cadence®

SD 6.0 UHS-I CQ / eMMC 5.1 Host Controller IP

for

SD devices up to 6.1

SDIO devices up to 4.1

and eMMC devices up to 5.1

with SD/eMMC interface

Part Number: IP6061

IP Rev: R602- Doc Rev: 1.6

User Guide

2022-04-14

Table of Contents

List of Tables

List of Figures

About this Document

This User Guide describes the interface and operational details for integration of the SD 6.0 UHS-I CQ / eMMC 5.1 Host Controller IP into a target design. The intended audience for this document is design engineers.

This document is organized as follows:

References and Related Documents

This guide references the following documents:

Table 1. References and Related Documents

Reference	Version/Description
SD Specifications - Part A2 SD Host Controller Standard Specification	version 6.00, 2017-12-12, SD Card Association [1]
SD Specifications - Part 1 Physical Layer Specification	version 6.10, 2018-03-18, SD Card Association
SD Specifications - Part E1 SDIO Specification	version 4.10, 2012-02-20, SD Card Association
Embedded Multi-Media Card (eMMC) Electrical Standard	version 5.1, July 2014, JEDEC
SD/eMMC Host Controller — Integration Manual	version 1.10, January 2019, Cadence Design Systems
Combo DLL PHY IP — User Guide	version 1.12, January 2019, Cadence Design Systems

[1] Title page indicates that this controller support device compatible up to 6.00. However, it does not support optional feature — UHS-II/UHS-III Interface.

Acronyms and Abbreviations

The following acronyms and abbreviations (2) are used in this guide:

Table 2. Acronyms and Abbreviations

Term	Definition
SD	Secure Digital card
SDSC	Standard Capacity SD card
SDHC	High Capacity SD card
SDXC	eXtended Capacity SD card
SDUC	Ultra Capacity SD card
SDIO	Secure Digital Input/Output card
MMC	MultiMediaCard
eMMC	Embedded MultiMediaCard
FIFO	First In First Out memory
SDMA	Simple Direct Memory Access
ADMA	Advanced Direct Memory Access
DMA	Direct Memory Access
CRC	Cyclic Redundancy Check
DUT	Design Under Test
UHS-I	Ultra High Speed Interface Phase I
UHS-II	Ultra High Speed Interface Phase II

Terminology

The following terminologies (3) are used in this guide:

Table 3. Terminology

Term	Definition
Data Transfer Terminology	
CPU	This transfer method uses direct access to the internal buffer through register interface. The buffer is mapped into address space as SRS08. The SRS08 works as a queue to internal buffer. The DMA engines in such transaction are disabled.
SDMA	Simple DMA — The (simple/single-operation) DMA algorithm introduced and defined in SD Host Controller Standard Specification Version 1.00
ADMA	Advanced DMA — The (advanced) DMA algorithm introduced and defined in SD Host Controller Specification Version 2.00 that provides the data transfers between system memory and the SD card without interruption of CPU operation. There are two types of ADMA: ADMA1 and ADMA2. Wherever the term 'ADMA' is used in this document, it applies to ADMA2 modes.
ADMA1	Advanced DMA mode 1 — The DMA algorithms defined in SD Host Controller Specification Version 2.00 — Appendix C. The ADMA1 can support transfer of data only if buffer is aligned to 4kB boundary in system memory. Since the SD Host specifications version 3.00 (and later) denotes ADMA1 mode as obsolete, therefore it is not supported by this version of SD Host Controller.
ADMA2	Advanced DMA mode 2 — The DMA algorithms defined as recommended ADMA algorithm in SD Host Controller Specification Version 2.00. The ADMA2 can support data transfer of any location and any size (removes the 4KB aligning restriction of the ADMA1 mode). The ADMA2 can operate in 32-bit or 64-bit addressing modes.

Term	Definition
DMA	Wherever the term DMA is used in this document, it applies to both, SDMA and ADMA2 modes.
SPRAM	Single-Port RAM — The internal FIFO module requires this memory type for data buffering.
Register Access Mode Terminology	
RO	Read-Only register. Write operation to this register is ignored. The reset does not alter the value.
ROC	Read-Only register. Content of this register is initialized to zero at the reset. Write operation is ignored.
RW	Read-Write register. It can be either set (’1’) or cleared (’0’) by software to the desired state.
RW1C	Read-Only register. This register is set to ’1’ by hardware on particular events. Writing ’1’ clears the register. Writing ’0’ has no effect.
RWAC	Read-Write, automatic clear register. When writing ’1’ to RWAC, the host is requested to perform given operation. When the requested operation is completed, this register is automatically cleared. Writing ’0’ to RWAC bits has no effect.
HwInit	Hardware Initialized register. It is read-only and a desired values is defined (hardcoded) in RTL. Write operation is ignored.
WO	Write-Only register. This is used to force the error status register setting. Each WO-type bit has a corresponding error status bit. Reading these bits returns 0.
P	The content of the register is write-protected by the SD Host Controller logic when a Command Inhibit (DAT) is active (set to 1) in the SRS09 register.
Text Formatting Conventions	
italics	Pin names are in italics inside the text. For example, *sdmclk*.

1. Introduction

1.1 Overview

1.1.1 Purpose of Document

This document contains the User Specification of the SD Host Controller core compatible with SD Host Controller Standard specification version 6.00. The Host Controller serves devices compatible with SD Physical Layer version 6.10 (SD, SDHC, SDXC, SDUC, SDIO) , SDIO version 4.10 Embedded Multi-Media Card Electrical Standard version 5.1. This Host Controller does not support the UHS-II/UHS-III Interface. The document provides the core architecture description, register set description and sub-modules specification.

1.2 Overview of the Controller IP

Figure 1 is a basic overview of system containing the Host Controller IP, showing the typical requirements to connect to an external SD/eMMC device and operate the software stacks.

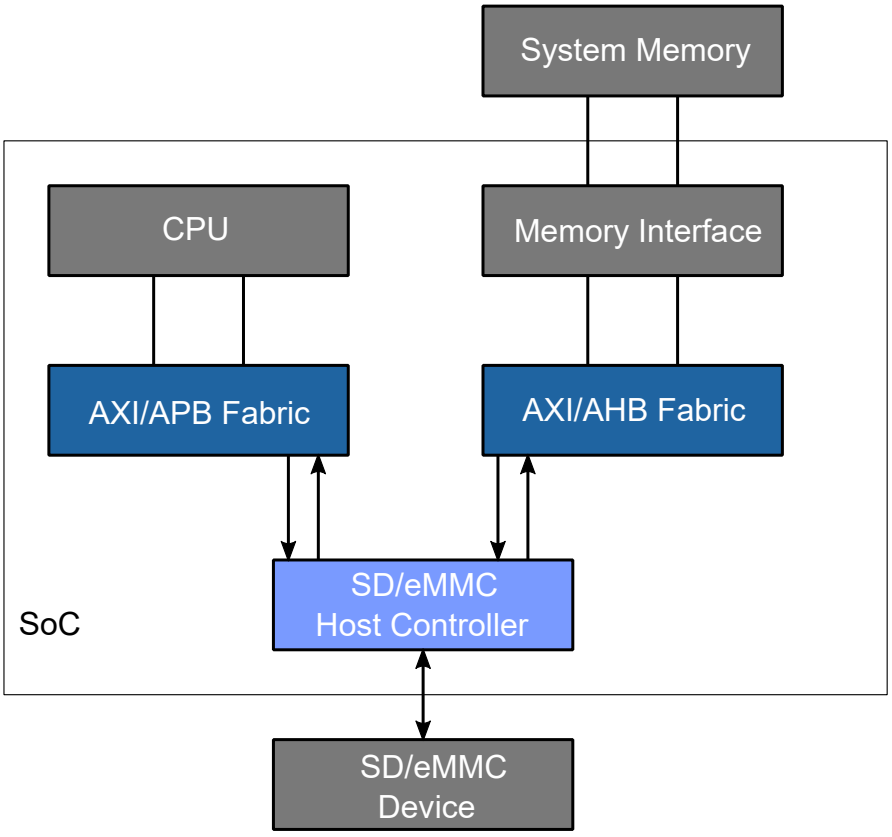

Figure 1. Example System Level Block Diagram

2. IP Configuration

The Controller ip is delivered with the configuration specified in table 4

Table 4. IP Configuration

Feature	Specification
Compatibility	• SD Host Controller Standard Specification version 6.00 • SD Physical Layer Specification version 6.10 • SDIO Specification version 4.10 • eMMC Specification 5.1
Standard Register Set	• Compatible with SD Host Controller Standard Specification version 6.00 • Independent Interrupt and Wakeup outputs
Integrated DMA (SDMA/ADMA) engines	• Simple DMA (SDMA) Controller — single operation DMA • Advanced DMA Controller (ADMA) — scatter-gather DMA • Programmable burst length for DMA transfers
Data buffering	• Configurable buffer size • Supports SPRAM
SD/UHS-I interface	• programmable bus width • Read-Wait mechanism • tuning mechanism for UHS-I SDR104 • signaling voltage switch support
eMMC interface	• programmable bus width • hardware implementation of eMMC boot • tuning mechanism for HS200 • HS400 supported • HS400 with enhanced strobe supported • Command Queuing
Low power features	• Master SD card side clock can be switched off • Card clock can be switched off independently
System Interface options	• AXI4-Lite or APB slave (register) interface • AXI3 or AHB-Lite master (DMA) interface
UHS-II/UHS-III interface	No support for UHS-II/UHS-III Interface

3. Architecture Overview

3.1 Architectural Block Diagram

Figure 2 depicts an overview of the SD Host Controller and its interfaces. The main components of the controller are:

- **BIU** — Bus Interface Unit that integrates the following components:
  - **HRS** — Host Register Set, implementation of additional registers that are not defined by the SD Host Controller specification (released by SD Association) and are specific to Cadence implementation of the SD Host Controller
  - **SRS** — Slot Register Set, implementation of registers defined by the SD Host Controller Standard Specification
  - **CRS** — Common Register Set, implementation of registers in the Common Register Area defined by SD Host Controller Standard Specification.
  - **CQRS** — Command Queuing Register Set, implementation of registers defined in the section B.4. CQE Registers of the eMMC Standard 5.1. The registers are located in the CQ module but are accessible as any other register.
  - **DMA** — unit responsible for the data transfer between the external memory location and the controller buffers. It works in one of two modes, i.e. SDMA or ADMA2. The SDMA is simple and single operation DMA engine. The ADMA is scatter-gather DMA engine which uses description table. The DMA has the OCP master internal interface to access system memory.
  - **ADDRDEC** — Address Decoder for Slave interface transactions.
  - **CTRL** — Command execution control module
  - **DCTRL** — Data transfer control module includes SDMA, ADMA2 and ADMA3 engines
  - **SETUPMECH** — DMA-based auto-reconfiguration engine
  - **FIN** — FIFO Input Multiplexer/Arbiter, facilitate access to the internal FIFO. Two components have access to the FIFO — Buffer Register (SRS08) and DMA engine.
  - **DMA_ARB** — DMA Arbiter, unit shares access to **DMA** between **DCTRL** and **CQ**
  - **TUNE** — UHS-I SDR104 Tune Control module, gathers result from 40 iterations to calculate the center of the data valid window
  - **BOOT** — eMMC boot control module, if enabled after the power up, it transfers configured data block from the eMMC memory to system memory using SDMA engine.
  - **CQ Engine** — Command Queuing Engine, control module which processes commands requested through CQ registers.
    - **TDQUE** — Task Doorbell Queue, list of requested tasks which waits to be sent to the device
    - **CQEQUE** — Execution Queue, list of tasks sent to device and not executed
- **RSTCTRL** — Reset controller, module distributes software resets to all clock domains
- **SYN** — Synchronization module between BIU (clk clock domain) and CIU (sdmclk clock domain) blocks.
- **SREG** — Shared Registers with Command Packets / Response Packets / Legacy Responses
- **FIU** — FIFO Interface Unit that implements buffering mechanism
- **CIU** — Card Interface Module
  - **CMD** — CMD Line Control module, it includes command transfer control logic
  - **DAT** — DAT line control module, it includes data transfer control logic
  - **IC** — Interrupt Controller, it includes SDIO card interrupt detection logic (SD mode)
  - **SCC** — Slot Clock Controller, it includes clock divider for SD interface (SD mode)
  - **SCU** — Slot Control Unit, it contains instance of **IC** and **SCC**.
- **SLAVE_IF** — Transaction wrapper between external interface and internal interface connected to register set.
- **MASTER_IF** — Transaction wrapper between internal interface driven by DMA and external interface used for integration.
- **Primitives**
  - **META** — Dual Flip-Flop synchronization module
  - **SYNCFLOP** — Dual Flip-Flop synchronization module instantiated inside META and provides randomized delay for CDC verification.
- **Debug modules**
  - **FSM_DBG_BIU** — collection of states of FSMs from **BIU** module and subcomponents
  - **FSM_DBG_CIU** — collection of states of FSMs from **CIU** module and subcomponents
  - **FSM_DBG** — unit that decodes request from HRS and provides current FSM status from any of **FSM_DBG_*** blocks

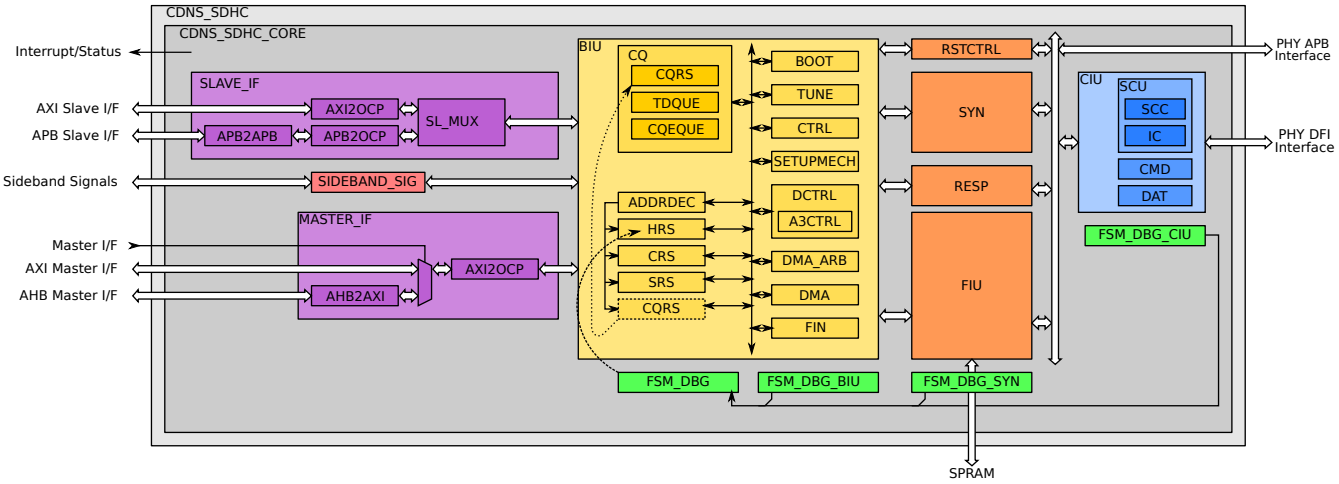

Figure 2. Core Internal Structure

4. User Definied Options

4.1 Controller IP Parametrization

Table 5. User Defined Options

Options	Description
FIFODEPTH	FIFO depth determines the maximum blocks size: • 6 — 512 bytes; data unit utilized by memory devices (SD, eMMC) • 8 — 2048 bytes; data unit could be utilized with SDIO
Slave Interface	Application can utilize either AXI or APB interface.
Master Interface	Application can utilize either AXI or AHB interface.

5. Pin List

5.1 Pin Description

The Table 6 provides controller pins definition.

Table 6. Pin List

Signal	Dir.	Clock	Bus Size	Description
Clock				
s_pclk	I	clock input	1	System clock
clk	I	clock input	1	System clock
sdmclk	I	clock input	1	SD Master clock
Resets				
s_presetn	I	s_pclk	1	Hardware reset in s_pclk clock domain
rstclk_n	I	clk	1	Hardware reset in clk clock domain
rstsdmclk_n	I	sdmclk	1	Hardware reset in sdmclk clock domain
Clock Control				
ics	I	clk	1	Internal Clock Stable
ice	O	clk	1	Internal Clock Enable
Interrupts				
interrupt	O	clk	1	System interrupt line
wakeup	O	clk	1	System wake-up line
Slave/Register AXI4-Lite Interface				
s_awaddr	I	clk	S_AWIDTH	Write address
s_awvalid	I	clk	1	Request on write address channel
s_awready	O	clk	1	Acknowledge on write address channel
s_wdata	I	clk	S_DWIDTH	Write data bus
s_wstrb	I	clk	S_DWIDTH/8	Write data strobe
s_wvalid	I	clk	1	Request on Write data channel
s_wready	O	clk	1	Acknowledge on write data channel
s_bready	I	clk	1	Acknowledge on write response channel
s_bresp	O	clk	2	Write response
s_bvalid	O	clk	1	Request on write response channel
s_araddr	I	clk	S_AWIDTH	Read address
s_arvalid	I	clk	1	Request on read address channel
s_arready	O	clk	1	Acknowledge on read address channel

s_rready	I	clk	1	Acknowledge on read data channel
s_rdata	O	clk	S_DWIDTH	Read data bus
s_rresp	O	clk	2	Read response
s_rvalid	O	clk	1	Request on read data channel
System/Register APB Interface				
s_paddr	I	s_pclk	S_AWIDTH	Address
s_pprot	I	s_pclk	3	Protection type
s_psel	I	s_pclk	1	Select
s_penable	I	s_pclk	1	Enable
s_pwrite	I	s_pclk	1	Direction
s_pwdata	I	s_pclk	S_DWIDTH	Write data
s_pstrb	I	s_pclk	S_DWIDTH/8	Write strobes
s_pready	O	s_pclk	1	Ready
s_prdata	O	s_pclk	S_DWIDTH	Read data
s_pslverr	O	s_pclk	1	Slave error
Master/DMA Interface Configuration				
master_if	I	constant	1	Master interface select
m_desc	O	clk	1	Descriptor transfer flag
Master/DMA AXI3 Interface				
m_awready	I	clk	1	AW Channel Ready
m_awaddr	O	clk	M_AWIDTH	AW Channel Address
m_awlen	O	clk	4	AW Channel Length
m_awsize	O	clk	3	AW Channel Size
m_awburst	O	clk	2	AW Channel Burst
m_awlock	O	clk	2	AW Channel Lock
m_awcache	O	clk	4	AW Channel Cache
m_awprot	O	clk	3	AW Channel Protect
m_awvalid	O	clk	1	AW Channel Valid
m_wready	I	clk	1	W Channel Ready
m_wdata	O	clk	M_DWIDTH	W Data Ready
m_wstrb	O	clk	M_DWIDTH/8	W Channel Strobe
m_wlast	O	clk	1	W Channel Last data
m_wvalid	O	clk	1	Write Channel Valid
m_bresp	I	clk	2	B Channel Response
m_bvalid	I	clk	1	B Channel Valid
m_bready	O	clk	1	B Channel Ready
m_arready	I	clk	1	AR Channel Ready
m_araddr	O	clk	M_AWIDTH	AR Channel Address
m_arlen	O	clk	4	AR Channel Length
m_arsize	O	clk	3	AR Channel Size
m_arburst	O	clk	2	AR Channel Burst
m_arlock	O	clk	2	AR Channel Lock
m_arcache	O	clk	4	AR Channel Cache
m_arprot	O	clk	3	AR Channel Protect
m_arvalid	O	clk	1	AR Channel Valid

m_rdata	I	clk	M_DWIDTH	R Channel Data
m_rresp	I	clk	2	R Channel Response
m_rlast	I	clk	1	R Channel Last data
m_rvalid	I	clk	1	R Channel Valid
m_rready	O	clk	1	R Channel Ready
Master/DMA AHB-Lite Interface				
m_hrdata	I	clk	M_DWIDTH	Read Data Bus
m_hready	I	clk	1	Ready
m_hresp	I	clk	1	Transfer Response
m_haddr	O	clk	M_AWIDTH	Address Bus
m_htrans	O	clk	2	Transfer type
m_hburst	O	clk	3	Burst type
m_hsize	O	clk	3	Transfer size
m_hmastlock	O	clk	1	Lock transfer
m_hprot	O	clk	4	Protection control
m_hwrite	O	clk	1	Transfer direction
m_hwdata	O	clk	M_DWIDTH	Write Data Bus
DMA Sideband Interface				
dma_rd_ack	I	clk	1	DMA Sideband Read Acknowledge
dma_wr_ack	I	clk	1	DMA Sideband Write Acknowledge
dma_rd_req	O	clk	1	DMA Sideband Read Request
dma_wr_req	O	clk	1	DMA Sideband Write Request
Combo PHY Interface				
sdphy_dfi_rebar	O	sdmclk	1	Read Enable
sdphy_dfi_rdcmd_a	I	N/A	1	Read Command (asynchronous input)
sdphy_dfi_rdcmd_valid	I	sdmclk	1	Read Command Valid
sdphy_dfi_rdcmd_en	O	sdmclk	1	Read Command Enable
sdphy_dfi_rdcmd	I	sdmclk	1	Read Command value
sdphy_dfi_wrcmd_en	O	sdmclk	1	Write Command Enable
sdphy_dfi_wrcmd	O	sdmclk	2	Write Command value
sdphy_dfi_rddata_a	I	sdmclk	8	Read Data (asynchronous input)
sdphy_dfi_rddata_valid	I	sdmclk	1	Read Data Valid
sdphy_dfi_rddata_en	O	sdmclk	1	Read Data Enable
sdphy_dfi_rddata	I	sdmclk	16	Read Data value
sdphy_dfi_wrdata_en	O	sdmclk	8	Write Data Enable
sdphy_dfi_wrdata	O	sdmclk	16	Write Data value
sdphy_dfi_webar	O	sdmclk	1	Write Enable
sdphy_dfi_webar_high	O	sdmclk	1	Write Enable High
sdphy_dfi_wpbar_a	I	sdmclk	1	Write Protect
sdphy_param_extended_rd_mode	O	sdmclk	1	Extended Read Mode
sdphy_param_extended_wr_mode	O	sdmclk	1	Extended Write Mode
sdphy_dfi_ctrlupd_ack	I	sdmclk	1	Control Update Acknowledge
sdphy_dfi_ctrlupd_req	O	sdmclk	1	Control Update Request
sdphy_dll_rst_n	O	sdmclk	1	DLL reset

sdphy_dfi_dqs_overflow	I	sdmclk	1	Data path overflow
sdphy_dfi_dqs_underrun	I	sdmclk	1	Data path underrun
sdphy_dfi_dqs_cmd_overflow	I	sdmclk	1	Command path overflow
sdphy_dfi_dqs_cmd_underrun	I	sdmclk	1	Command path underrun
sdphy_dfi_init_complete	I	sdmclk	1	Initialization Completed
sdphy_dfi_rstbar	O	clk	1	Reset
sdphy_dfi_ctrl_0_a	I	N/A	1	Control 0 (card detect)
sdphy_dfi_ctrl_1	O	clk	1	Control 1 (Bus Power)
sdphy_dfi_ale	O	clk	1	Pull-up / pull-down resistor control for LVS Identification
sdphy_dfi_bv	O	clk	3	Bus Voltage
sdphy_dfi_cle	I	clk	1	Current Limit Error
sdphy_dfi_led	O	clk	1	LED control
sdphy_dfi_1v8	O	clk	1	1.8V signaling enable
sdphy_dfi_drvss	O	clk	2	Drive Strength Select
Clock				
sdphy_reg_pclk	I	sdphy_reg_pclk	1	Select
Resets				
sdphy_reg_presetn	I	sdphy_reg_pclk	1	Select
Combo PHY Interface				
sdphy_reg_psel	O	sdphy_reg_pclk	1	Select
sdphy_reg_penable	O	sdphy_reg_pclk	1	Enable
sdphy_reg_paddr	O	sdphy_reg_pclk	16	Address
sdphy_reg_pwrite	O	sdphy_reg_pclk	1	Write direction
sdphy_reg_pwdata	O	sdphy_reg_pclk	32	Write Data bus
sdphy_reg_pready	I	sdphy_reg_pclk	1	Ready
sdphy_reg_prdata	I	sdphy_reg_pclk	32	Read Data bus
Host Settings				
s0_hwinit_srs16	I	constant	32	SRS16 - Capabilities Registers #1
s0_hwinit_srs17	I	constant	32	SRS17 - Capabilities Registers #2
s0_hwinit_srs18	I	constant	32	SRS18 - Capabilities Registers #3
s0_hwinit_srs19	I	constant	32	SRS19 - Capabilities Registers #4
s0_hwinit_srs24	I	constant	32	SRS24 - Preset Value Registers #1
s0_hwinit_srs25	I	constant	32	SRS25 - Preset Value Registers #2
s0_hwinit_srs26	I	constant	32	SRS26 - Preset Value Registers #3
s0_hwinit_srs27	I	constant	32	SRS27 - Preset Value Registers #4
hwinit_itcfmul	I	constant	4	Command Queuing Internal Timer Clock Frequency Multiplier
hwinit_itcfval	I	constant	10	Command Queuing Internal Timer Clock Frequency Value
hwinit_itcfsel	I	constant	5	Command Queuing Internal Timer Clock Frequency Select
FIFO - RAM Interface				
biu0_datar	I	clk	RAM_DWIDTH	Read Data

biu0_we	O	clk	1	Write Enable
biu0_re	O	clk	1	Read Enable
biu0_dataw	O	clk	RAM_DWIDTH	Write Data
biu0_addr	O	clk	FIFODEPTH	Address
biu1_datar	I	clk	RAM_DWIDTH	Read Data
biu1_we	O	clk	1	Write Enable
biu1_re	O	clk	1	Read Enable
biu1_dataw	O	clk	RAM_DWIDTH	Write Data
biu1_addr	O	clk	FIFODEPTH	Address
ciu0_datar	I	synclk	RAM_DWIDTH	Read Data
ciu0_we	O	synclk	1	Write Enable
ciu0_re	O	synclk	1	Read Enable
ciu0_dataw	O	synclk	RAM_DWIDTH	Write Data
ciu0_addr	O	synclk	FIFODEPTH	Address
ciu1_datar	I	synclk	RAM_DWIDTH	Read Data
ciu1_we	O	synclk	1	Write Enable
ciu1_re	O	synclk	1	Read Enable
ciu1_dataw	O	synclk	RAM_DWIDTH	Write Data
ciu1_addr	O	synclk	FIFODEPTH	Address
eMMC Boot Configuration Interface				
boot_en	I	constant	1	Boot operation enable
boot_ack_en	I	constant	1	Boot Acknowledge Pattern enable
boot_method	I	constant	1	Boot method select
boot_buswidth	I	constant	2	Boot width mode select
boot_sdclkdiv	I	constant	10	SD clock divider select
boot_addrwidth	I	constant	1	System address bus width
boot_addr	I	constant	64	System address
boot_blockcnt	I	constant	32	Block count
boot_blocksize	I	constant	12	Block size
boot_bvs	I	constant	3	Bus voltage select
boot_speedmode	I	constant	2	Speed mode select
boot_pow_clk_dly	I	constant	32	Power to Clock delay
boot_clk_cmd_dly	I	constant	32	Clock to Command delay
boot_timeout_ack	I	constant	4	Boot Acknowledge Timeout
boot_timeout_dat	I	constant	4	Boot Data Transfer Timeout
boot_ack	O	clk	1	Boot Acknowledge Status Received
boot_done	O	clk	1	Boot acknowledge
boot_err	O	clk	1	Boot error
boot_wr_dly	I	constant	32	Boot CMD/DAT output delay
boot_io_dly	I	constant	10	Boot IO delay
boot_exten	I	constant	4	Boot enable Extended PHY Mode
boot_clkadj	I	constant	4	Boot SDCLK adjustment
boot_phy_dqstm	I	constant	4	Boot PHY DQS timing
boot_phy_dqtm	I	constant	10	Boot PHY DQ timing
boot_phy_glpbk	I	constant	9	Boot PHY gate and loopback control
boot_phy_dllmst	I	constant	1	Boot PHY Master DLL control
boot_phy_dllslv	I	constant	16	Boot PHY Slave DLL control

boot_setup_mode	I	constant	1	Boot setup method
boot_desc_addr	I	constant	64	Boot descriptor list address
Descriptor Mechanism Status				
desc_mech_error	O	clk	1	Descriptor mechanism active
desc_mech_active	O	clk	1	Descriptor mechanism error

Note:

- S_DWIDTH — slave interface data width (fixed 32-bits)
- S_AWIDTH — slave interface address width (fixed 13-bits)
- M_DWIDTH — master interface data width (fixed 64-bits)
- M_AWIDTH — master interface address width (fixed 64- bits)
- RAM_DWIDTH — data path width (fixed 64-bits)
- FIFODEPTH — SPRAM memory depth connected to FIFO (6 or 8 (default))

6. Register Address Map

6.1 Memory Map

Table 7 lists the Controller IP registers.

Table 7. Registers

Start	End	Name	Description
000h	0EFh	Host Register Set (HRS)	Control and status registers specific to Cadence implementation of the Host Controller. Those registers are additional and not covered by the SD Host Controller Standard Specification (SD Card Association).
2F0h	2FFh	Common Register Set (CRS)	Status registers defined by the SD Host Controller Standard Specification (SD Card Association).
200h	2EFh	Slot Register Set (SRS)	Control and status registers defined by the SD Host Controller Standard Specification dedicated for SD/eMMC slot.
400h	4FFh	Command Queuing Register Set (CQRS)	Control and status register defined by the eMMC Standard Specification (JEDEC) dedicated for Command Queuing. Registers are defined by the eMMC 5.1 Standard in section "B.4. CQE Registers".

Note: Unimplemented fields/registers are not described. These are read 00000000h. Write to these registers is ignored.

7. Functional Description

7.1 Block-Level Description

Figure 2 on page 16 gives a brief overview on the Controller IP and its interfaces. In this section the functionality of interfaces and some major blocks is described.

7.2 Interfaces

This section describes system interfaces and back-end interfaces.

The slave interface is provided to support a register access. The slave interface is equipped in module handling AXI4-Lite or APB signaling.

The master interface is provided to support a DMA access. The master interface is equipped in modules handling AXI3 or AHB-Lite signaling.

The slave and master interfaces are selected upon the SD Host Controller (top level) integration. Input ports of unused interface have to be tied to 0. Output ports of unused interface should be left unconnected. Additionally, to select type of the master interface, *master_if* port has to be tied to 0 (AXI) or 1 (AHB-Lite).

The following sections describe the interfaces.

7.2.1 AXI4-Lite Slave Interface

The AXI4-Lite slave interface is provided to connect the SD/eMMC Host Controller to the AXI4-Lite compatible SoC. The interface operates synchronously to the *clk* clock signal which should be connected to the AXI bus. The Slave AXI interface supports 13-bit address and 32-bit data channels. Only single 32-bit data transfers are supported on AXI4-Lite interface. For more detailed information please refer to AXI4-Lite specification document.

If the AXI4-Lite Slave Interface must be connected to the system, it will be connected as in example (Figure 3).

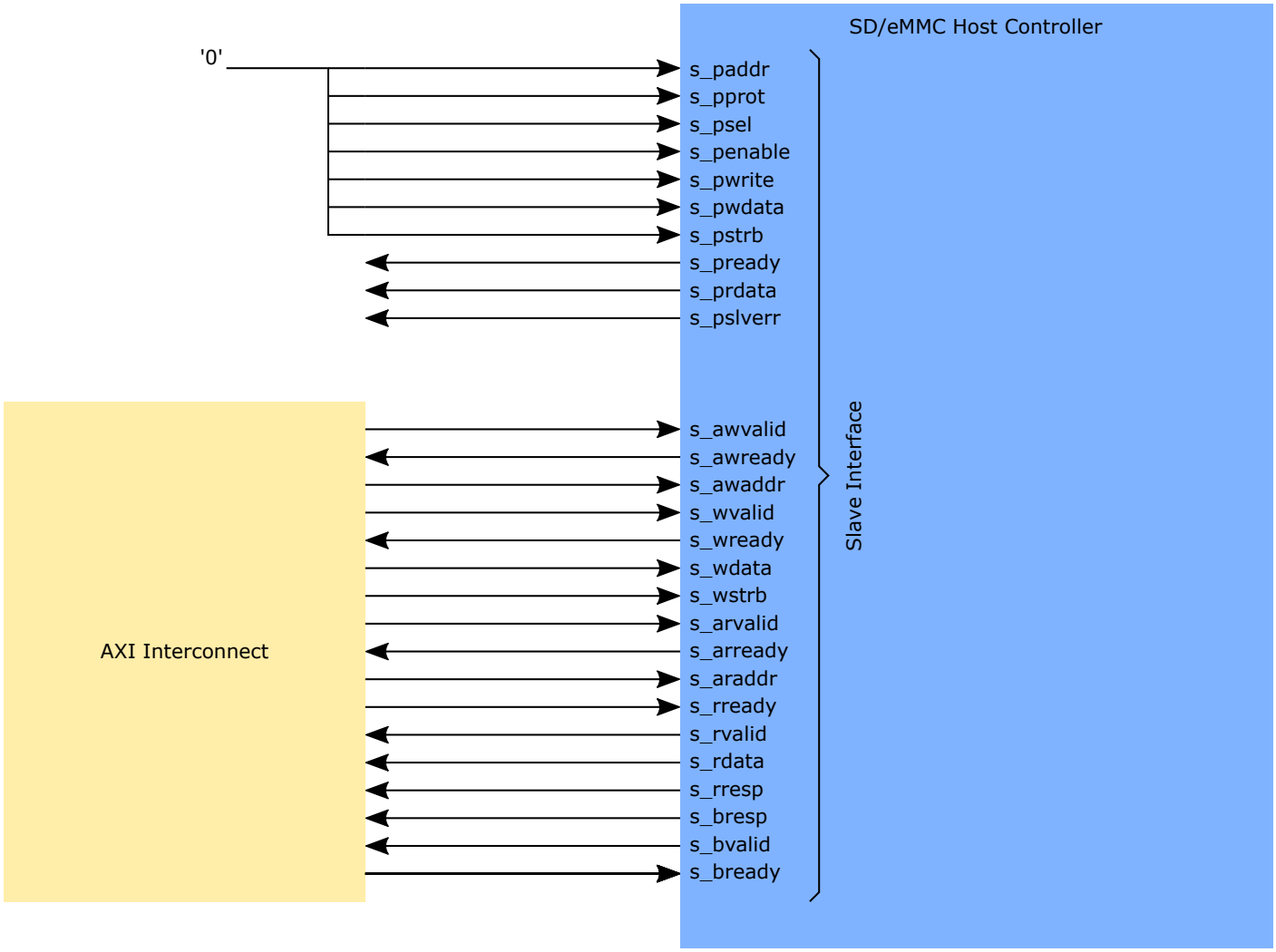

Figure 3. AXI Interface connection

An example write and read transactions are shown in Figure 4.

The AXI4-Lite interface is presented in Table 8.

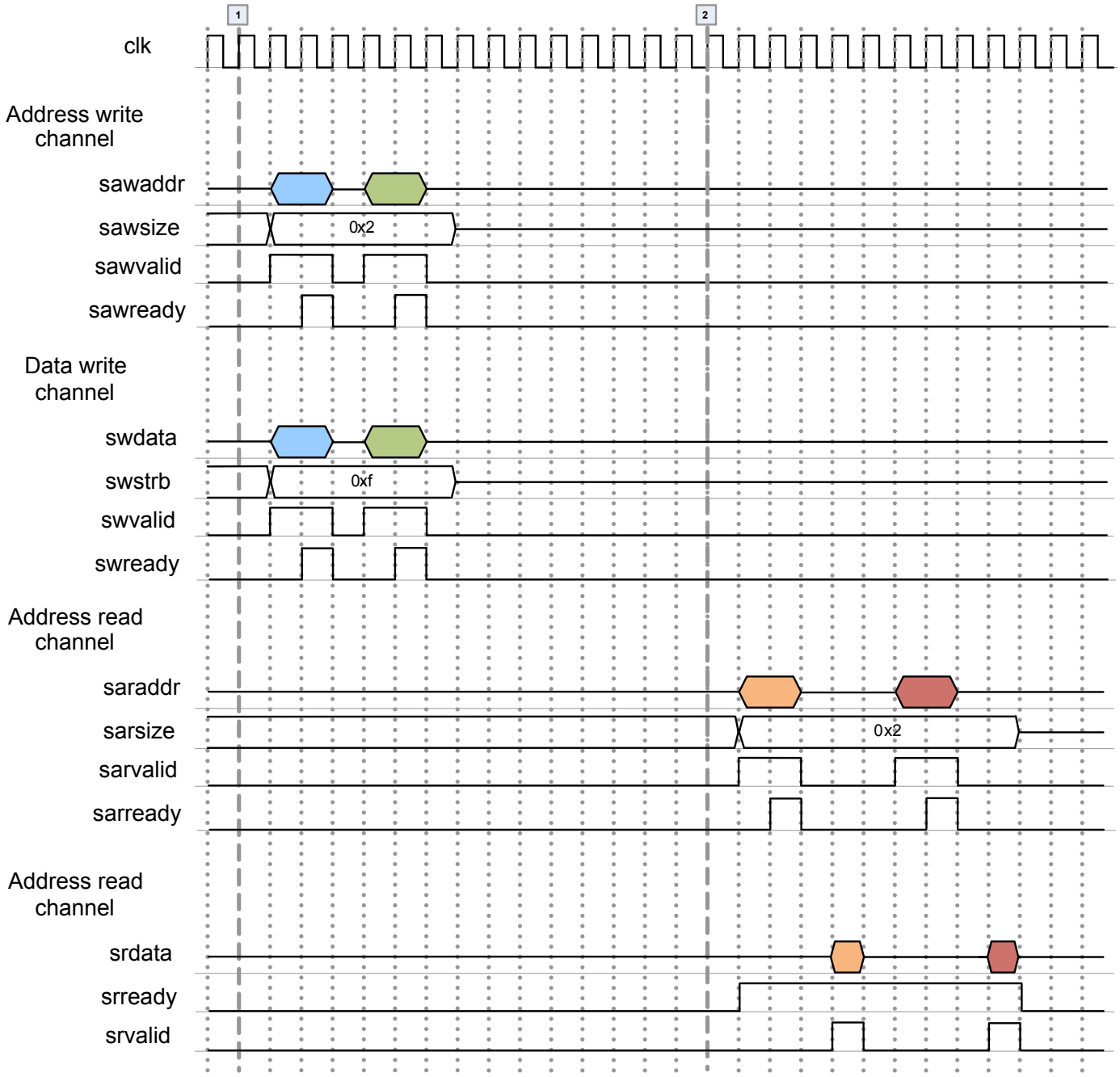

Figure 4. AXI4-Lite Slave Interface — write and read commands

Table 8. AXI4-Lite Slave Interface Signal Description

Name	Type	Polarity Bus Size	Description
AXI Slave address write channel			
s_awvalid	in	1	**Slave address write valid indicator**
s_awready	out	1	**AXI address ready signal**
s_awaddr	in	32	**AXI slave address bus**
AXI Slave write data channel			
s_wvalid	in	1	**Slave write data valid signal**
s_wready	out	1	**AXI data write ready signal**
s_wdata	in	32	**Slave write data bus**
s_wstrb	in	4	**Slave write strobe**
AXI Slave read address channel			
s_arvalid	in	1	**Slave read address valid signal**
s_arready	out	1	**AXI read address channel ready signal**
s_araddr	in	32	**Slave read address bus**
AXI Slave read data channel			
s_rready	in	1	**Slave read data acknowledge signal**
s_rvalid	out	1	**AXI read data valid indicator**
s_rdata	out	32	**AXI read data bus**
s_rresp	out	2	**AXI data read response**
AXI Master response channel			
s_bresp	out	2	**AXI write response**
s_bvalid	out	1	**AXI Slave write response valid indicator**
s_bready	in	1	**Master write response ready signal**

7.2.2 APB Slave Interface

The APB slave interface is provided to connect the core to the APB compatible SoC. The interface operates synchronously to the *s_pclk* clock signal which should be connected to the APB bus (PCLK signal), and reset asynchronous by *s_presetn* input port. The APB Slave interface has 13-bit address and 32-bit data channels. For more detailed information please refer to APB specification document (AMBA APB Protocol Version: 2.0, 13 April 2010).

