

FPS-TECH SD/SDIO/MMC HOST CONTROLLER DATASHEET

IP VERSION: 1.1

TABLE OF CONTENTS

Core Overview	3
Supported protocols	3
SD Host Controller Block Diagram	4
Signal Descriptions.....	5
Functional Description.....	6
Command and Response Decoder.....	6
Data Read and Write Drivers	7
Read and Write Avalon Bus Drivers	7
Profiling Registers	8
SDIO Interrupt Decoder	9
Clock Control.....	9
Avalon Slave Registers	9
SD Host Register Map	10
Offset 0: CTL_STAT.....	11
Offset 1: CLK_CTL.....	12
Offsets 2 to 5: CMD_ARG[0:3]	12
Offset 6: DMA_BASE	12
Offset 7: XFER_CTL.....	13
Offset 8: MISC.....	13
Offset 8: PROF_CNT[31:0]	14
Offset 9: PROF_CNT[63:32]	14
Offsets 10 to 14: RESERVED.....	14
Offset 15: VERSION_INFO.....	14
Resource Usage	15
Document revision history	15

CORE OVERVIEW

The FPS-Tech SD/SDIO/MMC controller (SD host for short) brings full SD/SDIO/MMC 2.0 host support to Altera SOPC-based systems, including uClinux based embedded systems. The host can be used to interface Altera FPGAs to Secure Digital (SD), Secure Digital I/O (SDIO) and Multi-media card (MMC) device targets such as Flash memory cards, or SDIO-based Wireless interfaces.

The host controller package provides all of the necessary components to implement SD Host functionality into your target system with minimal effort. The IP for the host is packaged into an encrypted Altera Avalon-MM (Memory-mapped) component that can be placed into any SOPC-based system. Sample software and an API are also provided to demonstrate basic functionality of the core. These can be used for systems that require fine-grained or low-latency access to the SD Host for the maximum possible performance. They can also be used as a building block for a custom device driver. Alternatively, systems that are based around uClinux can use the open-source MMC driver that can be obtained from the development uClinux branch (see <http://jotspot.nioswiki.com>).

The SD Host features a command and response encoder/decoder that is used to send standard SD commands to a device, and optionally receive a response. A hardware-based CRC7 engine is used for CRC generation and verification so that software resources are not tied up with intensive calculations. Both the standard 48-bit response, and extended 136-bit response types are supported.

Data read and write drivers inside the SD Host allow for both byte and block-oriented transfers to occur over the DAT signals. The SD Host supports both 1 and 4-bit Data modes, along with hardware CRC16 calculations for generation and verification. In addition to supporting basic data transfers, the SD Host contains a powerful 32-bit DMA engine, with a deep read/write FIFO, that can be used to free up the processor during data transfers. To accommodate for memory systems that can exert significant backpressure on the bus (DDR-memory for example), different block pre-fetching schemes can be selected to ensure stable data flow to and from the device. The host also has the ability to shut-off the clock to the device in the event that the internal read FIFO reaches the overflow condition.

Several interrupts are supported to again ensure maximum driver performance by eliminating polling requirements. In addition to Host-generated interrupts, the SD Host supports SDIO device interrupted that are sent over the SDIO interface. The SDIO interrupt allows for an SDIO-compliant device to interrupt the host using special DAT signaling. In addition to the CMD and DAT interface supported by the SD Host, the controller also supports external Card-detect (CD), Write-protect (WP) and BUSY signals. The card-detect signal goes to an internal de-bounce filter that is capable of generating interrupts upon card insertion and removal. The write-protect signal can be read from software to determine the write-protect status of the inserted card. The BUSY output signal indicates when the Host is in the middle of a transaction. This can be useful for an LED activity indicator for example.

