Debouncing</td> </tr> </table>	1	Card removed	0	Card state stable or Debouncing
1	Card removed					
0	Card state stable or Debouncing					
06	RW1C	<p><b>Card Insertion</b> This status is set if the <b>Card Inserted</b> in the <i>Present State</i> register changes from 0 to 1. When the Host Driver writes this bit to 1 to clear this status, the status of the <b>Card Inserted</b> in the <i>Present State</i> 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.</p> <table border="1"> <tr> <td>1</td> <td>Card inserted</td> </tr> <tr> <td>0</td> <td>Card state stable or Debouncing</td> </tr> </table>	1	Card inserted	0	Card state stable or Debouncing
1	Card inserted					
0	Card state stable or Debouncing					
05	RW1C	<p><b>Buffer Read Ready</b> This status is set if the <b>Buffer Read Enable</b> changes from 0 to 1. Refer to the <b>Buffer Read Enable</b> in the <i>Present State</i> register.</p> <table border="1"> <tr> <td>1</td> <td>Ready to read buffer</td> </tr> <tr> <td>0</td> <td>Not ready to read buffer</td> </tr> </table>	1	Ready to read buffer	0	Not ready to read buffer
1	Ready to read buffer					
0	Not ready to read buffer					
04	RW1C	<p><b>Buffer Write Ready</b> This status is set if the <b>Buffer Write Enable</b> changes from 0 to 1. Refer to the <b>Buffer Write Enable</b> in the <i>Present State</i> register.</p> <table border="1"> <tr> <td>1</td> <td>Ready to write buffer</td> </tr> <tr> <td>0</td> <td>Not ready to write buffer</td> </tr> </table>	1	Ready to write buffer	0	Not ready to write buffer
1	Ready to write buffer					
0	Not ready to write buffer					
03	RW1C	<p><b>DMA Interrupt</b> This status is set if the Host Controller detects the Host SDMA Buffer boundary during transfer. Refer to the <b>Host SDMA Buffer Boundary</b> in the <i>Block Size</i> register. Other DMA interrupt factors may be added in the future. In case of ADMA, by setting Int field in the descriptor table, Host Controller generates this interrupt. Suppose that it is used for debugging. This interrupt shall not be generated after the <b>Transfer Complete</b>.</p> <table border="1"> <tr> <td>1</td> <td><b>DMA Interrupt</b> is generated</td> </tr> <tr> <td>0</td> <td>No <b>DMA Interrupt</b></td> </tr> </table>	1	<b>DMA Interrupt</b> is generated	0	No <b>DMA Interrupt</b>
1	<b>DMA Interrupt</b> is generated					
0	No <b>DMA Interrupt</b>					
02	RW1C	<p><b>Block Gap Event</b> If the <b>Stop At Block Gap Request</b> in the <i>Block Gap Control</i> register is set, this bit is set when both a read / write transaction is stopped at a block gap. If <b>Stop At Block Gap Request</b> is not set to 1, this bit is not set to 1.</p> <p>(1) In the case of a Read Transaction This bit is set at the falling edge of the <b>DAT Line Active Status</b> (When the transaction is stopped at SD Bus timing. The Read Wait shall be supported in order to use this function. Refer to Section 3.12.3 about the detail timing.</p> <p>(2) Case of Write Transaction This bit is set at the falling edge of <b>Write Transfer Active Status</b> (After getting CRC status at SD Bus timing). Refer to Section 3.12.4 for more details on the sequence of events.</p> <table border="1"> <tr> <td>1</td> <td>Transaction stopped at block gap</td> </tr> <tr> <td>0</td> <td>No <b>Block Gap Event</b></td> </tr> </table>	1	Transaction stopped at block gap	0	No <b>Block Gap Event</b>
1	Transaction stopped at block gap					
0	No <b>Block Gap Event</b>					

01	RW1C	<p><b>Transfer Complete</b> This bit is set when a read / write transfer and a command with busy is completed.</p> <p>(1) In the case of a Read Transaction This bit is set at the falling edge of <b>Read Transfer Active Status</b>. This interrupt is generated in two cases. 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 <b>Stop At Block Gap Request</b> in the <i>Block Gap Control</i> register (After valid data has been read to the Host System). Refer to Section 3.12.3 for more details on the sequence of events.</p> <p>(2) In the case of a Write Transaction This bit is set at the falling edge of the <b>DAT Line Active Status</b>. This interrupt is generated in two cases. 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 <b>Stop At Block Gap Request</b> in the <i>Block Gap Control</i> register and data transfers completed. (After valid data is written to the SD card and the busy signal released). Refer to Section 3.12.4 for more details on the sequence of events.</p> <p>(3) In the case of a command with busy This bit is set when busy is de-asserted. Refer to <b>DAT Line Active</b> and <b>Command Inhibit (DAT)</b> in the <i>Present State</i> register.</p> <p>The table below shows that <b>Transfer Complete</b> has higher priority than <b>Data Timeout Error</b>. If both bits are set to 1, execution of a command can be considered to be completed.</p> <p><b>Relation between Transfer Complete and Data Timeout Error</b></p> <table border="1" data-bbox="516 1182 1377 1354"> <thead> <tr> <th>Transfer Complete</th> <th>Data Timeout Error</th> <th>Meaning of the status</th> </tr> </thead> <tbody> <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>Command Execution complete</td> </tr> </tbody> </table> <table border="1" data-bbox="516 1388 1214 1449"> <tbody> <tr> <td>1</td> <td>Command execution is completed</td> </tr> <tr> <td>0</td> <td>Not complete</td> </tr> </tbody> </table>	Transfer Complete	Data Timeout Error	Meaning of the status	0	0	Interrupted by another factor	0	1	Timeout occur during transfer	1	Don't Care	Command Execution complete	1	Command execution is completed	0	Not complete
Transfer Complete	Data Timeout Error	Meaning of the status																
0	0	Interrupted by another factor																
0	1	Timeout occur during transfer																
1	Don't Care	Command Execution complete																
1	Command execution is completed																	
0	Not complete																	



00	RW1C	<p><b>Command Complete</b> This bit is set when get the end bit of the command response. (Except Auto CMD12) Refer to <b>Command Inhibit (CMD)</b> in the <i>Present State</i> register.</p> <p>The table below shows that <b>Command Timeout Error</b> has higher priority than <b>Command Complete</b>. If both bits are set to 1, it can be considered that the response was not received correctly.</p> <table border="1" data-bbox="516 478 1352 684"> <thead> <tr> <th>Command Complete</th> <th>Command Timeout Error</th> <th>Meaning of the status</th> </tr> </thead> <tbody> <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> </tbody> </table> <table border="1" data-bbox="516 716 982 781"> <tbody> <tr> <td>1</td> <td>Command complete</td> </tr> <tr> <td>0</td> <td>No command complete</td> </tr> </tbody> </table>	Command Complete	Command Timeout Error	Meaning of the status	0	0	Interrupted by another factor	Don't Care	1	Response not received within 64 SDCLK cycles.	1	0	Response received	1	Command complete	0	No command complete
Command Complete	Command Timeout Error	Meaning of the status																
0	0	Interrupted by another factor																
Don't Care	1	Response not received within 64 SDCLK cycles.																
1	0	Response received																
1	Command complete																	
0	No command complete																	

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	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00	
Vendor Specific Error Status			Rsvd		ADMA Error	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	Register Field Explanation				
15-12	RW1C	<b>Vendor Specific Error Status</b> Additional status bits can be defined in this register by the vendor.				
11-10	Rsvd	<b>Reserved</b>				
09	RW1C	<p><b>ADMA Error</b> This bit is set when the Host Controller detects errors during ADMA based data transfer. The state of the ADMA at an error occurrence is saved in the <i>ADMA Error Status</i> Register, In addition, the Host Controller generates this Interrupt when it detects invalid descriptor data (Valid=0) at the ST_FDS state. <b>ADMA Error State</b> in the <i>ADMA Error Status</i> indicates that an error occurs in ST_FDS state. The Host Driver may find that Valid bit is not set at the error descriptor.</p> <table border="1"> <tr> <td>1</td> <td>Error</td> </tr> <tr> <td>0</td> <td>No Error</td> </tr> </table>	1	Error	0	No Error
1	Error					
0	No Error					
08	RW1C	<p><b>Auto CMD12 Error</b> This bit is set when detecting that one of the bits D00-D04 in <i>Auto CMD12 Error Status</i> 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.</p> <table border="1"> <tr> <td>1</td> <td>Error</td> </tr> <tr> <td>0</td> <td>No Error</td> </tr> </table>	1	Error	0	No Error
1	Error					
0	No Error					

**SD Host Controller Simplified Specification Version 2.00**

07	RW1C	<p><b>Current Limit Error</b> By setting the <b>SD Bus Power</b> bit in the <i>Power Control</i> 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.</p> <table border="1"> <tr> <td>1</td> <td>Power fail</td> </tr> <tr> <td>0</td> <td>No Error</td> </tr> </table>	1	Power fail	0	No Error
1	Power fail					
0	No Error					
06	RW1C	<p><b>Data End Bit Error</b> Occurs either when detecting 0 at the end bit position of read data which uses the <b>DAT</b> line or at the end bit position of the CRC Status.</p> <table border="1"> <tr> <td>1</td> <td>Error</td> </tr> <tr> <td>0</td> <td>No Error</td> </tr> </table>	1	Error	0	No Error
1	Error					
0	No Error					
05	RW1C	<p><b>Data CRC Error</b> Occurs when detecting CRC error when transferring read data which uses the <b>DAT</b> line or when detecting the Write CRC status having a value of other than "010".</p> <table border="1"> <tr> <td>1</td> <td>Error</td> </tr> <tr> <td>0</td> <td>No Error</td> </tr> </table>	1	Error	0	No Error
1	Error					
0	No Error					
04	RW1C	<p><b>Data Timeout Error</b> This bit is set 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.</p> <table border="1"> <tr> <td>1</td> <td>Time out</td> </tr> <tr> <td>0</td> <td>No Error</td> </tr> </table>	1	Time out	0	No Error
1	Time out					
0	No Error					
03	RW1C	<p><b>Command Index Error</b> This bit is set if a Command Index error occurs in the command response.</p> <table border="1"> <tr> <td>1</td> <td>Error</td> </tr> <tr> <td>0</td> <td>No Error</td> </tr> </table>	1	Error	0	No Error
1	Error					
0	No Error					
02	RW1C	<p><b>Command End Bit Error</b> This bit is set when detecting that the end bit of a command response is 0.</p> <table border="1"> <tr> <td>1</td> <td>End Bit Error Generated</td> </tr> <tr> <td>0</td> <td>No Error</td> </tr> </table>	1	End Bit Error Generated	0	No Error
1	End Bit Error Generated					
0	No Error					

01	RW1C	<p><b>Command CRC Error</b>  <b>Command CRC Error</b> is generated in two cases.  If a response is returned and the <b>Command Timeout Error</b> is set to 0 (indicating no timeout), this bit is set to 1 when detecting a CRC error in the command response.  The Host Controller detects a <b>CMD</b> line conflict by monitoring the <b>CMD</b> line when a command is issued. If the Host Controller drives the <b>CMD</b> line to 1 level, but detects 0 level on the <b>CMD</b> line at the next SD clock edge, then the Host Controller shall abort the command (Stop driving <b>CMD</b> line) and set this bit to 1. The Command Timeout Error shall also be set to 1 to distinguish <b>CMD</b> line conflict (Refer to Table 2-25).</p> <table border="1" data-bbox="500 573 954 638"> <tr> <td>1</td> <td>CRC Error Generated.</td> </tr> <tr> <td>0</td> <td>No Error</td> </tr> </table>	1	CRC Error Generated.	0	No Error
1	CRC Error Generated.					
0	No Error					
00	RW1C	<p><b>Command Timeout Error</b>  This bit is set only if no response is returned within 64 SD clock cycles from the end bit of the command. If the Host Controller detects a <b>CMD</b> line conflict, in which case <b>Command CRC Error shall</b> also be set as shown in Table 2-25, this bit shall be set without waiting for 64 SD clock cycles because the command will be aborted by the Host Controller.</p> <table border="1" data-bbox="500 863 786 928"> <tr> <td>1</td> <td>Time out</td> </tr> <tr> <td>0</td> <td>No Error</td> </tr> </table>	1	Time out	0	No Error
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	<b>CMD</b> line conflict

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

**SD Host Controller Simplified Specification Version 2.00****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	Register Field Explanation				
15	RO	<b>Fixed to 0</b> The Host Driver shall control error interrupts using the <i>Error Interrupt Status Enable</i> register.				
14-09	Rsvd	<b>Reserved</b>				
08	RW	<b>Card Interrupt Status Enable</b> If this bit is set to 0, the Host Controller shall clear interrupt request to the System. The <b>Card Interrupt</b> detection is stopped when this bit is cleared and restarted when this bit is set to 1. The Host Driver should clear the <b>Card Interrupt Status Enable</b> before servicing the <b>Card Interrupt</b> and should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts. <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
07	RW	<b>Card Removal Status Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
06	RW	<b>Card Insertion Status Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
05	RW	<b>Buffer Read Ready Status Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
04	RW	<b>Buffer Write Ready Status Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
03	RW	<b>DMA Interrupt Status Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					

**SD Host Controller Simplified Specification Version 2.00**

02	RW	<b>Block Gap Event Status Enable</b>	
		1	Enabled
		0	Masked
01	RW	<b>Transfer Complete Status Enable</b>	
		1	Enabled
		0	Masked
00	RW	<b>Command Complete Status Enable</b>	
		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.

**SD Host Controller Simplified Specification Version 2.00****2.2.20. Error Interrupt Status Enable Register (Offset 036h)**

Setting to 1 enables Interrupt Status.

D15	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
Vendor Specific Error Status Enable			Rsvd	ADMA Error Status Enable	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	Register Field Explanation				
15-12	RW	<b>Vendor Specific Error Status Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
11-10	Rsvd	<b>Reserved</b>				
09	RW	<b>ADMA Error Status Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
08	RW	<b>Auto CMD12 Error Status Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
07	RW	<b>Current Limit Error Status Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
06	RW	<b>Data End Bit Error Status Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
05	RW	<b>Data CRC Error Status Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
04	RW	<b>Data Timeout Error Status Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
03	RW	<b>Command Index Error Status Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					

**SD Host Controller Simplified Specification Version 2.00**

02	RW	<b>Command End Bit Error Status Enable</b>	
		1	Enabled
		0	Masked
01	RW	<b>Command CRC Error Status Enable</b>	
		1	Enabled
		0	Masked
00	RW	<b>Command Timeout Error Status Enable</b>	
		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.



**SD Host Controller Simplified Specification Version 2.00****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 same 1 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	Register Field Explanation				
15	RO	<b>Fixed to 0</b> The Host Driver shall control error interrupts using the <i>Error Interrupt Signal Enable</i> register.				
14-09	Rsvd	<b>Reserved</b>				
08	RW	<b>Card Interrupt Signal Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
07	RW	<b>Card Removal Signal Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
06	RW	<b>Card Insertion Signal Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
05	RW	<b>Buffer Read Ready Signal Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
04	RW	<b>Buffer Write Ready Signal Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
03	RW	<b>DMA Interrupt Signal Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
02	RW	<b>Block Gap Event Signal Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
01	RW	<b>Transfer Complete Signal Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					

00	RW	<b>Command Complete Signal Enable</b>	
		1	Enabled
		0	Masked

**Table 2-28 : Normal Interrupt Signal Enable Register**

**SD Host Controller Simplified Specification Version 2.00****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	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
Vendor Specific Error Signal			Rsvd	ADMA Error Signal Enable	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	Register Field Explanation				
15-12	RW	<b>Vendor Specific Error Signal Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
11-10	Rsvd	<b>Reserved</b>				
09	RW	<b>ADMA Error Signal Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
08	RW	<b>Auto CMD12 Error Signal Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
07	RW	<b>Current Limit Error Signal Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
06	RW	<b>Data End Bit Error Signal Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
05	RW	<b>Data CRC Error Signal Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
04	RW	<b>Data Timeout Error Signal Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					
03	RW	<b>Command Index Error Signal Enable</b> <table border="1"> <tr> <td>1</td> <td>Enabled</td> </tr> <tr> <td>0</td> <td>Masked</td> </tr> </table>	1	Enabled	0	Masked
1	Enabled					
0	Masked					

**SD Host Controller Simplified Specification Version 2.00**

02	RW	<b>Command End Bit Error Signal Enable</b>	
		1	Enabled
		0	Masked
01	RW	<b>Command CRC Error Signal Enable</b>	
		1	Enabled
		0	Masked
00	RW	<b>Command Timeout Error Signal Enable</b>	
		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	Register Field Explanation				
15-08	Rsvd	<b>Reserved</b>				
07	ROC	<p><b>Command Not Issued By Auto CMD12 Error</b> Setting this bit to 1 means CMD_wo_DAT is not executed due to an Auto CMD12 Error (D04-D01) in this register.</p> <table border="1"> <tr> <td>1</td> <td>Not Issued</td> </tr> <tr> <td>0</td> <td>No error</td> </tr> </table>	1	Not Issued	0	No error
1	Not Issued					
0	No error					
06-05	Rsvd	<b>Reserved</b>				
04	ROC	<p><b>Auto CMD12 Index Error</b> This bit is set if the Command Index error occurs in response to a command.</p> <table border="1"> <tr> <td>1</td> <td>Error</td> </tr> <tr> <td>0</td> <td>No error</td> </tr> </table>	1	Error	0	No error
1	Error					
0	No error					
03	ROC	<p><b>Auto CMD12 End Bit Error</b> This bit is set when detecting that the end bit of command response is 0.</p> <table border="1"> <tr> <td>1</td> <td>End Bit Error Generated</td> </tr> <tr> <td>0</td> <td>No error</td> </tr> </table>	1	End Bit Error Generated	0	No error
1	End Bit Error Generated					
0	No error					
02	ROC	<p><b>Auto CMD12 CRC Error</b> This bit is set when detecting a CRC error in the command response.</p> <table border="1"> <tr> <td>1</td> <td>CRC Error Generated</td> </tr> <tr> <td>0</td> <td>No error</td> </tr> </table>	1	CRC Error Generated	0	No error
1	CRC Error Generated					
0	No error					
01	ROC	<p><b>Auto CMD12 Timeout Error</b> This bit is set if no response is returned within 64 <b>SDCLK</b> cycles from the end bit of command. If this bit is set to 1, the other error status bits (D04-D02) are meaningless.</p> <table border="1"> <tr> <td>1</td> <td>Time out</td> </tr> <tr> <td>0</td> <td>No error</td> </tr> </table>	1	Time out	0	No error
1	Time out					
0	No error					

00	ROC	<p><b>Auto CMD12 Not Executed</b></p> <p>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.</p> <table border="1"> <tr> <td>1</td> <td>Not executed</td> </tr> <tr> <td>0</td> <td>Executed</td> </tr> </table>	1	Not executed	0	Executed
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 CRC Error and 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 make any effect on interrupt because it is set when one of bits D01 to D04 is set to 1.

**SD Host Controller Simplified Specification Version 2.00****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 D29	D28	D27	D26	D25	D24	D23	D22	D21	D20	D19	D18	D17-16	D15-14	D13 D08	D07	D06	D05 D00
Rsvd	64-bit System Bus Support	Rsvd	Voltage Support 1.8V	Voltage Support 3.0V	Voltage Support 3.3V	Suspend/Resume Support	SDMA Support	High Speed Support	Rsvd	ADMA2 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	Register Field Explanation				
63-32	Rsvd	<b>Reserved</b>				
31-29	Rsvd	<b>Reserved</b>				
28	Hwlnit	<b>64-bit System Bus Support</b> Setting 1 to this bit indicates that the Host Controller supports 64-bit address descriptor mode and is connected to 64-bit address system bus.				
27	Rsvd	<b>Reserved</b> Reserved for voltage support				
26	Hwlnit	<b>Voltage Support 1.8V</b> <table border="1"> <tr> <td>1</td> <td>1.8V Supported</td> </tr> <tr> <td>0</td> <td>1.8V Not Supported</td> </tr> </table>	1	1.8V Supported	0	1.8V Not Supported
1	1.8V Supported					
0	1.8V Not Supported					
25	Hwlnit	<b>Voltage Support 3.0V</b> <table border="1"> <tr> <td>1</td> <td>3.0V Supported</td> </tr> <tr> <td>0</td> <td>3.0V Not Supported</td> </tr> </table>	1	3.0V Supported	0	3.0V Not Supported
1	3.0V Supported					
0	3.0V Not Supported					
24	Hwlnit	<b>Voltage Support 3.3V</b> <table border="1"> <tr> <td>1</td> <td>3.3V Supported</td> </tr> <tr> <td>0</td> <td>3.3V Not Supported</td> </tr> </table>	1	3.3V Supported	0	3.3V Not Supported
1	3.3V Supported					
0	3.3V Not Supported					

**Table 2-32 : Capabilities Register (Part 1)****Implementation Note:**

The Host System should support 3.3V. 1.8V can be used when low voltage operation is defined in a future version of Physical Layer Specification. 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.