If the APB Slave Interface must be connected to the system, it will be connected as in example (Figure 5).

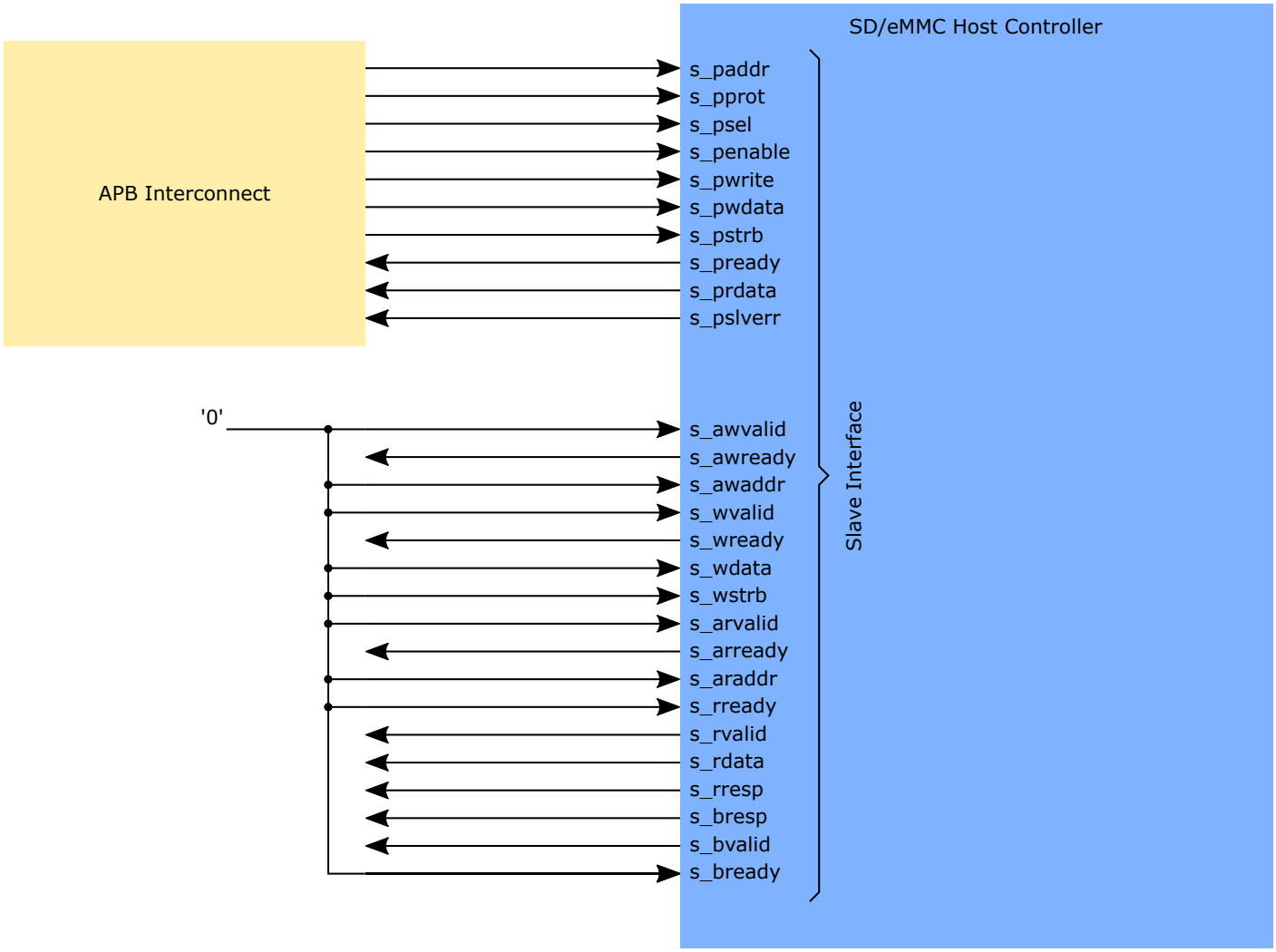

Figure 5. APB Interface connection

An example write and read transactions are shown in Figure 6.

The APB interface is presented in Table 9.

Table 9. APB Slave Interface Signal Description

Name	Type	Width	Clk Src	Description
s_paddr	in	13	s_pclk	**Address** — Selects one of the SD Host Controller register in a current transaction.
s_pprot	in	3	s_pclk	**Protection type** — Indicates access protection level; This value is ignored and all transactions are acknowledged identical regardless of this setting
s_psel	in	1	s_pclk	**Select** — Indicates that the SD Host Controller is a target in the active transaction.
s_penable	in	1	s_pclk	**Enable** — Indicates transfer phase (0 — Setup phase, 1 — Access phase)
s_pwrite	in	1	s_pclk	**Direction** — Defines transfer direction (0 — read transaction, 1 — write transaction).
s_pwdata	in	32	s_pclk	**Write data** — Has data content for the write transaction. It is stable during write transfer.

s_pstrb	in	4	s_pclk	**Write strobes** — Informs which bytes of s_pwdata are valid.
s_pready	out	1	s_pclk	**Ready** — Confirms that current transaction is completed.
s_prdata	out	1	s_pclk	**Read data** — Has data content for the read transaction. It is valid for the read transaction s_pready is high.
s_pslverr	out	1	s_pclk	**Slave error** — Tied to 0. The SD Host does not report slave interface error.

Notes:

- Width of data buses is fixed to 32-bit.
- Protection mechanism is not used, the s_pprot port value is ignored.
- SD Host Controller does not respond with transfer failure, the s_pslverr port value is tied to 0.

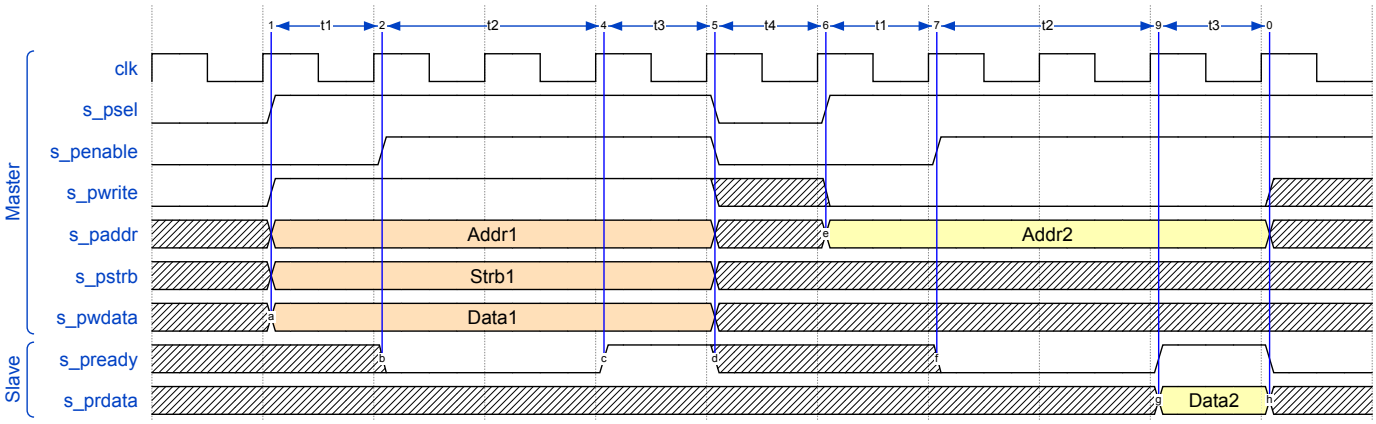

Figure 6. APB Slave Interface — write and read commands

The table 10 describes timings (in *s_pclk* clock cycles) used on the figure 6.

Table 10. APB Slave Interface Timings

Time	Min	Max	Description
t1	1	1	Period between interface selected and valid ready
t2	0	–	Period between Access phase started and ready acknowledge
t3	1	1	Period of ready acknowledge
t4	0	–	Period between ending Access phase and next Setup phase

7.2.3 AXI3 Master Interface

The AXI Master interface is provided to connect the core to the AXI3 interconnect compatible SoC. The interface operates synchronously to the *clk* clock signal. The AXI master interface supports 64-bit address and 64-bit data channels. The interface generates sequential read and write burst transfers of precise length (1 to 16 beats). The core supports programmable number of outstanding transfer in range from 1 to 4. Axi Master Interface Signal description is provided in Table 11 For more detailed information please refer to AXI3 specification document.

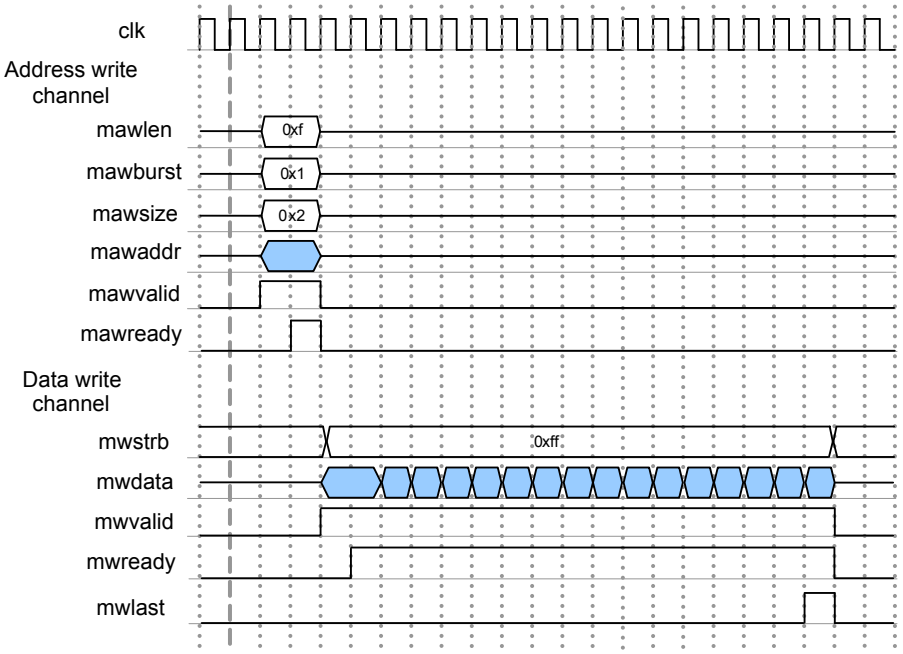

Figure 7. AXI Master Interface — write commands

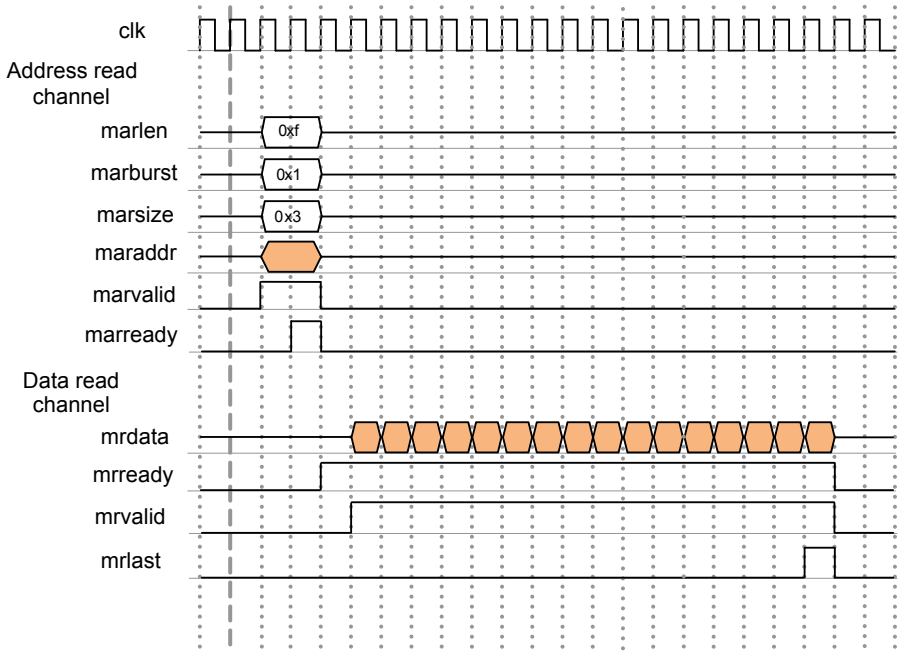

Figure 8. AXI Master Interface — read commands

Table 11. AXI Master Interface Signal Description

Name	Type	Polarity Bus Size	Description
AXI Master address write channel			
awready	in	1	**AXI address ready signal**
awaddr	out	64	**AXI master address bus**
awlen	out	4	**Write length**
awsize	out	3	**Write size**
awburst	out	2	**Master write burst type** Tied to ’b01
awlock	out	2	**Master write lock** Tied to ’b00
awcache	out	4	**Master write cache** Tied to ’b0000
awprot	out	3	**Master write protect** Tied to ’b010
awvalid	out	1	**Master address write valid indicator**
AXI Master write data channel			
wready	in	1	**AXI data write ready signal**
wdata	out	64	**Master write data bus**
wstrb	out	8	**Master write strobe**
wlast	out	1	**Master write data last signal**
wvalid	out	1	**Master write data valid signal**
AXI Master response channel			
bresp	in	2	**AXI write response**
bvalid	in	1	**AXI write response valid indicator**
bready	out	1	**Master write response ready signal**
AXI Master read address channel			
arready	in	1	**AXI read address channel ready signal**
araddr	out	64	**Master read address bus**
arlen	out	4	**Master read length**
arsize	out	3	**Master read size**
arburst	out	2	**Master read burst type** Tied to ’b01
arlock	out	2	**Master read lock signal** Tied to ’b00
arcache	out	4	**Master read cache** Tied to ’b0000
arprot	out	3	**Master read protect** Tied to ’b010
arvalid	out	1	**Master read address valid signal**
AXI Master read data channel			
rdata	in	64	**AXI read data bus**
rresp	in	2	**AXI data read response**
rlast	in	1	**AXI last data read indicator**
rvalid	in	1	**AXI read data valid indicator**
rready	out	1	**Master read data acknowledge signal**

Integration note: In order to use the AXI3 master interface, connect *master_if* input port to 0, connect all unused AHB-Lite master input to 0, and do not use AHB-Lite master outputs.

7.2.4 AHB-Lite Master Interface

The AHB-Lite Master interface is provided to connect the core to the AHB-Lite system bus. The interface operates synchronously to the *clk* clock signal. The AHB-Lite master interface supports 64-bit address and 64-bit data channels. AHB Master Interface Signal Description is provided in table For more detailed information please refer to AHB specification document.

Table 12. AHB Master Interface Signal Description

Name	Type	Polarity Bus Size	Description
m_hrdata	in	64	**Read Data**
m_hready	in	1	**Ready**
m_hresp	in	1	**Transfer Response**
m_haddr	out	64	**Address**
m_htrans	out	2	**Transfer type**
m_hburst	out	3	**Burst type**
m_hsize	out	3	**Transfer size**
m_hmastlock	out	1	**Lock transfer** Unused, it is tied to 0.
m_hprot	out	4	**Protection control** Tied to 'b0001.
m_hwrite	out	1	**Transfer direction**
m_hwdata	out	64	**Write Data**

Integration note: In order to use the AHB-Lite master interface, connect master_if input port to 1, connect all unused AXI3 master input to 0, and do not use AXI3 master outputs.

7.2.5 SPRAM Interfaces

The SD Host Controller provides SPRAM Interfaces (table 13) for the external data buffers (section 7.2.5.1). Four SPRAM instances are required, i.e. SPRAM#0 and SPRAM#1 on BIU side (in *clk* clock domain) and SPRAM#0 and SPRAM#1 on CIU side (in *sdmclk* clock domain). All RAM instances have identical organization, which is either 64x64-bits (when FIFODEPTH parameter is 6) or 256x64-bits (when FIFODEPTH parameter is 8). The RAM organization and FIFODEPTH parameter defines maximum block size supported by the Host Controller and it is 512 bytes or 2048 bytes, respectively.

Table 13. RAM Interface — Data Buffer

Name	Type	Polarity Bus Size	Description
biu0_addr	out	FIFODEPTH	**BIU SPRAM#0 Address**
biu0_datar	in	64	**BIU SPRAM#0 Read Data**
biu0_re	out	high	**BIU SPRAM#0 Read Enable**
biu0_dataw	out	64	**BIU SPRAM#0 Write Data**
biu0_we	out	high	**BIU SPRAM#0 Write Enable**
ciu0_addr	out	FIFODEPTH	**CIU SPRAM#0 Address**
ciu0_datar	in	64	**CIU SPRAM#0 Read Data**
ciu0_re	out	high	**CIU SPRAM#0 Read Enable**

Table 13. RAM Interface — Data Buffer

Name	Type	Polarity Bus Size	Description
ciu0_dataw	out	64	**CIU SPRAM#0 Write Data**
ciu0_we	out	high	**CIU SPRAM#0 Write Enable**
biu1_addr	out	FIFODEPTH	**BIU SPRAM#1 Address**
biu1_datar	in	64	**BIU SPRAM#1 Read Data**
biu1_re	out	high	**BIU SPRAM#1 Read Enable**
biu1_dataw	out	64	**BIU SPRAM#1 Write Data**
biu1_we	out	high	**BIU SPRAM#1 Write Enable**
ciu1_addr	out	FIFODEPTH	**CIU SPRAM#1 Address**
ciu1_datar	in	64	**CIU SPRAM#1 Read Data**
ciu1_re	out	high	**CIU SPRAM#1 Read Enable**
ciu1_dataw	out	64	**CIU SPRAM#1 Write Data**
ciu1_we	out	high	**CIU SPRAM#1 Write Enable**

An example SPRAM transactions are shown in figure 9.

Write to the SPRAM takes 1 clock cycle. In this clock cycle, the write enable is set high (*we*), the address bus (*addr*) points to a destination memory location and the write bus (*dataw*) carries written 64-bit data.

Read from SPRAM takes 2 clock cycles. In first clock cycle, the read enable is set high (*re*) and the address bus (*addr*) points to a source memory location. In the next clock cycle, 64-bit data is ready on the read data bus (*datar*).

Read or write requests can be back-to-back (without idle cycles). Read and write request won't be requested in the same clock cycle.

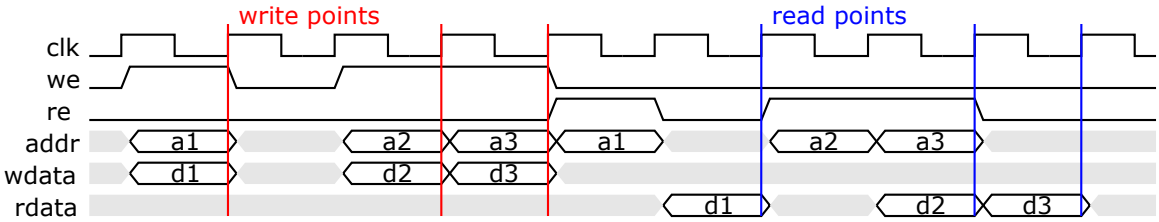

Figure 9. SPRAM Read/Write examples

7.2.5.1 External Data Buffers

The SD Host Controller requires four external data buffers for a read/write transaction. The buffers are controlled by the internal FIFO (FIU). The FIU utilizes those buffers to store SD/eMMC data blocks and to minimize wait states on the data path, i.e. a delay between consecutive data blocks on the BIU/CIU interface.

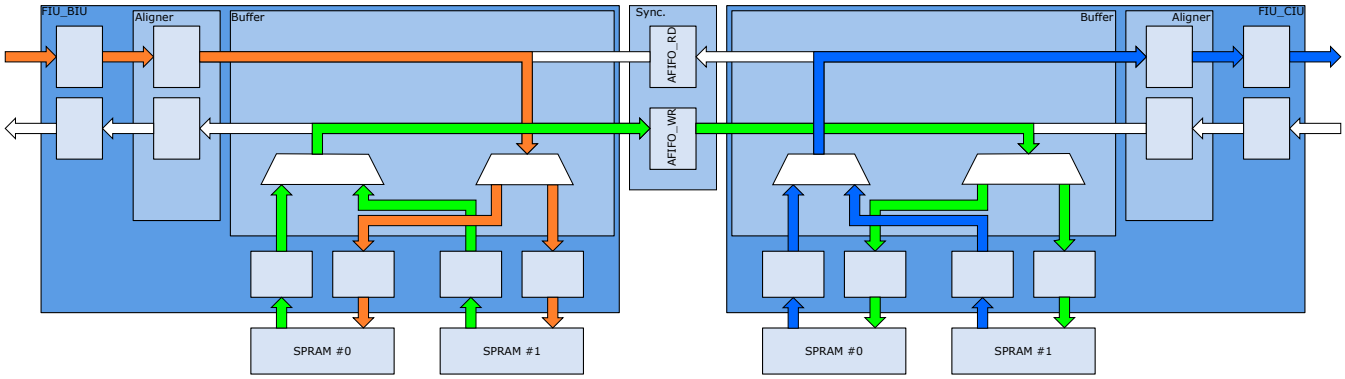

Figure 10. FIU in write transaction

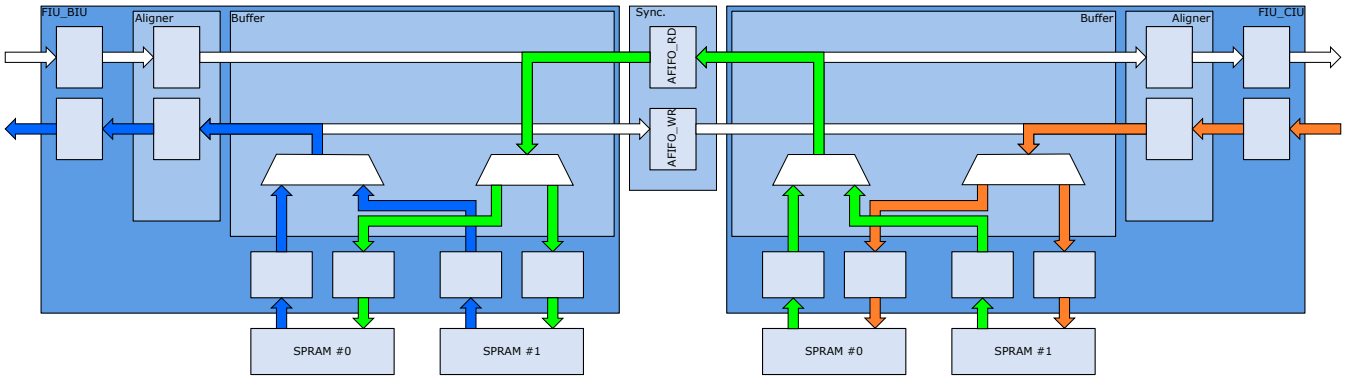

Figure 11. FIU in read transaction

The FIU data path is shown in two figures 10 and 11. First one illustrates a write transaction where data goes from system bus interface (BIU) to card interface (CIU). Second one illustrates a read transaction where data goes in opposite direction. Transfer through FIU is performed in three phases:

1. Write data to FIU (in orange)
2. Transfer data between clock domains (in green)
3. Read data from FIU (in blue)

All phases are realized in concurrent processes while each SPRAM (buffer) is utilized in one phase only. To minimize consequences of idle states between accesses on BIU and CIU interface, as there is time needed to write to SPRAM and then read from the SPRAM, the FIU can run phases 1 and 2, and phases 2 and 3 simultaneously. For example phase 1 can be run on SPRAM#0 and Phase 2 on SPRAM#1, or vice versa.

7.2.6 SD Interface (SD/eMMC)

In this section the SD Interface is presented. The interface supports SD (SD, SDHC, SDXC, SDUC memory cards) , SDIO cards , and eMMC devices.

The interface supports the following modes:

- SD Standard
  - Default Speed (DS)
  - High Speed (HS)
  - UHS-I SDR12
  - UHS-I SDR25
  - UHS-I SDR50
  - UHS-I SDR104
  - UHS-I DDR50
- eMMC Standard
  - Standard Speed
  - High Speed
  - DDR52
  - HS200
  - HS400
  - HS400 Enhanced Strobe

Table 14. SD Interface description

Name	Type	Polarity Bus Size	Description
sdphy_dfi_rebar	out	high	**Read enable** Control port for the Combo PHY.
sdphy_dfi_rdcmd_a	in	1	**Read CMD line - asynchronous** Port receives CMD line state from the Combo PHY. The signal is synchronized internally and sampled at the rising edge of *clk* and *sdmclk*. This signal is provided by the SD/eMMC PHY. The value represents a current state on the CMD line.
sdphy_dfi_rdcmd_valid	in	1	**Read CMD valid**
sdphy_dfi_rdcmd_en	out	high	**Read CMD enable**
sdphy_dfi_rdcmd	in	1	**Read CMD value**
sdphy_dfi_wrcmd_en	out	high	**Write CMD enable**
sdphy_dfi_wrcmd	out	2	**Write CMD value**
sdphy_dfi_rddata_a	in	8	**Read DAT line - asynchronous** The signal is sampled at the rising edge of *clk* and *sdmclk* This signal is provided by the SD/eMMC PHY. The value represents a current state on the DAT lines.
sdphy_dfi_rddata_valid	in	high	**Read DAT valid**
sdphy_dfi_rddata_en	out	high	**Read DAT enable**
sdphy_dfi_rddata	in	16	**Read DAT value**
sdphy_dfi_wrdata_en	out	8	**Write Data enable**

Table 14. SD Interface description (con't)

Name	Type	Polarity Bus Size	Description
sdphy_dfi_wrdata	out	16	**Write Data value**
sdphy_dfi_webar	out	high	**Write Enable**
sdphy_dfi_webar_high	out	high	**Write Enable hold in High**
sdphy_dfi_wpbar_a	in	low	**Write Protect - asynchronous** This signal is supplied by micro-switch in the SD/MMC socket. The socket informs about position of the 'LOCK' switch on the card's edge. The signal is sampled at the rising edge of *clk*. A state of this port indicates whether a write operation is permitted (1) or it is not (0). This is software only mechanism, so the software should not allow to perform any operation that changes device content of the memory. This is typically connected to the mechanical switch on the card slot. The signal level on this port can be checked by reading SRS9.WPSL register bit. Please note, this value is not debounced. It should be recognized as valid only if SRS09.CSS (Card State Stable) bit is set. If the given slot is not used, this pin is connected to high. For embedded devices (eSD/eMMC), this port is tied to high.
sdphy_param_extended_rd_mode	out	high	**Extended Read Mode**
sdphy_param_extended_wr_mode	out	high	**Extended Write Mode**
sdphy_dfi_ctrlupd_ack	in	high	**DLL Update Acknowledge**
sdphy_dfi_ctrlupd_req	out	high	**DLL Update Request**
sdphy_dll_rst_n	out	low	**DLL Reset**
sdphy_dfi_dqs_overflow	in	high	**Data Overflow**
sdphy_dfi_dqs_underrun	in	high	**Data Underrun**
sdphy_dfi_dqs_cmd_overflow	in	high	**Command Overflow**
sdphy_dfi_dqs_cmd_underrun	in	high	**Command Underrun**
sdphy_dfi_init_complete	in	high	**Initialization Complete**
sdphy_dfi_rstbar	out	1	**Device Reset**

Table 14. SD Interface description (con't)

<table>
<tr><th>Name</th><th>Type</th><th>Polarity
Bus Size</th><th>Description</th></tr>
<tr><td>sdphy_dfi_ctrl_0_a</td><td>in</td><td>low</td><td>Card Detect
This signal is supplied by micro-switch in the SD/MMC socket. The socket informs that the card is inserted by asserting this signal to 0. The signal remains 1 when there is no card in the slot.
The signal is sampled by clk.
The Host performs hardware debouncing on this pin to produce the reliable card detection state.
Software can read SRS09.CI (Card Inserted) and SRS09.CSS (Card State Stable) register bits to check the state of card detection logic, using the following table:
<table>
<tr><th>SRS09 CSS</th><th>SRS09 CI</th><th>card detection logic state</th></tr>
<tr><td>0</td><td>not valid</td><td>debouncing</td></tr>
<tr><td>1</td><td>0</td><td>no card</td></tr>
<tr><td>1</td><td>1</td><td>card inserted</td></tr>
</table>
Additionally, the inverted value of this port can be read directly (without debouncing functionality) by the software through SRS09.CDL register bit. This can be useful for debugging purposes.
The port transitioning from 0 to 1 indicates card removal. Host reacts on this event switching off the card power (sdphy_dfi_ctrl_1 gets asserted to 0) and turning off the SD Clock (SD Clock Enable (SRS11.SDCE) gets asserted to 0). This provides the "hot removal" support.
For embedded devices (eSD/eMMC), this port is tied to low which mimics constants availability of the device.</td></tr>
<tr><td>sdphy_dfi_ctrl_1</td><td>out</td><td>high</td><td>Power Supply
This port is updated on the rising edge of sdmclk.
Driven directly by the SRS10.BP (Bus Power) register bit in order to control an external power supply circuitry for the card slot.
The Host will automatically switch off power supply for device automatically after detecting No Card state.</td></tr>
<tr><td>sdphy_dfi_ale</td><td>out</td><td>1</td><td>SD pull-up/pull-down for DAT2</td></tr>
<tr><td>sdphy_dfi_bv</td><td>out</td><td>3</td><td>Bus Voltage select
This port is updated on the rising edge of sdmclk.
Driven directly by the SRS10.BVS (Bus Voltage Select) register bits in order to control an external voltage control circuitry for the device.
This port could be utilized in the application that has to support devices that supports various voltage range standards. If that is not a requirement, this port is not used and can be left unconnected.</td></tr>
<tr><td>sdphy_dfi_cle</td><td>in</td><td>1</td><td>Current Limit Error
The signal is sampled at the rising edge of clk.
This signal can be set by the external power supply logic when it is not able to supply power to the card. If it is not used, the s0_cle pin is tied to low. Rising edge on the s0_cle pin can trigger SRS12.ECL interrupt bit.</td></tr>
</table>

Table 14. SD Interface description (con't)

Name	Type	Polarity Bus Size	Description
sdphy_dfi_led	out	1	**LED output** It changes at the rising edge of *sdmclk*. It is driven directly by the SRS10.LEDC register in order to control the external LED diode. LEDC=1 switches LED on, while LEDC=0 switches it off. The software is intend to switch LED on to warn user not to remove the card while there is a transfer in progress.
sdphy_dfi_1v8	out	1	**Signaling in 1.8V**
sdphy_dfi_drvss	out	2	**Drive Strength Select** It changes at the rising edge of *sdmclk*. This signal is controlled by the SRS15.DSS. It is used to select electrical parameters of the interface. Please refer to SD Physical Layer Specification for details on electrical parameters. This signal is optional, it should be left unconnected if unused.
sdphy_reg_pclk	in	1	**Clock**
sdphy_reg_presetn	in	1	**Reset**
sdphy_reg_psel	out	1	**Select**
sdphy_reg_penable	out	1	**Enable**
sdphy_reg_paddr	out	16	**Address**
sdphy_reg_pwrite	out	1	**Write indicator**
sdphy_reg_pwdata	out	32	**Write data**
sdphy_reg_pready	in	1	**Ready**
sdphy_reg_prdata	in	32	**Read data**
sdphy_reg_pslverr	in	1	**Error** This port is unused. The PHY reports error when accessing to undefined address, but for the Host this behavior is absolutely correct and returns value 0 and do not report this as an error.

Table 15. Speed mode selection

Speed Mode	HRS06.EMM	SRS15.V18SE	SRS10.HSE	SRS15.UMS
Default Speed	0000b	0	0	–
High Speed	0000b	0	1	–
UHS-I SDR12	0000b	1	–	000b
UHS-I SDR25	0000b	1	–	001b
UHS-I SDR50	0000b	1	–	010b
UHS-I SDR104	0000b	1	–	011b
UHS-I DDR50	0000b	1	–	100b
eMMC Legacy	0001b	–	–	–
eMMC SDR	0010b	–	–	–
eMMC DDR	0011b	–	–	–
eMMC HS200	0100b	–	–	–
eMMC HS400	0101b	–	–	–
eMMC HS400ES	0110b	–	–	–

7.2.7 Timings

- The clock inputs need to be constrained as defined in section 10.1.
- It is assumed that the Slave and Master System Interfaces have maximum input and output delays defined as one third of the system clock frequency.
- The SD/eMMC interface timings are defined in external documents — SD Physical Layer Standard Specification and eMMC Standard Specification.

7.2.8 Boot Interface

Table 16. Boot settings

Name	Type	Width	Clk Src	Description
eMMC Boot Configuration Interface				
boot_en	in	1	static	**Boot operation enable** 1 — The boot operation starts automatically start after the hardware reset ended. 0 — The host does not start boot operation.
boot_ack_en	in	1	static	**Boot Acknowledge Pattern enable** 1 — The host expects to receive Boot Acknowledge Pattern before starting data transfer. Software must preconfigure eMMC device by setting BOOT_ACK bit in PARTITION_CONFIG to 1 before boot starts. 0 — The host does not expect Boot Acknowledge Pattern. Software must preconfigure eMMC device by setting BOOT_ACK bit in PARTITION_CONFIG to 0 before boot starts.
boot_method	in	1	static	**Boot method select** 0 — CMD line changes to low triggers boot operation 1 — CMD0 with argument 0xFFFFFFFA triggers boot operation

boot_buswidth	in	2	static	**Boot width mode select** 0 — 1-bit mode 1 — 4-bit mode 2 — 8-bit mode 3 — reserved (should not be used)
boot_sdclkdiv	in	10	static	**SD clock divider select** The value selects the frequency of the SDCLK clock. The clock frequency can be calculated in similarly to the SDCLK Frequency Select registers.
boot_addrwidth	in	1	static	**System address bus width** System address bus width (0 — 32 bits, 1 — 64 bits).
boot_addr	in	64	static	**System address** The data is streamed to the continuous system memory area and the start address is provided by this value.
boot_blockcnt	in	16	static	**Block count** The number of block to be streamed to system memory. The value has to be defined in range 1 to $2^{16}-1$.
boot_blocksize	in	12	static	**Block size** It defines block size. It can be set in range from 1 to $InternalBufferSize$ (and not over $2kB$). The value has to be supported by the boot.
boot_bvs	in	3	static	**Bus voltage select** The value overwrites the SRS10.BVS (SD Bus Voltage Select).
boot_speedmode	in	2	static	**Speed mode select** 0 – eMMC Legacy (backward compatible mode) 1 – eMMC SDR 2 – eMMC DDR
boot_pow_clk_dly	in	32	static	**Power to Clock delay** It defines delay between enabling power supply and enabling SDCLK. The delay is defined in number of CLK clock cycles.
boot_clk_cmd_dly	in	32	static	**Clock to Command delay** It defined delay between enabling SDCLK and issuing the boot trigger. The delay is defined in number of CLK clock cycles.
boot_timeout_ack	in	4	static	**Boot Acknowledge Timeout** It consists value used to calculate a timeout value for boot acknowledge. The timeout is equal $2^{SRS11.DTCV+15} * t_{CLK}$. Where t_{CLK} is a system clock interface (CLK) period.
boot_timeout_dat	in	4	static	**Boot Data Transfer Timeout** It defines timeout value for the trigger-first block period and block gap period. The delay is defined identically as the SRS11.DTCV.
boot_wr_dly	in	32	static	**Boot CMD/DAT output delay** The delay value that will be written to HRS16 register before starting boot operation.
boot_io_dly	in	10	static	**Boot IO delay** The delay value that will be written to HRS07 register before starting boot operation.
boot_exten	in	4	static	**Boot enable Extended PHY Mode** It controls PHY dfi read enable signals and sets PHY extended mode ports. See HRS09 register description.
boot_clkadj	in	4	static	**Boot SDCLK adjustment** The value adjusts flow control mechanism which disables SDCLK. Refer to HRS10 register, HCSDCLKADJ field.

boot_phy_dqstm	in	4	static	**Boot PHY DQS timing** The value that will be written to PHY phy_dqs_timing_reg[22:19] register bits before starting boot operation. Refer to SD/eMMC PHY documentation for more details.
boot_phy_dqtm	in	10	static	**Boot PHY DQ timing** The value that will be written to PHY phy_dq_timing_reg register before starting boot operation. Overwrites register fields: io_mask_always_on, io_mask_end, io_mask_start, data_select_oe_end Refer to SD/eMMC PHY documentation for more details.
boot_phy_glpbk	in	9	static	**Boot PHY gate and loopback control** The value that will be written to PHY phy_gate_lpbk_ctrl_reg register before starting boot operation. Overwrites register fields: sync_method, sw_half_cycle_shift, rd_del_sel, gate_cfg_always_on. Refer to SD/eMMC PHY documentation for more details.
boot_phy_dllmst	in	1	static	**Boot PHY Master DLL control** The value that will be written to PHY phy_dll_master_ctrl_reg register before starting boot operation. Overwrites register fields: param_dll_bypass_mode. Refer to SD/eMMC PHY documentation for more details.
boot_phy_dllslv	in	16	static	**Boot PHY Slave DLL control** The value that will be written to PHY phy_dll_slave_ctrl_reg register before starting boot operation. Overwrites register field: clk_wr_delay. Refer to SD/eMMC PHY documentation for more details.
boot_setup_mode	in	1	static	**Boot setup method** 0 – input port values used to perform boot check 1 – descriptor mechanism used to perform boot check
boot_desc_addr	in	64	static	**Boot descriptor list address** Address points to the first descriptor of the boot configuration list.

eMMC Boot Status Interface

boot_ack	out	1	static	**Boot Acknowledge Status Received** This port is set high when the Boot Acknowledge Status is received during boot operation. This port should be ignored when boot_ack_en is low.
boot_done	out	1	static	**Boot acknowledge** This port is set high when operation is successfully completed. This port should be ignored when boot_err is set high.
boot_err	out	1	static	**Boot error** This port is high when error was detected during boot operation. The error type can be read from HRS registers (HRS36 — eMMC Boot Status Register and HRS03 — AXI ERROR Responses Register).