SUPPORTED PROTOCOLS

The FPS-Tech SD Host control supports the following major memory-card physical layer interfaces:

- Secure Digital 1.1 and 2.0 (also known as Secure Digital High Capacity – SDHC)
- Secure Digital Input Output (SDIO 2.0)
- MultiMedia Card 4.3

SD HOST CONTROLLER BLOCK DIAGRAM

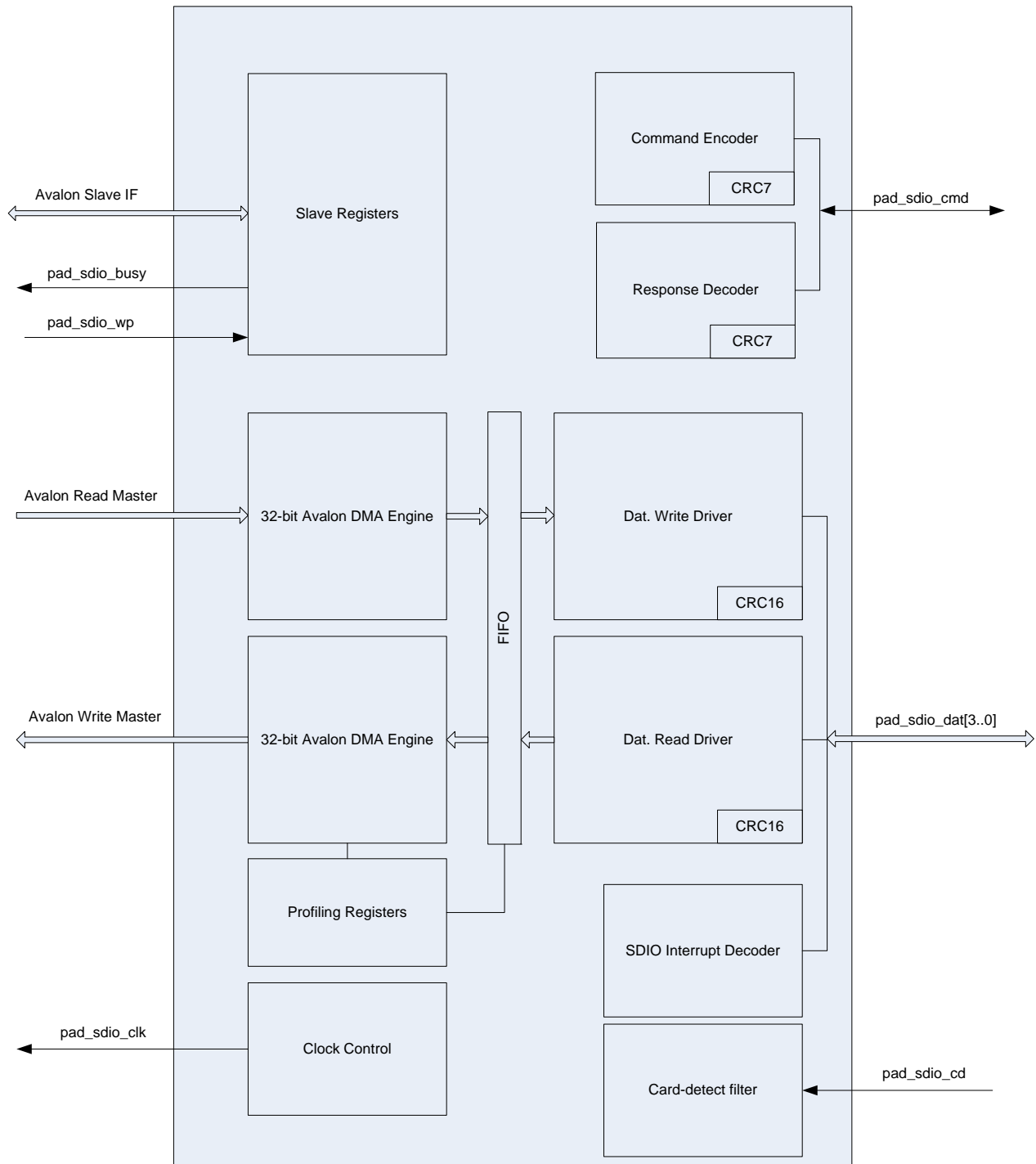


FIGURE 1: HOST CONTROLLER BLOCK DIAGRAM

SIGNAL DESCRIPTIONS

Signal	Direction	Width	Description
Global Signals			
clk_in	Input	1	System Clock for the Host IP
reset_in	Input	1	Active High synchronous reset
Avalon Slave Interface			
sl_address	Input	32	Avalon slave Address signal
sl_write	Input	1	Avalon slave write signal, active high
sl_read	Input	1	Avalon slave read signal, active high
sl_readdata	Output	32	Avalon slave readdata bus
sl_writedata	Input	32	Avalon slave writedata bus
sl_irq	Output	1	Avalon slave IRQ, active high
Avalon Read-Master Interface			
msr_address	Output	32	Avalon master address output
msr_read	Output	1	Avalon master read signal, active high
msr_readdata	Input	32	Avalon master readdata bus
msr_be	Output	4	Avalon master byte-enable signal, active high
msr_waitrequest	Input	1	Avalon master waitrequest input, active high
Avalon Write-Master Interface			
msw_address	Output	32	Avalon master address output
msw_write	Output	1	Avalon master write signal, active high
msw_writedata	Output	32	Avalon master writedata bus
msw_be	Output	4	Avalon master byte-enable signal, active high
msw_waitrequest	Input	1	Avalon master waitrequest input, active high
SD/SDIO/MMC Device Interface			
pad_sdio_clk	Tri-state	1	SD Device clock
pad_sdio_dat	Bi-directional	4	SD Device DAT bus. Tie pad_sdio_dat[3:1] high for 1-bit mode.
pad_sdio_cmd	Bi-directional	1	SD Device CMD.
pad_sdio_busy	Output	1	Host busy output, active high. Can be connected to LED for busy/progress indication
pad_sdio_wp	Input	1	Device WP input. WP high indicates that device has WP switch on. Usually connected to SD Socket. Tie low if signal not available.
pad_sdio_cd	Input	1	Device CD input. CD high indicates that card is present in socket. Usually connected to SD Socket. Tie high if signal not available.

TABLE 1: SIGNAL DESCRIPTIONS

COMMAND AND RESPONSE DECODER

The CMD/Response decoder handles sending commands on the SD CMD line and receiving responses from the device on the CMD line as well. The CMD Driver will translate the CMD_IDX and CMD_ARG0 registers into a 48-bit stream, along with a CRC7 calculation included. The XFER_START register bit will set the CMD Driver process in motion. The CMD_ARG[3:1] registers have no effect on sending the command.

The RESP_CODE bits determine which response type is expected on the CMD line. Responses can only be received after the command is set out. Currently, three types of response codes are supported:

- Type 0: No response expected
