ARASAN

SD/SDIO Host Controller

Userguide

Revision 1.6

02.10.2004

Copyright Notice

Disclaimer

This specification is preliminary and is subject to change at anytime without notice. Arasan Chip Systems, Inc. assumes no responsibility for any errors contained herein.

Questions and / or comments may be directed to:

Arasan Chip Systems Inc.,
1150, North First Street, Ste #211
Sanjose, CA 95112, USA.

Phone: 408-282-1600 X106
Fax : 408-282-7800
Email: support@arasan.com
Web : http://www.arasan.com

Table of Contents

REVISION HISTORY

Revision	Date	Author	Summary Of Changes
1.0	09.15.2003	SD Host Team	Original Document
1.1	07.16.2004	Yakgna, Saritha	Added * Multi Slot host in Functional block diagram * Signal Interfaces for Slot2 * Multi slot description
1.2	08.05.2004	Yakgna	* Added AHB Lite in features * Tuned the DATA FIFO description for Circular FIFO * Tuned table 1 for TARGET_HADDR[6:0] * Removed the CLK48 from Table 4 * Added second slot of registers in Table 5 and sec4.1 * Added e.g. under sec 6.2
1.3	12.03.2004	Yakgna	* Removed CPRM, Multi slot from Features and Functional Block Diagram
1.4	12.07.2004	Yakgna	* Added Randomization commands
1.5	12.13.2004	Yakgna	Tuned FIFO Size
1.6	02.10.2004	Yakgna	Added Clock Section (7.1)

1. General Description

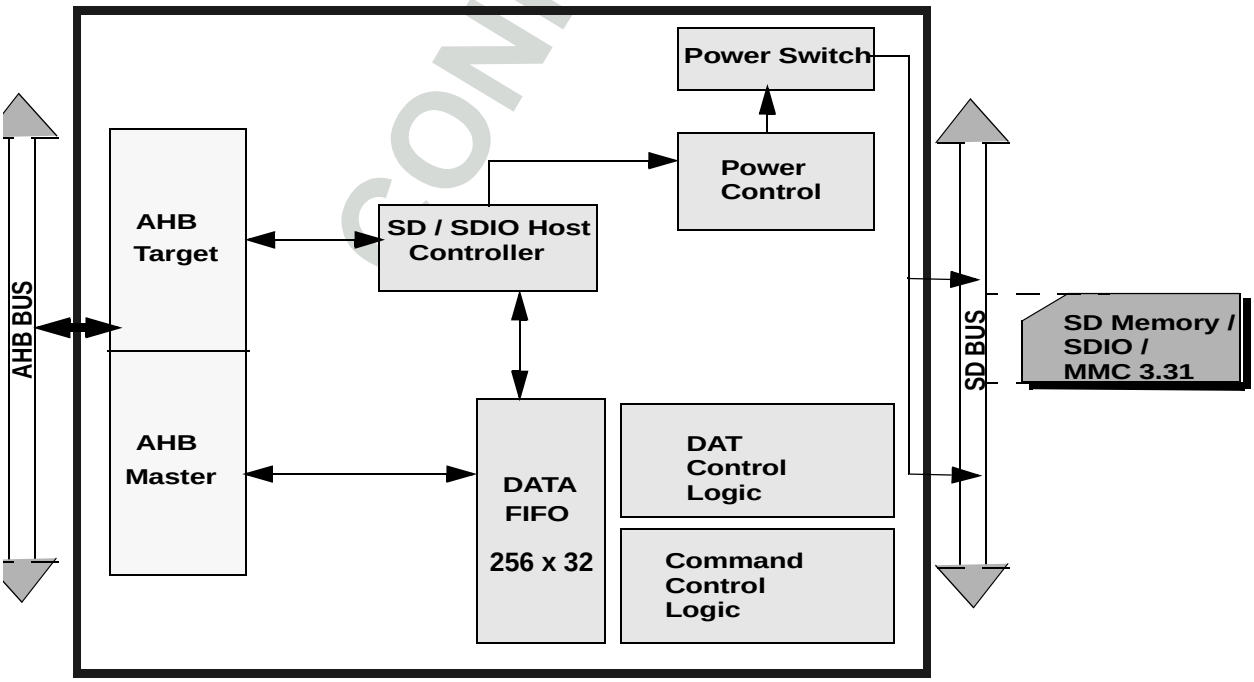
The Arasan SDIO/SD Host Controller is a Host Controller with a ARM processor interface. This product conforms to SD Host Controller Standard Specification Draft Version 1.0. Content R/W protection is supported.Power consumption of the system can be kept to a minimum through gated clock control. The Arasan SDIO/SD Host Controller handles SDIO/SD Protocol at transmission level, packing data, adding cyclic redundancy check (CRC), start/end bit, and checking for transaction format correctness. SD Mode wide bus width is also supported.

The SD/SDIO Host Controller provides Programmed IO method and DMA data transfer method. In programmed IO method, the ARM processor transfer data using the Buffer Data Port Register. Host controller support for DMA can be determined by checking the DMA support in the *Capabilities register*. DMA allows a peripheral to read or write memory without the intervention from the CPU. The *system address* register points to the first data address, and data is then accessed sequentially from that address.

1.1 Features:

- Meets SD Host Controller Standard Specification Draft Version 1.0
- Meets SDIO card specification version 1.0.
- Meets SD Memory Card Security Specification version 1.01.
- Meets MMC Specification version 3.31
- Meets AHB-Lite Specification
- Card Detection (Insertion / Removal)
- Password protection of Cards
- Supports 1 and 4 bit SD modes.
- Allows card to interrupt host in 1 and 4 bit SD modes.
- Upto 12.5Mbytes per second read and write rates using 4 parallel data lines for full speed card
- Cyclic Redundancy Check CRC7 for command and CRC16 for data integrity
- Supports Read wait Control, Suspend/Resume operation.
- Supports Multi Block read and Write
- Conforms to AMBA specification AHB (2.0)

2. Function block diagram



2.1 AHB Master

The AHB master initiates a read or write transaction with the memory if the Data transaction is done using DMA data transfer method.

2.2 AHB Target

The AHB target is having the SD/SDIO control registers and these registers are programmed by the ARM processor through the AHB target interface. The data transaction is performed through the AHB target interface in case of Programmed IO method of data transfer.

2.3 SD/SDIO Host Controller

The SD/SDIO Host Controller comprises of Host_AHB interface, SD/SDIO controller registers, Bus monitor, Clk_gen, CRC generator and checker (CRC7 and CRC16),.

The Host_AHB interface acts as the bridge between the ARM processor and Host Controller. The SD/SDIO controller registers are programmed by the ARM processor through AHB target interface. Interrupts are generated to the ARM processor based on the values set in the Interrupt status register and Interrupt enable registers. Bus monitor will check for any violations occurring in the SD bus and time-out conditions. The Clock generation block will generate the SD clock depending on the value programmed by the ARM processor in the Clock Control Register. The CRC7 and CRC16 generator calculate the CRC for command and Data respectively to send the CRC to the SD/SDIO card. The CRC7 and CRC16 checker checks for any CRC error in the Response and Data send by the SD/SDIO card.

2.4 DATA FIFO

A single 256 X 32 bits Circular FIFO is used for both read and write transactions with maximum block size of 512bytes (FIFO Size is equal to twice the Maximum block size). To increase the throughput, fifo size of twice the maixmum block size is used.

During a write transaction (data transferred from ARM Processor to SD/SDIO card), the data will be filled in to the first half and second half of the fifo alternatively. When data from first half of the fifo is transferring to the SD card, the second half of the fifo will be filled and vice versa. The two half's of fifo are alternatively used to store data which will give maximum throughput. During a read transaction (data transferred from SD/SDIO card to ARM Processor), the data from SD card will be written in to the two half's of the fifo alternatively. When data from one half of the fifo is transferring to the ARM Processor, the second half of the fifo will be filled and vice versa and thereby the throughput will be maximum. If the Host controller cannot accept any data from SD card, then it will issue read wait / stop SD clock to stop the data coming from card.

1. Current block size = Maximum block size

First block is filled in first half of the fifo. While transmitting the first block, the second block of data is received in second half of the fifo.

2. Current block size < Maximum block size

The host controller will receive a block of data either from System memory or Card only there is a house to store a block of data. Say if the current block size is 64 and the maximum block size is 512, at the most 8 blocks are stored in fifo at a time.

Note: Fifo depth can be varied using parameter defined in iface_defines.v.

2.5 Interrupt controller

The SD/SDIO Host Controller generate interrupt to the ARM Processor if any of the interrupt bits are set in the interrupt status register.

2.6 DAT[0-3] Control Logic

The DAT[0-3] control logic block transmit data through the data line during write transaction and receive data through the data line during read transaction. The interrupt generated from the SD/SDIO card is detected.

2.7 Command Control Logic

The Command control logic block sends the command through the cmd line and receive the response coming from the SD/SDIO card.

2.8 Power Control

The SD/SDIO Host Controller supply SD Bus Power depending on the value programmed in the Power Control Register by the ARM Processor. The ARM Processor has the responsibility to supply SD Bus Voltage according to card OCR and supply voltage capabilities depending on the Host Controller. If the SD Bus power is set to 1 in the Power Control Register, the Host Controller shall supply voltage to the Card. If the Host Driver selects an unsupported voltage in the SD Bus Voltage Select field, the Host Controller may ignore write to SD Bus Power and keep its value at zero.

3. Signal Interfaces

The Arasan SD/SDIO Host Controller has four main interface groups:
1. ARM processor Interface signals.
2. SD/SDIO Card Interface that forms the main card interface.
3. Power Supply Controller for supplying bus power.
4. System interface providing the clock and reset signals.

Table 1: AHB Master/Target Interface

Signal	DIR	Description
CLK_AHB	IN	AHB System Clock. SD Clock is derived from AHB System clock.
INT_TO_ARM	OUT	Interrupt to the ARM
M_HADDR[31:0]	OUT	Master Address Bus
M_HWDATA[31:0]	OUT	AHB master write data
M_HRDATA[31:0]	IN	Read data
M_HWRITE	OUT	Write / Read Direction Indication
M_HSIZE[1:0]	OUT	Size (byte, half word or word)
M_HBURST[2:0]	OUT	Burst Size
M_HREADY	IN	Ready signal
M_HTRANS[1:0]	OUT	Transfer type
M_HRESP[1:0]	IN	Transfer response
T_HSEL	IN	Slave Select
T_HADDR[7:0]	IN	Target Address bus
T_HWDATA[31:0]	IN	Write Data
T_HRDATA[31:0]	OUT	Read Data

Table 1: AHB Master/Target Interface

Signal	DIR	Description
T_HWRITE	IN	Write / Read Direction Indication
T_HSIZE[1:0]	IN	Size (Byte, Half Word or Word)
T_HTRANS[1:0]	IN	Transfer Type
T_HREADY	OUT	Slave Ready
T_HRESP[1:0]	OUT	Transfer Response

Table 2: SD/SDIO Card Interface

Signal	DIR	Description
CLK_SDCARD	O	SDIO/SD Clock
CMD CMD_IN CMD_OUT CMD_OUT_EN	 I O O	*SD4 bit mode*: Command Line *SD1 bit mode*: Command Line Command Input Command Output Command Output Enable
DATA0 DATA0_IN DATA0_OUT DATA0_OUT_EN	 I O O	*SD4 bit mode*: Data Line 0 *SD1 bit mode*: Data Line Data0 Input Data0 Output Data0 Output Enable
DATA1 DATA1_IN DATA1_OUT DATA1_OUT_EN	 I O O	*SD4 bit mode*: Data Line1 or Interrupt (optional) *SD1 bit mode*: Interrupt Data1 Input Data1 Output Data1 Output Enable
DATA2 DATA2_IN DATA2_OUT DATA2_OUT_EN	 I O O	SD4 bit mode:Data Line2 or Read Wait (Optional) SD1 bit mode: Read Wait (optional) Data2 Input Data2 Output Data2 Output Enable
DATA3 DATA3_IN DATA3_OUT DATA3_OUT_EN	 I O O	SD4 bit mode:Data Line 3 SD1 bit mode: Not Used Data3 Input Data3 Output Data3 Output Enable
SDCD_n	I	Active Low used for Card Detection ‘0’ - Card is insterted ‘1’ - Card is removed
SDWP_n	I	Active Low use for Card Write Protection ‘0’ - Card is Write protected ‘1’ - Card is not write protected

Table 2: SD/SDIO Card Interface

Signal	DIR	Description
LED_ON	O	LED ON: To Caution the user not to remove the card while the SD card is being accessed.

Table 3: Power Supply Controller

Signal	DIR	Description
BUS_POW	O	Control SD Card Power Supply. 1 - Power ON 0 - Power OFF Before setting this bit, the SD driver shall set SD Bus Voltage select. If the Host Controller detects the No Card State, this bit shall be cleared.
BUS_VOLT[2:0]	O	SD Bus voltage select. 111b - 3.3V 110b - 3.0V 101b - 1.8V (100b - 000b) - Reserved Host Driver program these bits. Based on the Voltage level for the Card Host driver appropriately set these bits. Before setting these register, the Host Driver shall check the Voltage Support bits in the Capabilities register.

Table 4: System Interface

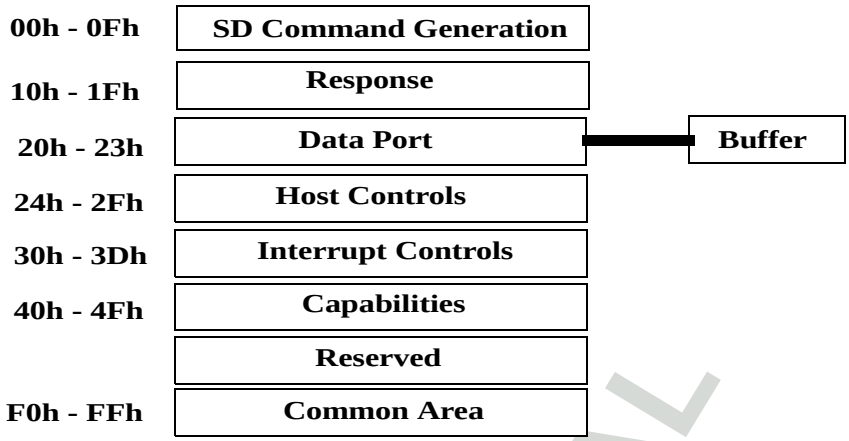
Signal	DIR	Description
RSTAHB_n	I	Active Low System Reset
CLK32	I	32Khz clock is used to detect Card insertion / removal.

4. SD Host Standard Register

Table 5: SD Host Controller Register Map

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

4.1 Classification of the Standard Register Map



4.2 Configuration Register Types

Table 6:

Register Attribute	Description
RO	Read Only Register: Register bits are read only and cannot be altered by software or any reset operation. Write 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 Controller shall clear the bit automatically when the operation of complete. Writing a 0 to RWAC bits has no effect.
Hwinit	Hardware Initialized: Register bits are freezed. 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.

4.3 Registers

4.3.1 System Address Register (offset 000h)

Table 7: ***System Address Register***

Signal	Field	Attrib	After Reset	Description
DMA System Address	31-0	RW	0	This register contains the system memory address for a DMA transfer. When the Host Controller (HC) stops a DMA transfer, this register shall point to the system address of the next contiguous data position. It can be accessed only if no transaction is executing (i.e after a transaction has stopped). Read operations during transfer return an invalid value. The Host Driver (HD) shall initialize this register before starting a DMA transaction. After DMA has stopped, the next system address of the next contiguous data position can be read from this register. The DMA transfer waits at the every boundary specified by the **Host DMA Buffer Size** in the Block Size register. The Host Controller generates **DMA Interrupt** to request to update this register. The HD set the next system address of the next data position to this register. When most upper byte of this register (003h) is written, the HC restart the DMA transfer. When restarting DMA by the resume command or by setting **Continue Request** in the Block Gap Control register, the HC shall start at the next contiguous address stored here in the System Address register

4.3.2 Block Size Register (offset 004h)

Table 8: *Block Size Register*

Signal	Field	Attrib	After Reset	Description
Reserved	15	Rsvd	0	Reserved
Host DMA Buffer Size	14:12	RW	0	To perform long DMA transfer, System Address register shall be updated at every system boundary during DMA transfer. These bits specify the size of contiguous buffer in the system memory. The DMA transfer shall wait at the every boundary specified by these fields and the HC generates the **DMA Interrupt** to request the HD to update the System Address register. These bits shall support when the **DMA Support** in the Capabilities register is set to 1 and this function is active when the **DMA Enable** in the Transfer Mode register is set to 1. *000b - 4KB(Detects A11 Carry out)* *001b - 8KB(Detects A11 Carry out)* *010b - 16KB(Detects A11 Carry out)* *011b - 32KB(Detects A11 Carry out)* *100b - 64KB(Detects A11 Carry out)* *101b -128KB(Detects A11 Carry out)* *110b - 256KB(Detects A11 Carry out)* *111b - 512KB(Detects A11 Carry out)*

Table 8: *Block Size Register*

Signal	Field	Attrib	After Reset	Description
Transfer Block Size	11:0	RW	0	This register specifies the block size for block data transfers for CMD17, CMD18, CMD24, CMD25, and CMD53. It can be accessed only if no transaction is executing (i.e after a transaction has stopped). Read operations during transfer return an invalid value and write operations shall be ignored. 0000h - No Data Transfer 0001h - 1 Byte 0002h - 2 Bytes 0003h - 3 Bytes 0004h - 4 Bytes --- --- 01FFh - 511 Bytes 0200h - 512 Bytes --- --- 0800h - 2048 Bytes

4.3.3 Block Count Register (offset 006h)

Table 9: *Block Count Register*

Signal	Field	Attrib	After Reset	Description
Blocks Count for Current Transfer	15:0	RW	0	This register is enabled when **Block Count Enable** in the Transfer Mode register is set to 1 and is valid only for multiple block transfers. The HC decrements the block count after each block transfer and stops when the count reaches zero. It can be accessed only if no transaction is executing (i.e after a transaction has stopped). Read operations during transfer return an invalid value and write operations shall be ignored. When saving transfer context as a result of Suspend command, the number of blocks yet to be transferred can be determined by reading this register. When restoring transfer context prior to issuing a Resume command, the HD shall restore the previously save block count. 0000h - Stop Count 0001h - 1 block 0002h - 2 blocks --- --- FFFFh - 65535 blocks

4.3.4 Argument Register (offset 008h)

Table 10: *Argument Register*

Signal	Field	Attrib	After Reset	Description
Command Argument	31:0	RW	0	The SD Command Argument is specified as bit39-8 of Command-Format.

4.3.5 Transfer Mode Register (offset 00Ch)

Table 11: *Transfer Mode Register*

Signal	Field	Attrib	After Reset	Description
Reserved	15:6	Rsvd	0	Reserved
Multi / Single Block Select	5	RW	0	This bit enables multiple block DAT line data transfers. 0 - Single Block 1 - Multiple Block
Data Transfer Direction Select	4	RW	0	This bit defines the direction of DAT line data transfers. 0 - Write (Host to Card) 1 - Read (Card to Host)
Reserved	3	Rsvd	0	Reserved
Auto CMD12 Enable	2	RW	0	Multiple block transfers for memory require CMD12 to stop the transaction. When this bit is set to 1, the HC shall issue CMD12 automatically when last block transfer is completed. The HD shall not set this bit to issue commands that do not require CMD12 to stop data transfer. 0 - Disable 1 - Enable
Block Count Enable	1	RW	0	This bit is used to enable the Block count register, which is only relevant for multiple block transfers. When this bit is 0, the Block Count register is disabled, which is useful in executing an infinite transfer. 0 - Disable 1 - Enable
DMA Enable	0	RW	0	DMA can be enabled only if **DMA Support** bit in the Capabilities register is set. If this bit is set to 1, a DMA operation shall begin when the HD writes to the upper byte of Command register (00Fh). 0 - Disable 1 - Enable

Determination of Transfer Type :

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

4.3.6 Command Register (offset 00Eh)

Table 12: *Command Register*

Signal	Field	Attrib	After Reset	Description
Reserved	15:14	Rsvd	0	Reserved
Command Index	13:8	RW	0	This bit shall be set to the command number (CMD0-63, ACMD0-63).

Table 12: *Command Register*

Signal	Field	Attrib	After Reset	Description
Command Type	7:6	RW	0	There are three types of special commands. Suspend, Resume and Abort. These bits shall bet set to 00b for all other commands. ***Suspend Command*** If the Suspend command succeeds, the HC shall assume the SD Bus has been released and that it is possible to issue the next command which uses the DAT line. The HC 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 HC shall maintain its current state. and the HD shall restart the transfer by setting **Continue Request** in the Block Gap Control Register. ***Resume Command*** The HD re-starts the data transfer by restoring the registers in the range of 000-00Dh. The HC shall check for busy before starting write transfers. ***Abort Command*** If this command is set when executing a read transfer, the HC shall stop reads to the buffer. If this command is set when executing a write transfer, the HC shall stop driving the DAT line. After issuing the Abort command, the HD should issue a software reset 00b - Normal 01b - Suspend 10b - Resume 11b - Abort

Table 12: *Command Register*

Signal	Field	Attrib	After Reset	Description
Data Present Select	5	RW	0	This bit is set to 1 to indicate that data is present and shall be transferred using the DAT line. If is set to 0 for the following: 1. Commands using only **CMD** line (ex. CMD52) 2. Commands with no data transfer but using busy signal on **DAT[0]** line (R1b or R5b ex. CMD38) 3. Resume Command 0 - No Data Present 1 - Data Present
Command Index Check Enable	4	RW	0	If this bit is set to 1, the HC 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. 0 - Disable 1 - Enable
Command CRC Check Enable	3	RW	0	If this bit is set to 1, the HC 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. 0 - Disable 1 - Enable
Reserved	2	Rsvd	0	Reserved
Response Type Select	1:0	RW	0	Response Type Select 00 - No Response 01 - Response length 136 10 - Response length 48 11 - Response length 48 check Busy after response

Relation between Parameters and the Name of Response Type

Response Type	Index Check Enable	CRC Check Enable	Name of Response Type
00	0	0	No Response
01	0	1	R2

10	0	0	R3, R4
10	1	1	R1, R6, R5
11	1	1	R1b, R5b

4.3.7 Response Register (offset 010h)

Table 13: *Response Register*

Signal	Field	Attrib	After Reset	Description
Command Response	127:0	ROC	0	The following table describes the mapping of command responses from the SD Bus to this register for each response type. In the table, R[] refers to a bit range within the response data as transmitted on the SD Bus, REP[] refers to a bit range within the Response register.

Response Bit Definition for Each Response Type

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]

4.3.8 Buffer Data Port Register (offset 020h)

Table 14: Buffer Data Port Register

Signal	Field	Attrib	After Reset	Description
Buffer Data	31:0	RW	0	The Host Controller Buffer can be accessed through this 32-bit Data Port Register.

4.3.9 Present State Register (offset 024h)

This bit indicates whether one of the **DAT** line on SD bus is in use.