7.2.9 Auto-configuration descriptor mechanism Interface

Table 17. Auto-configuration descriptor mechanism settings

Name	Type	Width	Clk Src	Description
Auto-configuration descriptor mechanism Status Interface				
desc_mech_active	out	1	clk	**Auto-configuration in progress** This port is set high when mechanism detected speed mode change and is updating core registers with proper values.
desc_mech_err	out	1	clk	**Error acknowledge** This port indicates that error occurred during auto-configuration descriptor mechanism run (used also in the boot performance).

7.2.10 Host Settings Interface

Table 18. Host Settings

Name	Type	Width	Clk Src	Description
Host Settings				
s0_hwinit_srs16	in	32	static	**SRS16 initial value** Value set on software reset
s0_hwinit_srs17	in	32	static	**SRS17 initial value** Value set on software reset
s0_hwinit_srs18	in	32	static	**SRS18 initial value** Value set on software reset
s0_hwinit_srs19	in	32	static	**SRS19 initial value** Value set on software reset
s0_hwinit_srs24	in	32	static	**SRS24 initial value** Value set on software reset
s0_hwinit_srs25	in	32	static	**SRS25 initial value** Value set on software reset
s0_hwinit_srs26	in	32	static	**SRS26 initial value** Value set on software reset
s0_hwinit_srs27	in	32	static	**SRS27 initial value** Value set on software reset
CQ Settings				
hwinit_itcfmul	in	4	static	**Command Queuing Internal Timer Clock Frequency Multiplier** Initial value of HWINIT / RO register CQRS01.ITCMFMUL.
hwinit_itcfval	in	10	static	**Command Queuing Internal Timer Clock Frequency Value** Initial value of HWINIT / RO register CQRS01.ITCFVAL.
hwinit_itcfsel	in	4	static	**Command Queuing Internal Timer Clock Frequency Select** This is a system clock divider value, the Internal Clock Timer frequency equals to clk (when hwinit_itcfsel=0) and $clk/2^{(hwinit\_itcfsel-1)}$

7.2.11 Combo PHY Register Interface

Software has an access to the Combo PHY registers via the HRS04 (PHY Registers Address) and HRS05 (PHY Register Data Port). The PHY registers can be read or written following steps described below. Next step can start when previous is completed.

7.2.11.1 Read access

1. Software sets the PHY Register Address HRS04.
2. Software read the PHY Register Data Port HRS05 - the value in HRS05 is the value from PHY register.

7.2.11.2 Write access

1. Software sets the PHY Register Address HRS04.
2. Software sets the PHY Register Data Port HRS05 - value placed in HRS05 register updates PHY register during write access.

7.3 Memory Requirements

The SD/eMMC Host Controller requires four external SPRAMs which function and interface is described in section 7.2.5.1. All SPRAMs are identical and depends of the FIFODEPTH parameter. When the parameter equals 6, each SPRAM is 64x64-bits (512 bytes). When the parameter equals 8 (default), each SPRAM is 256x64-bits (2048 bytes). The parameter must not be set to anything except values described above.

Table 19. Memory requirements

Data Width	Address Depth	Data Buffer Size	SPRAM Size	Total Memory Size
64-bit	64	512 bytes	512 bytes	2k bytes
64-bit	256	2048 bytes	2048 bytes	8k bytes

7.4 DMA Specification

The SD Host Controller supports two DMA modes:

- SDMA — introduced and defined in SD Host Controller Standard Specification Version 1.00
- ADMA2 — introduced and defined in SD Host Controller Standard Specification Version 2.00

Note: The ADMA1, introduced and defined in SD Host Controller Standard Specification Version 2.00, becomes obsolete in the standard and it is not supported by the SD Host Controller.

The DMA mode for current transfer is selected via SRS10.DMASEL register and can be different for each consecutive data transfer. The Host Driver can change DMA mode when neither the Write Transfer Active (SRS09.WTA) nor the Read Transfer Active (SRS09.RTA) status bit are set.

The DMA supports 64-bit and 32-bit addressing modes.

Table 20 shows how to select the DMA engine and addressing mode by setting SRS10.DMASEL, SRS15.HV4E and SRS15.A64B register fields. Software driver must avoid settings from rows with grayed out "DMA Mode".

Table 20. DMA Mode

SRS10.DMASEL	SRS15.HV4E	SRS15.A64B	DMA Mode
0	0	0	SDMA 32-bit
		1	-
	1	0	SDMA 32-bit
		1	SDMA 64-bit
1	0	0	-
		1	-
	1	0	-
		1	-
2	0	0	ADMA2 32-bit
		1	-
	1	0	ADMA2 32-bit
		1	ADMA2 64-bit
3	0	0	-
		1	ADMA2 64-bit
	1	0	-
		1	-

The DMA transfer in each mode can be stopped by setting Stop at the Block Gap Request bit (SRS10.SBGR). The DMA transfers can be restarted only by setting Continue Request bit (SRS10.CR).

If an error occurs, the Host Driver can abort the DMA transfer in each mode by setting Software Reset for DAT Line (SRS11.SRCMD) and issuing Abort command (if a multiple block transfer is executing).

7.4.1 SDMA

The Simple (single-operation) DMA mode uses SD Host registers to describe the data transfer. The SDMA System Address (SRS00.SAAR or SRS22.DMASA1 / SRS23.DMASA2) register defines the base address of the data block. The length of the data transfer is defined by the Block Count (SRS01.BCCT) and Transfer Block Size (SRS01.TBS) values. There is no limitation on the SDMA System Address value — the data block can start at any address (address need not to be aligned to any boundary).

The SDMA engine waits at every boundary specified in the SDMA Buffer Boundary (SRS01.SDMABB or U2SDMABB) register.

Once buffer boundary is reached, the SD Host Controller stops the current transfer and generates the DMA interrupt. Software needs to update the SDMA System Address register to continue the transfer.

When the SDMA engine stops at the buffer boundary, the SDMA System Address register points the next system address of the next data position to be transferred. The SDMA engine restarts the transfer when the uppermost byte of the SDMA System Address register is written.

The SDMA engine does not use the ADMA Error Status (SRS21) register.

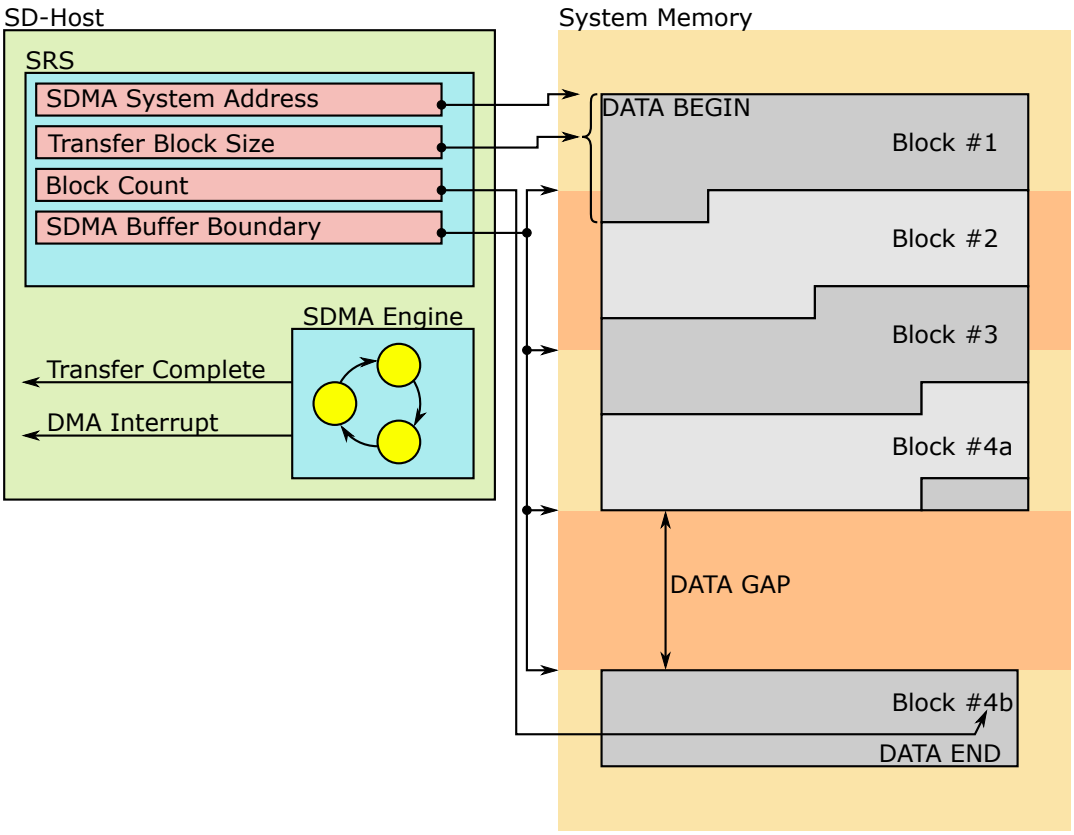

Figure 12. SDMA Block Diagram

7.4.2 ADMA2

7.4.2.1 Overview

The Advanced DMA Mode Version 2 (ADMA2) uses the Descriptors List to describe data transfers. The SD Host registers only defines the base address of the Descriptors List. The base addresses and sizes of the data pages are defined inside the descriptors. The ADMA2 mode is defined in the SD Host Controller Specification Version 2.00 (as well as in the more recent specification versions).

The SD Host supports ADMA2 in 64-bit or 32-bit addressing mode.

7.4.2.2 Data Pages

When in ADMA2 mode, the SD Host transfers data from data pages. Page is a block of valid data that is defined by a single ADMA2 Descriptor. Each ADMA2 descriptor can define only one data page. The starting address of the data page must by aligned to the 4 byte boundary (the 2 least significant bits set to 0) in 32-bit addressing mode, and to the 8 byte boundary (the 3 least significant bits are set to 0) in 64-bit addressing mode. The size of each data page is arbitrary and it depends on neither the previous nor the successive page size. It can also be different from the SD card transfer block size (SRS01.TBS).

7.4.2.3 ADMA2 Descriptors Table

The ADMA2 engine transfers are configured in a Descriptor List. The base address of the list is set in the ADMA System Address register (SRS22.DMASA1, SRS23.DMASA2), regardless of whether it is a read or write transfer. The ADMA2 Descriptor List consists of a number of 64-bit / 96-bit / 128-bit descriptors of different functions. Each descriptor can:

- Perform transfer of data page of specified size
- Link next descriptor address to an arbitrary memory location

Additionally, each descriptor has following flags:

- VAL — denoting a valid ADMA descriptor
- END — denoting last descriptor of the Descriptor List
- INT — denoting the ADMA interrupt request when the current descriptor action is complete

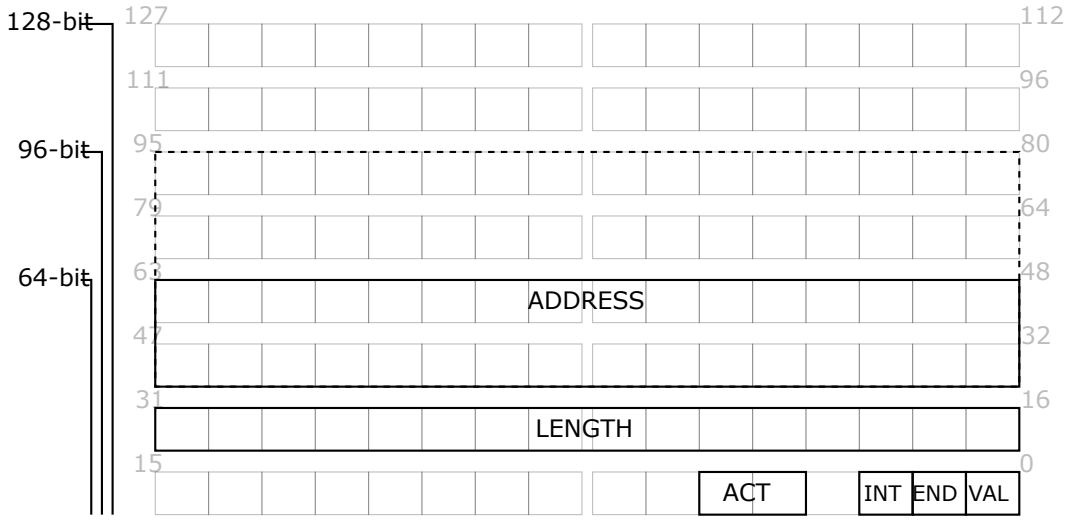

Figure 13. ADMA2 Descriptor Layout

Table 21. ADMA2 Descriptor Fields

Bit	Symbol	Description
95...32 / 63...32	**ADDRESS**	The field contains data page address or next descriptor list address depending on the descriptor type. When the descriptor is type TRAN, the field contains the page address. When the descriptor type is LINK, the field contains address for the next Descriptor List.
31..16	**LENGTH**	The field contains data page length in bytes. If this field is 0, the page length is 64kBytes.
5..4	**ACT**	The field defines the type of the descriptor. • `2'b00` (NOP) — no operation, go to next descriptor on the list; • `2'b01` (Reserved) — behavior identical to NOP; • `2'b10` (TRAN — transfer data from the pointed page and go to the next descriptor on the list; • `2'b11` (LINK) — go to the next descriptor list pointed by ADDRESS field of this descriptor.
2	**INT**	When this bit is set, the DMA Interrupt (SRS12.DMAINT) is generated when the ADMA2 engine completes processing of the descriptor.
1	**END**	When this bit is set, it signals termination of the transfer and generates Transfer Complete Interrupt when this transfer is completed.
0	**VAL**	When this bit is set, it indicates the valid descriptor on a list. When this bit is cleared, the ADMA Error Interrupt is generated and the ADMA2 engine stops processing the Descriptor List. This bit prevents ADMA2 engine runaway due to improper descriptors.

The figure below shows a schematic of the ADMA2 process:

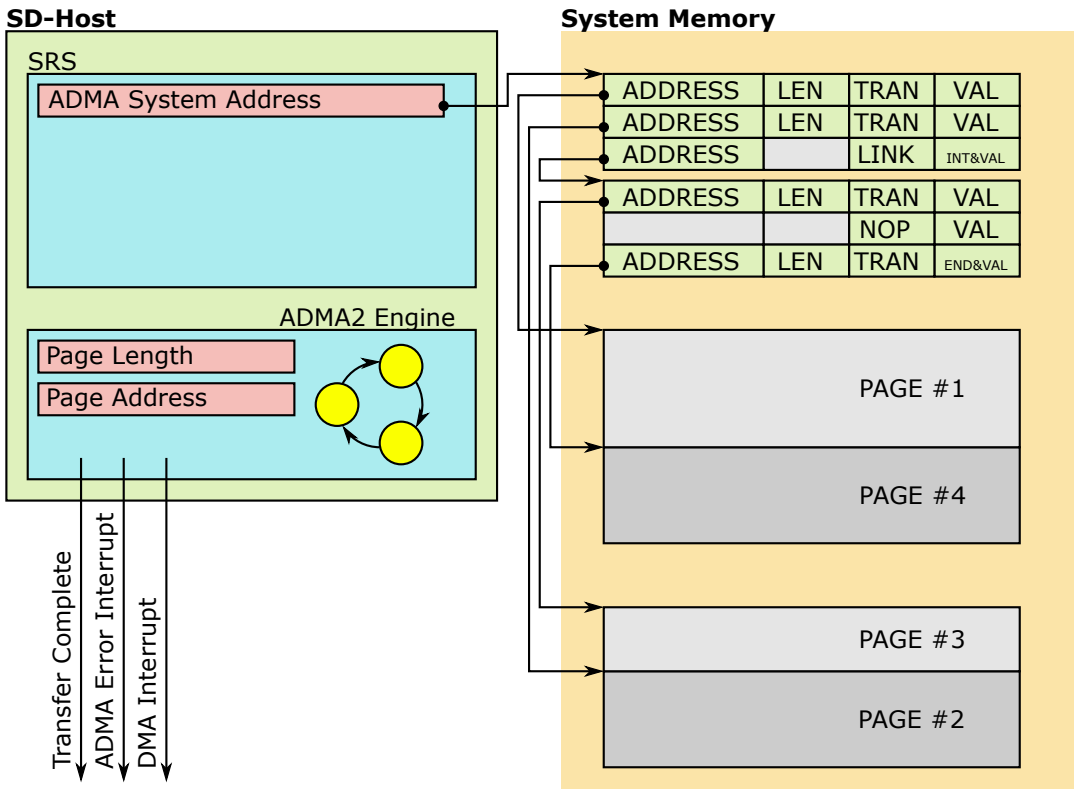

Figure 14. Block Diagram of ADMA2

7.4.3 DMA Sideband Specification

7.4.3.1 Overview

The purpose of the DMA sideband signals is to allow a system with, for example, one master DMA interface that does not use an advanced/master DMA mode of operation (SDMA or ADMA2) to have direct access to the DMA engine/status of the host controller. In this mode of operation transfers can be configured to not use the master DMA engine (SRS03.DMAE=0) mode of operation and a DMA transfer can be executed using continuous reads/writes to the SRS08 register using the slave port, with the additional sideband signals in indicating when transfers are complete or are in progress. The DMA sideband signals closely follow the operation of SRS12.BRR and SRS12.BWR bits and offer an additional connection to a DMA engine if required. Note. DMA sideband signaling is only operational in non UHS-II modes of operation.

Note. At all times in this section the nomenclature of *dma_(rd|wr)_req* and *dma_(rd|wr)_ack* is used. Each description applies to a either a read or write channel only, as the channels are mutually exclusive.

7.4.3.2 Signaling

DMA sideband signaling comprises of four additional signals known as *dma_wr_req*, *dma_rd_eq*, *dma_wr_ack* and *dma_rd_ack*. Depending on the card transfer direction (read or write) these signals indicate if a buffer can be written to or read from. Separate sideband signals exist for the read and write channels, with the read DMA sideband channel using the *dma_rd_req* and *dma_rd_ack* signals. The DMA sideband write channel uses the *dma_wr_req* and *dma_wr_ack* signals.

The DMA sideband signals are relatively straightforward in operation and comply with the following rules (Note. Example timing diagrams can be seen in section 7.4.3.5).

- *dma_(rd|wr)_req* is set high when a buffer can be written to or read from, *dma_(rd|wr)_ack* is low and a data transfer has been started (SRS03.DPS=1).
- *dma_(rd|wr)_ack* will be asserted after the configured number of bytes (register SRS01.TBS) have been transferred.
- *dma_(rd|wr)_req* will be set low when *dma_(rd|wr)_ack* is high.

Transfers to cards can vary in size from 1 to 2048 bytes by the setting in the SRS01.TBS register. The sideband signaling must follow the transfer size configured in this register. For an example case where a 512 byte write transfer has been configured with a block length of 3. The following interface interaction should occur:

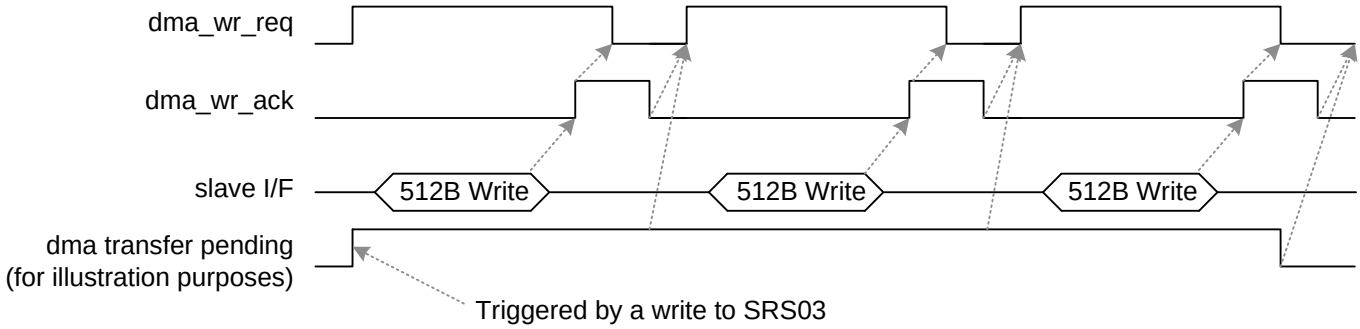

Figure 15. DMA Sideband Block Transfer

7.4.3.3 Fragmentation

No fragmentation of data transfers can occur. For example, if the host controller has again been configured for 3 x 512B write transfers then a 64B transfer, followed by a 512B transfer cannot occur. 3x512B transfers must occur and dma_wr_ack must be signaled after each transfer.

7.4.3.4 Error and Abort Scenarios

Under normal operation the DMA sideband signals with comply with the standard request/acknowledge protocol, where *dma_(rd|wr)_req* is kept high for a block size, until *dma_(rd|wr)_ack* is set high at the end of the block transfer. There are however some exceptions to this rule where *dma_(rd|wr)_req* can be set low before the end of a block transfer. These exceptions are at the following conditions:

- Abort - an abort command has been issued - see SRS03.CT for more details.
- Errors - an error condition on the card side such as a CRC error has occurred.

In these examples the transfer will be aborted and as a result the full transfer (block count * block size) will not be completed and *dma_(rd|wr)_req* will be set low shortly after the error or abort condition, regardless of the DMA current status. To resume from this abort situation the standard error recovery sequence should be executed, including an error/reset sequence for the external DMA engine.

7.4.3.5 DMA Sideband Timing

This subsection details timing specifics that are applicable to the DMA sideband operation.

The following figure gives an example of a terminating read burst, at the last read from the host controller:

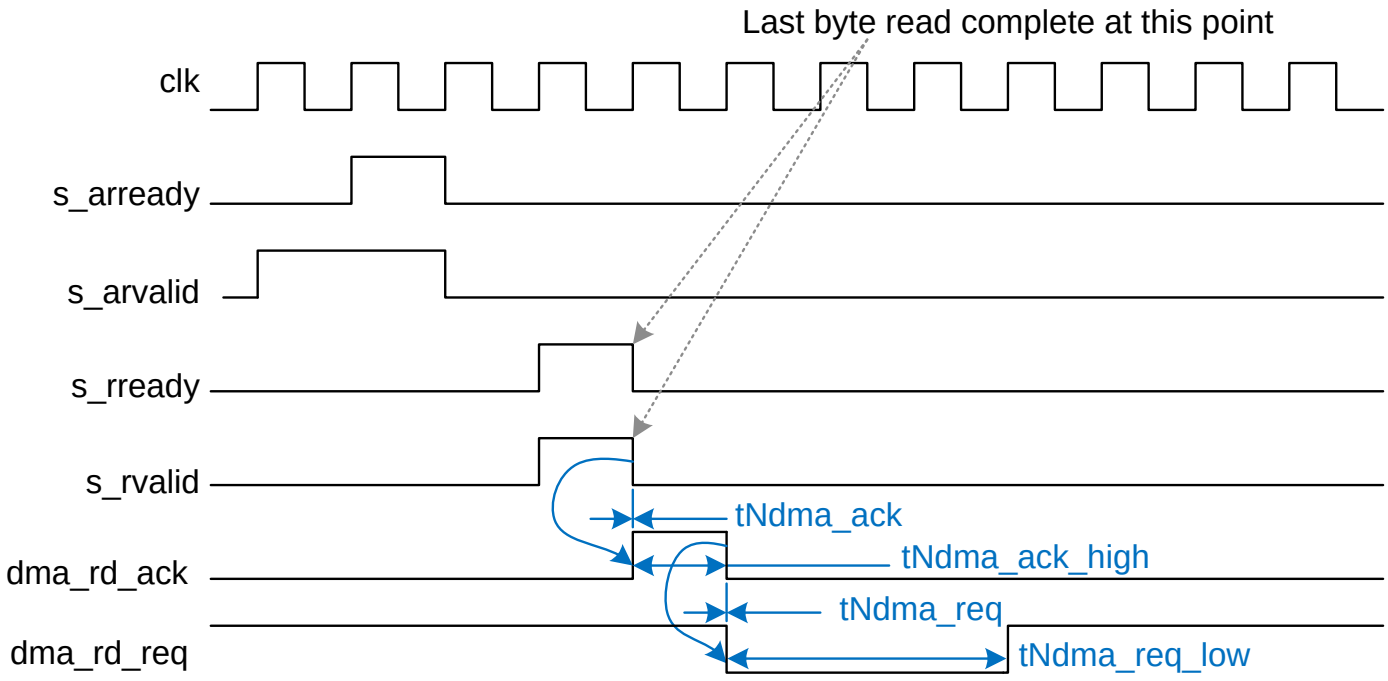

Figure 16. DMA Sideband Read Transfer Timing

The above diagram details the AXI specification only. When an APB interface is used instead of AXI, timing is identical and the *dma_rd_ack* and *dma_rd_req* signals follow the *pready* signal, rather than the *s_rready* and *s_rvalid* signals.

The following figure gives an example of a terminating write burst, at the last write to the host controller:

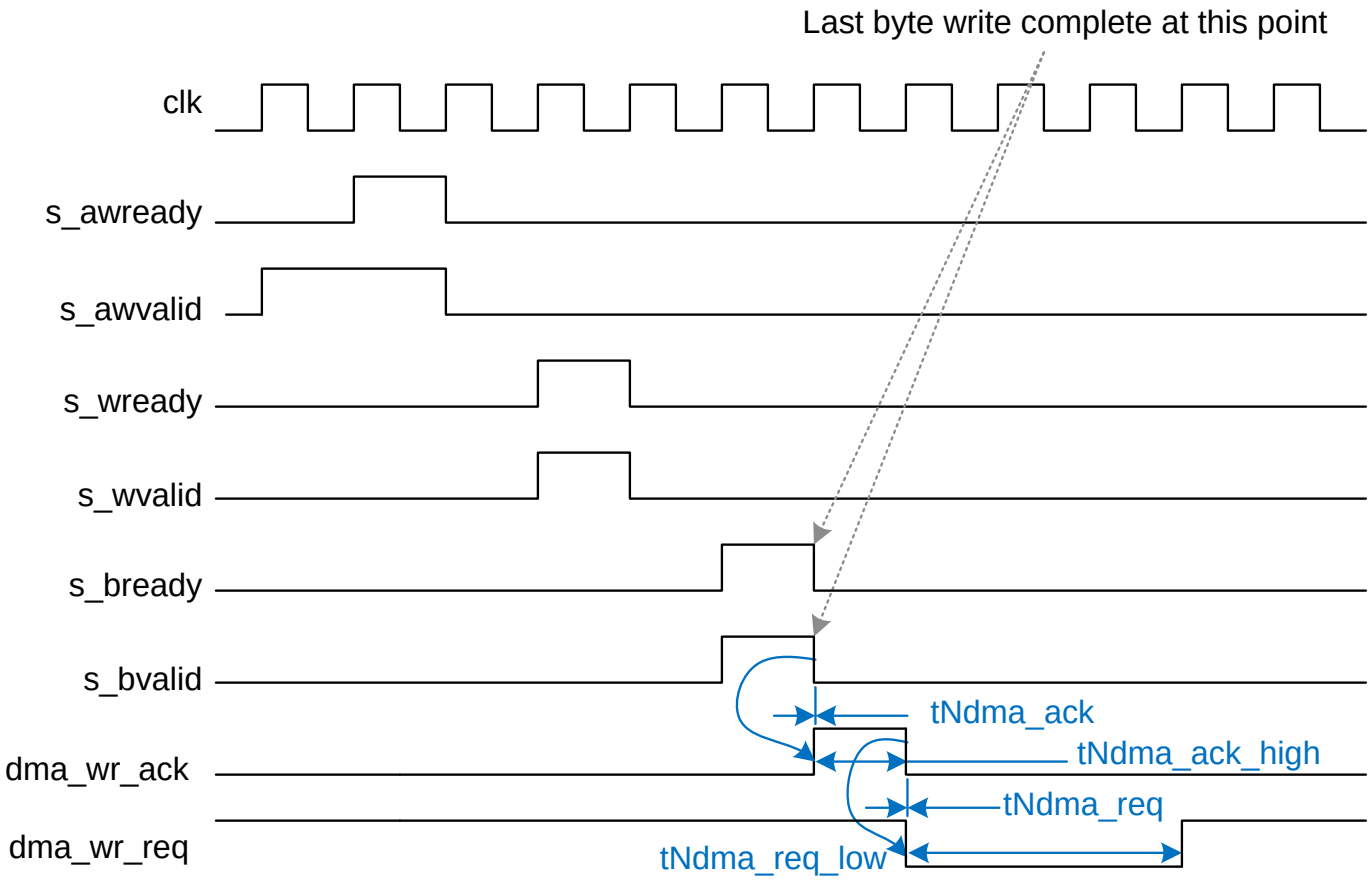

Figure 17. DMA Sideband Write Transfer Timing

The above diagram details the AXI specification only. When an APB interface is used instead of AXI, timing is identical and the *dma_wr_ack* and *dma_wr_req* signals follow the *pready* signal, rather than the *s_bready* and *s_bvalid* signals.

Table 22. DMA Sideband Timing

Parameter	Min	Max	Unit	Comments
tNdma_ack	0		CLK cycles	Number of cycles from last read or write to *dma_(rd|wr)_ack* high. No maximum value is specified but additional reads or writes should not take place until *dma_(rd|wr)_ack* has been signaled
tNdma_ack_high	1		CLK cycles	*dma_(rd|wr)_ack* high duration
tNdma_req	0	0	CLK cycles	Number of cycles from *dma_(rd|wr)_ack* high to *dma_(rd|wr)_req* low. The host controller will always respond to *dma_(rd|wr)_ack* on the clock cycle directly after *dma_(rd|wr)_ack* is set high.
tNdma_req_low	1		CLK cycles	Number of cycles that *dma_(rd|wr)_req* is low for. If more transfers are pending then *dma_(rd|wr)_req* will go high after 1 clock cycles, assuming *dma_(rd|wr)_ack* is low.

7.4.4 DMA Error

The Host Controller has following protection mechanisms on the DMA/Master interface:

- descriptor validation — this mechanism comes with SD Host Controller Standard. The Scatter-Gather DMA checks whether Valid Attribute is set 1 in the read descriptors. This mechanism is applicable for all variants of the descriptors including ADMA2 (SD) and Command Queuing (eMMC). If Valid Attribute is 0, the DMA stops transfer and reports DMA Error.
- AXI/AHB transaction error detection – this mechanism enables the DMA to stop any pending transfers and to report the DMA/Master interface error to system via status registers. The DMA does not send new requests, completes all already requested/queued transactions and stops when all transaction are finished without violating a bus protocol. In addition, in case of read transaction, read data is altered with 0x00 which in consequence protects Scatter-Gather DMA from interpreting potentially random data as a descriptor. On the bus transaction error, the host controller reports AXI Error (HRS03[3:0]) and may report DMA Error (SRS12[25]).

7.5 Timings on SD/eMMC Interface

7.5.1 Overview

Host Controller timings on SD and eMMC interface requires specific Host Controller and Combo PHY register settings. The settings calculation is done by the script (section 9.) based on the operation conditions. Software Driver / User Application is responsible for updating the specific registers according to the output calculations.

The settings calculation is applicable for all the speed modes. The higher speed modes requires additional preparation described in section 7.5.2.

7.5.2 Tuning

SD Host Controller / Physical Layer Standard Specifications define UHS-I SDR104 speed mode, and the eMMC 5.1 Standard Specification defines HS200 and HS400 speed modes. The modes are specific as those do not define position of the valid data window in respect to the clock, and just define the valid window width and assuring that it will send from the device within 0 to 2 clock cycles.

The modes are expected to be operational after the procedure adjusts sampling point to the center of the floating valid data window. The procedure is similar for both standards. It has 40 iterations where Host sends the read tuning pattern command. In each iteration the data sampling point is tuned (increased incrementally). The goal is to cover fully at least one SDCLK clock period changing the sampling clock phase from 0 to at least 360 degree with fixed step.

Each step begins of setting the sampling point, then host controller sends command, then receives response and read data pattern.

When step ends, the controller knows whether it is successful iteration (host controller was capable of receiving valid response and data pattern) or unsuccessful iteration (host detected a protocol error).

Based on post-tuning statistics of successful iterations, the hardware/software is able to set the sampling point close to the middle of the valid data window making having

The tuning sequences looks slightly different with SD (section 7.5.3) and eMMC (section 7.5.4).

Figure 18 shows moments when the Host/PHY settings update is required during transition between speed modes.

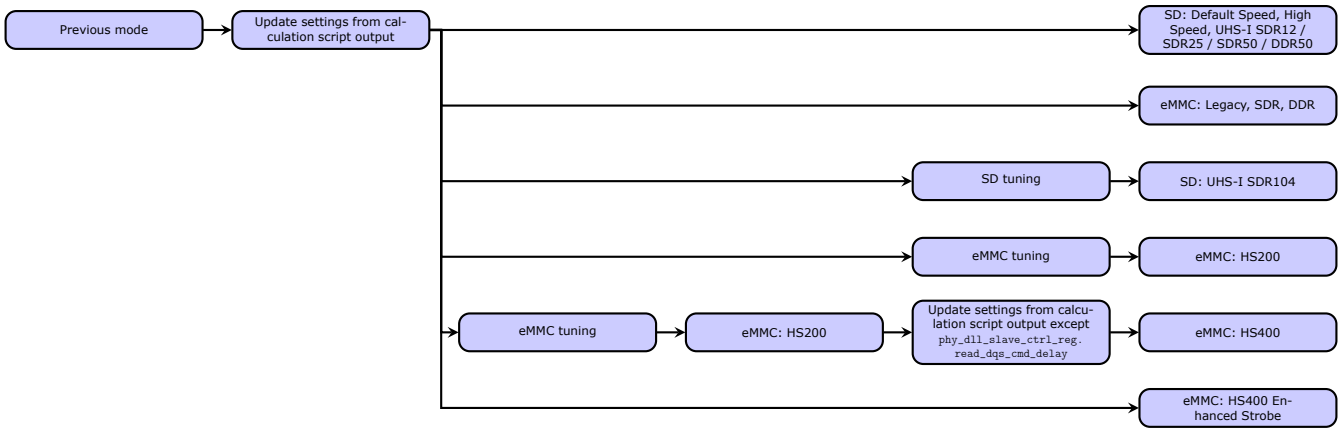

Figure 18. Settings update

7.5.3 Tuning sequence for SD

The sequence is shown in figure 19. Software sends 40 times CMD19 and hardware adjusts data sampling point (PHY DLL settings) and gathers pass/fail statistics. Based on these statistics, host either applies sampling point in the middle of the widest passing iterations set, or reports that it is not possible as none of iteration passed.

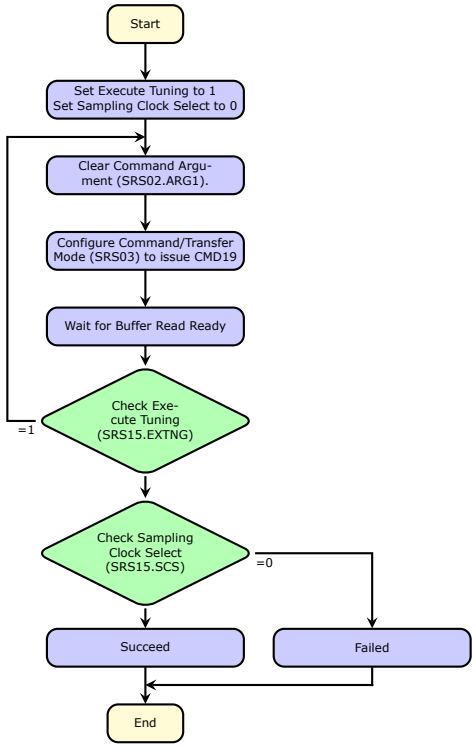

Figure 19. Tuning sequence for SD

7.5.4 Tuning sequence for eMMC

The sequence is shown in figure 20. It consists of 40 data tuning patter transfer commands (CMD21). For each command, software checks command and data statuses and compares received data block with the Tuning Block Pattern defined in eMMC 5.1 Standard section 6.6.5.1.

