14. TF

14.1

Multimedia Card MMC 1997 NAND Intel I
bits MMC SD MMC SD

Secure Digital SD Secure Digital Memory
Card SD MMC MMC

Micro SD TF T-
Flash TransFlash 2004 TransFlash T
SD 15 x 11 x 1 - SD 1/4 SD Memory Stick

eMMC (Embedded Multi Media Card MMC eMMC

SD SDIO GPS Wi-Fi FM

14.2

JEDEC JESD84-B451 JESD84-B51 JEDEC

SD Card Association

Physical_Layer_Simplified_Specification_Ver6.00

SD Host_Controller_Simplified_Specification_Ver3.00

SDIO SDIO_Simplified_Specification_Ver3.00

SD Card Association

Chapter 58 : Ultra Secured Digital Host Controller (uSDHC)

14.2.1

14.2.1.1 SD

-
-

1 SDSC Standard Capacity SD Memory Card 2GB

2 SDHC High Capacity SD Memory Card 2GB 32GB

3 SDXC, Extended Capacity SD Memory Card 32GB 2TB

-

1 SD 2.7-3.6V

2 UHS-II VDD1 2.7-3.6V VDD2 1.70V-1.95V

-
- 4
-
-
-
- SDMI
- CMD42-LOCK_UNLOCK
-
-
-
-
-
- 4

14.2.1.2 eMMC

eMMC e

-

image-20220112141205032

Image not found or type unknown

- eMMC4.5 1bit 8bit eMMC5.1

1 0~200MHz

2 1bit 4bit 8bit

-
-
- Erase Trim Sanitize
-
-
-
- HPI Packed Commands
- boot
- RPMB
- 512B 4KB

14.2.2

14.2.2.1 SD

Bus speed mode for UHS-I Card

image-20220112141322603

Image not found or type unknown

Default Speed Mode 3.3V signaling, Frequency up to 25MHz, up to 12.5MB/sec

High Speed Mode 3.3V signaling, Frequency up to 50MHz, up to 25MB/sec

SDR12 UHS-I 1.8V signaling, Frequency up to 25MHz, up to 12.5MB/sec

SDR25 UHS-I 1.8V signaling, Frequency up to 50MHz, up to 25MB/sec

SDR50 UHS-I 1.8V signaling, Frequency up to 100MHz, up to 50MB/sec

SDR104 UHS-I 1.8V signaling, Frequency up to 208MHz, up to 104MB/sec

DDR50 UHS-I 1.8V signaling, Frequency up to 50MHz, up to 50MB/sec

UHS156 UHS-II RCLK Frequency Range 26MHz~52MHz, up to 1.56Gbps per lane

UHS624 UHS-II RCLK Frequency Range 26MHz~52MHz, up to 6.24Gbps per lane

SD 3.3v SD default speed High speed

14.2.2.2 eMMC

image-20220112141353206

Image not found or type unknown

eMMC 3.3v default speed high speed

14.2.3 SD eMMC

14.2.3.1 sd

image-20220112141429981

Image not found or type unknown

SD VDD GND CLK CMD DAT0~DAT3 9

14.2.3.2 sd

CID	128	Card identification number	
RCA	16	Relative Card Address	
DSR	16	Driver Stage Register	
CSD	128	Card Specific Data	
SCR	64	SD Configuration Register SD	
OCR	32	Operation Condition Register	
SSR	512	SD status SD	
CSR	32	Card status Card	

14.2.3.3 MMC

image-20220112141532242

Image not found or type unknown

eMMC Nand flash Device Controller Device Controller flash

14.2.3.4 eMMC

image-20220112141550937

Image not found or type unknown

CID	16	Device Identification number	
RCA	2	Relative Device Address	
DSR	2	Driver Stage Register	
CSD	16	Device Specific Data	
OCR	4	Operations Conditions Register	
EXT_CSD	512	Extended Device Specific Data	v4.0

14.2.4 COMMAND RESPONSE

14.2.4.1

image-20220112141618129

Image not found or type unknown

“ 0” “ 1” 48 CRC

SD eMMC 4

- bc Broadcast commands
- bcr Broadcast commands
- ac addressed point-to-point commands DATA
- adtc addressed point-to-point data transfer commands DATA

image-20220112141644353

Image not found or type unknown

CMD command CMD command response CMD

image-20220112141707356

Image not found or type unknown

adtc RCA CMD command response DAT

image-20220112141712729

Image not found or type unknown

adtc

14.2.4.2

CMD

SD eMMC

- R1 normal response command 48bit
- R1b R1 DATA0
- R2 CID CSD register 136bit CID CMD2 CMD10 CSD CMD
- R3 OCR register 48bit OCR CMD1

image-20220112141745166

Image not found or type unknown

SD SD

- R6 published RCA response 48bit
- R7 Card interface condition 48bit

image-20220112141756135

Image not found or type unknown

eMMC eMMC

- R4 fast IO 48bit
- R5 Interrupt request 48bit

14.3 IMX6ULL uSDHC

14.3.1 uSDHC

- SD Host Controller Standard Specification version 3.0
- MMC System Specification version 4.2/4.3/4.4/4.41/4.5
- SD Memory Card Specification version 3.0 Extended Capacity SD Memory Card
- SDIO Card Specification version 3.0

- SD Memory, miniSD Memory, SDIO, miniSDIO, SD Combo, MMC, MMC plus, and MMC RS cards
- 208MHz
- 1-bit/4-bit SD SDIO 1-bti/4-bit SD/8-bit MMC

1 4 SDR(Single Data Rate) SDIO 832Mbps

2 4 DDR(Dual Data Rate) SDIO 400Mbps

3 4 SDR(Single Data Rate) SDXC 832Mbps

4 4 DDR(Dual Data Rate) SDXC 400Mbps

5 8 SDR(Single Data Rate) MMC 416Mbps

6 8 DDR(Dual Data Rate) MMC 832Mbps

-
- Wie 1~4096
-
-
-
- SDIO (suspend) (resume)
- CMD12
-
- 1-bit 4-bit SDIO
- 128x32-bit FIFO
- DMA
- vendor
- ADMA

14.3.2

uSDHC

- SD 1-bit
- SD 4-bit
- MMC 1-bit
- MMC 4-bit
- MMC 8-bit
- 400 kHZ
- MMC 26MHz
- MMC 52MHz
- MMC HS200 200MHz
- MMC DDR 52M

- SD/SDIO 25MHz
- SD/SDIO 50MHz
- SD/SDIO UHS-I SDR 208M DDR 50M

14.3.3

image-20220112142021189

Image not found or type unknown

100sdk_imx6ull SD uSDHC1

image-20220112142025607

Image not found or type unknown

100sdk_imx6ull eMMC uSDHC2

uSDHC 14 I/O

- CLK MMC SD SDIO
- CMD I/O
- 8 uSDHC
- CD WP CD# WP
- LCTL LED
- RST MMC
- VSELECT
- CD, WP, LCTL, RST VSELECT uSDHC 4bit DATA7~DATA4

14.3.4

Clock name	Clock Root	Description
hclk	ahb_clk_root	AHB bus clock
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_perclk	usdhc_clk_root	Base clock
ipg_clk_s	ipg_clk_root	Peripheral access clock for register accesses

uSDHC hclk AHB ipg_clk Peripheral ipg_clk_perclk uSDHC ipg_clk_s

image-20220112142122012

Image not found or type unknown

imx6ull reference manual chapter18 Clock Controller Module 18.3 CCM Clock Tree CCM_CSCMR1[USDHCn_CLK_SEL] CSCDRn[USDHC1_PODF] USDHCn_CLK_RC

CCM_CSCMR1

image-20220112142130658

Image not found or type unknown

CSCDR1[USDHC1_PODF]

image-20220112142135423

Image not found or type unknown

USDHC1 USDHC2 PLL2PFD0 PLL2PFD2 CCM_CSCMR1 CSCDR1[USDHCn_POD D0 396M USDHCn_PODF 1 PLL2PFD2 USDHC1_CLK_ROOT USDHC1_CLK_ROOT