- Type 1: 48-bit response expected
- Type 2: 136-bit response expected.

If a response is expected, the response decoder will verify the Start, Direction, CRC7 and end-bits in the CRC Frame. If the RESP_NOCRC flag is set, the response decoder will not attempt to verify the CRC. This is useful for responses that do not have CRCs appended.

The 4 CMD_ARG (CMD_ARG[3:0]) registers will be loaded with the response bits. These 4 registers effectively form a 128-bit shift register that shifts in the response (MSB first). CMD_ARG0 is the LSB of the 128-bit register, and CMD_ARG3 is the MSB. For a 48-bit response (Type 1), the useful payload is the 6-bit command index, and 32-bit response argument. Therefore, only 38 bits are shifted into the 128-bit register. CMD_ARG0 will have the 32-bit response argument, and the lower 6-bits in CMD_ARG1 will be the command index. For a 136-bit response (Type 2), the useful payload is the 127-bit register contents. To preserve bit alignment, the LSB (end-bit which is always '1') is also shifted into the LSB of CMD_ARG0. The 127-bit register contents + end-bit form the 128-bit register contents. Again, CMD_ARG0 is the LSB of this register.

Interrupt flags in the CTLSTAT register will reflect the status of the transaction when both the command encoder and response decoder have completed. The XFER_IF bit will indicate when the transfer is done, if no data blocks are to be sent/received. If data blocks are required to be sent/received, XFER_IF will go high when the data portion of the transfer has completed as well. If the response decoder times out waiting for the start bit (from the device), the TIMEOUT_IF flag will be set. If the CRC7 calculation on the response packet fails, the CRCERR_IF flag will be set. Again, CRC calculations are not performed if the RESP_NOCRC bit is set. If the end-bit is not detected successfully on the response packet, the FRMERR_IF flag will be set.

DATA READ AND WRITE DRIVERS

The DAT Read/write drivers are responsible for handling data transactions on the data lines. The DAT drivers can operate in 1 or 4-bit modes, decided by the DAT_WIDTH bit. Depending on the options purchased, the host may support 1-bit mode only, or both 1 and 4-bit DAT modes. This is reflected by the HOST_4BIT flag. Each DAT line has a CRC16 block that is used to calculate the CRC16 checksum on both read/write data operations. The DAT_RWn bit determines whether the DAT Read block or DAT Write block is in control of the transfer. For a DAT Write operation, the DAT Bytes/Blocks will be transmitted when the Response decoder has completed with the reception of the response from the device. For a DAT Read operation, the DAT Read driver will be ready to receive the start bits on the DAT line as soon as the Command has completed from the CMD Driver.