Table 15: *Present State Register*

Signal	Field	Attrib	After Reset	Description
Reserved	31:25	Rsvd	0	Reserved
CMD Line Signal Level	24	RO	0	This status is used to check **CMD** line level to recover from errors, and for debugging.
DAT[3:0] Line Signal Level	23:20	RO	0	This status is used to check **DAT** line level to recover from errors, and for debugging. This is especially useful in detecting the busy signal level from DAT[0]. D23 - DAT[3] D22 - DAT[2] D21 - DAT[1] D20 - DAT[0]
Write Protect Switch Pin Level	19	RO	0	The Write Protect Switch is supported for memory and combo cards. This bit reflects the SDWP# pin. 0 - Write protected (SDWP# = 1) 1 - Write enabled (SDWP# = 0)
Card Detect Pin Level	18	RO	0	This bit reflects the inverse value of the SDCD# pin. 0 - No Card present (SDCD# = 1) 1 - Card present (SDCD# = 0)

Table 15: *Present State Register*

Signal	Field	Attrib	After Reset	Description
Card State Stable	17	RO	0	This bit is used for testing. If it is 0, the **Card Detect Pin Level** is not stable. If this bit is set to 1, it means the Card Detect Pin Level is stable. The **Software Reset For All** in the Software Reset Register shall not affect this bit. 0 - Reset of Debouncing 1 - No Card or Inserted
Card Inserted	16	RO	0	This bit indicates whether a card has been inserted. Changing from 0 to 1 generates a **Card Insertion** interrupt in the Normal Interrupt Status register and changing from 1 to 0 generates a **Card Removal** Interrupt in the Normal Interrupt Status register. The **Software Reset For All** in the Software Reset register shall not affect this bit. If a Card is removed while its power is on and its clock is oscillating, the HC shall clear **SD Bus Power** in the Power Control register and **SD Clock Enable** in the Clock control register. In addition the HD should clear the HC by the **Software Reset For All** in Software register. The card detect is active regardless of the SD Bus Power. 0 - Reset or Debouncing or No Card 1 - Card Inserted
Reserved	15:12	Rsvd	0	Reserved
Buffer Read Enable	11	ROC	0	This status is used for non-DMA read transfers. This read only flag indicates that valid data exists in the host side buffer status. If this bit is 1, readable data exists in the buffer. A change of this bit from 1 to 0 occurs when all the block data is read from the buffer. A change of this bit from 0 to 1 occurs when all the block data is ready in the buffer and generates the **Buffer Read Ready** Interrupt. 0 - Read Disable 1 - Read Enable.

Table 15: *Present State Register*

Signal	Field	Attrib	After Reset	Description
Buffer Write Enable	10	ROC	0	This status is used for non-DMA write transfers. This read only flag indicates if space is available for write data. If this bit is 1, data can be written to the buffer. A change of this bit from 1 to 0 occurs when all the block data is written to the buffer. A change of this bit from 0 to 1 occurs when top of block data can be written to the buffer and generates the **Buffer Write Ready** Interrupt. 0 - Write Disable 1 - Write Enable.
Read Transfer Active	9	ROC	0	This status is used for detecting completion of a read transfer. This bit is set to 1 for either of the following conditions: 1) After the end bit of the read command 2) When writing a 1 to **continue Request** in the *Block Gap Control* register to restart a read transfer This bit is cleared to 0 for either of the following conditions: 1) When the last data block as specified by block length is transferred to the system. 2) When all valid data blocks have been transferred to the system and no current block transfers are being sent as a result of the **Stop At Block Gap Request** set to 1. A **transfer complete** interrupt is generated when this bit changes to 0. 1 - Transferring data 0 - No valid data

Table 15: *Present State Register*

Signal	Field	Attrib	After Reset	Description
Write Transfer Active	8	ROC	0	This status indicates a write transfer is active. If this bit is 0, it means no valid write data exists in the HC. This bit is set in either of the following cases: 1) After the end bit of the write command. 2) When writing a 1 to **Continue Request** in the *Block Gap Control* register to restart a write transfer. This bit is cleared in either of the following cases: 1) After getting the CRC status of the last data block as specified by the transfer count (Single or Multiple) 2) After getting a CRC status of any block where data transmission is about to be stopped by a **Stop At Block Gap Request**. During a write transaction, a **Block Gap Event** interrupt is generated when this bit is changed to 0, as a result of the **Stop At Block Gap Request** being set. This status is useful for the HD in determining when to issue commands during write busy. 1 - transferring data 0 - No valid data
Reserved	7:3	Rsvd	0	Reserved
DAT Line Active	2	ROC	0	This bit indicates whether one of the **DAT** line on SD bus is in use. 1 - DAT line active 0 - DAT line inactive

Table 15: *Present State Register*

Signal	Field	Attrib	After Reset	Description
Command Inhibit (DAT)	1	ROC	0	This status bit is generated if either the **DAT Line Active** or the **Read transfer Active** is set to 1. If this bit is 0, it indicates the HC can issue the next SD command. Commands with busy signal belong to **Command Inhibit (DAT)** (ex. R1b, R5b type). Changing from 1 to 0 generates a **Transfer Complete** interrupt in the *Normal interrupt status* register. Note: The SD Host Driver can save registers in the range of 000-00Dh for a suspend transaction after this bit has changed from 1 to 0. 1 - cannot issue command which uses the **DAT** line 0 - Can issue command which uses the **DAT** line
Command Inhibit (CMD)	0	ROC	0	If this bit is 0, it indicates the **CMD** line is not in use and the HC can issue a SD command using the **CMD** line. This bit is set immediately after the *Command register (00Fh)* is written. This bit is cleared when the command response is received. Even if the **Command Inhibit (DAT)** is set to 1, Commands using only the **CMD** line can be issued if this bit is 0. Changing from 1 to 0 generates a **Command complete** interrupt in the *Normal Interrupt Status* register. If the HC cannot issue the command because of a command conflict error or because of **Command Not Issued By Auto CMD12 Error**, this bit shall remain 1 and the **Command Complete** is not set. Status issuing Auto CMD12 is not read from this bit.

Note: **DAT line active** indicates whether one of the **DAT** line is on SD bus is in use.
a) In the case of read transactions
This status indicates if a read transfer is executing on the SD bus. Changes in this value from 1 to 0 between data blocks generates a **Block Gap Event** interrupt in the *Normal Interrupt Status* register. This bit shall be set in either of the following cases:
1) After the end bit of the read command.

2) When writing a 1 to **Continue Request** in the *Block Gap Control* register to restart a read transfer.
This bit shall be cleared in either of the following cases:
1) When the end bit of the last data block is sent from the SD bus to the HC.
2) When writing a 1 to **Continue Request** in the *Block Gap Control* register to restart a read transfer.
This bit shall be cleared in either of the following cases:
1) When the end bit of the last data block is sent from the SD bus to the HC.
2) When beginning a wait read transfer at a stop at the block gap initiated by a **Stop At Block Gap Request**
The HC shall wait at the next block gap by driving Read Wait at the start of the interrupt cycle. If the Read Wait signal is already driven (data buffer cannot receive data),the HC can wait for current block gap by continuing to drive the Read Wait signal. It is necessary to support Read Wait in order to use the suspend / resume function.

b) In the case of write transactions
This status indicates that a write transfer is executing on the SD bus. Changing this value from 1 to 0 generate a **Transfer complete** interrupt in the *Normal Interrupt status* register.
This bit shall be set in either of the following cases:
1) After the end of the write command.
2) When writing to 1 to **Continue Request** in the *Block Gap Control* register to continue a write transfer.
This bit shall be cleared in either of the following cases:
1) When the SD card releases write busy of the last data block the HC shall also detect if output is not busy. If SD card does not drive busy signal for 8 SD clocks, the HC shall consider the card drive “Not Busy”.
2) When the SD card releases write busy prior to waiting for write transfer as a result of a **Stop At block Gap Request**.

Implementation Note:
The HD 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.

4.3.10 Host Control Register (offset 028h)

Table 16: *Host Control Register*

Signal	Field	Attrib	After Reset	Description
Reserved	7:3	Rsvd	0	Reserved
High Speed Enable	2	RW	0	This bit is optional. Before setting this bit, the HD shall check the **High Speed Support** in the *capabilities* register. If this bit is set to 0 (default), the HC outputs **CMD** line and **DAT** lines at the falling edge of the SD clock (up to 25 MHz). If this bit is set to 1, the HC outputs **CMD** line and **DAT** lines at the rising edge of the SD clock (up to 50 MHz) 1 - High Speed Mode 0 - Normal Speed Mode

Table 16: *Host Control Register*

Signal	Field	Attrib	After Reset	Description
Data Transfer Width (SD1 or SD4)	1	RW	0	This bit selects the data width of the HC. The HD shall select it to match the data width of the SD card. 1 - 4 bit mode 0 - 1 bit mode
LED Control	0	RW	0	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 transactions. It is not necessary to change for each transaction. 1 - LED on 0 - LED off

4.3.11 Power Control Register (offset 029h)

Table 17: *Power Control Register*

Signal	Field	Attrib	After Reset	Description
Reserved	7:4	Rsvd	0	Reserved
SD Bus Voltage Select	3:1	RW	0	By setting these bits, the HD selects the voltage level for the SD card. Before setting this register, the HD shall check the **voltage support** bits in the *capabilities* register. If an unsupported voltage is selected, the Host System shall not supply SD bus voltage 111b - 3.3 V(Typ.) 110b - 3.0 V(Typ.) 101b - 1.8 V(Typ.) 100b - 000b - Reserved
SD Bus Power	0	RW	0	Before setting this bit, the SD host driver shall set SD Bus Voltage Select. If the HC detects the No Card State, this bit shall be cleared. 1 - Power on 0 - Power off

4.3.12 Block Gap Control Register (offset 02Ah)

Table 18: *Block Gap Control Register*

Signal	Field	Attrib	After Reset	Description
Reserved	7:4	Rsvd	0	Reserved
Interrupt At Block Gap	3	RW	0	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. If the SD card cannot signal an interrupt during a multiple block transfer, this bit should be set to 0. When the HD detects an SD card insertion, it shall set this bit according to the CCCR of the SDIO card.
Read Wait Control	2	RW	0	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 **DAT[2]** line. Otherwise the HC has to stop the SD clock to hold read data, which restricts commands generation. When the HD detects an SD card insertion, it shall set this bit according to the CCCR of the SDIO card. If the card does not support read wait, this bit shall never be set to 1 otherwise **DAT** line conflict may occur. If this bit is set to 0, Suspend / Resume cannot be supported 1 - Enable Read Wait Control 0 - Disable Read Wait Control

Table 18: *Block Gap Control Register*

Signal	Field	Attrib	After Reset	Description
Continue Request	1	RWAC	0	This bit is used to restart a transaction which was stopped using the **Stop At Block Gap Request.** To cancel stop at the block gap, set **Stop At block Gap Request** to 0 and set this bit to restart the transfer. The HC automatically clears this bit in either of the following cases: 1) In the case of a read transaction, the **DAT Line Active** changes from 0 to 1 as a read transaction restarts. 2) In the case of a write transaction, the **Write transfer active** changes from 0 to 1 as the write transaction restarts. Therefore it is not necessary for Host driver to set this bit to 0. If **Stop At Block Gap Request** is set to 1, any write to this bit is ignored. 1 - Restart 0 - Ignored

Table 18: *Block Gap Control Register*

Signal	Field	Attrib	After Reset	Description
Stop At Block Gap Request	0	RW	0	This bit is used to stop executing a transaction at the next block gap for both DMA and non- DMA transfers. Until the transfer complete is set to 1, indicating a transfer completion the HD shall leave this bit set to 1. Clearing both the Stop At Block Gap Request and Continue Request shall not cause the transaction to restart. Read Wait is used to stop the read transaction at the block gap. The HC shall honour **Stop At Block Gap Request** for write transfers, but for read transfers it requires that the SD card support Read Wait. Therefore the HD shall not set this bit during read transfers unless the SD card supports Read Wait and has set **Read Wait Control** to 1. In case of write transfers in which the HD writes data to the *Buffer Data Port* register, the HD shall set this bit after all block data is written. If this bit is set to 1, the HD shall not write data to *Buffer data port* register. This bit affects **Read Transfer Active**, **Write Transfer Active**, **DAT line active** and **Command Inhibit (DAT)** in the *Present State* register. 1 - Stop 0 - Transfer

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

1) If the HD does not issue Suspend command, the **Continue Request** shall be used to restart the transfer.

2) If the HD issues a Suspend command and the SD card accepts it, a **Resume Command** shall be used to restart the transfer.

3) If the HD 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 HD 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 HD shall clear **Stop At Block Gap Request** before or simultaneously.

4.3.13 Wakeup Control Register (offset 02Ah)

This register is mandatory for the HC, but wakeup functionality depends on the HC system hardware and software. The HD 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.

Table 19: *Wakeup Control Register*

Signal	Field	Attrib	After Reset	Description
Reserved	7:3	Rsvd	0	Reserved
Wakeup Event Enable On SD Card Removal	2	RW	0	This bit enables wakeup event via **Card Removal** assertion in the *Normal Interrupt Status* register. **FN_WUS** (Wake up Support) in CIS does not affect this bit. 1 - Enable 0 - Disable
Wakeup Event Enable On SD Card Insertion	1	RW	0	This bit enables wakeup event via **Card Insertion** assertion in the *Normal Interrupt Status* register. **FN_WUS** (Wake up Support) in CIS does not affect this bit. 1 - Enable 0 - Disable
Wakeup Event Enable On Card Interrupt	0	RW	0	This bit enables wakeup event via **Card Interrupt** assertion in the *Normal Interrupt Status* register. This bit can be set to 1 if FN_WUS (Wake Up Support) in CIS is set to 1. 1 - Enable 0 - Disable

4.3.14Clock Control Register (offset 02Ch)

At the initialization of the HC, the HD shall set the **SDCLK Frequency Select** according to the *Capabilities* register.

Table 20: *Clock Control Register*

Signal	Field	Attrib	After Reset	Description
SDCLK Frequency Select	15:8	RW	0	This register is used to select the frequency of the **SDCLK** pin. The frequency is not programmed directly; rather this register holds the divisor of the **Base Clock Frequency For SD clock** in the *capabilities* register. Only the following settings are allowed. 80h - base clock divided by 256 40h - base clock divided by 128 20h - base clock divided by 64 10h - base clock divided by 32 08h - base clock divided by 16 04h - base clock divided by 8 02h - base clock divided by 4 01h - base clock divided by 2 00h - base clock(10MHz-63MHz) Setting 00h specifies the highest frequency of the SD Clock. When setting multiple bits, the most significant bit is used as the divisor. But multiple bits should not be set. The two default divider values can be calculated by the frequency that is defined by the **Base Clock Frequency For SD Clock** in the *Capabilities* register. 1) 25 MHz divider value 2) 400 KHz divider value The frequency of the SDCLK is set by the following formula: Clock Frequency = (Baseclock) / divisor . Thus choose the smallest possible divisor which results in a clock frequency that is less than or equal to the target frequency.

Table 20: *Clock Control Register*

Signal	Field	Attrib	After Reset	Description
Reserved	7:3	Rsvd	0	Reserved
SD Clock Enable	2	RW	0	The HC shall stop **SDCLK** when writing this bit to 0. **SDCLK frequency Select** can be changed when this bit is 0. Then, the HC shall maintain the same clock frequency until **SDCLK** is stopped (Stop at **SDCLK** = 0). If the HC detects the No Card state, this bit shall be cleared. 1 - Enable 0 - Disable
Internal Clock Stable	1	ROC	0	This bit is set to 1 when SD clock is stable after writing to **Internal Clock Enable** in this register to 1. The SD Host Driver shall wait to set **SD Clock Enable** until this bit is set to 1. Note : This is useful when using PLL for a clock oscillator that requires setup time. 1 - Ready 0 - Not Ready
Internal Clock Enable	0	RW	0	This bit is set to 0 when the HD is not using the HC or the HC awaits a wakeup event. The HC 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 HC shall set **Internal Clock Stable** in this register to 1. This bit shall not affect card detection. 1 - Oscillate 0 - Stop

4.3.15 Timeout Control Register (offset 02Eh)

Table 21: *Timeout Control Register*

Signal	Field	Attrib	After Reset	Description
Reserved	7:4	Rsvd	0	Reserved
Data Timeout Counter Value	3:0	RW	0	This value determines the interval by which DAT line timeouts are detected. Refer to the **Data Timeout Error** in the *Error Interrupt Status* register for information on factors that dictate timeout generation. Timeout clock frequency will be generated by dividing the base clock TMCLK by this value. When setting this register, prevent inadvertent timeout events by clearing the **Data Timeout Error Status Enable** (in the *Error Interrupt Status Enable* register) 1111 - Reserved 1110 - TMCLK * 2^{27} ---------- ---------- 0001 - TMCLK * 2^{14} 0000 - TMCLK * 2^{13}

At the initialization of the HC, the HD shall set the **Data Timeout Counter Value** according to the *Capabilities* register.

4.3.16 Software Reset Register (offset 02Fh)

Table 22: *Software Reset Register*

Signal	Field	Attrib	After Reset	Description
Reserved	7:3	Rsvd	0	Reserved
Software Reset for DAT Line	2	RWAC	0	Only part of data circuit is reset. DMA circuit is also reset. The following registers and bits are cleared by this bit : *Buffer Data Port Register* Buffer is cleared and Initialized. *Present State* register **Buffer read Enable** **Buffer write Enable** **Read Transfer Active** **Write Transfer Active** **DAT Line Active** **Command Inhibit (DAT)** *Block Gap Control* register **Continue Request** **Stop At Block Gap Request** *Normal Interrupt Status* register **Buffer Read Ready** **Buffer Write Ready** **Block Gap Event** **Transfer Complete** 1 - Reset 0 - Work
Software Reset for CMD Line	1	RWAC	0	Only part of command circuit is reset. The following registers and bits are cleared by this bit : *Present State* register **Command Inhibit (CMD)** *Normal Interrupt Status* register **Command Complete** 1 - Reset 0 - Work

Table 22: *Software Reset Register*

Signal	Field	Attrib	After Reset	Description
Software Reset for All	0	RWAC	0	This reset affects the entire HC except for the card detection circuit. Register bits of type ROC, RW, RW1C, RWAC are cleared to 0. During its initialization, the HD shall set this bit to 1 to reset the HC. The HC shall reset this bit to 0 when capabilities registers are valid and the HD can read them. Additional use of **Software Reset For All** may not affect the value of the *Capabilities* registers. If this bit is set to 1, the SD card shall reset itself and must be reinitialized by the HD. 1 - Reset 0 - Work

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

4.3.17 Normal Interrupt Status Register (offset 030h)

The *Normal Interrupt Status Enable* affects read of this register, but *Normal Interrupt Signal* does not affect these reads. An Interrupt is generated when the Normal Interrupt Signal Enable is enabled and atleast 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. The **Card Interrupt** is cleared when the card stops asserting the interrupt: that is when the Card Driver services the Interrupt condition.

Table 23: *Normal Interrupt Status Register*

Signal	Field	Attrib	After Reset	Description
Error Interrupt	15	ROC	0	If any of the bits in the *Error Interrupt Status Register* are set, then this bit is set. Therefore the HD can test for an error by checking this bit first. 0 - No Error. 1 - Error.
Reserved	14:9	Rsvd	0	Reserved

Table 23: *Normal Interrupt Status Register*

Signal	Field	Attrib	After Reset	Description
Card Interrupt	8	ROC	0	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 HC 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 card and the interrupt to the Host system. when this status has been set and the HD needs to start this interrupt service, **Card Interrupt Status Enable** in the *Normal Interrupt Status* register shall be set to 0 inorder to clear the card interrupt statuses latched in the HC and stop driving the Host System. After completion of the card interrupt service (the reset factor in the SD card and the interrupt signal may not be asserted), set Card Interrupt Status Enable to 1 and start sampling the interrupt signal again. 0 - No Card Interrupt 1 - Generate Card Interrupt
Card Removal	7	RW1C	0	This status is set if the **Card Inserted** in the *Present State* register changes from 1 to 0. When the HD writes this bit to 1 to clear this status the status of the Card Inserted in the Present State register should be confirmed. Because the card detect may possibly be changed when the HD clear this bit an Interrupt event may not be generated. 0 - Card State Stable or Debouncing 1 - Card Removed

Table 23: *Normal Interrupt Status Register*

Signal	Field	Attrib	After Reset	Description
Card Insertion	6	RW1C	0	This status is set if the **Card Inserted** in the *Present State* register changes from 0 to 1. When the HD writes this bit to 1 to clear this status the status of the Card Inserted in the Present State register should be confirmed. Because the card detect may possibly be changed when the HD clear this bit an Interrupt event may not be generated. 0 - Card State Stable or Debouncing 1 - Card Inserted
Buffer Read Ready	5	RW1C	0	This status is set if the **Buffer Read Enable** changes from 0 to 1. 0 - Not Ready to read Buffer. 1 - Ready to read Buffer.
Buffer Write Ready	4	RW1C	0	This status is set if the **Buffer Write Enable** changes from 0 to 1. 0 - Not Ready to Write Buffer. 1 - Ready to Write Buffer.
DMA Interrupt	3	RW1C	0	This status is set if the HC detects the Host DMA Buffer Boundary in the Block Size regiser. 0 - No DMA Interrupt 1 - DMA Interrupt is Generated

Table 23: ***Normal Interrupt Status Register***

Signal	Field	Attrib	After Reset	Description
Block Gap Event	2	RW1C	0	If the **Stop At Block Gap Request** in the *Block Gap Control Register* is set, this bit is set. *Read Transaction :* This bit is set at the falling edge of the **DAT Line Active** Status (When the transaction is stopped at SD Bus timing. The Read Wait must be supported inorder to use this function). *Write Transaction :* This bit is set at the falling edge of **Write Transfer Active** Status (After getting CRC status at SD Bus timing). 0 - No **Block Gap Event** 1 - Transaction stopped at Block Gap

Table 23: *Normal Interrupt Status Register*

Signal	Field	Attrib	After Reset	Description
Transfer Complete	1	RW1C	0	This bit is set when a read / write transaction is completed. *Read Transaction :* This bit is set at the falling edge of **Read Transfer Active** Status. There are two cases in which the Interrupt is generated. The first is when a data transfer is completed as specified by data length (After the last data has been read to the Host System). The second is when data has stopped at the block gap and completed the data transfer by setting the **Stop At Block Gap Request** in the *Block Gap Control Register* (After valid data has been read to the Host System). *Write Transaction :* This bit is set at the falling edge of the **DAT Line Active** Status. There are two cases in which the Interrupt is generated. The first is when the last data is written to the card as specified by data length and Busy signal is released. The second is when data transfers are stopped at the block gap by setting **Stop At Block Gap Request** in the *Block Gap Control Register* and data transfers completed. (After valid data is written to the SD card and the busy signal is released). Note : **Transfer Complete** has higher priority than **Data Timeout Error.** If both bits are set to 1, the data transfer can be considered complete 0 - No Data Transfer Complete 1 - Data Transfer Complete

Table 23: *Normal Interrupt Status Register*

Signal	Field	Attrib	After Reset	Description
Command Complete	0	RW1C	0	This bit is set when get the end bit of the command response (Except Auto CMD12). Note : **Command Timeout Error** has higher priority than Command Complete. If both are set to 1, it can be considered that the response was not received correctly. 0 - No Command Complete 1 - Command Complete

Table 24: Relation between Transfer Complete and Data Timeout Error

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	Data Transfer Complete

Table 25: Relation between Command Complete and Command Timeout Error

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

4.3.18 Error Interrupt Status Register (offset 032h)

Status 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.