Card_clk peripheral source clock

image-20220112142205025

Image not found or type unknown

uSDHCx_SYS_CTRL[DVS] DIV Base ase/2, Base/3, ..., Base/16

uSDHCx_SYS_CTRL[SDCLKFS] card_clk

SDR DDR CLK SDR CLK card_clk DDR CLK card_clk/2

Base USDHCx_CLK_ROOT USDHCx_CLK_ROOT 198M uSDHCx_SYS_CTRL[DVS] uSDHCx_SY KFS] 400K 25M 26M 50M 52M

14.3.5 ADMA

SD DMA ADMA Advanced DMA advanced SDMA(Si

SDMA page boundary DMA DMA System Address (uSDHCx_DS_ADDR) System addreass

ADMA DMA scatter gather algorithm ADMA desc (uSDHCx_ADMA_SYS_ADDR) DMA CPU

ADMA ADMA1 ADMA2 ADMA1 4KB ADMA2

image-20220112142231919

Image not found or type unknown

PartA2_SD Host_Controller_Simplified_Specification_Ver3.00 page14

ADMA2 address length attribute

ADMA2 ADMA2

3

1 4

2 64KB

3 Length1+length2+…+lengthn

Block count 65535 ADMA2 65535 block count block size b

image-20220112142239359

Image not found or type unknown

image-20220112142245267

Image not found or type unknown

32-bit 64 Nop Rsv NOP Tran 32-b

14.3.6

SD Host_Controller_Simplified_Specification_Ver3.00 SD

image-20220112142257497

Image not found or type unknown

IMX6ULL

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
219_0000	DMA System Address (uSDHC1_DS_ADDR)	32	R/W	0000_0000h	58.8.1/4014
219_0004	Block Attributes (uSDHC1_BLK_ATT)	32	R/W	0000_0000h	58.8.2/4015
219_0008	Command Argument (uSDHC1_CMD_ARG)	32	R/W	0000_0000h	58.8.3/4017
219_000C	Command Transfer Type (uSDHC1_CMD_XFR_TYP)	32	R/W	0000_0000h	58.8.4/4017
219_0010	Command Response0 (uSDHC1_CMD_RSP0)	32	R	0000_0000h	58.8.5/4021
219_0014	Command Response1 (uSDHC1_CMD_RSP1)	32	R	0000_0000h	58.8.6/4021
219_0018	Command Response2 (uSDHC1_CMD_RSP2)	32	R	0000_0000h	58.8.7/4022
219_001C	Command Response3 (uSDHC1_CMD_RSP3)	32	R	0000_0000h	58.8.8/4022
219_0020	Data Buffer Access Port (uSDHC1_DATA_BUFF_ACC_PORT)	32	R/W	0000_0000h	58.8.9/4024
219_0024	Present State (uSDHC1_PRES_STATE)	32	R	0000_8080h	58.8.10/ 4024
219_0028	Protocol Control (uSDHC1_PROT_CTRL)	32	R/W	0880_0020h	58.8.11/ 4030
219_002C	System Control (uSDHC1_SYS_CTRL)	32	R/W	8080_800Fh	58.8.12/ 4035

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
219_0030	Interrupt Status (uSDHC1_INT_STATUS)	32	w1c	0000_0000h	58.8.13/ 4038
219_0034	Interrupt Status Enable (uSDHC1_INT_STATUS_EN)	32	R/W	0000_0000h	58.8.14/ 4044
219_0038	Interrupt Signal Enable (uSDHC1_INT_SIGNAL_EN)	32	R/W	0000_0000h	58.8.15/ 4047
219_003C	Auto CMD12 Error Status (uSDHC1_AUTOCMD12_ERR_STATUS)	32	R	0000_0000h	58.8.16/ 4050
219_0040	Host Controller Capabilities (uSDHC1_HOST_CTRL_CAP)	32	R	07F3_B407h	58.8.17/ 4053
219_0044	Watermark Level (uSDHC1_WTMK_LVL)	32	R/W	0810_0810h	58.8.18/ 4056
219_0048	Mixer Control (uSDHC1_MIX_CTRL)	32	R/W	8000_0000h	58.8.19/ 4057
219_0050	Force Event (uSDHC1_FORCE_EVENT)	32	W (always reads 0)	0000_0000h	58.8.20/ 4059
219_0054	ADMA Error Status Register (uSDHC1_ADMA_ERR_STATUS)	32	R	0000_0000h	58.8.21/ 4062
219_0058	ADMA System Address (uSDHC1_ADMA_SYS_ADDR)	32	R/W	0000_0000h	58.8.22/ 4064
219_0060	DLL (Delay Line) Control (uSDHC1_DLL_CTRL)	32	R/W	0000_0200h	58.8.23/ 4065
219_0064	DLL Status (uSDHC1_DLL_STATUS)	32	R	0000_0000h	58.8.24/ 4067

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
219_0068	CLK Tuning Control and Status (uSDHC1_CLK_TUNE_CTRL_STATUS)	32	R/W	0000_0000h	58.8.25/ 4068
219_00C0	Vendor Specific Register (uSDHC1_VEND_SPEC)	32	R/W	2000_7809h	58.8.26/ 4070
219_00C4	MMC Boot Register (uSDHC1_MMC_BOOT)	32	R/W	0000_0000h	58.8.27/ 4073
219_00C8	Vendor Specific 2 Register (uSDHC1_VEND_SPEC2)	32	R/W	0000_0006h	58.8.28/ 4074
219_00CC	Tuning Control Register (uSDHC1_TUNING_CTRL)	32	R/W	0021_2800h	58.8.29/ 4076

bit uSDHC SD

14.3.6.1 DMA System Address (uSDHCx_DS_ADDR)

SDMA ADMA2

image-20220112142313267

Image not found or type unknown

14.3.6.2 Block Attributes (uSDHCx_BLK_ATT)

image-20220112142340224

Image not found or type unknown

BLKCNT

BLKSIZE

14.3.6.3 Command Argument (uSDHCx_CMD_ARG)

SD/MMC

image-20220112142404487

Image not found or type unknown

14.3.6.4 Command Transfer Type (uSDHCx_CMD_XFR_TYP)

image-20220112142415095

Image not found or type unknown

Multi/Single Block Select	Block Count Enable	Block Count	Function
0	Don't Care	Don't Care	Single Transfer
1	0	Don't Care	Infinite Transfer
1	1	Positive Number	Multiple Transfer
1	1	Zero	No Data Transfer

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

CMDINDEX command index

CMDTYPE command type 0 normal command

DPSEL Data Present Select 1

CICEN command index check enable 1 response index

CCCEN command crc check enable response CRC

RSPTYP response type response 00 01 136 10 48 11 48

14.3.6.5 Command Response0 (uSDHCx_CMD_RSP0)

image-20220112142444186

Image not found or type unknown

14.3.6.6 Command Response1 (uSDHCx_CMD_RSP1)

image-20220112142451825

Image not found or type unknown

14.3.6.7 Command Response2 (uSDHCx_CMD_RSP2)

image-20220112142458607

Image not found or type unknown

14.3.6.8 Command Response3 (uSDHCx_CMD_RSP3)

image-20220112142504119

Image not found or type unknown

Response Type	Meaning of Response	Response Field	Response Register
R1,R1b (normal response)	Card Status	R[39:8]	CMDRSP0
R1b (Auto CMD12 response)	Card Status for Auto CMD12	R[39:8]	CMDRSP3
R2 (CID, CSD register)	CID/CSD register [127:8]	R[127:8]	{CMDRSP3[23:0], CMDRSP2, CMDRSP1, CMDRSP0}
R3 (OCR register)	OCR register for memory	R[39:8]	CMDRSP0
R4 (OCR register)	OCR register for I/O etc.	R[39:8]	CMDRSP0
R5, R5b	SDIO response	R[39:8]	CMDRSP0
R6 (Publish RCA)	New Published RCA[31:16] and card status[15:0]	R[39:9]	CMDRSP0