The DAT transfers are setup according to the BYTE_COUNT and BLOCK_COUNT bits. The number of blocks transmitted/received is determined by the BLOCK_COUNT register. A BLOCK_COUNT of '0' indicates one block. The total number of bytes transmitted can be determined by the following formula:

$$N_{bytes} = (BYTE_COUNT) \times (BLOCK_COUNT + 1)$$

Each block transmitted or received has a start, end and CRC16 bits. For read transactions, if the start bit is not detected after a long period, the TIMEOUT_IF flag is set. The DAT Read module implements a 24-bit counter for detecting timeouts, meaning a maximum block read access time of 2^{24} system clock cycles is allowed. During the read transaction, if any blocks' CRC16 doesn't match the calculated CRC16, the CRCERR_IF flag will be set. If any blocks' end-bit is not detected correctly, the FRMERR_IF bit will be set. For a block write operation, the device will respond with a 3-bit code that indicates whether it matched the CRC16 correctly. If the device responds with a negative CRC check code after any block during a block write procedure, the CRCERR_IF flag will be set. Lastly, the target device may stall the interface after a block write transaction by holding the DAT[0] line low after any block write. The host controller will detect this and wait for the device to become free before writing any further blocks that may be required.

READ AND WRITE AVALON BUS DRIVERS

The Avalon bus drivers have two major purposes: 1) Transfer Data to and from the Avalon bus to the Read/Write FIFO, and 2) Try to ensure that there are no wait states on the SD interface due to bus latency issues.

For Write transactions, the Avalon Read master is activated. For a Write transaction, the Read Avalon Master will start to pull data from the Avalon bus as soon as the CMD starts to get shifted out (pre-fetching). This will ensure that by the time the CMD has been sent, data will be available in the FIFO for the DAT Write driver to read from. For high latency memory systems, such as DDR-based systems, this pre-fetch time may not be enough. If the DAT write module tries to read a word from the FIFO and the FIFO is empty, the FIFO_UNDERRUN_IF flag will be set. This indicates that despite the initial pre-fetch period, the memory system was not able to keep up with the SD interface speed. In this situation, it is advised that the BLK_PREFETCH bit be set. When this bit is set, the Avalon driver will ensure that a full block of data is always present in the FIFO before proceeding with the actual write. This applies for multi-block transfers as well. For users with DDR-based systems, it is highly recommended that the BLK_PREFETCH be used. For users wishing to maximize interface throughput, or who are using SRAM-based memory systems, the BLK_PREFETCH bit should not be used. It is still advised that the FIFO_UNDERRUN_IF flag be checked to ensure that the memory system can keep up.

For a read transaction, since data will not be available in the FIFO until the DAT phase is active, no pre-fetching can be performed. If the DAT Read module attempts to receive a word into the FIFO, and the FIFO is full, the FIFO_OVERRUN_IF flag will be set. This indicates that the memory system was unable to get the data out of the FIFO fast enough. Data bytes will be discarded until the FIFO becomes empty again. Clock-stopping is enabled by default to prevent the FIFO Overrun condition (see Clock Control section).

To maximize the efficiency of any bus transactions for both the read and write Avalon modules, 32-bit transactions are used wherever possible. The base address of DMA transfers does not have to occur on word-aligned boundaries for this to happen.

The DMA_BASE register sets the base address of the read or write operations.

PROFILING REGISTERS

By default, the SD Host is built with profiling registers disabled since they can utilize an additional 100-200 flops. Profiling can be enabled by setting the PROF_EN parameter to '1' in SOPC Builder. The PROF_EN bit indicates whether the core was built with Profiling support enabled or disabled.

The purpose of the profiling registers is to analyze run-time in-system performance of the SD Host, and to identify where any bottlenecks exist. The profiling registers are intended to analyze time spent in wait states between the SD Host and the Avalon bus, and between the SD Host and Device. Only commands that require data bytes/blocks to be transferred will increment the profiling counters. Commands that don't require data bytes/blocks to be transferred do not require any interaction with the Avalon bus, and therefore have fixed transfer times.