Table 26: *Error Interrupt Status Register*

Signal	Field	Attrib	After Reset	Description
Vendor Specific Error Status	15:12	RW1C	0	Additional status bits can be defined in this register by the vendor.
Reserved	11:9	Rsvd	0	Reserved
Auto CMD12 Error	8	RW1C	0	Occurs when detecting that one of the bits in *Auto CMD12 Error Status register* has changed from 0 to 1. This bit is set to 1 also when Auto CMD12 is not executed due to the previous command error. 0 - No Error 1 - Error
Current Limit Error	7	RW1C	0	By setting the **SD Bus Power** bit in the *Power Control Register,* the HC is requested to supply power for the SD Bus. If the HC 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 HC is not supplying power to SD card due to some failure. Reading 0 means that the HC is supplying power and no error has occurred. This bit shall always set to be 0, if the HC does not support this function. 0 - No Error 1 - Power Fail
Data End Bit Error	6	RW1C	0	Occurs when detecting 0 at the end bit position of read data which uses the DAT line or the end bit position of the CRC status. 0 - No Error 1 - Error

Table 26: *Error Interrupt Status Register*

Signal	Field	Attrib	After Reset	Description
Data CRC Error	5	RW1C	0	Occurs when detecting CRC error when transferring read data which uses the DAT line or when detecting the Write CRC Status having a value of other than “010”. 0 - No Error 1 - Error
Data Timeout Error	4	RW1C	0	Occurs when detecting one of following timeout conditions. 1) Busy Timeout for R1b, R5b type. 2) Busy Timeout after Write CRC status 3) Write CRC status Timeout 4) Read Data Timeout 0 - No Error 1 - Timeout
Command Index Error	3	RW1C	0	Occurs if a Command Index error occurs in the Command Response. 0 - No Error 1 - Error
Command End Bit Error	2	RW1C	0	Occurs when detecting that the end bit of a command response is 0. 0 - No Error 1 - End Bit Error Generated

Table 26: *Error Interrupt Status Register*

Signal	Field	Attrib	After Reset	Description
Command CRC Error	1	RW1C	0	Command CRC Error is generated in two cases. 1) If a response is returned and the **Command Timeout Error** is set to 0, this bit is set to 1 when detecting a crc error in the command response 2) The HC detects a ***CMD*** line conflict by monitoring the ***CMD*** line when a command is issued. If the HC drives the ***CMD*** line to 1 level, but detects 0 level on the ***CMD*** line at the next SDCLK edge, then the HC shall abort the command (Stop driving ***CMD*** line) and set this bit to 1. The Command Timeout Error shall also be set to 1 to distinguish ***CMD*** line conflict. 0 - No Error 1 - CRC Error Generated
Command Timeout Error	0	RW1C	0	Occurs only if the no response is returned within 64 ***SDCLK*** cycles from the end bit of the command. If the HC detects a ***CMD*** line conflict, in which case **Command CRC Error** shall also be set. This bit shall be set without waiting for 64 **SDCLK** cycles because the command will be aborted by the HC. 0 - No Error 1 - Timeout

Relation between Command CRC Error and Command Timeout Error

Command CRC Error	Command Time-out Error	Kinds of Error
0	0	No Error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD Line Conflict

4.3.19 Normal Interrupt Status Enable Register (offset 034h)

Setting to 1 enables Interrupt Status.

Table 27: *Normal Interrupt Status Enable Register*

Signal	Field	Attrib	After Reset	Description
Fixed to 0	15	RO	0	The HC shall control error Interrupts using the *Error Interrupt Status Enable* register.
Reserved	14:9	Rsvd	0	Reserved
Card Interrupt Status Enable	8	RW	0	If this bit is set to 0, the HC shall clear Interrupt request to the System. The **Card Interrupt** detection is stopped when this bit is cleared and restarted when this bit is set to 1. The HD should clear the **Card Interrupt Status Enable** before servicing the **Card Interrupt** and should set this bit again after all Interrupt requests from the card are cleared to prevent inadvertent Interrupts. 0 - Masked 1 - Enabled
Card Removal Status Enable	7	RW	0	0 - Masked 1 - Enabled
Card Insertion Status Enable	6	RW	0	0 - Masked 1 - Enabled
Buffer Read Ready Status Enable	5	RW	0	0 - Masked 1 - Enabled
Buffer Write Ready Status Enable	4	RW	0	0 - Masked 1 - Enabled
DMA Interrupt Status Enable	3	RW	0	0 - Masked 1 - Enabled
Block Gap Event Status Enable	2	RW	0	0 - Masked 1 - Enabled
Transfer Complete Status Enable	1	RW	0	0 - Masked 1 - Enabled
Command Complete Status Enable	0	RW	0	0 - Masked 1 - Enabled

Note: The HC 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 HC shall clear all internal signals regarding **Card Interrupt.**

4.3.20 Error Interrupt Status Enable Register (offset 036h)

Setting to 1 enables Interrupt Status.

Table 28: ***Error Interrupt Status Enable Register***

Signal	Field	Attrib	After Reset	Description
Vendor Specific Error Status Enable	15:12	RW	0	0 - Masked 1 - Enabled
Reserved	11:9	Rsvd	0	Reserved
Auto CMD12 Error Status Enable	8	RW	0	0 - Masked 1 - Enabled
Current Limit Error Status Enable	7	RW	0	0 - Masked 1 - Enabled
Data End Bit Error Status Enable	6	RW	0	0 - Masked 1 - Enabled
Data CRC Error Status Enable	5	RW	0	0 - Masked 1 - Enabled
Data Timeout Error Status Enable	4	RW	0	0 - Masked 1 - Enabled
Command Index Error Status Enable	3	RW	0	0 - Masked 1 - Enabled
Command End Bit Error Status Enable	2	RW	0	0 - Masked 1 - Enabled
Command CRC Error Status Enable	1	RW	0	0 - Masked 1 - Enabled
Command Timeout Error Status Enable	0	RW	0	0 - Masked 1 - Enabled

Note: To Detect CMD Line conflict, the HD must set both **Command Time-out Error Status Enable** and **Command CRC Error Status Enable** to 1.

4.3.21 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 sample 1 bit interrupt line. Setting any of these bits to 1 enables Interrupt generation.

Table 29: *Normal Interrupt Signal Enable Register*

Signal	Field	Attrib	After Reset	Description
Fixed to 0	15	RO	0	The HD shall control error Interrupts using the *Error Interrupt Signal Enable* register.
Reserved	14:9	Rsvd	0	Reserved
Card Interrupt Signal Enable	8	RW	0	0 - Masked 1 - Enabled
Card Removal Signal Enable	7	RW	0	0 - Masked 1 - Enabled
Card Insertion Signal Enable	6	RW	0	0 - Masked 1 - Enabled
Buffer Read Ready Signal Enable	5	RW	0	0 - Masked 1 - Enabled
Buffer Write Ready Signal Enable	4	RW	0	0 - Masked 1 - Enabled
DMA Interrupt Signal Enable	3	RW	0	0 - Masked 1 - Enabled
Block Gap Event Signal Enable	2	RW	0	0 - Masked 1 - Enabled
Transfer Complete Signal Enable	1	RW	0	0 - Masked 1 - Enabled
Command Complete Signal Enable	0	RW	0	0 - Masked 1 - Enabled

4.3.22 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.

Table 30: *Error Interrupt Signal Enable Register*

Signal	Field	Attrib	After Reset	Description
Vendor Specific Error Signal Enable	15:12	RW	0	0 - Masked 1 - Enabled
Reserved	11:9	Rsvd	0	Reserved
Auto CMD12 Error Signal Enable	8	RW	0	0 - Masked 1 - Enabled
Current Limit Error Signal Enable	7	RW	0	0 - Masked 1 - Enabled
Data End Bit Error Signal Enable	6	RW	0	0 - Masked 1 - Enabled
Data CRC Error Signal Enable	5	RW	0	0 - Masked 1 - Enabled
Data Timeout Error Signal Enable	4	RW	0	0 - Masked 1 - Enabled
Command Index Error Signal Enable	3	RW	0	0 - Masked 1 - Enabled
Command End Bit Error Signal Enable	2	RW	0	0 - Masked 1 - Enabled
Command CRC Error Signal Enable	1	RW	0	0 - Masked 1 - Enabled
Command Timeout Error Signal Enable	0	RW	0	0 - Masked 1 - Enabled

4.3.23 Auto CMD12 Error Status Register (offset 03Ch)

When *Auto CMD12 Error Status* is set, the HD 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.

Table 31: *Auto CMD12 Error Status Register*

Signal	Field	Attrib	After Reset	Description
Reserved	15:8	Rsvd	0	Reserved

Table 31: *Auto CMD12 Error Status Register*

Signal	Field	Attrib	After Reset	Description
Command Not Issued By Auto CMD12 Error	7	ROC	0	Setting this bit to 1 means CMD_wo_DAT is not executed due to an Auto CMD12 error (D04 - D01) in this register. 0 - No Error 1 - Not Issued
Reserved	6:5	Rsvd	0	Reserved
Auto CMD12 Index Error	4	ROC	0	Occurs if the Command Index error occurs in response to a command. 0 - No Error 1 - Error
Auto CMD12 End Bit Error	3	ROC	0	Occurs when detecting that the end bit of command response is 0. 0 - No Error 1 - End Bit Error Generated
Auto CMD12 CRC Error	2	ROC	0	Occurs when detecting a CRC error in the command response. 0 - No Error 1 - CRC Error Generated
Auto CMD12 Timeout Error	1	ROC	0	Occurs if the no response is returned within 64 ***SDCLK*** cycles from the end bit of the command. If this bit is set to 1, the other error status bits (D04 - D02) are meaningless. 0 - No Error 1 - Timeout
Auto CMD12 not Executed	0	ROC	0	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 HC cannot issue Auto CMD12 to stop memory multiple block transfer due to some error. If this bit is set to 1, other error status bits (D04 - D01) are meaningless. 0 - Executed 1 - Not Executed

Table 32: Relation between Auto CMD12 CRC Error and Auto CMD12 Timeout Error

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

The timing of changing Auto CMD12 Error Status can be classified in three scenarios:
1) When the HC 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 Auto CMD12 response.
Check received responses by checking the error bits D01, D02, D03, 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.

4.3.24 Capabilities Register (offset 040h)
This register provides the HD with information specific to the HC implementation. The HC may implement these values as fixed or loaded from flash memory during power on initialization.

Table 33: *Capabilities Register*

Signal	Field	Attrib	After Reset	Description
Reserved	63:32	Rsvd	0	Reserved
Reserved	31:27	Rsvd	0	Reserved
Voltage Support 1.8 V	26	HwInit	1	0 - 1.8 V Not Supported 1 - 1.8 V Supported
Voltage Support 3.0 V	25	HwInit	1	0 - 3.0 V Not Supported 1 - 3.0 V Supported
Voltage Support 3.3 V	24	HwInit	1	0 - 3.3 V Not Supported 1 - 3.3 V Supported

Table 33: *Capabilities Register*

Signal	Field	Attrib	After Reset	Description
Suspend / Resume Support	23	HwInit	1	This bit indicates whether the HC supports Suspend / Resume functionality. If this bit is 0, the Suspend and Resume mechanism are not supported and the HD shall not issue either Suspend / Resume commands. 0 - Not Supported 1 - Supported
DMA Support	22	HwInit	1	This bit indicates whether the HC is capable of using DMA to transfer data between system memory and the HC directly. 0 - DMA Not Supported 1 - DMA Supported.
High Speed Support	21	Hwinit	1	This bit indicates whether the HC and the Host System support High Speed mode and they can supply SD Clock frequency from 25Mhz to 50 Mhz. 0 - High Speed Not Supported 1 - High Speed Supported
Reserved	21:18	HwInit	0	Reserved
Max Block Length	17:16	HwInit	01	This value indicates the maximum block size that the HD can read and write to the buffer in the HC. The buffer shall transfer this block size without wait cycles. Three sizes can be defined as indicated below. 00 - 512 byte 01 - 1024 byte 10 - 2048 byte 11 - Reserved
Reserved	15:14	Rsvd	0	Reserved

Table 33: *Capabilities Register*

Signal	Field	Attrib	After Reset	Description
Base Clock Frequency for SD Clock	13:8	HwInit	110000b	This value indicates the base (maximum) clock frequency for the SD clock. Unit values are 1Mhz. If the real frequency is 16.5 Mhz, the larger value shall be set 010001b (17 Mhz) because the HD uses this value to calculate the clock divider value and it shall not exceed the upper limit of the SD clock frequency. The supported range is 10Mhz to 63 Mhz. If these bits are all 0, the Host System has to get information via another method. Not 0 - 1 Mhz to 63 Mhz 000000b - Get information via another method **(Registry Entry)**.
Timeout Clock Unit	7	HwInit	1	This bit shows the unit of base clock frequency used to detect **Data Timeout Error.** 0 - Khz 1 - Mhz
Reserved	6	HwInit	0	Reserved
Timeout Clock Frequency	5:0	HwInit	110000b	This bit shows the base clock frequency used to detect **Data Timeout Error.** Not 0 - 1Khz to 63Khz or 1Mhz to 63Mhz 000000b - Get Information via another method.

Note: The Host System shall support at least one of these voltages above. The HD 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 the OCR value from the card.

These registers indicate maximum current capability for each voltage. The value is meaningful if **Voltage Support** is set in the *Capabilities* register.

4.3.25 Maximum Current Capabilities Register (offset 048h)

Table 34: ***Maximum Current Capabilities Register***

Signal	Field	Attrib	After Reset	Description
Reserved	63:56	Rsvd	0	Reserved
Reserved	55:48	Rsvd	0	Reserved
Reserved	47:40	Rsvd	0	Reserved
Reserved	39:32	Rsvd	0	Reserved
Reserved	31:24	Rsvd	0	Reserved
Maximum Current for 1.8V	23:16	HwInit	FFh	Maximum Current for 1.8V
Maximum Current for 3.0V	15:8	HwInit	FFh	Maximum Current for 3.0V
Maximum Current for 3.3V	7:0	HwInit	FFh	Maximum Current for 3.3V

Maximum Current Value Definition

Register Value	Current Value
0	Get Information via another method
1	4mA
2	8mA
3	12mA
----------	----------
255	1020mA

4.3.26 Slot Interrupt Status Register (offset 0FCh)

Table 35: ***Slot Interrupt Status Register***

Signal	Field	Attrib	After Reset	Description
Reserved	15:8	Rsvd	0	Reserved

Table 35: *Slot Interrupt Status Register*

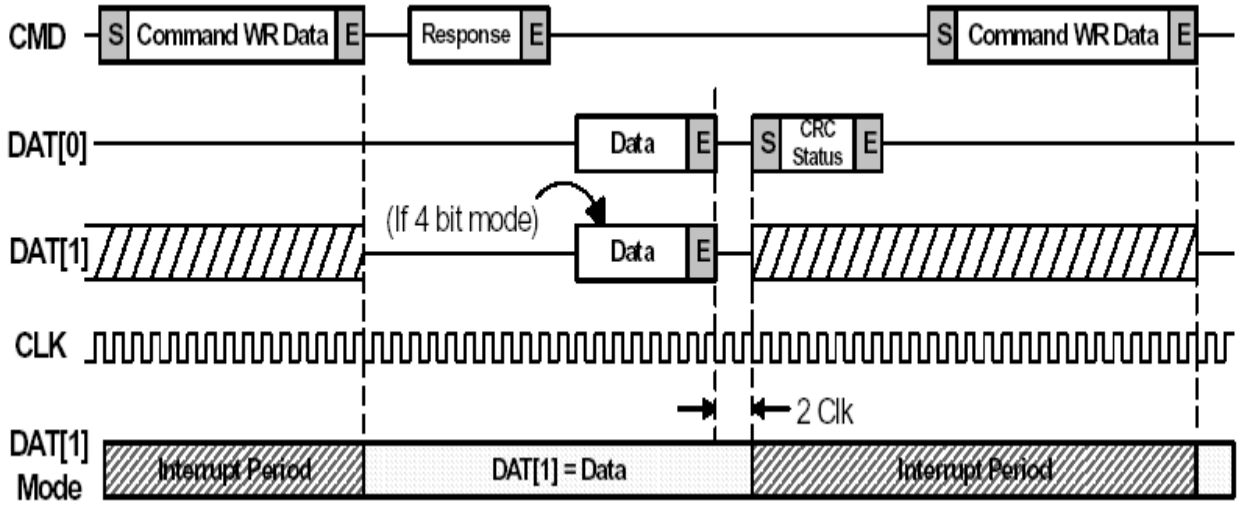
Signal	Field	Attrib	After Reset	Description
Interrupt Signal for Each Slot	7:0	ROC	0	These status bit indicate the logical OR of Interrupt signal and Wakeup signal for each slot. A maximum of 8 slots can be defined. If one interrupt signal is associated with multiple slots. the HD can know which interrupt is generated by reading these status bits. By a power on reset or by **Software Reset For All,** the Interrupt signal shall be deasserted and this status shall read 00h. Bit 00 - Slot 1 Bit 01 - Slot 2 Bit 02 - Slot 3 ----- ----- Bit 07 - Slot 8

4.3.27 Host Controller Version Register (offset 0FEh)

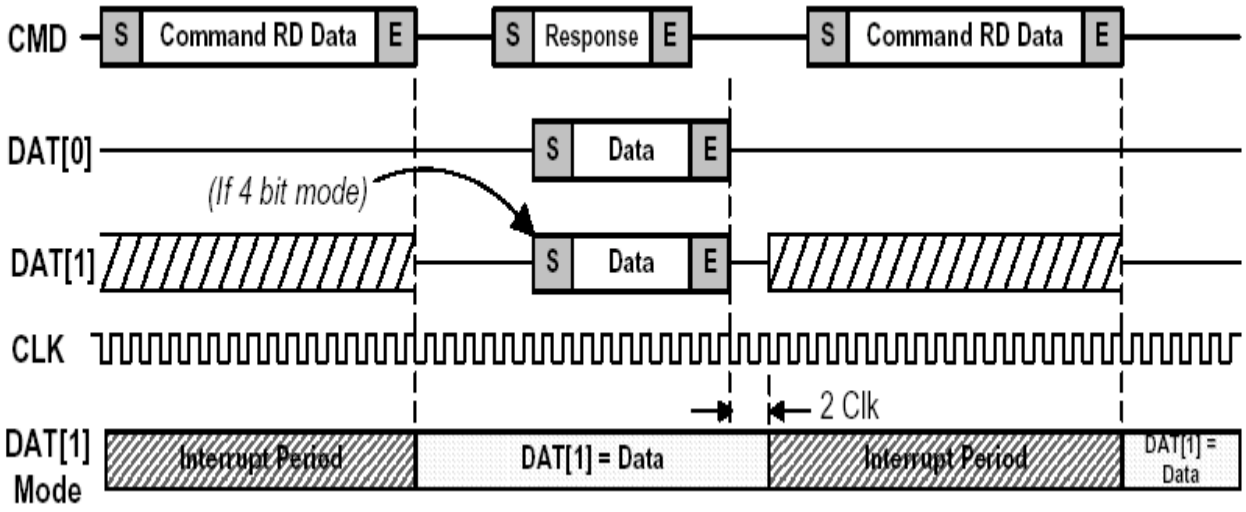
Table 36: *Host Controller Version Register*

Signal	Field	Attrib	After Reset	Description
Vendor Version Number	15:8	HwInit	0	This status is reserved for the vendor version number. The HD should not use this status.
Specification Version Number	7:0	HwInit	0	This Status indicates the Host Controller Spec Version. The Upper and Lower 4 bits indicate the version. 00 - SD Host Specification version 1.0 others - Reserved

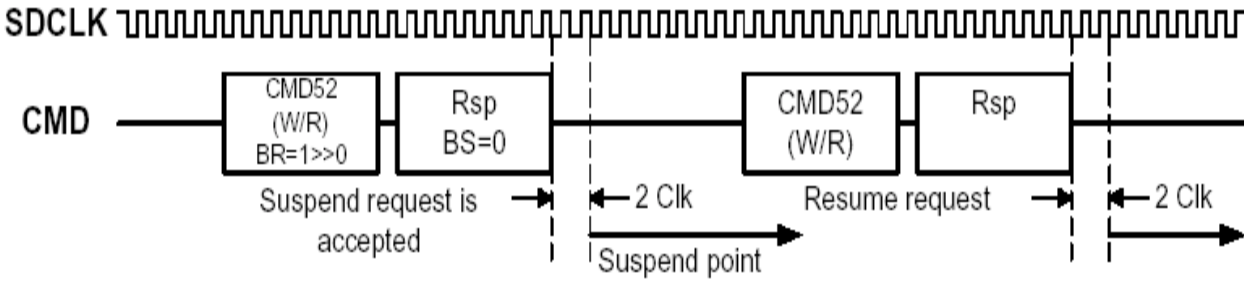
5.0 Timing Diagram:



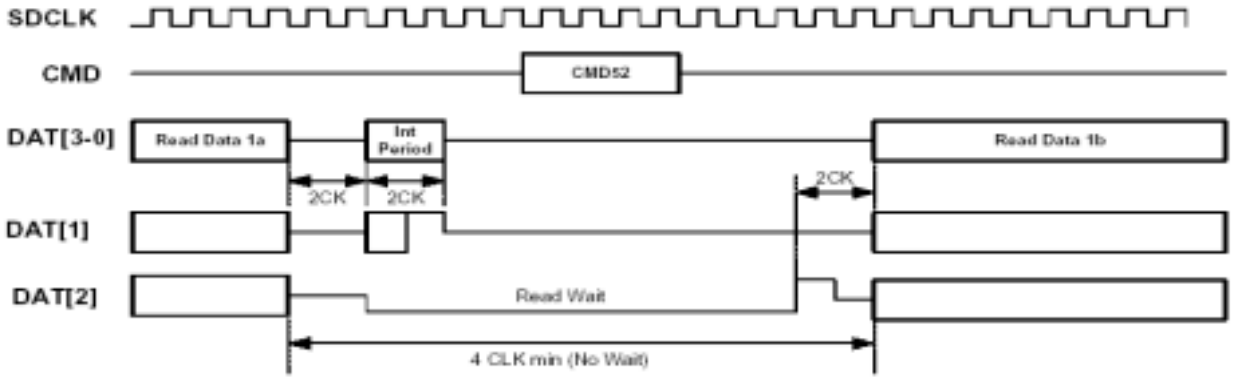
SDIO/SD - Write Interrupt Cycle Timing



SDIO/SD - Read Interrupt Cycle Timing



SDIO/SD - Suspend/Resume Timing



SDIO/SD - Read wait delay using DAT[2]

6.0 Data Transfer Protocol

Depending on whether DMA (optional) is used or not, there are two execution methods. The sequence not using DMA is shown in Fig 1 and the sequence using DMA is shown in Fig 2.