**SD Host Controller Simplified Specification Version 2.00**

Location	Attrib									
23	Hwlnit	<p><b>Suspend/Resume Support</b> This bit indicates whether the Host Controller supports Suspend / Resume functionality. If this bit is 0, the Host Driver shall not issue either Suspend or Resume commands because the Suspend and Resume mechanism (Refer to 1.6) is not supported.</p> <table border="1"> <tr> <td>1</td> <td>Supported</td> </tr> <tr> <td>0</td> <td>Not Supported</td> </tr> </table>	1	Supported	0	Not Supported				
1	Supported									
0	Not Supported									
22	Hwlnit	<p><b>SDMA Support</b> This bit indicates whether the Host Controller is capable of using SDMA to transfer data between system memory and the Host Controller directly.</p> <table border="1"> <tr> <td>1</td> <td>SDMA Supported</td> </tr> <tr> <td>0</td> <td>SDMA not Supported</td> </tr> </table>	1	SDMA Supported	0	SDMA not Supported				
1	SDMA Supported									
0	SDMA not Supported									
21	Hwlnit	<p><b>High Speed Support</b> 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.</p> <table border="1"> <tr> <td>1</td> <td>High Speed Supported</td> </tr> <tr> <td>0</td> <td>High Speed not Supported</td> </tr> </table>	1	High Speed Supported	0	High Speed not Supported				
1	High Speed Supported									
0	High Speed not Supported									
20	Hwlnit	<p><b>Reserved (New assignment is not allowed)</b> Refer to Appendix C.</p>								
19	Hwlnit	<p><b>ADMA2 Support</b> This bit indicates whether the Host Controller is capable of using ADMA2.</p> <table border="1"> <tr> <td>1</td> <td>ADMA2 Supported</td> </tr> <tr> <td>0</td> <td>ADMA2 not Supported</td> </tr> </table>	1	ADMA2 Supported	0	ADMA2 not Supported				
1	ADMA2 Supported									
0	ADMA2 not Supported									
18	Hwlnit	<p><b>Reserved (New assignment is not allowed)</b></p>								
17-16	Hwlnit	<p><b>Max Block Length</b> 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. It is noted that transfer block length shall be always 512 bytes for SD Memory Cards regardless this field.</p> <table border="1"> <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>	00	512(byte)	01	1024	10	2048	11	Reserved
00	512(byte)									
01	1024									
10	2048									
11	Reserved									
15-14	Rsvd	<p><b>Reserved</b></p>								



**SD Host Controller Simplified Specification Version 2.00**

13-08	Hwlnit	<p><b>Base Clock Frequency For SD Clock</b></p> <p>This value indicates the base (maximum) clock frequency for the SD Clock. Unit values are 1MHz. If the real frequency is 16.5MHz, the larger value shall be set 01 0001b (17MHz) because the Host Driver use this value to calculate the clock divider value (Refer to the <b>SDCLK Frequency Select</b> in the <i>Clock Control</i> 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.</p> <table border="1"> <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>	Not 0	1MHz to 63MHz	00 0000b	Get information via another method
Not 0	1MHz to 63MHz					
00 0000b	Get information via another method					
07	Hwlnit	<p><b>Timeout Clock Unit</b></p> <p>This bit shows the unit of base clock frequency used to detect <b>Data Timeout Error</b>.</p> <table border="1"> <tr> <td>0</td> <td>KHz</td> </tr> <tr> <td>1</td> <td>MHz</td> </tr> </table>	0	KHz	1	MHz
0	KHz					
1	MHz					
06	Rsvd	<b>Reserved</b>				
05-00	Hwlnit	<p><b>Timeout Clock Frequency</b></p> <p>This bit shows the base clock frequency used to detect <b>Data Timeout Error</b>. The <b>Timeout Clock Unit</b> defines the unit of this field's value.</p> <p><b>Timeout Clock Unit</b> =0 [KHz] unit: 1KHz to 63KHz  <b>Timeout Clock Unit</b> =1 [MHz] unit: 1MHz to 63MHz</p> <table border="1"> <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>	Not 0	1KHz to 63KHz or 1MHz to 63MHz	00 0000b	Get information via another method
Not 0	1KHz to 63KHz or 1MHz to 63MHz					
00 0000b	Get information via another method					

**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				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	Register Field Explanation
63-56	Rsvd	<b>Reserved</b>
55-48	Rsvd	<b>Reserved</b>
47-40	Rsvd	<b>Reserved</b>
39-32	Rsvd	<b>Reserved</b>
31-24	Rsvd	<b>Reserved</b>
23-16	Hwlnit	<b>Maximum Current for 1.8V</b>
15-08	Hwlnit	<b>Maximum Current for 3.0V</b>
07-00	Hwlnit	<b>Maximum Current for 3.3V</b>

**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	Register Field Explanation										
15-08	Rsvd	<b>Reserved</b>										
07-00	ROC	<p><b>Interrupt Signal For Each Slot</b></p> <p>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 <b>Software Reset For All</b>, the interrupt signal shall be de-asserted and this status shall read 00h.</p> <table border="1" style="margin-left: 20px;"> <tr> <td>Bit 00</td> <td>Slot 1</td> </tr> <tr> <td>Bit 01</td> <td>Slot 2</td> </tr> <tr> <td>Bit 02</td> <td>Slot 3</td> </tr> <tr> <td>.....</td> <td>.....</td> </tr> <tr> <td>Bit 07</td> <td>Slot 8</td> </tr> </table>	Bit 00	Slot 1	Bit 01	Slot 2	Bit 02	Slot 3	.....	.....	Bit 07	Slot 8
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	Register Field Explanation						
15-08	Hwlnit	<p><b>Vendor Version Number</b> This status is reserved for the vendor version number. The Host Driver should not use this status.</p>						
07-00	Hwlnit	<p><b>Specification Version Number</b> This status indicates the Host Controller Spec. Version. The upper and lower 4-bits indicate the version.</p> <table border="1"> <tr> <td>00</td> <td>SD Host Specification Version 1.00</td> </tr> <tr> <td>01</td> <td>SD Host Specification Version 2.00 Including the feature of the ADMA and Test Register,</td> </tr> <tr> <td>others</td> <td>Reserved</td> </tr> </table>	00	SD Host Specification Version 1.00	01	SD Host Specification Version 2.00 Including the feature of the ADMA and Test Register,	others	Reserved
00	SD Host Specification Version 1.00							
01	SD Host Specification Version 2.00 Including the feature of the ADMA and Test Register,							
others	Reserved							

Table 2-37 : Host Controller Version

### 2.2.28. Force Event Register for Error Interrupt Status (Offset 052h)

The *Force Event* Register is not a physically implemented register. Rather, it is an address at which the *Error Interrupt Status* register can be written. The effect of a write to this address will be reflected in the *Error Interrupt Status* Register if the corresponding bit of the *Error Interrupt Status Enable* Register is set.  
 Writing 1 : set each bit of the *Error Interrupt Status* Register  
 Writing 0 : no effect

Note: By setting this register, the Error Interrupt can be set in the *Error Interrupt Status* register. In order to generate interrupt signal, both the *Error Interrupt Status Enable* and **Error Interrupt Signal Enable** shall be set.

D15	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
Force Event for Vendor Specific Error Status		Rsvd		Force Event for ADMA Error	Force Event for Auto CMD12 Error	Force Event for Current limit Error	Force Event for Data End Bit Error	Force Event for Data CRC Error	Force Event for Data Timeout Error	Force Event for Command Index Error	Force Event for Command End Bit Error	Force Event for Command CRC Error	Force Event for Command Timeout Error

Figure 2-32 : Force Event Register for Error Interrupt Status

**SD Host Controller Simplified Specification Version 2.00**

Location	Attrib	Register Field Explanation				
15-12	WO	<b>Force Event for Vendor Specific Error Status</b> Additional status bits can be defined in this register by the vendor. <table border="1" data-bbox="443 310 886 380"> <tr> <td>1</td> <td>Interrupt is generated</td> </tr> <tr> <td>0</td> <td>No Interrupt</td> </tr> </table>	1	Interrupt is generated	0	No Interrupt
1	Interrupt is generated					
0	No Interrupt					
11-10	Rsvd	<b>Reserved</b>				
09	WO	<b>Force Event for ADMA Error</b> <table border="1" data-bbox="443 478 886 548"> <tr> <td>1</td> <td>Interrupt is generated</td> </tr> <tr> <td>0</td> <td>No Interrupt</td> </tr> </table>	1	Interrupt is generated	0	No Interrupt
1	Interrupt is generated					
0	No Interrupt					
08	WO	<b>Force Event for Auto CMD12 Error</b> <table border="1" data-bbox="443 611 886 680"> <tr> <td>1</td> <td>Interrupt is generated</td> </tr> <tr> <td>0</td> <td>No Interrupt</td> </tr> </table>	1	Interrupt is generated	0	No Interrupt
1	Interrupt is generated					
0	No Interrupt					
07	WO	<b>Force Event for Current Limit Error</b> <table border="1" data-bbox="443 743 886 812"> <tr> <td>1</td> <td>Interrupt is generated</td> </tr> <tr> <td>0</td> <td>No Interrupt</td> </tr> </table>	1	Interrupt is generated	0	No Interrupt
1	Interrupt is generated					
0	No Interrupt					
06	WO	<b>Force Event for Data End Bit Error</b> <table border="1" data-bbox="443 875 886 945"> <tr> <td>1</td> <td>Interrupt is generated</td> </tr> <tr> <td>0</td> <td>No Interrupt</td> </tr> </table>	1	Interrupt is generated	0	No Interrupt
1	Interrupt is generated					
0	No Interrupt					
05	WO	<b>Force Event for Data CRC Error</b> <table border="1" data-bbox="443 1008 886 1077"> <tr> <td>1</td> <td>Interrupt is generated</td> </tr> <tr> <td>0</td> <td>No Interrupt</td> </tr> </table>	1	Interrupt is generated	0	No Interrupt
1	Interrupt is generated					
0	No Interrupt					
04	WO	<b>Force Event for Data Timeout Error</b> <table border="1" data-bbox="443 1140 886 1209"> <tr> <td>1</td> <td>Interrupt is generated</td> </tr> <tr> <td>0</td> <td>No Interrupt</td> </tr> </table>	1	Interrupt is generated	0	No Interrupt
1	Interrupt is generated					
0	No Interrupt					
03	WO	<b>Force Event for Command Index Error</b> <table border="1" data-bbox="443 1272 886 1341"> <tr> <td>1</td> <td>Interrupt is generated</td> </tr> <tr> <td>0</td> <td>No Interrupt</td> </tr> </table>	1	Interrupt is generated	0	No Interrupt
1	Interrupt is generated					
0	No Interrupt					
02	WO	<b>Force Event for Command End Bit Error</b> <table border="1" data-bbox="443 1404 886 1474"> <tr> <td>1</td> <td>Interrupt is generated</td> </tr> <tr> <td>0</td> <td>No Interrupt</td> </tr> </table>	1	Interrupt is generated	0	No Interrupt
1	Interrupt is generated					
0	No Interrupt					
01	WO	<b>Force Event for Command CRC Error</b> <table border="1" data-bbox="443 1537 886 1606"> <tr> <td>1</td> <td>Interrupt is generated</td> </tr> <tr> <td>0</td> <td>No Interrupt</td> </tr> </table>	1	Interrupt is generated	0	No Interrupt
1	Interrupt is generated					
0	No Interrupt					
00	WO	<b>Force Event for Command Timeout Error</b> <table border="1" data-bbox="443 1669 886 1738"> <tr> <td>1</td> <td>Interrupt is generated</td> </tr> <tr> <td>0</td> <td>No Interrupt</td> </tr> </table>	1	Interrupt is generated	0	No Interrupt
1	Interrupt is generated					
0	No Interrupt					

**Table 2-38 : Force Event for Error Interrupt Status Register**

**2.2.29. Force Event Register for Auto CMD12 Error Status (Offset 050h)**

The Force Event Register is not a physically implemented register. Rather, it is an address at which the Auto CMD12 Error Status Register can be written.

Writing 1 : set each bit of the Auto CMD12 Error Status Register

Writing 0 : no effect

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

**Figure 2-33 : Force Event Register for Auto CMD12 Error Status**

Location	Attrib	Register Field Explanation				
15-08	Rsvd	<b>Reserved</b>				
07	WO	<b>Force Event for Command Not Issued By Auto CMD12 Error</b> <table border="1"> <tr> <td>1</td> <td>Interrupt is generated</td> </tr> <tr> <td>0</td> <td>No Interrupt</td> </tr> </table>	1	Interrupt is generated	0	No Interrupt
1	Interrupt is generated					
0	No Interrupt					
06-05	Rsvd	<b>Reserved</b>				
04	WO	<b>Force Event for Auto CMD12 Index Error</b> <table border="1"> <tr> <td>1</td> <td>Interrupt is generated</td> </tr> <tr> <td>0</td> <td>No Interrupt</td> </tr> </table>	1	Interrupt is generated	0	No Interrupt
1	Interrupt is generated					
0	No Interrupt					
03	WO	<b>Force Event for Auto CMD12 End Bit Error</b> <table border="1"> <tr> <td>1</td> <td>Interrupt is generated</td> </tr> <tr> <td>0</td> <td>No Interrupt</td> </tr> </table>	1	Interrupt is generated	0	No Interrupt
1	Interrupt is generated					
0	No Interrupt					
02	WO	<b>Force Event for Auto CMD12 CRC Error</b> <table border="1"> <tr> <td>1</td> <td>Interrupt is generated</td> </tr> <tr> <td>0</td> <td>No Interrupt</td> </tr> </table>	1	Interrupt is generated	0	No Interrupt
1	Interrupt is generated					
0	No Interrupt					
01	WO	<b>Force Event for Auto CMD12 Timeout Error</b> <table border="1"> <tr> <td>1</td> <td>Interrupt is generated</td> </tr> <tr> <td>0</td> <td>No Interrupt</td> </tr> </table>	1	Interrupt is generated	0	No Interrupt
1	Interrupt is generated					
0	No Interrupt					
00	WO	<b>Force Event for Auto CMD12 Not Executed</b> <table border="1"> <tr> <td>1</td> <td>Interrupt is generated</td> </tr> <tr> <td>0</td> <td>No Interrupt</td> </tr> </table>	1	Interrupt is generated	0	No Interrupt
1	Interrupt is generated					
0	No Interrupt					

**Table 2-39 : Force Event Register for Auto CMD12 Error Status**

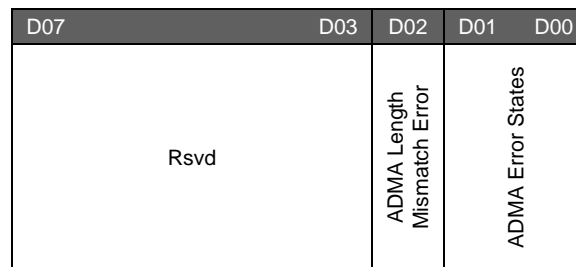
**SD Host Controller Simplified Specification Version 2.00****2.2.30. ADMA Error Status Register (Offset 054h)**

When **ADMA Error** Interrupt is occurred, the **ADMA Error States** field in this register holds the ADMA state and the *ADMA System Address* Register holds the address around the error descriptor. For recovering the error, the Host Driver requires the ADMA state to identify the error descriptor address as follows:

- ST\_STOP: Previous location set in the ADMA System Address register is the error descriptor address  
 ST\_FDS: Current location set in the ADMA System Address register is the error descriptor address  
 ST\_CADR: This state is never set because do not generate ADMA error in this state.  
 ST\_TFR: Previous location set in the ADMA System Address register is the error descriptor address

In case of write operation, the Host Driver should use ACMD22 to get the number of written block rather than using this information, since unwritten data may exist in the Host Controller.

The Host Controller generates the **ADMA Error** Interrupt when it detects invalid descriptor data (Valid=0) at the ST\_FDS state. In this case, ADMA Error State indicates that an error occurs at ST\_FDS state. The Host Driver may find that the Valid bit is not set in the error descriptor.

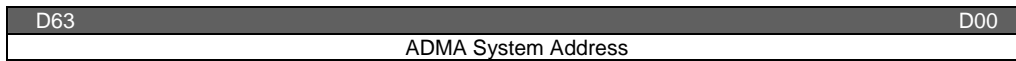
**Figure 2-34 : ADMA Error Status Register**

Location	Attrib	Register Field Explanation															
07-03	Rsvd	<b>Reserved</b>															
02	RW	<p><b>ADMA Length Mismatch Error</b>            This error occurs in the following 2 cases.</p> <p>(1) While Block Count Enable being set, the total data length specified by the Descriptor table is different from that specified by the Block Count and Block Length.</p> <p>(2) Total data length can not be divided by the block length.</p> <table border="1" style="margin-left: 20px;"> <tr> <td>1</td> <td>Error</td> </tr> <tr> <td>0</td> <td>No Error</td> </tr> </table>	1	Error	0	No Error											
1	Error																
0	No Error																
01-00	RW	<p><b>ADMA Error State</b>            This field indicates the state of ADMA when error is occurred during ADMA data transfer. This field never indicates "10" because ADMA never stops in this state.</p> <table border="1" style="margin-left: 20px;"> <tr> <td>D01 – D00</td> <td>ADMA Error State when error is occurred</td> <td>Contents of SYS_SDR register</td> </tr> <tr> <td>00</td> <td>ST_STOP (Stop DMA)</td> <td>Points next of the error descriptor</td> </tr> <tr> <td>01</td> <td>ST_FDS (Fetch Descriptor)</td> <td>Points the error descriptor</td> </tr> <tr> <td>10</td> <td>Never set this state</td> <td>(Not used)</td> </tr> <tr> <td>11</td> <td>ST_TFR (Transfer Data)</td> <td>Points the next of the error descriptor</td> </tr> </table>	D01 – D00	ADMA Error State when error is occurred	Contents of SYS_SDR register	00	ST_STOP (Stop DMA)	Points next of the error descriptor	01	ST_FDS (Fetch Descriptor)	Points the error descriptor	10	Never set this state	(Not used)	11	ST_TFR (Transfer Data)	Points the next of the error descriptor
D01 – D00	ADMA Error State when error is occurred	Contents of SYS_SDR register															
00	ST_STOP (Stop DMA)	Points next of the error descriptor															
01	ST_FDS (Fetch Descriptor)	Points the error descriptor															
10	Never set this state	(Not used)															
11	ST_TFR (Transfer Data)	Points the next of the error descriptor															

**Table 2-40 : ADMA Error Status Register**

**2.2.31. ADMA System Address Register (Offset 058h)**

This register contains the physical Descriptor address used for ADMA data transfer.