Host Controller may report a protocol error in any of the iterations. In that case, the software runs basic Error Recovery

which consists of setting Software Reset for CMD (SRS11.SRCMD) and Software Reset for DAT (SRS11.SRDAT) to 1. Then, software waits till the resets are completed.

Having gathered iterations result, software seeks for the widest range of iterations that successfully ended. Software considers iteration results as wrapped through 39 to 0. For example, if the succeed results are in iteration 0-4 and 38-39, then software identified the center in position 1.

The settings for the center of the data window are updated the DLL settings (phy_dll_slave_ctrl_reg).

If non of iteration succeeded, software transits the host and device to other mode, lower speed mode.

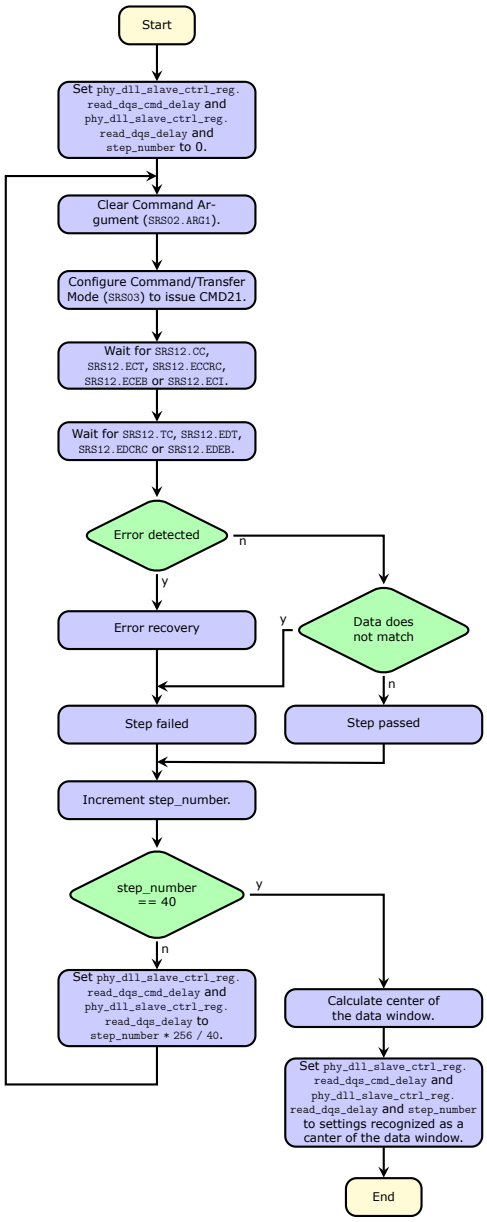

Figure 20. Tuning sequence for eMMC

7.6 Software sequences

The SD Host Controller Standard describes software sequences in Chapter 3. This section adds an information regarding the error conditions and how to react on them.

7.7 Error handling

7.7.1 Generic Operation Error Recovery

Figure 27: Generic Operation Error Recovery procedure shows an example software procedure to handle errors detected by the SD/eMMC Host Controller. Once an error is detected and reported in any interrupt status register, software identifies whether the error is recoverable or non-recoverable. If the error is recoverable, software follows the procedure requesting Software Reset for CMD and/or DAT logic. Main purpose of this procedure is to clear logic affected by an error and prepare to a next transaction (e.g. to redo interrupted read/write operation).

This procedure probably won't help in case of a non-recoverable error. The non-recoverable error usually requires a full software/hardware reset and a device power cycle.

The same procedure can be used with SD, SDIO and eMMC device.

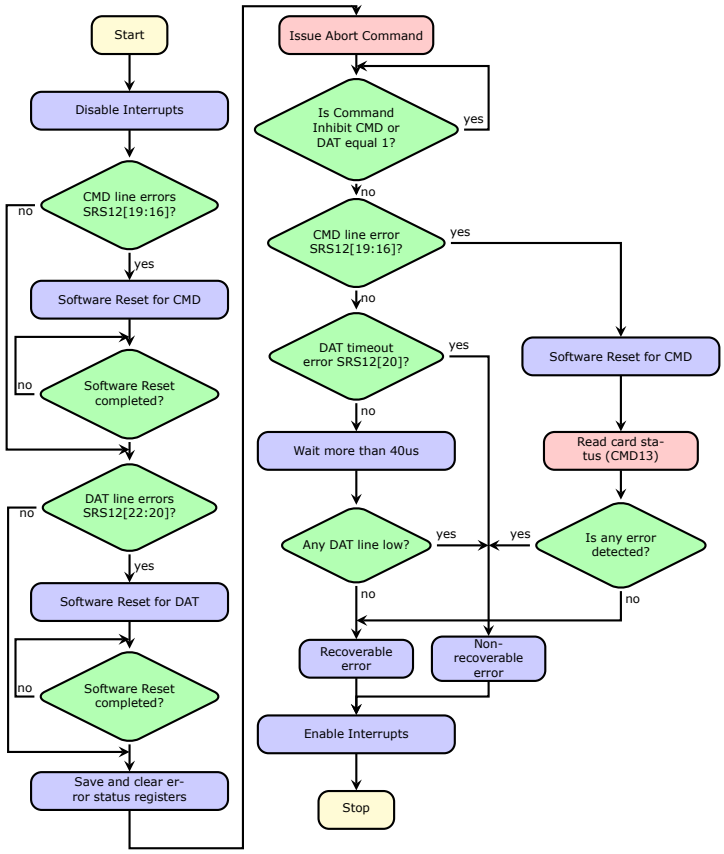

Figure 21. Generic Operation Error Recovery procedure

7.7.2 Command Queuing Error Recovery

Figure 28:Command Queuing Error Recovery procedure shows an example Command Queuing error recovery procedure required to get the host controller and a device from error state and prepare for following operations.

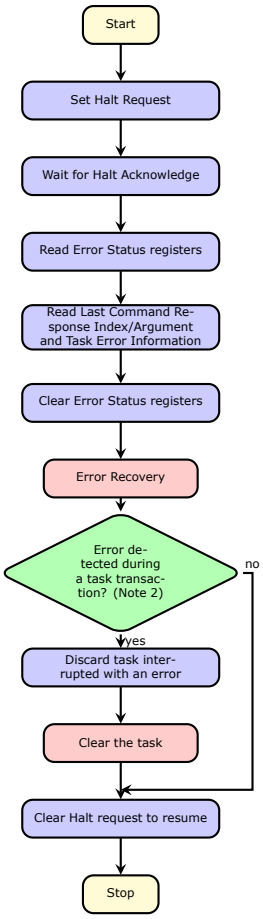

Figure 22. Command Queuing Error Recovery procedure

Note: The SD Host Controller always replies to the halt request by setting halt acknowledge. Software needs to wait for the acknowledge before running Generic Operation Error Recovery. The acknowledge is signalized after unpredicted amount of time — the time required to finish any pending operation. The acknowledge is set when command and data path reports its statuses (whether those are successfully completion or interrupted by an error).

Note 2: The discard / task clear is required for the task interrupted by an error.

The Command Queuing reuses protection mechanisms built into the SD Host Controller. It also adds an automatic error detection for the R1 device response.

Following is a full list of errors that can be detected by the Command Queuing:

- R1 response error
- Command errors (Index/CRC/End Bit/Timeout)
- Data transfer errors (CRC/End Bit/Timeout)
- DMA error
- Master Interface Error

Once any of those error is detected, an internal logic can stop, reports type of error and waits for Command Queuing Error Recovery procedure.

Command Queuing is capable of sending CMD13 SQS (Send Queue Status) to a device during the data transaction. If this feature is enabled (i.e. CQRS16.CQSSCBC > 0) and a data transaction is in-progress, the controller sends CMD13 SQS concurrently to the data transaction and reports possible CMD13 SQS error in the auto command error status registers (SRS15[4:1]).

SD/eMMC Host Controller may report Current Limit Error but this error is a command queuing task agnostic and won't interrupt any transaction. This error is application specific and thus should be classified as non-recoverable error. Software reaction must be adequate and follow the application assumptions.

SD/eMMC Host Controller does not use mechanism of Response Error Check when Command Queuing is enabled.

Software may either poll the status registers or rely on the interrupts generated by the Host Controller. If error occurrence must raise a system interrupt, software enables the required interrupt status registers before task submitting. The interrupt generation is configured by the following registers:

- Error Status Enable (SRS13[31:16])
- Error Signal Enable (SRS14[31:16])
- AXI Error Response (HRS03[19:16], HRS03[11:8])
- Response Error Detected Status Enable (CQRS05[2])
- Response Error Detected Signal Enable (CQRS06[2])

Table 23. Error cases

Operation	**Event** (any event)	**Consequence**
Read Task Descriptor before CMD44 (Task Queuing)	Master Error	Halt and report Task ID in Response Mode Error
Read Task Descriptor before CMD46 / CMD47 (Task Execution)	Master Error	Halt and report Task ID in Data Transfer Error
Read Task Descriptor before DCMD (Task Execution)	Master Error	Halt and report Task ID in Response Mode Error
CMD44	Command Error	Halt and report Task ID in Response Mode Error
CMD45	Command Error	Halt and report Task ID in Response Mode Error
CMD13 while controller in idle	Command Error	Halt and Task ID is not reported
CMD13 during CQ transaction	Command Error	Halt and Task ID is not reported
CMD46 / CMD47	Command Error Data Error DMA Error	Halt and report ID in Data Transfer Error
DCMD	Command Error	Halt and report ID in Response Mode Error
DCMD	Data Timeout Error	Halt and report ID in Response Mode Error

Data Transfer Error in Task Error Information is:

- set to 1 when error appears during CMD46 or CMD47
- clear to 0 when error appears during CMD44, CMD45 or DCMD

Response Mode Error in Task Error Information is:

- set to 1 when error appears during CMD44, CMD45 or DCMD
- clear to 0 when error appears during CMD46 or CMD47

Task Error Information does not changed when CQ is disabled or halted.

7.7.3 Combo PHY Underrun/Overflow Error Recovery

Combo PHY Underrun and Overflow flags indicate the PHY FIFO violation. Once this error is reported, Software must take an action and do following:

- Set Software Reset for CMD and Software Reset for DAT.
- Set PHY_SW_RESET (dll_rst_n) to 0 (assert reset)
- Set PHY_SW_RESET (dll_rst_n) to 1 (release reset)
- Run the error recovery procedure from Section 9.2.1

7.8 Command Queuing

7.8.1 Overview

The Command Queuing (CQ) is a feature available in the eMMC standard since version 5.1. It allows to issue multiple (up to 32) tasks by setting Task Doorbell register bits. Unless no errors appear during the transfer, the host is able to read commands configuration from the Task Descriptors List using internal DMA and run operations.

The CQ Engine runs all the operations sequentially in order the tasks were submitted. If the host is ready for task execution but the next task in order is not ready, it takes the first one that is ready to minimize idle time.

The execution order of the tasks considers tasks priority applied during Task Doorbell write and the device readiness for the task execution.

The Command Queuing is compliant with the eMMC standard. For more details please see the eMMC5.1 Standard, subsection *Annex B Host Controller Interface for Command Queuing*.

7.8.2 Optional modes of operation

- DCMD is executed only when other tasks are not executed even though the standard allows to send the DCMD during the data transfer when the CMD Timing field is set.

7.8.3 Task execution order

The Command Queuing Engine process and executes task in order there were submitted. Two rules define task order:

- If the Task Doorbell register is filled with more than one bit set high, the lower significant bit gets higher priority.
- If there are more than one writes to the Task Doorbell register, the task submitted with earlier access gain higher priority.

7.9 eMMC Boot

7.9.1 Overview

The boot block uses internal SDMA engine to execute and control data stream from the eMMC memory device to system memory utilizing eMMC feature called Boot.

The module initializes the host and PHY with settings/configuration provided from outside through the dedicated ports (7.2.8) or through descriptor list (Auto-configuration descriptor mechanism - chapter 7.10). Modules initialized through dedicated ports are ready to handle eMMC device, so the device is powered up and supplied with the SD clock. Once backend interface is ready, the host sends the Boot trigger (either CMD line goes to low or the CMD0 is sent) and waits for either Boot Acknowledge (optional) or first data block (which starts the data stream). Each received data block is validated (as during normal read data transfer operation) and correct data block is transferred to the system memory. When block is written to the system memory, the block counter is decremented.

Transfer ends when the block counter reaches 0. The host sets CMD line back to 1, or sends CMD0 with argument 0x00000000.

The module starts software reset of SD Host HRS00.SWR preparing the host to be initialized by CPU once boot is completed.

The boot performance initialized through auto-configuration descriptor mechanism requires preparing list of descriptors in system memory. Each descriptor contains settings for the host or PHY registers which form same performance of the boot as described above for dedicated ports.

The boot status can be checked after performance is done (boot_done set high or boot_err set high if error occured) by reading the eMMC Boot Status Register (HRS36).

7.9.2 List of descriptors required for the boot performance

Table 24. Boot descriptor list

Nr	Reg Name	Address	Field	CMD type	Comment
Boot triggered by sending CMD0					
1	SRS09	0x0224	CI	READ_BLOCK	
2a	SRS00	0x0200	SAAR	WRITE	Descriptor is omitted if SRS01.BCCT is set (see next descriptor)
2b	SRS01	0x0204	TBS, BCCT	WRITE	
3	SRS10	0x0228	BVS, BP, EDTW, DTW	WRITE	
4	SRS11	0x022C	DTCV, SDCFSL, SDCFSH, ICE	WRITE	
5	SRS15	0x023C	A64B, HV4E	WRITE	
6	SRS22	0x0258	DMASA1	WRITE	
7	SRS23	0x025C	DMASA2	WRITE	
8	SRS13	0x0234	ECL_SE, EDEB_SE, EDCRC_SE, EDT_SE, TC_SE, CC_SE	WRITE	
9	HRS06	0x0018	EMM	WRITE	
10	HRS03	0x000C	AER_SENBS, AER_SENBD, AER_SENRS, AEN_SENRD	WRITE	
11	SRS11	0x022C	ICS	READ_BLOCK	
12	HRS16	0x0040	All fields	WRITE	
13	HRS07	0x001C	RW_COMPENSATE, IDELAY_VAL	WRITE	
14	HRS10	0x0028	HCSDCLKADJ	WRITE	
15	HRS09	0x0024	PHY_SW_RESET	WRITE	
16	HRS09	0x0024	RDDATA_EN, RDCMD_EN, EXTENDED_WR_MODE, EXTENDED_RD_MODE	WRITE	PHY_SW_RESET set as previous. This descriptor can't be merged with previous one.
17a	HRS04	0x0010	0x2004	WRITE	DQSTM register address
17b	HRS05	0x0014	USE_EXT_LPBK_DQS, USE_LPBK_DQS, USE_PHONY_DQS, USE_PHONY_DQS_CMD	WRITE	DQSTM register required fields
18a	HRS04	0x0010	0x2000	WRITE	DQTM register address
18b	HRS05	0x0014	IO_MASK_ALWAYS_ON, IO_MASK_END, IO_MASK_START, DATA_SELECT_OE_END	WRITE	DQTM register required fields
19a	HRS04	0x0010	0x2008	WRITE	GLPBK register address
19b	HRS05	0x0014	SYNC_METHOD, SW_HALF_CYCLE_SHIFT, RD_DEL_SEL, GATE_CFG_ALWAYS_ON	WRITE	GLPBK register required fields
20a	HRS04	0x0010	0x200C	WRITE	DLLMC register address
20b	HRS05	0x0014	DLL_BYPASS_MODE	WRITE	DLLMC register required field
21a	HRS04	0x0010	0x2010	WRITE	DLLSC register address
21b	HRS05	0x0014	CLK_WR_DELAY	WRITE	DLLSC register required fields

22	HRS09	0x0024	PHY_SW_RESET	WRITE	
23	HRS09	0x0024	PHY_INIT_COMPLETE	READ_BLOCK	
24	SRS11	0x022C	SDCE	WRITE	ICE and SDCFSH set as previous
25	––	——	wait cycles value	WAIT	Set delay for clock to command enable
26	SRS03	0x020C	DPS, MSBS, DTDS, BCE, DMAE	WRITE	
27	SRS02	0x0208	ARG1	WRITE	16'hFFFF_FFFA
28a	SRS12	0x0230	TC	READ_NONBLOCK	
28b	HRS03	0x000C	AER_BS, AER_BD, AER_RS, AER_RD	READ_BLOCK	
29	SRS12	0x0230	TC, CC	WRITE	
30	SRS03	0x020C	All field cleared	WRITE	
31	SRS02	0x0208	ARG1	WRITE	16'h0000_0000
32	SRS12	0x0230	CC	READ_BLOCK	Internally prepared error checks for this register
33	HRS00	0x0000	SWR	WRITE	
34	HRS00	0x0000	SWR	READ_BLOCK	
Nr	**Reg Name**	**Address**	**Field**	**CMD type**	**Comment**
			Boot triggered by driving CMD line Low		
1-28b					Steps same as above.
29	HRS00	0x0000	SWR	WRITE	
30	HRS00	0x0000	SWR	READ_BLOCK	

Note: Ports which are required to perform boot with auto-configuration descriptor mechanism: boot_en, boot_ack_en, boot_method, boot_timeout_ack, boot_setup_mode, boot_desc_addr

7.9.3 Limitations

- All boot parameters are provided through dedicated external ports.
- The boot parameters have to be stable after hardware reset is released (e.g. during boot operation).
- The software cannot write core registers when the boot operation is active. The host behavior is unpredictable in that case.
- Transfer length is limited – the block count is 32-bit register. This limits the block number to 4294967296. Assuming block size is 512 bytes, the maximum transfer size is 2199023255552 bytes (2TB – 512B).
- Boot does not work in HS200 and HS400 mode (eMMC standard limitation).
- Boot can be triggered only by the hardware reset.
- The host behavior can be unpredicted if any configuration value is out of range.
- Delays set by auto-configuration descriptor mechanism (descriptors with command WAIT) takes longer as time of descriptor read need to be added to them.

7.10 Auto-configuration descriptor mechanism

7.10.1 Overview

The auto-configuration descriptor mechanism performs Pre-initialization sequence (chapter 9.1) automatically - with-outh software interference. This mechanism requires two list of descriptors prepared in the system memory. First list contains link descriptors for each available speed mode. Second list contains settings for pre-initialization sequence pointed by link descriptor.

HRS42.DESCMECH_EN and HRS40 registers (with HRS41 register if 64-bit address in use) need to be set to enable auto-configuration descriptor mechanism. If change of speed mode is detected than mechanism blocks software from writing to core registers and starts reading descriptors with ADMA engine. desc_mech_active signal is set high and indicates that auto-configuration is in progress. It automatically updates Host and PHY registers with new provided values.

If error occurs during auto-configuration (desc_mech_error signal set high) than mechanism will stop performance and error type will be set in HRS43.ERROR_VAL register field. Re-enabling (disable-enable sequence) mechanism in the HRS40 register is required for error recovery and to clear HRS43 register and re-use auto-configuration descriptor mechanism.

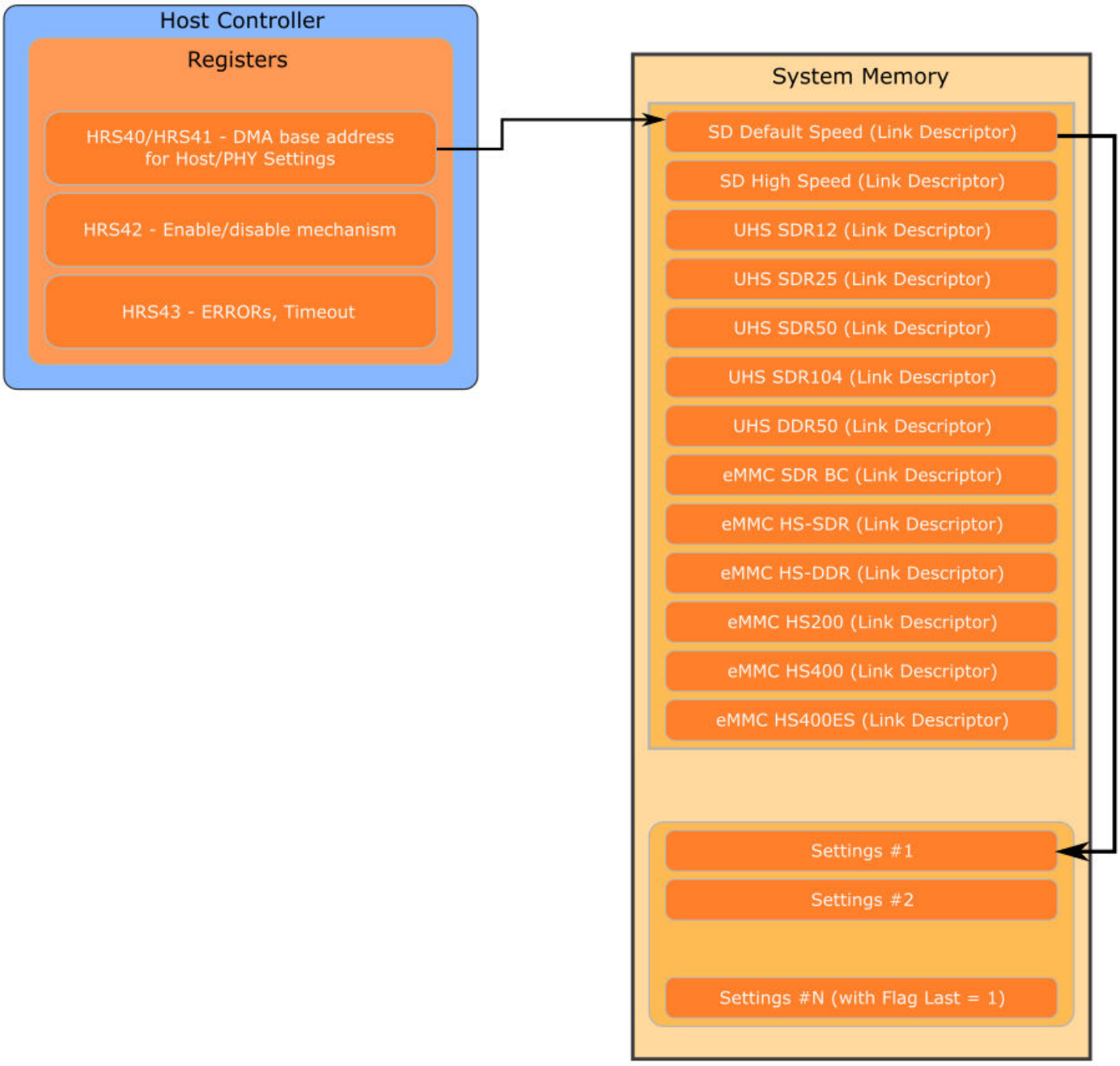

Figure 23. Descriptor lists in system memory

7.10.2 Descriptor content

Lenght of descriptors prepared for auto-configuration descriptor mechanism is always 128-bits. Other lengths are forbidden. Link descriptors contain fields same as described in chapter 7.4.2.3 ADMA2 Descriptors Table - ACT field is set as LINK, LENGTH field is set to ’0’ and ADDRESS field provides address to descriptor list with settings.

Descriptor with settings contains fields as shown below.

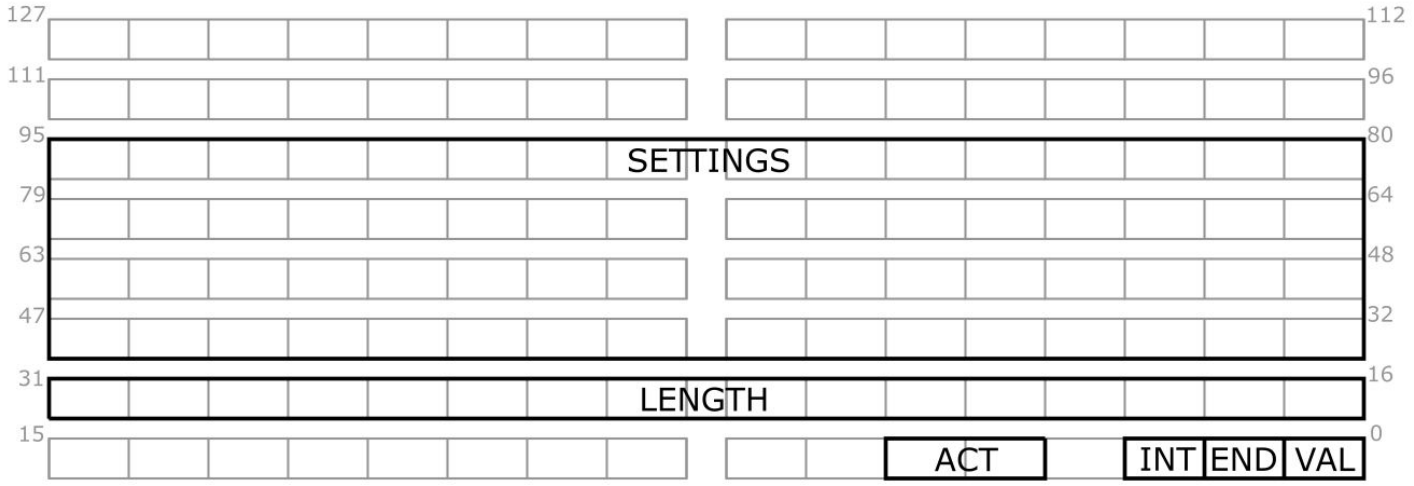

Figure 24. Descriptor with settings Layout

VAL, END and INT fields have same purpose as described in chapter 7.4.2.3 ADMA2 Descriptors Table. Descriptor is not pointing to any data in memory, so LENGTH is set to '0'. Address for next descriptor with settings is automatically incremented.

ACT field in descriptors with setting can be set to any value.

Descriptor 'SETTINGS' field contains:

- command type (CMD)
- register address (REG_ADDRESS)
- new value/expected value of register (EX_VALUE/NEW_VALUE)
- mask/bit range (MASK)

Table 25. Descriptor 'SETTINGS' field

Field name	Bit possition	Width	Description
CMD	[63:62]	2	Commands types: 00b - WRITE 01b - READ_BLOCK 10b - READ_NONBLOCK 11b - WAIT
MASK	[61:52]	10	Bit range for expected read value MASK [4:0] – lower bit range MASK [9:5] – upper bit range Same value for lower and upper bit range selects one bit from 32 bits in the expected read value.
REG_ADDRESS	[47:32]	16	Address of required register.
EX_VALUE/NEW_VALUE	[31:0]	32	For write command it is a new value to be written to pointed register. For read commands it is expected value.

7.10.3 Descriptor list with pre-initialization sequence example

Table 26. Descriptor list example with required registers set

Nr	Reg Name	Address	Field	CMD type	Comment
1	HRS09	0x0024	PHY_SW_RESET	WRITE	
2	HRS09	0x0024	RDDATA_EN, RDCMD_EN, EXTENDED_WR_MODE, EXTENDED_RD_MODE	WRITE	PHY_SW_RESET set as previous. This descriptor can't be merged with previous one.
3a	HRS04	0x0010	0x2004	WRITE	DQSTM register address
3b	HRS05	0x0014	USE_EXT_LPBK_DQS, USE_LPBK_DQS, USE_PHONY_DQS, USE_PHONY_DQS_CMD	WRITE	DQSTM register required fields
4a	HRS04	0x0010	0x2000	WRITE	DQTM register address
4b	HRS05	0x0014	IO_MASK_ALWAYS_ON, IO_MASK_END, IO_MASK_START, DATA_SELECT_OE_END	WRITE	DQTM register required fields
5a	HRS04	0x0010	0x2008	WRITE	GLPBK register address
5b	HRS05	0x0014	SYNC_METHOD, SW_HALF_CYCLE_SHIFT, RD_DEL_SEL, GATE_CFG_ALWAYS_ON	WRITE	GLPBK register required fields
6a	HRS04	0x0010	0x200C	WRITE	DLLMC register address
6b	HRS05	0x0014	DLL_BYPASS_MODE	WRITE	DLLMC register required field
7a	HRS04	0x0010	0x2010	WRITE	DLLSC register address
7b	HRS05	0x0014	CLK_WR_DELAY	WRITE	DLLSC register required fields
8	HRS09	0x0024	PHY_SW_RESET	WRITE	
9	HRS09	0x0024	PHY_INIT_COMPLETE	READ_BLOCK	

7.11 Low/High Voltage Signaling

Prior to SD Physical Layer Standard version 6.0, SD interface operates in 3.3V signaling (in mandatory non-UHS mode) and 1.8V (in optional UHS-I mode). Host Controller supporting 3.3V is considered by SD Physical Standard version 6.0 and later as High Voltage Signaling (HVS) SD Host Controller.

SD Physical Layer Standard version 6.0 introduced a variant where 3.3V signaling is not supported and only 1.8V signaling is required. Host Controller supporting 1.8V only is considered by the standard as Low Voltage Signaling (LVS) SD Host Controller.

Host Controller has Low Voltage Signaling Host (SRS17.LVSH) capability register that informs whether the implementation is LVS (1) or HVS (0). The register default value is 1. Upon integration, based on application requirements this field may require update.

7.11.1 LVS

LVS Host Controller support LVS cards and does not support HVS card. All cards based on SD Physical Layer Standard version 6.0 and all cards base on SD Physical Layer version 6.0 or above but without LV (Low Voltage) or A2 (Application Performance Class 2) logo are not supported.

If lack of backward compatibility is not an issue, SDA (standard body) decided to release LVS to simplify IO PADs requirements.

LVS requires an initialization sequence to enable SD card into LVS mode.

LVS sequence runs following steps:

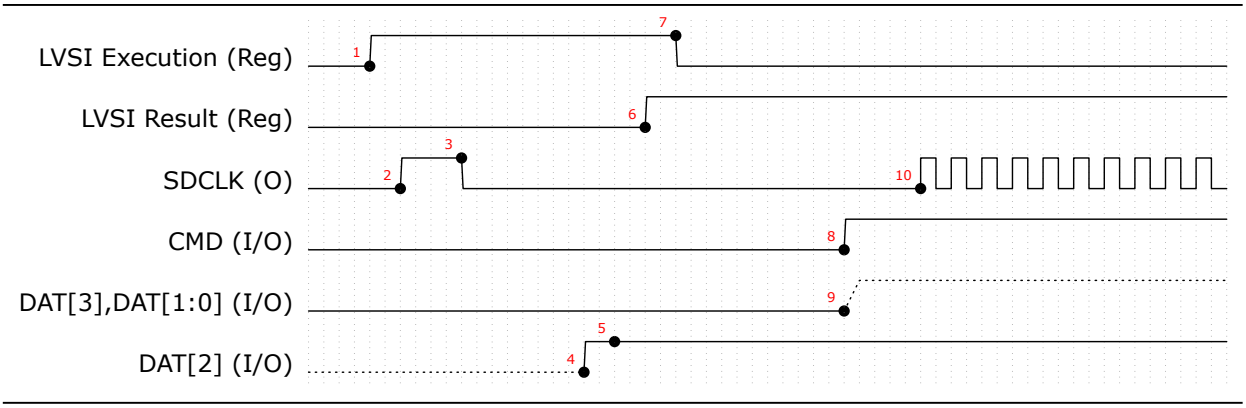

Figure 25. LVS Sequence

- Software confirms the Low Voltage Signaling Host capability register (SRS17.LVSH) is high
- Software sets HRS09.LVSI_CNT and HRS09.LVSI_TCKSEL calculating SDCLK pulse with of minimum 15us
- Software enables Internal Clock Enable (SRS11.ICE) and waits for Internal Clock Stable (SRS11.ICS)
- Software Enables LVSI Execution (SRS15.LVSIEXEC) (1)
- Host drives CMD, DAT[3], DAT[1:0] low and pulls DAT[2] down
- Host Controller generate one clock pulse of the width defined in HRS09 (2–3)
- Software waits at least 5010us (3–5)
- Card drive DAT[2] to High in 5ms after end of the clock fall edge (3–4)
- Software checks LVSI Result (SRS09.LVSIRSLT) register (6)
- If LVSI Result = 1 (success)
  - Host pull ups CMD, DAT lines (8,9)
  - Software enables SDCLK (10)
- If LVSI Result = 0 (fail)
  - Software disables card power supply

7.11.2 HVS

HVS Host Controller support both HVS and LVS card.

If target application implements HVS Host Controller, SRS17.LVSH shall be set to 0. Change the capability register field value:

- assign 0 to s0_hwinit_srs17[31] input, and
- optionally modify RTL source code — open cdns_sdhc_biu_srs_rtl.v file and replace srs17_lvsh_r assignment.
  File before modification:

```
always @(posedge clk or negedge rst_n)
begin
  if (rst_n == 1'b0)
  begin
    (...)
    srs17_lvsh_r <= 1'h1;
  end
  else
```

File after modification:

```
always @(posedge clk or negedge rst_n)
begin
  if (rst_n == 1'b0)
  begin
    (...)
    srs17_lvsh_r <= 1'h0;
  end
  else
```

7.11.3 Reference

More information can be found in:

- SD Host Controller Specification ver. 6, section 1.18
- Low Voltage Addendum ver. 100

7.12 Low-Power Features

Master SD card side clock can be switched off.

Card Clock can be switched off independently.

7.13 Design For Test (DFT) Features

The Host Controller is fully scanable, there are no scan signal and functionality inserted in RTL.

7.14 Debug Features

The CDNS_SDHC HRS32 enables read current state of most FSMs. Following table provides addresses where each FSM is mapped.

Table 27. FSM addresses

FSM	Address
fsm_deb	0x0000
fsm_fin	0x0001
fsm_buffer	0x0002
fsm_busy	0x0003
fsm_wrx	0x0004
fsm_rdx	0x0005
fsm_xfr	0x0006
fsm_biu	0x0007
fsm_abort	0x0008
fsm_adma	0x0009
fsm_dctrl	0x000a
fsm_dma_datapath	0x000b
fsm_dma_ctrl	0x000c
fsm_tune_ctrl	0x000d
fsm_step	0x000e
fsm_status	0x000f
fsm_read_pattern	0x0010
fsm_boot	0x0011
fsm_cqe	0x0012
fsm_exec	0x0013
fsm_citimer	0x0014
fsm_queue1	0x0015
fsm_axi2ahblite	0x0016

Table 27. FSM addresses (cont'd)

FSM	Date
fsm_ip	0x2000
fsm_cmd	0x2001
fsm_cmd_ctrl	0x2002
fsm_cmd2	0x2003
fsm_block	0x2004
fsm_inf_xfer_end	0x2005
fsm_inf_xfer_rend	0x2006

7.15 BIST Features

There are no BIST Features in the Host Controller IP component.

7.16 Simulation support features

There are no simulation support features in the Host Controller IP.

8. How to Set Up the IP

8.1 Start-Up Sequence

Before SD/eMMC Initialization sequence some registers in controller and PHY should be set as described in section 9.1.

8.2 Encryption/Decryption Support

SD Memory Card may support Content Protection mechanism, which detailed description is present in Part3 Security Specification Version 3.00 or later.

9. How to Program the IP

The SD Host Controller Standard describes software sequences in Chapter 3. This section adds an information regarding the start-up sequence and error conditions and how to react on them.

9.1 Pre-Initialization Sequence

The sequence presented on Figure 26 describe the pre-initialization sequence, which should be used before SD/eMMC Initialization sequence, and before each change of speed mode. The values that need to be write to specific fields are calculated by script calc_settings.py. How to use script is described in subsection 9.1.1. To program phy registers follow instructions in section 7.2.11. For PHY register description please refer PHY implementation specification.

- To Switch On DLL Reset write 0 to field PHY_SW_RESET in HRS09 register 42.
- Program following fields in PHY_DQS_TIMING_REG: use_ext_lpbk_dqs, use_lpbk_dqs, use_phony_dqs, use_phony_dqs_cmd according to script settings.
- Program following fields in PHY_GATE_LPBK_CTRL_REG: sync_method, sw_half_cycle_shift, rd_del_sel, gate_cfg_always_on.
- Program following fields in PHY_DLL_MASTER_CTRL_REG: param_dll_bypass_mode, param_phase_detect_sel, param_dll_start_point.
- Program following fields in PHY_DLL_SLAVE_CTRL_REG: read_dqs_cmd_delay, clk_wr_delay, read_dqs_delay.
- Program following fields in PHY_CTRL_REG: phony_dqs_timing.
- To Switch Off DLL Reset write 1 to field PHY_SW_RESET in HRS09 register 42.
- Wait for phy_init_complete- read field PHY_INIT_COMPLETE in HRS09 register until it is 1.42
- Program following fields in PHY_DQ_TIMING_REG: io_mask_always_on, io_mask_start, io_mask_end, dma_select_oe_end.
- Program following fields in HRS09 register 42: RDDATA_EN, RDCMD_EN, EXTENDED_WR_MODE, EXTENDED_RD_MODE.
- Program following fields in HRS10 register 43: HCSDCLKADJ.
- Program following fields in HRS16 register 48: WRDATA1_SDCLK_DLY, WRDATA0_SDCLK_DLY, WRCMD1_SDCLK_DLY, WRCMD0_SDCLK_DLY, WRDATA1_DLY, WRDATA0_DLY, WRCMD1_DLY, WRCMD0_DLY.
- Program following fields in HRS16 register 40: RW_COMPENSATE, IDELAY_VAL.