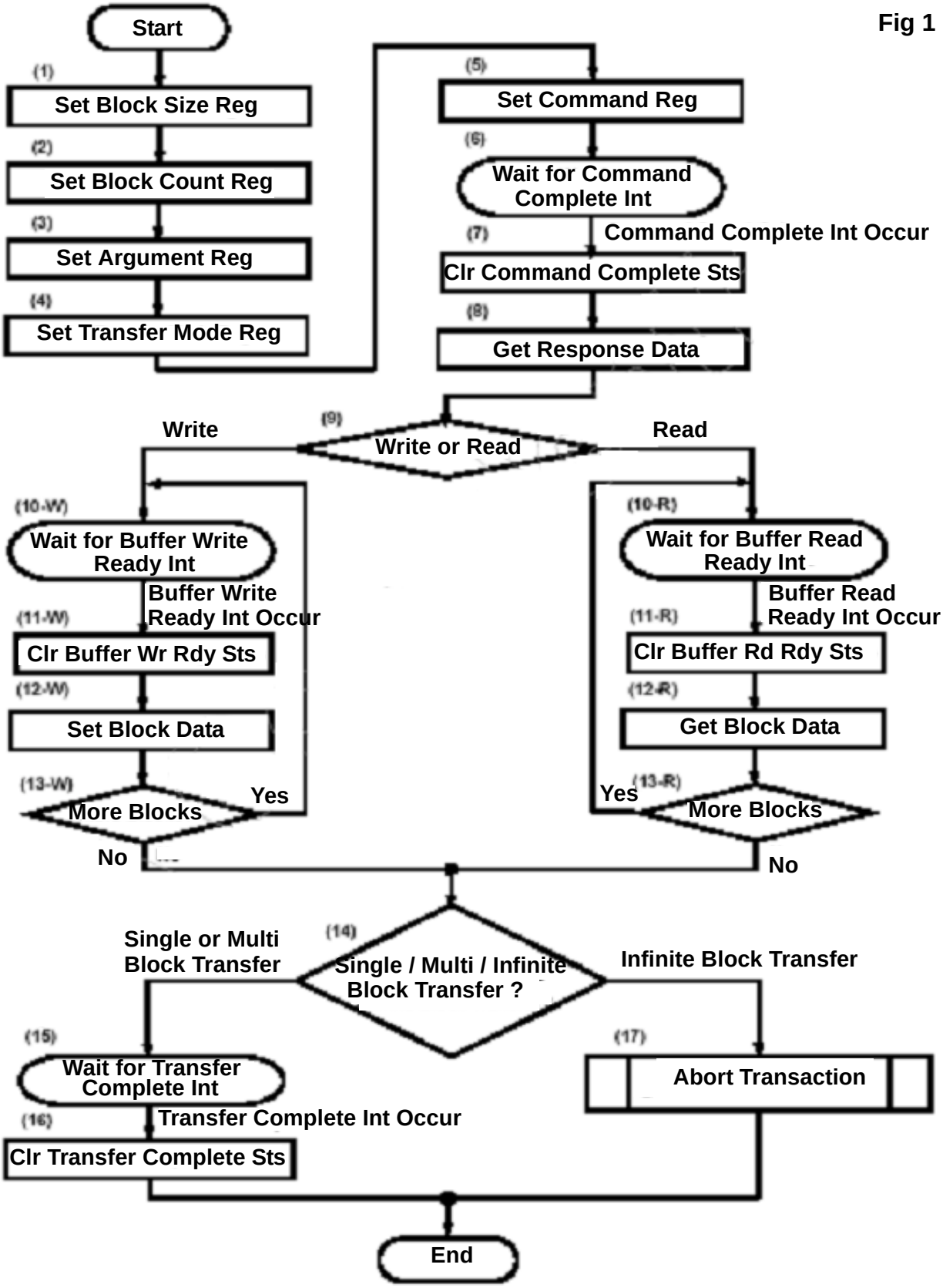
SD Transfers are basically classified into following three kinds according to how the number of blocks is specified:
1) Single Block Transfer:
The number of blocks is specified to the HC before the transfer. The number of blocks specified is always one.
2) Multiple Block Transfer:
The number of blocks is specified to the HC 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 HC 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.

6.1 Not Using DMA

The sequence for Not Using DMA is shown below
(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.
(3) Set the value corresponding to the issued command to Argument register.
(4) Set the value to **Multi / Single Block Select** and **Block Count Enable**. Set the value corresponding to the issued command to **Data Transfer Direction, Auto CMD12 Enable** and **DMA Enable.**

Data Transfer Using DAT Line Sequence (Not Using DMA)

Fig 1



(5) Set the value corresponding to the issued command to Command register.
Note: When writing the upper byte of *Command* register, SD command is issued.
(6) And then wait for the **Command Complete** Interrupt
(7) Write 1 to the **Command Complete** in the *Normal Interrupt Status* register for clearing this bit.
(8) Read *Response* register and get necessary information in accordance with the issued command.
(9) In the Case Where this sequence is for write to a card, go to step (10-W). In case of read from a card, go to step (10-R).

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

(10.1) **Non DMA Write Transfer**
On receiving the Buffer Write Ready interrupt the ARM processor will act as a master and start transferring the data via Buffer data port register (first half of the fifo). Transmitter start sending the data in SD bus when a block of data is ready in first half of the fifo. While transmitting the data in sd bus the buffer write ready interrupt is sent to the ARM Processor for the second block of data. The ARM processor will act as a master and start sending the second block of data via Buffer data port register to second half of the fifo. Buffer write ready interrupt will be asserted only when a fifo is empty to receive a block of data.

(11-W) Write 1 to the **Buffer Write Ready** in the *Normal Interrupt Status* register for clearing this bit.
(12-W) Write block data (in according to the number of bytes specified at the step (1)) to *Buffer Data Port* register.
(13-W) Repeat until all blocks are sent and then go to step (14).

(13.1) **Non DMA Read Transfer**
Buffer Read Ready interrupt is asserted whenever a block of data is ready in one of the fifo's. On receiving the Buffer Read Ready interrupt the ARM processor will act as a master and start reading the data via Buffer data port register (first half of the fifo). Receiver start reading the data from SD bus only when a fifo is empty to receive a block of data. When both the fifo's are full the host controller will stop the data coming from the card through read wait mechanism (if card supports read wait) or through clock stopping.

(10-R) And then wait for **Buffer Read Ready** Interrupt
(11-R) Write 1 to the **Buffer Read Ready** in the *Normal Interrupt Status* register for clearing this bit.
(12-R) Read block data (in according to the number of bytes specified at the step (1)) from the *Buffer Data Port* register.
(13-R) Repeat until all blocks are received and then go to step (14).

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

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

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

(17) Perform the sequence for Abort Transaction.

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

6.2 Using DMA

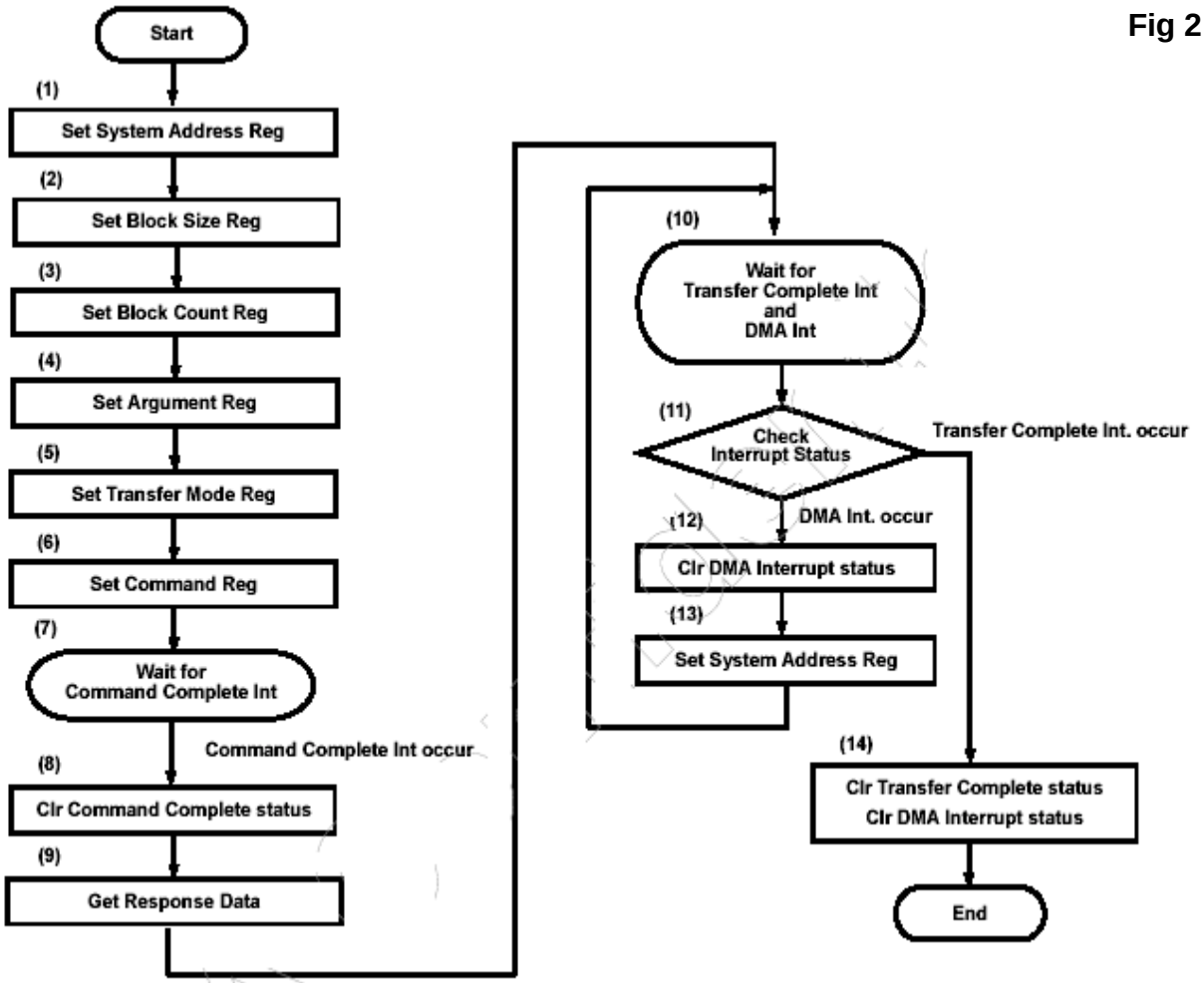
The bursts type like 8-beat incrementing burst or 4-beat incrementing burst or Single transfer is used to transfer or receive the data from the system memory mailny to avoid the hold of the AHB bus by the master for a longer time.

The sequence for Using DMA is shown below

(1) Set the system address for DMA in the *System Address* register.
(2) Set the value corresponding to the executed data byte length of one block in the Block Size register.
(3) Set the value corresponding to the executed data block count in the Block Count register.
(4) Set the value corresponding to the issued command to Argument register.
(5) Set the value to **Multi / Single Block Select** and **Block Count Enable**. Set the value corresponding to the issued command to **Data Transfer Direction, Auto CMD12 Enable** and **DMA Enable.**
(6) Set the value corresponding to the issued command to Command register.
Note: When writing the upper byte of *Command* register, SD command is issued.
(7) And then wait for the **Command Complete** Interrupt

Data Transfer Using DAT Line Sequence (Using DMA)

Fig 2



(8) Write 1 to the **Command Complete** in the *Normal Interrupt Status* register for clearing this bit.
(9) Read *Response* register and get necessary information in accordance with the issued command.

(9.1) ***DMA Read Transfer***

On receiving the response end bit from the card for the write command (data flowing from Host to Card) the SD Host controller will act as the master and request the AHB bus. After receiving the grant the host controller will start reading a block of data from the system memory and fills the first half of the fifo. Whenever a block of data is ready the transmitter will start sending the data in SD bus. While transmitting the data in SD bus the host controller request the bus to fill the second block in second half of the fifo. Ping

Pong fifo's are used to increase the through put. Similarly the host controller read a block of data from the system memory whenever a fifo is empty. This will continue till all the blocks are read from the System memory. Transfer complete Interrupt will be set only after transferring all the blocks of data to the card.

(9.2) ***DMA Write Transfer***
The block of data received from the Card (data flowing from Card to Host) is stored in first half of the fifo. Whenever a block of data is ready the SD Host controller will act as the master and request the AHB bus. After receiving the grant the host controller will start writing a block of data into the system memory from the first half of the fifo.
While transmitting the data into System memory the host controller will receive the second block of data and store in second half of the fifo. Similarly the host controller write a block of data into the system memory whenever data is ready. This will continue till all the blocks are transferred to the System memory. Transfer complete Interrupt will be set only after transferring all the blocks of data to the System memory.
Note: Host controller will receive a block of data from the card only when it has room to store a block of data in fifo. When both the fifo's are full the host controller will stop the data coming from the card through read wait mechanism (if card supports read wait) or through clock stopping.

(10) Wait for the **Transfer Complete** Interrupt and **DMA** Interrupt.
(11) If **Transfer Complete** is set 1, go to Step(4) 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 at same time. Step (5) and Step (6) can also be executed at same time.

For e.g if the host wants to transfer 4Kb of data to the Card. Assume the maximum block size is 256 bytes. Then the host driver will program the block size register as 256 and block count register with the value 16. The AHB Master and Transmitter residing inside the Arasan SD/SDIO Host Controller get the information (how much data to transfer) from these registers. Using the above informations, the AHB master will act as a master and initiate a data read transaction (to read a block of data - 256bytes from the system memory). The following types of burst used mainly to avoid hold of the AHB bus by the master for a longer time.
a) Single transfer
b) 4-beat incrementing burst
c) 8-beat incrementing burst
First block is received in first half of the fifo and the second block in second half of the fifo. Similarly the remaining blocks are recieved in alternate half's. Whenever a block of data is ready in fifo, the transmitter will start transmitting the block of data (256) in SD bus. After transmitting the entire block of data to the card, the transmitter will wait for a status response from the card. Transmitter will send the next block of data only, when it receive a good status response from the card for the previous block of data otherwise the transaction will be aborted and the host will do for a fresh transaction..

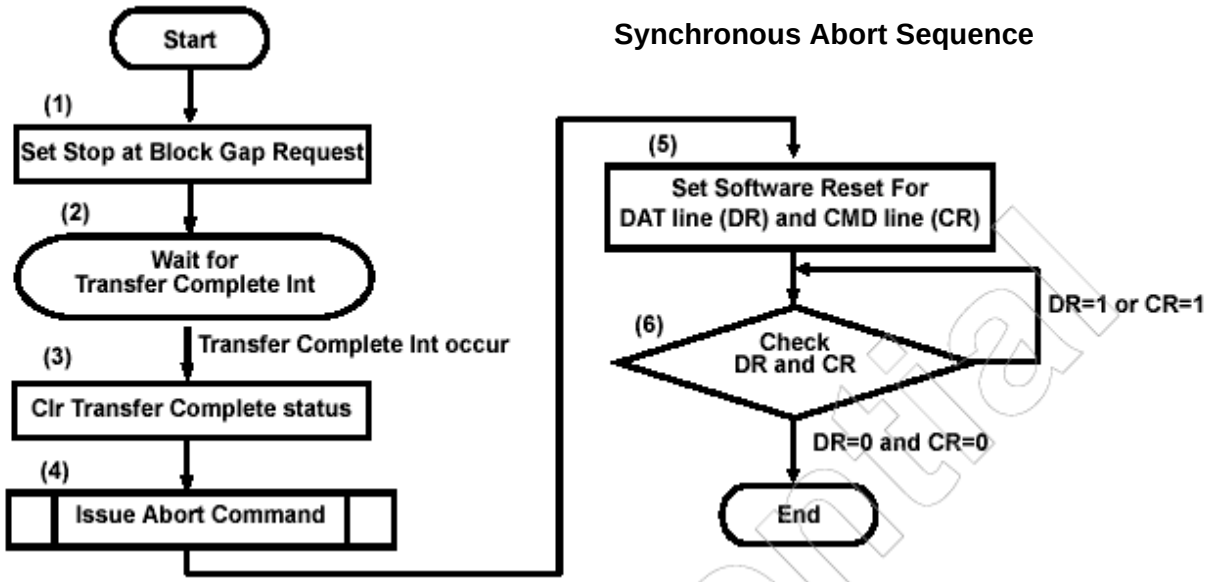
6.3 Abort Transaction
An Abort transaction is performed using CMD12 for a SD Memory Card and by using CMD52 for a SDIO Card. There are two cases where the HD needs to do an Abort Transaction.
(1) When the HD stops infinite block transfers.
(2) When HD stops transfers while a Multiple block transfer is exacting.

There are two ways to issue an Abort command. The first is an Asynchronous abort. The second is an Synchronous Abort. In an Asynchronous Abort sequence, the HD 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 HD

shall issue an Abort command after the data transfer stopped by using **Stop At Block Gap Request** in the *Block Gap Control* register.

6.3.1 Synchronous Abort

The essence for Synchronous Abort is shown below.
(1) Set the **Stop At Block Gap Request** in the *Block Gap Control* register to 1 to stop SD transactions.
(2) Wait for **Transfer Complete** Interrupt.
(3) Set the **Transfer Complete** to 1 in the *Normal Interrupt Status* register to clear this bit
(4) Issue Abort Command



(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.
(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(6).

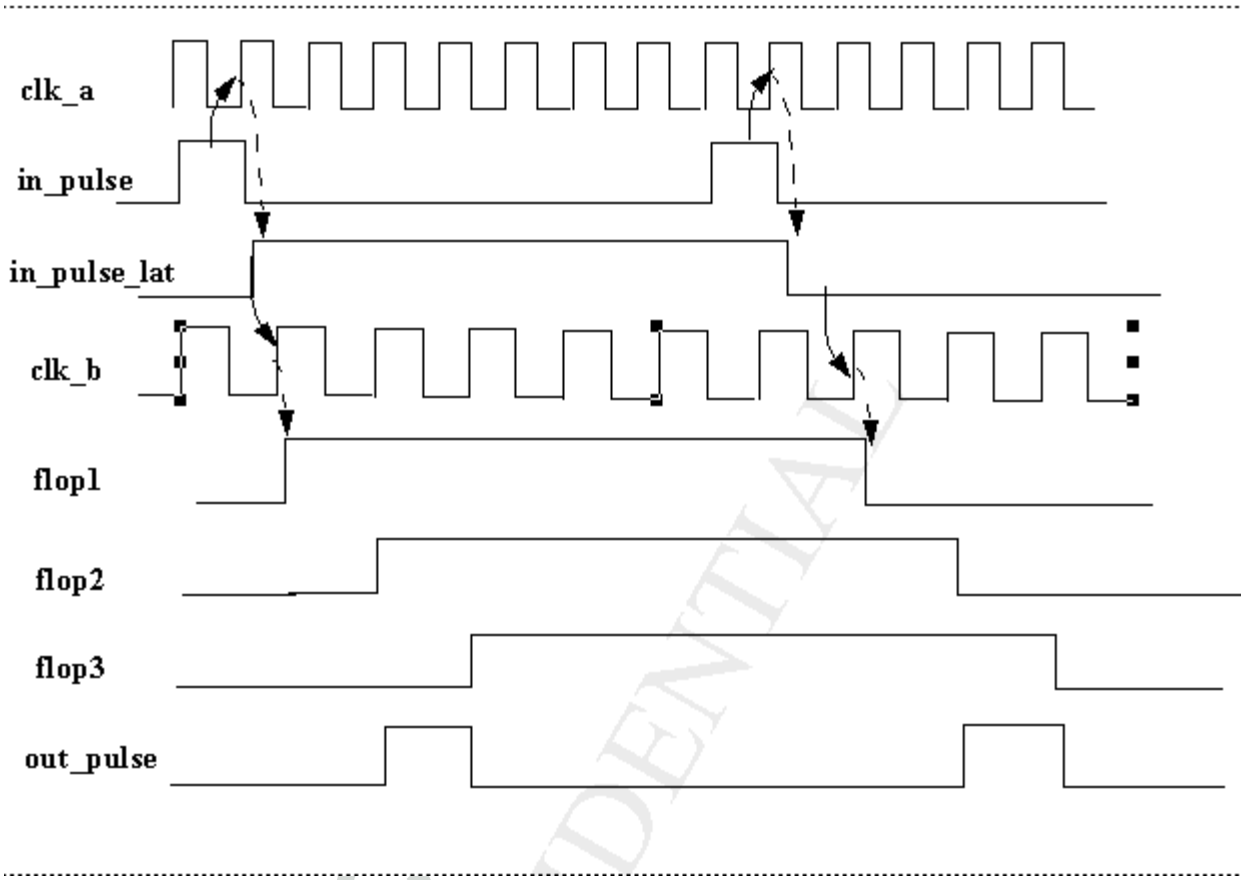
7.0 Synchronization

Data Path Synchronization

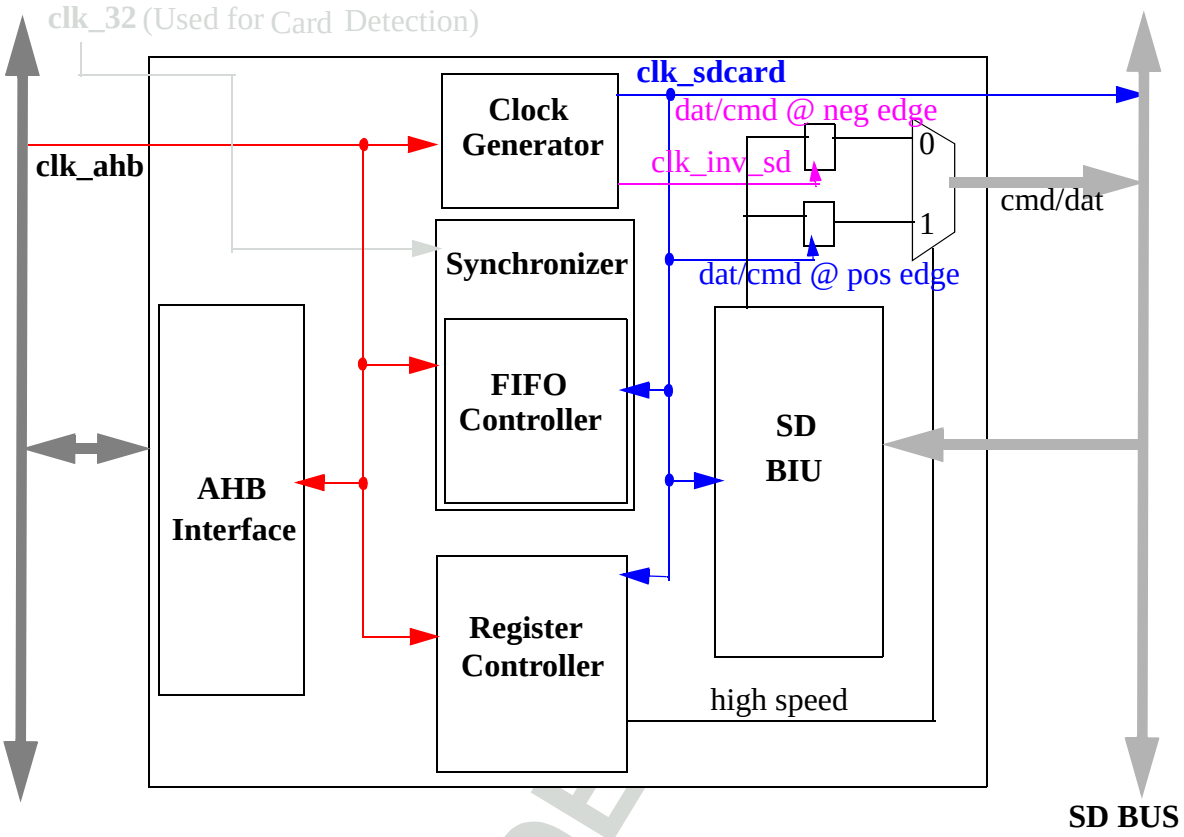
Dual port RAM is used to store data using one clock domain and to retrieve data using another clock domain for both read and write transaction.