**Figure 2-35 : ADMA System Address Register**

Location	Attrib	Register Field Explanation																												
63-00	RW	<p><b>ADMA System Address</b></p> <p>This register holds byte address of executing command of the Descriptor table. 32-bit Address Descriptor uses lower 32-bit of this register. At the start of ADMA, the Host Driver shall set start address of the Descriptor table. The ADMA increments this register address, which points to next line, when every fetching a Descriptor line. When the ADMA Error Interrupt is generated, this register shall hold valid Descriptor address depending on the ADMA state. The Host Driver shall program Descriptor Table on 32-bit boundary and set 32-bit boundary address to this register. ADMA2 ignores lower 2-bit of this register and assumes it to be 00b.</p> <p>32-bit Address ADMA</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%;">Register Value</th> <th style="width: 50%;">32-bit System Address</th> </tr> </thead> <tbody> <tr> <td>xxxxxxxx 00000000h</td> <td>00000000h</td> </tr> <tr> <td>xxxxxxxx 00000004h</td> <td>00000004h</td> </tr> <tr> <td>xxxxxxxx 00000008h</td> <td>00000008h</td> </tr> <tr> <td>xxxxxxxx 0000000Ch</td> <td>0000000Ch</td> </tr> <tr> <td style="text-align: center;">.....</td> <td style="text-align: center;">.....</td> </tr> <tr> <td>xxxxxxxx FFFFFFFFCh</td> <td>FFFFFFFCh</td> </tr> </tbody> </table> <p>64-bit Address ADMA</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%;">Register Value</th> <th style="width: 50%;">64-bit System Address</th> </tr> </thead> <tbody> <tr> <td>00000000 00000000h</td> <td>00000000 00000000h</td> </tr> <tr> <td>00000000 00000004h</td> <td>00000000 00000004h</td> </tr> <tr> <td>00000000 00000008h</td> <td>00000000 00000008h</td> </tr> <tr> <td>00000000 0000000Ch</td> <td>00000000 0000000Ch</td> </tr> <tr> <td style="text-align: center;">.....</td> <td style="text-align: center;">.....</td> </tr> <tr> <td>FFFFFFFF FFFFFFFFCh</td> <td>FFFFFFFF FFFFFFFFCh</td> </tr> </tbody> </table>	Register Value	32-bit System Address	xxxxxxxx 00000000h	00000000h	xxxxxxxx 00000004h	00000004h	xxxxxxxx 00000008h	00000008h	xxxxxxxx 0000000Ch	0000000Ch	.....	.....	xxxxxxxx FFFFFFFFCh	FFFFFFFCh	Register Value	64-bit System Address	00000000 00000000h	00000000 00000000h	00000000 00000004h	00000000 00000004h	00000000 00000008h	00000000 00000008h	00000000 0000000Ch	00000000 0000000Ch	.....	.....	FFFFFFFF FFFFFFFFCh	FFFFFFFF FFFFFFFFCh
Register Value	32-bit System Address																													
xxxxxxxx 00000000h	00000000h																													
xxxxxxxx 00000004h	00000004h																													
xxxxxxxx 00000008h	00000008h																													
xxxxxxxx 0000000Ch	0000000Ch																													
.....	.....																													
xxxxxxxx FFFFFFFFCh	FFFFFFFCh																													
Register Value	64-bit System Address																													
00000000 00000000h	00000000 00000000h																													
00000000 00000004h	00000000 00000004h																													
00000000 00000008h	00000000 00000008h																													
00000000 0000000Ch	00000000 0000000Ch																													
.....	.....																													
FFFFFFFF FFFFFFFFCh	FFFFFFFF FFFFFFFFCh																													

**Table 2-41 : ADMA System Address Register**



### 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 3-24), 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

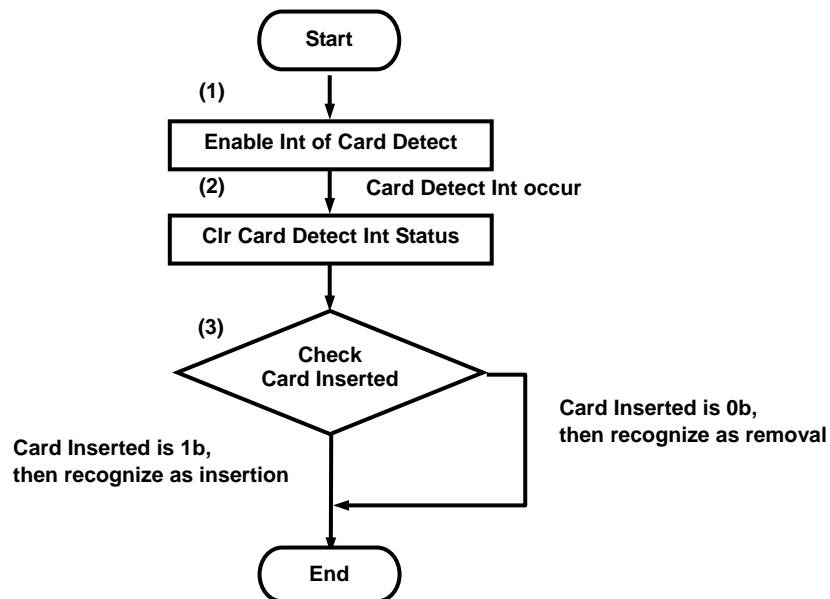


Figure 3-2: SD Card Detect Sequence

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

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

**SD Host Controller Simplified Specification Version 2.00**

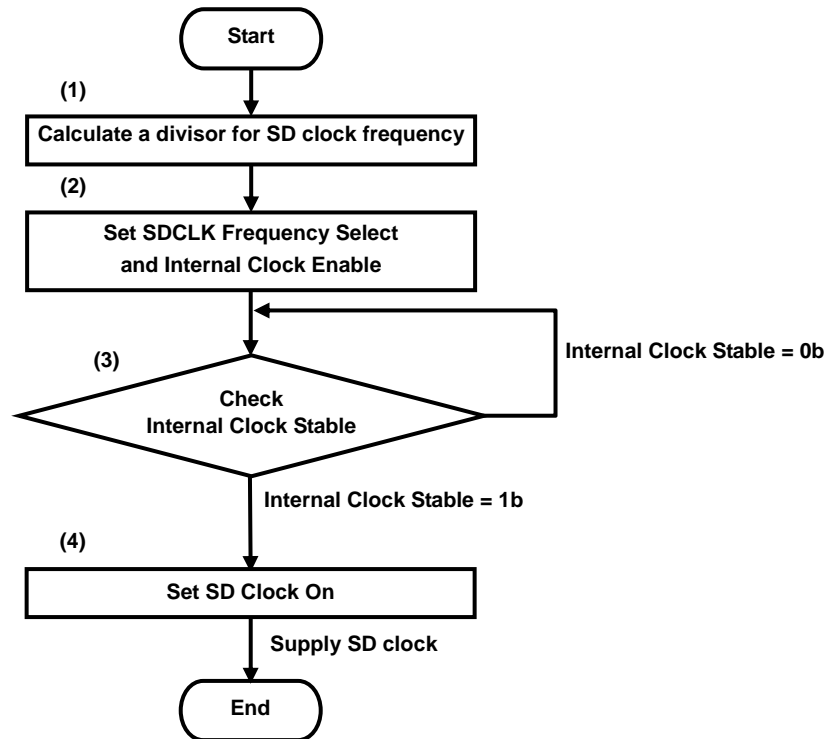
---

- (1) 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.
- (2) 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.

If miniSD adaptor is used for standard SD slot and miniSD card is inserted or extracted from the adaptor, card detect interrupt may not be generated. When host does not receive response to any commands, miniSD card is extracted or in idle state, the host should try to re-initialize the card. In this case, all card information shall be re-loaded.

## 3.2. SD Clock Control

### 3.2.1. SD Clock Supply Sequence

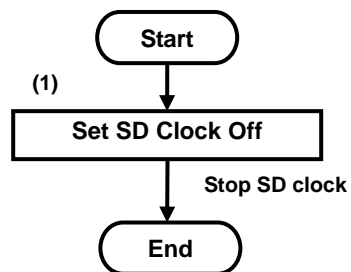


**Figure 3-3: SD Clock Supply Sequence**

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

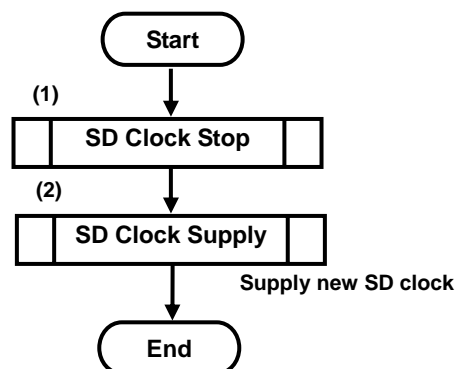


**Figure 3-4: SD Clock Stop Sequence**

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.

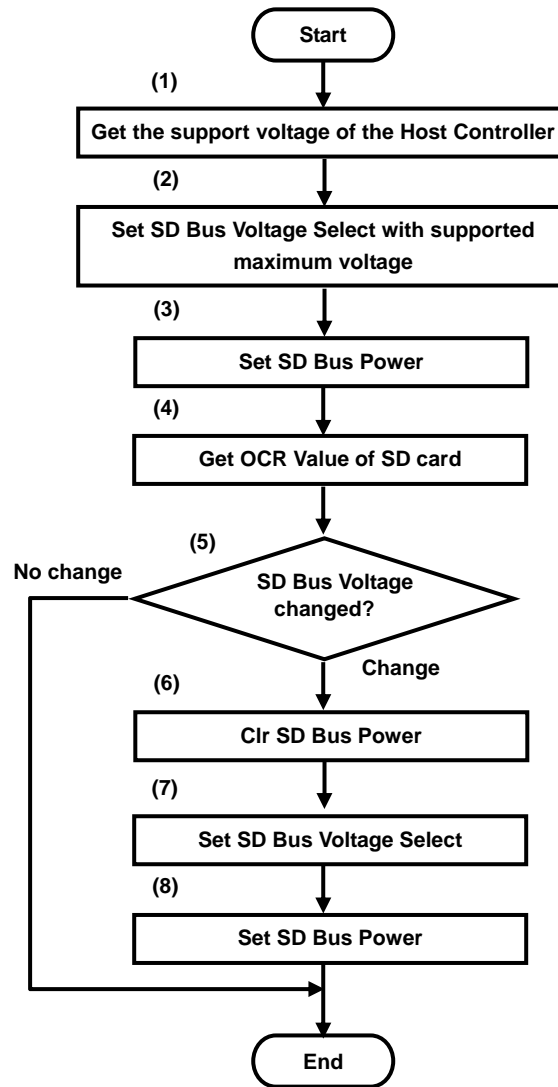


Figure 3-6: SD Bus Power Control Sequence

- (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

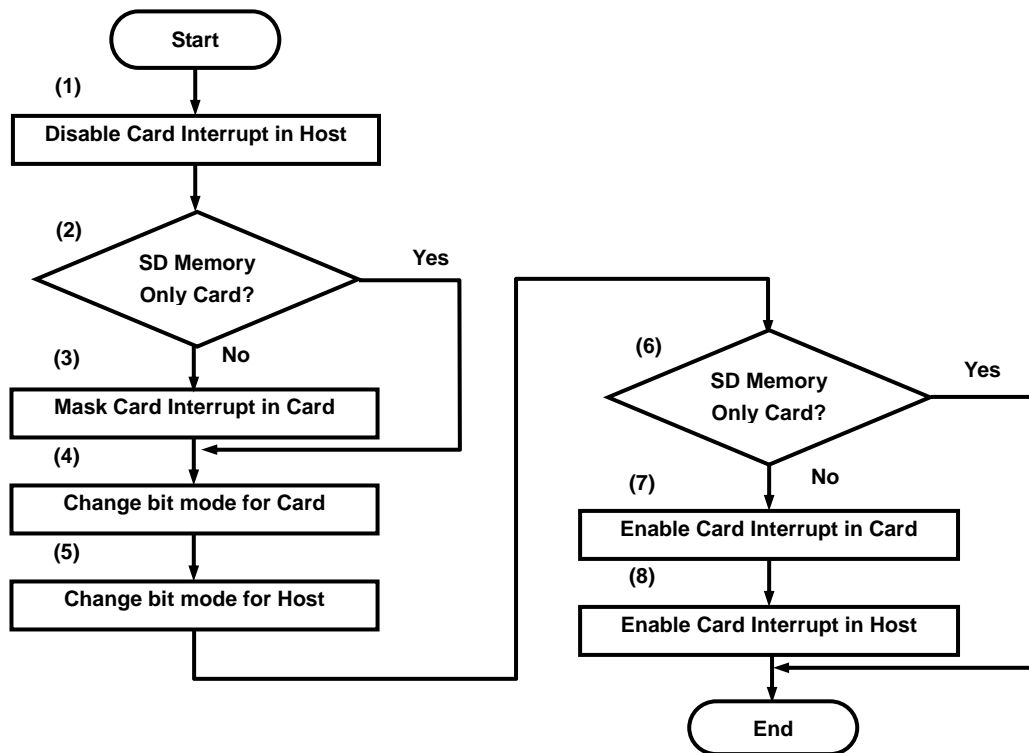


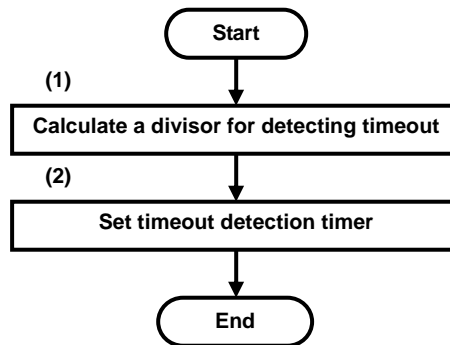
Figure 3-7: Change Bus Width Sequence

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 3.7.1 for how to generate CMD52.
- (4) Change the bus width mode for a SD card. SD Memory Card bus width is changed by ACMD6 and SDIO card bus width is changed 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.

Note that If the card is locked, bus width cannot be changed. Unlock the card is required before changing bus width.

### 3.5. Timeout Setting on DAT Line



**Figure 3-8: Timeout Setting Sequence**

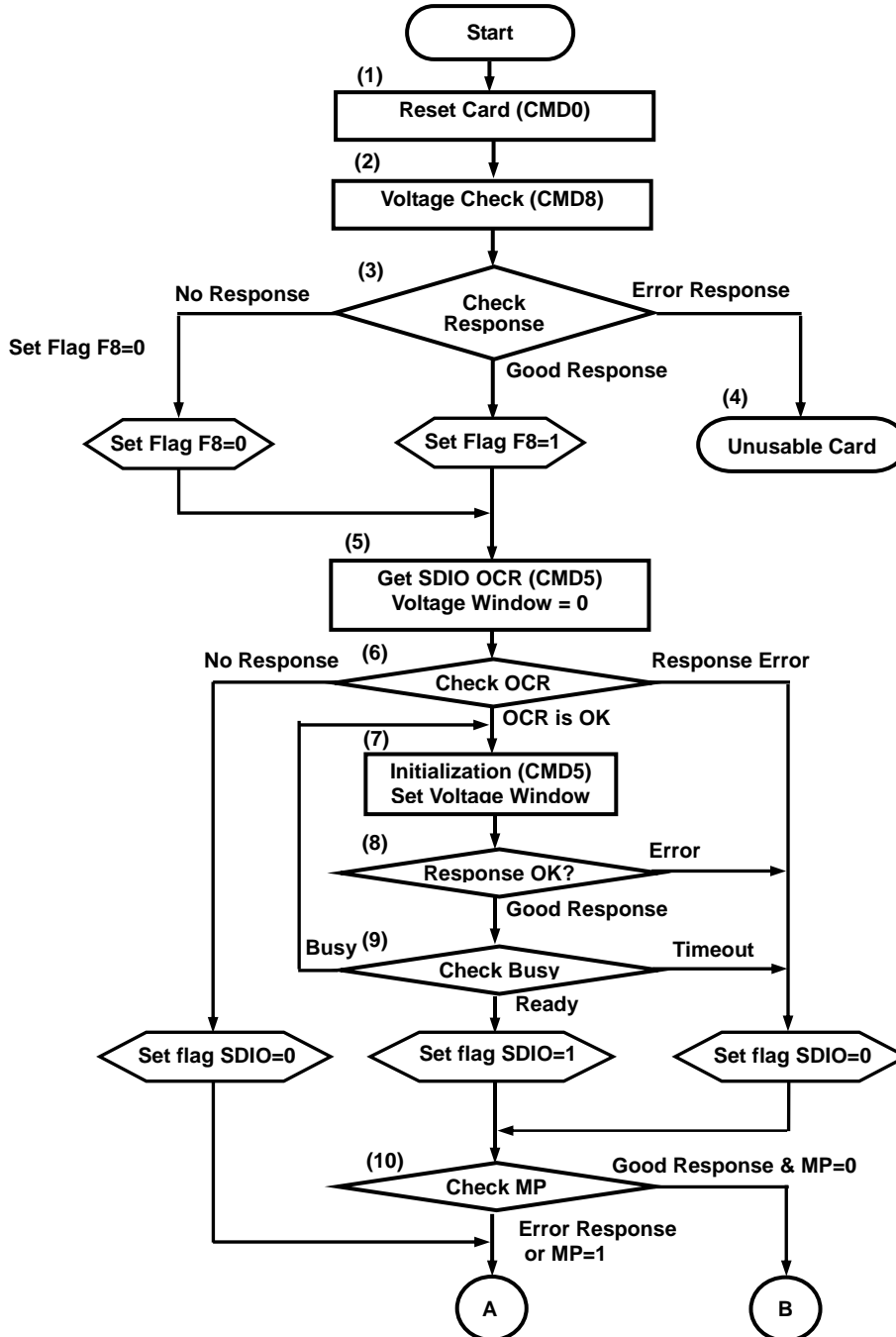
In order to detect timeout errors on DAT line, the Host Driver shall execute the following two steps before any SD transaction. For more information regarding SD transactions, refer to Section 3.7.2

- (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. Card Initialization and Identification

Figure 3-9 shows new initialization and card identification sequence for the Standard Capacity SD Memory Card and the High Capacity SD Memory Card.



