eMMC 接口设计方案

V1.0

目 录

图

表

1 主要功能

eMMC 接口可用于连接 SD（eSD）、SDIO（eSDIO）、CE-ATA、MMC、eMMC。所有的数据交换都由 Host 端的 eMMC 控制器发起。eMMC 接口具有如下特点：

- 接口支持 SD3.01 标准，SDIO3.01 标准，CE-ATA1.1 标准，MMC4.4.1 标准，eMMC4.5.1 标准；
- 支持 SDR 标准传输模式（最高传输率 25MB/s）、SDR 高速传输模式（最高传输率 50MB/s）与 DDR 高速传输模式（最高传输率 100MB/s）；
- 与 MMC（eMMC）的读写数据位宽可配置为 1bit/4bit/8bit；与 SD/SDIO 的读写数据位宽可配置为 1bit/4bit；
- 集成 DMA 引擎。

2 eMMC 接口总体结构

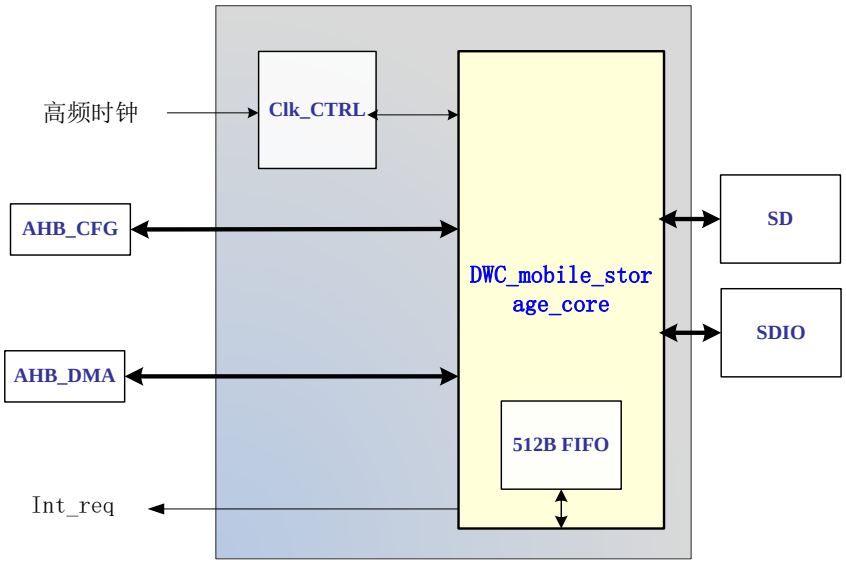


图 2-1 eMMC 接口结构示意图

其中 Clk_CTRL 用于时钟生成和相位控制。

3 硬件接口

3.1 接口信号

表 3-1 eMMC 接口部件的接口

信号名	位宽	来源	说明
时钟复位信号			
clk_2x	1	输入	外部输入的高频时钟 注意：要求 clk_2x 分频产生的时钟<=10*clk
clk	1	输入	AHB 时钟；
reset_n	1	输入	系统复位信号，低有效，与时钟 clk 同步。 注意：复位时，以频率较低者为准，至少保持两个 clk_2x 分频后的时钟或 clk 时钟周期有效。
AHB Slave 接口信号			
hsel	1	输入	Slave 选择信号
hready	1	输入	AHB 数据总线准备好（输入）
haddr	[31:0]	输入	地址。
hwrite	1	输入	AHB 读写选择信号。
htrans	[1:0]	输入	传输类型。(IDLE/BUSY/NONSEQ/SEQ)
hsize	[2:0]	输入	传输大小。**只支持 1B/2B/4B 三种类型** ● 000—8bits ● 001—16bits ● 010—32bits
hburst	[2:0]	输入	突发类型。（**无意义**）
hwdata	[31:0]	输入	AHB 写数据
hready_resp	1	输出	AHB 数据总线准备好（输出）
hresp	[1:0]	输出	响应类型。注意：只会产生 OKAY 响应。
hrdata	[31:0]	输出	AHB 读数据

AHB Master 接口（内部 DMA）			
m_hreq	1	输出	AHB 总线请求信号。
m_hgrant	1	输入	总线授权信号。 为 1 表示当前 Master 获得总线的最高优先级。当 m_hgrant 和 m_hready 同时为 1 时，当前 Master 才真正获得总线的使用权。
m_haddr	[31:0]	输出	传输地址
m_htrans	[1:0]	输出	AHB 传输类型。支持以下类型： ● 00—IDLE ● 10—NONSEQ ● 11—SEQ
m_hwrite	1	输出	指示 AHB 读/写传输
m_hsize	[2:0]	输出	传输大小。一定给出 3’b010（表示 32bit）
m_hburst	[2:0]	输出	突发类型。 ● 000—SINGLE ● 001—INCR ● 010—WRAP4（不会出现） ● 011—INCR4 ● 100—WRAP8（不会出现） ● 101—INCR8 ● 110—WRAP16（不会出现） 111—INCR16
m_hwdata	[31:0]	输出	写数据
m_hready	1	输入	传输完成指示信号。
m_hresp	[1:0]	输入	传输回答类型。 ● 00—OKAY，表示传输被正常处理，当 m_hready 同时有效时，表示该传输成功结束； ● 01—ERROR，表示发生传输错误；

			● 10—RETRY，传输无法立即完成，Master 需要重试； 11—SPLIT，传输过程被 Slave 分离，Master 需要重试。
m_hrdata	[63:0]	输入	读数据。
与中断相关信号			
int	1	输出	中断
与 Card 相关信号			
cclk_out	[1:0]	输出	内部时钟分频器产生，Card 时钟
ccmd_in	[1:0]	输入	Card 命令（根据时钟 cclk_in_sample 采样）
ccmd_out	[1:0]	输出	Card 命令（根据时钟 cclk_in_drv 输出）
ccmd_out_en	[1:0]	输出	Card 命令使能位（根据时钟 cclk_in_drv 输出）
cdata_in	[15:0]	输入	Card 数据
cdata_out	[15:0]	输出	Card 数据
cdata_out_en	[15:0]	输出	Card 数据使能位
card_detect_n	[1:0]	输入	Card detect 信号，0 表示 Card 存在。
card_write_prt	[1:0]	输入	Card 写保护信号，1 表示对应的 Card 写保护有效
card_power_en	[1:0]	输出	Card 的电源选择信号； 0-power off，1-power on。
card_volt_a	[3:0]	输出	电压适配器 A 的选择信号
card_volt_b	[3:0]	输出	电压适配器 B 的选择信号
ccmd_od_pullup_en_n	1	输出	命令通路 open-drain/pull-up 使能位
biu_volt_reg	[1:0]	输出	SD3.0 中，外置电压适配器的电压选择信号； 0-3.3v/1-1.8v； 对应寄存器 UHS[1:0]：VOLT_REG
card_int_n	[1:0]	输入	Card 中断信号，仅对 eSDIO 有含义。
back_end_power	[1:0]	输出	Corresponds to MMC_VOLT_REG. nd is used incombination with biu_volt_reg port to decode the requiredvoltage.