response reference manual P4022

14.3.6.9 Data Buffer Access Port(uSDHCx_DATA_BUFF_ACC_PORT)

DMA ADMA2 0

image-20220112142534276

Image not found or type unknown

14.3.6.10 Present State (uSDHCx_PRES_STATE)

image-20220112142541528

Image not found or type unknown

DLSL[7:0] data line signal level DATA DATA DATA0

CLSL CMD line Signal Level CMD CMD

WPSPL Write Protect Swith Pin Level WP

CDPL Card Detect Pin Level CD_B CD_B uSDHC

CINST Card Inserted uSDHC

TSCD Tape Select Change Done tuning

RTR Re-Tuning Request tuning SD3.0 SDR104

BREN Buffer READ ENABLE DMA

BWEN Buffer Write ENABLE DMA

RTA READ Transfer active

WTA Write Transfer Active

SDOFF SDCLOCK Gated Off Internally SD

PEROFF IPG_PERCLK Gated Off Internally IPG_PERCLK

HCKOFF HCLK Gated Off Internally HCLK

IPGOFF IPG_CLK Gated Off Internally IPG_CLK

SDSTB SD

DLA Data Line Active Data

CDIHB Command Inhabit(Data) data

CIHB Command Inhabit 0 CMD uSDHC SD/MMC

14.3.6.11 Protocol Control (uSDHCx_PROT_CTRL)

image-20220112142627014

Image not found or type unknown

NON_EXACT_BK_READ SDIO

BURST_ELN_EN

WECRM Wakeup Event Enable On SD Card Removal

WECINS Wakeup Event Enable On SD Card Insert

WECINT Wakeup Event Enable On Card Interrupt

RD_DONE_NO_8CLK

IABG interrupt At Block GAP SDIO 4bit

RWCLT READ wait control SDIO

CREQ continue request

DMASEL DMA select DMA 00 DMA SDMA 01 ADMA1, 10 ADMA2

CDSS Card Detect Signal Selection

CDTL Card Detect Test Level

EMODE Endian Mode

D3CD Data3 as Card Detection Pin data3

DTW Data Transfer Width

LCTL LED

14.3.6.12 System Control (uSDHCx_SYS_CTRL)

image-20220112142639489

Image not found or type unknown

RSTT Reset Tuning tuning

INITA Initialization Active 80 SDCLK

RSTD Software Reset for DATA Line DMA SW

RSTC Software Reset for CMD line

RSTA Software Reset for ALL

IPP_RST_N

DTOCV Data Timeout Counter Value

SDCLKFS SDCLK Frequency Select SDCLK

DVS divisorSDCLK

14.3.6.13 Interrupt Status (uSDHCx_INT_STATUS)

Normal Interrupt Signal Enable 1 0

image-20220112142650887

Image not found or type unknown

image-20220112142709227

Image not found or type unknown

Command Complete	Command Timeout Error	Meaning of the Status
0	0	X
X	1	Response not received within 64 SDCLK cycles
1	0	Response received

Transfer Complete	Data Timeout Error	Meaning of the Status
0	0	X
0	1	Timeout occurred during transfer
1	X	Data Transfer Complete

Command Complete	Command Timeout Error	Meaning of the Status
0	0	No error
0	1	Response Timeout Error
1	0	Response CRC Error

Command Complete	Command Timeout Error	Meaning of the Status
1	1	CMD line conflict

DMAE DMAError DMA

TNE TuningERROR tuning

AC12E Auto CMD12 ERROR Auto CMD12

DEBE Data End Bit ERROR

DCE Data CRC ERROR CRC

DTOE data timeout ERROR

CIE Command Index Error

CEBE Command End Bit ERROR

CCE Command CRC error crc

CTOE Command timeout error

TP tuning pass tuning SD3.0 SDR104

RTE Re-Tuning Event Tuning SD3.0 SDR104

CINT card interrupt

CRM card remove

CINS card insert

BRR Buffer Read Ready

BWR Buffer Read Ready

DINT DMA interrupt DMA

BGE block gap event

TC Transfer Complete

CC Command Complete

14.3.6.14 Interrupt Status Enable (uSDHCx_INT_STATUS_EN)

image-20220112142737457

Image not found or type unknown

image-20220112142740821

Image not found or type unknown

Interrupt Status uSDHCx_INT_STATUS_EN 0 Interrupt Status

14.3.6.15 Interrupt Signal Enable (uSDHCx_INT_SIGNAL_EN)

image-20220112142801615

Image not found or type unknown

14.3.6.16 Auto CMD12 Error Status (uSDHCx_AUTOCMD12_ERR_STATUS)

image-20220112142807960

Image not found or type unknown

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

SMP_CLK_SEL Sample Clock Select

EXECUTE_TUNING execute tuning tuning

CNIBAC12E Command not Issued By Auto CMD12 Error Auto CMD12

AC12IE Auto CMD12/23 Index Error Auto CMD12

AC12CE Auto CMD12/23 CRC Error Auto CMD12 CRC

AC12EBE Auto CMD12/23 End Bit Error Auto CMD12

AC12TOE Auto CMD12/23 Timeout Error Auto CMD12

AC12NE Auto CMD12 Not Executed Auto CMD12

14.3.6.17 Host Controller Capabilities (uSDHCx_HOST_CTRL_CAP)

image-20220112142823623

Image not found or type unknown

image-20220112142827981

Image not found or type unknown

VS18 Voltage Support 1.8v 1.8v

VS30 Voltage Support 3.0v 3.0v

VS33 Voltage Support 3.3v 3.3v

SRS Suspend/Resume support suspend/resume

DMAS DMA support DMA

HSS high speed support HS

ADMAS ADMA support ADMA

MBL Max Block Length

RETURNIGN MODE retuning

USE_TUNING_SDR50 SDR50 tuning

TIMER_COUNT_RETUNING

DDR50_SUPPORT DDR50

SDR104_SUPPORT SDR104

SDR50_SUPPORT SDR50

14.3.6.18 Watermark Level (uSDHCx_WTMK_LVL)

image-20220112142845425

Image not found or type unknown

14.3.6.19 Mixer Control (uSDHCx_MIX_CTRL)

image-20220112142848599

Image not found or type unknown

FBCLK_SEL FeedBack clock Source Selection SD3.0 SDR104

AUTO_TUNE_EN tune

SMP_CLK_SEL Tuned clk fixed clk data/cmd

EXE_TUNE Execute Tuning SD3.0 SDR104

AC23EN Auto CMD23 Enable Auto CMD23

NIBBLE_POS DDR 4bit

MSBSEL

DTDSEL Data transfer Direction Select

DDR_EN Dual Data rate Mode selection DDR

AC12EN Auto CMD12 Enable Auto CMD12

BCEN Block Count Enable

DMAEN DMA Enable DMA

14.3.6.20 Force Event (uSDHCx_FORCE_EVENT)

image-20220112142905813

Image not found or type unknown

14.3.6.21 ADMA Error Status Register(uSDHCx_ADMA_ERR_STATUS)

ADMA ADMA Error Status Register ADMA ADMA System Address register

ST_STOP ADMA System Address register

ST_FDS ADMA System Address register

ST_CADR change ADDRESS

ST_TRF ADMA System Address register

ADMA Error State Coding

D01-D00	ADMA Error State (when error has occurred)	Contents of ADMA System Address Register
00	ST_STOP (Stop DMA)	Holds the address of the next executable Descriptor command
01	ST_FDS (Fetch Descriptor)	Holds the valid Descriptor address
10	ST_CADR (Change Address)	No ADMA Error is generated
11	ST_TFR (Transfer Data)	Holds the address of the next executable Descriptor command