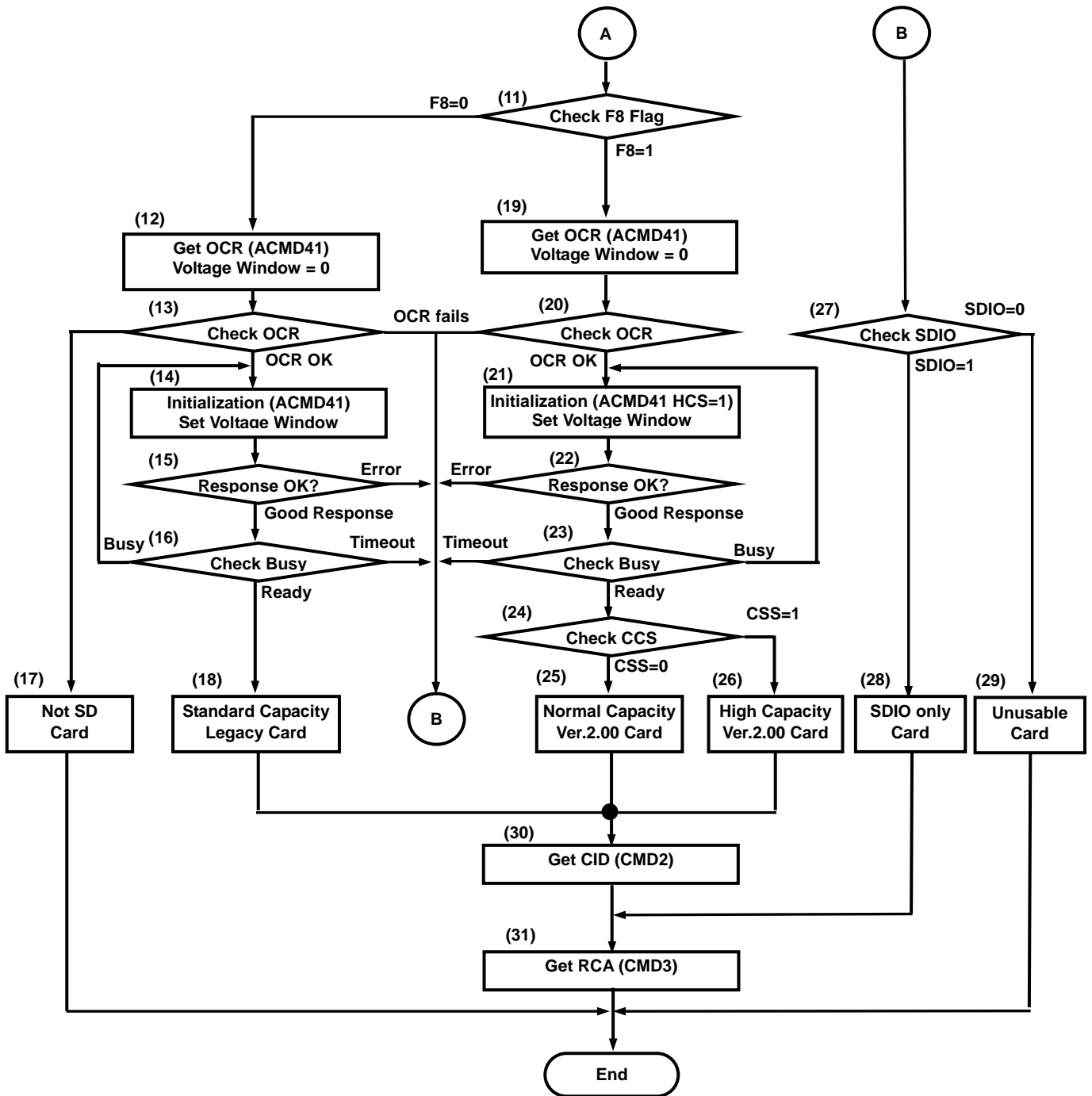


Figure 3-9 : Card Initialization and Identification

- (1) SD Bus mode is selected by CMD0 (Keep Pin 1 to high during CMD0 execution).
- (2) New CMD8 shall be issued after CMD0 to support High Capacity SD Memory Card.
- (3) Voltage check command enables the Hosts to support future low voltage specification. However, at this time, only one voltage is defined. Legacy cards and Not SD cards do not respond to CMD8. In this case, set F8 to 0 (F8 is CMD8 valid flag used in step (11)) and go to Step (5). Only Version 2.00 or higher cards can respond to CMD8. The host needs to check whether CRC of the response is valid and whether VHS and check pattern in the argument are equal to VCA and check pattern in the response. Passing all these checks results in CMD8 response OK. In this case, set F8 to 1 and go to step (5). If one of the checks is failed, go to step (4).
- (4) Initialization is stopped by CMD8 fails. The host driver should retry step (1) to (3) one more time. (This is not described in the figure).
- (5) SDIO OCR is available by issuing CMD5 with setting voltage window (bit 23 to 0) in the argument to 0. SDIO initialization is not started.
- (6) No response means the card does not have SDIO function. Set SDIO flag to 0 and go to step (11). If the card responds to CMD5 and the response is OK, go to step (7). If the response is error, set SDIO flag to 0 and go to step (10). SDIO flag indicates whether SDIO functions are initialized or not.
- (7) The SDIO portion starts initialization by CMD5 with setting the supply voltage to the voltage window. If the supplied voltage is not matched with voltage window of card, the card goes into inactive state and does not return the response.
- (8) If no response or error response is receive, set SDIO flag to 0 and go to step (10). If good response is received, go to step (9).
- (9) Check busy status in the response. If busy is released, set SDIO flag to 1 and go to step (10). Repeat from step (7) while busy is indicated. Detecting timeout of 1 second exits the loop. In this case, set SDIO flag to 0 and go to step (10).
- (10) Good response in this step means that all responses received at (6) and (8) are valid. When response is good, MP (memory present) flag in the response can be checked. If the response valid and MP=0, go to step (28). Otherwise, go to step (11).
- (11) Check F8 flag set in step (3). If CMD8 is executed correctly (F8=1), go to step (19). Otherwise, go to step (12).
- (12) OCR is available by issuing ACMD41 with the voltage window (bit 23 to 0) in the argument is set to 0. Memory initialization is not started. The response of CMD55 (ACMD41) may indicate illegal command error due to some SD cards do not recognized CMD8. The Host Driver should ignore this error or issue CMD0 before ACMD41 to clear this error status.
- (13) If response of CMD55 is not received, the card is not SD cards and goes to (17). If the card responds to CMD55, it may also respond to CMD41. If the responses of ACMD41 are OK, go to Step (14). Otherwise, go to step (27). Locked card can be detected by the card status in the response of CMD55.
- (14) The memory portion starts initialization by Issuing ACMD41 with setting the supply voltage to the voltage window. If the supplied voltage is not matched with voltage window of card, the card goes into inactive state and does not return the response.
- (15) If no response or error response is received, go to step (27). If good response is received, go to step (16).
- (16) Check busy status in the response. If busy is released, go to step (18). Repeat from step (14) while busy is indicated. The interval of ACMD41 shall be less than 50ms. Detecting timeout of 1 second exits the loop and go to step (27).
- (17) The host recognizes that the card is not SD memory card and quits SD card initialization.
- (18) The host recognizes that the card is Version 1.xx Standard Capacity SD Memory Card. Go to Step (30).
- (19) OCR is available by issuing ACMD41 with setting the voltage window (bit 23 to 0) in the argument is set to 0. Memory initialization is not started. Setting of HCS does not affect this operation.
- (20) If the card responds to CMD55, it may also respond to CMD41. If the responses of ACMD41 are

**SD Host Controller Simplified Specification Version 2.00**

---

- OK, go to Step (21). Otherwise, go to step (27). Locked card can be detected by the card status in the response of CMD55.
- (21) The memory portion starts initialization by Issuing ACMD41 with setting the supply voltage to the voltage window. If the supplied voltage is not matched with voltage window of card, the card goes into inactive state and does not return the response. HCS in the argument is set to 1, which indicates supporting High Capacity Memory Card.
  - (22) If no response or error response is received, go to step (27). If good response is received, go to step (23).
  - (23) Check busy status in the response. If busy is released, go to step (24). Repeat from step (21) while busy is indicated. The interval of ACMD41 shall be less than 50ms. Detecting timeout of 1 second exits the loop and go to step (27).
  - (24) CCS in the response is valid after busy is released. If CCS = 0, it indicates the Standard Capacity SD Memory Card and go to step (25). If CSS = 1, it indicates the High Capacity SD Memory Card and go to Step (26).
  - (25) The host recognizes that the card is Standard Capacity SD Memory Card. Optimal functions defined in Version 2.00 or higher are available. Go to Step (30).
  - (26) The host recognizes that the card is the High Capacity SD Memory Card. Go to Step (30).
  - (27) Check SDIO flag. If SDIO=1, go to step (28). Otherwise, go to step (29).
  - (28) The host recognizes that the card is SDIO only card and go to step (31).
  - (29) The host recognizes that the card is unusable.
  - (30) In case of memory card, CMD2 is issued to get CID and Go to Step (31).
  - (31) CMD3 is issued to get RCA. If the RCA number is 0, the Host should issue CMD3 again.

### 3.7. 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".

Refer to the latest SD Physical Layer Specification and SDIO Specification for more detail about SD commands specification.

### 3.7.1. Transaction Control without Data Transfer Using DAT Line

In this section, the sequence for how to issue SD Command and how to complete SD Command is explained. Figure 3-10 shows the sequence to issue a SD Command and Figure 3-11 shows the sequence to finalize a SD Command.

#### 3.7.1.1. The sequence to issue a SD Command

The sequence to issue the SD Command is detailed below.

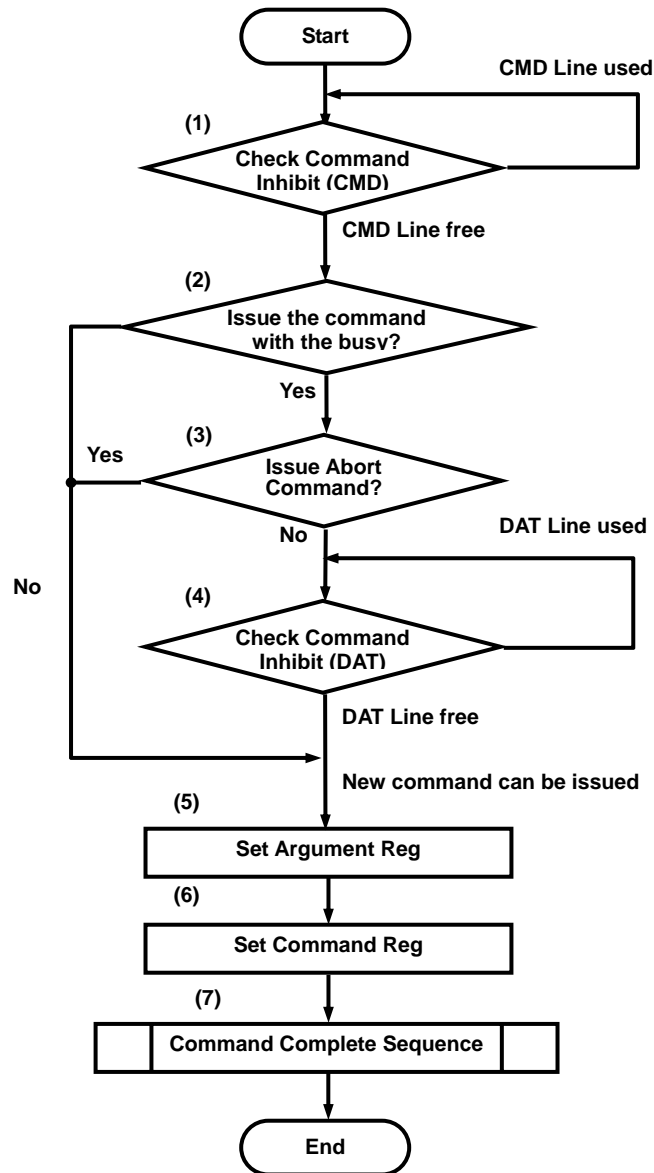


Figure 3-10: SD Command Issue Sequence

- (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 set to 0.
- (5) Set the value of command argument to the *Argument* register.
- (6) Set the *Command* register.  
Note: Writing the upper byte [3] in the *Command* register causes the host controller to issue a SD command to the SD card.
- (7) Perform Command Completion Sequence in accordance with 3.7.1.2.

## 3.7.1.2. The sequence to finalize a command

Figure 3-11 shows the sequence to finalize a SD Command. There is a possibility that some errors (Command Index/End bit/CRC/Timeout Error) occur during this sequence.

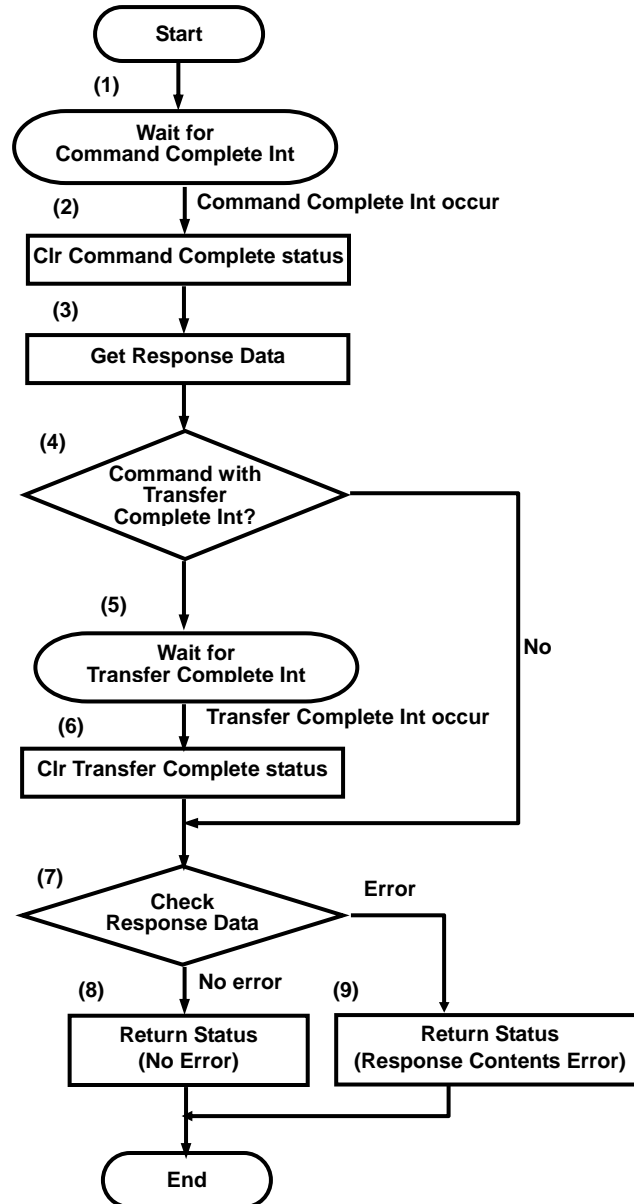


Figure 3-11: Command Complete Sequence



- (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 of 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.

### 3.7.2. 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 3-12 and the sequence using DMA is shown in Figure 3-13.

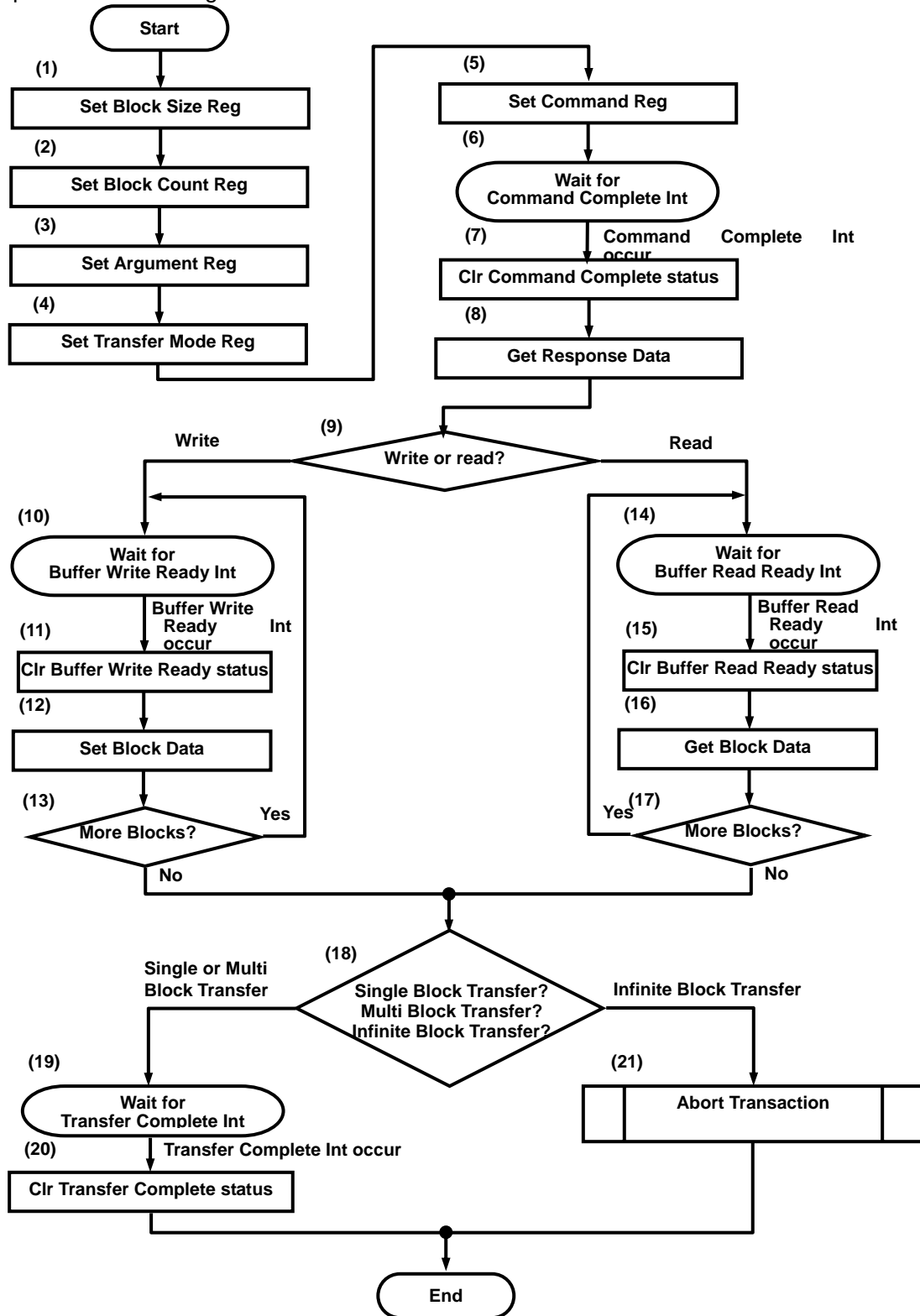
In addition, the sequences for SD transfers are classified into following three kinds according to how the number of blocks is specified:

- (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.

**SD Host Controller Simplified Specification Version 2.00**

**3.7.2.1. Not using DMA**

The sequence for not using DMA is shown below.



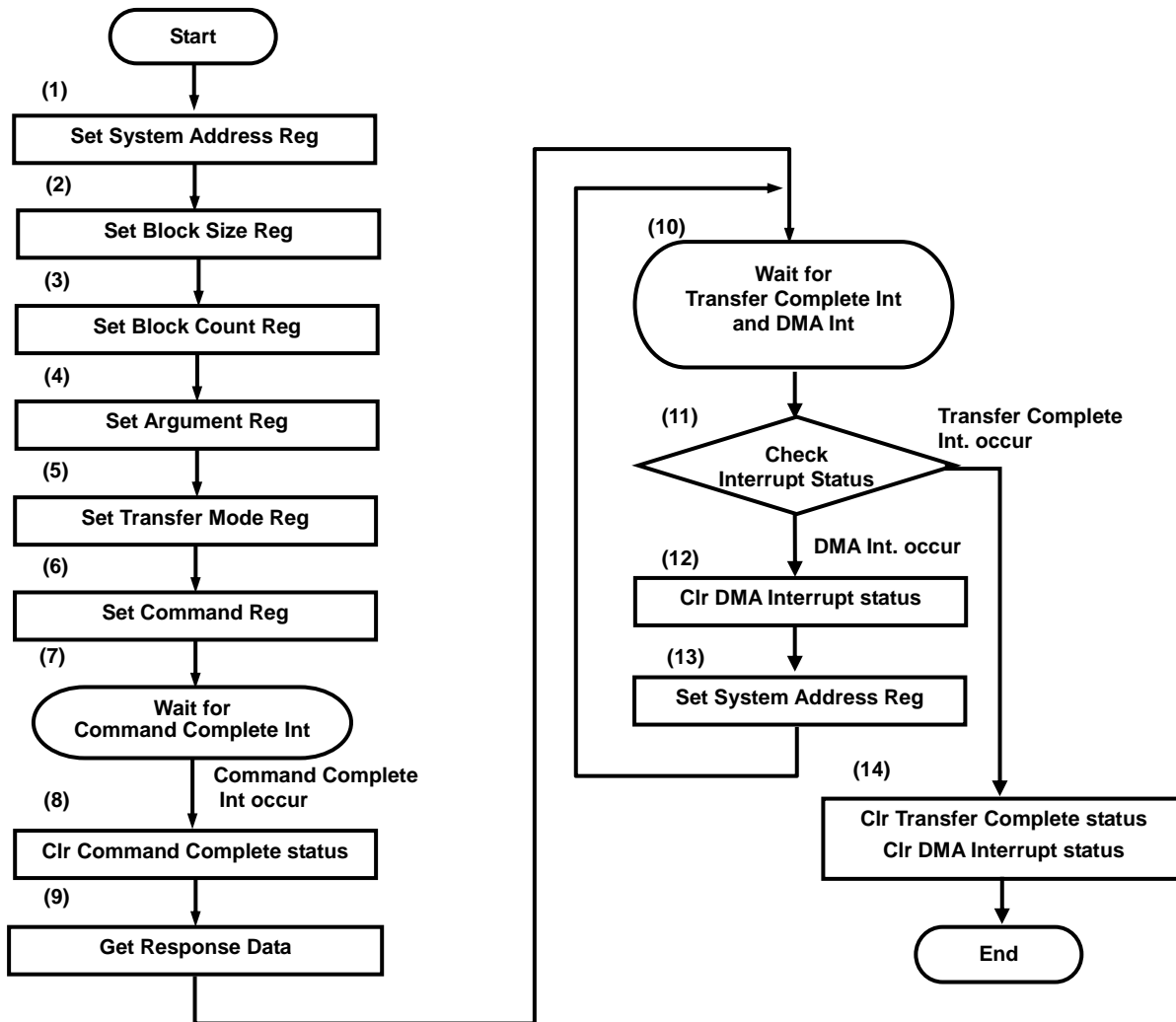
**Figure 3-12: Transaction Control with Data Transfer Using DAT Line Sequence (Not using DMA)**

- (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 argument value to *Argument* register.
- (4) Set the value to the *Transfer Mode* register. The host driver determines **Multi / Single Block Select, Block Count Enable, Data Transfer Direction, Auto CMD12 Enable** and **DMA Enable**. **Multi / Single Block Select** and **Block Count Enable** are determined according to Table 2-8.
- (5) Set the value to *Command* register.  
Note: When writing the upper byte [3] of *Command* register, SD command is issued.
- (6) 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 of the issued command.
- (9) In the case where this sequence is for write to a card, go to step (10). In case of read from a card, go to step (14).
- (10) Then wait for **Buffer Write Ready** Interrupt.
- (11) Write 1 to the **Buffer Write Ready** in the *Normal Interrupt Status* register for clearing this bit.
- (12) Write block data (in according to the number of bytes specified at the step (1)) to *Buffer Data Port* register.
- (13) Repeat until all blocks are sent and then go to step (18).
- (14) Then wait for the **Buffer Read Ready** Interrupt.
- (15) Write 1 to the **Buffer Read Ready** in the *Normal Interrupt Status* register for clearing this bit.
- (16) Read block data (in according to the number of bytes specified at the step (1)) from the *Buffer Data Port* register.
- (17) Repeat until all blocks are received and then go to step (18).
- (18) If this sequence is for Single or Multiple Block Transfer, go to step (19). In case of Infinite Block Transfer, go to step (21).
- (19) Wait for **Transfer Complete** Interrupt.
- (20) Write 1 to the **Transfer Complete** in the *Normal Interrupt Status* register for clearing this bit.
- (21) Perform the sequence for Abort Transaction in accordance with Section 3.8.  
Note: Step (1) and Step (2) can be executed at same time. Step (4) and Step (5) can be executed at same time.