rst_n	[1:0]	输出	硬件复位信号，低有效。
biu_volt_reg_1_2	1	输出	Back-end power supply for embedded device.

3.2 芯片引脚说明

表 3-2 eMMC 接口引脚信号

信号名称	位宽	IO	描述	引脚类型*
CLK	1* NUM_CARD_BUS	输出	时钟信号	
RST	1* NUM_CARD_BUS	输出	复位信号，低有效	
CMD	1* NUM_CARD_BUS	双向	命令/响应通道	
DAT	8* NUM_CARD_BUS	双向	输入输出数据	
card_detect_n	1* NUM_CARD_BUS	输入		
card_write_prt	1* NUM_CARD_BUS	输入		
od_pullup_en_n	1	输出		
card_volt_a	4	输出		
card_volt_b	4	输出		
card_power_en	1* NUM_CARD_BUS	输出		
biu_volt_reg	1* NUM_CARD_BUS	输出	SD3.0	
biu_volt_reg_1_2	1* NUM_CARD_BUS	输出	eMMC4.5	
rst_n	1* NUM_CARD_BUS	输出	eMMC4.4	
card_int_n	1* NUM_CARD_BUS	输入	SDIO3.0	
back_end_power	1* NUM_CARD_BUS	输出	SDIO3.0	

引脚 CMD 与 DAT 都通过三态 Buffer 与 IP 例化时的内部接口信号相连。它们之间的互连关系如下图所示。

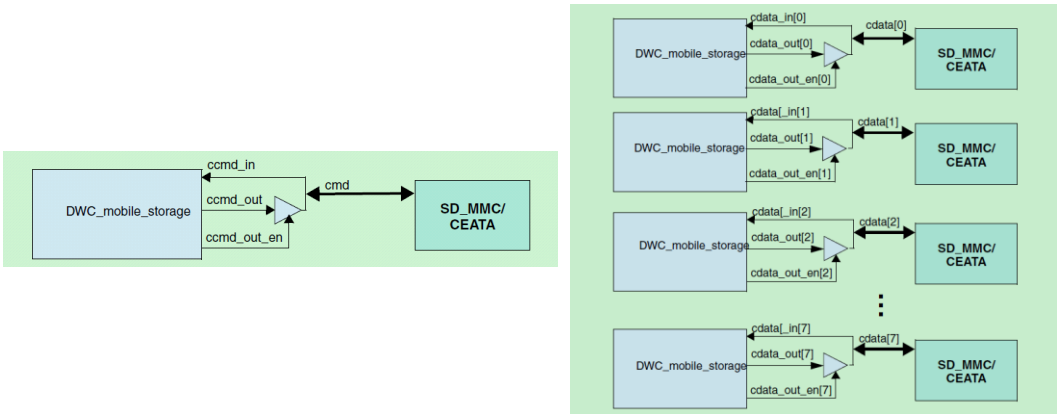


图 3-1 引脚 CMD 与 DAT 通过三态 Buffer 与 IP 内部接口信号相连

4 DWC_mobile_storage_core

4.1 基本机构

下图给出了 DWC_mobile_storage_core 的基本结构。各模块具体功能如下：

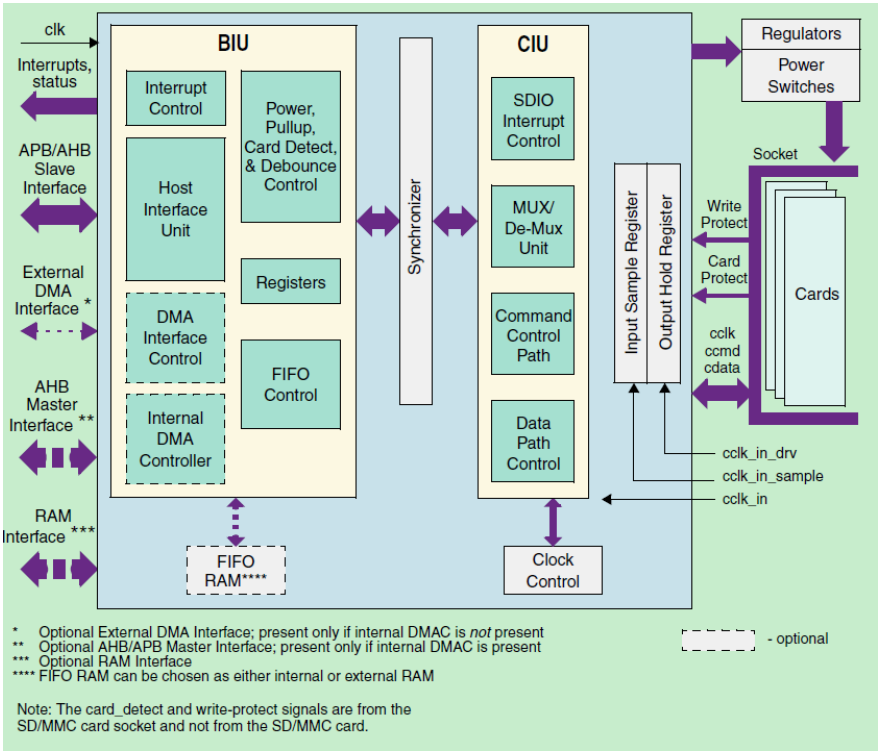


图 4-1DWC_mobile_storage_core 结构示意图

- BIU（Bus Interface Unit）：AHB Slave/Master 接口，工作在 AHB 时钟域。具体包括如下模块：
  - Host Interface Unit：AHB Slave 接口；AHB-Slave 接口不支持 Split、Retry 以及产生 AHB 错误响应；
  - Register File：软件可见的寄存器；
  - 内置 DMA 引擎：单引擎单通道，基于描述符的机制；
  - 中断控制部件；
  - FIFO 控制器；FIFO 宽度为 32 位，深度为 128 条目；
  - 上拉电阻控制
- CIU（Card Interface Unit）：与 SD/SDIO 的接口部分，工作频率可配置为外部高频时钟的 2/4/6/8 分频。CIU 具体包括如下模块：