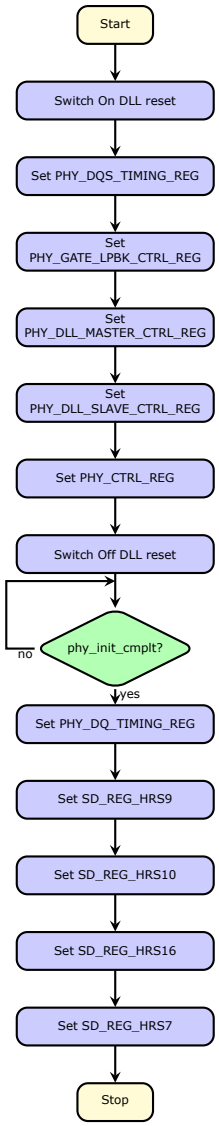

Figure 26. Pre-Initialization procedure

9.1.1 How to run calc_setting script.

To run calc_settings script change directory to software/calc_settings/. Then execute the script as follows: *./calc_settings.py -i <input_file> -o <output_file>*

Where

- <input_file> - input file being comply with the input file format described below
- <output_file> - output file being comply with the output file format described below

9.1.1.1 Input file format

This is a text file which generic structure looks as follow:

mode <speed_mode>
sdmclk <period>
sdclk <period>
iocell_input_delay <time>
iocell_output_delay <time>

delay_element <time>

List of parameters:

<speed_mode> - string value, following are listed the acceptable values
sd_ds_id - identification
sd_ds - default speed
sd_hs - high speed
sd_uhs_sdr12 - ultra high speed SDR12
sd_uhs_sdr25 - ultra high speed SDR25
sd_uhs_sdr50 - ultra high speed SDR50
sd_uhs_sdr104 - ultra high speed SDR104
sd_uhs_ddr50 - ultra high speed DDR50
emmc_sdr_bc - SDR backward compatible
emmc_sdr - SDR
emmc_ddr - DDR
emmc_hs200 - high speed 200MHz in SDR
emmc_hs400 - high speed 200MHz in DDR
emmc_hs400es - high speed 200MHz in DDR with Enhanced Strobe
<period> - integer value in picosecond
<time> - integer value in picosecond

Notes:

- input_delay and output_delay have been verified in range 0ps (0) to 20ns (20000) with 1.25ns step.
- delay_element has been fixed to 24ps.

The example of what input file contain is presented bellow:

sdmclk 5000
sdclk 10000
iocell_input_delay 2500
iocell_output_delay 2500
delay_element [ps]: 24

9.1.1.2 Output file format

The is a text file which consist a list of parameter and its value:

<parameter1> <value1>
<parameter2> <value2>
<parameter3> <value3>
(...)
<parameterN-1> <valueN-1>
<parameterN> <valueN>

Number of parameters in the file (N) is not specified and may vary.

Following parameters represents a register fields in:

- SD/eMMC Host Controller:
  - sdhc_extended_rd_mode
  - sdhc_extended_wr_mode
  - sdhc_hcsdclkadj
  - sdhc_idelay_val
  - sdhc_rdcmd_en
  - sdhc_rddata_en
  - sdhc_rw_compensate
  - sdhc_sdcfsh
  - sdhc_sdcfsl
  - sdhc_wrcmd0_dly

  - sdhc_wrcmd0_sdclk_dly
  - sdhc_wrcmd1_dly
  - sdhc_wrcmd1_sdclk_dly
  - sdhc_wrdata0_dly
  - sdhc_wrdata0_sdclk_dly
  - sdhc_wrdata1_dly
  - sdhc_wrdata1_sdclk_dly
- Combo PHY
  - cp_clk_wr_delay
  - cp_data_select_oe_end
  - cp_dll_bypass_mode
  - cp_dll_locked_mode
  - cp_dll_start_point
  - cp_gate_cfg_always_on
  - cp_io_mask_always_on
  - cp_io_mask_end
  - cp_io_mask_start
  - cp_rd_del_sel
  - cp_read_dqs_cmd_delay
  - cp_read_dqs_delay
  - cp_sw_half_cycle_shift
  - cp_sync_method
  - cp_use_ext_lpbk_dqs
  - cp_use_lpbk_dqs
  - cp_use_phony_dqs
  - cp_use_phony_dqs_cmd

Value is an integer (decimal) number to be written to the register field.

9.2 Error handling

9.2.1 Generic Operation Error Recovery

Figure 27: Generic Operation Error Recovery procedure shows an example software procedure to handle errors detected by the SD/eMMC Host Controller. Once an error is detected and reported in any interrupt status register, software identifies whether the error is recoverable or non-recoverable. If the error is recoverable, software follows the procedure requesting Software Reset for CMD and/or DAT logic. Main purpose of this procedure is to clear logic affected by an error and prepare to a next transaction (e.g. to redo interrupted read/write operation).

This procedure probably won’t help in case of a non-recoverable error. The non-recoverable error usually requires a full software/hardware reset and a device power cycle.

The same procedure can be used with SD, SDIO and eMMC device.

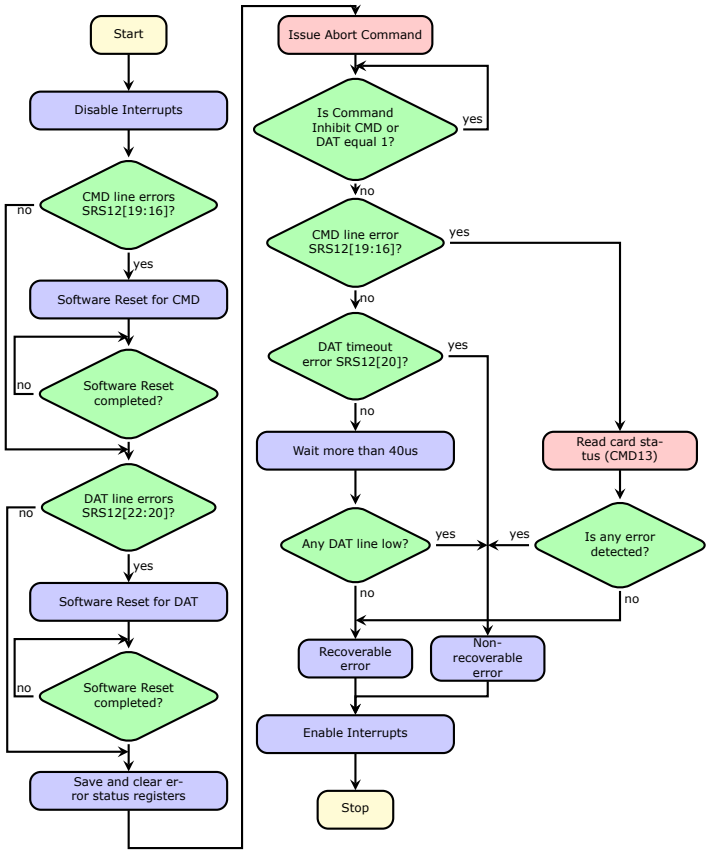

Figure 27. Generic Operation Error Recovery procedure

9.2.2 Command Queuing Error Recovery

Figure 28:Command Queuing Error Recovery procedure shows an example Command Queuing error recovery procedure required to get the host controller and a device from error state and prepare for following operations.

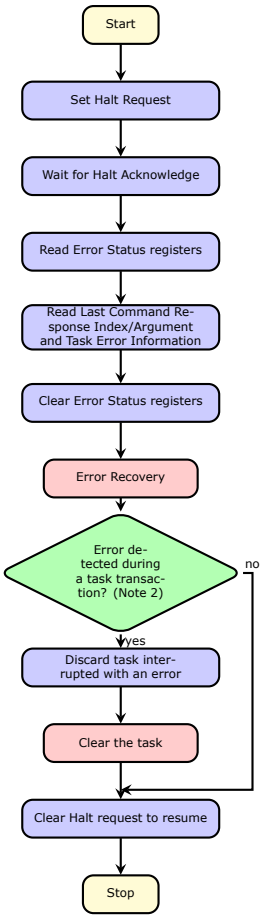

Figure 28. Command Queuing Error Recovery procedure

Note: The SD Host Controller always replies to the halt request by setting halt acknowledge. Software needs to wait for the acknowledge before running Generic Operation Error Recovery. The acknowledge is signalized after unpredicted amount of time — the time required to finish any pending operation. The acknowledge is set when command and data path reports its statuses (whether those are successfully completion or interrupted by an error).

Note 2: The discard / task clear is required for the task interrupted by an error.

The Command Queuing reuses protection mechanisms built into the SD Host Controller. It also adds an automatic error detection for the R1 device response.

Following is a full list of errors that can be detected by the Command Queuing:

- R1 response error
- Command errors (Index/CRC/End Bit/Timeout)
- Data transfer errors (CRC/End Bit/Timeout)
- DMA error
- Master Interface Error

Once any of those error is detected, an internal logic can stop, reports type of error and waits for Command Queuing Error Recovery procedure.

Command Queuing is capable of sending CMD13 SQS (Send Queue Status) to a device during the data transaction. If this feature is enabled (i.e. CQRS16.CQSSCBC > 0) and a data transaction is in-progress, the controller sends CMD13 SQS concurrently to the data transaction and reports possible CMD13 SQS error in the auto command error status registers (SRS15[4:1]).

SD/eMMC Host Controller may report Current Limit Error but this error is a command queuing task agnostic and won't interrupt any transaction. This error is application specific and thus should be classified as non-recoverable error. Software reaction must be adequate and follow the application assumptions.

SD/eMMC Host Controller does not use mechanism of Response Error Check when Command Queuing is enabled.

Software may either poll the status registers or rely on the interrupts generated by the Host Controller. If error occurrence must raise a system interrupt, software enables the required interrupt status registers before task submitting. The interrupt generation is configured by the following registers:

- Error Status Enable (SRS13[31:16])
- Error Signal Enable (SRS14[31:16])
- AXI Error Response (HRS03[19:16], HRS03[11:8])
- Response Error Detected Status Enable (CQRS05[2])
- Response Error Detected Signal Enable (CQRS06[2])

Table 28. Error cases

Operation	**Event** (any event)	**Consequence**
Read Task Descriptor before CMD44 (Task Queuing)	Master Error	Halt and report Task ID in Response Mode Error
Read Task Descriptor before CMD46 / CMD47 (Task Execution)	Master Error	Halt and report Task ID in Data Transfer Error
Read Task Descriptor before DCMD (Task Execution)	Master Error	Halt and report Task ID in Response Mode Error
CMD44	Command Error	Halt and report Task ID in Response Mode Error
CMD45	Command Error	Halt and report Task ID in Response Mode Error
CMD13 while controller in idle	Command Error	Halt and Task ID is not reported
CMD13 during CQ transaction	Command Error	Halt and Task ID is not reported
CMD46 / CMD47	Command Error Data Error DMA Error	Halt and report ID in Data Transfer Error
DCMD	Command Error	Halt and report ID in Response Mode Error
DCMD	Data Timeout Error	Halt and report ID in Response Mode Error

Data Transfer Error in Task Error Information is:

- set to 1 when error appears during CMD46 or CMD47
- clear to 0 when error appears during CMD44, CMD45 or DCMD

Response Mode Error in Task Error Information is:

- set to 1 when error appears during CMD44, CMD45 or DCMD
- clear to 0 when error appears during CMD46 or CMD47

Task Error Information does not changed when CQ is disabled or halted.

10. Clocks and Resets

10.1 Clock

The SD/eMMC Host Controller has several clock domains which are shown in Figure 29 and 30 and are described in Table 29. Due to the Slave Interface configurability, the clock connection looks different depends on whether the AXI Slave Interface (Figure 29) or the APB Slave Interface (30) is connected.

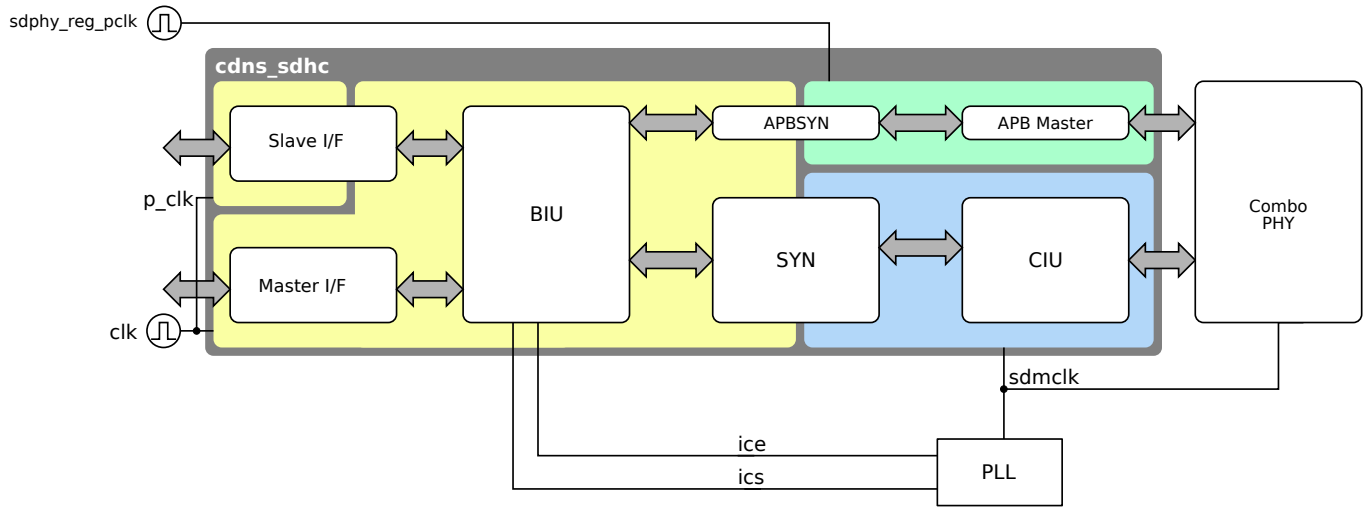

Figure 29. Clock Diagram (AXI Interface)

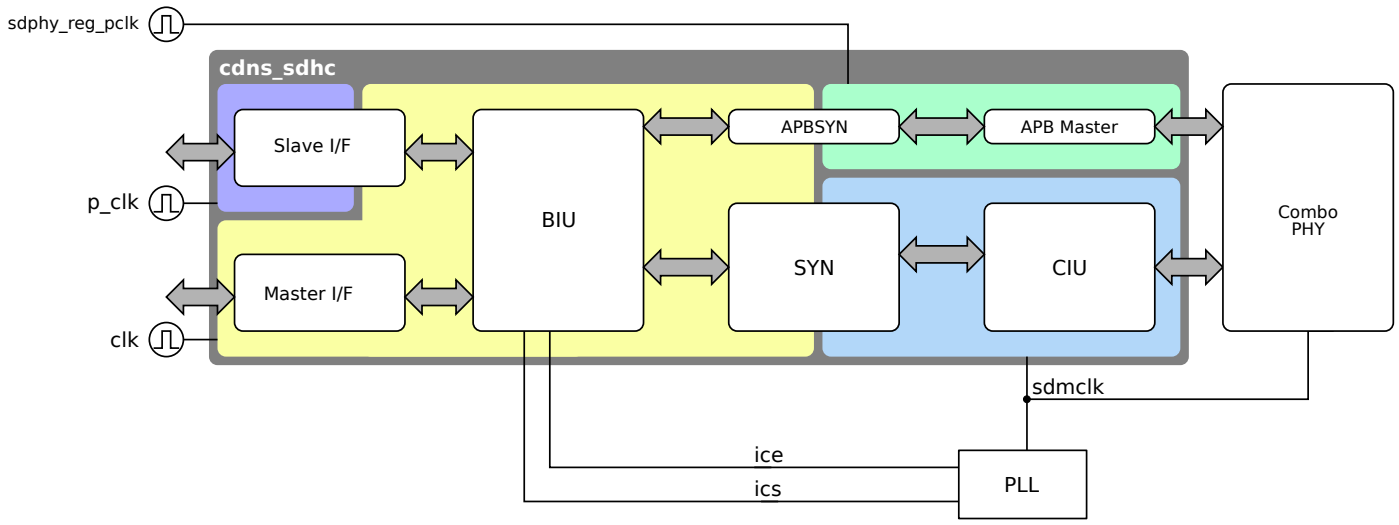

Figure 30. Clock Diagram (APB Interface)

Table 29. Clock Inputs

Clock	Min freq.	Max freq.	Description
s_pclk	20MHz	400MHz*	When the system APB master is connected to the SD/eMMC Host Controller APB slave interface, this input is supplied from the APB clock. The minimum and maximum frequencies are not specified by the SD standard, and those are defined as a verification assumption. The APB clock frequency must be defined within range from 20MHz to 400MHz. If the system AXI master is connected to the AXI slave interface and the APB slave interface is remained unconnected, this input is supplied from the same source clock as the clk.
clk	20MHz	400MHz*	This clock input must be supplied from the AXI slave interface clock or the AHB slave interface clock (user makes a decision which one is to be used). In case the system AXI master is connected to the AXI slave interface, the AXI master interface and the AXI/AHB iternface works on the common clock. The minimum and maximum frequencies are not specified by the SD standard, and those are defined as a verification assumption. The AXI/AHB clock frequency must be defined within range from 20MHz to 400MHz.
sdmclk	50MHz	200MHz	This is the SD Master clock. It supplies the Card Interface Unit/Logic in the SD and eMMC mode.
sdphy_reg_pclk	20MHz	400MHz*	Clock utilized by the APB interface that connects Host Controller APB Master and Combo PHY APB Slave. This clock and *clk* are supplied from the same source clock.

* — Reference maximum frequency is based on TSMC 28nm technology and the actual achievable maximum frequency may vary per the specified process/library and margin requirements. The maximum achievable system clock frequency may be reduced further at geometries above TSMC 55nm.

10.2 Reset

10.2.1 Asynchronous Hardware Reset

The SD/eMMC Host Controller is asynchronously reset core. The asynchronous resets are active low. The core has separate hardware reset inputs for each clock domain. The resets can be asserted at any time and are to be de-asserted on the rising edge of related clock (table 30). In order to reset the host controller, all resets are to be activated and all active resets period have to overlap before the resets are de-asserted. Minimum time of overlapping is not defined and depend on the target technology. The time is to be sufficient to reset controller logic.

Table 30. Hardware resets inputs

Reset name	Related clock domain
s_presetn	s_pclk
rstclk_n	clk
rstsdmclk_n	sdmclk
sdphy_reg_presetn	sdphy_reg_pclk

10.2.2 Software Reset

The SD/eMMC Host Controller can be reset by software in several ways depends on the selected mode. Software reset can be started at any time by setting one of the following bits:

- HRS00.SWR (Software Reset) resets all internal logic. This reset is common for the SD Host Controller and operates similarly to the hardware reset. Refer to the HRS00.SWR register description for details.
- SRS11.SRFA (Software Reset For All) resets all internal logic for a given slot. Please refer to the SRS11.SRFA register description for details.
- SRS11.SRCMD (Software Reset For CMD) resets the part of the design responsible for command generation and response checking for a given slot. Please refer to the SRS11.SRCMD register description for details.
- SRS11.SRDAT (Software Reset For DAT) resets the part of the design responsible for data transfers for a given slot. Please refer to the SRS11.SRDAT register description for details.

11. Verification

11.1 Overview of the environment

The Environment was designed to allow simulation of the C written programs with the HDL simulator. The Environment is split into a two separated modules:

- The SV module
- The C Program module

The SV module is written in System Verilog and it is the main part of the Environment. The SV module is split into two parts:

- Synthesizable code of the DUT and all required project specific elements.
- Set of transactors to interact with the host controller and the C program module.

The C Program module is a test code written in the C language. The test code contains user procedures controlling the DUT and a set of library functions to interact with the SV Environment during simulation. Both SV simulation and the C Program run on the same host computer.

The SV module and C Program module interact with each other through the SV DPI-C interface.

11.2 Environment description

11.2.1 SV Module

The Figure 31 shows the block diagram of the SV Module.

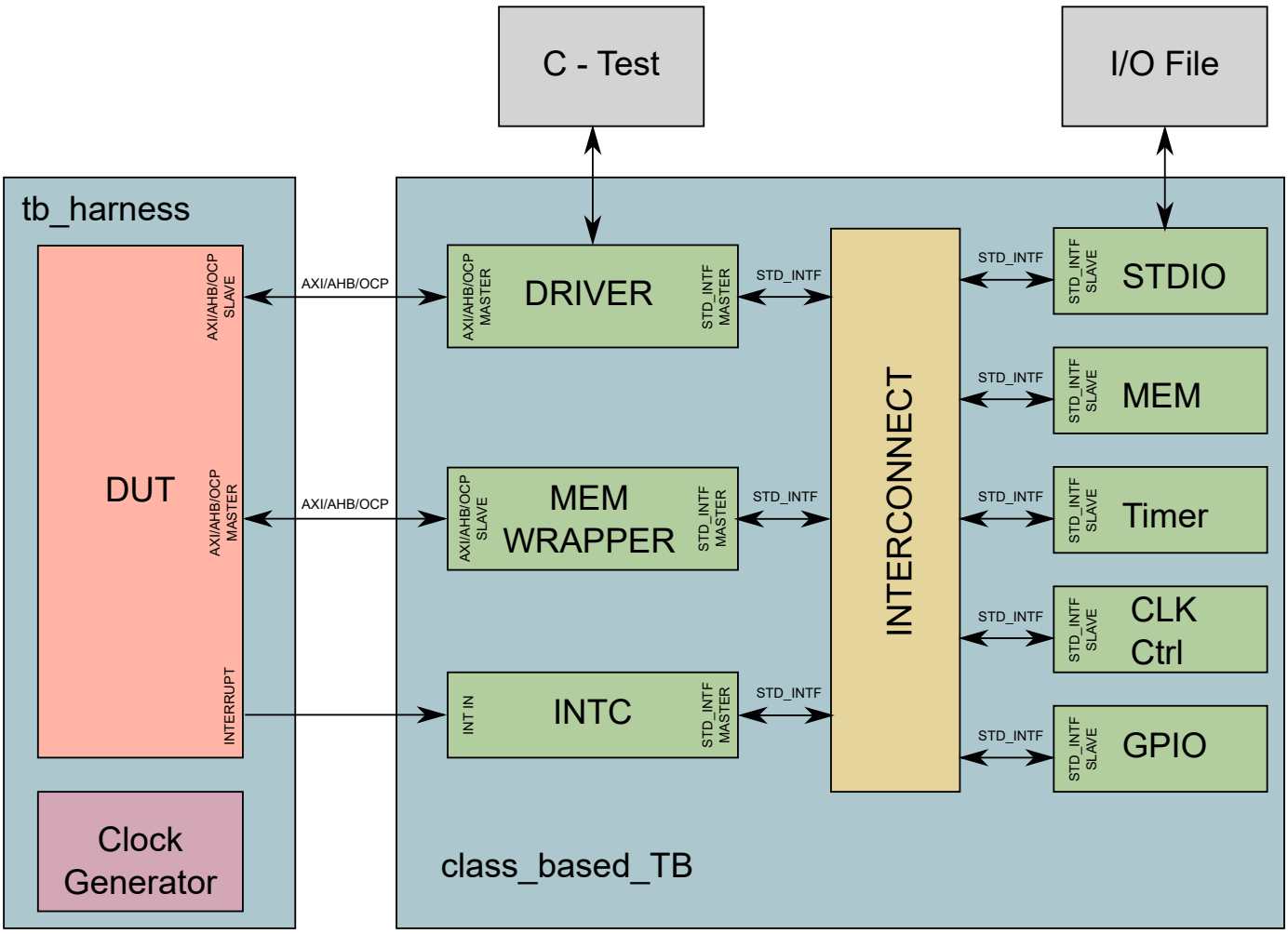

Figure 31. Block Diagram of SV Module

The *tb_harness* sub module integrates DUT, Clock Generator, and the test bench interfaces. The AHB/AXI/OCP slave interface is used to configure the DUT. The master interface is used for DMA transfers between the DUT and the system memory model located in test bench.

The *class_based_TB* sub module is the main part of the SV Module. It contains a few elements which are responsible for emulating example embedded system, namely:

- Driver
- Standard bus interconnect (INTERCONNECT)
- Standard I/O component (STDIO)
- Memory model (MEM)
- Interrupt controller (INTC)
- System clock controller (CLK CTRL)
- Timer (TIMER)
- General purpose IO (GPIO)

The Driver takes care of communication between the SV module and the C Program module. The Driver uses DPI-C interfaces for this purpose. When the C program calls I/O access function (IOWR_32 or IORD_32) the Driver module will recognize which part of environment is addressed and it will make an access over the external AHB/APB/AXI/OCP interface (depending on the configuration) or over the internal interface (STD_INTF - internal virtual interface, dedicated for components used in the test bench). All transfers on the STD_INTF are maintained by the INTERCONNECT module, which recognizes destination address and sends data to the corresponding module.

The MEM element works as the system memory model. It is necessary to run any DMA transactions. To ensure the interface capability between external (AHB/AXI/OCP) interface and internal (STD_INTF) interface the "MEM WRAPPER" module is used. The system memory model used in test environment is equipped with simple memory protect mechanism. When the mechanism is enabled, all transactions addressed to the memory are compared with memory areas defined by dedicated C functions. If tested component's DMA engine is trying to access unallocated region the transfer is ignored and corresponding message is sent to the log file.

The system memory model can work in two different modes: INTERNAL and EXTERNAL. In the INTERNAL mode, the memory is simulated in the SV environment. Therefore, the memory contents can be viewed by the user in the simulator, for C program the memory is seen as an element of I/O space (program can access it via IORD_32 and IOWR_32 function only; access via memory pointer is not possible). In the EXTERNAL mode, the memory is simulated inside the C program environment. The memory contents can be accessed through the memory pointer and by the I/O functions. In this mode the memory contents cannot be viewed by the user in HDL simulator. All accesses must be made from the C program. All delivered example programs require the use of the EXTERNAL memory mode.

The *class_based_TB* module contains model of interrupt controller (INTC) which allows checking if DUT's interrupt mechanism works correctly. It can handle up to 32 interrupt sources. Inputs selection and configuration of the interrupt controller can be done from C program level with dedicated procedures. Interrupt is generated when rising signal edge is detected on enabled interrupt input.

The STDIO component is used to log C program messages in selected file (by default: console_log.txt file). Additionally, it provides the "inbyte()" C function which emulates reading one char from standard input. Simulation environment uses data from preselected file (by default: infile.txt file) as a standard input.

To grant possibility of changing system clock frequency directly from the C program the CLK CTRL component was introduced. It is connected to the STD_INTF and controls clock generator on tb_harness level of the test environment.

The TIMER component is used to measure execution time of various operations in a hardware. It is especially dedicated to create performance tests. This component requires value of the system clock period (in ns) - delivered by internal *clk_period_ns* signal.

The GPIO component is used to drive up to 32 general purpose ports. This feature allows driving DUT control pins that are not part of the standard system bus including a reset and bootstrap pins.

11.2.2 Environment configuration

All devices connected to the STD_INTF interface can be accessed from the C program with IOWR_32 and IORD_32 functions. The table 31 shows address location of each slave component:

Table 31. Slave Components Addresses

Name	Base Address	High Address
MEM	0x00000000	0x4fffffff
INTC	0x60000000	0x6000000f
CLK CTRL	0x70000000	0x7000000f
TIMER	0x72000000	0x7200000f
STDIO	0x74000000	0x7400000f
GPIO	0x75000000	0x750000ff

The test environment can be configured with file *sv_c_env_conf_params.svh*. It contains following parameters:

- AXI_SFR_INTF_NUM - number of external AXI master interfaces connected with core control slave interface. The minimum value of this parameter must be one. If system doesn't contain any AXI master interfaces this parameter should be commented.
- AHB_SFR_INTF_NUM - number of external AHB master interfaces connected with core control slave interface. The minimum value of this parameter must be one. If system does not haven any AHB master interfaces this parameter should be commented.
- APB_SFR_INTF_NUM - number of external APB master interfaces connected with core control slave interface. The minimum value of this parameter must be one. If system does not contain any APB master interfaces this parameter

should be commented.
- OCP_SFR_INTF_NUM - number of external OCP master interfaces connected with core control slave interface. The minimum value of this parameter must be one. If system does not contain any OCP master interfaces this parameter should be commented.
- AXI_DMA_INTF - if this parameter is defined, the high-speed AXI master interface is available in the system
- AHB_DMA_INTF - if this parameter is defined, the high-speed AHB master interface is available in the system
- OCP_DMA_INTF - if this parameter is defined, the high-speed OCP master interface is available in the system
- MEM_MASTER_A_WIDTH - address bus width of external master interfaces
- MEM_MASTER_D_WIDTH - data bus width of external master interfaces
- MEM_MASTER_ID_WIDTH - ID width size for external master interfaces
- PARITY_ENABLE - enable parity support for system interfaces
- MEM_PROTECT_EN - enable memory protect mechanism
- MEM_PROTECT_MAX_BUFF_CNT - maximum number of memory regions, if memory protect mechanism is enabled
- WD_TIMEOUT - value of internal timeout. When simulation time exceed the timeout value it will be interrupted with fatal error.
- PARITY_TYPE - parameter allow to select used parity option. Allowed values are:
  - 0 - the even parity,
  - 1 - the odd parity.
- DRIVER_DBG_MSG, TIMER_DBG_MSG, CLKCTRL_DBG_MSG, AHB_WRAP_DBG_MSG, AXI_WRAP_DBG_MSG, OCP_WRAP_DBG_MSG, INTC_DBG_MSG, MEM_DBG_MSG, STDIO_DBG_MSG, GPIO_DBG_MSG, - if defined, debug messages from corresponding components are enabled

11.2.3 C module part

11.2.3.1 Environment global functions (file: sw_hw_wrapper.c)

The Environment uses *sw_hw_wrapper.c* file compiled together with the test C program. The test C program contains all the functions interacting with either SV Module or real hardware. Selection is made by SIM define. The table 32 list functions.

Table 32. Function Mapping

Name	Implementation for Hardware (SIM=0)	Implementation for SV environment (SIM=1)
IOWR_32	Writing data through memory pointer	Execution of SV task sv_axi_write() - write to the I/O space
IORD_32	Reading data through memory pointer	Execution of SV task sv_axi_read()- read from the I/O space
sd_delay	Waiting specific time with for loop construction	Execution of SV task sv_wait() -waiting for selected clock AXI/AHB/OCP cycles
sd_printf	Simple printf function	Printf with additional logging to file console_log.txt
inbyte	Simple inbyte() function	Returns one byte from file infile.txt
sv_malloc	The malloc function	Malloc function with additional passing information memory region information to memory protect mechanism
sv_free	The free function	Free functions with additional removing information about memory region from memory protect mechanism
set_mem	N/A	Function executed by SV environment for writing data to memory located in the C program space
read_mem	N/A	Function executed by SV environment for reading data from memory located in the C program space

The IOWR_32 and IORD_32 functions are used for accessing the I/O space of SV Module.

All instructions written in C language are executed in zero simulation time; consequently, to advance simulation time, Verilog tasks must be executed. Execution time of a task depends on its internal construction - for example sv_delay(1)

will be executed in one cycle of AHB/AXI/OCP clock.

The sd_printf and inbyte() functions can be used to emulate access to standard i/o device by the C program.

In order to use built in memory protect mechanism the software must use sv_malloc and sv_free functions instead of standard malloc and free functions. They add/remove information about memory region to/from the MEM component inside the test environment.

The last two functions (set_mem and read_mem) are required only if the system memory model works in EXTERNAL mode. They access memory array located in the C Program, allowing the SV module to write or read data from the memory.

11.2.3.2 Interrupt controller functions (file: xintc_l.c)

A set of functions is provided to handle interrupt controller. These functions are grouped in xintc_l.c file:
void XIntc_mMasterEnable(uint32_t BaseAddress)

- BaseAddress : Base address of the Interrupt Controller
- Description : Global enable of the interrupt controller

void XIntc_mMasterDisable(uint32_t BaseAddress)

- BaseAddress : Base address of the Interrupt Controller
- Description : Global disable of the interrupt controller

void XIntc_mEnableIntr(uint32_t BaseAddress, uint32_t EnableMask)

- BaseAddress : Base address of the Interrupt Controller
- EnableMask : Value of EnableMask
- Description : Enable interrupt inputs with EnableMask:
  - 1 - enable interrupt,
  - 0 - no change.

void XIntc_mDisableIntr(uint32_t BaseAddress, uint32_t DisableMask)

- BaseAddress : Base address of the Interrupt Controller
- DisableMask : Value of DisableMask
- Description : Disable interrupt inputs with DisableMask:
  - 1 - disable interrupt,
  - 0 - not change.

void XIntc_RegisterHandler (uint32_t BaseAddress, int InterruptId, void* Handler, void *CallBackRef)

- BaseAddress : Base address of the Interrupt Controller
- InterruptId : Interrupt source identification number
- Handler : Pointer to interrupt handling function
- CallBackRef : Arguments to interrupt handling functions
- Description : Connect interrupt source (InterruptId) to a selected function. Handler argument is used to pass pointer to the selected function while CallBackRef is used to pass target function argument.

int XIntc_DeviceInterruptHandler(void* DeviceID)

- DeviceID : Device number
- Description : This function will be executed by SV Module if interrupt from any of enabled inputs occur. This function should identify interrupt source and execute function connected with this source by the XIntc_Register_Handler function.

11.2.3.3 Timer dedicated functions (file: timer.c)

The following C functions were provided to control the TIMER component:
void timer_start (unsigned long timer_addr, unsigned char dev_id)

- timer_addr : Base address of the Timer component
- dev_id : Device identification number
- Description : Start timer.

void timer_stop (unsigned long timer_addr, unsigned char dev_id)

- timer_addr : Base address of the Timer component
- dev_id : Device identification number
- Description : Stop timer

unsigned long timer_getcnt (unsigned long timer_addr, unsigned char dev_id)

- timer_addr : Base address of the Timer component
- dev_id : Device identification number
- Description : Get time value (in system clock cycles or ns). The system clock is taken from clk_generator component (clock 0).

11.2.3.4 Clock generator dedicated functions (file: clk_generator.c)

Those functions are used to control internal clock controller/generator of the verification environment:
void clkgen_change (uint32_t freq_kHz, uint8_t clk_sel)

- freq_kHz : Target frequency (in kHz)
- clk_sel : Clock select
- Description : Change frequency of selected with clk_sel clock to freq_kHz kilohertz. Clock from source 0 is connected to environment's Timer as system clock.

void clkgen_enable (uint8_t enable_vect)

- enable_vect : Clock enable signal (each bit corresponds to single clock output)
- Description : Enable selected clocks. Clocks corresponding to location of logic 1 in enable_vect will be enabled.

void clkgen_disable (uint8_t disable_vect)

- disable_vect : Clock disable signal (each bit corresponds to single clock output)
- Description : Disable selected clocks. Clocks corresponding to location of logic 1 in disable_vect will be disabled.

11.2.3.5 General purpose IO controller functions (file: gpio.c)

The following C functions were provided to control the GPIO component:
void gpio_write (uint32_t data)

- data : Value which will be written
- Description : Write 32-bit vector to GPIO ports configured as outputs.

uint32_t gpio_read (void)

- Description : Read state of GPIO port as a 32-bit width vector

uint32_t gpio_set_tri (uint32_t tri_mask)

- tri_mask : Mask to select direction of each GPIO bit
- Description : Change direction of selected bit. All ports which have corresponding bit in tri_mask set to 1 will be set as input port (default option).

11.2.3.6 DMA controller dedicated functions (file dma_ctrl.c)

A set of functions is provided to handle DirectMemoryAccess controller:
void dma_write_data (char* buff, uint64_t dest_addr, uint32_t size)

- buff : Memory buffer contains data which should be written
- dest_addr : Address of the destination slave DMA port.
- size : Number of bytes to transfer. Must be aligned to word size.
- Description: Copy data from system memory buffer (buff) to DUTs destination address (dest_addr). Number of bytes to copy is specified with the size parameter.

void dma_read_data (char* buff, uint64_t src_addr, uint32_t size)

- buff : Memory buffer utilized to store read data
- src_addr : Address of the source slave DMA port.
- size : Number of bytes to transfer. Must be aligned to word size.

- Description : Copy data from DUTs source address (src_addr) to system memory buffer (buff). Number of bytes to copy is specified with the size parameter.

void dma_wait (void)

- Description : Wait for finishing all transactions by the DMA engine.