**SD Host Controller Simplified Specification Version 2.00**

**3.7.2.2. Using SDMA**

The sequence for using SDMA is shown below.



**Figure 3-13: Transaction Control with Data Transfer Using DAT Line Sequence (Using SDMA)**

**SD Host Controller Simplified Specification Version 2.00**

---

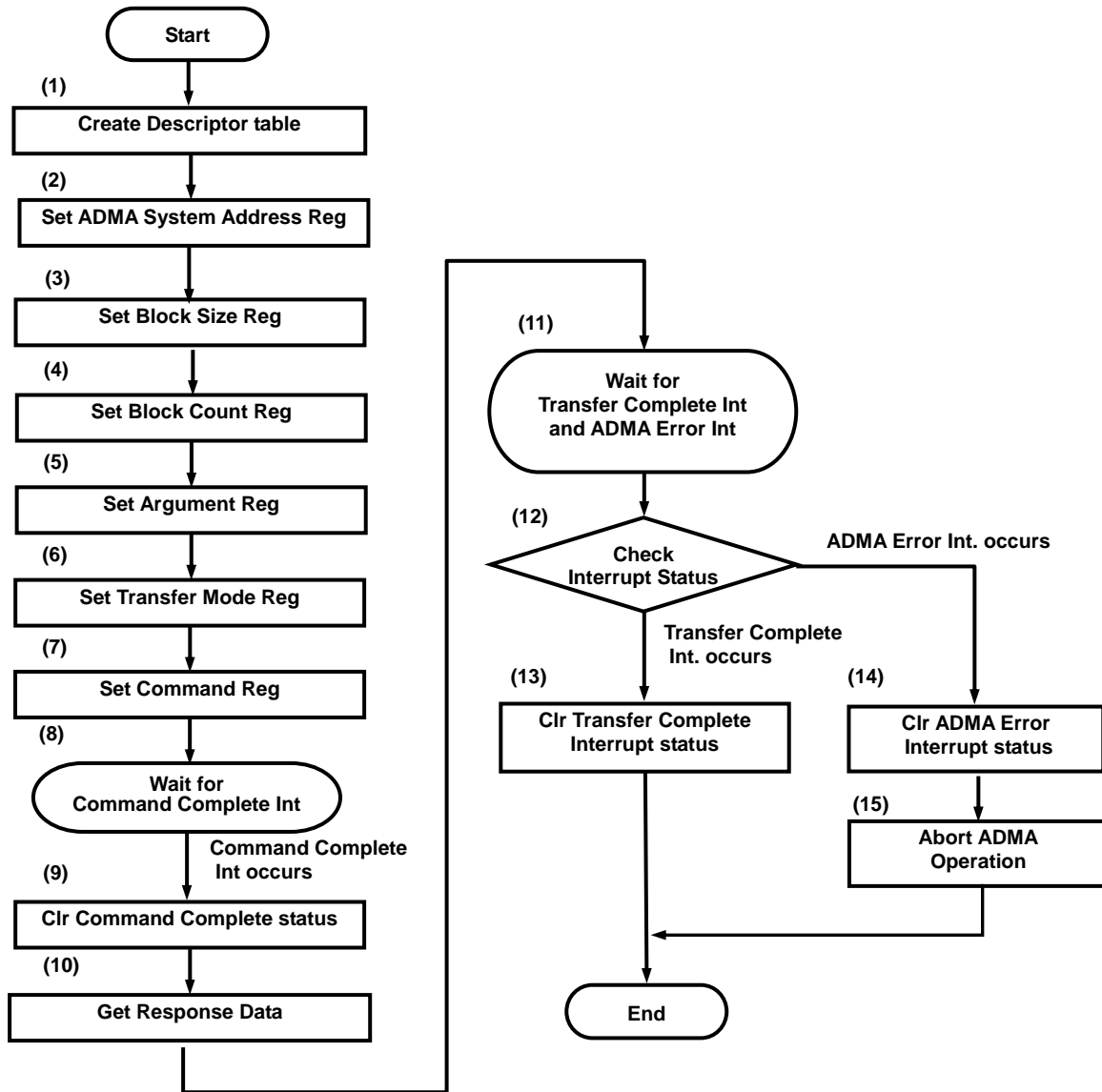
- (1) Data location of system memory is set to the *SDMA System Address* register.
- (2) Set the value corresponding to the executed data byte length of one block in the *Block Size* register.
- (3) Set the value corresponding to the executed data block count in the *Block Count* register in accordance with Table 2-8.
- (4) Set the argument value to the *Argument* register.
- (5) Set the value to the *Transfer Mode* register. The host driver determines **Multi / Single Block Select, Block Count Enable, Data Transfer Direction, Auto CMD12 Enable** and **DMA Enable**. **Multi / Single Block Select** and **Block Count Enable** are determined according to Table 2-8.
- (6) Set the value to the *Command* register.  
Note: When writing to the upper byte [3] of the *Command* register, the SD command is issued and SDMA is started.
- (7) 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 of 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.

**SD Host Controller Simplified Specification Version 2.00**

**3.7.2.3. Using ADMA**

The sequence for using ADMA is shown below.



**Figure 3-14: Transaction Control with Data Transfer Using DAT Line Sequence (Using ADMA)**

**SD Host Controller Simplified Specification Version 2.00**

---

- (1) Create Descriptor table for ADMA in the *system memory*
- (2) Set the Descriptor address for ADMA in the *ADMA System Address* register.
- (3) Set the value corresponding to the executed data byte length of one block in the *Block Size* register.
- (4) Set the value corresponding to the executed data block count in the *Block Count* register in accordance with Table 2-8.

If the **Block Count Enable** in the *Transfer Mode* register is set to 1, total data length can be designated by the *Block Count* register and the Descriptor Table. These two parameters shall indicate same data length. However, transfer length is limited by the 16-bit *Block Count* register.

If the **Block Count Enable** in the *Transfer Mode* register is set to 0, total data length is designated by not *Block Count* register but the Descriptor Table. In this case, ADMA reads more data than length programmed in descriptor from SD card. Too much read operation is aborted asynchronously and extra read data is discarded when the ADMA is completed

- (5) Set the argument value to the *Argument* register.
- (6) Set the value to the *Transfer Mode* register. The host driver determines **Multi / Single Block Select, Block Count Enable, Data Transfer Direction, Auto CMD12 Enable** and **DMA Enable**. **Multi / Single Block Select** and **Block Count Enable** are determined according to Table 2-8.
- (7) Set the value to the *Command* register.  
 Note: When writing to the upper byte [3] of the *Command* register, the SD command is issued and DMA is started.
- (8) Then wait for the **Command Complete** Interrupt.
- (9) Write 1 to the **Command Complete** in the *Normal Interrupt Status* register to clear this bit.
- (10) Read *Response* register and get necessary information of the issued command.
- (11) Wait for the **Transfer Complete** Interrupt and **ADMA Error Interrupt**.
- (12) If **Transfer Complete** is set 1, go to Step (13) else if **ADMA Error Interrupt** is set to 1, go to Step (14).
- (13) Write 1 to the **Transfer Complete Status** in the *Normal Interrupt Status* register to clear this bit.
- (14) Write 1 to the **ADMA Error Interrupt Status** in the *Error Interrupt Status* register to clear this bit.
- (15) Abort ADMA operation. SD card operation should be stopped by issuing abort command. If necessary, the host driver checks *ADMA Error Status* register to detect why **ADMA error** is generated.

Note: Step (3) and Step (4) can be executed simultaneously. Step (6) and Step (7) can also be executed simultaneously.

### 3.8. 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.

#### 3.8.1. Asynchronous Abort

The sequence for Asynchronous Abort is shown in Figure 3-15.

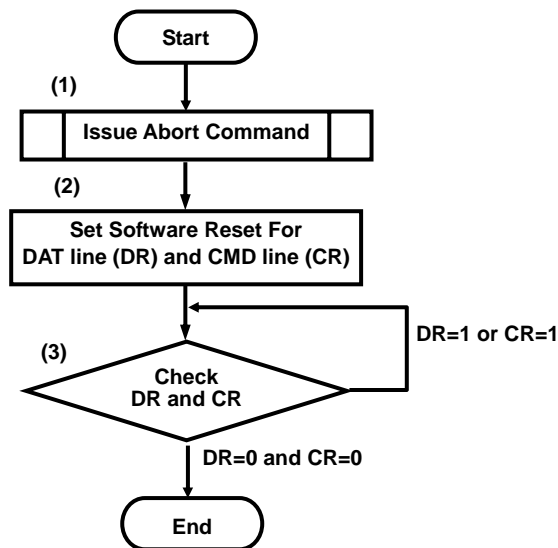


Figure 3-15: Asynchronous Abort Sequence

- (1) Issue Abort Command in accordance with Section 3.7.1
- (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).



### 3.8.2. Synchronous Abort

The sequence for Synchronous Abort is shown in Figure 3-16

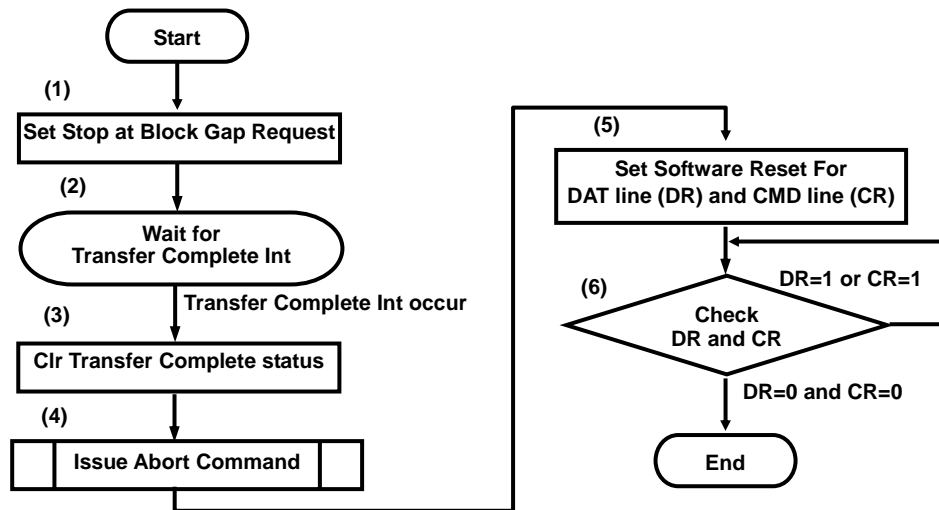


Figure 3-16: Synchronous Abort Sequence

- (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 3.7.1
- (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).

### 3.9. Changing Bus Speed Mode Setting

This section describes the sequence for switching the bus speed mode; Default Speed and High Speed mode. The switch command (CMD6) is used to change memory card bus speed mode. The EHS bit in the CCCR register is used to change the bus speed mode for SDIO card. In case of Combo card, either of the switch method changes both memory and IO bus speed mode. This means the first switch is effective. Refer to Physical Layer Specification and SDIO Specification for more information about switching bus speed. The sequence for switching bus speed mode for Combo Card is described in Figure 3-17. Note that if the card is locked, bus with cannot be changed. Unlock the card is required before changing bus width.

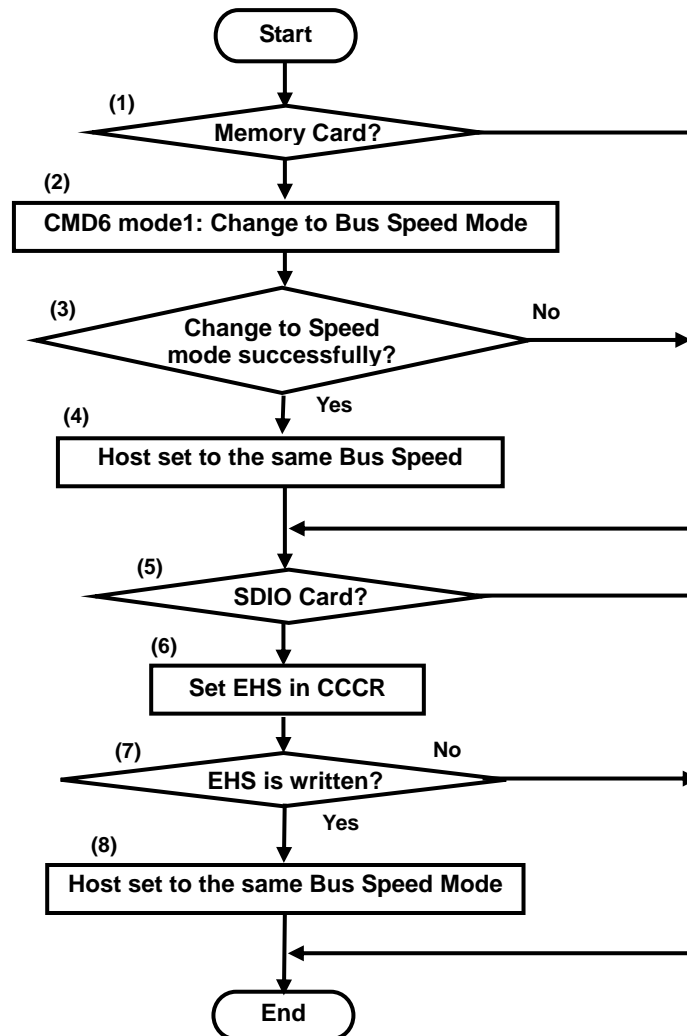


Figure 3-17 : High Speed Mode Setting for Combo Card

- (1) The Host Driver should check whether it is memory card and whether the card supports CMD6. The card supports CMD6 when SD\_SPEC field in SCR register indicates that the Physical Layer Specification Version is 1.10 or later. If not, go to (5).
- (2) Issue CMD6 with mode 1 to change bus speed mode (Default or High Speed mode).
- (3) Get response of CMD6 and check whether bus speed mode is changed successfully. If bus speed is not changed, go to step (5).
- (4) If the card is changed the bus speed mode successfully, the Host Driver changes the Host Controller bus speed to the same mode.
- (5) The Host Driver should check whether it is SDIO card. If not, go to the end.
- (6) Issue CMD52 to write EHS bit in CCCR to change bus speed mode. (The same bus speed mode of (1) shall be set.)
- (7) Read EHS and check whether EHS is changed as write it. If bus speed is not changed, go to the end.
- (8) If the card is changed the bus speed mode successfully, the Host Driver changes the Host Controller bus speed to the same mode. In case of Combo card, bus speed is already changed at step (4) and this step does not affect changing bus speed.

### 3.10. Error Recovery

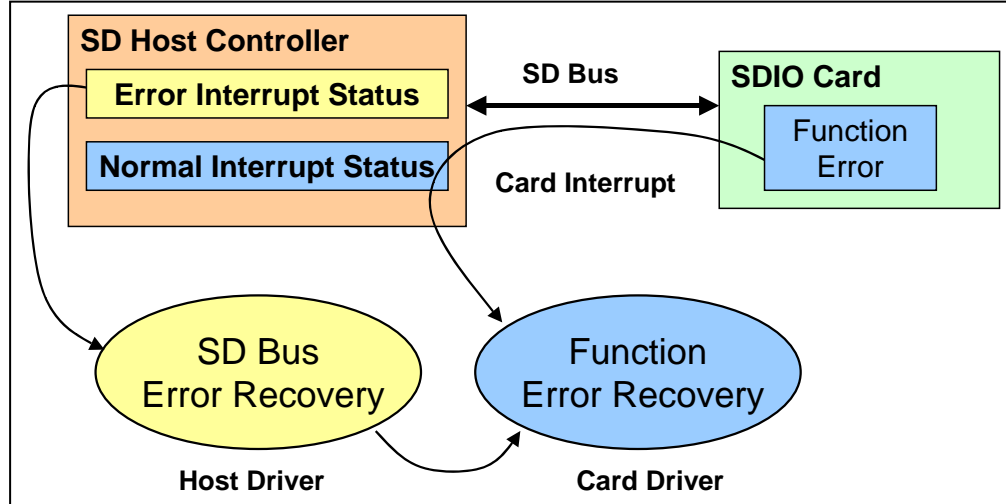


Figure 3-18 : Error Report and Recovery

Figure 3-18 shows concept of error report and its recovery. The Host Controller has 2 interrupt status registers. If an error occurs 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 to inform not only error statuses but also normal information. For example, to inform function ready.) The Card Driver shall do function error recovery because the Host Driver does not 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, and 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.

(4) 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.

(5) Using reset command for memory and SDIO card

Re-initialization sequence is required.

(6) Off and on power supply for the SD Bus

The card may be recovered by the power on reset. Re-initialization sequence is required.