- 命令通路控制模块
- 数据通路控制模块
- 输入采样寄存器
- 输出保持寄存器
- 时钟控制：分频控制输出给 SD/SDIO 的时钟

IP 的配置参数请见附录 1。

4.2 基本工作原理

总线操作由命令、数据块和响应组成。总线操作一般都包含命令和响应，涉及到数据传输的操作带有数据块。使用 DWC_mobile_storage_core 的基本流程如下：

- 软件通过写寄存器 CMD 与寄存器 CMDARG 告诉 DWC_mobile_storage_core 需要发送给 SD/SDIO 的具体命令信息，DWC_mobile_storage_core 会自动将寄存器 CMD 与寄存器 CMDARG 中的信息转化为引脚 CMD 上的串行报文发给 SD/SDIO；下图给出了寄存器 CMD 到命令报文的转换示意；

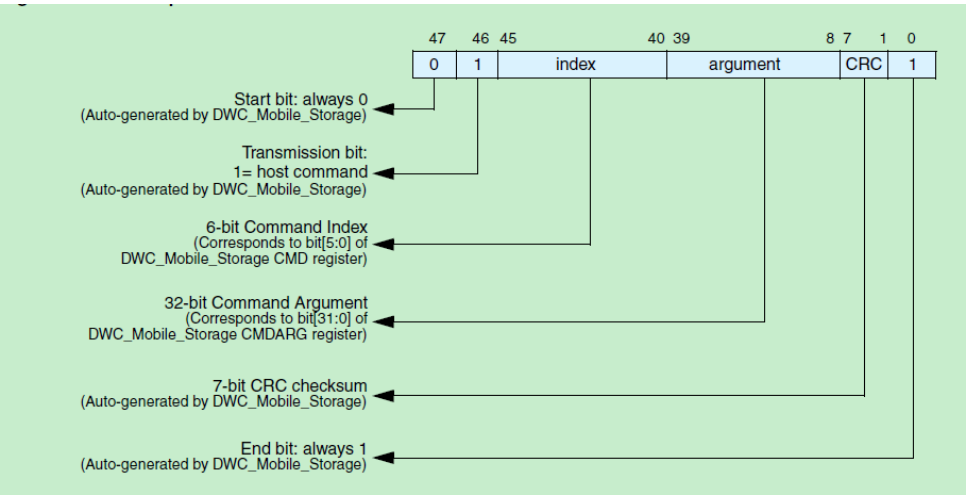


图 4-2 命令格式

- 如果涉及到数据传输
  - 在非 DMA 模式下，软件可往/从 DWC_mobile_storage_core 中的 FIFO 中写入/读出要传输的数据（FIFO 映射为寄存器空间），当软件启动发送/接收命令时，DWC_mobile_storage_core 会自动将传输的数据从 FIFO 中读出/写入，通过引脚 DAT0-DAT3 发送给 SD/SDIO 或者从 SD/SDIO 处接收；
  - 在 DMA 模式下，当软件启动发送/接收命令时，DWC_mobile_ storage_core 内

置的 DMA 引擎会从/往主存中读出/写入要传输的数据并写入/读出到 FIFO 中，同时 CIU 部分会自动将传输的数据从 FIFO 中读出/写入，通过引脚 DAT0-DAT3 发送给 SD/SDIO 或者从 SD/SDIO 处接收；

- SD/SDIO 返回响应，DWC_mobile_storage_core 会自动解析响应，并将响应中有用信息保存入寄存器 RESP0~3 待软件查询；下图给出了一种响应报文到寄存器 RESP0 转换的示意。

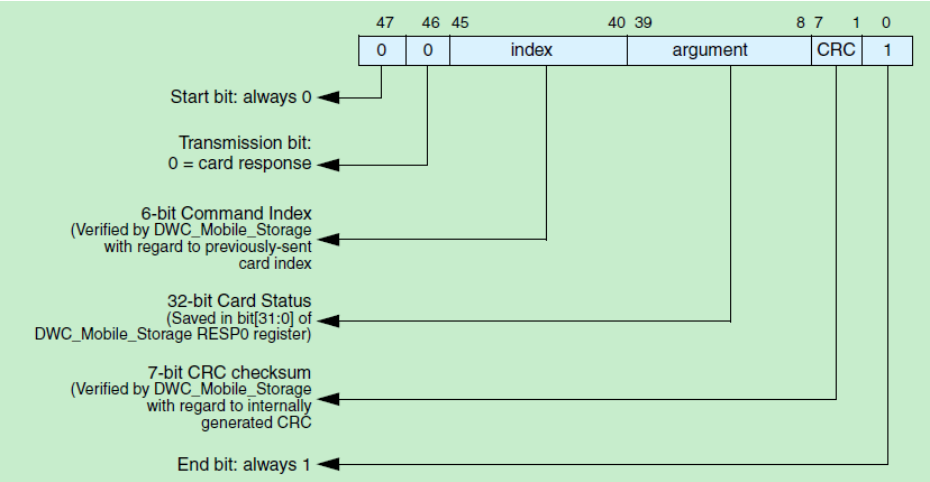


图 4-3 响应格式之一

5 时钟生成模块

5.1 概述

DWC_mobile_storage_core 需要如下表所示的 5 个时钟。

表 5-1 eMMC 时钟列表

Clock Name	Input/Output	Edge Used Within DWC_mobile_storage
clk	Input	Rising edge
cclk_in	Input	Rising and falling edges
cclk_in_drv	Input	Rising and falling edges
cclk_in_sample	Input	Rising and falling edges
cclk_out	Output	–