The profiling registers contain a set of 3 64-bit counters, PROF_RESET bit to reset the counters and PROF_CNT_SEL bits to select which counter is mapped to the Avalon slave registers. The counters will increment off the system clock whenever their event is active. Descriptions of the counters are given below:

Counter	DAT Read/Write	Operation / Event
XFER_LEN	Read	Counter will increment for period that the complete DAT transaction is active. This is the period from when XFER_START is asserted to when XFER_IF is asserted. This period encompasses all delays encountered.
	Write	Same operation as Read.
BUS_WAIT_LEN <i>(delay from Avalon bus)</i>	Read	Counter will increment for the period that starts when the device has completed transmitting the last data block (+CRC), and ends when the SD Host has complete writing all data from the FIFO to the Avalon interface.
	Write	Counter will increment for the period that starts when the SD Host starts to pre-fetch data from the Avalon interface, and ends when a complete block has been fetched from memory and Data transmission begins. This applies when Block pre-fetching is enabled.
BUSY_WAIT_LEN <i>(delay from device)</i>	Read	Counter will increment for the period that starts when the end-bit of the CMD has been transmitted, and ends when the start-bit on the Data interface is transmitted from the device. Counter will also be incremented whenever waiting for the start-bit from the device on subsequent blocks.
	Write	Counter will increment for the period that the device asserts the BUSY signal after each data block has been written from the host.

TABLE 2: PROFILING COUNTERS

SDIO INTERRUPT DECODER

The SDIO interrupt decoder is always active. In the event that the device does not support SDIO interrupts, the DEV_IF flag should be ignored. The DEV_IF flag reflects the interrupt state of the SDIO interface in real-time. The only way this flag can be cleared is if the SDIO Device stops asserting the interrupt (by the host clearing it). The interrupt decoder will automatically decode 1 or 4-bit mode interrupts based on DAT_WIDTH bit.

CLOCK CONTROL

The clock control block is responsible for generating the output clock for the SD Device. It can be shut on and off by using the CLK_EN bit, and the speed can be changed during run-time. Since the clock is generated using synchronous logic internally, no external PLL is required to generate the SD clock. Due to limitations of the synchronous divider, the maximum frequency of the output clock is limited to the cores clock frequency / 4.

By default, the CLKSTOP_DIS bit is set to '0' which means the host will shut-off the clock during a DAT READ transaction if it detects the FIFO is about to overflow. This can occur if the memory sub-system cannot get data out of the FIFO fast enough to keep up with the physical interface speed. If the user wishes to disable the clock-stop operation, set the CLKSTOP_DIS bit to '1'. Setting this to '1' means the FIFO has the potential to overflow, and the FIFO_OVERRUN_IF bit should be checked.

AVALON SLAVE REGISTERS

All of the registers in the SD Host controller are accessed through the connected Avalon Master(s). The Interrupt output is also linked to the Avalon Slave module. A Slave interrupt is generated when both an Interrupt Enable and the corresponding Interrupt Flag is set in the CTLSTAT register. All interrupt flags will be cleared on any write to the CTLSTAT register.

SD HOST REGISTER MAP

The following pages document the several registers present in the SD Host controller core. All bits marked as RES (reserved) should be written as '0'. All the default values of registers are '0' unless specified otherwise.

Bit-types in registers are listed below in Table 3.

Bit Type	Definition
RO	Read only
RW	Read/write
WC	Write to clear
WO	Write-only (read always 0)

TABLE 3: REGISTER BIT-TYPES

OFFSET 0: CTL_STAT

[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
CRCERR_IF	FRMERR_IF	XFER_IF	DEV_IF	CD_IF	WP	CD	BUSY
RO,WC	RO,WC	RO,WC	RO	RO,WC	RO	RO	RO

[14]	[13]	[12:11]	[10]	[9]	[8]
HOST_4BIT	PROF_EN	RES	FIFO_UNDERRUN_IF	FIFO_OVERRUN_IF	TIMEOUT_IF
RO	RO	RO	RO,WC	RO,WC	RO,WC

[31]	[30:20]	[20]	[19]	[18]	[17]	[16]
SOFT_RST	RES	BLK_PREFETCH	ERR_IE	XFER_IE	DEV_IE	CD_IE
RW	RO	RW	RW	RW	RW	RW

Bit Descriptions:

- **BUSY**: Indicates that the SD interface is currently transmitting or receiving a response or data packet. Do not issue any new commands to the interface while it is busy.
- **CD**: Card-detect bit. This reflects the state of the CD signal on the slot.
- **WP**: Write-protect bit. This reflects the state of the WP signal on the slot.
- **CD_IF**: Interrupt generated from CD edge-trigger. Check state of CD to determine whether card was inserted or removed.
- **DEV_IF**: Interrupt generated from SDIO Device (over SDIO interface). If the device is not an SDIO device, or it doesn't support interrupts, this flag should be ignored by the host.
- **XFER_IF**: Interrupt generated from command completion
- **FRMERR_IF**: Interrupt generated from Response decoder. This bit indicates that the end-bit was not detected successfully on the CMD line.
- **CRCERR_IF**: Interrupt generated from Response decoder. This bit indicates that the CRC calculated on the Response packet did not match what was sent.
- **TIMEOUT_IF**: Interrupt generated from Response decoder OR Data receiver. This bit indicates that the process timed out while waiting for start-bit from device on either the CMD line or Data lines.
- **FIFO_OVERRUN_IF/FIFO_UNDERRUN_IF**: These are generated from the RDDAT/WRDAT modules respectively. They indicate that the internal FIFOs have been overrun or underrun.
- **PROF_EN**: Indicates if the core was built with Profiling registers enabled.
- **HOST_4BIT**: Indicates if the core was built with 4-bit DAT drivers, or just 1-bit.
- **CD_IE**: Card-detect interrupt enable
- **DEV_IE**: SDIO Device interrupt enable.
- **XFER_IE**: Transfer completion interrupt enable.
- **ERR_IE**: Error condition interrupt enable. This will enable allow both the CRCERR_IF and FRMERR_IF flags to generate an interrupt.
- **BLK_PREFETCH**: When this bit is set, the WRDAT DMA engine will prefetch an entire block before starting DAT transmission. This is to ensure that no FIFO underruns will take place. Try setting this bit if the FIFO_UNDERRUN_IF flag becomes set.
- **SOFT_RST**: Set this bit to perform a soft-reset on the SD Host Core. This bit will automatically clear after one-cycle.

OFFSET 1: CLK_CTL

[31]	[30]	[29:24]	[23:0]
CLK_EN	CLKSTOP_DIS	RES	CLK_DIV
RW	RW	RO	RW

Bit Descriptions

- **CLK_DIV:** Determines the output frequency of the SD Clock. The **minimum value** for this register is '1'. A value of '0' will result in erroneous behavior. $F_{sdio_{clk}} = \frac{SYSCLK}{(CLK_{DIV}+1) \times 2}$
- **CLKSTOP_DIS:** By default, clock will be stopped during a DAT READ operation if FIFO will overflow. Setting this bit will disable this function.

OFFSETS 2 TO 5: CMD_ARG[0:3]

[31:0]
CMD_ARGx
RW

Bit Descriptions:

- **CMD_ARGx:** CMD_ARG0 is used to send the 32-bit argument of the SD Command on the CMD line (once transfer is initiated). If a response is requested (indicated by the RESP_CODE bits), the argument of the response will be loaded in these registers once the transfer has completed.

OFFSET 6: DMA_BASE

[31:0]
DMA_BASE
RW

Bit Descriptions:

- **DMA_BASE:** Register that sets the base address for a block read or block write. On a SD read operation, this determines the start address for the Avalon write module to write incoming bytes into. For an SD write operation, this determines the start address for the Avalon read module to read bytes from.