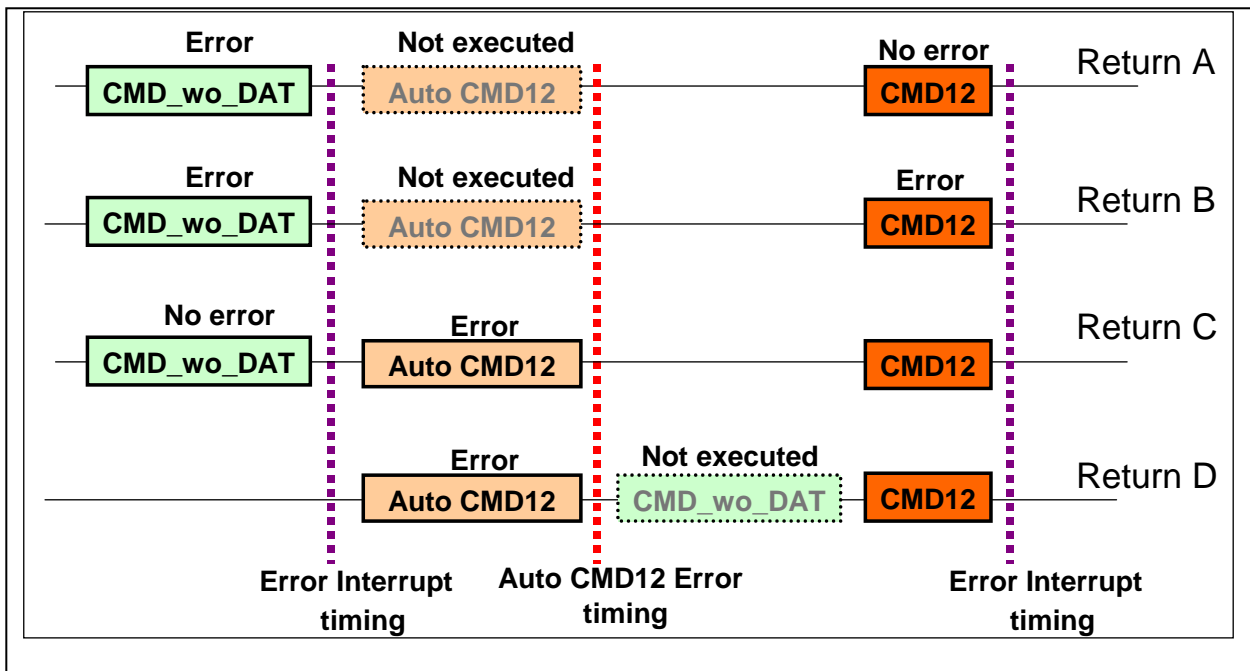
The two cases where the Host Driver needs the "Error Recovery" sequence are classified as follows:

(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 3-19. 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 3-19: 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.

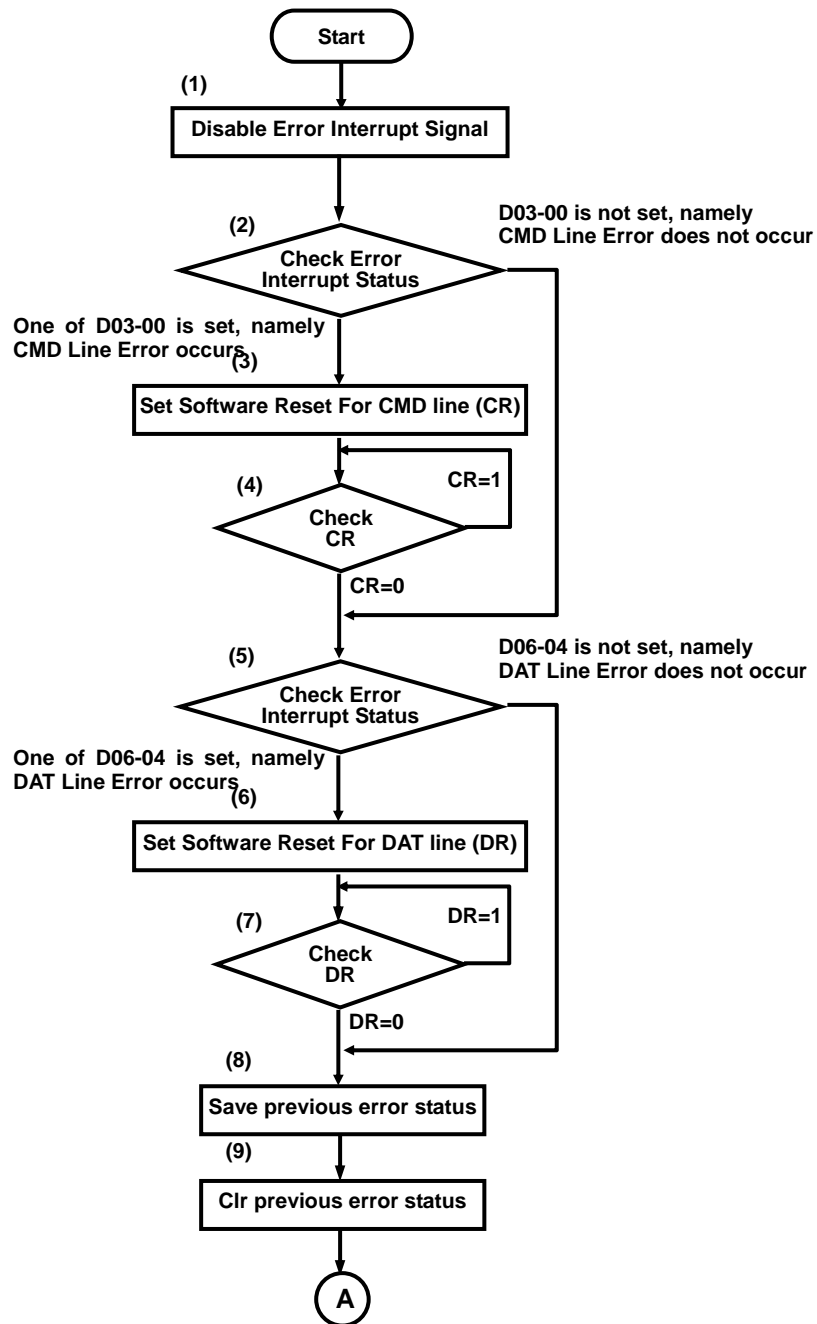
**Implementation Note:**

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.

### 3.10.1. Error Interrupt Recovery

The sequence for Error Interrupt Recovery is shown in Figure 3-20.



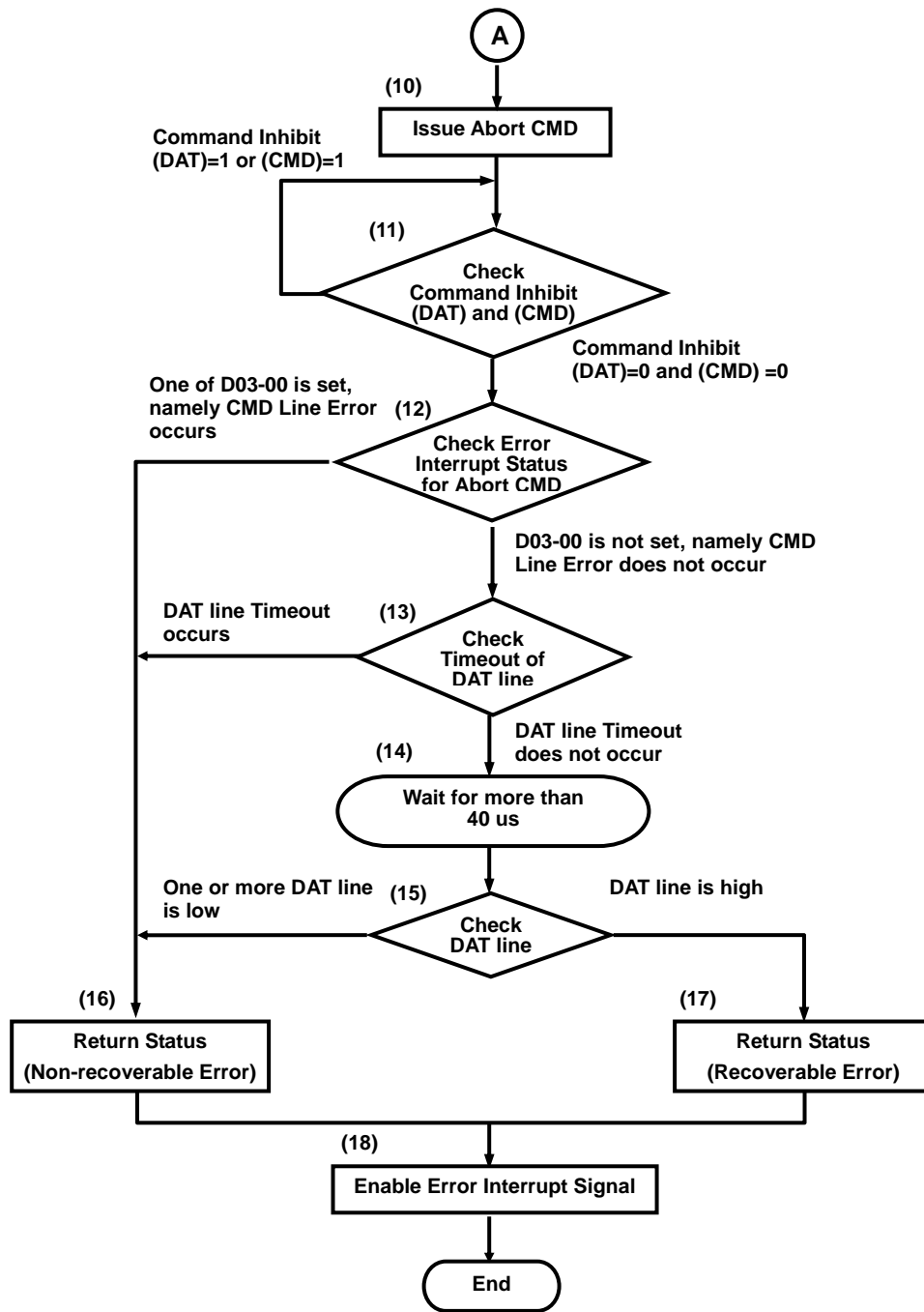


Figure 3-20: Error Interrupt Recovery Sequence

- (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.



3.10.2. Auto CMD12 Error Recovery

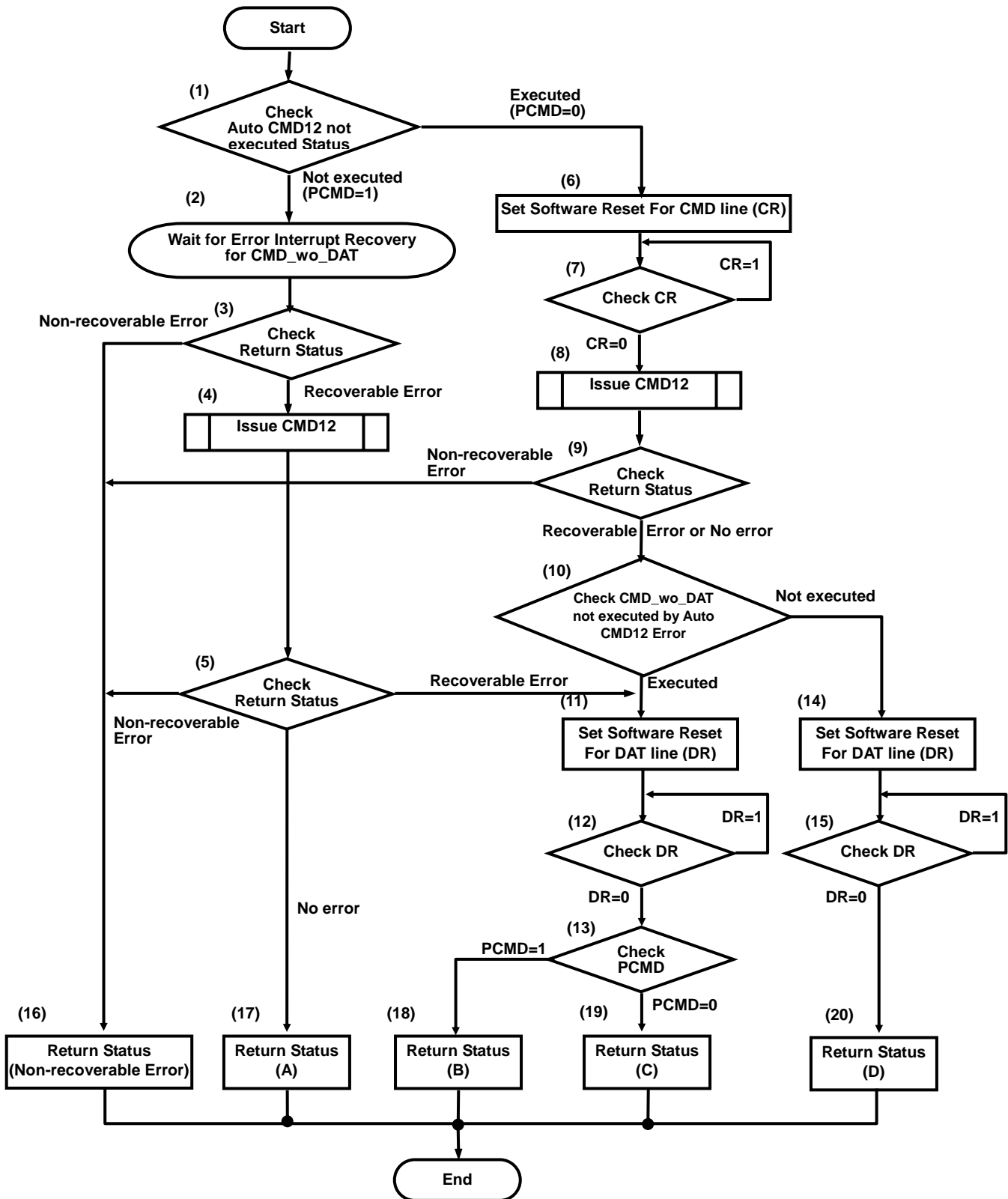


Figure 3-21 : Auto CMD12 Error Recovery Sequence

The sequence for Auto CMD12 Error Recovery is shown in Figure 3-21. Following four cases A-D shall be covered.

- 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.

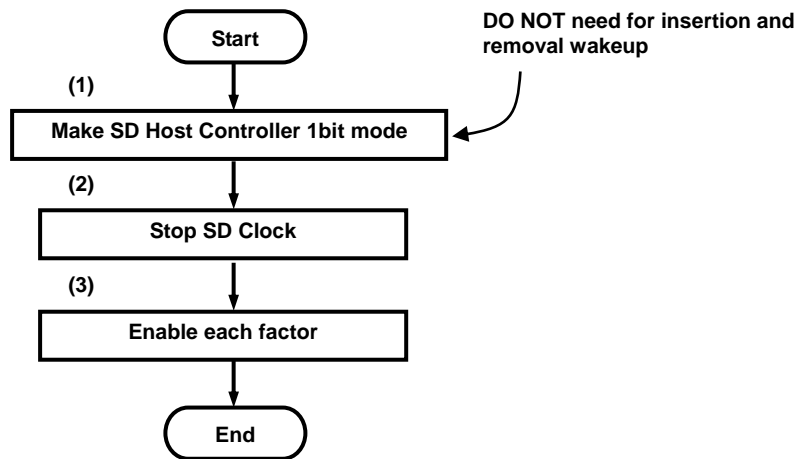
- (1) Check **Auto CMD12 Not Executed** in the *Auto CMD12 Error Status* register. If this bit is set to 1, go to step (2). If this bit is set to 0, go to step (6). In addition, the Host Driver shall define **PCMD** flag, which changes to 1 if **Auto CMD12 Not Executed** is set to 1.
- (2) Wait for Error 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 3.7.1
- (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 according to Section 3.7.1. Acceptance of CMD12 depends on the state of the card. CMD12 may make the card to return to tran state. If the card is already in tran state, the card does not response to CMD12.
- (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.

### 3.11. 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.

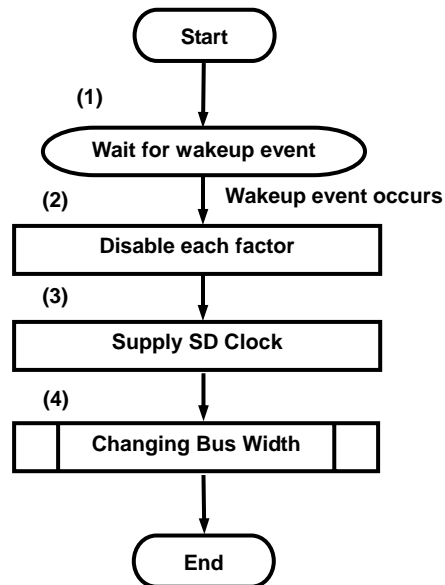
The sequence for preparing wakeup before the Host System goes into standby mode is shown in Figure 3-22.



**Figure 3-22: 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 3-23.



**Figure 3-23: 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.

### 3.12. 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". ADMA operation does not support this function.

#### 3.12.1. Suspend Sequence

The sequence for suspend is shown in Figure 3-24.

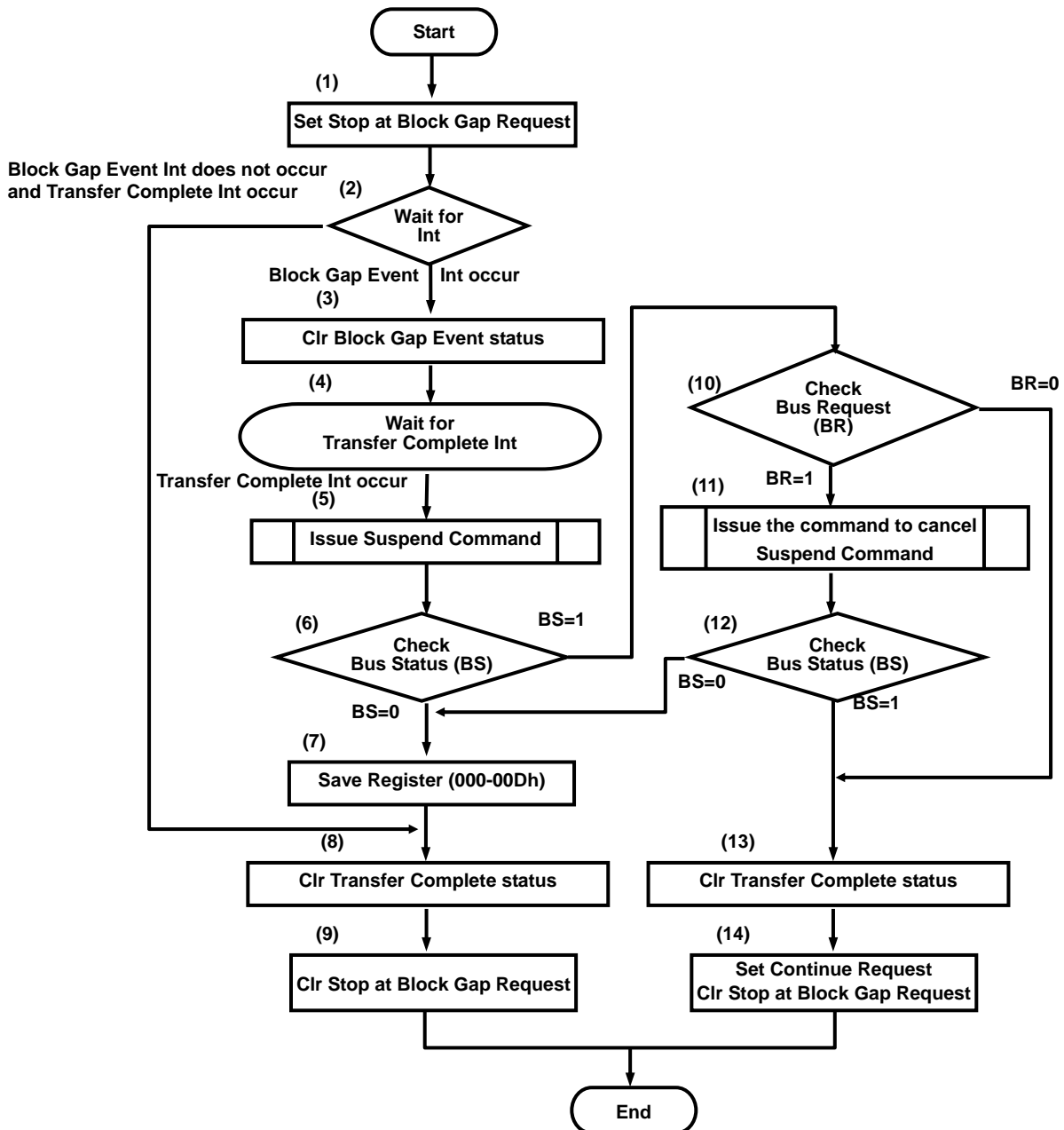


Figure 3-24 : The sequence for suspend

**SD Host Controller Simplified Specification Version 2.00**

- (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 3.7.1.
- (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 3.7.1 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.