- clk 为 AHB 总线工作时钟，DWC_mobile_storage_core 中的 BIU（BUS Interface Unit）以及相关配置寄存器工作在该时钟域；
- cclk_in 为 DWC_mobile_storage_core 中 CIU（CARD Interface Unit）的工作时钟；可由 clk_2x 分频得到，分频数可以为 2、4、6、8；
- cclk_in_drv 为 DWC_mobile_storage_core 中 CMD/DATA 引脚的输出驱动时钟；可以配置 cclk_in_drv 相对 cclk_in 的相位偏移；相位偏移的最小粒度为 1 个 clk_2x 的周期。
- cclk_in_sample 为 DWC_mobile_storage_core 中 CMD/DATA 引脚的输入采样时钟；可以配置 cclk_in_sample 相对 cclk_in 的相位偏移；相位偏移的最小粒度为 1 个 clk_2x 的周期。
- cclk_out 作为 DWC_mobile_storage_core 中 CLK 引脚的输出。cclk_out 可以是 cclk_in 的同频时钟或者是 cclk_in 的分频时钟。在 DWC_mobile_storage_core 的内部实现对 cclk_in 的分频。cclk_out 相对于 cclk_in 的偏移=cclk_out 的分频选择门控延迟+IP 内线延迟+Pad 延迟。

其中，cclk_in_drv、cclk_in_sample 与时钟 cclk_in 的同频同源，但需要独立产生相对 cclk_in 不同相位的偏移，并支持软件选择相位偏移。

5.2 复位结束后状态

UHS_REG_EXT[31:30]复位值设为 0，GPIO[8]复位值设为 0,，即默认时钟关闭，分频数为 2。

复位结束后，时钟生成模块将按照 clk_2x 的 2 分频产生 cclk_in、cclk_in_drv、cclk_in_sample，且所有时钟相位偏移为 0。2 个 cclk_in 时钟周期产生后，DWC_mobile_storage_core 的 CIU 部分的复位撤销。

5.3 时钟切换流程

这里时钟切换是指 cclk_in 的工作频率选择以及 cclk_in_drv、cclk_in_sample 与时钟 cclk_in 之间的相位偏移选择。需要软件利用 DWC_mobile_storage_core 的寄存器 GPIO 以及寄存器 UHS_REG_EXT 实现时钟切换。

当 CIU 部分已经工作，需要切换 cclk_in 的工作频率或者 cclk_in_drv、cclk_in_sample 的相位偏移时，需要做如下配置：

- 关闭发送给 card 的时钟
  - 软件通过读寄存器 status[data_busy]为零来确保接口不处于数据传输状态；
  - 软件写寄存器 CLKENA[1:0]为 0（用于关闭发送给 card 的时钟），然后写寄存器 CMD[start_cmd]为 1，CMD[update_clock_registers_only]为 1（仅用于更新 CIU 部分的时钟控制信息），并且 CMD[wait_previous_data_complete]为 1，然后反复读寄存器 CMD，直至寄存器 CMD[start_cmd]被硬件清 0。
- 关闭 CIU 的工作时钟
  - 软件写寄存器 GPIO[8]为 0；
  - 时钟生成模块通过打三拍挤脉冲的方式，获知需要关闭时钟，则在各个时钟下降沿分别关闭 cclk_in、cclk_in_drv 与 cclk_in_sample。在此期间，电平信号 CLK_READY 仍旧为 1；
  - 所有时钟关闭完成后，时钟生成模块将电平信号 CLK_READY 置 0；CLK_READY 会异步交接保存入寄存器 GPIO[0]，软件可查询寄存器 GPIO[0]直至为 0 时，表示原有时钟已经完全关闭；
- 软件将新的 cclk_in 的工作频率选择信息以及 cclk_in_drv、cclk_in_sample 的相

位偏移选择信息写入寄存器 UHS_REG_EXT；然后软件将寄存器 GPIO[8]置为 1；

- 寄存器 UHS_REG_EXT 信息会异步交接到时钟生成模块；
- 时钟生成模块开始生成时钟 cclk_in、cclk_in_drv、cclk_in_sample，同时将电平信号 CLK_READY 置 1；
- CLK_READY 会异步交接保存入寄存器 GPIO[0]，软件需要查询寄存器 GPIO[0]直至为 1，表示 CIU 部分可以开始工作。

5.4 时钟生成模块的实现

模块工作频率： clk_2x；

输入：CLK_ENABLE（GPIO[8]）、UHS_REG_EXT、clk_2x、rst_n；

输出：cclk_in、cclk_in_drv、cclk_in_sample、CLK_READY。

5.4.1 异步交接

- CLK_ENABLE：时钟生成模块用 clk_2x 打三拍挤脉冲表示时钟使能或者关闭。当时钟使能有效时，将 UHS_REG_EXT 信息保存至模块内部寄存器，并开始分频产生时钟；当时钟关闭有效时，则在各个时钟下降沿逐步关闭 cclk_in、cclk_in_drv、cclk_in_sample；
- CLK_READY：为电平信号。当时钟 cclk_in、cclk_in_drv 与 cclk_in_sample 开始产生时，CLK_READY 置为 1；当所有时钟都彻底关闭时，将 CLK_READY 置为 0。DWC_mobile_storage_core 会将该信号打 2 拍后保存入寄存器 GPIO[0]。