image-20220112142915422

Image not found or type unknown

ADMADCE ADMA Descriptor Error ADMA ADMA

ADMALME ADMA Length Mismatch Error ADMA BLOCK COUNT ENABLE LENGTH BLOCK LENGTH

ADMAAES ADMA Error State ADMA

14.3.6.22 ADMA System Address (uSDHCx_ADMA_SYS_ADDR)

ADMA ADMA

image-20220112142922547

Image not found or type unknown

14.3.6.23 DLL (Delay Line) Control (uSDHCx_DLL_CTRL)

14.3.6.24 DLL Status (uSDHCx_DLL_STATUS)

14.3.6.25 CLK Tuning Control and Status (uSDHCx_CLK_TUNE_CTRL_STATUS)

tuning default speed high speed

14.3.6.26 Vendor Specific Register (uSDHCx_VEND_SPEC)

Image not found or type unknown

ADMA EXT_DMA_EN 0

14.3.6.27 MMC Boot Register (uSDHCx_MMC_BOOT)

image-20220112142955603

Image not found or type unknown

BOOT_CLK_CNT

DISABLE_TIME_OUT 0 1

AUTO_SABG_EN boot

BOOT_EN fastboot

BOOT_MODE 0 normal boot 1 Alternative boot

BOOT_ACK boot ACK

DTOCVACK BOOT ack

14.3.6.28 Vendor Specific 2 Register (uSDHCx_VEND_SPEC2)

SDR tuning

image-20220112143020040

Image not found or type unknown

14.3.6.29 Tuning Control Register (uSDHCx_TUNING_CTRL)

default speed high speed

14.3.7 CMD Response

14.3.7.1

14.3.7.1-14.3.7.3 SD Host_Controller_Simplified_Specification_Ver3.00 imx

image-20220112143043372

Image not found or type unknown

PartA2_SD Host_Controller_Simplified_Specification_Ver3.00 figure3-11

1 Present State Command Inhabit(CMD) Command Inhabit(CMD) 0 Command Inh

2 command busy 3 5

3 abort command 5 abort command 4

4 Present State Command Inhabit(DAT) Command Inhabit(DAT) 0

5 command

6 command

command command

7 command complete sequence

14.3.7.2

PartA2_SD Host_Controller_Simplified_Specification_Ver3.00 figure3-12

image-20220112143222038

Image not found or type unknown

1 Transfer Mode Response Interrupt Disable 1 response check 4 Command Co
Complete 2

2 Normal Interrupt Status Command Complete 1 bit

3 Response

4 Transfer Complete 5 7

5 Transfer Complete 6

6 Normal Interrupt Status Transfer Complete 1 bit

7 Response data 8 9

8 No ERROR

9 Response Contents Error

14.3.7.3 DAT ADMA

PartA2_SD Host_Controller_Simplified_Specification_Ver3.00 figure3-15

image-20220112143230507

Image not found or type unknown

1 ADMA

2 ADMA ADMA System Address

3 Block Size

4 Block Count

5 Argument

6 Transfer Mode Multi/Single Block Select Block Count Enable Data Transfer Direction Enable DMA enable

7 Command

8 Response check 11 Command Complete Command Complete 1 9

9 Normal Interrupt Status Command Complete 1 bit

10 Response

11 Transfer Complete ADAM Error

12 Transfer Complete 1 13 ADAM Error 1 14

13 Normal Interrupt Status Transfer Complete 1 bit

14 Error Interrupt Status ADMA Error Interrupt Status 1 bit

15 abort ADMA

14.3.7.4 DATA

14.3.7.4-14.3.7.6 SD Host_Controller_Simplified_Specification_Ver3.00 imx6ull uSDHC Host_Controller_Simplified_Specification_Ver3.00 imx6ull

1 CIHB 0

2 CMDTYP, DPSEL, CICEN, CCCEN, RSTTYP, DTDSEL

3 CMDARG

4 XFERTYP

5 INT_STATUS CC

6 1 CC

14.3.7.5 DATA

1 CIHB 0

2 CDIHB 0

3 ADMA

4 ADMA_SYS_ADDR ADMA

5 BLK_ATT

6PROT_CTRL DMA ADMA2

7MIX_CTRL DTDSEL DMAEN

8 CMDARG

9 XFERTYP

10 INT_STATUS CC

11 1 CC

12 INT_STATUS TC

13 1 TC

14.3.7.6 /

1 CIHB 0

2 CDIHB 0

3 ADMA

4 ADMA_SYS_ADDR ADMA

5 BLK_ATT

6PROT_CTRL DMA ADMA2

7MIX_CTRL DTDSEL DMAEN 8 BCEN MSBSEL AUTO CMD12 AC12

9 CMDARG

10 XFERTYP

11 INT_STATUS CC

12 1 CC

13 INT_STATUS TC

14 1 TC

14.3.7.7 SD Version2.0

1 CMD0

2 SDHC CMD8

3 Version2.0 CMD8 legacy cards SD CMD8

4 ACMD41 OCR 0

5 1S ACMD41 HCS HCS 1, big31 1 CCS CCS 0 SDSC 1

6 1.8v ACMD41 S18R S18A 1 1.8v CMD11

7 CMD2 CID

8 CMD3 RCA 0

9 CMD9 RCA CSD

10 CMD7 RCA

11 ACMD51 RCA SCR

12 ACMD6 0 1bit 2 4bit

13 CMD6 function group 1 speed mode

14.3.7.8 eMMC JESD84-B51 A.6

a.

1 (2.7~3.6v)

2 400k

3 1ms 74

4 CMD0

5 CMD1 0x00FF8000 0x00000080

6 R3

7 OCR busy 0 5 6

8 R3 High Voltage Dual Voltage 0x80FF8000 High Voltage 0x80FF8080

9 R3 inactive

16

10 Dual Voltage

11 1.70~1.95V

12 1ms 74

13 CMD1 0x00000080

14 R3 0x80FF8080

15 OCR busy 0 13 14

b. CID RCA

16 CMD2

17 R2 CID

18 CMD3 1

CSD

19 CMD9

20 R2 CSD

21 CSD SPEC_VERS V4.0 high speed SWITCH SEND_EXT_CSD

High Speed

high speed high speed

22 CMD7 RCA tran state

23 CMD8 SEND_EXT_CSD EXT_CSD power class 26-37

24 CMD6 0x03B9_0100 EXT_CSD HS_TIMING 0x1

24.1 R1 busy

24.2 busy high speed timing

25 0~26/52MHz

a.

26 CMD19

27

27.1 8 MSB LSB 0x0000_0000_0000_AA55

27.2 4 MSB LSB 0x0000_AA55

27.3 1 0x80

28 NCR

29 CMD14

29.1 8 8

29.2 4 4

29.3 1 1

30 27 XNOR

31 MASK

31.1 8 MSB LSB 0x0000_0000_0000_FFFF

31.2 4 MSB LSB 0x0000_FFFF

31.3 1 0xC0

32 0

b

33

34 power class power class CMD6 EXT_CSD POWER_CLASS power class

35 CMD6

36 CMD6 EXT_CSD BUS_WIDTH 4bit 0x03B7_0100 8bit 0x03B7_0200

14.3.8 eMMC

image-20220112143422049

Image not found or type unknown

User Data Area Boot Area Partitions Replay Protected AreaPartition RPMB Boot Area Partitions

- Boot Area Partitions 128 KB eMMC
- RPMB 128 KB
- General Purpose Area Partitions Write Protect Group

General Purpose Area Partition enhanced or extended technological features media features boot RPMB

boot RPMB General Purpose Area Partitions

image-20220112143458999

Image not found or type unknown

General Purpose Area Partition Enhanced User Data Area

boot rpmb user data area CSD

14.4

led.imx TF eMMC

led.imx led.imx led.imx.h unsigned char led_imx_image l

14.4.1

image-20220112143657344

Image not found or type unknown

100sdk_imx6ull SD uSDHC1

image-20220112143701287

Image not found or type unknown

100sdk_imx6ull eMMC uSDHC2

TF USDHC1_CMD USDHC1_CLK USDHC1_DATA0-3 USDHC1_CD_B eMMC USDHC2_CMD USDHC2_CLK USDHC2_RESET_B USDHC2_DATA PinMux IOMUXC_SetPinConfig