11.3 Environment usage

11.3.1 Running tests with IRUN

The simulation environment provided in the delivery package includes several test to demonstrate some key features of the Host Controller. In order to run test(s), user utilizes Cadence Xcelium Simulator version 21.08.001. Change your working directory to the run script directory

```
cd {path_to_delivery_package}/cdns_sdhc/func_ver/sanity/c_env/scripts
```

and execute the shell script

```
./simulate.sh [parameters]
```

This script supports following parameters:

- `-test_num <TEST_NUM>` — select desired test number; if not specified, all tests are run one-by-one in single simulation
- `-ctype <CARD_TYPE>` — select desired card type
  - `sd_card` — SD memory model
  - `sdio_card` — SDIO model
  - `emmc_card` — eMMC model

  If not specified, SD memory card is set by default.
- `-phy_type <PHY_TYPE>` — select desired phy model (if available)
  - `combo_dll_phy` — ASIC phy model
  - `ultrascaleplus_combophy` — Xilinx Ultrascale+ FPGA model (optional)

  If not specified, ASIC phy model is the default setting.
- `-cvendor <CARD_VENDOR>` — select desired card vendor
  - `eva` — Internal model
  - `denali` — Denali model (SDIO option not available)

  If not selected, Internal model is set by default. Denali model is Cadence Verification IP which requires a separate license. Make sure your model is setup properly. If your have no license VIP, contact us.
- `-gui` — enable simulator in GUI mode
- `-m32` — run simulator in 32-bit mode; not recommended option; should be used for 32-bit machines only
- `-linedebug` — enable breakpoints inside the RTL code; feature increases simulation time
- `-nosimulate` — break after compilation and elaboration stage
- `-clean` — clean files created by the script
- `-help` — display list of all parameters with a short description; obtain full list of test;

For example, executing the below command user runs simulation of the test 0 on the default SD memory model in GUI mode.

```
./simulate.sh -gui -test_num 0
```

12. Registers

12.1 HRS00 (0x0000)

HRS00 - General Information Register

Table 33. HRS00 Register Fields

Bit	Type	Description	Reset
23-16	R	**SAV - Slot Available** Field informs that the Host Controller supports one slot.	8'h1
0	W1S	**SWR - Software Reset** When set to 1, the entire core is reset. After reset operation complete, SWR bit is automatically cleared. It takes some time to complete the requested reset operation, so the software should always poll SWR bit status, and continue the other operations only when SWR is cleared to 0. There is no difference between SWR and SRS11.SRFA software resets. Both resets the same flip-flops.	1'h0

12.2 HRS01 (0x0004)

HRS01 - Debounce Setting Register

Table 34. HRS01 Register Fields

Bit	Type	Description	Reset
23-0	RW	**DP - Debounce Period** Defines the number of system (clk) clock cycles used by the debounce logic, which detects card insertion and removal events. The debounce period is equal to DP * tclk, where tclk is the period of clk clock. If there is no change on pad_mem_ctrl_0 signal level for a programmed debounce period, the core logic decodes the card state as stable and triggers card_inserted or card_removed event. Typically, DP value should be chosen to obtain the period of 20ms. This register is reset to DEBOUNCE_PERIOD.	24'h32

12.3 HRS02 (0x0008)

HRS02 - Bus Setting Register

Table 35. HRS02 Register Fields

Bit	Type	Description	Reset
17-16	RW	**OTN - Number of Outstanding Transfers** Specifies number of outstanding transfers on DMA (Master) interface. The number of outstadings is (OTN + 1), where OTN can be defined in range 0 to 3. This register is set to 3 after reset (i.e. 4 outstanding transfers).	2'h3
3-0	RW	**PBL - Programmable Burst Length** This field defines a maximum number of beats in DMA burst. The value can be changed when no active transfer. This register is 0 after reset. • 0001b - 1 beat in burst • 0010b - 2 beats in burst • 0011b - 4 beats in burst • 0100b - 8 bits in burst • other - 16 beats in burst	4'h0

12.4 HRS03 (0x000c)

HRS03 - AXI ERROR Responses Register

These registers extend the standard set of SD-HOST interrupt statuses by information about AXI interface exceptions.

The registers are divided into three groups:

- Signal Enable registers allow to enable/mask signaling the Interrupt Status registers (HRS03[3:0]) on interrupt port

- Status Enable registers allow to enable/disable interrupt source for each Interrupt Status separately

- Interrupt Status are triggered whenever the interrupt source is detected and the Status Enable register is enabled

Table 36. HRS03 Register Fields

Bit	Type	Description	Reset
19	RW	**AER_IEBS - Signal Enable for AXI ERROR Response B channel: SLVERR** 1 - interrupt enable 0 - interrupt masked	1'h0
18	RW	**AER_IEBD - Signal Enable for AXI ERROR Response B channel: DECERR** 1 - interrupt enable 0 - interrupt masked	1'h0
17	RW	**AER_IERS - Signal Enable for AXI ERROR Response R channel: SLVERR** 1 - interrupt enable 0 - interrupt masked	1'h0
16	RW	**AER_IERD - Signal Enable for AXI ERROR Response R channel: DECERR** 1 - interrupt enable 0 - interrupt masked	1'h0
11	RW	**AER_SENBS - Status Enable for AXI ERROR Response B channel: SLVERR** 1 - status enable 0 - status disable	1'h0
10	RW	**AER_SENBD - Status Enable for AXI ERROR Response B channel: DECERR** 1 - status enable 0 - status disable	1'h0

Table 36. HRS03 Register Fields

Bit	Type	Description	Reset
9	RW	**AER_SENRS - Status Enable for AXI ERROR Response R channel: SLVERR** 1 - status enable 0 - status disable	1'h0
8	RW	**AER_SENRD - Status Enable for AXI ERROR Response R channel: DECERR** 1 - status enable 0 - status disable	1'h0
3	W1C	**AER_BS - AXI ERROR Response B channel: SLVERR** This bit is set when a SLVERR is detected on AXI Master bus in B channel (Write Response Channel).	1'h0
2	W1C	**AER_BD - AXI ERROR Response B channel: DECERR** This bit is set when a DECERR is detected on AXI Master bus in B channel (Write Response Channel).	1'h0
1	W1C	**AER_RS - AXI ERROR Response R channel: SLVERR** This bit is set when a SLVERR is detected on AXI Master bus in R channel (READ Response Channel).	1'h0
0	W1C	**AER_RD - AXI ERROR Response R channel: DECERR** This bit is set when a DECERR is detected on AXI Master bus in R channel (READ Response Channel).	1'h0

12.5 HRS04 (0x0010)

Table 37. HRS04 Register Fields

Bit	Type	Description	Reset
15-0	RW	**PHYREGADDR - PHY Register Address** This filed defines the PHY Register Address for read / write accesses to PHY Register done through the HRS05.	16'h0

12.6 HRS05 (0x0014)

Table 38. HRS05 Register Fields

Bit	Type	Description	Reset
31-0	RW	**PHYREGDATA - PHY Register Data Port** Access to this register generates read or write transaction to Combo PHY Register. When this field is read, a read transaction is sent to PHY through APB interface. Value received in the transaction is passed back as the access result. When this field is write, a write transaction is sent to PHY through APB interface along with the written data. Host Controller – Combo PHY APB interface does not support unaligned transfers, and so access to this field is limited to 32-bit transactions only. Unaligned (16-bit or 8-bit) access to this field may lead to unexpected result.	32'h0

12.7 HRS06 (0x0018)

HRS06 - eMMC control registers

Table 39. HRS06 Register Fields

Bit	Type	Description	Reset
2-0	RW	**EMM - eMMC Mode select** This field sets eMMC mode. The mode should reflect to the eMMC device setting. If the SD card is in use, this field needs to be 000b. • 000b - SD Card in use • 010b - SDR • 011b - DDR • 100b - HS200 • 101b - HS400 • 110b - HS400 Enhanced Strobe • others - Legacy	3'h0

12.8 HRS07 (0x001c)

HRS07 - IO Delay Information Register

Table 40. HRS07 Register Fields

Bit	Type	Description	Reset
20-16	RW	**RW_COMPENSATE - Read Wait Compensate value** According to delays between PAD and dfi_rddata, dfi_wrdata and PAD and to Read Wait timing requirements, the signal dat[2] should be set to 0 earlier than controller read the end bit of read data. Designer should update this register with delay of data path count in sdmclk clock cycles. If the value is greater than 10 and value of field SDCLK Frequency Select (concatenation of SRS11.SDCFSH, SRS11.SDCFSL) is equal 0, then io_mask_start parameter in PHY register phy_dq_timing_reg should be set with value equal (RW_COMPENSATE-10)*2.	5'h0
4-0	RW	**IDELAY_VAL - Input delay value for IO.** Designer should update this register with delay value of IO with appriopriate input delay. Delay is count in half of period of sdmclk. If sdmclk is working at 200MHz frequency, then 1 is 2,5 ns. This value will be used to compensate delay of DAT line when controller is reading Card Interrupt.	5'h0

12.9 HRS08 (0x0020)

HRS08 - PHY DLL Update Control and Status Register

Table 41. HRS08 Register Fields

Bit	Type	Description	Reset
1	R	**PHY_DLL_UPDACK** This register contains value read from sdphy_dfi_ctrlupd_ack port.	1'h0
0	RW	**PHY_DLL_UPDREQ** This register controls sdphy_dfi_ctrlupd_req port.	1'h0

12.10 HRS09 (0x0024)

HRS09 - PHY Control and Status Register

Table 42. HRS09 Register Fields

Bit	Type	Description	Reset
31-28	RW	**LVSI_CNT** This field defines period of SDCLK pulse during LVS Identification. The period varies in range from LVSI_CNT * 2^(LVSI_TCKSEL+1) * t_CLK to (LVSI_CNT + 1) * 2^(LVSI_TCKSEL+1) * t_CLK. It is recommended to use lower LVSI_TCKSEL value and higher LVSI_CNT value to reduce SDCLK pulse period variation. The variation is +1/-1 LVSI_TCLKSEL unit.	4'hf
27-22	RW	**LVSI_TCKSEL** This field defines unit for LVSI_CNT. • 0 - 2*t_CLK (two CLK clock cycles) • 1 - 4*t_CLK (four CLK clock cycles) • 2 - 8*t_CLK (eight CLK clock cycles) • 3 - 16*t_CLK (sixteen clock cycles) • N - (2^(N+1))*t_CLK where N must equal 0 to 47. Values above 47 are reserved.	6'h7
16	RW	**RDDATA_EN** If 1, dfi_rddata_en is forced to 1, else host logic controls the signal.	1'h0
15	RW	**RDCMD_EN** If 1, dfi_rdcmd_en is forced to 1, else host logic controls the signal.	1'h0
3	RW	**EXTENDED_WR_MODE** Controls sdphy_param_extended_wr_mode port. Non of software resets clear this register.	1'h0
2	RW	**EXTENDED_RD_MODE** Controls sdphy_param_extended_rd_mode port. Non of software resets clear this register.	1'h0
1	R	**PHY_INIT_COMPLETE** This field contains a value read from sdphy_dfi_init_complete port.	1'h0
0	RW	**PHY_SW_RESET** This field controls sdphy_dll_rst_n.	1'h1

12.11 HRS10 (0x0028)

HRS10 - Host Controller SDCLK start point adjustment

Table 43. HRS10 Register Fields

Bit	Type	Description	Reset
22	RW	**RDDATA_SWAP** If 1, dfi_rddata bytes [7:0] and [15:8] are swapped.	1'h0
19-16	RW	**HCSDCLKADJ** This field allows to adjust flow control mechanism which disables SDCLK. With value 0, the clock (dfi_webar/dfi_webar_high) will be disabled right after end bit of the data block. Increasing this value will cause that clock signal is to be disabled earlier with SDCLK period step.	4'h0

12.12 HRS11 (0x002c)

HRS11 - eMMC Control

Table 44. HRS11 Register Fields

Bit	Type	Description	Reset
0	RW	**EMMC_RST** This field drives dfi_rstbar which is used as eMMC reset.	1'h0

12.13 HRS12 (0x0030)

HRS12 - Host Interrupt Status

Table 45. HRS12 Register Fields

Bit	Type	Description	Reset
3	W1C	**PHY DAT Overflow - Status received from Combo PHY informing about DAT FIFO status.**	1'h0
2	W1C	**PHY DAT Underrun - Status received from Combo PHY informing about DAT FIFO status.**	1'h0
1	W1C	**PHY CMD Overflow - Status received from Combo PHY informing about CMD FIFO status.**	1'h0
0	W1C	**PHY CMD Underrun - Status received from Combo PHY informing about CMD FIFO status.**	1'h0

12.14 HRS13 (0x0034)

HRS13 - Host Status Enable

Table 46. HRS13 Register Fields

Bit	Type	Description	Reset
3	RW	**Mask for PHYDATOF status** • 1 - enabled • 0 - masked	1'h0
2	RW	**Mask for PHYDATUR status** • 1 - enabled • 0 - masked	1'h0
1	RW	**Mask for PHYCMDOF status** • 1 - enabled • 0 - masked	1'h0
0	RW	**Mask for PHYCMDUR status** • 1 - enabled • 0 - masked	1'h0

12.15 HRS14 (0x0038)

HRS14 - Host Signal Enable

Table 47. HRS14 Register Fields

Bit	Type	Description	Reset
3	RW	**Interrupt enable for PHYDATOF status** • 1 - enabled • 0 - masked	1'h0
2	RW	**Interrupt enable for PHYDATUR status** • 1 - enabled • 0 - masked	1'h0
1	RW	**Interrupt enable for PHYCMDOF status** • 1 - enabled • 0 - masked	1'h0
0	RW	**Interrupt enable for PHYCMDUR status** • 1 - enabled • 0 - masked	1'h0

12.16 HRS16 (0x0040)

HRS16 - CMD/DAT output delay

Table 48. HRS16 Register Fields

Bit	Type	Description	Reset
31-28	RW	**WRDATA1_SDCLK_DLY** Value in this field defines a delay of the dfi_wrdata[15:8] signal. The delay is equal value * t_SDCLK/2.	4'h0
27-24	RW	**WRDATA0_SDCLK_DLY** Value in this field defines a delay of the dfi_wrdata[7:0] signal. The delay is equal value * t_SDCLK/2.	4'h0
23-20	RW	**WRCMD1_SDCLK_DLY** Value in this field defines a delay of the dfi_wrcmd signal. The delay is equal value * t_SDCLK/2.	4'h0
19-16	RW	**WRCMD0_SDCLK_DLY** Value in this field defines a delay of the dfi_wrcmd signal. The delay is equal value * t_SDCLK/2.	4'h0
15-12	RW	**WRDATA1_DLY** Value in this field defines a delay of the dfi_wrdata[15:8] signal. The delay is equal value * t_SDMCLK.	4'h0
11-8	RW	**WRDATA0_DLY** Value in this field defines a delay of the dfi_wrdata[7:0] signal. The delay is equal value * t_SDMCLK.	4'h0
7-4	RW	**WRCMD1_DLY** Value in this field defines a delay of the dfi_wrcmd[1] signal. The delay is equal value * t_SDMCLK.	4'h0
3-0	RW	**WRCMD0_DLY** Value in this field defines a delay of the dfi_wrcmd[0] signal. The delay is equal value * t_SDMCLK.	4'h0

12.17 HRS29 (0x0074)

HRS29 - SD Magic Number

"Product Number" - identification number aligned to the right (LSB).

Table 49. HRS29 Register Fields

Bit	Type	Description	Reset
31-0	R	**This product number is IP6061**	32'h6061

12.18 HRS30 (0x0078)

HRS30 - Host Capability Register

This register states whether configurable options are available or are not available in the SD/eMMC Host Controller configuration. This register gives such information about features not been covered by the standard capability registers (SRS16-SRS18).

Table 50. HRS30 Register Fields

Bit	Type	Description	Reset
1	R	**High Speed 400 Enhance Strobe supported** This field informs whether HS400 Enhance Strobe mode is supported (1) or is not supported (0).	1'h1
0	R	**Command Queuing supported** This field informs whether Command Queuing is supported (1) or is not supported (0).	1'h1

12.19 HRS31 (0x007c)

HRS31 - Host Controller Version

This register contains the host controller version number.

Table 51. HRS31 Register Fields

Bit	Type	Description	Reset
27-16	R	**Host Controller Version** Release number of the Host Controller	12'h602
7-0	R	**Fix Version Number** Number of the fix related to the Host Controller Version.	8'h2

12.20 HRS32 (0x0080)

HRS32 - FSM Monitor Register

Table 52. HRS32 Register Fields

Bit	Type	Description	Reset
31	RW	**LOAD - FSM monitor update request** Setting this bit to 1 starts internal FSM monitor to load value from selected FSM. After finishing this bit will be automatically cleared by hardware and FSM status can be read.	1'h0
30-16	W	**ADDR - FSM address** This field selects which FSM status will be read. All available status machines are listed in Debug section of User Guide.	15'h0
15-0	R	**DATA - FSM status** This register contains read FSM status. Before reading it user should select FSM address (ADDR), set LOAD bit and wait until hardware clears it.	16'h0

12.21 HRS33 (0x0084)

HRS33 - Tune Status 0 Register

Table 53. HRS33 Register Fields

Bit	Type	Description	Reset
31-0	R	**STAT0 - Tune status 0** After invoking UHS-I tunning procedure each bit of this register represents status of one tuning step. This field correspond to tuning steps 31-0. • 0 - Step failed • 1 - Step passed	32'h0

12.22 HRS34 (0x0088)

HRS34 - Tune Status 1 Register

Table 54. HRS34 Register Fields

Bit	Type	Description	Reset
7-0	R	**STAT1 - Tune status 1** This field is continuation of STAT0 field. Its value represents status of tuning steps 39-32.	8'h0

12.23 HRS36 (0x0090)

HRS36 - Boot Status Register

Table 55. HRS36 Register Fields

Bit	Type	Description	Reset
6	R	**Boot Error - Descriptor Mechanism Error**	1'h0
5	R	**Boot Error - End Bit Error**	1'h0
4	R	**Boot Error - Data CRC Error**	1'h0
3	R	**Boot Error - Data Timeout Error**	1'h0
2	R	**Boot Error - Invalid Acknowledge Error**	1'h0
1	R	**Boot Error - Acknowledge Timeout Error**	1'h0

Table 55. HRS36 Register Fields

Bit	Type	Description	Reset
0	R	**Boot Active** Informs that the BOOT is active and the operation cannot be interfere by writing any registers.	1'h0

12.24 HRS40 (0x00a0)

HRS40 - BASE Address 0 for Auto-configuration descriptor mechanism

Table 56. HRS40 Register Fields

Bit	Type	Description	Reset
31-0	RW	**BASE_ADDR0 - lower descriptor list base address** Lower 32-bits of the base address required for auto-configuration descriptor mechanism.	32'h0

12.25 HRS41 (0x00a4)

HRS41 - BASE Address 1 for Auto-configuration descriptor mechanism

Table 57. HRS41 Register Fields

Bit	Type	Description	Reset
31-0	RW	**BASE_ADDR1 - higher descriptor list base address** Higher 32-bits of the base address required for auto-configuration descriptor mechanism and if 64-bit address is supported.	32'h0

12.26 HRS42 (0x00a8)

HRS42 - Auto-configuration descriptor mechanism enable/disable

Table 58. HRS42 Register Fields

Bit	Type	Description	Reset

Table 58. HRS42 Register Fields

Bit	Type	Description	Reset
4-1	RW	**DESCMECH_TM - Descritor mechanism timeout value** This value determines the interval by which descriptor mechanism timeout is detected. The interval can be computed as below: • 1111b - clk*2^(29+2) • 1110b - clk*2^(28+2) • 1101b - clk*2^(27+2) • ... • 0001b - clk*2^(15+2) • 0000b - clk*2^(14+2) Where clk is the system clock period.	4'h0
0	RW	**DESCMECH_EN - Enable/disable auto-configuration descritor mechanism for Host/PHY pre-initialization sequence update.**	1'h0

12.27 HRS43 (0x00ac)

HRS43 - Error status for auto-configuration descriptor mechanism

Table 59. HRS43 Register Fields

Bit	Type	Description	Reset
3-1	R	**ERROR_VAL - auto-configuration descriptor mechanism error type** Error types: • 000b - first descriptor other than LINK • 001b - invalid descriptor (VAL field = 0) • 010b - DMA error • 011b - timeout (descriptor fetching failed or register update failed) • 100b - wrong mask set for read command (mask[9:5] value is less than mask[4:0]) • 101b - descriptor with command other than READ_BLOCK read after descriptor with command READ_NONBLOCK	3'h0
0	R	**Error occured during auto-configuration descriptor mechanism performance.**	1'h0

12.28 SRS00 (0x0200)

SRS00 - SDMA System Address / Argument 2 / 32-bit block count

Table 60. SRS00 Register Fields

Bit	Type	Description	Reset
31-0	RW	**SAAR - System Address / Argument 2 / 32-bit block count** This field is used as: • 32-bit Block Count register • SDMA system memory address • Auto CMD23 Argument 32-bit block count: This field enables to define number of data blocks for the next transfer. It is used when Host Controller is set in version 4 compatibility mode (SRS15.HV4E=1), Block Count Enable is enabled (SRS03.BCE=1) and 16-bit Block Count for Current Transfer is cleared (SRS01.BCCT=0). Value of this field is decremented after each block transfer. When this field is set 0, no data blocks is transferred. Software should not read this field during data transfer as return unexpected value. Write to this field during data transfer are ignored. When Host Controller is not set in version 4 compatibility mode (SRS15.HV4E=0) or 16-bit Block Count for Current Transfer is not equal to 0, (SRS01.BCCT) defines the block count. • 00000000h - no block transfer • 00000001h..FFFFFFFFh - 1..4294967295 block(s) transfer. System Address: This register is used as base address when SDMA engine is selected (SRS03.DMAE=1 and SRS10.DMASEL=0) and SRS15.HV4E=0. When SDMA stops at SDMA Buffer Boundary, software updates System Address and write to SAAR[31:24] resumes SDMA transfer. Software driver sets address to the next data location in system memory. Auto CMD23 Argument: Auto CMD23 can be used in non-DMA or ADMA2 mode (when SRS15.HV4E=0) or in non-DMA, SDMA, ADMA2 mode (when SRS15.HV4E=1).	32'h0

12.29 SRS01 (0x0204)

SRS01 - Block Size / Block Count

Table 61. SRS01 Register Fields

Bit	Type	Description	Reset

Table 61. SRS01 Register Fields

Bit	Type	Description	Reset
31-16	RW	**BCCT - Block Count For Current Transfer** With this field, the number of data blocks can be defined for next transfer. This register is used when SRS03.BCE is set 1, and SRS15.HV4E is set to 0 or SRS15.H4VE is set to 1 and this field is different than 0, otherwise it will be ignored. When SRS15.HV4E==1 and this field == 0. 32-bit block count register is selected (SRS00.SAAR). The value is decremented after each block transfer. When this field is set 0, no data blocks will be transferred. During data transfer read operation may return invalid value, and write operations are ignored. • 0000h - no block transfer • 0001h..FFFFh - 1..65535 block(s) transfer.	16'h0
14-12	RW	**SDMABB - SDMA Buffer Boundary** In this field, the system address boundary can be set for SDMA engine. The SDMA transfer stops crossing the address boundary and generates the DMA Interrupt (SRS12.DMAINT). After the DMA Interrupt, when the SRS15.HV4E is 0, the software should write new SDMA System Address (SRS00.SAAR / SRS22.DMASA1) in order to resume the SDMA transaction. • 0 - 4k bytes address boundary • 1 - 8k bytes address boundary • 2 - 16k bytes address boundary • 3 - 32k bytes address boundary • 4 - 64k bytes address boundary • 5 - 128k bytes address boundary • 6 - 256k bytes address boundary • 7 - 512k bytes address boundary	3'h0
11-0	RW	**TBS - Transfer Block Size** This field defines block size for block data transfers. During data transfer, read operations may return an invalid value, and write operations are ignored. The software will not set value that exceeds the physically implemented internal FIFO buffer size. The buffer size is equal to $2\hat{}$FIFODEPTH, where FIFODEPTH is the generic parameter of the core. The SD/MMC (memory) uses block size up to 512 bytes. The SDIO can use up to 2048 bytes. • 000h - not used • 001h - 1 data byte • 002h - 2 data bytes • 003h - 3 data bytes • ... • 1FFh - 511 data bytes • 200h - 512 data bytes • ... • 800h - 2048 data bytes • others - not used. Note: It is recommended for the software to use native data block size (512B) in case of multiple data block transfer (SRS03.MSBS==1). Using smaller block may cause unexpected response error when flow control is activated (i.e. SDCLK is disabled) during response transfer.	12'h0

12.30 SRS02 (0x0208)

SRS02 - Argument 1

Table 62. SRS02 Register Fields

Bit	Type	Description	Reset
31-0	RW	**ARG1 - Command Argument 1** This field contains 32-bits argument for command issued by SRS03.CIDX file write.	32'h0

12.31 SRS03 (0x020c)

SRS03 - Command/Transfer Mode

Table 63. SRS03 Register Fields

Bit	Type	Description	Reset
29-24	RW	**CIDX - Command Index** This field contains a command number (index) of the command to be sent. The index can be defined in range 00-63, which means all commands (CMD00-CMD63 and ACMD00-ACMD63) defined in related specifications are supported. Writing this filed triggers the actual command transfer. This field is to be written only when Command Inhibit CMD bit is 0 in Present State Register (SRS09.CICMD). To check the list of available commands, refer to the appropriate card/device specifications.	6'h0
23-22	RW	**CT - Command Type** This field defines specific type of command. • Normal Command (0) - used by default when other command types are not intended to be used • Suspend Command (01b) - not used • Resume Command (10b) - not used • Abort Command (11b) - used when the software wants to stop the current data transfer (read or write data transfer). The read transfer ends by stopping transfer to the internal buffer. The write transfer ends with releasing DAT line to High-Z state. Then, after sending an Abort Command, the software will issue the software reset. The Suspend and Resume Mechanism is not supported by the SD Host version 4.00 and later, and the Suspend and Resume Commands will not be used.	2'h0
21	RW	**DPS - Data Present Select** Set to 1 for commands which transfer data (i.e. read or write data using DAT line). Set to 0 for all other commands, including: • Commands using only CMD line • Commands with busy signalized on DAT[0] line (SRS03.RTS=11b)	1'h0

Table 63. SRS03 Register Fields

Bit	Type	Description	Reset
20	RW	**CICE - Command Index Check Enable** When set to 1, the host checks if the Command Index field in the response is equal to the SRS03.CIDX value. When 0, the check is not performed and Command Index field of the response is ignored. Recommended settings depends on response type, see following table for details: • SRS03.RTS=00: 0 - No Response • SRS03.RTS=01: 0 - R2 • SRS03.RTS=10: 0 - R3, R4 • SRS03.RTS=10: 1 - R1, R5, R6, R7 • SRS03.RTS=11: 1 - R1b, R5b	1'h0
19	RW	**CRCCE - Command CRC Check Enable** When set to 1, the host checks if the CRC field of the response is valid. When 0, the CRC check is disabled and the CRC field of the response is ignored. The CRC check should be disabled for responses which do not contain an actual CRC value (some responses contain all 1s in place of the CRC field), and enabled for all other kinds of responses. Recommended settings depends on response type, see following table for details: SRS03.RTS=00: 0 - No Response SRS03.RTS=01: 1 - R2 SRS03.RTS=10: 0 - R3, R4 SRS03.RTS=10: 1 - R1, R5, R6, R7 SRS03.RTS=11: 1 - R1b, R5b	1'h0
18	RW	**SCF - Sub Command Flag** This bit is added from Version 4.10 to distinguish a main command or sub command. When issuing a main command, this bit is set to 0 and when issuing a sub command, this bit is set to 1. Setting of this bit is checked by Sub Command Status in Present State register SRS09.SCMDS). Host Driver manages whether main or sub command. Host Controller does not refer to this bit to issue the command.	1'h0
17-16	RW	**RTS - Response Type Select** Defines the expected response length. 00b - no response 01b - 136-bit response 10b - 48-bit response 11b - 48-bit response with BUSY Every command implies one of the response types listed above. To check the response type corresponding to a given command, please refer to the appropriate card/device specifications.	2'h0
8	RW	**RID - Response Interrupt Disable** When set to 1, the Command Complete Interrupt (SRS12.CC) will be disabled. The host will ignore the SRS13.CC_SE and behave as the SRS13.CC_SE would be 0. When set to 0, the SRS12.CC will be enabled or disabled depend on the SRS13.CC_SE bit only.	1'h0
7	RW	**RECE - Response Error Check Enable** When set 1, the host will look after R1/R5 responses. If any error will be detected in the response, the SRS12.ERSP bit is set to 1. The software will set this bit only when R1/R5 response is expected. The software will set SRS03.RID and RECE bits to 1 when the host checks R1/R5 errors. And both bits will be clear to 0, when the Software Driver will checks R1/R5 errors. On response error, the SRS12.ERSP bit (in Interrupt Status) is set 1.	1'h0

Table 63. SRS03 Register Fields

Bit	Type	Description	Reset
6	RW	**RECT - Response Type R1/R5** Select R1 or R5 response type for the response content checker. Listed below error bits will be evaluated. RECT = 0, Response Type - R1 (SD Memory): • bit 31 OUT_OF_RANGE • bit 30 ADDRESS_ERROR • bit 29 BLOCK_LEN_ERROR • bit 26 WP_VIOLATION • bit 25 CARD_IS_LOCKED • bit 23 COM_CRC_ERROR • bit 21 CARD_ECC_FAILED • bit 20 CC_ERROR • bit 19 ERRORRECT RECT = 1, Response Type - R5 (SDIO): • bit 7 COM_CRC_ERROR • bit 3 ERROR • bit 1 FUNCTION_NUMBER • bit 0 OUT_OF_RANGE This field is ignored when SRS03.RECE=0.	1'h0
5	RW	**MSBS - Multi/Single Block Select** Multi-block or single-block data transfer can be selected with this field. 0 - Single-block 1 - Multi-block This field is hardware-protected by Command Inhibit DAT bit in Present State Register (SRS09.CIDAT). When CIDAT=1, all writes to this field are ignored.	1'h0
4	RW	**DTDS - Data Transfer Direction Select** Selects direction of data transfer for commands with DPS=1. • 0 - Write • 1 - Read For commands with SRS03.DPS=0, this field is ignored. This field is hardware-protected by Command Inhibit DAT bit in Present State Register (SRS09.CIDAT). When CIDAT=1, all writes to this field are ignored.	1'h0
3-2	RW	**ACE - Auto CMD Enable** The field allows to send one additional command to the card/device when the command is issued. • 00b - No auto command • 01b - Auto CMD12 • 10b - Auto CMD23 • 11b - Auto CMD Auto Select If Auto CMD disable (00b) is set, the host does not send any additional command. This setting will be used when auto command is not required or not intended. If Auto CMD12 (01b) is set, the host sends CMD12 (Abort) automatically when last block of multi-block transfer is completed. If Auto CMD23 (10b) is set, the host sends CMD23 (Set Block Count) automatically before issued transfer data command. An argument of this command can be set in SRS00. If Auto CMD Auto Select (11b) is set, the host sends CMD12 or CMD23 according to result of identification process of card. If SRS15.CMD23E == 1 the host sends CMD23 as when as Auto CMD23 is set. If SRS15.CMD23E == 0 the host sends CMD12 as when as Auto CMD12 is set. On any error the issued command will not be sent. This field is hardware-protected by Command Inhibit DAT bit in Present State Register (SRS09.CIDAT). When SRS09.CIDAT=1, all writes to this field are ignored.	2'h0

Table 63. SRS03 Register Fields

Bit	Type	Description	Reset
1	RW	**BCE - Block Count Enable** When set to 1, active block count register is enabled for the next data transfer. The active register is either SRS01.BCCT or SRS00.SAAR. Transfer of each block automatically decrements the counter value. The multi-block transfer ends when the counter reaches 0. So the finite transfer can not be performed with this setting. When 0, block counting is disabled, and SRS01.BCCT retains its value. The transfer will be infinite in non-DMA and SDMA modes. For ADMA mode the transfer can be infinite or finite. The finite transfer ends on reading the descriptor with END status (so the transfer length is designated by the table of descriptors). In case of infinite transfer, the software will explicitly set ABORT command type to stop transfer. This field is hardware-protected by Command Inhibit DAT bit in Present State Register (SRS09.CIDAT). When CIDAT=1, all writes to this field are ignored.	1'h0
0	RW	**DMAE - DMA Enable** When set to 1, it enables DMA functionality. DMA can be enabled only if it is supported as indicated in the DMA Support in the SRS16.DMAS register. If DMA is not supported (due to host configuration), this bit is ignored. This field is hardware-protected by Command Inhibit DAT bit in Present State Register (SRS09.CIDAT). When SRS09.CIDAT=1, all writes to this field are ignored. Note: The ADMA2 mode uses only the finite transfer mode, i.e. this bit is to be set 1.	1'h0

12.32 SRS04 (0x0210)

The SRS04 - SRS07 registers store the response returned by the card.

The mapping of the actual device response and the SRS04 - SRS07 contents depends on the type of response. The type of response is determined by the RTS field (Response Type) for all user-defined commands.

The separate cases are the Auto-CMD12 response (called R1b in the SD Memory Specification) and Auto-CMD23 response (called R1 in the SD Memory Specification). Auto-CMD12 and Auto-CMD23 responses are handled by the core automatically and goes to the SRS07 register regardless of the RTS value.

SRS04-SRS07 relation to received response field:

- Auto-CMD12 resp: Response field R[39:8] - RESP3[31:0]
- Auto-CMD23 resp: Response field R[39:8] - RESP3[31:0]
- No response: RTS=00b
- 136-bit: RTS=01b, Response field R[127:8] - RESP3[23:0], RESP2[31:0], RESP1[31:0], RESP0[31:0]
- 48-bit: RTS=10b, Response field R[39:8] - RESP0[31:0]
- 48-bit with BUSY: RTS=11b, Response field R[39:8] - RESP0[31:0]

Implementation note: Registers value are undefined after reset, and will be valid after response is received.

Table 64. SRS04 Register Fields

Bit	Type	Description	Reset
31-0	R	**RESP0 - Response Register #0**	32'h0

12.33 SRS05 (0x0214)

Described in SRS04.

Table 65. SRS05 Register Fields

Bit	Type	Description	Reset
31-0	R	**RESP1 - Response Register #1**	32'h0

12.34 SRS06 (0x0218)

Described in SRS04.

Table 66. SRS06 Register Fields

Bit	Type	Description	Reset
31-0	R	**RESP2 - Response Register #2**	32'h0

12.35 SRS07 (0x021c)

Described in SRS04.