Signals Flowing from Clock domain A to Clock Domain B

The input pulse (in_pulse) in clock domain A is latched at clock A and the latched signal (in_pulse_lat) being inverted whenever a input pulse (in_pulse) is detected. The latched signal is triple flopped in clock domain B. The output pulse in clock domain B is generated by XOR-ing the output of second and third stage synchronizers (flip flops).



7.1 Clocks

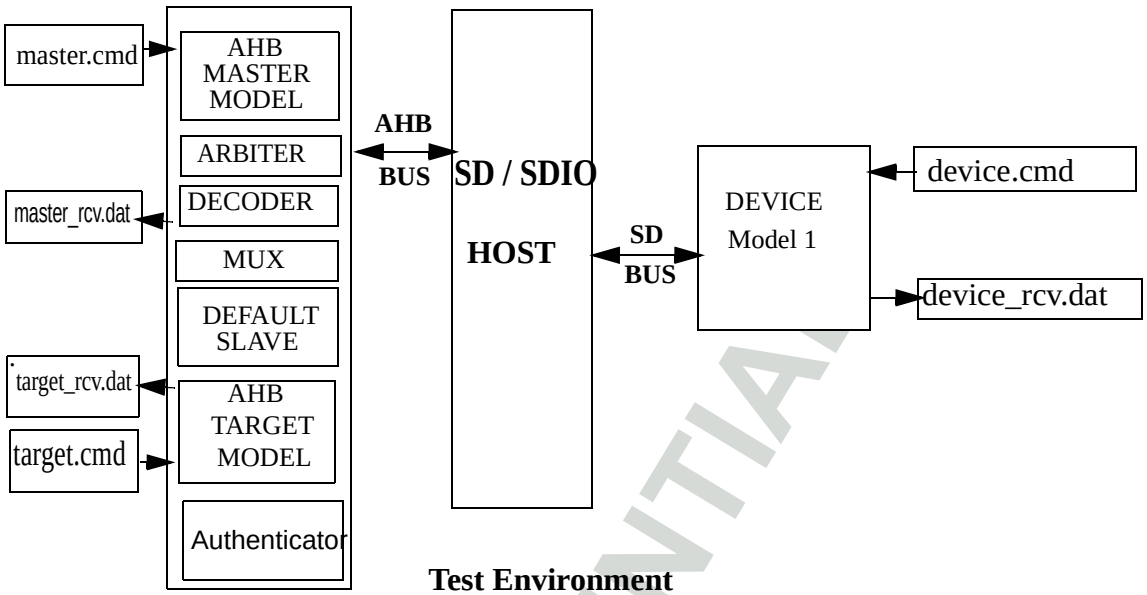


1. clk_ahb - AHB System Clock (upto 300Mhz)
2. clk_sdcard - SD Clock (upto 50Mhz) derived from AHB system clock (clk_ahb)
3. clk_32 - Optional 32Khz Clock used only for Card Detection mechanism

Note:
clk_inv_sd - Inverted SD Clock (Used to clock out the command and data at the falling edge of the sd clock - Rising edge of clk_inv_sd).
Our Host Controller supports both full speed card and hig speed card. For Full speed Card the Host Controller should clock out the cmd and data at the falling edge of the sd clock and for High speed Card the Host Controller should clock out the cmd and data at the rising edge of the sd clock.

8. Test Environment

This integrated test environment provides user writable command files for testing the sdio host controller core. The test environment reads the commands from the command files written by the user, initiates transaction with the core and reports the results of each transaction.



Test Environment

The organisation of the test environment is given below. The AHB master, target models assist in generating AHB transactions along with arbiter, decoder and mux. They emulate the function of a ARM processor.

8.1 MASTER MODEL

Master model emulates the function of a AHB master and initiates AHB transactions by performing tasks in master.v. It performs AHB transactions to access the host controller registers in the core and also data transactions during non dma mode of operation. The results of each transaction are printed on the transcript as simulation time advances.

8.2 ARBITER MODEL

The arbiter block takes care of the bus arbitration for the AHB bus when the core is the master. It takes the requests lines from the master core and gives grant according to the priority.

8.3 AHB Decoder

This decoder model is used to select a particular target among the various target. It reads the address in the ahb bus and checks with the address range of various targets and asserts the hsel.

8.4 AHB Default Target.

The default target model is used to drive the target signals when there are no more target accessing the ahb bus.

8.5 AHB Mux Model

This file acts as the central multiplexer.

8.6 TARGET MEMORY MODEL

The target memory model responds to all AHB transactions initiated by master core.

Functions of target model

1. Initialisation of ram with write data(write transaction).

2. Dumping data to output files(read transaction).
3. To check master core initiated transaction.

This block emulates the function of system memory. Separate address ranges are assigned for data needed for write transaction and data dumped during read transaction.
Target memory model is initialised with the data from host_sen.dat file.This Model responds to core initiated read and write AHB transactions.

DATA INITIALISATION FILE
This file contains the data which is transmitted to the devices for the WRITE transactions.Address allotment in memory

aa000000- aa000fff	**Memory to Device write data**
bb000000 - bb000fff	**Device to Memory read data**

Table 37: AHB model files

FILE	DESCRIPTION
ahbmaster_model.v	This file does the functionality of arm master.
ahbtarget_model.v	This file contains the tasks to emulate the functionality of ahb target.
decoder_model.v	This file decodes the address and asserts hsel to the addressed target.
mux_model.v	This model acts as the central multiplexer in ahb.
arbiter_model.v	This file gets the request signal from the core master and gives the grant as per the priority.
ram.v	This file is instantiated in ahbtarget_model.v and stores the initialized data value.
default_slave.v	This file is the default slave which asserts target signals when no other target is selected for the transaction.

.User Interface Files For The AHB Master And Target Models

Table 38: Interface Files For AHB Master And Target Model

FILE	DESCRIPTION
master.v	This file contains the tasks which is used to instruct the ahb master model to do transactions with the core.
master_rcv.dat	This file contains the read data received from the core during non dma transactions.
target.v	This file contains the tasks which instructs the target model to check the functionality of the master core.
target_rcv.dat	This file contains the data that is received from the card during non dma read transaction.

8.7 -SDIO/SD DEVICE MODEL :

SD/SDIO device model is having the functionality of sdio controllers memory and also mmc card for sd 1 or 4 bit mode based on the requirement of host controller. Device model reads the response command from command file name “devicex.cmd” where the response command is placed in such a manner that for each command from sdio host will have the corresponding response command.

The functionality turn between the sdio controllers memory and mmc based on the way response is placed in the cmd file.

When there is SDIO I/O read write command (DIRECT OR EXTENDED) for cccr and fbr then the data read or write is done via the “func0_reg.v” which got cccr and fbr register sets.

When there is SDIO I/O read write command (DIRECT OR EXTENDED) csa then the data read or write is done via the “csa.v” respectively, where csa.v will behave like ram where u can write and can read.

NOTE:

1.csa read can be done only when the csa support bit in the FBR reg 0*0100 will be set.

2.read can be done only after write in the csa block other wise while reading host will get the initialized data in each byte which is kept here as ‘FF’.

When the device model will receive IO_RW_EXTENTED read commands for fun1 the data bytes sent to the host are fetched from an ram.

When the device will receive IO_RW_EXTENDED write commands for fun1 then the data bytes received from host are written to the output file, namely, “device_rcvx.dat”.

For SD memory when device will get the command, which is related to sd memory data, then data will take from the internal mem when there is mem read command and will get dump in the “device_rcvx.dat” file when there will be mem write command, This model also monitors the data and cmd line and reports as error when violation occurs.

Authentication commands are also supported.After proper authentication session key is created and data

Table 39:

FILE	DESCRIPTION
devicex.cmd	This file contains the commands used for sending the response for the command initiate by the host controller
cis_data.v	This file consists of cis data that has to be loaded into the ram during initialization.
sdio_device_defines.v	This file contain the all the definitions that are used in the device model.
sdio_device_arrays.v	This file contains array initialization for the message to be displayed during the simulation.
task.v	This file contains all task used in the device model
func0_reg.v	This file consists of default function 0 data and load in to respective field during initialization.
device_rcvx.dat	This file contains the data that is expected during function, memory and mmc read transaction.

of secure read/write commands are encrypted and decrypted using session key.Data transaction command following the secure read/write command access the corresponding data in user area related to title key.To perform encryption/decryption, device model can make a function call for cipher algorithms in authenticator block.

9. Command List

9.1 Command list for the master model.

The following section explains the commands that the test environment supports for testing the sdio host controller core. Command list for the AHB master model is given below. These commands are used to perform configuration read, configuration write, memory read and memory write AHB transactions with the core. Configuration transactions are used to access the configuration registers in the core and memory transactions to access the Host controllers operational registers. This test environment supports both single and burst data phase configuration, memory transactions. Errors can be forced in the transactions and the robustness of the sdio host controller can be verified.

9.1.1 Commands for Accessing Host Controller registers

9.1.1.1 Write Commands

wr_sys_addr <32 bit value in hex>
This command initiates a AHB single phase transaction to write the system address register. The data value to be written to the register is given as 32 bit argument.

wr_blkcnt <16 bit value in hex>
This command initiates a AHB single phase transaction to write the block count register. The data value to be written to the register is given as 16 bit argument.

wr_ blksize <12 bit value in hex>
This command initiates a AHB single phase transaction to write the block size register. The data value to be written to the register is given as 16 bit argument.

wr_argument <32 bit value in hex>
This command initiates a AHB single phase transaction to write the argument register. The data value to be written to the register is given as 32 bit argument.

wr_tran_mode <16 bit value in hex>
This command initiates a AHB single phase transaction to write the transaction mode register. The data value to be written to the register is given as 16 bit argument.

wr_sd_cmd <16 bit value in hex>
This command initiates a AHB single phase transaction to write the sd command register. The data value to be written to the register is given as 16 bit argument.

wr_resp0 <16 bit value in hex>
This command initiates a AHB single phase transaction to write the response0 register. The data value to be written to the register is given as 16 bit argument.

wr_resp1 <16 bit value in hex>
This command initiates a AHB single phase transaction to write the response1 register. The data value to be written to the register is given as 16 bit argument.

wr_resp2 <16 bit value in hex>
This command initiates a AHB single phase transaction to write the response2 register. The data value to be written to the register is given as 16 bit argument.

wr_resp3 <16 bit value in hex>
This command initiates a AHB single phase transaction to write the response3 register. The data value to be written to the register is given as 16 bit argument.

wr_resp4 <16 bit value in hex>
This command initiates a AHB single phase transaction to write the response4 register. The data value to be written to the register is given as 16 bit argument.

wr_resp5 <16 bit value in hex>
This command initiates a AHB single phase transaction to write the response5 register. The data value to be written to the register is given as 16 bit argument.

wr_resp6 <16 bit value in hex>
This command initiates a AHB single phase transaction to write the response6 register. The data value to be written to the register is given as 16 bit argument.

wr_resp7 <16 bit value in hex>
This command initiates a AHB single phase transaction to write the response7 register. The data value to be written to the register is given as 16 bit argument.

wr_buff_data_port <32 bit value in hex>
This command initiates a AHB single phase transaction to write the buffer port register. The data value to be written to the register is given as 32 bit argument.

wr_pres_state <32 bit value in hex>

This command initiates a AHB single phase transaction to write the present state register. The data value to be written to the register is given as 32 bit argument.

wr_wakeup_ctrl <8 bit value in hex>
This command initiates a AHB single phase transaction to write the wakeup control register. The data value to be written to the register is given as 8 bit argument.

wr_blk_gap_ctrl <8 bit value in hex>
This command initiates a AHB single phase transaction to write the block gap control register. The data value to be written to the register is given as 8 bit argument.

wr_power_ctrl <8 bit value in hex>
This command initiates a AHB single phase transaction to write the power control register. The data value to be written to the register is given as 8 bit argument.

wr_host_ctrl <8 bit value in hex>
This command initiates a AHB single phase transaction to write the host control register. The data value to be written to the register is given as 8 bit argument.

wr_soft_rst <8 bit value in hex>
This command initiates a AHB single phase transaction to write the soft reset register. The data value to be written to the register is given as 8 bit argument.

wr_timeout_ctrl <8 bit value in hex>
This command initiates a AHB single phase transaction to write the timeout control register. The data value to be written to the register is given as 8 bit argument.

wr_clk_ctrl <16 bit value in hex>
This command initiates a AHB single phase transaction to write the clock control register. The data value to be written to the register is given as 16 bit argument.

wr_err_int_sts <16 bit value in hex>
This command initiates a AHB single phase transaction to write the error interrupt status register. The data value to be written to the register is given as 16 bit argument.

wr_nor_int_sts <16 bit value in hex>
This command initiates a AHB single phase transaction to write the normal interrupt status register. The data value to be written to the register is given as 16 bit argument.

wr_err_int_sts_en <16 bit value in hex>
This command initiates a AHB single phase transaction to write the error interrupt status enable register. The data value to be written to the register is given as 16 bit argument.

wr_nor_int_sts_en <16 bit value in hex>
This command initiates a AHB single phase transaction to write the normal interrupt status enable register. The data value to be written to the register is given as 16 bit argument.

wr_err_int_sig_en <16 bit value in hex>
This command initiates a AHB single phase transaction to write the error interrupt signal enable register. The data value to be written to the register is given as 16 bit argument.

wr_nor_int_sig_en <16 bit value in hex>

This command initiates a AHB single phase transaction to write the normal interrupt signal register. The data value to be written to the register is given as 16 bit argument.

wr_auto_cmd12_err_sts <16 bit value in hex>
This command initiates a AHB single phase transaction to write the auto command12 error status register. The data value to be written to the register is given as 16 bit argument.

wr_sd_hc_capabilities <32 bit value in hex>
This command initiates a AHB single phase transaction to write the host controller capabilities register. The data value to be written to the register is given as 32 bit argument.

wr_max_curr_cap <32 bit value in hex>
This command initiates a AHB single phase transaction to write the maximum current capabilities register. The data value to be written to the register is given as 32 bit argument.

wr_sd_hc_ver <16 bit value in hex>
This command initiates a AHB single phase transaction to write the host controller version register. The data value to be written to the register is given as 16 bit argument.

wr_slot_int_sts <16 bit value in hex>
This command initiates a AHB single phase transaction to write the slot interrupt status register. The data value to be written to the register is given as 16 bit argument.

9.1.1.2 Read Commands

rd_sys_addr <32 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the system address register. 32 bit argument is the data which is expected as the result of read.

rd_blkcnt <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the block count register. 16 bit argument is the data which is expected as the result of read.

rd_ blksize <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the block size register. 16 bit argument is the data which is expected as the result of read.

rd_argument <32 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the argument register. 32 bit argument is the data which is expected as the result of read.

rd_tran_mode <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the transaction mode register. 16 bit argument is the data which is expected as the result of read.

rd_sd_cmd <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the sd command register. 16 bit argument is the data which is expected as the result of read.

rd_resp0 <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the response0 register. 16 bit argument is the data which is expected as the result of read.

rd_resp1 <16 bit exp_value in hex>

This command initiates a AHB single phase transaction to read the response1 register. 16 bit argument is the data which is expected as the result of read.

rd_resp2 <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the response2 register. 16 bit argument is the data which is expected as the result of read.

rd_resp3 <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the response3 register. 16 bit argument is the data which is expected as the result of read.

rd_resp4 <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the response4 register. 16 bit argument is the data which is expected as the result of read.

rd_resp5 <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the response5 register. 16 bit argument is the data which is expected as the result of read.

rd_resp6 <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the response6 register. 16 bit argument is the data which is expected as the result of read.

rd_resp7 <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the response7 register. 16 bit argument is the data which is expected as the result of read.

rd_buff_data_port <32 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the buffer port register. 32 bit argument is the data which is expected as the result of read.

rd_pres_state <32 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the present state register. 32 bit argument is the data which is expected as the result of read.

rd_wakeup_ctrl <8 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the wakeup control register. 8 bit argument is the data which is expected as the result of read.

rd_blk_gap_ctrl <8 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the block gap control register. 8 bit argument is the data which is expected as the result of read.

rd_power_ctrl <8 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the power control register. 8 bit argument is the data which is expected as the result of read.

rd_host_ctrl <8 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the host control register. 8 bit argument is the data which is expected as the result of read.

rd_soft_rst <8 bit exp_value in hex>

This command initiates a AHB single phase transaction to read the soft reset register. 8 bit argument is the data which is expected as the result of read.

rd_timeout_ctrl <8 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the timeout control register. 8 bit argument is the data which is expected as the result of read.

rd_clk_ctrl <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the clock control register. 16 bit argument is the data which is expected as the result of read.

rd_err_int_sts <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the error interrupt status register. 16 bit argument is the data which is expected as the result of read.

rd_nor_int_sts <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the normal interrupt status register. 16 bit argument is the data which is expected as the result of read.

rd_err_int_sts_en <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the error interrupt status enable register. 16 bit argument is the data which is expected as the result of read.

rd_nor_int_sts_en <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the normal interrupt status enable register. 16 bit argument is the data which is expected as the result of read.

rd_err_int_sig_en <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the error interrupt signal enable register. 16 bit argument is the data which is expected as the result of read.

rd_nor_int_sig_en <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the normal interrupt signal register. 16 bit argument is the data which is expected as the result of read.

rd_auto_cmd12_err_sts <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the auto command12 error status register. 16 bit argument is the data which is expected as the result of read.

rd_sd_hc_capabilities <32 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the host controller capabilities register. 32 bit argument is the data which is expected as the result of read.

rd_max_curr_cap <32 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the maximum current capabilities register. 32 bit argument is the data which is expected as the result of read.

rd_sd_hc_ver <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the host controller version register. 16 bit argument is the data which is expected as the result of read.

rd_slot_int_sts <16 bit exp_value in hex>
This command initiates a AHB single phase transaction to read the slot interrupt status register. 16 bit argument is the data which is expected as the result of read.

9.1.2 Commands for Transaction Control

Table 40: Registers accessed during cmd53* executions

No	Registers	Offset	Dma bit
1.	System address register	00	1
2.	Block size and Block count register	04 & 06	don't care
3.	Argument register	08	don't care
4.	Transfer mode and command register	0C & 0E	don't care

cmd53_wr_byte_fn1 <12 bit byte_cnt in hex> <1 bit dma>

This command when executed writes the set of registers given in **table 9** based on the dma bit. This command will initiate a byte mode write transaction for function1 in sd bus.

byte count defines the 12 bit transfer block size value of the block size register which is the number of bytes to be transmitted to the card.

dma is the value of bit 0 in Transfer mode register which when '1' enables dma mode of transaction.

The other necessary bits belonging to the registers mentioned in **table 9** required for this transaction will be set internally by the model.

cmd53_wr_blk_fn1 <12 bit block_size><16 bit blk_cnt> <1 bit dma>

This command when executed writes the set of registers given in **table 9** based on the dma bit.

This command will initiate a block mode write transaction for function1 in sd bus.

byte count defines the 12 bit transfer block size value of the block size register which is the number of bytes to be transmitted to the card.

blk_cnt is the block count register value which is the number of blocks to be transmitted.

dma is the value of bit 0 in Transfer mode register which when '1' enables dma mode of transaction.

The other necessary bits belonging to the registers mentioned in **table1** required for this transaction will be set internally by the model.

cmd53_rd_byte_fn1 <12 bit byte_cnt in hex> <1 bit dma>

This command when executed writes the set of registers given in table 9 based on the dma bit.

This command will initiate a byte mode read transaction for function1 in sd bus.

byte count defines the 12 bit transfer block size value of the block size register which is the number of bytes to be received from the card.

dma is the value of bit 0 in Transfer mode register which when '1' enables dma mode of transaction.

The other necessary bits belonging to the registers mentioned in **table 9** required for this transaction will be set internally by the model.

cmd53_rd_blk_fn1 <12 bit block_size in hex> <blk_cnt in hex> <1 bit dma >

This command when executed writes the set of registers given in **table 9** based on the dma bit.