5.4.2 状态机

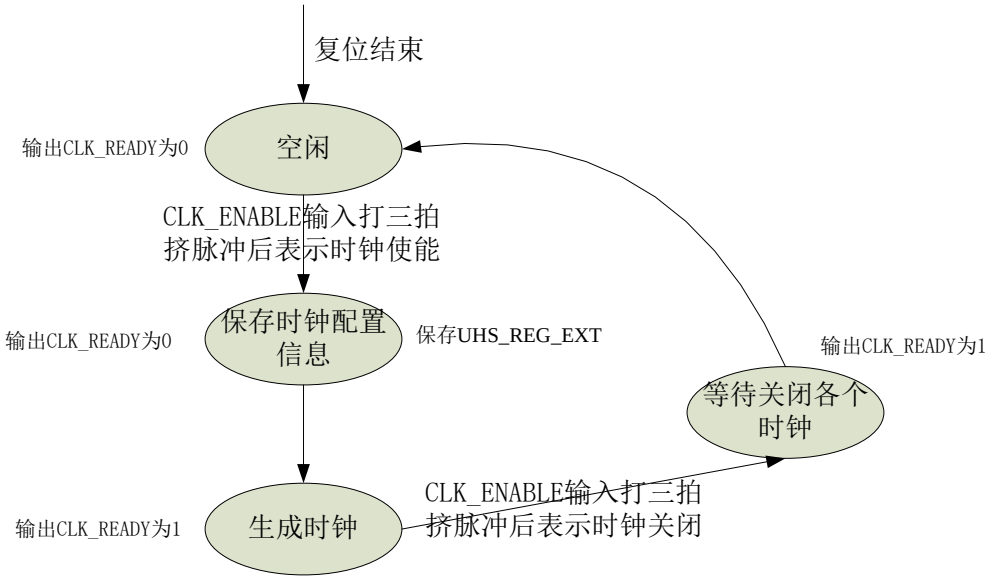


图 5-1 时钟控制模块状态机

- 每个时钟必须在下降沿发生以后才关闭（保证时钟的高电平宽度不受影响）；
- cclk_in、cclk_in_drv、cclk_in_sample 的时钟均为触发器输出。

5.4.3 时钟分频实现

- 采用计数器实现时钟 clk_2x 的 2、4、6、8 分频作为 cclk_in；
- 将分频后的时钟信号 cclk_in 用时钟 clk_2x 拍拍寄存实现用于相位的偏移。

6 IOR 寄存器

IOR 寄存器作为软硬件接口，对 IP 的操作流程和底层驱动而言至关重要。

6.1 寄存器编址

- 所有寄存器逻辑上都是 32 位，按字节编址。
- FIFO 空间也编入寄存器地址。

表 6-1 内部寄存器编址

	寄存器名	地址	宽度	描述
内部寄存器	CTRL	0x00	32	控制寄存器
	PWREN	0x04	32	电源使能寄存器
	CLKDIV	0x08	32	时钟分频寄存器
	CLKSRC	0x0c	32	时钟源寄存器
	CLKENA	0x10	32	时钟使能寄存器
	TMOUT	0x14	32	超时寄存器
	CTYPE	0x18	32	存储卡类型寄存器
	BLKSIZ	0x1c	32	块大小寄存器
	BYTCNT	0x20	32	字节计数寄存器
	INTMASK	0x24	32	中断屏蔽寄存器
	CMDARG	0x28	32	命令参数寄存器
	CMD	0x2c	32	命令寄存器
	RESP0	0x30	32	响应寄存器 0
	RESP1	0x34	32	响应寄存器 1
	RESP2	0x38	32	响应寄存器 2
	RESP3	0x3c	32	响应寄存器 3
	MINTSTS	0x40	32	屏蔽中断状态寄存器
	RINTSTS	0x44	32	中断向量寄存器

	STATUS	0x48	32	状态寄存器
	FIFOTH	0x4c	32	FIFO 临界值
	CDETECT	0x50	32	存储卡检测寄存器
	WRTPRT	0x54	32	写保护寄存器
	GPIO	0x58	32	通用 IO 寄存器
	TCBCNT	0x5c	32	CIU 传输字节计数寄存器
	TBBCNT	0x60	32	BIU 传输字节计数寄存器
	DEBNCE	0x64	32	存储卡边界检测寄存器
	USRID	0x68	32	用户 ID 寄存器
	VERID	0x6c	32	synonsys 版本寄存器
	HCON	0x70	32	硬件配置寄存器
	UHS_REG	0x74	32	UHS-1 寄存器
	RST_n	0x78	32	硬件复位寄存器
IDMAC 相关寄存器	BMOD	0x80	32	总线模式寄存器
	PLDMND	0x84	32	Poll Demand Register
	DBADDR	0x88	32	描述符基地址寄存器
	IDSTS	0x8C	32	IDMAC 状态寄存器
	IDINTEN	0x90	32	IDMAC 内部中断寄存器
	DSCADDR	0x94	32	当前描述符地址寄存器
	BUFADDR	0x98	32	当前数据缓冲地址寄存器
	Reserved	0x9C~ 0xFF	—	—
	CardThrCtl	0x100	32	存储卡读边界寄存器
	Back_end_power	0x104	16	Back-end Power
	UHS_REG_EXT	0x108	32	eMMC 4.5 1.2V register
	EMMC_DDR_REG	0x10C	32	eMMC DDR START bit detection control register
	ENABLE_SHIFT	0x110	32	Phase shift control register

数据 FIFO 部分	DATA	>= 0x200		

6.2 寄存器定义

6.2.1 CTRL

索引地址：0x00

控制寄存器。

表 6-2CTRL 定义

名称	范围	类型	描述
Reserved	[31:26]	—	—
use_internal_dmac	25	RW,1’h0	只有使用IDMAC时该位才有意义。 ● 0—使用Slave接口进行数据传输 ● 1—使用IDMAC进行数据传输
enable_OD_pullup	24	RW, 1’h1	open-drain开关； ● 0—关闭 ● 1—打开 当打开open-drain模式时，命令总线的输出总是0或者高阻态。
Card_voltage_b	[23:20]	RW,4’h0	电压适配端口B的设置； 输出到端口card_volt_b。
Card_voltage_a	[19:16]	RW,4’h0	电压适配端口A的设置； 输出到端口card_volt_a。
Reserved	[15:12]	—	—
ceata_device_interrupt_status	11	RW,1’h0	CE_ATA设备的中断使能位； ● 0—中断使能关闭 ● 1—中断使能打开

send_auto_stop_ccsd	10	RW,1’h0	为1表示在发送CCSD（Command Completion Stop Disable）命令后，自动发送一个STOP命令。 注意：①该位的设置应该与send_ccsd位的设置同步；②发送CCSD之后，DWC_mobile_storage会自动清除该位。
send_ccsd	9	RW,1’h0	为1表示自动向CE_ATA设备发送CCSD命令。 注意：①当等待CCS命令超时时，且CE_ATA设备允许中断，软件置该位为1；②发送CCSD命令之后，DWC_mobile_storage自动清除该位；③如果Command Done中断没有被屏蔽，那么发送完CCSD命令之后，DWC_mobile_storage会自动设置中断寄存器RINTSTS中的CD位；③当置该位为1时，需要两拍的时间发送命令CCSD，因此有可能发生如下情况：在向设备发送CCSD期间，设备产生CCS信号。
abort_read_data	8	RW,1’h0	中止读数据。在读数据期间，发送中止命令后，复位读数据状态机。当状态机进入IDLE时，清除该位。
send_irq_response	7	RW,1’h0	自动发送IRQ响应，并跳出状态—wait for interrupt state。 为了等待来自MMC的中断请求，主机发送CMD40后，DWC_mobile_storage状态机进入相应的状态（waiting for interrupt state）。当处于上述状态时，如果置该位为1，DWC_mobile_storage