Table 67. SRS07 Register Fields

Bit	Type	Description	Reset
31-0	R	**RESP3 - Response Register #3**	32'h0

12.36 SRS08 (0x0220)

SRS08 - Data Buffer

Table 68. SRS08 Register Fields

Bit	Type	Description	Reset
31-0	RW	**BDP - Buffer Data Port** The field is to access the internal buffer (data block) in non-DMA transfer mode. 8-bit, 16-bit, or 32-bit access to SRS08 is possible with the following restrictions: - Only sequential contiguous access in Little Endian mode is possible. For example, if the software accesses BDP[7:0], then the next transfer will access BDP[15:8]. No byte skipping is allowed. - Each new block will start at the least significant byte of BDP, which is BDP[7:0]. - If the block size is not a multiple of 32-bits, and the software accesses BDP using 32-bit words, then the excess bytes of the last word are ignored. This allows the software driver to use only 32-bit data transfers regardless of the block size. - Access to the register with precaution - the FIFO pointers can be damaged when buffer is not ready or when number of accesses exceed the transfer block size (SRS01.TBS). Following shows all transfers (byte enable variations) that are allowed on SRS08: Transfer width = 32-bit: • be[3:0] = 'b1111 -> BDP[31:0] Transfer width = 16-bit: • be[3:0] = 'b0011 -> BDP[15:0] • be[3:0] = 'b1100 -> BDP[31:16] Transfer width = 8-bit: • be[3:0] = 'b0001 -> BDP[7:0] • be[3:0] = 'b0010 -> BDP[15:8] • be[3:0] = 'b0100 -> BDP[23:16] • be[3:0] = 'b1000 -> BDP[31:24]	32'h0

12.37 SRS09 (0x0224)

SRS09 - Present State Register

Table 69. SRS09 Register Fields

Bit	Type	Description	Reset
28	R	**SCMDS - Sub Command Status** The SRS03 register and Response registers (SRS04-SRS07) are commonly used for main command and sub command. This status is used to distinguish which response error statuses, main command or sub command, indicated the Error Interrupt Status register. Just before reading of this register, the SRS03.SCF is copied to this status. This status is effective when not only Response Error interrupt is generated (SRS12.ERSP) but also data error interrupt is generated with Command Not Issued by Error (SRS09.CNIBE) or Auto CMD Error (SRS12.EAC) interrupt is generated with Command Not Issued by Error by Auto CMD12 (SRS15.CNIACE). SRS09.SCMDS indicate which command is not issued (main or sub).	1'h0
27	R	**CNIBE - Command Not Issued By Error** Setting of CNIBE status indicates that a command cannot be issued to an error, except Auto CMD12 error. (Equivalent error status by Auto CMD12 error is defined as SRS15.CNIACE.) This status is set to 1 when Host Controller cannot issue a command after setting Command register.	1'h0

Table 69. SRS09 Register Fields

Bit	Type	Description	Reset
26	R	**LVSIRSLT - LVS Identification Result** Result of the Low Voltage Signaling Identification. This bit contains a valid information only when LVS Identification Execution bit has changed from 1 to 0. This field is cleared when any of following condition is met: (a) SD Bus Power for VDD1 is set to 0 (b) Card Inserted indicates card removal (c) HRS00.SWR (software reset)	1'h0
24	R	**CMDSL - CMD Line Signal Level** The value is equal to the actual signal level on CMD line of the SD interface (pad_mem_cmd). Is useful for debugging purposes.	1'h0
23-20	R	**DATSL1 - DAT[3:0] Line Signal Level** The value is equal to the actual signal level on DAT input pad of the SD/eMMC interface: • SRS09.23 - pad_mem_data[3] • SRS09.22 - pad_mem_data[2] • SRS09.21 - pad_mem_data[1] • SRS09.20 - pad_mem_data[0]	4'h0
19	R	**WPSL - Write Protect Switch Pin Level** The value is equal to the actual signal level on Write Protect pad of the SD/eMMC interface (*pad_mem_wpbar*). • 1 - means that the write operation is enabled • 0 - means that the write operations is disabled	1'h0
18	R	**CDSL - Card Detect Pin Level** The value is equal to the inverted signal level on Card Detect pin of the SD/eMMC interface (*pad_mem_ctrl_0*). • 1 - means that the card is inserted • 0 - means no card is inside the slot Debouncing is not performed on CDSL, therefore the use of Card Inserted (CI) bit is recommended during normal work. CDSL bit is useful only for debugging purposes.	1'h0
17	R	**CSS - Card State Stable** Indicates if Card Detect Pin Level (CDSL) is stable. • 1 - means that the CDSL value is stable • 0 - means that the CDSL is not stable (during card insertion/removal or during the reset) Field is useful for debugging purposes.	1'h0
16	R	**CI - Card Inserted** Indicates if the card is inserted inside the slot. • 0 - no card in slot • 1 - card is inserted Unlike SRS09.CDSL, value of SRS09.CI bit is guaranteed to be stable (i.e. debouncing is performed on this bit). Use of this bit is recommended during the normal operation of host.	1'h0
11	R	**BRE - Buffer Read Enable** This field represents data buffer (SRS08.BDP) state for read transfer in non-DMA mode. • 1 - valid data can be read from the data buffer • 0 - no valid data inside the data buffer After reading the entire data block, this bit changes to 0.	1'h0

Table 69. SRS09 Register Fields

Bit	Type	Description	Reset
10	R	**BWE - Buffer Write Enable** This bit represents data buffer (SRS08.BDP) state for write transfer in non-DMA mode. • 1 - data can be written to the data buffer • 0 - data cannot be written After reading the entire data block, this changes to 0. This bit will be cleared in case of SBGR at non-DMA write transfer (even if the internal buffer is ready). The buffer must not be written after the SBGR. If the BWR was set, the only action from the S/W is to clear the interrupt status.	1'h0
9	R	**RTA - Read Transfer Active** Indicates the status of the read data transfer. • 0 - no data read transfer in progress • 1 - data read transfer in progress Bit is set 1 after sending the read command, or after restarting the read transfer by the Continue Request (SRS10.CREQ). Bit is set 0 by the hardware after the last block of the read transfer, or after stopping the read transfer by the Stop at Block Gap Request (SRS10.SBGR). In both cases, the entire data is to be read by the system from the internal data buffer before setting this bit to 0. In other words, SRS09.RTA=0 means that the entire data is already transferred to the system, and internal data buffer is empty.	1'h0
8	R	**WTA - Write Transfer Active** Indicates the status of the write data transfer. • 0 - no data write transfer in progress • 1 - data write transfer in progress Bit is set 1 after sending the write command, or after restarting the write transfer by the Continue Request (SRS10.CREQ). Bit is set 0 by the hardware after the last block of the write transfer, or after stopping the write transfer by the Stop at Block Gap Request (SRS10.SBGR). In both cases, the entire data has to be transferred to the card from the internal data buffer before setting this bit to 0. In other words, WTA=0 means that the entire data is already transferred to the card, and CRC response for the last data block is already received.	1'h0
7-4	R	**DATSL2 - DAT[7:4] Line Signal Level** The value is equal to the actual signal level on DAT input pad of the SD/eMMC interface: • SRS9.7 - pad_mem_data[7] • SRS9.6 - pad_mem_data[6] • SRS9.5 - pad_mem_data[5] • SRS9.4 - pad_mem_data[4]	4'h0
2	R	**DLA - DAT Line Active** Indicates if the DAT lines of SD interface are currently in use. • 1 - DAT lines are active (in use) • 0 - DAT lines are released (not in use) This bit set to 1, when Read or Write Transfer bits are active (SRS09.RTA=1 or SRS09.WTA=1), or if the card indicates busy state on the DAT lines. The card can become busy immediately after the write operation, or after command which requires response with busy. Falling edge of this bit (change from 1 to 0) directly triggers Transfer Complete Interrupt (SRS12.TC).	1'h0
1	R	**CIDAT - Command Inhibit DAT** Indicates if the host can issue a command which uses DAT line. Commands which use DAT line include write and read data commands and commands with busy response. • 1 - command using DAT line cannot be sent • 0 - command using DAT line can be sent When CIDAT=1 then the SRS03[15:0] is write-protected. The software can write SRS03[15:0] only when CIDAT=0.	1'h0

Table 69. SRS09 Register Fields

Bit	Type	Description	Reset
0	R	**CICMD - Command Inhibit CMD** Indicates if the host can issue a command. • 0 - command can be sent • 1 - command cannot be sent If this bit is 0, indicates the CMD line is not in use and the Host Controller can issue an SD command using the CMD line. This bit is set immediately after the CI is written, indicating start of command transmission. 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 the Command Inhibit CMD is 0. Change from 1 to 0 directly triggers Command Complete Interrupt (SRS12.CC).	1'h0

12.38 SRS10 (0x0228)

SRS10 - Host Control 1 (General / Power / Block-Gap / Wake-Up)

Table 70. SRS10 Register Fields

Bit	Type	Description	Reset
26	RW	**WORM - Wakeup Event Enable On SD Card Removal** When set to 1, enables wake-up event via Card Removal assertion in the SRS12.CR register.	1'h0
25	RW	**WOIS - Wake-Up Event Enable On Card Inserted** When set to 1, enables wake-up event via Card Insertion assertion in the SRS12.CIN register.	1'h0
24	RW	**WOIQ - Wakeup Event Enable On Card Interrupt** When set to 1, enables wake-up event via Card Interrupt assertion in the SRS12.CINT	1'h0
19	RW	**IBG - Interrupt At Block Gap** When set to 1, enables interrupt detection at the block gap for a multiple block transfer. This bit is valid only in SD4 mode. If the SD card cannot signal an interrupt during a multiple block transfer, this bit should be set to 0.	1'h0
18	RW	**RWC - Read-Wait Control** When set to 1, enables Read Wait control. The Read Wait function is optional for SDIO cards. If the card does not support read wait, this bit would never be set to 1; otherwise, DAT line conflict may occur.	1'h0
17	W1S	**CR - Continue Request** When set to 1, restarts the transfer previously stopped using the Stop At Block Gap. The software will set SRS10.SBGR (Stop At Block Gap) bit to 0 before setting the (CR) Continue Request. When SRS10.SBGR=1, then all write operations to Continue Request are ignored. Clearing SRS10.SBGR can be done before or simultaneously with writing the CR. Continue Request bit is cleared automatically by the host when SRS09.DLA (Dat Line Active) changes from 0 to 1, indicating the actual restart of the transfer.	1'h0

Table 70. SRS10 Register Fields

Bit	Type	Description	Reset
16	RW	**SBGR - Stop At Block Gap Request** When set to 1, orders the stop executing read and write transaction at the next possible block gap for non-DMA, SDMA and ADMA transfers. The software will maintain SBGR=1 until the current transfer is complete (typically by waiting for - Transfer Complete bit). After Transfer Complete event, the software will set SBGR back to 0. In case of the read transfer, the host stops after the next data block received from the card. This uses the Read-Wait mechanism if it is enabled by SRS10.RWC, or stops the card clock (*pad_mem_webar_t*) if Read-Wait is disabled. In the case of the write transfer, host stops after the last block written to the data buffer. The host sends all data already written to the internal data buffer before stopping the transfer. In case of stopping non-DMA write transfer, the software will set this bit only at block gap (block unit (SD mode)).	1'h0
11-9	RW	**BVS - SD Bus Voltage Select** This field is used to configure VDD1 voltage level.The state of this field directly drives pad_bv port. • 000b-100b - Reserved • 101b - 1.8V (typical) for embedded • 110b - 3.0V (typical) • 111b - 3.3V (typical) • others - Reserved	3'h0
8	RW	**BP - SD Bus Power for VDD1** When set to 1, the VDD1 voltage is supplied to card/device. The state of this bit directly drives *pad_mem_ctrl_1* pad. Setting bit to 0 cause that host stops driving SDCLK, CMD/DAT lines. If the device is connected to the host, lines go low before disabling VDD1. The host will set this bit automatically to 0 when card is removed from the slot (i.e. after high to low transition on pad_mem_ctrl_0 pin). This is to provide the hot removal support.	1'h0
7	RW	**CDSS - Card Detect Signal Selection** A card detection mechanism will base on either pad_mem_ctrl_0 port or register value. 0 - pad_mem_ctrl_0 pin (normal mode) 1 - CDTL(SRS10.6) bit (testing mode)	1'h0
6	RW	**CDTL - Card Detect Test Level** Designates card insertion status when SRS10.CDSS=1. Bit provided for test purposes. 0 - no card 1 - card inserted	1'h0
5	RW	**EDTW - Extended Data Transfer Width** This bit is to enable/disable 8-bit DAT bus width mode. 0 - bus width selected by SRS10.DTW 1 - 8-bit mode enabled	1'h0

Table 70. SRS10 Register Fields

Bit	Type	Description	Reset
4-3	RW	**DMASEL - DMA Select** In this field the DMA mode can be selected. The field behaviour depends on the Host Controller Compatibility bit (SRS15.HV4E). Host Controller version 3.00 compatible mode (SRS15.HV4E=0) 00b - SDMA mode 01b - Reserved 10b - ADMA2 (32-bit Address) 11b - ADMA2 (64-bit Address) Host Controller version 4.00 compatibility mode (SRS15.HV4E=1) 00b - SDMA mode 01b - Not Used 10b - ADMA2 mode (ADMA3 is not supported or disabled) 11b - ADMA2 or ADMA3 is selected The ADMA2 address bus width is configured by 64-bit Addressing bit in Host Controller 2 register when SRS15.HV4E=1.	2'h0
2	RW	**HSE - High Speed Enable** Selects operating mode to Default Speed (HSE=0) or High Speed (HSE=1). The maximum SD clock frequency is defined as 0-25MHz in the default speed mode, and 0-50MHz in the High Speed mode.	1'h0
1	RW	**DTW - Data Transfer Width** Bit used to configure DAT bus width to 1 or 4. 0 - 1-bit mode 1 - 4-bit mode This bit is ignored when the SRS10.EDTW is set 1 (8-bit mode selected).	1'h0
0	RW	**LEDC - LED Control** State of this bit directly drives led port of the host in order to control the external LED diode. LEDC=1 will switch LED on, while LEDC=0 will switch it off. The software will switch LED on to caution the user not to remove the card while the transfer is in progress.	1'h0

12.39 SRS11 (0x022c)

SRS11 - Host Control 2 (Clock, Timeout, Reset)

Table 71. SRS11 Register Fields

Bit	Type	Description	Reset

Table 71. SRS11 Register Fields

Bit	Type	Description	Reset
26	W1S	**SRDAT - Software Reset For DAT Line** When set to 1, resets the logic related to the data path, including data buffers and the DMA logic. The following registers and bits are cleared: SRS08 register: • Buffer SRS09 register: • Buffer Read Enable • Buffer Write Enable • Read Transfer Active • Write Transfer Active • DAT Line Active • Command Inhibit DATSRS10 register: • Continue Request • Stop At Block Gap Request SRS12 register: • Buffer Read Ready • Buffer Write Ready • DMA Interrupt • Block Gap Event • Transfer Complete After completing the reset operation, SRS11.SRDAT bit is automatically cleared. It takes some time to complete the reset operation, so the software will wait until SRS11.SRDAT=0, and continue the other operations only when SRS11.SRDAT=0.	1'h0
25	W1S	**SRCMD - Software Reset For CMD Line** When set to 1, resets the logic related to the command generation and response checking. The following registers and bits are cleared: • SRS09 register: Command Inhibit CMD • SRS12 register: Command Complete After completing the reset operation, SRS11.SRCMD bit is automatically cleared. It takes some time to complete the reset operation, so the software will wait until SRCMD=0, and continue the other operations only when SRS11.SRCMD=0.	1'h0
24	W1S	**SRFA - Software Reset For All** When set to 1, the entire slot is reset.After completing the reset operation, SRFA bit is automatically cleared. It takes some time to complete the reset operation, so the software will wait until SRFA=0, and continue the other operations only when SRFA=0. Additionally, after SRFA, software should reset and reinitialize card inserted to the slot. SD Card Power may be enabled 1 ms after this bit is cleared to ensure SD Card has been reset properly.	1'h0
19-16	RW	**DTCV - Data Timeout Counter Value** This value determines the interval by which DAT line timeouts are detected. The interval can be computed as below: • 1111b - Reserved • 1110b - t_sdmclk*2^(27+2) • 1101b - t_sdmclk*2^(26+2) • ... • 0001b - t_sdmclk*2^(14+2) • 0000b - t_sdmclk*2^(13+2) Where t_sdmclk is the sdmclk clock periodRefer to the Data Timeout Error (SRS12.EDT) register for information on factors which generate data timeouts.	4'h0

Table 71. SRS11 Register Fields

Bit	Type	Description	Reset
15-8	RW	**SDCFSL - SDCLK Frequency Select (lower part)** This register and SRS11.SDCFSH are used to calculate frequency of SDCLK clock. The SDCLK frequency is calculated with following expressions: - sdclk = sdmclk; when (N=0) - sdclk = sdmclk/2N; when (N>0) Variable N is concatenation of SRS11.SDCFSH and SRS11.SDCFSL. The value of SDCFSL, SDCFSH registers can be changed only when SRS11.SDCE (SD Clock Enable)=0.	8'h0
7-6	RW	**SDCFSH - SDCLK Frequency Select (higher part)** This register is an extension to SDCFSL.	2'h0
2	RW	**SDCE - SD Clock Enable** When set to 1, SDCLK clock is enabled. When cleared to 0, SDCLK clock is stopped. The host clears SDCE automatically when card is removed from the slot (i.e. after the high to low transition on *pad_mem_ctrl_0* pad). The SDCLK clock should be stopped by the software when changing the clock divider (i.e. SDCE bit will be cleared before writing SRS11.SDCFSL, SRS11.SDCFSH).	1'h0
1	R	**ICS - Internal Clock Stable** When read as 1, indicates that the clock on sdmclk pin of the host is stable after setting ICE to 1. When read as 0, indicates that the clock is not stable yet (for example the external PLL that generates the clock is not yet locked). The value of ICS is equal to the actual signal level on ics pin of the host. The user will connect ics to the external PLL if required. Otherwise, ics should be connected directly to the ice output of the host.	1'h0
0	RW	**ICE - Internal Clock Enable** This field is designated to controls (enable/disable) external clock generator (e.g. PLL). The ICE bits of every slot are logically OR-ed together and then drive the ice pin. It means, the ice pin is 0 only when ICE in 0 for every slot implemented inside the host. The ice pin is 1 if at least one of the ICE bits is set to 1. When set to 0, the clock on sdmclk pin can be stopped externally. If the sdmclk is stopped, then host goes to a very low power state. The hosts registers are still operable (read and written operation are valid) even if the clock on sdmclk is stopped. Setting of the ICE bit does not affect card detection. It means, the card detection works even if the clock on sdmclk is stopped.	1'h0

12.40 SRS12 (0x0230)

SRS12 - Error/Normal Interrupt Status

Table 72. SRS12 Register Fields

Bit	Type	Description	Reset
27	W1C	**ERSP - Response Error** Generated on error detection inside R1 or R5 response. Errors will be checked only if RECE is set 1.	1'h0
25	W1C	**EADMA - ADMA Error** Generated when an error occurs during ADMA read or write transfer. To resolve the cause of the error, the state of the ADMA engine at error occurrence is saved in ADMA Error Status register, and the address of the descriptor processed at error occurrence is provided in ADMA System Address register.	1'h0

Table 72. SRS12 Register Fields

Bit	Type	Description	Reset
24	W1C	**EAC - Auto CMD Error (SD mode only)** Generated when an error occurs during Auto CMD12/Auto CMD23 command transmission. It indicates one of the following conditions: - one of the bits in SRS15 register has changed from 0 to 1, - Auto CMD12 is not executed due to the previous command error.	1'h0
23	W1C	**ECL - Current Limit Error** This fields carries an error/failure reported on the *pad_cle* input pad of the Host Controller. The error/failure generation is located outside of this soft IP. Note: If the external power supply for SD/eMMC device does not monitor and report this type of error, connect the Current Limit Error (*sdphy_dfi_cle* input of the Host Controller core) to 0.	1'h0
22	W1C	**EDEB - Data End Bit Error (SD mode only)** When set to 1, indicates detecting 0 at the end bit position of read data transfer which uses the DAT line, or at the end bit position of the Write CRC Status.	1'h0
21	W1C	**EDCRC - Data CRC Error (SD mode only)** When set to 1, indicates 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. This bit will be set to 1 immediately when conflict on CMD line detected. The conflict is signalized by setting this bit and SRS12.EDT to 1.	1'h0
20	W1C	**EDT - Data Timeout Error (SD mode only)** When set to 1, indicates detecting one of the following timeout conditions: 1. Busy timeout for the response with busy. 2. Busy timeout after Write CRC status. 3. Write CRC Status timeout. 4. Read data timeout. This bit will be set to 1 immediately when conflict on CMD line conflict detected.	1'h0
19	W1C	**ECI - Command Index Error (SD mode only)** When set to 1, indicates that Index error occurs in the command response.	1'h0
18	W1C	**ECEB - Command End Bit Error (SD mode only)** When set to 1, indicates detecting that the end bit of a command response is 0.	1'h0
17	W1C	**ECCRC - Command CRC Error (SD mode only)** When set to 1, indicates that command CRC error has occurred.	1'h0
16	W1C	**ECT - Command Timeout Error** When set to 1, indicates that no response was returned within 64 SDCLK cycles from the end bit of the command.	1'h0
15	R	**EINT - Error Interrupt** This bit is set if any of bits in range SRS12[31:16] is set; The software can check for an error by reading this single bit first.	1'h0
14	R	**CQINT - Command Queuing Interrupt** This interrupt is asserted when at least one of the bits in CQIS register is set. This interrupt is cleared only by clearing the source interrupt in CQIS register.	1'h0
13	R	**FXE - FX Event** If SRS03.RECE is set to 1, and SRS03.RECT is set to 0 this interrupt indicates that 14th bit of response stored as 6th bit of SRS04 register is set to 1. If SRS03.RECE is set to 1 only next response with type R1 containing card status bit 14th equal 0 can clean this interrupt. If SRS03.RECE is set to 0, this interrupt indicates that 14th bit of response stored as 6th bit of SRS04 register is set to 1 except cases: 1. Argument of CMD13 bit 15 is equal 1 - then response won't change value of this interrupt. 2. Issued command does not have a response - then value of this interrupt won't change.	1'h0

Table 72. SRS12 Register Fields

Bit	Type	Description	Reset
8	R	**CINT - Card Interrupt** Indicates the card interrupt. CINT is not sampled by the card clock, so the interrupt can be detected even with SD clock stopped (SRS11.SDCE=0). Also, CINT is not cleared by writing 1. Instead, the software will clear the source of an interrupt inside the card.After detecting the Card Interrupt, the software will stop further interrupt detection by clearing SRS13.CINT_SE to 0. Then, the software will clear the interrupt source inside the card by using the appropriate commands. For the details, please refer to the SDIO Card Specification. After clearing the interrupt source, the card will stop to drive the interrupt signal to the host. Finally, when the interrupt service routine is finished, the interrupt detection can be enabled by setting SRS13.CINT_SE back to 1.	1'h0
7	W1C	**CR - Card Removal** Generated when the SRS09.CI bit changes from 1 to 0, indicating card removal event. When read as 1, indicates that the card was removed from the slot. When read as 0, indicates that the card state is stable (still inserted or removed) or that the debouncing is in progress.	1'h0
6	W1C	**CIN - Card Insertion** Generated when the SRS09.CI bit changes from 0 to 1, indicating card insertion. When read as 1, indicates that the card was inserted to the slot. When read as 0, indicates that the card state is stable (still inserted or removed) or that the debouncing is in progress.	1'h0
5	W1C	**BRR - Buffer Read Ready** Generated when the BRE changes from 0 to 1, indicating that the data buffer can be read by the software. This field works differently in the SD Tuning Sequence, i.e. when Sampling Clock Select (SRS15.SCS) equals 1. It is set to 1 on the tune step completion despite of the step's result. As per the Standard, during the SD tuning, none of the interrupts is notified except Buffer Read Ready.	1'h0
4	W1C	**BWR - Buffer Write Ready** Generated when the BWE changes from 0 to 1, indicating that the data buffer can be written by the software.	1'h0
3	W1C	**DMAINT - DMA Interrupt** In SDMA mode, DMA interrupt is generated when the host controller detects the Host SDMA Buffer boundary. In ADMA mode, DMA interrupt is generated when the INT flag is set in a currently serviced ADMA descriptor.	1'h0
2	W1C	**BGE - Block Gap Event** Generated when the read/write transaction is stopped at a block gap as the result of setting SRS10.SBGR to 1.	1'h0
1	W1C	**TC - Transfer Complete** SD Mode: Generated when the transfer which uses the DAT line is complete. Transfers which use the DAT line include the read/write transfers and commands with a busy response. In case of the read transfer, TC indicates that the entire data was transferred from the card to the host system (i.e. the host FIFO is empty after reading the last data block). In case of the write transfer, TC indicates that the entire data was transferred from the host to the card (i.e. the host FIFO is empty after writing the last data block), and the card accepted the data (busy signal released after the last block). In the case of the command with a busy response, TC indicates that the busy signal is released after the response.	1'h0
0	W1C	**CC - Command Complete** Generated when the end bit of the response is received, except the response for Auto-CMD12 command. Auto-CMD12 command does not generate CC.	1'h0

12.41 SRS13 (0x0234)

SRS13 - Error/Normal Status Enable

Table 73. SRS13 Register Fields

Bit	Type	Description	Reset
27	RW	**ERSP_SE - Response Error Status Enable (SD mode only)** 1 - enabled 0 - masked	1'h0
25	RW	**EADMA_SE - ADMA Error Status Enable (SD mode only)** 1 - enabled 0 - masked	1'h0
24	RW	**EAC_SE - Auto CMD Error Status Enable (SD mode only)** 1 - enabled 0 - masked	1'h0
23	RW	**ECL_SE - Current Limit Error Status Enable (SD mode only)** 1 - enabled 0 - masked	1'h0
22	RW	**EDEB_SE - Data End Bit Error Status Enable (SD mode only)** 1 - enabled 0 - masked	1'h0
21	RW	**EDCRC_SE - Data CRC Error Status Enable (SD mode only)** 1 - enabled 0 - masked	1'h0
20	RW	**EDT_SE - Data Timeout Error Status Enable (SD mode only)** 1 - enabled 0 - masked	1'h0
19	RW	**ECI_SE - Command Index Error Status Enable (SD mode only)** 1 - enabled 0 - masked	1'h0
18	RW	**ECEB_SE - Command End Bit Error Status Enable (SD mode only)** 1 - enabled 0 - masked	1'h0
17	RW	**ECCRC_SE - Command CRC Error Status Enable (SD mode only)** 1 - enabled 0 - masked	1'h0
16	RW	**ECT_SE - Command Timeout Error Status Enable (SD mode only)** 1 - enabled 0 - masked	1'h0
14	RW	**CQINT_SE - Command Queuing Status Enable** 1 - enabled 0 - masked	1'h0
13	RW	**FXE_SE - FX Event Status Enable** 1 - enabled 0 - masked	1'h0
8	RW	**CINT_SE - Card Interrupt Status Enable** 1 - enabled 0 - masked	1'h0
7	RW	**CR_SE - Card Removal Status Enable** 1 - enabled 0 - masked	1'h0

Table 73. SRS13 Register Fields

Bit	Type	Description	Reset
6	RW	**CIN_SE -Card Insertion Status Enable** 1 - enabled 0 - masked	1'h0
5	RW	**BRR_SE - Buffer Read Ready Status Enable** 1 - enabled 0 - masked	1'h0
4	RW	**BWR_SE - Buffer Write Ready Status Enable** 1 - enabled 0 - masked	1'h0
3	RW	**DMAINT_SE - DMA Interrupt Status Enable** 1 - enabled 0 - masked	1'h0
2	RW	**BGE_SE - Block Gap Event Status Enable** 1 - enabled 0 - masked	1'h0
1	RW	**TC_SE - Transfer Complete Status Enable** 1 - enabled 0 - masked	1'h0
0	RW	**CC_SE - Command Complete Status Enable** 1 - enabled 0 - masked	1'h0

12.42 SRS14 (0x0238)

SRS14 - Error/Normal Signal Enable

Table 74. SRS14 Register Fields

Bit	Type	Description	Reset
27	RW	**ERSP_IE - Response Error Interrupt Enable** 1 - enabled 0 - masked	1'h0
25	RW	**EADMA_IE - ADMA Error Interrupt Enable (SD mode only)** 1 - enabled 0 - masked	1'h0
24	RW	**EAC_IE - Auto CMD Error Interrupt Enable (SD mode only)** 1 - enabled 0 - masked	1'h0
23	RW	**ECL_IE - Current Limit Error Interrupt Enable (SD mode only)** 1 - enabled 0 - masked	1'h0
22	RW	**EDEB_IE - Data End Bit Error Interrupt Enable (SD mode only)** 1 - enabled 0 - masked	1'h0
21	RW	**EDCRC_IE - Data CRC Error Interrupt Enable (SD mode only)** 1 - enabled 0 - masked	1'h0

Table 74. SRS14 Register Fields

Bit	Type	Description	Reset
20	RW	**EDT_IE - ata Timeout Error Interrupt Enable (SD mode only)** 1 - enabled 0 - masked	1'h0
19	RW	**ECI_IE - Command Index Error Interrupt Enable (SD mode only)** 1 - enabled 0 - masked	1'h0
18	RW	**ECEB_IE - Command End Bit Error Interrupt Enable (SD mode only)** 1 - enabled 0 - masked	1'h0
17	RW	**ECCRC_IE - Command CRC Error Interrupt Enable (SD mode only)** 1 - enabled 0 - masked	1'h0
16	RW	**ECT_IE - Command Timeout Error Interrupt Enable (SD mode only)** 1 - enabled 0 - masked	1'h0
14	RW	**CQINT_IE - Command Queuing - Interrupt Enable** 1 - enabled 0 - masked	1'h0
13	RW	**FXE_IE - FX Event Interrupt Enable** 1 - enabled 0 - masked	1'h0
8	RW	**CINT_IE - Card Interrupt - Interrupt Enable** 1 - enabled 0 - masked	1'h0
7	RW	**CR_IE - Card Removal Interrupt Enable** 1 - enabled 0 - masked	1'h0
6	RW	**CIN_IE - Card Insertion Interrupt Enable** 1 - enabled 0 - masked	1'h0
5	RW	**BRR_IE - Buffer Read Ready Interrupt Enable** 1 - enabled 0 - masked	1'h0
4	RW	**BWR_IE - Buffer Write Ready Interrupt Enable** 1 - enabled 0 - masked	1'h0
3	RW	**DMAINT_IE - DMA Interrupt Enable** 1 - enabled 0 - masked	1'h0
2	RW	**BGE_IE - Block Gap Event Interrupt Enable** 1 - enabled 0 - masked	1'h0
1	RW	**TC_IE - Transfer Complete Interrupt Enable** 1 - enabled 0 - masked	1'h0
0	RW	**CC_IE - Command Complete Interrupt Enable** 1 - enabled 0 - masked	1'h0

12.43 SRS15 (0x023c)

SRS15 - Host Control #2 / Auto CMD Error Status

Table 75. SRS15 Register Fields

Bit	Type	Description	Reset
31	RW	**PVE - Preset Value Enable** Setting this bit to 1 triggers an automatically update of SRS11.SDCFSL, SRS11.SDCFSH, SRS11.CGS, SRS15.DSS registers by the host. Values for an update are taken from SRS24 - SRS27 and depends on SRS15.UMS.	1'h0
29	RW	**A64B - 64-bit Addressing** Specifies the addressing mode for DMA ending. This field is ignored when SRS15.HV4E=0. 0 - 32-bit addressing 1 - 64-bit addressing	1'h0
28	RW	**HV4E - Host Version 4.00 Enable** Selects backward (SD Host 3.00 Version) compatibility mode or SD Host 4.00 Version mode. 0 - Version 3.00 1 - Version 4.00 The software can select system address register SRS00 (when this bit is 0) or SRS23 / SRS22 (when this bit is 1) for the SDMA engine.	1'h0
27	RW	**CMD23E - CMD23 Enable** In result of Card Identification process, Host Driver set this bit to 1 if Card supports CMD23 (SCR[33]==1).	1'h0
26	RW	**ADMA2LM - ADMA2 Length Mode** This bit selects one of ADMA2 Length Modes either 16-bit or 26-bit. 0 - 16-bit Data Length Mode 1 - 26-bit Data Length Mode	1'h0
24	RW	**LVSIEXEC - LVS Identification Execution** Setting this field to 1, generates one pulse on SDCLK output. This bit does not change while DAT[2] stays LOW. After detecting DAT[2] is HIGH, this field automatically changes its value to 0 confirming end of the Low Voltage Sequence.	1'h0
23	W0C	**SCS - Sampling Clock Select** The host updates this bit when the tuning procedure is finished. If this bit is set to 1, this means that the tuning procedure is successfully completed. Otherwise it means that procedure failed and clock tuning logic is disabled. This bit is valid only after the procedure is finished.Writing 1 will be ignored. Writing 0 will reset and disable tuning block.	1'h0
22	RW	**EXTNG - Execute Tuning** This register controls tuning procedure. The procedure starts when the bit is set 1. The procedure can be aborted when the bit is cleared. The bit is read 1 while the procedure is in progress, and 0 when the procedure is finished. SCS = 0, EXTNG = 0 - Reset and disable clock tuning logic SCS = 0, EXTNG = 1 - Reset and restart tuning process SCS = 1, EXTNG = 0 - Stop tuning procedure SCS = 1, EXTNG = 1 - Start retuning (without clock tuning logic reset)	1'h0

Table 75. SRS15 Register Fields

Bit	Type	Description	Reset
21-20	RW	**DSS - Driver Strength Select** This bit controls the electric parameters of I/O driver via *sdphy_dfi_drvss* output of the SD/eMMC Host Controller core). Up to 4 configurations of I/O driver settings can be implemented: • 00 - Driver Type B (default) • 01 - Driver Type A • 10 - Driver Type C • 11 - Driver Type D The bit is ignored when the V18SE is cleared.	2'h0
19	RW	**V18SE - 1.8V Signaling Enable** This bit controls I/O signaling voltage level. If the bit is 0 or 1, the I/O uses the 3.3V or 1.8V signaling, respectively. The SW driver will set this bit 1 when UHS-I mode. Depend on the selected SD interface mode, the software will set this field as follows: - 0 - for Default Speed, High Speed mode - 1 - for UHS-I mode	1'h0
18-16	RW	**UMS - UHS Mode Select** Used to select one of UHS-I modes. 000b - SDR12 001b - SDR25 010b - SDR50 011b - SDR104 100b - DDR50 101b - Reserved 110b - Reserved 111b - Reserved The selected UHS-I mode (when value is in range 000b-100b) will be ignored when V18SE is 0.	3'h0
7	R	**CNIACE - Command Not Issued By Auto CMD12 Error** When read as 1, the command was not executed by the Host due to the previous Auto CMD12 error. When Host detects any error during Auto-CMD12, then all further command generation attempts are blocked. The software reset sequence is needed for recovery. Bit is set to 0, when Auto CMD23 Error is detected (any of bits SRS15[4:1] is set).	1'h0
5	R	**ACRE - Auto CMD Response Error** When read as 1, means an error is detected in response to Auto Command.	1'h0
4	R	**ACIE - Auto CMD Index Error** When read as 1, means that Command Index error occurred in the Auto CMD response.	1'h0
3	R	**ACEBE - Auto CMD End Bit Error** When read as 1, indicates that the end bit of the Auto-CMD response is 0.	1'h0
2	R	**ACCE - Auto CMD CRC Error** When read as 1, indicates a CRC error was detected in the Auto CMD response or conflict on the CMD lines is detected: ACCE(SRS15.2) = 0, ACTE(SRS15.1) = 0 - No error ACCE(SRS15.2) = 0, ACTE(SRS15.1) = 1 - Auto CMD Timeout error detected ACCE(SRS15.2) = 1, ACTE(SRS15.1) = 0 - Auto CMD CRC error detected ACCE(SRS15.2) = 1, ACTE(SRS15.1) = 1 - Conflict on the CMD line detected	1'h0
1	R	**ACTE - Auto CMD Timeout Error** When read as 1, indicates that there was no response within 64 SDCLK clock cycles from the end bit of the Auto CMD or conflict on the CMD lines is detected (see table in SRS15.ACCE field description). If this bit is set to 1, the other error status bits (SRS15[4:2]) are meaningless.	1'h0

Table 75. SRS15 Register Fields

Bit	Type	Description	Reset
0	R	**ACNE - Auto CMD12 Not Executed** When set to 1, means the host cannot issue Auto CMD12 due to some error. If this bit is set to 1, other error status bits (SRS15[4:1]) are meaningless. Bit is updated with 0, when Auto CMD23 Error is detected (any of bits SRS15[4:1] is set).	1'h0

12.44 SRS16 (0x0240)

SRS16 - Capabilities #1

Table 76. SRS16 Register Fields

Bit	Type	Description	Reset
31-30	R	**SLT - Slot Type** These bits inform what type of slot is provided. • 00 - Removable Card Slot • 01 - Embedded Slot for One Device • 10 - Shared Bus Slot • 11 - Reserved	2'h0
29	R	**AIS - Asynchronous Interrupt Support** 0 - not supported	1'h0
28	R	**A64SV3 - 64-bit System Addressing Support** 0 - 64-bit Addressing for V3 is not supported 1 - 64-bit Addressing for V3 is supported	1'h1
27	R	**A64SV4 - 64-bit System Addressing Support for V4** 0 - 64-bit Addressing for V4 is not supported 1 - 64-bit Addressing for V4 is supported	1'h1
26	R	**VS18 - Voltage Support 1.8V** 0 - not supported 1 - supported	1'h1
25	R	**VS30 - Voltage Support 3.0V** 0 - not supported 1 - supported	1'h1
24	R	**VS33 - Voltage Support 3.3V** 0 - not supported 1 - supported	1'h1
23	R	**SRS - Suspend / Resume Support** 0 - not supported 1 - supported The host controller does not support Suspend / Resume mechanism.	1'h0
22	R	**DMAS - SDMA Support** 0 - not supported 1 - supported This bit defines whether the SDMA is supported.	1'h1
21	R	**HSS - High Speed Support** 0 - not supported 1 - supported	1'h1

Table 76. SRS16 Register Fields

Bit	Type	Description	Reset
20	R	**ADMA1S - ADMA1 Support** 0 - not supported 1 - supported	1'h0
19	R	**ADMA2S - ADMA2 Support** 0 - not supported 1 - supported	1'h1
18	R	**8EDS - 8-bit Embedded Device Support** 0 - not supported 1 - supported If this bit is 0, the SRS10.EDTW register is not implemented.	1'h0
17-16	R	**SRS16.MBL - Max Block Length** This value indicates the maximum block size that can be transferred by the host. Three sizes can be defined as indicated below: • 00b - 512 Bytes • 01b - 1024 Bytes • 10b - 2048 Bytes • 11b - Reserved The physical FIFO buffer size is defined by the separate FIFODEPTH generic parameter, and the physical buffer size is equal to 2^FIFODEPTH * 8 bytes. Therefore, the Maximum Block Size defined by MBL will always be less or equal to the physical buffer size.	2'h2
15-8	R	**BCSDCLK - Base Clock Frequency for SD Clock** Field defines the base clock frequency for the SD Clock in 1MHz units. The base clock is the clock sourced to sdmclk pin of the host. The maximum clock frequency supported is between 10MHz to 255MHz. If BCSDCLK = 0, the Host System has to obtain the clock information via another method (i.e. not defined by the specification).	8'hc8
7	R	**TCU - Timeout Clock Unit** Field defines the frequency unit for the SRS16.TCF. 0 - kHz 1 - MHz	1'h1
5-0	R	**TCF - Timeout Clock Frequency** Defines the base clock frequency used to detect Data Timeout Error. The SRS16.TCU bit determines the unit used. 111111b - 63kHz(SRS16.TCU=0) or 63MHz(SRS16.TCU=1) 111110b - 63kHz(SRS16.TCU=0) or 63MHz(SRS16.TCU=1) ... 000001b - 1kHz(SRS16.TCU=0) or 1MHz(SRS16.TCU=1) 000000b - Host System has to obtain the clock information via another method (i.e. not defined by the spec).	6'h32

12.45 SRS17 (0x0244)

SRS17 - Capabilities #2

Table 77. SRS17 Register Fields

Bit	Type	Description	Reset
31	R	**LVSH - Low Voltage Signaling Host** • 1 - LVS Host • 0 - Not LVS Host	1'h1
28	R	**VDD2S - VDD2 Supported** • 1 - VDD2 supported • 0 - VDD2 not supported	1'h0
27	R	**ADMA3SUP - ADMA3 Supported.** • 1 - ADMA3 supported • 0 - ADMA3 not supported	1'h1
23-16	R	**CLKMPR - Clock Multiplier** This field is to be 0 (fixed), as the Clock Multiplier is not supported.	8'h0
15-14	R	**RTNGM - Re-Tuning Modes** Depending on the retuning method, the some restrictions are assumed for the data length between re-tunings. The core can work with supporting one of the three method: • 0 - Mode1: The software driver will use timer to calculate when the re-tuning is to be rerun. The data length between operations is limited to the 4MB. • 1 - Mode2: The driver will use either the re-tuning request (external input pin uhsi_retune_req is used for this purpose) or timer to predict when next retuning should be performed. The data length between operations is limited to the 4MB. • 2 - Mode3: This mode is similar to the mode2 with one exception. The core is able to perform auto retuning during the transmission, so data length limitation is not exists. Mode 3 is currently not supported.The driver can configure the timer by getting the RTNGCNT.This field is to be 0 or 1, because the mode 3 is not supported.	2'h0
13	R	**UTSM50 - Use Tuning for SDR50** • 1 - tuning operation is necessary in SDR50 mode • 0 - tuning operation is not necessary in SDR50 mode	1'h0
11-8	R	**RTNGCNT - Timer Count for Re-Tuning** These bits contain initial value for timer used to starting periodically Re-Tuning Operation. • 0h - Re-Tuning Timer disabled • 1h - 1 second • ... • n - 2^(n-1) seconds • ... • Bh - 1024 seconds • Eh-Ch - Reserved • Fh - Obtain this info in other way	4'h0
6	R	**DRVD - 1.8V Line Driver Type D Supported** • 1 - Driver Type D supported • 0 - Driver Type D not supported	1'h1
5	R	**DRVC - 1.8V Line Driver Type C Supported** • 1 - Driver Type C supported • 0 - Driver Type C not supported	1'h1

Table 77. SRS17 Register Fields

Bit	Type	Description	Reset
4	R	**DRVA - 1.8V Line Driver Type A Supported** • 1 - Driver Type A supported • 0 - Driver Type A not supported	1'h1
3	R	**UHSII - UHS-II / UHS-III Supported** 0 - UHS-II not supported	1'h0
2	R	**DDR50 - DDR50 Supported** • 1 - DDR50 mode supported • 0 - DDR50 mode not supported	1'h1
1	R	**SDR104 - SDR104 Supported** • 1 - SDR104 mode supported • 0 - SDR104 mode not supported	1'h1
0	R	**SDR50 - SDR50 Supported** • 1 - SDR50 mode supported • 0 - SDR50 mode not supported	1'h1

12.46 SRS18 (0x0248)

SRS18 - Capabilities #3

Table 78. SRS18 Register Fields

Bit	Type	Description	Reset
23-16	R	**MC18 - Maximum Current for 1.8V** • 0 - Host System has to obtain the current value via another method • 1 - 4 mA • 2 - 8 mA • 3 - 12 mA • ... • 255 - 1020 mA	8'h20
15-8	R	**MC30 - Maximum Current for 3.0V** • 0 - Host System has to obtain the current value via another method • 1 - 4 mA • 2 - 8 mA • 3 - 12 mA • ... • 255 - 1020 mA	8'h20
7-0	R	**MC33 - Maximum Current for 3.3V** • 0 - Host System has to obtain the current value via another method • 1 - 4 mA • 2 - 8 mA • 3 - 12 mA • ... • 255 - 1020 mA	8'h20

12.47 SRS19 (0x024c)

SRS19 - Capabilities #4

Table 79. SRS19 Register Fields

Bit	Type	Description	Reset
7-0	R	**MC18V2 - Maximum Current for 1.8V VDD2** • 0 - Host System has to obtain the current value via another method • 1 - 4 mA • 2 - 8 mA • 3 - 12 mA • ... • 255 - 1020 mA	8'h20

12.48 SRS20 (0x0250)

SRS20 - Force Event

Each field of this register is related to the specific error status. Writing 1 to field will set the status error. This function is provided for SW debug purpose.