14.4.2

USDHCx_CLK_ROOT 198M uSDHCx_SYS_CTRL[DVS] uSDHCx_SYS_CTRL[SDCLKFS] M

14.4.3 CMD_XFR_TYP 16bit

CRC DATA CMD_XFR_TYP 16bit

Git NoosProgramProject/(14_TF /006_sd/sd.h

```
/*
 * CMDx:  normal command,
 * ACMDx: app command,
 * SCMDx: sd command,
 * MCMDx: mmc command
 * R1 R1b R3 R6 R7 48bit;
 * R2 136bit
 * bit1-0(bit17-16) RSPTYP 00 no response; 01 lengeh 136; 02 length 48; 03 length 48 and check busy
 * bit3(bit19) CCCEN command crc check enable
 * bit4(bit20) CICEN Command index check enable
 * bit5(bit21) DPSEL data present select
 */
#define CMD0　0x0000　/* GO_IDLE_STATE　no response */
#define CMD1　0x0102　/* SEND_OP_COND　R3 */
#define CMD2　0x0209　/* ALL_SEND_CID　R2 */
#define CMD3　0x031a　/* SET_RELATIVE_ADDR (R6/R1) */
#define ACMD6　0x061a　/* SET_BUS_WIDTH R1　*/
#define ACMD23　0x171a　/* SET_WR_BLK_ERASE_CNT　R1 */
#define SCMD6　0x063a　/* SWITCH_BUSWIDTH　R1 */
#define MCMD6　0x061b　/* SET_EXT_CSD R1B */
#define CMD7　0x071b　/* SELECT_CARD R1B */
#define CMD8　0x081a　/* SEND_IF_COND　R7 */
#define MCMD8　0x083a　/* GET_EXT_CSD R1 */
#define CMD9　0x0909　/* GET_THE_CSD R2 */
#define CMD11　0x0b1a　/* SWITCH VOLTAGE R1 */
#define CMD13　0x0d1a　/* SEND_STATUS R1 */
#define ACMD13　0x0d3a　/* SEND_STATUS R1 */
#define CMD16　0x101a　/* SET_BLOCKLEN　R1 */
#define CMD17　0x113a　/* READ_SINGLE_BLOCK R1 */
#define CMD18　0x123a　/* READ_MULTIPLE_BLOCK R1 for SD R7 */
#define SCMD19  0x133a  /* SEND_TUNING R1 */
#define MCMD21  0x153a  /* SEND_TUNING R1 */
#define CMD24　0x183a　/* WRITE_BLOCK R1 */
#define CMD25　0x193a　/* WRITE_MULTIPLE_BLOCK　R1 */
#define CMD32　0x201a　/* ERASE_WR_BLK_START R1 */
#define CMD33　0x211a　/* ERASE_WR_BLK_END　R1 */
#define CMD35　0x231a　/* ERASE_GROUP_START R1 */
#define CMD36　0x241a　/* ERASE_GROUP_END　R1 */
```

```
#define CMD38□0x261b□/* ERASE R1B */
#define ACMD41□0x2902□/* SD_SEND_OP_COND R3 */
#define ACMD42□0x2a1b□/* LOCK_UNLOCK R1B */
#define ACMD51□0x333a□/* SEND_SCR□R1 */
#define CMD55□0x371a□/* APP_CMD□R1 */
```

14.4.4 DATA

CMD CMD PRES_STATE CIHB 0 CMD CMD_XFR_T

Git NoosProgramProject/(14_TF /006_sd/sd.c)

```
static int USDHC_SendCommand(USDHC_Type *base, u32 command, u32 argument)
{
□/* Wait until command/data bus out of busy status. */
□while (base->PRES_STATE & USDHC_PRES_STATE_CIHB_MASK) {
□}

□/* config the command xfertype and argument */
□base->CMD_ARG = argument;
□base->CMD_XFR_TYP = command << 16;

□return USDHC_WaitCommandDone(base);
}
```

INT_STATUS CC CRC

Git NoosProgramProject/(14_TF / 006_sd/sd.c)

```
static int USDHC_WaitCommandDone(USDHC_Type *base)
{
□int error = 0;
□uint32_t interruptStatus = 0U;

□/* Wait command complete or USDHC encounters error. */
□while (!(base->INT_STATUS & (USDHC_INT_STATUS_CC_MASK | kUSDHC_CommandErrorFlag))) {
□}

□interruptStatus = base->INT_STATUS;
□if ((interruptStatus & kUSDHC_CommandErrorFlag) != 0U)
```

```
    {
        printf("cmd errror, CMD is 0x%x, INT_STATUS is 0x%x\r\n", base->CMD_XFR_TYP, interruptStatus);
        USDHC_Dump_All(base);
        error = -1;
    }

    USDHC_ClearInterruptStatusFlags(
        base, (kUSDHC_CommandCompleteFlag | kUSDHC_CommandErrorFlag | kUSDHC_TuningErrorFlag));

    return error;
}
```

14.4.5 ADMA

address length attribute ADMA2 DMA System

Git NoosProgramProject/(14_TF /006_sd/sd.c)

```
static int USDHC_CreateDescTable(u8 *data, u32 len)
{
    int i;
    u32 dma_len, entries;

    entries = len / USDHC_ADMA2_DESCRIPTOR_MAX_LENGTH_PER_ENTRY;
    if ((len % USDHC_ADMA2_DESCRIPTOR_MAX_LENGTH_PER_ENTRY) != 0U)
        entries++;

    for (i = 0; i < entries; i++)
    {
        if (len > USDHC_ADMA2_DESCRIPTOR_MAX_LENGTH_PER_ENTRY)
        {
            dma_len = USDHC_ADMA2_DESCRIPTOR_MAX_LENGTH_PER_ENTRY;
            len -= USDHC_ADMA2_DESCRIPTOR_MAX_LENGTH_PER_ENTRY;
        }
        else
        {
```

```
        dma_len = len;
    }

    /* Each descriptor for ADMA2 is 64-bit in length */
    g_adma2_tablbe[i].address = (uint32_t *)data;
    g_adma2_tablbe[i].attribute = (dma_len << USDHC_ADMA2_DESCRIPTOR_LENGTH_SHIFT);
    g_adma2_tablbe[i].attribute |= kUSDHC_Adma2DescriptorTypeTransfer;
    data += dma_len;
  }
  g_adma2_tablbe[entries - 1].attribute |= kUSDHC_Adma2DescriptorEndFlag;
  /*for (i = 0; i < entries; i++) {
    printf("g_adma2_tablbe[i] address is 0x%x, attribute is 0x%x\r\n",
    g_adma2_tablbe[i].address, g_adma2_tablbe[i].attribute);
    }
    */

  return 0;
}
```

14.4.6 DATA

data PRES_STATE CIHB CDIHB 0 CMD DATA CMD A ADMA ADMA_SYS_ADDR PROT_CTRL ADMA ADMA2 _WaitCommandDone USDHC_WaitDataDone

USDHC_WaitDataDone INT_STATUS TC CMD12 DMA USDHC_Dump_All dump uSDHC INT_STATUS ADMA_ERR

GitNoosProgramProject/(14_TF /006_sd/sd.c)

```
static int USDHC_WaitDataDone(USDHC_Type *base)
{
  int error = 0;
  uint32_t interruptStatus = 0U;

  /* Wait command complete or USDHC encounters error. */
  while (!(base->INT_STATUS & (kUSDHC_DataCompleteFlag | kUSDHC_DataErrorFlag |
  kUSDHC_DmaErrorFlag))) {
  }
```

```
    interruptStatus = base->INT_STATUS;

    if ((interruptStatus & (kUSDHC_DataErrorFlag | kUSDHC_DmaErrorFlag)) != 0U) {
        printf("data or ADMA errror, CMD is 0x%x, INT_STATUS is 0x%x\r\n", base->CMD_XFR_TYP,
interruptStatus);
        USDHC_Dump_All(base);
        error = -1;
    }

    USDHC_ClearInterruptStatusFlags(
        base, (kUSDHC_DataCompleteFlag | kUSDHC_DataErrorFlag | kUSDHC_DmaErrorFlag));

    return error;
}
```