			会跳出该状态，返回对应响应并进入IDLE状态。
read_wait	6	RW,1’h0	为1表示插入read wait状态
dma_enable	5	RW,1’h0	只有使用外置DMA接口时才有意义。 表示是否使用DMA模式。
int_enable	4	RW,1’h0	总的中断使能位
Reserved	3	—	—
dma_reset	2	RW,1’h0	IDMAC复位信号，两个AHB时钟周期之后自动清零。
fifo_reset	1	RW,1’h0	复位数据FIFO（通过复位指针完成）。 复位完成后改为自动清零。
controller_reset	0	RW,1’h0	DWC_mobile_storage控制逻辑的复位信号，两个AHB时钟周期+两个cclk_in时钟周期之后自动清零。 注意：被复位的逻辑包括BIU/CIU接口、CIU及相关状态机、控制寄存器部分位（abort_read_data、send_irq_response、read_wait）、命令寄存器的部分位（start_cmd）

6.2.2 PWREN

索引地址：0x04

电源开关寄存器。

表 6-3 PWREN 定义

名称	范围	类型	描述
Reserved	[31:30]	—	—
power_enable	[29: 0]	RW,30’h0	1表示电源打开。对应的输出端口为card_power_en。

			bit[0]对应card 0； bit[1]对应card 1； ……

6.2.3 CLKDIV

索引地址：0x08

时钟分频寄存器。

表 6-4 CLKDIV 定义

名称	范围	类型	描述
clk_divider3	[31:24]	RW,8’h0	无意义
clk_divider2	[23:16]	RW,8’h0	无意义
clk_divider1	[15: 8]	RW,8’h0	时钟分频器1的分频数，2*clk_divider1。
clk_divider0	[7: 0]	RW,8’h0	时钟分频器0的分频数，2*clk_divider0。

6.2.4 CLKSRC

索引地址：0x0c。

时钟源寄存器。

表 6-5 CLKSRC 定义

名称	范围	类型	描述
clk_source	[31: 0]	RW,32’h0	● bit[1:0]表示Card0的时钟由哪个分频器产生，时钟并通过cclk[0]输出到card0； ● bit[3:2]表示Card1的时钟由哪个分频器产生，时钟并通过cclk[1]输出到card1； ● …… ● bit[31:30]表示Card15的时钟由哪个分频器产生，时钟并通过cclk[15]输出到card15。

6.2.5 CLKENA

索引地址：0x10。

时钟使能寄存器。

表 6-6CLKENA 定义

名称	范围	类型	描述
cclk_low_power	[31:16]	RW,16’h0	表示是否允许相应的Card进入低功耗模式。 对于SD和MMC Card，当相应Card进入IDLE时，可以关闭相应时钟；对于SDIO Card，如果需要检测中断，则可以不关闭时钟。
clk_enable	[15: 0]	RW,16’h0	对应16个card的时钟使能位；

6.2.6 TMOUT

索引地址：0x14。

超时寄存器。

表 6-7TMOUT 定义

名称	范围	类型	描述
data_timeout	[31:8]	RW,0xffffff	读存储卡数据超时值，也可以用于主机返回数据超时。 注意：①当存储卡时钟停止后开始计时，超过data_timeout个cclk_out个时钟周期后视为超时；②软件的超时器以100ms为单位，如果使用软件超时器，需要屏蔽掉存储卡读数据引起的超时中断。
response_timeout	[7:0]	RW,0x40	响应超时值。 经过response_timeout个clk_out周期后仍然没有返回，则认为响应超时。

6.2.7 CTYPE

索引地址：0x18。

存储卡模式寄存器。

表 6-8CTYPE 定义

名称	范围	类型	描述
card_width	[31:16]	RW,16’h0	指示存储卡是否是8bit模式 ● Bit[31]为1表示Card15是8bit模式，为0表示Card15不是8bit模式； ● Bit[30]为1表示Card14是8bit模式，为0表示Card14不是8bit模式； ● …… ● Bit[16]为1表示Card0是8bit模式，为0表示Card0不是8bit模式； 注意：如果某个存储卡是8bit模式，则对应表示其是1bit/4bit的位将失去意义。
card_width	[15: 0]	RW,16’h0	指示存储卡是1bit模式还是4bit模式。 ● Bit[15]为1表示Card15是4bit模式，为0表示Card15是1bit模式； ● Bit[14]为1表示Card14是4bit模式，为0表示Card14是1bit模式； ● …… ● Bit[0]为1表示Card0是4bit模式，为0表示Card0是1bit模式；

6.2.8 BLKSIZE

索引地址：0x1c。

数据块大小寄存器。

表 6-9BLKSIZE 定义

名称	范围	类型	描述
Reserved	[31:16]	—	—
block_size	[15: 0]	RW,16’h200	数据块大小

6.2.9 BYTCNT

索引地址：0x20。

字节计数寄存器。

表 6-10BYTCNT 定义

名称	范围	类型	描述
byte_count	[31:0]	RW,31’h200	需要传输的字节数。 注意：①一定是Block_size的整数倍；②当byte_count为0时，表示需要传输的字节数不确定，Host需要显示的发送stop/abort命令。