This command will initiate a block mode read transaction for function1 in sd bus.

byte count defines the 12 bit transfer block size value of the block size register which is the number of bytes to be received from the card.

blk_cnt is the block count register value which is the number of blocks to be received.

dma is the value of bit 0 in Transfer mode register which when '1' enables dma mode of transaction. The other necessary bits belonging to the registers mentioned in **table 9** required for this transaction will be set internally by the model.

cmd53_wr_byte_fn0 <17 bit reg_addr in hex><12 bit byte_cnt in hex> <1 bit dma><1bit opcode>

This command when executed writes the set of registers given in **table 9** based on the dma bit. This command will initiate a byte mode write transaction for function0 in sd bus.
reg_addr is the address of the register from which the transaction is going to be started.
byte count defines the 12 bit transfer block size value of the block size register which is the number of bytes to be transmitted to the card.
dma is the value of bit 0 in Transfer mode register which when '1' enables dma mode of transaction.
opcode when '1' initiates cmd53 with incrementing address and when '0' initiates fixed addressing.
The other necessary bits belonging to the registers mentioned in **table 9** required for this transaction will be set internally by the model.

cmd53_wr_blk_fn0 <17 bit reg_addr in hex><12 bit block_size in hex> <16 bit blk_cnt in hex> <1 bit dma > <1 bit opcode>

This command when executed writes the set of registers given in **table 9** based on the dma bit.
This command will initiate a block mode write transaction for function0 in sd bus.
reg_addr is the address of the register from which the transaction is going to be started.
byte count defines the 12 bit transfer block size value of the block size register which is the number of bytes to be transmitted to the card.
blk_cnt is the block count register value which is the number of blocks to be transmitted.
dma is the value of bit 0 in Transfer mode register which when '1' enables dma mode of transaction.
opcode when '1' initiates cmd53 with incrementing address and when '0' initiates fixed addressing.
The other necessary bits belonging to the registers mentioned in **table 9** required for this transaction will be set internally by the model.

cmd53_rd_byte_fn0 <17 bit reg_addr in hex><12 bit byte_cnt in hex> <1 bit dma><1 bit opcode>

This command when executed writes the set of registers given in **table 9** based on the dma bit.
This command will initiate a byte mode read transaction for function0 in sd bus.
reg_addr is the address of the register from which the transaction is going to be started.
byte count defines the 12 bit transfer block size value of the block size register which is the number of bytes to be received from the card.
dma is the value of bit 0 in Transfer mode register which when '1' enables dma mode of transaction.
opcode when '1' initiates cmd53 with incrementing address and when '0' initiates fixed addressing.

cmd53_rd_blk_fn0 <reg_addr><block_size><blk_cnt> <dma><opcode>

This command when executed writes the set of registers given in **table 9** based on the dma bit.
This command will initiate a block mode read transaction for function0 in sd bus.
reg_addr is the address of the register from which the transaction is going to be started.
byte count defines the 12 bit transfer block size value of the block size register which is the number of bytes to be transmitted to the card.
blk_cnt is the block count register value which is the number of blocks to be transmitted.
dma is the value of bit 0 in Transfer mode register which when '1' enables dma mode of transaction.
opcode when '1' initiates cmd53 with incrementing address and when '0' initiates fixed addressing.
The other necessary bits belonging to the registers mentioned in **table 9** required for this transaction will be set internally by the model.

cmd52_wr_fn1 <17 bit reg_addr in hex> <8 bit wr_data in hex> <8 bit exp_data in hex> <1 bit raw>

Table 41: For Commands with No data Transfer

No	Registers	Offset
1.	Argument register	08
2.	Command register	0E

This command when executed writes registers of **table 10**. It initiates cmd52 write transaction for function1 in sd bus.
reg_addr is the address of the card register to which this transaction is intended.
wr_data is the value to be written to the register.
exp_data is the value expected to be returned in the response.
raw if '1' sets read after write bit.

cmd52_rd_fn1 <17 bit reg_addr in hex> <8 bit exp_data in hex>

This command when executed writes registers of table 10. It initiates cmd52 read transaction for function1 in sd bus.
reg_addr is the address of the card register to which this transaction is intended.
exp_data is the value expected to be returned in the response.

cmd52_wr_fn0 <reg_addr> <wr_data> <exp_data> <raw>

This command when executed writes registers of table 10. It initiates cmd52 write transaction for function0 in sd bus.
reg_addr is the address of the card register to which this transaction is intended.
wr_data is the value to be written to the register.
exp_data is the value expected to be returned in the response.
raw if '1' sets read after write bit.

cmd52_rd_fn0 <reg_addr> <exp_data>

This command when executed writes registers of table 10. It initiates cmd52 read transaction for function0 in sd bus.
reg_addr is the address of the card register to which this transaction is intended.
exp_data is the value expected to be returned in the response.

validate_ocr <32 bit ocr in hex><mem>
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd5 in sd bus. **ocr** is the operating voltage range of the host controller**. mem** should be '1' for memory card and 0 for i/o card.

mem_mul_wr <12 bit min_block_size in hex><32 bit max_block_size in hex> <32 bit blk_cnt in hex> <32 bit data_addr in hexdma > <1 bit dma> <1 bit auto_cmd12>

This command when executed writes the set of registers given in **table 9** based on the dma bit.
This command will initiate a block mode write transaction with effect from minimum block size to maximum block size in sd bus.
min_block_size is the size of the block in the transaction is minimum,
max_block_size is the size of the block in the transaction is maximum.

reg_addr is the address of the register from which the transaction is going to be started.
blk_cnt is the block count register value which is the number of blocks to be transmitted.
dma is the value of bit 0 in Transfer mode register which when '1' enables dma mode of transaction.
auto_cmd12 is the value of bit in transfer mode register which when '1' enables auto command execution at the end of the transaction.

mem_mul_rd <12 bit min_block_size in hex><32 bit max_block_size in hex> <32 bit blk_cnt in hex> <32 bit data_addr in hexdma > <1 bit dma> <1 bit auto_cmd12>

This command when executed writes the set of registers given in **table 9** based on the dma bit.
This command will initiate a block mode read transaction with effect from minimum block size to maximum block size in sd bus.
min_block_size is the size of the block in the transaction is minimum,
max_block_size is the size of the block in the transaction is maximum.
reg_addr is the address of the register from which the transaction is going to be started.
blk_cnt is the block count register value which is the number of blocks to be transmitted.
dma is the value of bit 0 in Transfer mode register which when '1' enables dma mode of transaction.
auto_cmd12 is the value of bit in transfer mode register which when '1' enables auto command execution at the end of the transaction.

cmd53_fn1_wr_rand <16 bit min_block_size in hex><16 bit max_block_size in hex> <16 bit blk_cnt in hex> <32 bit data_addr in hexdma > <1 bit dma>
This command when executed writes the set of registers given in **table 9** based on the dma bit.
This command will initiate a block mode io write transaction for function1 with effect from minimum block size to maximum block size in sd bus for random testing.
min_block_size is the size of the block in the transaction is minimum,
max_block_size is the size of the block in the transaction is maximum.
reg_addr is the address of the register from which the transaction is going to be started.
blk_cnt is the block count register value which is the number of blocks to be transmitted.
dma is the value of bit 0 in Transfer mode register which when '1' enables dma mode of transaction.

cmd53_fn1_rd_rand <16 bit min_block_size in hex><16 bit max_block_size in hex> <16 bit blk_cnt in hex> <32 bit data_addr in hexdma > <1 bit dma>
This command when executed writes the set of registers given in **table 9** based on the dma bit.
This command will initiate a block mode io read transaction for function1 with effect from minimum block size to maximum block size in sd bus for random testing.
min_block_size is the size of the block in the transaction is minimum,
max_block_size is the size of the block in the transaction is maximum.
reg_addr is the address of the register from which the transaction is going to be started.
blk_cnt is the block count register value which is the number of blocks to be transmitted.
dma is the value of bit 0 in Transfer mode register which when '1' enables dma mode of transaction.

cmd0
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd0 for i/o in sd bus.

cmd2
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd2 in sd bus.

cmd3
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd3 in sd bus.

cmd4
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd4 in sd bus.

cmd7 <1 bit sel_bit>
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd7 in sd bus.
sel_bit if '1' will instruct the model to issue cmd7 with the rca value received from the card to select the card and '0' will issue cmd7 with rca as 0x0000 to deselect the card.

cmd9
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd9 in sd bus.

cmd10
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd10 in sd bus.

cmd12
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd12 in sd bus.

cmd13
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd13 in sd bus.

cmd14
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd14 in sd bus.

cmd15
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd15 in sd bus.

cmd16 <32 bit blk_len in hex>
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd16 in sd bus.
blk_len is used to set the block length in the card.

cmd17 <12 bit blk_size in hex><32 bit data_addr in hex> <dma>
This command when executed writes the set of registers defined in table 9 which will instruct the host controller to issue cmd17 in sd bus.
blk_size is the transfer block size value of the block size register**.**
data_addr is the starting address of the read transaction.
dma is the value of bit 0 in Transfer mode register which when '1' enables dma mode of transaction.
The other bits required for this transaction will be set internally by the model.

cmd18<12 bit blk_size in hex><16 bit blk_cnt in hex><32 bit data_addr in hex> <1 bit dma><1 bit auto_cmd12_en>
This command when executed writes the set of registers defined in table 9 which will instruct the host controller to issue cmd18 in sd bus.
blk_size is the transfer block size value of the block size register**.**
blk_cnt is the block count value of the block count register.

data_addr is the starting address of the read transaction.
dma is the value of bit 0 in Transfer mode register which when '1' enables dma mode of transaction.
auto_cmd12_en if '1' will set the auto cmd12 enable bit in transfer mode register.
The other bits required for this transaction will be set internally by the model.

cmd24 <blk_size> <32 bit data_addr in hex><dma>
This command when executed writes the set of registers defined in table 9 which will instruct the host controller to issue cmd24 in sd bus.
blk_size is the transfer block size value of the block size register**.**
data_addr is the starting address of the write transaction.
dma is the value of bit 0 in Transfer mode register which when '1' enables dma mode of transaction.
The other bits required for this transaction will be set internally by the model.

cmd25 <blk_size><blk_cnt> <32 bit data_addr in hex><dma> <auto_cmd12_en>
This command when executed writes the set of registers defined in table 9 which will instruct the host controller to issue cmd25 in sd bus.
blk_size is the transfer block size value of the block size register**.**
blk_cnt is the block count value of the block count register.
data_addr is the starting address of the read transaction.
dma is the value of bit 0 in Transfer mode register which when '1' enables dma mode of transaction.
auto_cmd12_en if '1' will set the auto cmd12 enable bit in transfer mode register.
The other bits required for this transaction will be set internally by the model.

cmd27
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd27 in sd bus.

cmd28 <32 bit data_addr in hex>
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd28 in sd bus.
data_addr is the address of the write protect group.

cmd29 <32 bit data_addr in hex>
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd32 in sd bus.
data_addr is the address of the write protect group.

cmd30 <32 bit write_prot_data_addr in hex>
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd30 in sd bus.
data_addr is the write protect data address.

cmd32 <32 bit data_addr in hex>
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd32 in sd bus.
data_addr is the address of the first write block to be erased**.**

cmd33 <32 bit data_addr in hex>
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd33 in sd bus.
data_addr is the address of the last write block to be erased**.**

cmd38

This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd38 in sd bus.

cmd41
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd41 in sd bus. Since cmd41 is a reserved command in sd memory card, cmd55 should be issued before this command otherwise the driver model reports error.

cmd42
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd42 in sd bus.

cmd55
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd55 in sd bus.

cmd56_rd<1 bit dma>
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd56 for read in sd bus.
dma is the value of bit 0 in Transfer mode register which when '1' enables dma mode of transaction.

cmd56_wr<1 bit dma>
This command when executed writes the set of registers defined in table 102 which will instruct the host controller to issue cmd56 for write in sd bus.
dma is the value of bit 0 in Transfer mode register which when '1' enables dma mode of transaction.

SD Memory Application specific commands

Acmd6 <1 bit bus_width>
This command when executed writes the set of registers defined intable 10 which will instruct the host controller to issue Acmd6 in sd bus.
bus_width if 0 indicates 1 bit wide and 1 indicates 4 bit wide.

Acmd13
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue Acmd13 in sd bus.

Acmd22
This command when executed writes the set of registers defined in table 10which will instruct the host controller to issue Acmd22 in sd bus.

Acmd23 < 23 bit no_of_blks in hex>
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue Acmd23 in sd bus.
no_of_blks is the number of block to be pre-erased before writing.

Acmd41 < ocr>
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue Acmd41 in sd bus.
ocr is the voltage range supported by the host controller**.**

Acmd42 <1 bit set_cd>

This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue cmd42 in sd bus.
set_cd if '1' will connect the pull up register on cd/dat3 line and '0' will disconnect.

Acmd51
This command when executed writes the set of registers defined in table 10 which will instruct the host controller to issue Acmd51 in sd bus.

Security commands:
Acmd43
This command when executed retrieves the MKB from card.MKB's are sent as data block.

Acmd44
This command gets the media-ID from card.

Acmd45<arguments>
arguments field is defined as:
For secure read:
<31:24>unit_count,
<23>1'b0,
<22:0>unit_address.

For secure write:
<31:24>unit_count,
<23>mode,
<22:0>unit_address.

For secure erase:
<31:24>unit_count,
<23:0>unit_address.

For change of secure area:
<23:0>unit_address,
<31:24>stuff_bits.

For secure write MKB:
<31:24>unit_count>,
<23:16>MKB_ID,
<15:0>reserved.

Writes the challenge1(in an uncrypted form--in order to calculate session key to derive title key) to card.

Acmd46
This command retrives challenge2 from the card.

Acmd47
This command writes RES2 to the card.

Acmd48
This command reads RES1 from card.

Acmd18
This command performs a secure read to protected area.Data field is decrypted by Host controller using title key.

Acmd25
This command performs a secure write to protected area.Data field to be written to Host controller should not be encrypted.

Acmd38
This command performs secure erase.
Acmd49
This command performs change of secure area.
Acmd26
Writes MKB values.

9.1.3 Functionality Checking Commands

1). suspend
This command executes the suspend sequence explained in the spec. Model internally handles all the actions essential to suspend a io card.

2). resume
This command executes the resume sequence explained in the spec. Model internally handles all the actions essential to resume the suspended io card.

3). abort
This command executes the abort sequence explained in the spec.

4). clock_stop
This command is used to stop the sd clock.

5.) clock_supply<divisor_value>
This command is used to select the clock frequency for the sd bus.
divisor_value is used to divide the base clock frequency to get the desired sd clock frequency as given below

00 -- same as base clock
01 -- divide base clock by 2
02 -- divide base clock by 4
04 -- divide base clock by 8
08 -- divide base clock by 16
10 -- divide base clock by 32
20 -- divide base clock by 64
40 -- divide base clock by 128
80 -- divide base clock by 256

6.) clock_change<divisor_value>
This command is used to change the clock frequency for the sd bus.
divisor_value is same as explained for the previous command.

Table 42: card type

Card type	Descrption
00	IO
01	MEM
10	Combo
11	Multi-media

7.) bus_power<card_type>

This command is used to set the bus power for the sd bus based on the type of the card given through the argument.

8.) set_bus_width <1 bit data_transfer_width>

This command is used to change the buswidth. **data_transfer_width** if '1' sets the buswidth as 4 bit and '0' as 1 bit.

9.) mem_card_init

This command executes the initialization sequence for MEMORY card.

10.) io_card_init

This command executes the initialization sequence for SDIO card.

11.) combo_card_init

This command executes the initialization sequence for COMBO card.

12.) mmc_card_init

This command executes the initialization sequence for MMC card.

13.) delay_read<block_no><period>

This command will introduce delay in the immediately following read command in the block mentioned as the first argument. The second argument is the number of clocks that the transaction should be delayed.

9.1.4 Blocking commands

1.) wait_for_intrpt

This command will wait for any interrupt which is asserted either due to error conditions or normal events.

2.) clock_n.<16 bit delay in hex>

This command delays the execution of the next command for the number of clock cycles specified in delay argument.

9.2 Command lists for Target model

9.2.1 Error Forcing Commands

1.) force_retry<trans_no><d_phase_no>

e.g. force_retry 5 2

This command is used by the target model to force retry response in the transaction mentioned in trans_no.

2.) force_error<trans_no><d_phase_no>

e.g. force_retry 10 1

This command is used by the target model to force error response in the transaction mentioned in trans_no.

3.) deassert_hready<trans_no><d_phase_no><delay>

e.g deassert_hready 3 2 a

This command is used to deassert the hready signal in the above mentioned transaction for specified no of clocks that is given in delay.

9.3 COMMANDS READ BY DEVICE MODEL

NOTE:

1.The field of <No of clock> is define as 4 bit in hex(xxxx).

2,The field of <Ncr > is define as 4 bit in hex(xxxx)

3.The field of <Nbusy> is define as 4 bit in hex(xxxx)

4.The field of <Nbusy_start> is define as 4 bit in hex

9.3.1 GENERAL COMMAND

1. connect <type>

If type is 0 then device is disconnected
If type is 1 then device is connected

2. cmd7_r1

To send the response R1 for cmd7 (select and deselect cmd).

3.cmd7_r1_err<Ncr>

To send the response R1 for cmd7 (select and deselect cmd) after the Ncr clock of cmd received.

4. cmd7_r1b<Nbusy_start><Nbusy>

To send the response R1 for cmd7(select and deselect cmd) with busy period for Nbusy clock and the busy start after the Nbusy_start clock of response.

5. cmd7_r1b_err<Nbusy_start><Nbusy>

To send the response R1 for cmd7 (select and deselect cmd) with busy end bit error and the busy period if for Nbusy clock the busy will start after the Nbusy_start clock of response.

6. cmd12_r1

To send the response R1 for cmd12 (stop transmission cmd).

7. cmd12_r1_err<Ncr>
To send the response R1 for cmd12 after the Ncr clock of cmd received.

8. cmd12_r1b<Nbusy_start><Nbusy>
To send the response R1 for cmd12 with busy period for Nbusy clock and the busy start after the Nbusy_start clock of response.

9. cmd12_r1b_err<Nbusy_start><Nbusy>
To send the response R1 for cmd12 with busy end bit error and the busy period if for Nbusy clock the busy will start after the Nbusy_start clock of response.

10. cmd13_r1
To send the response R1 for cmd13(ask for status reg cmd).

11. cmd13_r1_err<Ncr>
To send the response R1 for cmd13(ask for status reg cmd) after the Ncr clock of cmd received.

12. cmd16_r1
To send the response R1 for cmd16(set block length cmd).

13. cmd16_r1_err<Ncr>
To send the response R1 for cmd16 (set block length cmd) after the Ncr clock of cmd received.

14. cmd17_r1
To send the response R1 for cmd17(single block read cmd).

15. cmd17_r1_err<Ncr><No of clock resp-data> <crc error> <data end bit error>
To send the response R1 for cmd17(single block read) after the specified Ncr clock and the data will be available after the No of clock specified in the No of clock resp-data<4 bit in hex>and this data block will be padded as -
if crc error field (1bit in hex) is ‘0’ then the data block will be pad with correct crc
if crc error field (1bit in hex) is ‘1’ then the data block will be pad with the wrong crc.
if data end bit error field (one bit in hex) is ‘1’ then the data block will send with out the end bit error.
if data end bit error field (one bit in hex)is ‘0’ then the data block will send with the end bit error.

16. cmd18_r1
To send the response R1 for cmd18(multiblock read command) followed by data block.

17. cmd18_r1_err<Ncr><Block No> <No of clock between block> <crc err>< end bit error>
To send the response R1 for cmd18 (multiblock read command) after the specified Ncr clock of cmd received followed by data block.
the block no<4 bit in hex> field will specified that after which block next following field will take part.
the No of clock between block field<4 bit in hex> will defined that next block of specified in block no will be available after No of clock between block.
if the crc err(1 bit in hex) is ‘1’ then the next block of specified in block no will pad with wrong crc
if the crc err(1 bit in hex) is ‘0’ then the next block of specified in block no will pad with correct crc
if the end bit error(1 bit in hex) is ‘1’ then the next block of specified in block no will send with end bit error
if the end bit error(1 bit in hex) is ‘1’ then the next block of specified in block no will send without end bit error

18. cmd24_r1

To send the response R1 for cmd24(single block write) followed by the crc status for the data block received.

19. cmd24_r1_err<Ncr><clk between data-crc status> <crc status err> <crc status end bit error)><Nbusy_start><Nbusy> <busy end bit err>

To send the response R1 for cmd24 of after the specified Ncr clock of cmd received followed by the crc status for the data block received after the clock specified in the clk between data-crc status <4 bit in hex>field and

if the crc status err (1 bit in hex) is '1' then the crc status will go as crc err.

if the crc status err (1 bit in hex) is '0' then the crc status will go without crc err.

if the crc status end bit (1 bit in hex) is '1' then the crc status will go with crc status end bit error

if the crc status end bit (1 bit in hex) is '0' then the crc status will go without crc status end bit error

and the busy period if for Nbusy clock the busy will start after the Nbusy_start clock of crc status.

if the busy end bit err (1 bit in hex) is '1' then the busy period will end with end bit error

if the busy end bit err (1 bit in hex) is '0' then the busy period will end without end bit error

20. cmd25_r1

To send the response R1 for cmd25(multi block write) followed by the data block crc status.

21. cmd25_r1_err<Ncr><block No><clk between data-crc status><crc status err><end bit><Nbusy_start*><Nbusy> <busy end bit err>

To send the response R1 for cmd25 after the specified Ncr clock of cmd received followed by the crc status for the data block received

the block no<4 bit in hex> field will specified that after which block next following field will take part.

the clk between data-crc status field<4 bit in hex> will defined that next crc status of specified in block no will be available after clk between data-crc status clock.

if the crc status err (1 bit in hex) is '1' then the crc status will go as crc err.

if the crc status err (1 bit in hex) is '0' then the crc status will go without crc err.

if the crc status end bit (1 bit in hex) is '1' then the crc status will go with crc status end bit error

if the crc status end bit (1 bit in hex) is '0' then the crc status will go without crc status end bit error

and the busy period if for Nbusy clock the busy will start after the Nbusy_start clock of crc status.

if the busy end bit err (1 bit in hex) is '1' then the busy period will end with end bit error

if the busy end bit err (1 bit in hex) is '0' then the busy period will end without end bit error

22. cmd27_r1

To send the response R1 for cmd27(to set the programmable bit of csd).

22. cmd27_r1_err<Ncr>

To send the response R1 for cmd27 after the Ncr clock of cmd received.

23. cmd28_r1

To send the response R1 for cmd28(to set the write protected bit).

24. cmd28_r1_err<Ncr>

To send the response R1 for cmd28 after the Ncr clock of cmd received.

25. cmd28_r1b<Nbusy_start><Nbusy>

To send the response R1 for cmd28 with busy period for Nbusy clock and the busy start after the Nbusy_start clock of response.

26. cmd28_r1b_err<Nbusy_start><Nbusy>
To send the response R1 for cmd28 with busy end bit error and the busy period if for Nbusy clock the busy will start after the Nbusy_start clock of response.

27. cmd29_r1
To send the response R1 for cmd29(to clear the write protected bit).

28. cmd29_r1_err<Ncr>
To send the response R1 for cmd29 after the Ncr clock of cmd received.

29. cmd29_r1b<Nbusy_start*><Nbusy>
To send the response R1 for cmd29 with busy period for Nbusy clock and the busy start after the Nbusy_start clock of response.

30. cmd29_r1b_err<Nbusy_start*><Nbusy>
To send the response R1 for cmd29 with busy end bit error and the busy period if for Nbusy clock the busy will start after the Nbusy_start clock of response.

31. cmd30_r1
To send the response R1 for cmd30(ask to send the status of write protected bit).

32. cmd30_r1_err<Ncr>
To send the response R1 for cmd30 after the Ncr clock of cmd received.

33. cmd32_r1
To send the response R1 for cmd32(erase write block start cmd).

34. cmd32_r1_err<Ncr>
To send the response R1 for cmd32 after the Ncr clock of cmd received.

35. cmd33_r1
To send the response R1 for cmd33(erase write block end).

36. cmd33_r1_err<Ncr>
To send the response R1 for cmd33 after the Ncr clock of cmd received.

37.cmd38_r1
To send the response R1 for cmd38 (erase cmd).

37.cmd38_r1_err<Ncr>
To send the response R1 for cmd38 after the Ncr clock of cmd received.

38.cmd38_r1b<Nbusy_start><Nbusy>
To send the response R1 for cmd38 with busy period for Nbusy clock and the busy start after the Nbusy_start clock of response.