14.4.7

PRES_STATE CIHB CDIHB 0 CMD data ADMA ADMA ADM
MA2 MIX_CTRL DTDSEL 1 DMAEN MIX_CTRL MSBSEL BCEN
XFR_TYP **Git NoosProgramProject/(14_TF /006_sd/sd.c)**

```
int sd_read_blocks(USDHC_Type *base, uint8_t *buffer, uint32_t startBlock, uint32_t
blockCount)
{
    int err = 0;

    /* Wait until command/data bus out of busy status. */
    while (base->PRES_STATE & USDHC_PRES_STATE_CIHB_MASK) {
    }
    while (base->PRES_STATE & USDHC_PRES_STATE_CDIHB_MASK) {
    }
    /* set ADMA descriptor */
    USDHC_CreateDescTable(buffer, blockCount * 512);

    /* When use ADMA, disable simple DMA */
    base->DS_ADDR = 0U;
    base->ADMA_SYS_ADDR = (u32) g_adma2_tablbe;
```

```
    /* config data block size/block count */
    base->BLK_ATT = (USDHC_BLK_ATT_BLKSIZE(512) | USDHC_BLK_ATT_BLKCNT(blockCount));

    /* disable the external DMA if support */
    base->VEND_SPEC &= ~USDHC_VEND_SPEC_EXT_DMA_EN_MASK;
    /* select DMA mode and config the burst length */
    base->PROT_CTRL &= ~(USDHC_PROT_CTRL_DMASEL_MASK | USDHC_PROT_CTRL_BURST_LEN_EN_MASK);
    base->PROT_CTRL |=
        USDHC_PROT_CTRL_DMASEL(kUSDHC_DmaModeAdma2) |
    USDHC_PROT_CTRL_BURST_LEN_EN(kUSDHC_EnBurstLenForINCR);

    if (blockCount == 1) {
        /* single block read*/
        /* direction:read, enable DMA */
        base->MIX_CTRL &= ~(USDHC_MIX_CTRL_MSBSEL_MASK | USDHC_MIX_CTRL_BCEN_MASK |
    USDHC_MIX_CTRL_DTDSEL_MASK |
            USDHC_MIX_CTRL_AC12EN_MASK | USDHC_MIX_CTRL_DMAEN_MASK);
        base->MIX_CTRL |= USDHC_MIX_CTRL_DTDSEL_MASK | USDHC_MIX_CTRL_DMAEN_MASK;

        /* config the command xfertype and argument */
        base->CMD_ARG = startBlock;
        base->CMD_XFR_TYP = CMD17 << 16;
    } else {
        /* multi block read */
        /* block count enable, Multiblcok, direction:read, enable DMA */
        base->MIX_CTRL &= ~(USDHC_MIX_CTRL_MSBSEL_MASK | USDHC_MIX_CTRL_BCEN_MASK |
    USDHC_MIX_CTRL_DTDSEL_MASK |
            USDHC_MIX_CTRL_AC12EN_MASK | USDHC_MIX_CTRL_DMAEN_MASK);
        base->MIX_CTRL |= USDHC_MIX_CTRL_MSBSEL_MASK | USDHC_MIX_CTRL_BCEN_MASK |
            USDHC_MIX_CTRL_DTDSEL_MASK | USDHC_MIX_CTRL_AC12EN_MASK | USDHC_MIX_CTRL_DMAEN_MASK;

        /* config the command xfertype and argument */
        base->CMD_ARG = startBlock;
        base->CMD_XFR_TYP = CMD18 << 16;
    }

    err = USDHC_WaitCommandDone(base);
    if (err < 0)
```

```
    return err;

    err =  USDHC_WaitDataDone(base);
    if (err < 0)
    return err;

    err = SD_WaitReadWriteComplete(base);
    if (err & 0xc0200000) {
    err = -1;
    printf("%s read error\r\n", __func__);
    }

    return err;
}
```

14.4.8

MIX_CTRL DTDSEL 0 CMD24 CMD25 cmd13

Git NoosProgramProject/(14_TF /006_sd/sd.c)

```
int sd_write_blocks(USDHC_Type *base, uint8_t *buffer, uint32_t startBlock, uint32_t
blockCount)
{
    int err = 0;

    /* Wait until command/data bus out of busy status. */
    while (base->PRES_STATE & USDHC_PRES_STATE_CIHB_MASK) {
    }
    while (base->PRES_STATE & USDHC_PRES_STATE_CDIHB_MASK) {
    }
    /* set ADMA descriptor */
    USDHC_CreateDescTable(buffer, blockCount * 512);

    /* When use ADMA, disable simple DMA */
    base->DS_ADDR = 0U;
    base->ADMA_SYS_ADDR = (u32) g_adma2_tablbe;

    /* config data block size/block count */
```

```
    base->BLK_ATT = (USDHC_BLK_ATT_BLKSIZE(512) | USDHC_BLK_ATT_BLKCNT(blockCount));

    /* disable the external DMA if support */
    base->VEND_SPEC &= ~USDHC_VEND_SPEC_EXT_DMA_EN_MASK;
    /* select DMA mode and config the burst length */
    base->PROT_CTRL &= ~(USDHC_PROT_CTRL_DMASEL_MASK | USDHC_PROT_CTRL_BURST_LEN_EN_MASK);
    base->PROT_CTRL |=
        USDHC_PROT_CTRL_DMASEL(kUSDHC_DmaModeAdma2) |
    USDHC_PROT_CTRL_BURST_LEN_EN(kUSDHC_EnBurstLenForINCR);

    if (blockCount == 1) {
        /* single block wrtie*/
        /* direction:read, enable DMA */
        base->MIX_CTRL &= ~(USDHC_MIX_CTRL_MSBSEL_MASK | USDHC_MIX_CTRL_BCEN_MASK |
    USDHC_MIX_CTRL_DTDSEL_MASK |
            USDHC_MIX_CTRL_AC12EN_MASK | USDHC_MIX_CTRL_DMAEN_MASK);
        base->MIX_CTRL |= USDHC_MIX_CTRL_DMAEN_MASK;

        /* config the command xfertype and argument */
        base->CMD_ARG = startBlock;
        base->CMD_XFR_TYP = CMD24 << 16;
    } else {
        /* multi block write */
        /* block count enable, Multiblcok, direction:read, enable DMA */
        base->MIX_CTRL &= ~(USDHC_MIX_CTRL_MSBSEL_MASK | USDHC_MIX_CTRL_BCEN_MASK |
    USDHC_MIX_CTRL_DTDSEL_MASK |
            USDHC_MIX_CTRL_AC12EN_MASK | USDHC_MIX_CTRL_DMAEN_MASK);
        base->MIX_CTRL |= USDHC_MIX_CTRL_MSBSEL_MASK | USDHC_MIX_CTRL_BCEN_MASK |
            USDHC_MIX_CTRL_AC12EN_MASK | USDHC_MIX_CTRL_DMAEN_MASK;

        /* config the command xfertype and argument */
        base->CMD_ARG = startBlock;
        base->CMD_XFR_TYP = CMD25 << 16;
    }

    err = USDHC_WaitCommandDone(base);
    if (err < 0)
        return err;
```

```
    err =  USDHC_WaitDataDone(base);
    if (err < 0)
        return err;

    err = SD_WaitReadWriteComplete(base);
    if (err & 0xe4000000) {
        err = -1;
        printf("%s write error\r\n", __func__);
    }

    return err;
}
```