OFFSET 7: XFER_CTL

[31:22]	[21:12]	[11]	[10]	[9:8]	[7]	[6:1]	[0]
BLOCK_COUNT	BYTE_COUNT	RESP_NOCRC	DAT_RWn	RESP_CODE	DAT_WIDTH	CMD_IDX	XFER_START
RW	RW	RW	RW	RW	RW	RW	WO

Bit Descriptions:

- **XFER_START**: Initiate the command and data transfer as determined by the XFER_CTL and CMD_ARG registers. This bit will self-clear within a few cycles, so it should not be used for polling for completion.
- **CMD_IDX**: 6-bit SD Command to send on the CMD line.
- **DAT_WIDTH**: DAT_WIDTH of '1' means that 4-bit SD mode will be used. DAT_WIDTH of '0' means 1-bit SD mode will be used. This bit should not change in the middle of transactions.
- **RESP_CODE**: 2-bit code to determine what response type to expect on the CMD line. Currently supported response types are:
 - **0**: No response expected.
 - **1**: 48-bit response expected.
 - **2**: 136-bit response expected
 - **Other**: Reserved, do not use.
- **DAT_RWn**: Determines direction of data transfer. DAT_RWn of '1' indicates a read from the SD Device.
- **RESP_NOCRC**: When set, the response decoder will not attempt to verify the CRC on the CMD line. This should be used for responses that do not append a CRC7 at the end.
- **BYTE_COUNT**: Determines how many data bytes to transfer. For commands that do not require data to be sent/received, this should be set to '0'. If BLOCK_COUNT is used, this will determine how many bytes are in a block. Do not change the value of this register when a transfer is in progress.
- **BLOCK_COUNT**: Determines how many blocks to transfer + 1. For example, a Block count of '0' would signify 1 block of BYTE_COUNT bytes to transfer.

OFFSET 8: MISC

[31:4]	[3:1]	[0]
RES	PROF_CNT_SEL	PROF_RESET
RO	RW	WO

Bit Descriptions:

- **PROF_RESET**: This will reset the 64-bit profiling counters when a '1' is written. Bit will self-clear. This bit does nothing if profiling is disabled (PROF_EN = 0).
- **PROF_CNT_SEL**: Selects between the available 64-bit profiling counters. Refer to the Profiling Counters section for descriptions on the available counters. These bits have no effect if PROF_EN is '0'.
 - **0**: XFER_LEN profile counter
 - **1**: BUSY_WAIT_LEN profile counter
 - **2**: BUS_WAIT_LEN profile counter

OFFSET 8: PROF_CNT[31:0]

[31:0]
PROF_CNT[31:0]
RO

- PROF_CNT[31:0]: Register contains the lower 32-bits of the currently selected profiling counter. The selected counter is determined by the PROF_CNT_SEL bits. If PROF_EN = 0, this register will always read 0.

OFFSET 9: PROF_CNT[63:32]

[31:0]
PROF_CNT[63:32]
RO

- PROF_CNT[63:32]: Register contains the upper 32-bits of the currently selected profiling counter. The selected counter is determined by the PROF_CNT_SEL bits. If PROF_EN = 0, this register will always read 0.

OFFSETS 10 TO 14: RESERVED

- These registers are reserved for internal usage and should not be written to.

OFFSET 15: VERSION_INFO

[31:24]	[23:16]	[15:0]
MAJOR_REV	MINOR_REV	MAGIC_COOKIE
RO	RO	RO

Bit Descriptions:

- MAGIC_COOKIE: This value will always be 0xBEEF.
- MINOR_REV: This represents the minor revision number of the core.
- MAJOR_REV: This represents the major revision number of the core.

RESOURCE USAGE

The SD Host controller IP was compiled and fitted using the Altera Cyclone 3 NIOS II Evaluation platform (NEEK). The following represents the core usage with no optimizations applied on the fitter level.

Logic Elements	Memory Bits
1,700	8,192 (1 M9K blocks)

TABLE 4: RESOURCE USAGE INFORMATION

DOCUMENT REVISION HISTORY

Table 4 outlines the revision history for this document. The **current document revision is 1.3**.

Date / Revision	Changes Made	Summary of Changes
6/28/2008 V1.3	<ul style="list-style-type: none"> Changed IP Rev. to 1.1 Updated block-diagram for shared FIFO and profiling registers Updated resource usage for IP 1.1 Added Profiling Registers information section 	IP Revision 1.1 Updates
6/26/2008 V1.2	<ul style="list-style-type: none"> Updated pinout information and maximum clock frequency in CLK_DIV description. Removed draft marking. Correction in BLOCK_COUNT description 	
6/23/2008 v1.1	<ul style="list-style-type: none"> Added information about supported PHY protocols and SDHC support 	SDHC, SD 2.0 support notes
5/1/2008 V1.0	<ul style="list-style-type: none"> Initial Release 	

TABLE 5: RESOURCE USAGE INFORMATION