39.cmd38_r1b_err<Nbusy_start><Nbusy>
To send the response R1 for cmd38 with busy end bit error and the busy period if for Nbusy clock the busy will start after the Nbusy_start clock of response.

40.cmd42_r1

To send the response R1 for cmd42(to set or reset the password or lock and unlock the card).and if the there is cmd55 before cmd42 then will send the response for ACMD42(set clr_card_detect).

41.cmd42_r1_err<Ncr>
To send the response R1 for cmd42 after the Ncr clock of cmd received.and if the there is cmd55 before cmd42 then will send the response for ACMD42(set clr_card_detect).

42.cmd55_r1
To send the response R1 for cmd55(to indicate that the next command is ACMD command).

43.cmd55_r1_err<Ncr>
To send the response R1 for cmd55 after the Ncr clock of cmd received.

44.cmd56_r1
To send the response R1 for cmd56 (to get/ to send the data block for general type).

45.cmd56_r1_rd<Ncr><clock between resp and data> <data crc error> <data end bit error>
To send the response R1 for cmd56 after the Ncr clock of cmd received.and the data will be available after the No of clock specified in the No of clock resp-data.and this data block will be padded as -
if crc error field (1bit in hex) is '0' then the data block will be pad with correct crc
if crc error field (1bit in hex) is '1' then the data block will be pad with the wrong crc.
if data end bit error field (one bit in hex) is '1' then the data block will send with out the end bit error.
if data end bit error field (one bit in hex)is '0' then the data block will send with the end bit error.

46.cmd56_r1_wr<Ncr><clk between data-crc status><crc status err><crc end bit error><Nbusy_start><Nbusy> <busy end bit err>
To send the response R1 for cmd56 of after the Ncr clock of cmd received followed by the crc status for the data block received after the clock specified in the clk between data-crc status field and
if the crc status err (1 bit in hex) is '1' then the crc status will go as crc err.
if the crc status err (1 bit in hex) is '0' then the crc status will go without crc err.
if the crc status end bit (1 bit in hex) is '1' then the crc status will go with crc status end bit error
if the crc status end bit (1 bit in hex) is '0' then the crc status will go without crc status end bit error
and the busy period if for Nbusy clock the busy will start after the Nbusy_start clock of crc status.
if the busy end bit err (1 bit in hex) is '1' then the busy period will end with end bit error
if the busy end bit err (1 bit in hex) is '0' then the busy period will end without end bit error

47. acmd6_r1
To send the response R1 for acmd6 (set bus width).

47. acmd6_r1_err<Ncr>
To send the response R1 for cmd6 after the Ncr clock of cmd received.

48. acmd13_r1
To send the response R1 for acmd13(ask for SD mem card status).

49. acmd13_r1_err<Ncr>
To send the response R1 for acmd13 after the Ncr clock of cmd received.

50.acmd22_r1
To send the response R1 for acmd22(ask to send the No of clock block written with out error).

51.acmd22_r1_err<Ncr>
To send the response R1 for cmd22 after the Ncr clock of cmd received.

52. acmd23_r1
To send the response R1 for acmd23(set no of block to be erase before writing).

53. acmd23_r1_err<Ncr>
To send the response R1 for cmd23 after the Ncr clock of cmd received.

54. acmd42_r1
To send the response R1 for acmd42(set/ reset the pull up resister).

55. acmd42_r1_err<Ncr>
To send the response R1 for cmd42 after the Ncr clock of cmd received.

56. cmd9_r2
To send the response R2 (CSD *card specific data*) for cmd9.

57. cmd9_r2_err<Ncr>
To send the response R2 (CSD *card specific data*) for cmd9 after the Ncr clock of cmd received.

58. cmd10_r2
To send the response R2 (CID *card identification data*) for cmd 10.

58. cmd10_r2_err<Ncr>
To send the response R2 (CID *card identification data*) for cmd 10 after the Ncr clock of cmd received.

59. acmd41_r3<busy bit>
To send the response R3 for acmd41.
if busy bit is '1' then 31 bit of OCR will go as 1
if busy bit is '0' then 31 bit of OCR will go as 0

60. acmd41_r3_err<busy bit><Ncr>
To send the response R3 for acmd41 after the specified Ncr clock.
if busy bit (1 bit in hex) is '1' then 31 bit of OCR in the response will go as 1
if busy bit (1 bit in hex)is '0' then 31 bit of OCR in the response will go as 0

61. cmd5_r4<card ready bit>
To send the response R4 for cmd5
if card ready bit (1 bit in hex)is '1' then in the response the 39 bit will go as 1
if card ready bit (1 bit in hex)is '0' then in the response the 39 bit will go as 0

62. cmd5_r4_err<card ready bit><Ncr>
To send the response R4 for cmd5 after the specified Ncr clock of cmd received
if card ready bit (1 bit in hex)is '1' then in the response the 39 bit will go as 1
if card ready bit (1 bit in hex)is '0' then in the response the 39 bit will go as 0

63. cmd52_r5
To send the response R5 for cmd52.

64. cmd52_r5_err<Ncr>
To send the response R1 for cmd52 after the Ncr clock of cmd received.

65. cmd53_rd_r5
To send the response R5 for cmd53 read.

66. cmd53_rd_r5_err<Ncr><block No><No of clock between block > <crc error><end bit error>

To send the response R1 for cmd53 after the specified Ncr clock of cmd received followed by data block.
the block no<4 bit in hex> field will specified that in after which block next following field will take part.
the No of clock between block field<4 bit in hex> will defined that next block of specified in block no will be available after No of clock between block.
if the crc err(1 bit in hex) is '1' then the next block of specified in block no will pad with wrong crc
if the crc err(1 bit in hex) is '0' then the next block of specified in block no will pad with correct crc
if the end bit error(1 bit in hex) is '1' then the next block of specified in block no will send with end bit error
if the end bit error(1 bit in hex) is '1' then the next block of specified in block no will send without end bit error

67. cmd53_wr_r5

To send the response R5 for cmd53 write in byte/block mode followed by the crc status.

68. cmd53_wr_r5_err<Ncr><block No><clk between data-crc status><crc status err><crc end bit err><Nbusy_start><Nbusy> <busy end bit err>

To send the response R1 for cmd53 after the specified Ncr clock of cmd received followed by the crc status for the data block received
the block no<4 bit in hex> field will specified that after which block next following field will take part.
the **clk between data-crc status** field<4 bit in hex> will defined that next crc status of specified in block no will be available after **clk between data-crc status** clock.
if the crc status err (1 bit in hex) is '1' then the crc status will go as crc err.
if the crc status err (1 bit in hex) is '0' then the crc status will go without crc err.
if the crc status end bit (1 bit in hex) is '1' then the crc status will go with crc status end bit error
if the crc status end bit (1 bit in hex) is '0' then the crc status will go without crc status end bit error
and the busy period if for Nbusy clock the busy will start after the Nbusy_start clock of crc status.
if the busy end bit err (1 bit in hex) is '1' then the busy period will end with end bit error
if the busy end bit err (1 bit in hex) is '0' then the busy period will end without end bit error

69. cmd3_r6

To send the response R6 for the cmd3.

69. cmd3_r6_err<Ncr>

To send the response R1 for cmd12 after the Ncr clock of cmd received.

70. cmd0_no_resp

when the device mode will received the cmd0 it will not send the response and will wait for next cmd

71. cmd4_no_resp

when the device mode will received the cmd04it will not send the response and will wait for next cmd

72. cmd15_no_resp

when the device mode will received the cmd15 it will not send the response and will wait for next cmd

73. busy<No clock>

To feed the default busy period.

74. send_inta <No of clock ><start time (in neno-sec.) ><end_time(in neno_sec) ><block No>

To send the interrupt for host during the read or write transfer the interrupt will assert after (start time till end time)the specified No of clock and after the specified no of block.

75. send_inta_err <No of clock ><start time (in neno_sec)><end_time(in neno_sec)><block No><type>

To send the interrupt for host during the read or write transfer the interrupt will assert after (start time till end time)the specified No of clock <4 bit in hex>and after the specified block No.<4 bit in hex>
and here the type will work during the four bit mode and specified as
if type is '0' then during the interrupt time device will send 0 during both clk.
if type is '1' then during the interrupt time device will send 1 drumming both clk.
if type is '2' then during the interrupt time device will send as per as interrupt mean one cycle low and next cycle as high.

76. no_of_io<1 bit in hex>

To set the no of io present in the device model.

77. mem_not_present

To set the memory is not present in the model

78. ocr< 6 bit in hex>

To set the ocr value.

79. rca<4 bit in hex>

To set the rca value.

80.data_access_time<No of clock>

To feed the default data access time period.
No of clock (4 bit in hex)

81.end

To indicate for model that now this is the time for ending the simulation or now no more command to be force.

82.cmd15_no_resp

when ever there is cmd15 then this cmd will not send the response.

83.cmd7_no_resp

when ever there is cmd7 for deselction of card.

84.cmd1_r3

To send response r3 for cmd1 for mmc card

85. cmd3_mmc_r1

To send r1 response for cmd3 for mmc card

86. send_intr_no_clk <delay_time><intr_time>

To send interrupt to host when sdclk isn't present .
delay_time - To send interrupt after this much of time.
intr_time - To driver interrupt for this much of time.

87.cmd_conflict

To drive data line to zero level when the host issues command for the purpose of getting command conflict.
Security commands:

88.acmd43_r1

To send a r1 respose in cmd line and MKB values as data block.

89.acmd44_r1
To send r1 response in cmd line and MID value as data block.

90.acmd45_r1
Send r1 response and store challenge1 that comes as argument in this command.This argument field is the argument for forthcoming protected area access command.

91.acmd46_r1
Send r1 response in command line and send challenge2 as a data block.

92.acmd47_r1
Send r1 response in command line and receive res2.

93.acmd48_r1
Send r1 response in command line and send res1 as a data block.

99.acmd18_r1
Send r1 response in command line and send requested number of blocks.

100.acmd25_r1
Send r1 response in command line and receive number of blocks as per the argument value in acmd45_r1.

101.acmd38_r1
Send r1 response in command line and perform erase sequence as per the requested number of blocks.

102.acmd49_r1
Send r1 response in command line and perform a change of secure area as per the argument value in acmd45_r1.

103.acmd26_r1
Send r1 response in command line and perform a secure write MKB transaction to the card.

104. no_of_trans<32 bit **trans_cnt1** in hex><1 bit **trans_mode1**>
This command is used in random testing for memory command only. This task shall be called before memory write or read transaction.
trans_cnt1 - The following command will be executed for this much of count
trans_mode1 - Indicates dma or nondam mode, '1' for non_dma mode and '0' for dma mode.

101 . cmd53_rd_wr<32 bit **rd_wr_cnt** in hex><1 bit **trans_mode1**>
This command is used in random testing for io command only.
rd_wr_cnt - The following command will be executed for this much of count
trans_mode1 - Indicates dma or nondam mode, '1' for non_dma mode and '0' for dma mode.

9.3.2 BLOCKING COMMAND

1.clock_n<delay>

Delay - (4-bit value in hex) -- hexadecimal value within 0000 - ffff)
This command is used to force some delay.

2. delay<time in neno_sec>

During the specified time there will not be any operation used when there is no clock or during the time when there is clock got stop.

9.3.3 ERROR FORCING COMMAND

1. set_error_card_status <4 bit in hex>

It will set the respective field of card status register which will defined in the type field

[0].out_of_range
[1].address_error
[3].block_len_error
[4].erase_seq_error
[5].erase_param
[6].wp_violation
[7].lock_unlock_failed
[8].com_crc_error
[9].illegal_command
[10].card_ecc_failed
[11].cc_error
[12].csd_cid_overwrite
[13].ready_for_data
[14].ake_seq_error
[15].RFU

2.set_error_resp<3 bit in hex >

[0] bit.com_crc_error
The crc error check of the previous command failed.
[1]. illegal_command
Command not legal for the card state.
[3:2].io_current_state
0 -DIS-when the card is in initalized or standby or inactive states (or card is not selected).
1-CMD-1.Command waiting (no transaction suspended).
2.Command waiting (All cmd53 transaction suspended).
3. Executing CMD52.
2-TRN-Command is executing with data transfer using DAT line.
3- RFU.
[4].error
A general or an unknown error occurred during the operation.
[5].function_no
An invalid function no was requested.
[6].out of range
The command's argument was out of the allowed range for this card.
[7].command_index
[8].force_crc_error
[9] resp_end_bit_err (To set the end bit error in the response).
[11:10] RFU

4. time_out_cmd

The device model will not send response for the command will result as a time-out for that.

5. time_out_data <block No>

The device model will not send data block, next to data block which is specified in the block no field and will result as a time-out for that data.

6.suspend<type>

If the type is 0 then the device will denied to suspend the function.By default the type is kept as one.

10. MESSAGES DISPLAYED BY MODELS

10.1 Messages Displayed by AHB Model

10.1.1 General Messages

SYSTEM ADDRESS REGISTER HAS BEEN WRITTEN

BLOCK SIZE REGISTER HAS BEEN WRITTEN

BLOCK COUNT REGISTER HAS BEEN WRITTEN

ARGUMENT REGISTER HAS BEEN WRITTEN

TRANSFER MODE REGISTER HAS BEEN WRITTEN

COMMAND REGISTER HAS BEEN WRITTEN

RESPONSE0 REGISTER HAS BEEN WRITTEN

RESPONSE1 REGISTER HAS BEEN WRITTEN

RESPONSE2 REGISTER HAS BEEN WRITTEN

RESPONSE3 REGISTER HAS BEEN WRITTEN

RESPONSE4 REGISTER HAS BEEN WRITTEN

RESPONSE5 REGISTER HAS BEEN WRITTEN

RESPONSE6 REGISTER HAS BEEN WRITTEN

RESPONSE7 REGISTER HAS BEEN WRITTEN

BUFFER DATA PORT REGISTER HAS BEEN WRITTEN

PRESENT STATE REGISTER HAS BEEN WRITTEN

WAKEUP CONTROL REGISTER HAS BEEN WRITTEN

BLOCK GAP CONTROL REGISTER HAS BEEN WRITTEN

POWER CONTROL REGISTER HAS BEEN WRITTEN

HOST CONTROL REGISTER HAS BEEN WRITTEN

SOFTWARE RESET REGISTER HAS BEEN WRITTEN

TIMEOUT CONTROL REGISTER HAS BEEN WRITTEN

CLOCK CONTROL REGISTER HAS BEEN WRITTEN

NORMAL INTERRUPT STATUS REGISTER HAS BEEN WRITTEN

ERROR INTERRUPT STATUS REGISTER HAS BEEN WRITTEN

NORMAL INTERRUPT STATUS ENABLE REGISTER HAS BEEN WRITTEN

ERROR INTERRUPT STATUS ENABLE REGISTER HAS BEEN WRITTEN

NORMAL INTERRUPT SIGNAL ENABLE REGISTER HAS BEEN WRITTEN

ERROR INTERRUPT SIGNAL ENABLE REGISTER HAS BEEN WRITTEN

AUTO CMD12 ERROR STATUS REGISTER HAS BEEN WRITTEN

HOST CONTROLLER CAPABILITIES REGISTER HAS BEEN WRITTEN

MAX CURRENT CAPABILITIES REGISTER HAS BEEN WRITTEN

SLOT INTERRUPT STATUS REGISTER HAS BEEN WRITTEN

HC VERSION REGISTER HAS BEEN WRITTEN

BLOCK COUNT AND BLOCK SIZE REGISTERS HAS BEEN WRITTEN

RECEIVED EXPECTED DATA FROM SYSTEM ADDRESS REGISTER

RECEIVED EXPECTED DATA FROM BLOCK COUNT REGISTER

RECEIVED EXPECTED DATA FROM BLOCK SIZE REGISTER

RECEIVED EXPECTED DATA FROM ARGUMENT REGISTER

RECEIVED EXPECTED DATA FROM TRANSFER MODE REGISTER

RECEIVED EXPECTED DATA FROM SD COMMAND REGISTER

RECEIVED EXPECTED DATA FROM RESPONSE0 REGISTER

RECEIVED EXPECTED DATA FROM RESPONSE1 REGISTER

RECEIVED EXPECTED DATA FROM RESPONSE2 REGISTER

RECEIVED EXPECTED DATA FROM RESPONSE3 REGISTER

RECEIVED EXPECTED DATA FROM RESPONSE4 REGISTER

RECEIVED EXPECTED DATA FROM RESPONSE5 REGISTER

RECEIVED EXPECTED DATA FROM RESPONSE6 REGISTER

RECEIVED EXPECTED DATA FROM RESPONSE7 REGISTER

RECEIVED EXPECTED DATA FROM BUFFER DATA PORT REGISTER

RECEIVED EXPECTED DATA FROM PRESENT STATE REGISTER

RECEIVED EXPECTED DATA FROM WAKEUP CONTROL REGISTER

MISMATCH IN RECEIVED DATA FROM WAKEUP CONTROL REGISTER

RECEIVED EXPECTED DATA FROM BLOCK GAP CONTROL REGISTER

RECEIVED EXPECTED DATA FROM POWER CONTROL REGISTER

RECEIVED EXPECTED DATA FROM SD HOST CONTROL REGISTER

RECEIVED EXPECTED DATA FROM SOFTWARE RESET CONTROL REGISTER

RECEIVED EXPECTED DATA FROM TIMEOUT CONTROL REGISTER

RECEIVED EXPECTED DATA FROM CLOCK CONTROL REGISTER

RECEIVED EXPECTED DATA FROM ERROR INTERRUPT STATUS REGISTE

RECEIVED EXPECTED DATA FROM NORMAL INTERRUPT STATUS REGISTER

RECEIVED EXPECTED DATA FROM ERROR INTERRUPT STATUS ENABLE REGISTER

RECEIVED EXPECTED DATA FROM NORMAL INTERRUPT STATUS ENABLE REGISTER

RECEIVED EXPECTED DATA FROM ERROR INTERRUPT SIGNAL ENABLE REGISTER

RECEIVED EXPECTED DATA FROM NORMAL INTERRUPT SIGNAL ENABLE REGISTER

RECEIVED EXPECTED DATA FROM AUTO CMD12 ERROR STATUS REGISTER

RECEIVED EXPECTED DATA FROM HOST CONTROLLER CAPABILITIES REGISTER

RECEIVED EXPECTED DATA FROM MAX CURRENT CAPABILITIES REGISTER

RECEIVED EXPECTED DATA FROM HOST CONTROLLER VERSION REGISTER

RECEIVED EXPECTED DATA FROM SLOT INTERRUPT STATUS REGISTER

10.1.2 Error Messages

MISMATCH IN RECEIVED DATA FROM SLOT INTERRUPT STATUS REGISTER

MISMATCH IN RECEIVED DATA FROM HOST CONTROLLER VERSION REGISTER

MISMATCH IN RECEIVED DATA FROM MAX CURRENT CAPABILITIES REGISTER

MISMATCH IN RECEIVED DATA FROM HOST CONTROLLER CAPABILITIES REGISTER

MISMATCH IN RECEIVED DATA FROM AUTO CMD12 ERROR STATUS REGISTER

MISMATCH IN RECEIVED DATA FROM NORMAL INTERRUPT SIGNAL ENABLE REGISTER

MISMATCH IN RECEIVED DATA FROM ERROR INTERRUPT SIGNAL ENABLE REGISTER

MISMATCH IN RECEIVED DATA FROM NORMAL INTERRUPT STATUS ENABLE REGISTER

MISMATCH IN RECEIVED DATA FROM ERROR INTERRUPT STATUS ENABLE REGISTER

MISMATCH IN RECEIVED DATA FROM NORMAL INTERRUPT STATUS REGISTER

MISMATCH IN RECEIVED DATA FROM ERROR INTERRUPT STATUS REGISTE

MISMATCH IN RECEIVED DATA FROM CLOCK CONTROL REGISTER

MISMATCH IN RECEIVED DATA FROM TIMEOUT CONTROL REGISTER

MISMATCH IN RECEIVED DATA FROM SOFTWARE RESET CONTROL REGISTER

MISMATCH IN RECEIVED DATA FROM SD HOST CONTROL REGISTER

MISMATCH IN RECEIVED DATA FROM POWER CONTROL REGISTER

MISMATCH IN RECEIVED DATA FROM BLOCK GAP CONTROL REGISTER

MISMATCH IN RECEIVED DATA FROM PRESENT STATE REGISTER

MISMATCH IN RECEIVED DATA FROM BUFFER DATA PORT REGISTER

MISMATCH IN RECEIVED DATA FROM RESPONSE0 REGISTER

MISMATCH IN RECEIVED DATA FROM RESPONSE1 REGISTER

MISMATCH IN RECEIVED DATA FROM RESPONSE2 REGISTER

MISMATCH IN RECEIVED DATA FROM RESPONSE3 REGISTER

MISMATCH IN RECEIVED DATA FROM RESPONSE4 REGISTER

MISMATCH IN RECEIVED DATA FROM RESPONSE5 REGISTER

MISMATCH IN RECEIVED DATA FROM RESPONSE6 REGISTER

MISMATCH IN RECEIVED DATA FROM RESPONSE7 REGISTER

MISMATCH IN RECEIVED DATA FROM SD COMMAND REGISTER

MISMATCH IN RECEIVED DATA FROM SYSTEM ADDRESS REGISTER

MISMATCH IN RECEIVED DATA FROM BLOCK COUNT REGISTER

MISMATCH IN RECEIVED DATA FROM BLOCK SIZE REGISTER

MISMATCH IN RECEIVED DATA FROM ARGUMENT REGISTER

MISMATCH IN RECEIVED DATA FROM TRANSFER MODE REGISTER

10.2 Messages Displayed by Device Model

10.2.1 General Messages

CMD0 IS RECEIVED

CMD1 IS RECEIVED

CMD2 IS RECEIVED

CMD3 IS RECEIVED

CMD4 IS RECEIVED

CMD5 IS RECEIVED

CMD7 IS RECEIVED FOR SELECTION OF CARD

CMD9 IS RECEIVED

CMD10 IS RECEIVED

CMD12 IS RECEIVED

CMD13 IS RECEIVED

CMD15 IS RECEIVED

CMD16 IS RECEIVED ---------->(BLOCK LENGTH)

CMD17 IS RECEIVED---------->(READ ADDRESS FOR SINGLE BLOCK)

CMD18 IS RECEIVED---------->(READ ADDRESS FOR MULTI-BLOCK)

CMD24 IS RECEIVED---------->(WRITE ADDRESS FOR SINGLE BLOCK)

CMD25 IS RECEIVED---------->(WRITE ADDRESS FOR MULTI-BLOCK)

CMD26 IS RECEIVED

CMD27 IS RECEIVED

CMD28 IS RECEIVED---------->(SET WRITE PROTECTION ADDRESS)

CMD29 IS RECEIVED---------->(CLEAR WRITE PROTECTION ADDRESS)

CMD30 IS RECEIVED--------->(SEND WRITE PROTECT DATA ADDRESS)

CMD32 IS RECEIVED->(SET FIRST WRITE BLOCK ADDRESS FOR ERASE)

CMD33 IS RECEIVED-->(SET LAST WRITE BLOCK ADDRESS FOR ERASE)