6.2.10 INTMASK

索引地址：0x24。

中断屏蔽寄存器。

表 6-11 INTMASK 定义

名称	范围	类型	描述
sdio_int_mask	[31:16]	RW, 16’h0	16位表示相应的16个SDIO卡是否允许中断，1表示允许，0表示禁止。
int_mask	[15: 0]	RW, 16’h0	为0表示屏蔽相应中断，为1表示允许相应中断。 ● Bit[15]—结束位错误（读）/写数据没有CRC校验 ● Bit[14]—自动命令停止（ACD） ● Bit[13]—命令开始位错误（SBE）/Busy

			Complete中断（BCI） ● Bit[12]—寄存器发生锁写错误（HLE） ● Bit[11]—FIFO上溢/下溢（FRUN） ● Bit[10]—主机返回数据超时（HTO）/电压转换 ● Bit[9]—数据读取超时（DRTO） ● Bit[8]—响应超时（RTO） ● Bit[7]—数据CRC校验错（DCRC） ● Bit[6]—响应CRC校验错（RCRC） ● Bit[5]—接收FIFO数据请求（RXDR） ● Bit[4]—发送FIFO数据请求（TXDR） ● Bit[3]—数据传输结束（DTO） ● Bit[2]—命令完成（CD） ● Bit[1]—响应错误（RE） ● Bit[0]—检测到存储卡（CD）

6.2.11 CMDARG

索引地址：0x28。

命令参数寄存器。

表 6-12CMDARG 定义

名称	范围	类型	描述
CMDARG	[31:0]	RW,32’h0	发送给Card的命令参数

6.2.12 CMD

索引地址：0x2c。

命令寄存器CMD和命令参数寄存器CMDARG记录发向Card的信息。通过置寄存器CMD[31]：start_cmd为1启动命令传输，硬件负责自动将start_cmd清零，在此之前，不可以对相关寄存器进行写操作，主要包括如下寄存器：CMD、CMDARG、BYTCNT、BLKSIZ、CLKDIV、CLKENA、CLKSRC、TMOUT及CTYPE。硬件可以保证在start_cmd为1期间无法成功写上述相关寄存器，

同时产生“硬件锁(hardware lock)”错误，并根据中断使能情况决定是否触发相应的中断。

表 6-13CMD 定义

名称	范围	类型	描述
Start_cmd	[31]	RW, 1’h0	启动命令标志。软件写该位为1，会指示CIU根据寄存器CMD和寄存器CMDARG发送命令给card。一旦命令被CIU接收，硬件会自动将该位清0。 在该位为1期间，不能对CMD相关的寄存器进行写操作。
RSV	[30]	RSV	RSV
Use_hold_reg	[29]	RW, 1’h0	为1时，命令和数据的发送用clk_in_drive以保证信号的保持时间
volt_switch	[28]	RW, 1’h0	当使用CMD11进行电压调节时，该位需要为1
boot_mode	[27]	RW, 1’h0	Boot模式。 0：Mandatory Boot operation; 1：Alternate Boot operation
disable_boot	[26]	RW, 1’h0	当设置该位时，CIU将会终止当前的boot操作
expect_boot_ack	[25]	RW, 1’h0	Expect Boot Acknowledge
enable_boot	[24]	RW, 1’h0	只有当boot_mode是Mandatory才设置该位为1。 注意：该位不能和disable_boot同时设置为1
ccs_expected	[23]	RW, 1’h0	0：Interrupts are not enabled in CE-ATA device (nIEN = 1 inATA control register), or command does not expect CCSfrom device 1：Interrupts are enabled in CE-ATA device (nIEN = 0), andRW_BLK command expects command completion signalfrom CE-ATA device
read_ceata_devic	[22]	RW,	0：Host is not performing read access (RW_REG

e		1’h0	or RW_BLK)towards CE-ATA device 1：Host is performing read access (RW_REG or RW_BLK)towards CE-ATA device
Update_clock_registers_only	[21]	RW, 1’h0	为1时表示不发送命令，仅仅用于更新CIU部分的时钟相关信息：CLKDIV、CLKSRC、CLKENA。
Card_number	[20:16]	RW, 5’h0	当前操作的Card Number
Send_initialization	[15]	RW, 1’h0	0：发送命令前不发送初始化序列 1：发送命令前发送初始化序列（80个周期1） card上电后第一个命令该位必须设置为1
Stop_abort_cmd	[14]	RW, 1’h0	0：表示该命令不是stop命令 1：表示该命令是stop命令，将终止正在进行的传输
wait_prvdata_complete	[13]	RW, 1’h0	0：表示该命令马上被发送，即使前一个命令还没有完成 1：等待前一个命令完成才发送下一个命令
send_auto_stop	[12]	RW, 1’h0	1：传输结束后，自动发送stop命令
transfer_mode	[11]	RW, 1’h0	0：块式传输 1：流式传输
Read/write	[10]	RW, 1’h0	0：表示读Card 1：表示写Card
Data_expected	[9]	RW, 1’h0	0：该命令不涉及读写数据传输操作 1：该命令涉及读写数据传输操作
check_response_crc	[8]	RW, 1’h0	1表示需要对该命令收到的响应进行CRC校验
response_ length	[7]	RW, 1’h0	0：该命令对应的响应为短响应 1：该命令对应的响应为长响应
response_expect	[6]	RW,	0：该命令发送后没有响应接收

		1’h0	1：该命令发送后有响应接收
Cmd_index	[5:0]	RW, 6’h0	命令索引号

6.2.13 RESP0

索引地址：0x30。

响应寄存器 RESP0 记录 Card 发送的响应[31：0]。

表 6-14RESP0 定义

名称	范围	类型	描述
RESP0	[31:0]	RO,32’h 0	记录Card发送的响应[31：0]。短响应只记录在 REP0中。

6.2.14 RESP1

索引地址：0x34。

响应寄存器 RESP1 记录 Card 发送的响应[63：32]。

表 6-15RESP1 定义

名称	范围	类型	描述
RESP1	[31:0]	RO,32’h0	记录Card发送的长响应[63：32] Auto_stop_cmd的短响应也记录在RESP1

6.2.15 RESP2

索引地址：0x38。

响应寄存器 RESP2 记录 Card 芯片发送的响应[95：64]。

表 6-16RESP2 定义

名称	范围	类型	描述
RESP2	[31:0]	RO,32’h0	记录Card芯片发送的长响应[95：64]