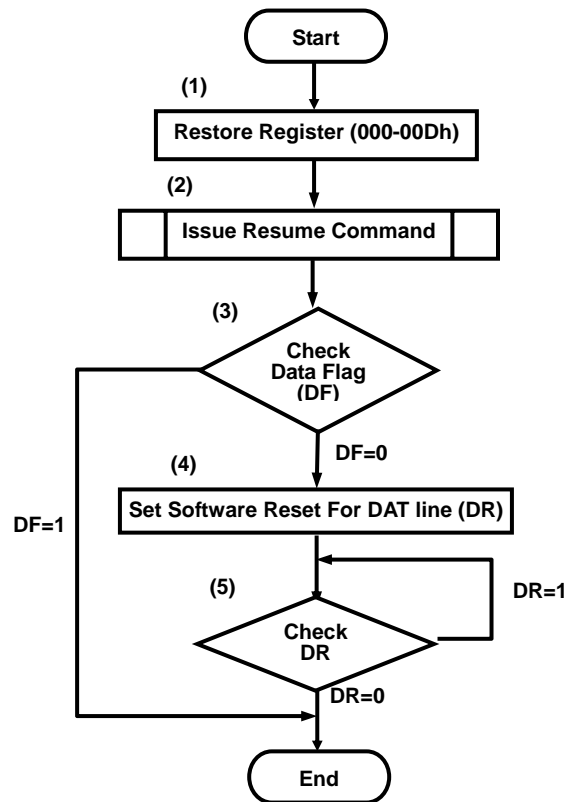
The Table 3-1 shows conditions to be able to use Suspend / Resume function.

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**

### 3.12.2. Resume Sequence

The sequence for resume is shown in Figure 3-25.



**Figure 3-25 : The sequence for resume**

- (1) Restore the register (000h-00Dh).
- (2) Issue the Resume Command in accordance with Section 3.7.1.
- (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).

### 3.12.3. Read transaction wait / continue timing

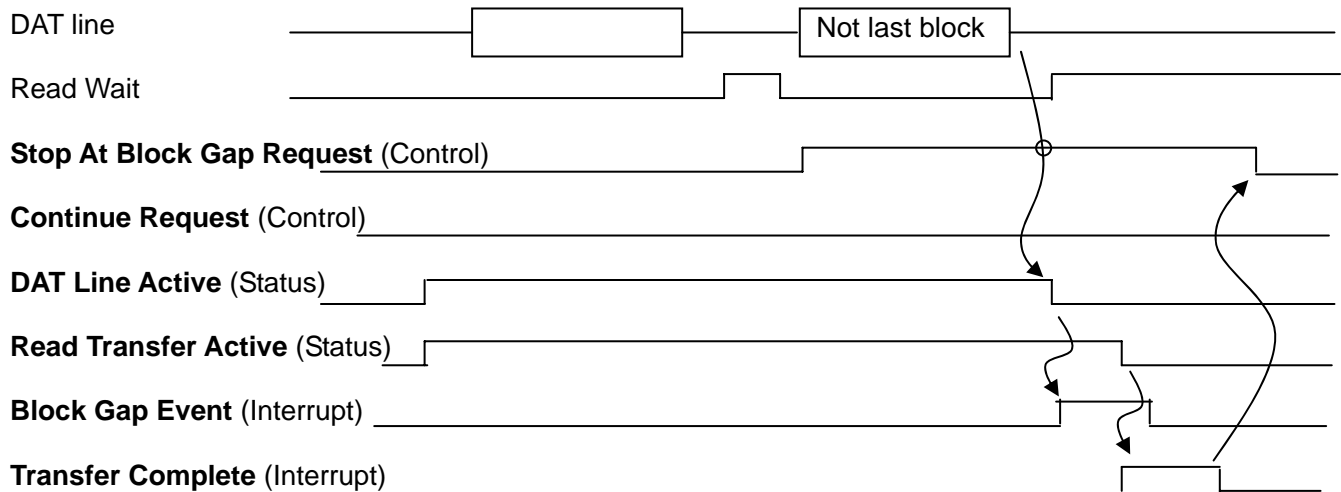
**Implementation Note:**

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 3-26 : Wait read transfer by Stop At Block Gap Request**

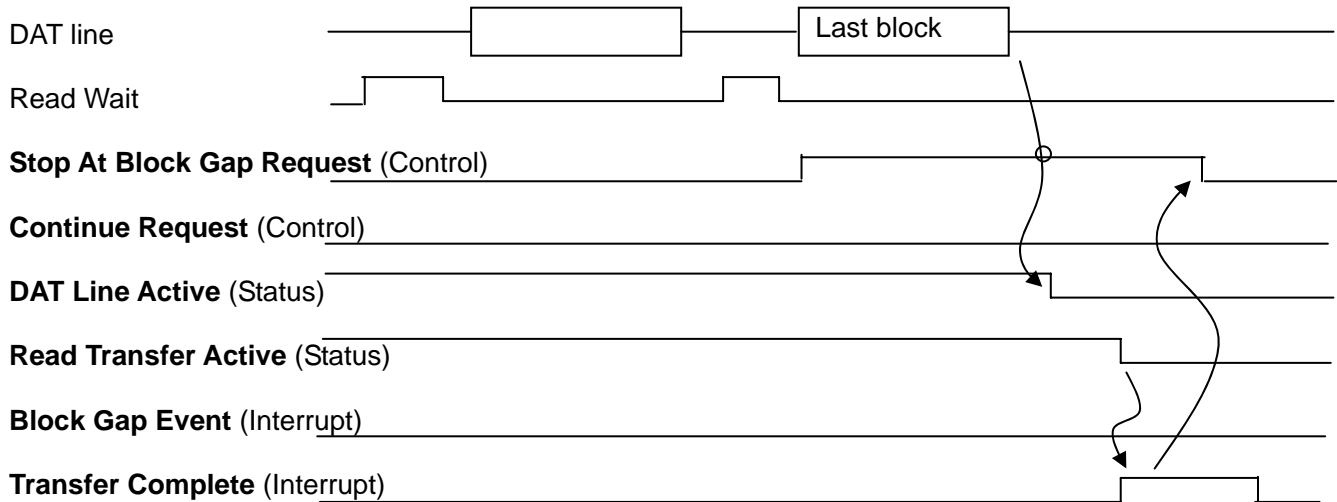
The Host Controller can accept a **Stop At Block Gap Request** when all the following conditions are met.

- (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

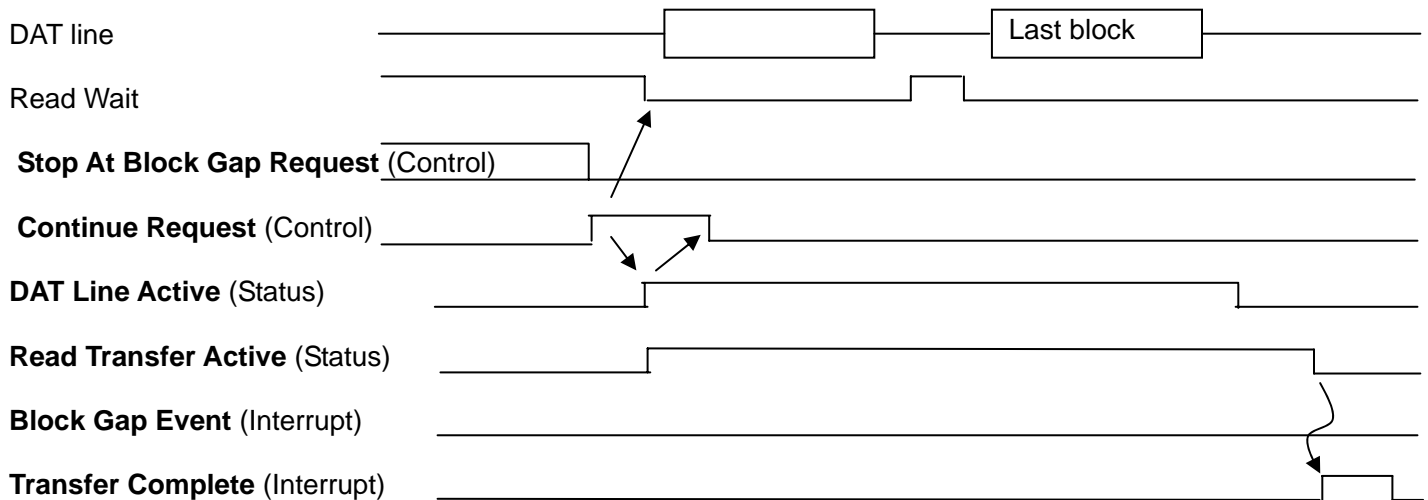
- (1) Clear **DAT Line Active** status and generate the **Block Gap Event** Interrupt
- (2) After all valid data has been read (No valid read data remains in the Host Controller), clear the **Read Transfer Active** status and generate the **Transfer Complete** Interrupt.
- (3) After accepting **Transfer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





**Figure 3-27 : 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 3-28 : 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 is 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.

### 3.12.4. Write transaction wait / continue timing

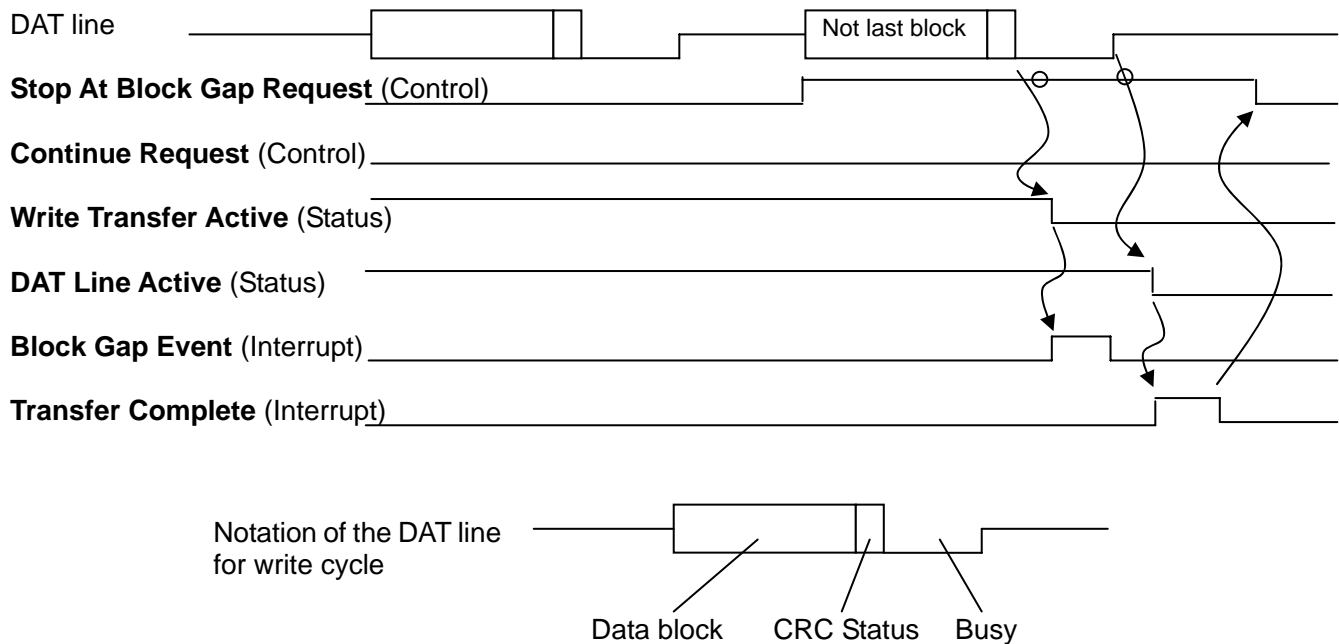
Implementation Note:

**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.



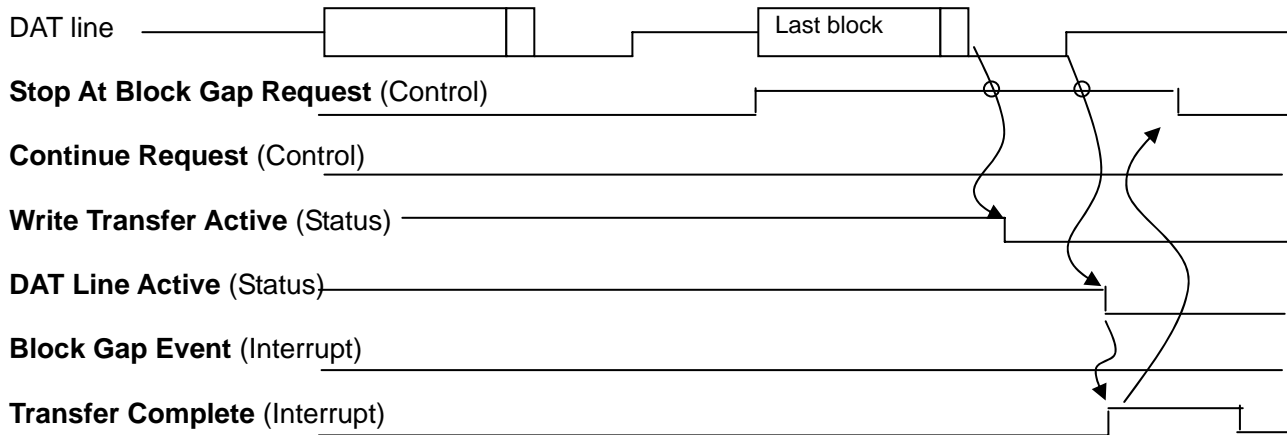
**Figure 3-29 : Wait write transfer by Stop At Block Gap Request**

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

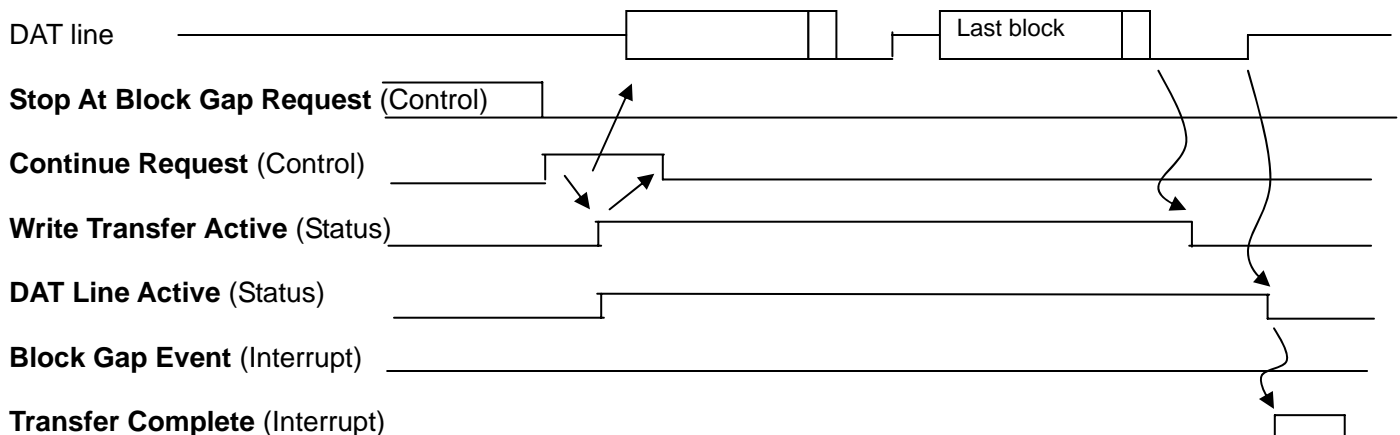
After accepting the **Stop At Block Gap Request**

- (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 3-30 : 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 3-31 : 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).

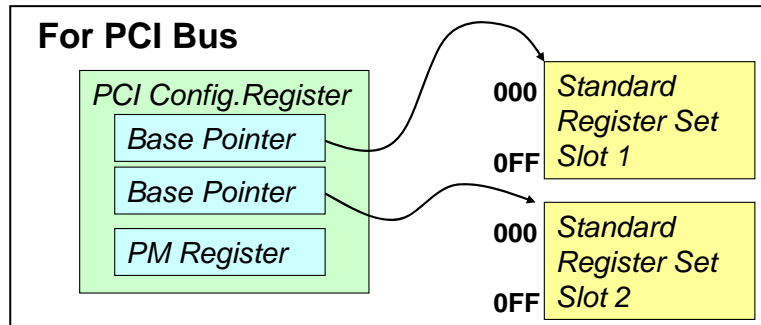
The end of transfer is specified by data length.

- (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 : PCI Configuration Register

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 Register Maps



**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, though the use of multiple Base Addresses is limited 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.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.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.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	<b>Basic Class</b> <table border="1"> <tr> <td>08h:</td> <td>General Peripheral</td> </tr> </table>	08h:	General Peripheral				
08h:	General Peripheral							
23-16 (0Ah)	RO	<b>Sub Class</b> <table border="1"> <tr> <td>05h:</td> <td>for SD Host Controller</td> </tr> </table>	05h:	for SD Host Controller				
05h:	for SD Host Controller							
15-08 (09h)	RO	<b>Interface Code</b> <table border="1"> <tr> <td>00h:</td> <td>Standard Host not supported DMA</td> </tr> <tr> <td>01h:</td> <td>Standard Host supported DMA</td> </tr> <tr> <td>02h:</td> <td>Vendor unique SD Host Controller</td> </tr> </table>	00h:	Standard Host not supported DMA	01h:	Standard Host supported DMA	02h:	Vendor unique SD Host Controller
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.3.2 Base Address Register (Offset 10h)

Maximum 6 base addresses can be supported, In case of multiple functions controller, these registers are used to point location not only for the SD Host Controller register sets but also for other functions. Refer to Slot Information Register (A.3.3) to identify which base address is used for the SD Host Controller.

D31	D08	D7	D01	D00
Base Address			00 0000	Space Indicator

**Figure A - 3 : PCI Config. Base Address Register**

Location	Attrib	
31-08	RW	<b>Base Address</b> 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	<b>Space Indicator</b> 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.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	<b>Reserved</b>																		
06-04	RO	<p><b>Number Of Slots</b> These statuses indicate the number of slots the Host Controller supports. In the case of single function, maximum 6 slots can be assigned.</p> <table border="1"> <tr> <td>000b:</td> <td>1 slot</td> </tr> <tr> <td>001b:</td> <td>2 slot</td> </tr> <tr> <td>010b:</td> <td>3 slot</td> </tr> <tr> <td>011b:</td> <td>4 slot</td> </tr> <tr> <td>100b:</td> <td>5 slot</td> </tr> <tr> <td>101b:</td> <td>6 slot</td> </tr> </table>	000b:	1 slot	001b:	2 slot	010b:	3 slot	011b:	4 slot	100b:	5 slot	101b:	6 slot						
000b:	1 slot																			
001b:	2 slot																			
010b:	3 slot																			
011b:	4 slot																			
100b:	5 slot																			
101b:	6 slot																			
03	Rsvd	Reserved																		
02-00	RO	<p><b>First Base Address Register Number</b> 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. <b>Number Of Slot</b> specifies number of base address.</p> <table border="1"> <tr> <td>000b:</td> <td>Base Address 10h</td> <td>(BAR0)</td> </tr> <tr> <td>001b:</td> <td>Base Address 14h</td> <td>(BAR1)</td> </tr> <tr> <td>010b:</td> <td>Base Address 18h</td> <td>(BAR2)</td> </tr> <tr> <td>011b:</td> <td>Base Address 1Ch</td> <td>(BAR3)</td> </tr> <tr> <td>100b:</td> <td>Base Address 20h</td> <td>(BAR4)</td> </tr> <tr> <td>101b:</td> <td>Base Address 24h</td> <td>(BAR5)</td> </tr> </table>	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)
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.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:

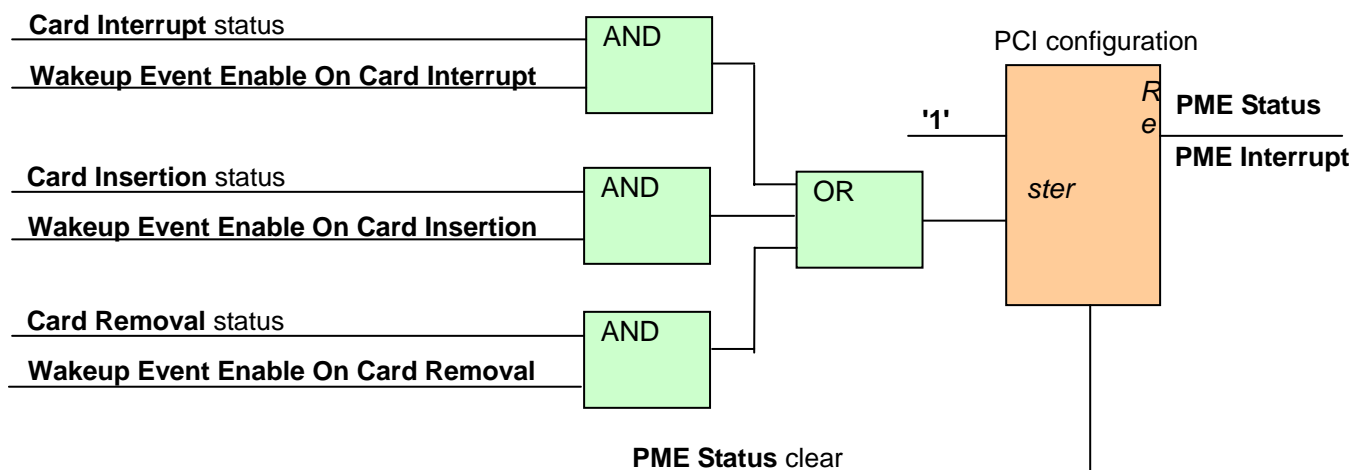
In the D0 state, while a SD card is inserted or not rejected, card power shall be supplied by setting the **SD Bus Power** in the *Power Control* register. In the D1 or D2 states, the Host Driver shall keep the preceding power supply state. In the D3 state, if the Host System supports card interrupt 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.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 refers extensively to any released version of the following SD specifications and the related Supplementary Notes.