CMD38 IS RECEIVED

CMD42 IS RECEIVED

CMD52 WRITE IS RECEIVED ---------->

CMD52 READ IS RECEIVED ---------->

CMD53 WRITE IS RECEIVED FOR FUNCTION NO ---------->

CMD53 READ IS RECEIVED FOR FUNCTION NO ---------->

CMD55 IS RECEIVED

CMD56 IS RECEIVED

CMD59 IS RECEIVED

ACMD6 IS RECEIVED

ACMD13 IS RECEIVED

ACMD22 IS RECEIVED

ACMD23 IS RECEIVED

ACMD41 IS RECEIVED

ACMD42 IS RECEIVED

ACMD51 IS RECEIVED

SECURITY COMMAND(CMD_INDEX) IS RECEIVED

CMD5 WITH OCR MATCH HAS RECEIVED

CMD5 WITH OCR MISMATCH HAS RECEIVED

CMD7 IS RECEIVED FOR DISSELECTION OF CARD

HOST HAS REQUESTED TO CARD FOR SUSPENDED THE FUNCTION --->

CARD HAS ACCEPTED THE REQUEST FOR SUSPENDED THE FUNCTION ->

REQUESTED FOR CARD SUSPEND HAS DICLINE FOR THE FUNCTION ->

CMD52 READ IS RECEIVED FOR CIS DATA ADD ---------->

R1, RESPONSE FOR CMD0 HAS SENT

R1, RESPONSE WITH (COMMAND ARGUMENT) WAS OUT OF THE ALLOWED RANGE HAS SENT

R1, RESPONSE WITH A MISALINGNED ADDRESS WAS USED IN THE COMMAND HAS SENT

RESPONSE WITH THE BLOCK LENGTH IS NOT ALLOWED FOR THIS CARD HAS SENT

R1, RESPONSE WITH AN ERR IN THE SEQUENCE OF ERASE COMMANDS HAS SENT

R1, RESPONSE WITH AN INVALID SELECTION OF WRITE BLOCK FOR ERASE HAS SENT

R1, RESPONSE WITH ATTEMPT TO PROGRAM A WRITE-PROTECTED BLOCK HAS SENT

R1, RESPONSE WITH CARD IS LOCKED BY THE HOST HAS SENT

R1, RESPONSE WITH AN ATTEMPT TO ACCESS A LOCKED CARD HAS SENT

R1, RESPONSE WITH THE CRC CHECK OF THE PREVIOUS COMMAND FAILED HAS SENT

R1, RESPONSE WITH THE COMMAND NOT LEAGAL FOR THE CARD STATE HAS SENT

R1, RESPONSE WITH ECC APPLIED BUT FAILED TO CORRECT THE DATA HAS SENT

R1, RESPONSE WITH INTERNAL CARD CONTROLLED ERR HAS SENT

R1, RESPONSE WITH CARD IS IN IDLE STATE

R1, RESPONSE WITH CARD IS IN READY STATE HAS SENT

R1, RESPONSE WITH CARD IS IN IDENTIFICATION STATE HAS SENT

R1, RESPONSE WITH CARD IS IN STAND-BY STATE HAS SENT

R1, RESPONSE WITH CARD IS IN TRANSFER STATE HAS SENT

R1, RESPONSE WITH CARD IS IN SENDING-DATA STATE HAS SENT

R1, RESPONSE WITH CARD IS IN RECEIVE-DATA STATE HAS SENT

R1, RESPONSE WITH CARD IS IN PROGRAMMING STATE HAS SENT

R1, RESPONSE WITH CARD IS IN DISCONNECT STATE HAS SENT

R1, RESPONSE FOR CMD1 HAS SENT

R1, RESPONSE FOR CMD7 HAS SENT

R1, RESPONSE FOR CMD9 HAS SENT

R1, RESPONSE FOR CMD59 HAS SENT

R1, RESPONSE FOR CMD12 HAS SENT

R1, RESPONSE FOR CMD13 HAS SENT

R1, RESPONSE FOR CMD16 HAS SENT

R1, RESPONSE FOR CMD17 HAS SENT

R1, RESPONSE FOR CMD18 HAS SENT

R1, RESPONSE FOR CMD24 HAS SENT

R1, RESPONSE FOR CMD27 HAS SENT

R1, RESPONSE FOR CMD28 HAS SENT

R1, RESPONSE FOR CMD29 HAS SENT

R1, RESPONSE FOR CMD30 HAS SENT

R1, RESPONSE FOR CMD32 HAS SENT

R1, RESPONSE FOR CMD33 HAS SENT

R1, RESPONSE FOR CMD38 HAS SENT

R1, RESPONSE FOR CMD42 HAS SENT

R1, RESPONSE FOR CMD55 HAS SENT

R1, RESPONSE FOR ACMD6 HAS SENT

R1, RESPONSE FOR ACMD13 HAS SENT

R1, RESPONSE FOR ACMD22 HAS SENT

R1, RESPONSE FOR ACMD23 HAS SENT

R1, RESPONSE FOR ACMD42 HAS SENT

R1,RESPONSE FOR (SECURITY CMD_INDEX)HAS BEEN SENT

R1, RESPONSE WITH AN UNKNOWN ERR OCCURED DURING THE OPERATION HAS SENT

R1, RESPONSE WITH ERR IN CID/CSD OVER WRITE HAS SENT

R1, RESPONSE WITH CARD IS IN RFU STATE HAS SENT

R1, RESPONSE WITH CARD BUFFER IS EMPTY HAS SENT

R1, RESPONSE WITH CARD IS WAITTING FOR APP SPECIFIC COMMAND HAS SENT

R1, RESPONSE WITH ERR IN THE SEQUENCE OF AUTHENTICATION HAS SENT

R1, RESPONSE WITH CARD IS SDIO ONLY HAS SENT

R2, RESPONSE FOR CMD2 HAD SENT

R2, RESPONSE FOR CMD9 HAD SENT

R2, RESPONSE FOR CMD10 HAS SENT

R2, RESPONSE WITH CARD IS LOCKED BY USER HAS SENT

R2, RESPONSE WITH CARD IS UNLOCKED BY USER HAS SENT

R2, RESPONSE WITH HOST IS ATTEMPTS TO ERASE A WRITE PROTECTED SECTOR HAS SENT OR MAKES A ERASE SEQUENCE FOR WRITE PROTECTED SECTOR

R2, RESPONSE WITH THERE IS ERR DURING LOCK /UNLOCKING OPEARTION

R2, RESPONSE WITH A GENERAL OR AN UKKNOWN ERR OCCURRED DURING THE OPERATION HAS SENT

R2, RESPONSE WITH INTERNAL CARD CONTROLLED ERR HAS SENT

R2, RESPONSE WITH CARD INTERNAL ECC WAS APPLIED BUT FAILED TO CORRECT THE DATA HAS SEN

R2, RESPONSE WITH THE COMMAND TRIED TO WRITE PROTECTED BLOCK

R2, RESPONSE WITH AN INVALID SELECTION,SECTORS OR GROUPS,FOR ERASE HAS SENT

R2, RESPONSE WITH THE COMMAND'S ARGUMENT WAS OUT OF THE ALLOWED RANGE FOR THIS CARD

R2, RESPONSE WITH THERE IS CDS OVERWRITE HAS SENT

R2, RESPONSE WITH THE CARD IS IN IDLE STATE AND RUNNING THE INITIALIZING PROCESS HAS SENT

R2, RESPONSE WITH AN ERASE SEQUENCE IN THE SEQUENCE OF ERASE COMMAND OCCURRED HAS SENT

R2, RESPONSE WITH AN ERASE SEQUENCE WAS CLEARED BEFORE EXECTING BECAUSE AN OUT OF ERASE SEQUENCE COMMAND WAS RECEIVED HAS SENT

R2, RESPONSE WITH AN ILLEGAL COMMAND CODE WAS DETECTED HAS SENT

R2, RESPONSE WITH THE CRC CHECK OF THE LAST COMMAND FAILED HAS SENT

R2, RESPONSE WITH A MISALIGNED ADDRESS,WHICH DID NOT MATCH THE BLOCK LENGTH,WAS USED IN THE COMMAND HAS SENT

R2, RESPONSE WITH THE PARAMETER ERR HAS SENT

R3, RESPONSE WITH CARD POWER UP STATUS BUSY HAS SENT

R3, RESPONSE FOR ACMD41 HAS SENT

R3, RESPONSE WITH CARD POWER UP STATUS NOT BUSY HAS SENT

R4, RESPONSE FOR CMD5 IS HAS SENT

R4, RESPONSE WITH CARD IS NOT READY TO OPERATE HAS SENT

R4, RESPONSE WITH CARD IS READY TO OPERATE HAS SENT

R4, RESPONSE WITH MEMORY PRESENT HAS SENT

R4, RESPONSE WITH MEMORY NOT-PRESENT HAS SENT

R5, RESPONSE FOR CMD52 READ HAS SENT ---------->

R5, RESPONSE FOR CMD52 WRITE HAS SENT ---------->

R5, RESPONSE WITH COM CRC ERR HAS SENT

R5, RESPONSE WITH CMD NOT LEAGAL FOR THE CARD STATE HAS SENT

R5, RESPONSE WITH CARD IS IN DISABLE STATE HAS SENT

R5, RESPONSE WITH CARD IS IN COMMAND STATE HAS SENT

R5, RESPONSE WITH CARD IS IN DATA TRAN STATE HAS SENT

R5, RESPONSE WITH CARD IS IN RFU STATE HAS SENT

R5, RESPONS WITH GENERAL ERR OCCURED HAS SENT

R5, RESPONSE WITH FUNCTION NUMBER ERR HAS SENT

R5, RESPONSE WITH CMD ARG OUT OF RANGE FOR THIS CARD HAS SENT

R5, RESPONSE FOR CMD53 WRITE HAS SENT

R5, RESPONSE FOR CMD53 READ HAS SENT FOR FUNCTION NO--------->

R6, RESPONSE FOR CMD3 IS HAS SENT

R6, RESPONSE WITH CRC CHECK OF THE PREVIOUS COMMAND FAILED HAS SENT

R6, RESPONSE WITH COMMAND NOT LEGAL FOR THE CARD STATE HAS SENT

R6, RESPONSE WITH GENERAL ERR OCCURED DURING THE OPERATION HAS SENT

HOST IS FORCING TO CARD TO STOP TRANSMISSION

USER HAS REQUESTED TO DEVICE TO SEND RESP WITH CRC ERR

CARD SPECIFIC DATA (CSD) SENT SUCCESSFULLY

CARD IDENTIFICATION DATA (CID) SENT SUCCESSFULLY

USER HAS REQUESTED TO DEVICE TO SEND RESP WITH END BIT ERR

BLOCK NO IS ---------->

START BIT FOR SD 1 BIT MODE SENT SUCCESSFULLY

START BIT FOR SD 4 BIT MODE SENT SUCCESSFULLY

DATA BLOCK ARE SENT SUCCESSFULLY WITH CRC ERR

DATA BLOCK ARE SENT SUCCESSFULLY WITHOUT CRC ERR

DATA BLOCK ARE SENT SUCCESSFULLY WITH END BIT ERR

DATA BLOCK ARE SENT SUCCESSFULLY WITHOUT END BIT ERR

CARD HAS ENTER IN TO READ-WAIT STATE

CARD HAS RESUME FROM READ-WAIT STATE

DEVICE IS SENDING DATA AFTER -->

START BIT FOR SD 1 BIT MODE RECEIVED SUCCESSFULLY

START BIT FOR SD 4 BIT MODE RECEIVED SUCCESSFULLY

BYTES OF DATA RECEIVED SUCCESSFULLY -->No OF BYTE ARE

BYTES OF DATA ARE SENT SUCCESSFULLY -->No OF BYTE ARE

USER HAS REQUESTED TO DEVICE TO SEND CRC STATUS WITH ERR

USER HAS REQUESTED TO DEVICE TO SEND CRC STATUS END BIT ERR

CRC STATUS HAS SENT SUCCESSFULLY

BUSY SINGNAL START AFTER ----->

<---------- CARD IS BUSY ---------->

<---------- BUSY OVER WITHOUT END BIT ---------->

<---------- BUSY OVER WITH END BIT ERR ---------->

HOST HAS REQUESTED TO CARD FOR ABORT THE FUNCTION -->

CARD HAS RELEASED THE DATA LINES(z is driven after stop bit)

FUNCTION1 IS RESUMED

FUNCTION1 IS SUSPENDED AND DATA BUS IS RELEASED

DEVICE IS WORKING AS A SD MEMORY

DEVICE IS WORKING AS SDIO

DEVICE IS WORKING AS COMBO

DEVICE HAS CONNECTED TO SDIO CONTROLLER

DEVICE HAS DISCONNECTED TO SDIO CONTROLLER

INTERRUPT ASSERTED TO SDIO HOST IN SD MODE

INTERRUPT HAS DEASSERTED

IN 4-BIT MULT-BLK Ist CLK OF INTR PERIOD DAT[1] IS HIGH

IN 4-BIT MULT-BLK Ist CLK OF INTR PERIOD DAT[1] IS LOW

IN 4-BIT MULT-BLK IInd CLK OF INTR PERIOD DAT[1] IS LOW

IN 4-BIT MULT-BLK IInd CLK OF INTR PERIOD DAT[1] IS HIGH

DEVICE IS SENDING RESPONSE AFTER --> CMD0 IS RECEIVED

10.2.2 Error messages

ERR IN OPENING CMD FILE

END OF FILE REACHED IN DEVICE SEND DATA FILE

ILLEGAL COMMAND FOR SDIO ONLY CARD

ILLEGAL COMMAND FOR MEM ONLY CARD

ILLEGAL ADDRESS FOR FUNCTION

ERR IN RFU FIELD

ERR IN CMD INDEX OR ILLEGAL COMMAND FOR MEM AND IO

ERR IN DIRECTION BIT

INVALID FUNCTION NUMBER HAS REQUESTED BY THE HOST

ILLEGAL COMMAND FOR COMBO CARD

MISMATCH IN STUFF FIELD ----> FOR COMMAND NO:

TRANSMISSION CYCLEC REDUNDANCY ERR (CRC ERR)

SD COMMAND RECEIVED VIOLATES THE TIMING CONSTRAINTS

DATA RECEIVED VIOLATES THE TIMING CONSTRAINTS

ERR IN STOP BIT FOR DATA

REQUESTING FUNCTION IS NOT ENABLE

X IS DRIVEN IN DATA[0]

X IS DRIVEN IN DATA[1]

X IS DRIVEN IN DATA[2]

X IS DRIVEN IN DATA[3]

X IS DRIVEN IN CMD

ERR IN THE DIRECTION BIT

ERR IN THE CARD RELATIVE ADDRESS RECEVED

MISMATCH IN THE WORKING VOLTAGE FOR CMD5

ERR IN END BIT HAS DETECTED

11. DATA/COMMAND FILES USED BY THE MODEL

11.1 Data Files used by the AHB Master Model

File Format Of "master_rcv.dat" file:

```
01
02
03
04
05
|
|
|
|
|
n
```

The "master_rcv.dat" file is a read/write file. This is opened by the AHB master model during simulation to dump the data bytes received during memory burst read transaction. The data is dumped in byte format. Here 'n' specifies the last byte of data received.

11.2 Command File Used By The AHB MASTER Model

File Format Of "master.v" file:

```
task host_commands;
begin
rst_n(24'hff);

--
//comments
end_task;
end
endtask
```

The "master.v" is read by the AHB master model to read commands. Each command is placed in separate line one after the other in the order it needs to be executed. Comments can be added wherever necessary for the user's better understanding. The last command should compulsorily be an "end" command. The AHB master model performs the tasks in master.v.

11.3 Data Files Used By The AHB Target Model:

File Format Of “target_rcv.dat” file:

```
11
55
99
dd
20
|
|
|
|
|
n
```

The “target_rcv.dat” file is a read/write file. AHB target model dumps the data bytes received during memory read transaction. The data is dumped in DWORD format. Here ‘n’ specifies the last byte of data received.

11.4 Command File Used By The AHB Target Model

File Format Of “target.v” file:

```
task target_commands;
begin
.
.
end_cmd;
end
endtask
```

The “target.v” is read by target model. Each command is placed in separate line one after the other in the order it needs to be executed. Comments can be added wherever necessary for the user’s better understanding. The last command should compulsorily be an “end” command. The AHB target model performs the tasks in target.v.

11.5 Data Files Used By The Device Model:

File Format Of “device_rcvx.dat” file:

```
01
02
03
04
05
 |
 |
 |
 |
 n
```

The “device_rcvx.dat” file is a read/write file. This is opened by the device model during simulation to dump the data bytes received from the card. Each and every byte of data received is written in separate line one after the other in the order it was received. Here ‘n’ specifies the last byte of data received.

11.6 Command File Used By The Device Model
File Format Of “devicex.cmd” file:

```
task device_response;
begin
.
.
 end_cmd;
 end
 endtask
```

The “devicex.v” is read by the device model to read commands. Each command is placed in separate line one after the other in the order it needs to be executed. Comments can be added wherever necessary for the user’s better understanding. The last command should compulsorily be an “end” command. The device model after reading the “end” command will understand the end of the command file and stop the simulation.

12. User modifiable parameters

This section describes some parameters which are present in the mas_defines.v which allows the users to set some values which suit their requirements. The changes to this file should be done before compilation. The following are the user modifiable parameters as of now.

1.burst_sup<1bit argument>

This command decides whether the AHB master model uses burst or single data phase transactions during non dma transactions. The 1 bit argument if given ‘1’ will allow the master model to use burst mode transaction for all register accesses and data transactions. ‘0’ on the other handwill make the target model to use single data phase transactions for register access and data transactions.

2.dma_buff<3 bit value>

This command decides the size of the host DMA buffer size which will be written in the block size register.

13. Randomization:

Dma mode :
Write transaction :

eg :

cmd53_fn1_wr_rand(16'h1,16'h40,16'h4,32'h0,1'b1);
Master model initiates transaction from blksize = 16'h1 to 16'h40 with block count = 16'h4.
Device model receives the read data from ahb and performs on the fly checking.

Corresponding command in device.v is
cmd53_rd_wr(32'h40,1'b1);

Read transaction :

eg :

cmd53_fn1_rd_rand(16'h1,16'h40,16'h4,32'h0,1'b1);
Master model initiates transaction from blksize = 16'h1 to 16'h40 with block count = 16'h4.
Target model receives the write data to be written to target and performs on the fly checking.

Corresponding command in device.v is
cmd53_rd_wr(32'h40,1'b1);

NonDma mode :
Write transaction :

eg :

mem_mul_wr(32'h1, 32'h40 , 32'h4 , 32'h0 , 1'b0 , 1'b1);
Master model initiates transaction from blksize = 16'h1 to 16'h40 with block count = 16'h4.
Device model receives the read data from ahb and performs on the fly checking.

Corresponding command in device.v is
no_of_trans(32'h40, 1'b0);

cmd25_r1;

Read transaction :

eg :

mem_mul_rd(32'h1, 32'h4 , 32'h4 , 32'h0 , 1'b0 , 1'b1);
Master model initiates transaction from blksize = 16'h1 to 16'h40 with block count = 16'h4.
Master model receives the data to be written and performs on the fly checking.

Corresponding command in device.v is
no_of_trans(32'h40, 1'b0);

cmd18_r1;

14. Perl Scripts
1. sim_modelsim - To Compile and Simulate in Modelsim
2. sim_vcs - To Compile and Simulate in VCS

To Compile & Simulate in Modelsim:
The following Files are needed for Compiling the database.
1. comp_modelsim.pl
2. comp_options.pm

The command for compiling the database is given below
Go to sim_modelsim folder and type the below command
> perl comp.pl all

The following files are needed for Simulation :
1. sim_modelsim.pl
2. test_array.txt

The command for simulating the test data base is :
Go to sim_modelsim folder and type the below command
> perl sim_modelsim.pl run all

The command for simulating single testcase
> perl sim_modelsim.pl run testtree/initialization/init_bus_width_mem

To Compile & Simulate in VCS:
The following Files are needed for Compiling the database.
1. sim_vcs.pl
2. list
3. test_array.txt

Go to sim_vcs folder and type the below commands
The command for simulating the test data base is :
> perl sim_vcs.pl run all
The command for simulating single testcase
> perl sim_vcs.pl run testtree/initialization/init_bus_width_mem
The script performs the following actions after simulation of each testcase
1. Checks whether end of file reached in the cmd files.

2. Compares the dat files against its .g_d file.

   a) master_rcv.dat Vs master_rcv.g_d

   b) target_rcv.dat Vs target_rcv.g_d

   c) device_rcv.dat Vs device_rcv.g_d

3. Checks for the error messages in the transcript which is saved as 'sim.log'.

After simulation the results will be displayed in Standard Output. In each folder an error.log file will be there, to know the errors.

When we can say a testcase is passed ?

1. NO ERR DISPLAYED should come in sim.log
2. The expected data is in golden dump files *.g_d. This is compared with the corresponding files after simulation. When both the files are same then we can say that testcase is passed.

   a) master_rcv.dat Vs master_rcv.g_d

   b) target_rcv.dat Vs target_rcv.g_d

   c) device_rcv.dat Vs device_rcv.g_d

Simulation :

Regression:

Regression:

~~~~~~~~~~~

Run the perl script sim_vcs.pl or sim_modelsim.pl for regression type

perl sim_vcs.pl run all

perl sim_modelsim.pl run all

Then see the report under testtree/report.txt

if there are any failing testcases then go to that particular folder and view error.log file to find why the testcase is failing

Note : While simulating each testcase the command files (device.v, master.v, target.v) shall be copied in the script directory.

i.e device.v, master.v and target.v are copied to scripts directory

*So during regression for each testcase the *.v files are copied in to the scripts directory, then compiled and then finally simulated.*

Simulating Single Testcase using scripts:

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

perl sim_vcs.pl run <dir> OR perl sim_modelsim.pl run <dir>

e.g to run testtree/initialization/init_bus_width_mem

type

perl sim_vcs.pl run testtree/initialization/init_bus_width_mem

perl sim_modelsim.pl run testtree/initialization/init_bus_width_mem

15. Abbreviations and Terms

AHB - Advanced HIgh-performance Bus
Block Gap **-** period between blocks of data
Block - a number of bytes, basic data transfer unit
Busy - Busy signal that SD card drives on DAT[0] line
CCCR - Card Common Control Register
CID - Card Identification number register
Clr - Clear
CMD - command line or SD Bus command
CMD_wo_DAT - Commands without using DAT line
CRC - Cyclic Redundancy Check
CSD - Card Specific Data Register
DAT - data line
DMA - Direct Memory Access
HC - Host Controller
HD - Host Driver
HW - Hardware
Int - Interrupt
LED - Light Emitting Diode
OCR - Operations Condition Register
Resume - Restart Suspended Function
RCA - Relative Card Address Register
SDCD# - a signal, which is active low, for detecting SD Card
SDCLK - a clock for supplying to SD card
SDWP - a signal, which is active high for detecting SD card to be protected writing
Suspend - Stop multiple transaction

•