Table 80. SRS20 Register Fields

Bit	Type	Description	Reset
27	W	**ERESP_FE - Force Response Error Event**	1'h0
26	W	**ETUNE_FE - Force Tuning Error Event**	1'h0
25	W	**EADMA_FE - Force ADMA Error Event**	1'h0
24	W	**EAC_FE - Force Auto CMD Error Event**	1'h0
23	W	**ECL_FE - Force Current Limit Error Event**	1'h0
22	W	**EDEB_FE - Force Data End Bit Error Event**	1'h0
21	W	**EDCRC_FE - Force Data CRC Error Event**	1'h0
20	W	**EDT_FE - Force Data Timeout Error Event**	1'h0
19	W	**ECI_FE - Force Command Index Error Event**	1'h0
18	W	**ECEB_FE - Force Command End Bit Error Event**	1'h0
17	W	**ECCRC_FE - Force Command CRC Error Event**	1'h0
16	W	**ECT_FE - Force Command Timeout Error Event**	1'h0
7	W	**CNIACE_FE - Force Command Not Issued By Auto CMD12 Error Event**	1'h0

Table 80. SRS20 Register Fields

Bit	Type	Description	Reset
4	W	**ACIE_FE - Force Auto CMD Index Error Event**	1'h0
3	W	**ACEBE_FE - Force Auto CMD End Bit Error Event**	1'h0
2	W	**ACCE_FE - Force Auto CMD CRC Error Event**	1'h0
1	W	**ACTE_FE - Force Auto CMD Timeout Error Event**	1'h0
0	W	**ACNE_FE - Force Auto CMD12 Not Executed Event**	1'h0

12.49 SRS21 (0x0254)

SRS21 - ADMA Error Status

Table 81. SRS21 Register Fields

Bit	Type	Description	Reset
2	R	**EADMAL - ADMA Length Mismatch Error** This bit is set when: - total data length specified in ADMA descriptors is different from that specified by the Block Count and Block Length fields (if Block Count Enable is set). - total data length cannot be divided into complete blocks of specified length (if Block Count Enable is not set).	1'h0
1-0	R	**EADMAS - ADMA Error State** The value of this field reflects the state of the ADMA state machine. The possible values are: 00b - ST_STOP (ADMA Stopped) 01b - ST_FDS (Fetching descriptor) 10b - not used 11b - ST_TRF (Transfer data)	2'h0

12.50 SRS22 (0x0258)

SRS22 ADMA/SDMA System Address 1

Table 82. SRS22 Register Fields

Bit	Type	Description	Reset
31-0	RW	**DMASA1 - ADMA System Address** This field contains the physical address of the currently processed ADMA descriptor or SDMA system address. The Host Driver will set this register with the descriptors table base address before it starts the ADMA transfers. The Host Driver should not write this register while the data transfer is active. While the ADMA engine is processing the descriptors list, the ADMASA value is always incremented to point the next descriptor to be fetched. If the ADMA Error occurs, the register holds the descriptor address depending on the ADMA Error State (SRS21.EADMAS) register value, as listed in the table below: 00b - Points next of the error descriptor 01b - Points the error descriptor 10b - not used 11b - Points next of the error descriptor The host ADMA engine ignores 2 or 3 least significant bits in this register when the 32-bit or 64-bit addressing is active, respectively. If SRS15.HV4E is set 1 and SDMA engine is selected, this field is used instead of SRS00 to define system memory address. This register incremented and points to the next memory location that will be accessed.	32'h0

12.51 SRS23 (0x025c)

SRS23 ADMA/SDMA System Address 2

Table 83. SRS23 Register Fields

Bit	Type	Description	Reset
31-0	RW	**DMASA2 - ADMA System Address #2** In ADMA mode, if 64-bit addressing is enabled (SRS15.A64B=1), this field holds bits 63-32 of the physical address pointing on ADMA descriptor table. In SDMA mode, if host compatibility with version 4.x and 64-bit addressing are enabled (SRS15.HV4E=1 and SRS15.A64B=1), this field holds bits 63-32 of system address.	32'h0

12.52 SRS24 (0x0260)

SRS24 - Preset Value (Default Speed)

SRS24[31:16] - Default Speed if:

SRS15.V18SE=0

SRS10.HSE=0

HWINIT Register - Note this register is hardware initialized after reset and the value read back will match the IP configuration.

Table 84. SRS24 Register Fields

Bit	Type	Description	Reset
31-30	R	**DSSPV## - Driver Strength Select - Preset Value** This field can be used by the software to update SRS15.DSS.	2'h0
25-16	R	**SDCFSPV## - SDCLK Clock Frequency Select - Preset Value** This field can be used by the software to update SRS11.SDCFSH and SRS11.SDCFSL.	10'h4

12.53 SRS25 (0x0264)

SRS25 - Preset Value (High Speed and SDR12)

SRS25[15:0] - High Speed if:

SRS15.V18SE=0

SRS.HSE=1

SRS25[31:16] - SDR12 if:

SRS15.V18SE=1

SRS15.UMS=000b

HWINIT Register - Note this register is hardware initialized after reset and the value read back will match the IP configuration.

Table 85. SRS25 Register Fields

Bit	Type	Description	Reset
31-30	R	**DSSPV## - Driver Strength Select - Preset Value** This field can be used by the software to update SRS15.DSS.	2'h0
25-16	R	**SDCFSPV## - SDCLK Clock Frequency Select - Preset Value** This field can be used by the software to update SRS11.SDCFSH and SRS11.SDCFSL.	10'h4
15-14	R	**DSSPV## - Driver Strength Select - Preset Value** This field can be used by the software to update SRS15.DSS.	2'h0
9-0	R	**SDCFSPV## - SDCLK Clock Frequency Select - Preset Value** This field can be used by the software to update SRS11.SDCFSH and SRS11.SDCFSL.	10'h2

12.54 SRS26 (0x0268)

SRS26 - Preset Value (SDR25 and SDR50)

SRS26[15:0] - SDR25 if:

SRS15.V18SE=1

SRS15.UMS=001b

SRS26[31:16] - SDR50 if:

SRS15.V18SE=1

SRS15.UMS=010b

HWINIT Register - Note this register is hardware initialized after reset and the value read back will match the IP configuration.

Table 86. SRS26 Register Fields

Bit	Type	Description	Reset
31-30	R	**DSSPV## - Driver Strength Select - Preset Value** This field can be used by the software to update SRS15.DSS.	2'h0
25-16	R	**SDCFSPV## - SDCLK Clock Frequency Select - Preset Value** This field can be used by the software to update SRS11.SDCFSH and SRS11.SDCFSL.	10'h1
15-14	R	**DSSPV## - Driver Strength Select - Preset Value** This field can be used by the software to update SRS15.DSS.	2'h0
10	R	**CGSPV## - Clock Generator Select - Preset Value** This field can be used by the software to update SRS11.CGS.	1'h0
9-0	R	**SDCFSPV## - SDCLK Clock Frequency Select - Preset Value** This field can be used by the software to update SRS11.SDCFSH and SRS11.SDCFSL.	10'h2

12.55 SRS27 (0x026c)

SRS27 - Preset Value (SDR104 and DDR50)

SRS27[15:0] - SDR104 if:

SRS15.V18SE=1

SRS15.UMS=011b

SRS27[31:16] - DDR50 if:

SRS15.V18SE=1

SRS15.UMS=100b

HWINIT Register - Note this register is hardware initialized after reset and the value read back will match the IP configuration.

Table 87. SRS27 Register Fields

Bit	Type	Description	Reset
31-30	R	**DSSPV## - Driver Strength Select - Preset Value** This field can be used by the software to update SRS15.DSS.	2'h0
25-16	R	**SDCFSPV## - SDCLK Clock Frequency Select - Preset Value** This field can be used by the software to update SRS11.SDCFSH and SRS11.SDCFSL.	10'h2

Table 87. SRS27 Register Fields

Bit	Type	Description	Reset
15-14	R	**DSSPV## - Driver Strength Select - Preset Value** This field can be used by the software to update SRS15.DSS.	2'h0
9-0	R	**SDCFSPV## - SDCLK Clock Frequency Select - Preset Value** This field can be used by the software to update SRS11.SDCFSH and SRS11.SDCFSL.	10'h0

12.56 SRS30 (0x0278)

SRS30 ADMA3 ID Address 1

Table 88. SRS30 Register Fields

Bit	Type	Description	Reset
31-0	RW	**ADMA3 Integrated Descriptor Address #1** This field contains the physical address of the currently processed ADMA3 Integrated Descriptor address. The Host Driver will set this register with the ID table base address before it starts the ADMA3 transfers. The Host Driver should not write this register while the data transfer is active. While the ADMA3 engine is processing the descriptors list, the ADMA3ID value is always incremented to point the next ID to be fetched. The host ADMA3 engine ignores 2 or 3 least significant bits in this register when the 32-bit or 64-bit addressing is active, respectively. When ADMA3 uses 32-bit addressing mode, write to this register starts ADMA3.	32'h0

12.57 SRS31 (0x027c)

SRS30 ADMA3 ID Address 2

Table 89. SRS31 Register Fields

Bit	Type	Description	Reset
31-0	RW	**ADMA3 Integrated Descriptor Address #2** If 64-bit addressing is enabled (SRS15.A64B=1), this field holds bits 63-32 of the physical address pointing on ADMA3 Integrated Descriptor table. When ADMA3 uses 64-bit addressing mode, write to this register starts ADMA3.	32'h0

12.58 CRS63 (0x02fc)

CRS63 - Host Controller Version/Slot Interrupt Status

Table 90. CRS63 Register Fields

Bit	Type	Description	Reset
23-16	R	**SVN - Specification Version Number.** This field identifies the Host Controller Specification Version. • 0h - Version 1.00 • 1h - Version 2.00 • 2h - Version 3.00 • 3h - Version 4.00 • 4h - Version 4.10 • 5h - Version 4.20 • 6h-FFh - reserved	8'h5
0	R	**ISES - Interrupt Signal For Each Slot.** This field informs whether Interrupt or Wake-Up signal is active.	1'h0

12.59 CQRS00 (0x0400)

CQRS00 - Command Queuing Version

Table 91. CQRS00 Register Fields

Bit	Type	Description	Reset
11-8	R	**CQVN1 - eMMC Major Version Number** Major version number of supported eMMC standard - 5.	4'h5
7-4	R	**CQVN2 - eMMC Minor Version Number** Minor version number of supported eMMC standard - 1.	4'h1
3-0	R	**CQVN3 - eMMC Version Suffix** Suffix version number of supported eMMC standard - 0.	4'h0

12.60 CQRS01 (0x0404)

CQRS01 - Command Queuing Capabilities

Table 92. CQRS01 Register Fields

Bit	Type	Description	Reset
15-12	R	**ITCFMUL - Internal Timer Clock Frequency Multiplier (ITCFMUL)** Defines multiplier of internal clock frequency for the coalescing timer and for the SQS polling period. 0 - 0.001 MHz 1 - 0.01 MHz 2 - 0.1 MHz 3 - 1 MHz 4 - 10 MHz The ITCFMUL and ITCFVAL defines the clock frequency.	4'h0

Table 92. CQRS01 Register Fields

Bit	Type	Description	Reset
9-0	R	**ITCFVAL - Internal Timer Clock Frequency Value (ITCFVAL)** Value defines internal clock frequency for the coalescing timer and for the SQS polling period. The frequency is equal to ITCFMUL * ITCFVAL.	10'h0

12.61 CQRS02 (0x0408)

CQRS02 - Command Queuing Configuration

Table 93. CQRS02 Register Fields

Bit	Type	Description	Reset
12	RW	**CQDCE - Direct Command (DCMD) Enable** Process Task Descriptor for slot 31 as Data Transfer Task Descriptor (0) or Direct Command Task Descriptor (1).	1'h0
8	RW	**CQTDS - Task Descriptor** Size Expect 128 bit (1) or 64 bit (0) task descriptor. This setting can be changed only when Command Queuing is disabled (CQE = 0).	1'h0
0	RW	**CQE - Command Queuing Enable** Enables (1) or disables (0) the Command Queuing. This bit can be enabled only when all previous transactions are completed. This bit can be cleared only when all tasks are completed or cleared. Setting this bit to 1, set SRS15.HV4E automatically to 1.	1'h0

12.62 CQRS03 (0x040c)

CQRS03 - Command Queuing Control

Table 94. CQRS03 Register Fields

Bit	Type	Description	Reset
8	W1S	**CQCAT - Clear All Tasks** Clears (1) all active tasks in the host controller. Software has to poll this register until operation is completed (bit is automatically cleared). Software can set this bit only when the CQ Engine is halted. Software has to clear all requested tasks in the eMMC device. Writing (0) has no effect.	1'h0
0	RW	**CQHLT - Halt CQ** Engine can be halted by writing this bit 1. Any pending operation will be completed, and awaiting operation will be stopped. Once all tasks are completed or stopped this bit is set 1. The host controller will not automatically start any new operation, but software can use SRS registers to issue any command directly bypassing CQE. CQ Engine starts operation after being halted by writing 0 to this register. Writing 0 is ignored when CQ Engine is not halted.	1'h0

12.63 CQRS04 (0x0410)

CQRS04 - Command Queuing Interrupt Status

This register has several status bit related to specific interrupt event. When even happened and related Command Queuing Interrupt Status Enable is set, the status bit is set to 1. The bits can be cleared by S/W.

Write 0 clears bit.

Write 1 is ignored.

Table 95. CQRS04 Register Fields

Bit	Type	Description	Reset
3	W1C	**CQTCL - Task Cleared (TCL)** When task clear operation or clear individual task is completed, the CQE sets this bit to 1.	1'h0
2	W1C	**CQREDI - Response Error Detected Interrupt (RED)** When an error is detected in the response received from eMMC device, the CQE sets this bit to 1. S/W can select which bits are analyzed by selecting CQRMEM.	1'h0
1	W1C	**CQTCC - Task Complete Interrupt (TCC)** CQE sets this bit when either a task with INT=1 is completed or Interrupt Coalescing reports interrupt.	1'h0
0	W1C	**CQHAC - Halt Complete Interrupt (HAC)** CQE sets this bit when value of CQHLT changed from 0 to 1.	1'h0

12.64 CQRS05 (0x0414)

CQRS05 - Command Queuing Interrupt Status Enable

Statuses Enable bits enables interrupt sources. The status is enabled when bit is set 1 (S/W wrote 1 to the field).

Table 96. CQRS05 Register Fields

Bit	Type	Description	Reset
3	RW	**CQTCLST - Task Cleared Status Enable (TCL)** Enables CQTCL register.	1'h0
2	RW	**CQREDST - Response Error Detected Status Enable (RED)** Enables CQREDI register.	1'h0
1	RW	**CQTCCST - Task Complete Status Enable (TCC)** Enables CQHAC register.	1'h0
0	RW	**CQHACST - Halt Complete Status Enable (HAC)** Enables CQTCLST register.	1'h0

12.65 CQRS06 (0x0418)

CQRS06 - Command Queuing Interrupt Signal Enable

This register allows to turn on or turn off interrupt notification separately for each bit of the Command Queuing Interrupt Status. When Interrupt status bit is set 1 and related field in this register is set (S/W wrote 1 to the filed), the Interrupt Status is reported on interrupt port.

Table 97. CQRS06 Register Fields

Bit	Type	Description	Reset
3	RW	**CQTCLSI - Task Cleared Signal Enable (TCL)** Enables interrupt signaling from CQTCL register.	1'h0
2	RW	**CQREDSI - Response Error Detected Signal Enable (TCC)** Enables interrupt signaling from CQREDI register.	1'h0
1	RW	**CQTCCSI - Task Complete Signal Enable (TCC)** Enables interrupt signaling from CQTCC register.	1'h0
0	RW	**CQHACSI - Halt Complete Signal Enable (HAC)** Enables interrupt signaling from CQHLT register.	1'h0

12.66 CQRS07 (0x041c)

CQRS07 - Interrupt Coalescing

This register allows to group a CQ transfer and report single interrupt for entire group of requested tasks.

Table 98. CQRS07 Register Fields

Bit	Type	Description	Reset
31	RW	**CQICED - Interrupt Coalescing Enable/Disable** Enables coalescing mechanism allowing to generate coalescing interrupts.	1'h0
20	R	**CQICSB - Interrupt Coalescing Status Bit (ICSB)** CQE sets this bit 1 when any task with INT=0 is completed.	1'h0
16	W	**CQICCTR - Counter and Timer Reset(ICCTR)** S/W resets interrupt coalescing timer and counter.	1'h0
15	W	**CQICCTHWEN - Interrupt Coalescing Counter Threshold Write Enable (ICCTHWEN)** This is write enable for CQICCTH. When this bit is set 1, the field will be updated.	1'h0
12-8	RW	**CQICCTH - Interrupt Coalescing Counter Threshold (ICCTH)** CQE increments internal counter when task with INT=0 is completed. When internal counter reaches this value the coalescing generates interrupt. S/W can select treshold value in range 1 to 31. S/W can disable internal counter and interrupt generation by setting this field to 0.	5'h0
7	W	**CQICTOVALEN - Interrupt Coalescing Timeout Value Write Enable (ICTOVALWEN)** This is write enable for CQICTOVAL. When this bit is set 1, the field will be updated.	1'h0
6-0	RW	**CQICTOVAL - Interrupt Coalescing Timeout Value (ICTOVAL)** CQE generates interrupt when internal counter reaches period defined in this field. The counter starts when first transfer with INT=0 is completed. The counter increments each time when Internal Clock * 1024 period elapsed. S/W can disable this timer by setting this filed to 0.	7'h0

12.67 CQRS08 (0x0420)

CQRS08 - Command Queuing Task Descriptor List Base Address

Table 99. CQRS08 Register Fields

Bit	Type	Description	Reset
31-0	RW	**CQTDLBA - Task Descriptor List Base Address (lower)** Base address (32 lower bits) of the Task Descriptor List. S/W will write values aligned to 1kB boundary (lower 10 bits have to be 0). The hardware ignores 10 lower bits. S/W will update this register only when CQE is disabled.	32'h0

12.68 CQRS09 (0x0424)

CQRS09 - Command Queuing Task Descriptor List Base Address Upper 32 Bits

Table 100. CQRS09 Register Fields

Bit	Type	Description	Reset
31-0	RW	**CQTDLBAU - Task Descriptor List Base Address (upper)** Base address (32 upper bits) of the Task descriptor List. This register is not used in 32 bit addressing mode (S/W does not change this value). S/W will update this register only when CQE is disabled.	32'h0

12.69 CQRS10 (0x0428)

CQRS10 - Command Queuing Task Doorbell

CQ has 32 tasks have individual bits to start operation on desired task. The S/W writes 1 on any position from 0 to 31 to start task 0 to 31. The S/W can request more than one task in single write. The CQ Engine process tasks in order they were requested:

- when more than one task is requested in single register write, the task with lower number has higher priority over task with higher number

- task(s) requested in earlier register write has higher priority over task(s) in later register write

The order of the tasks are maintained during all phases of transaction. If given task is not ready for execution, the CQ Engine takes next task with highest number.

CQ Engine needs several clock cycles to push requested in the single register write Task Doorbell to queue. The slave interface ends write transfer as soon all tasks are in the queue.

When S/W writes 0 to bit in this register, the related task won't start - this value is ignored.

Task Doorbell bit remain 1 until task execution is completed, task is cleared by Clear All Task or Clear Task with this number or CQ Engine is disabled (CQE=0).

Table 101. CQRS10 Register Fields

Bit	Type	Description	Reset
31	W1S	**CQTD31 - Command Queuing Task Doorbell #31**	1'h0
30	W1S	**CQTD30 - Command Queuing Task Doorbell #30**	1'h0
29	W1S	**CQTD29 - Command Queuing Task Doorbell #29**	1'h0
28	W1S	**CQTD28 - Command Queuing Task Doorbell #28**	1'h0
27	W1S	**CQTD27 - Command Queuing Task Doorbell #27**	1'h0
26	W1S	**CQTD26 - Command Queuing Task Doorbell #26**	1'h0
25	W1S	**CQTD25 - Command Queuing Task Doorbell #25**	1'h0
24	W1S	**CQTD24 - Command Queuing Task Doorbell #24**	1'h0
23	W1S	**CQTD23 - Command Queuing Task Doorbell #23**	1'h0
22	W1S	**CQTD22 - Command Queuing Task Doorbell #22**	1'h0
21	W1S	**CQTD21 - Command Queuing Task Doorbell #21**	1'h0
20	W1S	**CQTD20 - Command Queuing Task Doorbell #20**	1'h0
19	W1S	**CQTD19 - Command Queuing Task Doorbell #19**	1'h0
18	W1S	**CQTD18 - Command Queuing Task Doorbell #18**	1'h0
17	W1S	**CQTD17 - Command Queuing Task Doorbell #17**	1'h0
16	W1S	**CQTD16 - Command Queuing Task Doorbell #16**	1'h0
15	W1S	**CQTD15 - Command Queuing Task Doorbell #15**	1'h0
14	W1S	**CQTD14 - Command Queuing Task Doorbell #14**	1'h0
13	W1S	**CQTD13 - Command Queuing Task Doorbell #13**	1'h0
12	W1S	**CQTD12 - Command Queuing Task Doorbell #12**	1'h0
11	W1S	**CQTD11 - Command Queuing Task Doorbell #11**	1'h0
10	W1S	**CQTD10 - Command Queuing Task Doorbell #10**	1'h0
9	W1S	**CQTD09 - Command Queuing Task Doorbell #09**	1'h0
8	W1S	**CQTD08 - Command Queuing Task Doorbell #08**	1'h0
7	W1S	**CQTD07 - Command Queuing Task Doorbell #07**	1'h0

Table 101. CQRS10 Register Fields

Bit	Type	Description	Reset
6	W1S	**CQTD06 - Command Queuing Task Doorbell #06**	1'h0
5	W1S	**CQTD05 - Command Queuing Task Doorbell #05**	1'h0
4	W1S	**CQTD04 - Command Queuing Task Doorbell #04**	1'h0
3	W1S	**CQTD03 - Command Queuing Task Doorbell #03**	1'h0
2	W1S	**CQTD02 - Command Queuing Task Doorbell #02**	1'h0
1	W1S	**CQTD01 - Command Queuing Task Doorbell #01**	1'h0
0	W1S	**CQTD00 - Command Queuing Task Doorbell #00**	1'h0

12.70 CQRS11 (0x042c)

CQRS11 - Task Complete Notification

32 bits related to 32 tasks. If task N is completed N bit is set 1. Bit that is set 1 can be cleared by writing 1 to this bit.

Table 102. CQRS11 Register Fields

Bit	Type	Description	Reset
31	W1C	**CQTCN31 - Task Completion Notification #31**	1'h0
30	W1C	**CQTCN30 - Task Completion Notification #30**	1'h0
29	W1C	**CQTCN29 - Task Completion Notification #29**	1'h0
28	W1C	**CQTCN28 - Task Completion Notification #28**	1'h0
27	W1C	**CQTCN27 - Task Completion Notification #27**	1'h0
26	W1C	**CQTCN26 - Task Completion Notification #26**	1'h0
25	W1C	**CQTCN25 - Task Completion Notification #25**	1'h0
24	W1C	**CQTCN24 - Task Completion Notification #24**	1'h0
23	W1C	**CQTCN23 - Task Completion Notification #23**	1'h0
22	W1C	**CQTCN22 - Task Completion Notification #22**	1'h0
21	W1C	**CQTCN21 - Task Completion Notification #21**	1'h0
20	W1C	**CQTCN20 - Task Completion Notification #20**	1'h0

Table 102. CQRS11 Register Fields

Bit	Type	Description	Reset
19	W1C	**CQTCN19 - Task Completion Notification #19**	1'h0
18	W1C	**CQTCN18 - Task Completion Notification #18**	1'h0
17	W1C	**CQTCN17 - Task Completion Notification #17**	1'h0
16	W1C	**CQTCN16 - Task Completion Notification #16**	1'h0
15	W1C	**CQTCN15 - Task Completion Notification #15**	1'h0
14	W1C	**CQTCN14 - Task Completion Notification #14**	1'h0
13	W1C	**CQTCN13 - Task Completion Notification #13**	1'h0
12	W1C	**CQTCN12 - Task Completion Notification #12**	1'h0
11	W1C	**CQTCN11 - Task Completion Notification #11**	1'h0
10	W1C	**CQTCN10 - Task Completion Notification #10**	1'h0
9	W1C	**CQTCN09 - Task Completion Notification #09**	1'h0
8	W1C	**CQTCN08 - Task Completion Notification #08**	1'h0
7	W1C	**CQTCN07 - Task Completion Notification #07**	1'h0
6	W1C	**CQTCN06 - Task Completion Notification #06**	1'h0
5	W1C	**CQTCN05 - Task Completion Notification #05**	1'h0
4	W1C	**CQTCN04 - Task Completion Notification #04**	1'h0
3	W1C	**CQTCN03 - Task Completion Notification #03**	1'h0
2	W1C	**CQTCN02 - Task Completion Notification #02**	1'h0
1	W1C	**CQTCN01 - Task Completion Notification #01**	1'h0
0	W1C	**CQTCN00 - Task Completion Notification #00**	1'h0

12.71 CQRS12 (0x0430)

CQRS12 - Device Queue Status

Table 103. CQRS12 Register Fields

Bit	Type	Description	Reset
31-0	R	**CQDQS - Device Queue Status** This register reflects to eMMC device status. Task N is ready for execution when bit N in this register is set to 1. This register is updated each time response for SEND_QUEUE_STATUS (CMD13) is received.	32'h0

12.72 CQRS13 (0x0434)

CQRS13 - Device Pending Tasks

This register information which task is submitted to eMMC (CMD44 and CMD45 was sent) and is not executed. Task N is submitted and not executed when N bit is set 1. Bit N is cleared when task N is completed.

Table 104. CQRS13 Register Fields

Bit	Type	Description	Reset
31	R	**CQDPT31 - Device Pending Tasks #31**	1'h0
30	R	**CQDPT30 - Device Pending Tasks #30**	1'h0
29	R	**CQDPT29 - Device Pending Tasks #29**	1'h0
28	R	**CQDPT28 - Device Pending Tasks #28**	1'h0
27	R	**CQDPT27 - Device Pending Tasks #27**	1'h0
26	R	**CQDPT26 - Device Pending Tasks #26**	1'h0
25	R	**CQDPT25 - Device Pending Tasks #25**	1'h0
24	R	**CQDPT24 - Device Pending Tasks #24**	1'h0
23	R	**CQDPT23 - Device Pending Tasks #23**	1'h0
22	R	**CQDPT22 - Device Pending Tasks #22**	1'h0
21	R	**CQDPT21 - Device Pending Tasks #21**	1'h0
20	R	**CQDPT20 - Device Pending Tasks #20**	1'h0
19	R	**CQDPT19 - Device Pending Tasks #19**	1'h0
18	R	**CQDPT18 - Device Pending Tasks #18**	1'h0
17	R	**CQDPT17 - Device Pending Tasks #17**	1'h0

Table 104. CQRS13 Register Fields

Bit	Type	Description	Reset
16	R	**CQDPT16 - Device Pending Tasks #16**	1'h0
15	R	**CQDPT15 - Device Pending Tasks #15**	1'h0
14	R	**CQDPT14 - Device Pending Tasks #14**	1'h0
13	R	**CQDPT13 - Device Pending Tasks #13**	1'h0
12	R	**CQDPT12 - Device Pending Tasks #12**	1'h0
11	R	**CQDPT11 - Device Pending Tasks #11**	1'h0
10	R	**CQDPT10 - Device Pending Tasks #10**	1'h0
9	R	**CQDPT09 - Device Pending Tasks #09**	1'h0
8	R	**CQDPT08 - Device Pending Tasks #08**	1'h0
7	R	**CQDPT07 - Device Pending Tasks #07**	1'h0
6	R	**CQDPT06 - Device Pending Tasks #06**	1'h0
5	R	**CQDPT05 - Device Pending Tasks #05**	1'h0
4	R	**CQDPT04 - Device Pending Tasks #04**	1'h0
3	R	**CQDPT03 - Device Pending Tasks #03**	1'h0
2	R	**CQDPT02 - Device Pending Tasks #02**	1'h0
1	R	**CQDPT01 - Device Pending Tasks #01**	1'h0
0	R	**CQDPT00 - Device Pending Tasks #00**	1'h0

12.73 CQRS14 (0x0438)

CQRS14 - Task Clear

S/W writes 1 to N bit of this register to clear task N. Bit remains 1 until clear operation is completed. Once operations ends, the CQE clears this bit to 0. The S/W has to ensure the CQ Engine is halted before clearing tasks. The S/W can clear only single task. When any bit of this register is set, the S/W has no to request new task clear. This operation clears only task in the Host Controller. The S/W should take care about clearing task in the device.

Writing 0 to register is ignored.

Table 105. CQRS14 Register Fields

Bit	Type	Description	Reset
31	W1S	**CQTC31 - Command Queuing Task Clear #31**	1'h0
30	W1S	**CQTC30 - Command Queuing Task Clear #30**	1'h0
29	W1S	**CQTC29 - Command Queuing Task Clear #29**	1'h0
28	W1S	**CQTC28 - Command Queuing Task Clear #28**	1'h0
27	W1S	**CQTC27 - Command Queuing Task Clear #27**	1'h0
26	W1S	**CQTC26 - Command Queuing Task Clear #26**	1'h0
25	W1S	**CQTC25 - Command Queuing Task Clear #25**	1'h0
24	W1S	**CQTC24 - Command Queuing Task Clear #24**	1'h0
23	W1S	**CQTC23 - Command Queuing Task Clear #23**	1'h0
22	W1S	**CQTC22 - Command Queuing Task Clear #22**	1'h0
21	W1S	**CQTC21 - Command Queuing Task Clear #21**	1'h0
20	W1S	**CQTC20 - Command Queuing Task Clear #20**	1'h0
19	W1S	**CQTC19 - Command Queuing Task Clear #19**	1'h0
18	W1S	**CQTC18 - Command Queuing Task Clear #18**	1'h0
17	W1S	**CQTC17 - Command Queuing Task Clear #17**	1'h0
16	W1S	**CQTC16 - Command Queuing Task Clear #16**	1'h0
15	W1S	**CQTC15 - Command Queuing Task Clear #15**	1'h0
14	W1S	**CQTC14 - Command Queuing Task Clear #14**	1'h0
13	W1S	**CQTC13 - Command Queuing Task Clear #13**	1'h0
12	W1S	**CQTC12 - Command Queuing Task Clear #12**	1'h0
11	W1S	**CQTC11 - Command Queuing Task Clear #11**	1'h0

Table 105. CQRS14 Register Fields

Bit	Type	Description	Reset
10	W1S	**CQTC10 - Command Queuing Task Clear #10**	1'h0
9	W1S	**CQTC09 - Command Queuing Task Clear #09**	1'h0
8	W1S	**CQTC08 - Command Queuing Task Clear #08**	1'h0
7	W1S	**CQTC07 - Command Queuing Task Clear #07**	1'h0
6	W1S	**CQTC06 - Command Queuing Task Clear #06**	1'h0
5	W1S	**CQTC05 - Command Queuing Task Clear #05**	1'h0
4	W1S	**CQTC04 - Command Queuing Task Clear #04**	1'h0
3	W1S	**CQTC03 - Command Queuing Task Clear #03**	1'h0
2	W1S	**CQTC02 - Command Queuing Task Clear #02**	1'h0
1	W1S	**CQTC01 - Command Queuing Task Clear #01**	1'h0
0	W1S	**CQTC00 - Command Queuing Task Clear #00**	1'h0

12.74 CQRS16 (0x0440)

CQRS16 - Send Status Configuration 1

Table 106. CQRS16 Register Fields

Bit	Type	Description	Reset
19-16	RW	**CQSSCBC - Send Status Command Block Counter (CBC)** S/W can define if and when CQE sends SEND_QUEUE_STATUS (CMD13) command during data transfer. When this register is set 0, the CQE does not send CMD13 during data transfer. The value is 1, 2, or N means, the CQE sends CMD13 is transferred during last, one before last, or (N-1) before last block, respectively. Accepted register value range is 0 to 15.	4'h0
15-0	RW	**CQSSCIT - Send Status Command Idle Timer (CIT)** When CQE is in idle, the host controller can poll device by sending SEND_QUEUE_STATUS (CMD13) with interval defined by this register. Accepted register value is in range 1 to 65535. The interval can be calculated as CQSSICT * internal clock period.	16'h0

12.75 CQRS17 (0x0444)

CQRS17 - Send Status Configuration 2

Table 107. CQRS17 Register Fields

Bit	Type	Description	Reset
15-0	RW	**CQSQSR - Send Queue Status RCA** S/W writes 16-bit RCA value which is send as an argument in SEND_QUEUE_STATUS (CMD13) command.	16'h0

12.76 CQRS18 (0x0448)

CQRS18 - Command Response for Direct-Command Task

Table 108. CQRS18 Register Fields

Bit	Type	Description	Reset
31-0	R	**CQDCLR - Direct Command Last Response** CQE holds the last DCMD command response.	32'h0

12.77 CQRS20 (0x0450)

CQRS20 - Response Mode Error Mask

Table 109. CQRS20 Register Fields

Bit	Type	Description	Reset
31-0	RW	**CQRMEM - Response Mode Error Mask** CQE is able to automatically detect errors in response. The S/W defines which bits of the response need to be checked. All bits set to 1 (written by S/W) are analyzed. The CQE reports Response Error Detected Interrupt (CQREDI) when N bit of CQRMEM is 1 and N bit of response is 1. Response for SEND_QUEUE_STATUS (CMD13) automatically sent by CQE is ignored.	32'h0

12.78 CQRS21 (0x0454)

CQRS21 - Task Error Information

Table 110. CQRS21 Register Fields

Bit	Type	Description	Reset
31	R	**CQDTEFV - Data Transfer Error Fields Valid** Host sets this bit to 1 when error is detected during data transfer. Host clears this bit to 0 when error is detected and there is no active data transfer.	1'h0
28-24	R	**CQDTETID - Data Transfer Error Task ID** Host updates this field with ID of the task with active data transfer when error occurred during the data transfer.	5'h0
21-16	R	**CQDTECI - Data Transfer Error Command Index** Host updates this field with index of the data transfer command executed when error occurred during the data transfer.	6'h0
15	R	**CQRMEFV - Response Mode Error Fields Valid** Host sets this bit to 1 when error is detected and command transaction is active. Host sets this bit to 0 when error is detected and command transaction is no active.	1'h0
12-8	R	**CQRMETID - Response Mode Error Task ID** Host updates this field with ID of the task with active command transfer when error occurred during the command transaction.	5'h0
5-0	R	**CQRMECI - Response Mode Error Command Index** Host updates this field with index of the command executed when error occurred during the command transaction.	6'h0

12.79 CQRS22 (0x0458)

CQRS22 - Command Response Index

Table 111. CQRS22 Register Fields

Bit	Type	Description	Reset
5-0	R	**CQLCRI - Last Command Response Index** Host updates this field with command index when response is received.	6'h0

12.80 CQRS23 (0x045c)

CQRS23 - Command Response Argument

Table 112. CQRS23 Register Fields

Bit	Type	Description	Reset
31-0	R	**CQLCRA - Last Command Response Argument** Host updates this field with command argument when response is received.	32'h0

Change history

Table 113. Change history

Revision	Date	Description
1.0	2019-06-28	Released IP6061
1.1	2020-01-15	Remove hwinit_* ports not available in IP6061 configuration Correct reference to SRS15.A64B Correct enumeration of RESP registers
1.2	2020-02-06	Correct wstrb width
1.3	2020-02-12	Correct master interface on the block diagram Add value of constant AXI output ports
1.31	2021-11-19	Updated Xcelium version
1.6	2022-04-14	Corrected source clock of Slave/Register APB Interface in table 6
Revision	**Date**	**Description**