14.4.9 TF

Git NoosProgramProject/(14_TF /006_sd)/sd.c

```
int sd_init(USDHC_Type *base)
{
    int err;
    u32 retries, acmd41arg = 0, resp[4], raw_scr[2];

    USDHC_Init(base);
    CardInsertDetect(base);

    /* set DATA bus width */
    USDHC_SetDataBusWidth(base, kUSDHC_DataBusWidth1Bit);
    /*set card freq to 400KHZ*/
    g_sd_card.busClock_Hz = USDHC_SetSdClock(base, 198000000U, SDMMC_CLOCK_400KHZ);
    /* send card active */
    USDHC_SetCardActive(base, 100U);
    /* Get host capability. ignore just decision form spec HOST_CTRL_CAP*/

    /* CMD0 - GO_IDLE_STATE software reset and set into idle */
    err = USDHC_SendCommand(base, CMD0, 0x0);
    if (err < 0)
        return -1;

    /* verify card interface operating condition. */
```

```
    for (retries = 0; retries < 10; retries++) {
        /* CMD8 (physical layer spec Ver2.0 is mandatory) */
        err = USDHC_SendCommand(base, CMD8, 0x01aa);
        if (err == 0)
            break;
    }

    if (err == 0) {
        /* SDHC or SDXC card */
        acmd41arg |= kSD_OcrHostCapacitySupportFlag;
    } else {
        /* SDSC card */
        err = USDHC_SendCommand(base, CMD0, 0x0);
        if (err !=  0)
            return -1;
    }

    acmd41arg |= (kSD_OcrVdd32_33Flag | kSD_OcrVdd33_34Flag);
    for(retries = 0; retries < 5000; retries++) {
        /* rca = 0 since card is in IDLE state */
        err = USDHC_SendCommand(base, CMD55, 0x0);
        if (err < 0)
            return -1;

        /* ACMD41 to query OCR */
        err = USDHC_SendCommand(base, ACMD41, acmd41arg);
        if (err < 0)
            return -1;

        if (base->CMD_RSP0 & kSD_OcrPowerUpBusyFlag) {
            g_sd_card.ocr = base->CMD_RSP0;
            printf("ocr is 0x%x\r\n", g_sd_card.ocr);
            break;
        }
    }

    if (retries >= 1000 ) {
        printf("HandShakeOperationConditionFailed\r\n");
        return -1;
```

```
    }

    /* check 1.8V support */
    if (g_sd_card.ocr & kSD_OcrSwitch18AcceptFlag) {
        printf("support 1.8v\r\n");
    }
    // our board just support 3.3v, ignore it
    //SD_SwitchVoltage(card))

    /* get CID number */
    err = USDHC_SendCommand(base, CMD2, 0x0);
    if (err < 0)
        return -1;
    USDHC_GetResponse(base, resp);
    memcpy(g_sd_card.rawCid, resp, sizeof(resp));
    SD_DecodeCid(&g_sd_card, g_sd_card.rawCid);

    /* publish a new relative card address(RCA) */
    err = USDHC_SendCommand(base, CMD3, 0x0);
    if (err < 0)
        return -1;
    g_sd_card.relativeAddress = base->CMD_RSP0 >> 16;
    printf("relative address is 0x%x\r\n", g_sd_card.relativeAddress);

    /* get CID number */
    err = USDHC_SendCommand(base, CMD9, g_sd_card.relativeAddress << 16);
    if (err < 0)
        return -1;
    USDHC_GetResponse(base, resp);
    memcpy(g_sd_card.rawCsd, resp, sizeof(resp));
    SD_DecodeCsd(&g_sd_card, g_sd_card.rawCsd);
    printf("card is %s", g_sd_card.csd.csdStructure ? "SDHC/SDXC" : "SDSC");
    printf("card block count is %d\r\n", g_sd_card.blockCount);
    printf("card sector size is %d\r\n", g_sd_card.blockSize);
    printf("card command class is 0x%x\r\n", g_sd_card.csd.cardCommandClass);

    /* CMD7: SelectCard */
    err = USDHC_SendCommand(base, CMD7, g_sd_card.relativeAddress << 16);
    if (err < 0)
```

```
    return err;

    /* ACMD51: Read SCR */
    err = USDHC_SendCommand(base, CMD55, g_sd_card.relativeAddress << 16);
    if (err < 0)
        return err;
    err = USDHC_SendCommand_with_data(base, ACMD51, 0, raw_scr, 8);
    if (err < 0)
        return err;
    raw_scr[0] = SWAP_32(raw_scr[0]);
    raw_scr[1] = SWAP_32(raw_scr[1]);
    memcpy(g_sd_card.rawScr, raw_scr, sizeof(raw_scr));
    SD_DecodeScr(&g_sd_card, g_sd_card.rawScr);

    printf("scr[0] is 0x%x, scr[1] is 0x%x\r\n", raw_scr[0], raw_scr[1]);
    printf("sd specification is 0x%x\r\n", g_sd_card.scr.sdSpecification);
    if ((g_sd_card.scr.sdBusWidths & 0x4U) == 0) {
        printf("The card can not support 4bit width");
        return -1;
    }
    /* speed class control cmd */
    if ((g_sd_card.scr.commandSupport & 0x01U) == 0)
    {
        printf("The card can not support Speed Class Control (CMD20)\r\n");
    }
    /* set block count cmd */
    if ((g_sd_card.scr.commandSupport & 0x02U) == 0)
    {
        printf("The card can not support Support SetBlockCountCmd (CMD23)\r\n");
    }

    /* Set to max frequency in non-high speed mode. */
    g_sd_card.busClock_Hz = USDHC_SetSdClock(base, 198000000U, SD_CLOCK_25MHZ);

    /* Set to 4-bit data bus mode. */
    /* set card to 4 bit width*/
    err = USDHC_SendCommand(base, CMD55, g_sd_card.relativeAddress << 16);
    if (err < 0)
        return err;
```

```
    err =  USDHC_SendCommand(base, ACMD6, 2);
    if (err < 0)
        return err;
    /* set host to 4 bit width*/
    base->PROT_CTRL = ((base->PROT_CTRL & ~USDHC_PROT_CTRL_DTW_MASK) | USDHC_PROT_CTRL_DTW(1));

    /* set block size to 512 : CMD16 SET_BLOCKLEN */
    err = USDHC_SendCommand(base, CMD16, FSL_SDMMC_DEFAULT_BLOCK_SIZE);
    if (err < 0)
        return err;

    /* select high speed successful, switch clock to 50M */
    if (SD_SelectBusTiming(base) == 0)
        g_sd_card.busClock_Hz = USDHC_SetSdClock(base, 198000000U, SD_CLOCK_50MHZ);
        //g_sd_card.busClock_Hz = USDHC_SetSdClock(base, 396000000U, SD_CLOCK_50MHZ);
    //printf("clock is %d, base->SYS_CTRL is 0x%x\r\n", g_sd_card.busClock_Hz, base->SYS_CTRL);
    printf("sd init sccessful\r\n");

    return 0;
}
```

14.4.10 TF

led.imx led.imx.h led_imx_image 1024 TF

Git NoosProgramProject/(14_TF /006_sd/sd.c)

```
printf("start test sd\r\n");
    sd_init(USDHC1);

    memset(sd_read_buf, 0, sizeof(sd_read_buf));
    sd_read_blocks(USDHC1, sd_read_buf, 2, 2);

    printf("read 2 sectors from 2nd sector\r\n");
    for(i = 0; i < 1024; i++) {
        printf("%02x ", sd_read_buf[i]);
        if((i + 1)%16 == 0)
            printf("\r\n");
    }
```

```
    printf("burn led.imx to sd\r\n");
    sd_write_blocks(USDHC1, led_imx_image + 1024, 2, (sizeof(led_imx_image) - 1024) >> 9);
    printf("please set dip switch to SD boot, and push reset, then green led blink \r\n");

    memset(sd_read_buf, 0, sizeof(sd_read_buf));
    sd_read_blocks(USDHC1, sd_read_buf, 2, 2);

    printf("read 2 sectors from 2nd sector after burn led_imx_image:\r\n");
    for(i = 0; i < 1024; i++) {
        printf("%02x ", sd_read_buf[i]);
        if((i + 1)%16 == 0)
            printf("\r\n");
    }
    printf("please set dip switch to SD boot, and push reset, then green led blink \r\n");
```

eMMC TF TF

14.4.11 eMMC boot partition

Extended CSD register

PARTITION_CONFIG[179]

image-20220112144324646

Image not found or type unknown

image-20220112144330003

Image not found or type unknown

PARTITION_ACCESS 0 user partition 1 boot1 partition 2 boot2 partition

image-20220112144334889

Image not found or type unknown

CMD6

Bit31-26 0

Bit25-24 access mode

Bit23-16

Bit15-8

Bit7-3 0

Bit2-0 cmd set

image-20220112144339189

Image not found or type unknown

00 command set

01 value 1

10 value 1

11 value

user partition boot1 partition

(1<< 24) | (179 << 16) | (1 << 8) | 0

boot1 partition user partition

(2<<24) | (179 << 16) | (1 << 8) | 0

Git NoosProgramProject/(14_TF /006_sd/sd.c)

```
static status_t MMC_SetExtendedCsdConfig(USDHC_Type *base, mmc_extended_csd_access_mode_t
access_mode, uint8_t index, uint8_t value)
{
    int err;

    err = USDHC_SendCommand(base, MCMD6, (access_mode << 24) | (index << 16) | (value << 8));
    if (err < 0)
        return -1;

    return 0;

}
static status_t MMC_SelectPartition(USDHC_Type *base, mmc_extended_csd_access_mode_t
access_mode, mmc_access_partition_t part_number)
```

```
{
    int err;

    err = MMC_SetExtendedCsdConfig(base, access_mode, 179,  part_number);
    if (err < 0)
        return -1;

    err = SD_WaitReadWriteComplete(base);
    if (err & (1 << 7)) {
        printf("failed select partition\r\n");
        return -1;
    }

    return 0;
}
int MMC_SelectBoot1Partition(USDHC_Type *base)
{
    return MMC_SelectPartition(base, kMMC_ExtendedCsdAccessModeSetBits, 1);
}
```