SD Specifications  
Part 1 Physical Layer Specification

SD Specifications  
Part 2 File System Specification

SD Specifications  
Part 3 File Security Specification

SD Specifications  
Part E1 SDIO Card Specification

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: PCI bus supports ACPI.
ADMA	Advanced DMA: This term stands for ADMA1 and ADMA2.
ADMA1	ADMA Version 1: 4KByte boundary base ADMA
ADMA2	ADMA Version 2: Without 4KByte boundary limitation.(Recommended to support)
API	Application Program Interface
Block Gap	Period between blocks of data
Block	a number of bytes, basic data transfer unit
Busy	Busy signal: SD card drives busy on DAT[0] line.
CCCR	Card Common Control Register: One of registers defined in SDIO card.
CCS	Card Capacity Status: A field name in the response of ACMD41.
CDCLK	Card Detect Clock: a clock for detecting SD card
CID	Card Identification number register
Clr	Clear
CMD	SD bus command line
CMDXX	SD commands: XX indicates one or two digit decimal command number.
CMD_wo_DAT	Commands without using DAT line
CRC	Cyclic Redundancy Check
CSD	Card Specific Data register
DAT	SD bus 4-bit Data line: It is also expressed by DAT[3:0]
Descriptor Table	Sequence of ADMA Programs created on system memory.
DMA	Direct Memory Access: This term stands for SDMA, ADMA1 and ADMA2.
GPS	Global Positioning System
HCS	Host Capacity Support: A field name in the argument of ACMD41.
HW	Hardware
Int	Interrupt: SD card drives interrupt on DAT[1] line.
LED	Light Emitting Diode
OCR	Operation Conditions Register
OS	Operating System
Page Size	Unit of system memory management. Most host system adopts 4KB page size.
PCI	Peripheral Component Interconnect
PHS	Personal Handyphone System
PME	Power Management Enable
Resume	Restore and restart a suspended function. It is defined in SDIO spec.
RCA	Relative Card Address Register: RCA is received from CMD3.
SDCD#	Card Detect Signal: a signal, which is active in a level of low, for detecting SD card.
SDCLK	SD bus clock line: Host supplies clock to card through this line.
SDMA	Single Operation DMA defined in the Host Controller Specification Ver1.00.
SDWP	a signal, which is active in a level of high, for detecting SD card to be protected writing
Suspend	Stop and save a function to be able to resume. It is defined in SDIO spec.
TMCLK	a clock for detecting a timeout on DAT line

## Appendix C

### C.1 Advanced DMA1

ADMA1 was proposed while drafting SD Host Controller Standard Specification Version 2.00. 4KByte boundary basis architecture is adopted to reduce size of descriptor table. There is a restriction that start address shall be programmed at the top of 4KByte boundary. However, architectural discussion resulted that ADMA should be able to be programmed with any address and any length. Host TG strongly recommends to support ADMA2 but using ADMA1 is also permitted for applicable devices.

### C.2 Block Diagram of ADMA

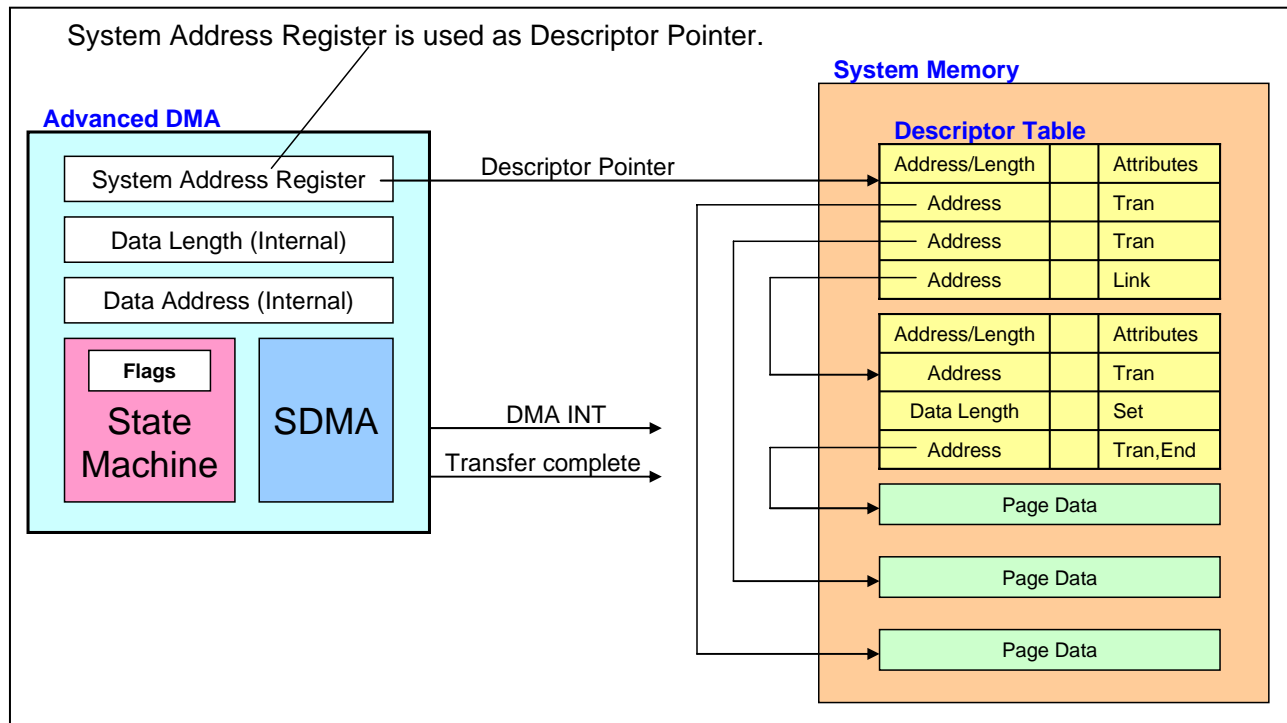


Figure C - 1 : Block Diagram of ADMA

Figure C - 1 shows block diagram of ADMA1. The descriptor Table is created in system memory by the Host Driver. Each line of the Descriptor Table consists with address/length field and attribute field. The attribute defines kind of operations for one executable unit; loading address or length to the ADMA register, transfer data within a page boundary. Lower 12-bit of address is considered to be 0. In another word, ADMA1 always start transfer from the top of 4Kbyte boundary. ADMA1 supports system with 32-bit addressing. ADMA Address register is defined as 64-bit length but only lower 32-bit address is used for ADMA1.

## C.3 An Example of ADMA1 Programming

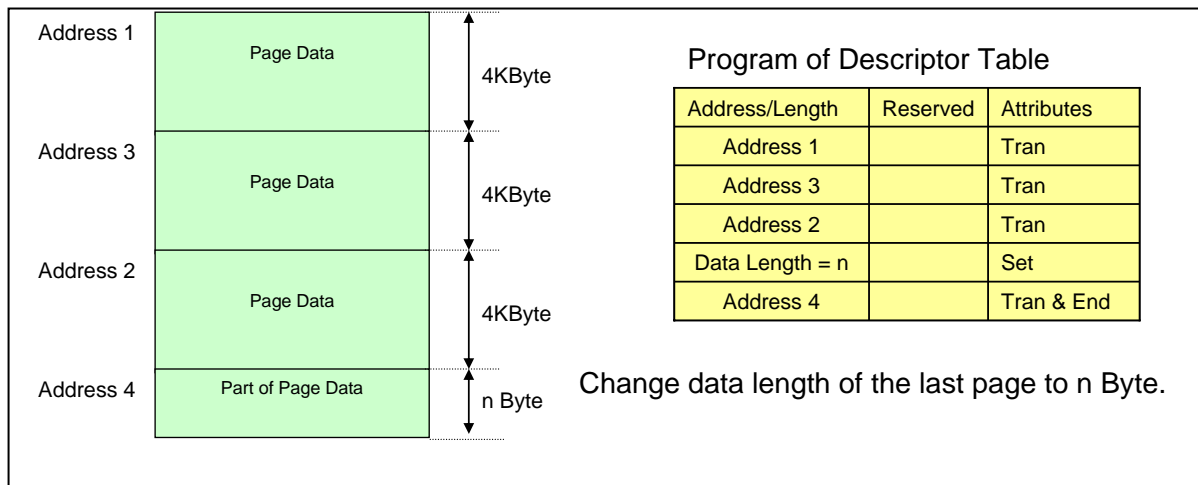


Figure C - 2 : Example of ADMA Transfer

The ADMA1 supports data transfer for the memory management of paging, which is adopted by the host system. Assuming that the page size is 4KB, the ADMA sets 4KB default page size to the Data Length register before starting DMA. In case that the page size is used larger than 4KB, the size can be programmed by the Set operation.

Figure C - 2 shows an example transfer. The data is divided in 4KB page unit and stored in 4 pages of any address. The order of transfer is Address1, Address3, Address 2 and Address 4. The total data length is adjusted by the last page, which can be programmed by the Set operation. Figure C - 2 also shows an example of Descriptor programming.

## C.4 Transfer Length Requirement

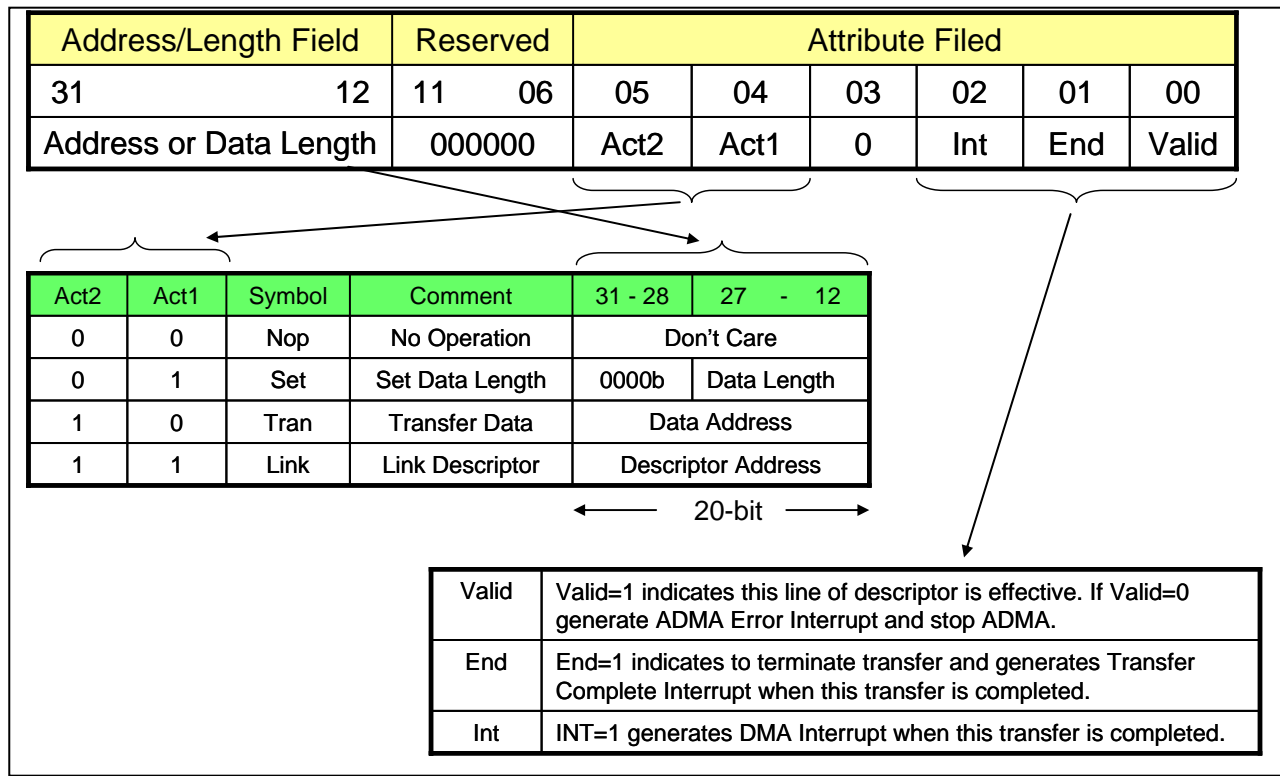
The relation between total transfer length of system memory and that of SD Card is defined as follows:

The amount of data transfer

= multiple of 4K-byte page + n Byte last page

= multiple of card block length designated by the Block Size register.

## C.5 32-bit Address Descriptor Table



**Figure C - 3 : Definition of the 32-bit Descriptor Table for ADMA1**

Figure C - 3 shows the definition of Descriptor Table for ADMA1. The Symbol used in the Descriptor Table is defined in Table C - 1.

Symbol	Operation
Nop	It indicates no operation. The ADMA skips the line of the Descriptor.
Set	The Set operation fetches transfer length from the descriptor and set the length to the Data Length register. Every starting of ADMA, the 4KB default page size is set to Data Length register.
Tran	The Tran operation fetches a system memory address from the descriptor and set the address to the Data Address register. Subsequently, ADMA starts data transfer.
Link	The Link operation fetches a system memory address from the descriptor and set the address to the ADMA System Address register. The address points to the next Descriptor Table. Refer to 2.2.31 about ADMA System Address register for more detail.

**Table C - 1 : Symbol Definition used in Descriptor Table of ADMA1**

## C.6 ADMA States

In the ADMA1 operation, there are 4 states as described in Figure C - 4 and Table C - 2.

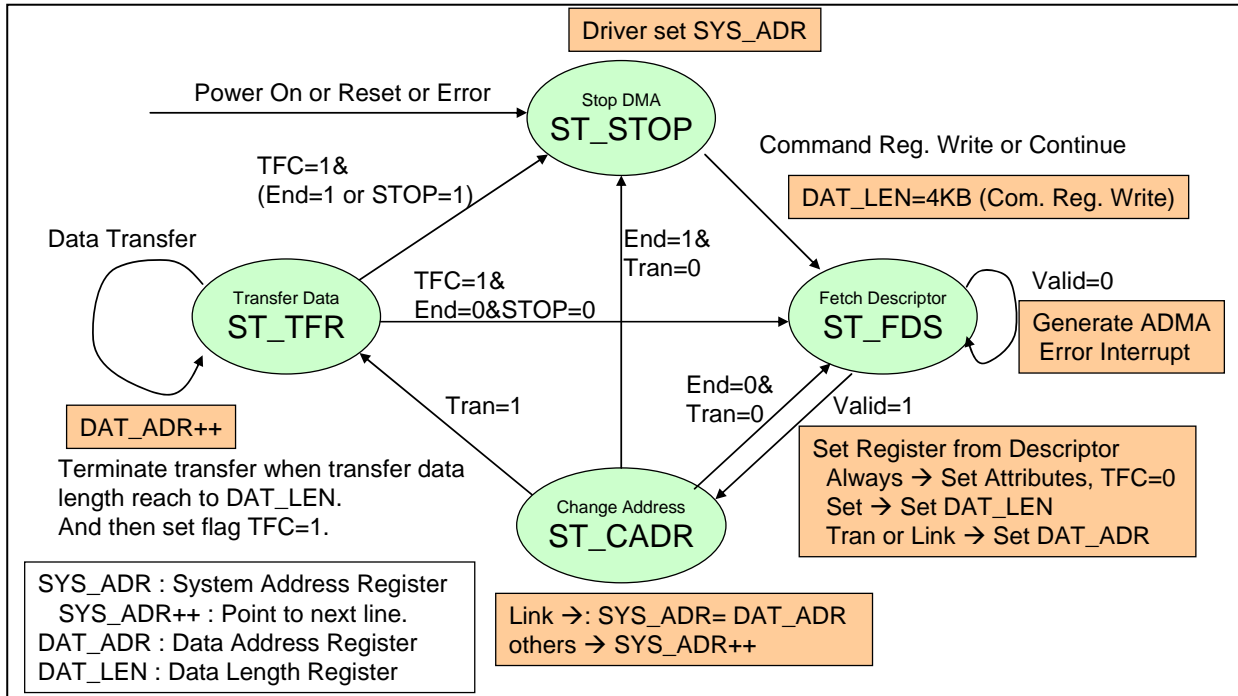


Figure C - 4 : State Diagram of the ADMA1

State Name	Operation
ST_FDS (Fetch Descriptor)	ADMA1 fetches a descriptor line and set parameters in internal registers. Next go to ST_CADR state.
ST_CADR (Change Address)	Link operation loads another Descriptor address to ADMA System Address register. In other operations, ADMA System Address register is incremented to point next descriptor line. If End=0, go to ST_TFR state. ADMA1 shall not be stopped at this state even if some errors occur.
ST_TFR (Transfer Data)	Data transfer of one descriptor line is executed between system memory and SD card. If data transfer continues (End=0) go to ST_FDS state. If data transfer completes, go to ST_STOP state.
ST_STOP (Stop DMA)	ADMA1 stays in this state in following cases: (1) After Power on reset or software reset. (2) All descriptor data transfers are completed If a new ADMA1 operation is started by writing Command register, go to ST_FDS state.

Table C - 2 : ADMA1 States

The Stop At Block Gap Request in the Block Gap Control Register is supported in the ADMA1 data transfer. When the **Stop At Block Gap Request** is set during the ADMA1 operation, the **Block Gap Event Interrupt** will be asserted when the current page data transfer is completed.

In case of read operation from SD card, the Host Controller shall stop read operation of SD card by Read Wait or stopping SD Clock.

## C.7 ADMA1 Control Registers

ADMA1 is selected and supported by following registers.

Location	Attrib	Register Field Explanation
04-03	RW	<b>DMA Select</b> Several types of DMA modes (including No DMA) can be selected by this field. 00 : No DMA or SDMA is selected <b>01 : 32-bit Address ADMA1 is selected</b> 10 : 32-bit Address ADMA2 is selected 11 : 64-bit Address ADMA2 is selected

**Table C - 3 : ADMA1 Select in Host Control Register**

Location	Attrib	Register Field Explanation				
20	Hwlnit	<b>ADMA1 Support</b> This bit indicates whether the Host Controller is capable of using ADMA1. <table border="1" data-bbox="451 808 959 871"> <tbody> <tr> <td>1</td> <td>ADMA Supported</td> </tr> <tr> <td>0</td> <td>ADMA not Supported</td> </tr> </tbody> </table>	1	ADMA Supported	0	ADMA not Supported
1	ADMA Supported					
0	ADMA not Supported					

**Table C - 4 : ADMA1 Support Bit in Capabilities register**