14.4.12 eMMC

Git NoosProgramProject/(14_TF /006_sd/sd.c)

```
int mmc_init(USDHC_Type *base)
{
    int err;
    u32 retries, resp[4], maxBusClock_Hz, bus_width = 4;

    USDHC_Init(base);

    /* set DATA bus width */
    USDHC_SetDataBusWidth(base, kUSDHC_DataBusWidth1Bit);
    /*set card freq to 400KHZ*/
    g_mmc_card.busClock_Hz = USDHC_SetSdClock(base, 198000000U, SDMMC_CLOCK_400KHZ);
    /* send card active */
    USDHC_SetCardActive(base, 100U);
```

```
    /* Get host capability. ignore just decision form spec HOST_CTRL_CAP*/

    /* CMD0 - GO_IDLE_STATE software reset the bus and set into idle */
    err = USDHC_SendCommand(base, CMD0, 0x0);
    if (err < 0)
        return -1;

        /* Hand-shaking with card to validata the voltage range Host first sending its expected
           information.*/
        err = USDHC_SendCommand(base, CMD1, 0x0);
    if (err < 0)
        return -1;
    g_mmc_card.ocr = base->CMD_RSP0;
    g_mmc_card.ocr |= 2 << 29;/* set access mode to sector mode */
    for (retries = 0; retries < 1000; retries++) {
        err = USDHC_SendCommand(base, CMD1, 0x0);
        if (err < 0)
            return -1;

        if ((base->CMD_RSP0 & MMC_OCR_BUSY_MASK))
            break;
    }

    if (retries >= 1000 ) {
        printf("HandShakeOperationConditionFailed\r\n");
        return -1;
    }
    g_mmc_card.ocr = base->CMD_RSP0;
    printf("ocr is 0x%x\r\n", g_mmc_card.ocr);

    /* switch the host voltage which the card can support */
    /* ignore switch voltage, our board just support 3.3v */



    /* CMD2 Get card CID */
    err = USDHC_SendCommand(base, CMD2, 0x0);
    if (err < 0)
        return -1;
    USDHC_GetResponse(base, resp);
```

```
    memcpy(g_mmc_card.rawCid, resp, sizeof(resp));
    MMC_DecodeCid(&g_mmc_card, g_mmc_card.rawCid);

    /* Send CMD3 with a chosen relative address, with value greater than 1 */
    g_mmc_card.relativeAddress = 2;
    err = USDHC_SendCommand(base, CMD3, g_mmc_card.relativeAddress << 16);
    if (err < 0)
        return -1;

    /* CMD9 Get the CSD register content */
    err = USDHC_SendCommand(base, CMD9, g_mmc_card.relativeAddress << 16);
    if (err < 0)
        return -1;
    USDHC_GetResponse(base, resp);
    memcpy(g_mmc_card.rawCsd, resp, sizeof(resp));
    MMC_DecodeCsd(&g_mmc_card, g_mmc_card.rawCsd);

    /* Set to maximum speed in normal mode. */
    /*used to calculate the max speed in normal
        mode not high speed mode.
        For cards supporting version 4.0, 4.1, and 4.2 of the specification, the value shall be
    20MHz(0x2A).
        For cards supporting version 4.3, the value shall be 26 MHz (0x32H). In High speed mode,
    the max
        frequency is decided by CARD_TYPE in Extended CSD. */
    printf("csd tran speed is 0x%x\r\n", g_mmc_card.csd.transferSpeed);
    if (g_mmc_card.csd.transferSpeed == 0x32)
        maxBusClock_Hz = 26000000;
    else if ( g_mmc_card.csd.transferSpeed == 0x2A )
        maxBusClock_Hz = 20000000;

    g_mmc_card.busClock_Hz = USDHC_SetSdClock(base, 198000000U, maxBusClock_Hz);

        /* Send CMD7 with the card's relative address to place the card in transfer state. Puts
    current selected card in
        transfer state. */
    err = USDHC_SendCommand(base, CMD7, g_mmc_card.relativeAddress << 16);
    if (err < 0)
        return err;
```

```
    /* Get Extended CSD register content. */
    err = USDHC_SendCommand_with_data(base, MCMD8, 0, g_mmc_card.rawExtendedCsd, 512);
    if (err < 0)
        return err;
    MMC_DecodeExtendedCsd(&g_mmc_card, g_mmc_card.rawExtendedCsd);
    printf("g_mmc_card.extendedCsd.sectorCount is %d\r\n", g_mmc_card.extendedCsd.sectorCount);

    /* set block size to 512 : CMD16 SET_BLOCKLEN */
    err = USDHC_SendCommand(base, CMD16, FSL_SDMMC_DEFAULT_BLOCK_SIZE);
    if (err < 0)
        return err;

        /* switch to host support speed mode, then switch MMC data bus width and select power
    class */
    /* select bus width */
    err = MMC_SetBusWidth(base, bus_width);
    if (err < 0)
        return -1;
    printf("bus width is %d\r\n", bus_width);

    /* switch to high speed mode */
    err = MMC_SwitchBusMode(base, kMMC_HighSpeedTiming);
    if (err < 0)
        return err;
    g_mmc_card.busClock_Hz = USDHC_SetSdClock(base, 198000000U, MMC_CLOCK_52MHZ);

    printf("init MMC sucessful\r\n");

    return 0;
    }
```

14.4.13 eMMC

eMMC led partition boot1 partition user partition boot1 partition partition user partition

Git NoosProgramProject/(14_TF /006_sd/sd.c)

```
    ```c
```
```

mmc_init(USDHC2); MMC_SelectBoot1Partition(USDHC2);

```
memset(sd_read_buf, 0, sizeof(sd_read_buf));
sd_read_blocks(USDHC2, sd_read_buf, 2, 2);

printf("read 2 sectors from eMMC boot1 partition 2nd sector\r\n");
for(i = 0; i < 1024; i++) {
    printf("%02x ", sd_read_buf[i]);
    if((i + 1)%16 == 0)
        printf("\r\n");
}

/* wirte led_imx_image to eMMC boot1 partition */
sd_write_blocks(USDHC2, led_imx_image + 1024, 2, (sizeof(led_imx_image) - 1024) >> 9);
printf("please set dip switch to eMMC boot, and push reset, then green led blink\r\n");

printf("read 2 sectors from eMMC boot1 part 2nd sector after burn led_imx_image\r\n");
memset(sd_read_buf, 0, sizeof(sd_read_buf));
sd_read_blocks(USDHC2, sd_read_buf, 2, 2);
MMC_ExitBoot1Partition(USDHC2);

for(i = 0; i < 1024; i++) {
    printf("%02x ", sd_read_buf[i]);
    if((i + 1)%16 == 0)
        printf("\r\n");
}
```

14.4.14 4-1.4

** Git NoosProgramProject/(14009_timer_epit_poll)***

14.4.15 3-1.4

14.4.16 SD

1. sd.imx emmc

2. TF emmc

3.

4. 1 SD

image-20220112144530783

Image not found or type unknown

c

image-20220112144536176

Image not found or type unknown

2 **please set dip switch to SD boot, and push reset, then green led blink**

14.4.17 EMMC

1. sd.imx TF

2. TF SD

3.

4. 2 EMMC

image-20220112144557698

Image not found or type unknown

boot1 partition 2 0 led.imx 2 0 TF

image-20220112144605048

Image not found or type unknown

Revision #1
Created 3 March 2022 02:46:16 by
Updated 3 March 2022 02:46:35 by