6.2.16 RESP3

索引地址：0x3c。

响应寄存器 RESP3 记录 Card 发送的响应[127：96]。

表 6-17RESP3 定义

名称	范围	类型	描述
RESP3	[31:0]	RO,32’h0	记录Card发送的长响应[127：96]

6.2.17 MINTSTS

索引地址：0x40。

中断状态寄存器，只读，是寄存器 RIINTSTS 与寄存器 INTMASK 按位“与”的结果。

表 6-18MINTSTS 定义

名称	范围	类型	描述
sdio_interrupt	[31:16]	RO,16’h0	对应SDIO卡中断的情况
int_status	[15: 0]	RO,16’h0	对应各种中断的情况

6.2.18 RINTSTS

索引地址：0x44。

中断状态寄存器，表示发生中断的情况，没有看中断屏蔽使能向量。

表 6-19RINISTS 定义

名称	范围	类型	描述
sdio_interrupt	[31:16]	RW,16’h0	Interrupt from SDIO card; one bit for each card.
EBE	[15]	RW,1h0	End-bit error /write no CRC
ACD	[14]	RW,1h0	Auto command done
SBE/BCI	[13]	RW,1h0	Start-bit error/Busy Clear Interrupt
HLE	[12]	RW,1h0	Hardware locked write error 在硬件对某些寄存器“锁”期间，对相关寄存器

			进行写操作。
FRUN	[11]	RW,1h0	FIFO underrun/overrun error
HTO	[10]	RW,1h0	Data starvation-by-host timeout / Volt_switch_int
DRTO/BDS	[9]	RW,1h0	Data read timeout /Boot Data Start
RTO/BAR	[8]	RW,1h0	Response timeout/Boot Ack Received
DCRD	[7]	RW,1h0	Data CRC error
RCRC	[6]	RW,1h0	Response CRC error
RXDR	[5]	RW,1h0	Receive FIFO data request **注意：在DMA模式下，需要屏蔽该中断。**
TXDR	[4]	RW,1h0	Transmit FIFO data request **注意：在DMA模式下，需要屏蔽该中断。**
DTO	[3]	RW,1h0	Data transfer over
CD	[2]	RW,1h0	Command done
RE	[1]	RW,1h0	Response error
CDT	[0]	RW,1h0	Card detect

6.2.19 STATUS

索引地址：0x48。

状态寄存器，只读。

表 6-20STATUS 定义

名称	范围	类型	描述
dma_req	[31]	RO,1'h0	DMA等待响应的状态
dma_ack	[30]	RO,1'h0	DMA收到响应的状态
fifo_count	[29:17]	RO,1'h0	FIFO计数，表示FIFO中有效条目数
response_index	[16:11]	RO,1'h0	上一个命令的索引
data_state_mc_busy	[10]	RO,1'h1	表示数据发送/接收状态机是否忙
data_busy	[9]	RO,cdata_in	信号card_data[0]的取反，表示相应的存储卡

			是否忙。 0-对应Card不是busy； 1-表示对应Card是busy。
data_3_status	[8]	RO,cdata_in	来自信号card_data[3]，表示相应的Card是否存在。 0-对应card不存在； 1-对应card存在
command status states	[7: 4]	RO,4’h0	命令状态机。 ● 0—空闲状态 ● 1—发送顺序序列 ● 2—发送命令的开始位 ● 3—发送命令的tx位 ● 4—发送命令的索引位和参数位 ● 5—发送命令的crc校验码 ● 6—发送命令的结束位 ● 7—接收响应的开始位 ● 8—接收IRQ响应 ● 9—接收响应的tx位 ● 10—接收响应的索引位 ● 11—接收响应的数据部分 ● 12—接收响应的crc校验码 ● 13—接收响应的结束位 ● 14—等待NCC ● 15—等待状态 注意：命令状态机有19个状态，寄存器STATUS没有记录以下三个状态：Wait For CCS、Send CCSD及Boot Mode。当处于三个装填时，STATUS[7:4]为0。
fifo_full	[3]	RO,1’h0	数据FIFO满

fifo_empty	[2]	RO,1’h1	数据FIFO空
fifo_tx_watermark	[1]	RO,1’h1	数据FIFO中发送的数据超过了临界值
fifo_rx_watermark	[0]	RO,1’h0	数据FIFO中接收的数据超过了临界值

6.2.20 FIFOTH

索引地址：0x4c。

FIFO 临界值寄存器。

表 6-21FIFOTH 定义

名称	范围	类型	描述
Reserved	[31]	—	—
MSIZE	[30:28]	RW, 3’h0	读/写事务包含传输的个数
RX_Wmark	[27:16]	RW,FIFO_DEPTH-1	当数据从主存发向FIFO时，如果FIFO中的数据等于或者大于该值，将会产生一个DMA/FIFO请求；如果允许中断的话，还将产生中断。
Reserved	[15:12]	—	—
TX_Wmark	[11:0]	RW,12’h0	当数据从FIFO发向Card时，如果FIFO中的数据等于或者小于该值，将会产生一个DMA/FIFO请求；如果允许中断的话，还将产生中断。

6.2.21 CDETECT

索引地址：0x50。

存储卡检测寄存器，只读。

表 6-22CDETECT 定义

名称	范围	类型	描述
Reserved	[31:30]	—	—
card_detect_n	[29:0]	RO , card_detect_n	表示是否检测到相应的Card

		inputs	

6.2.22 WRTPRT

索引地址：0x54。

写保护寄存器。

表 6-23WRTPRT 定义

名称	范围	类型	描述
Reserved	[31:30]	—	—
write_protect	[29:0]	RO， card_write_prt inputs	表示相应的Card是否进行写保护

6.2.23 GPIO

索引地址：0x58。

GPIO 是 DWC_mobile_storage_core 为二次开发所预留的。该寄存器的输入以及输出都与 DWC_mobile_storage_core 模块的输入/输出端口相连。

表 6-24GPIO 定义

名称	范围	类型	描述
RSV	[31:24]	-	保留
GPO	[23:9]	RW,15’h0	反应到输出gp_out上
CLK_ENABLE	[8]	RW,1’h1	输出给时钟生成模块。1表示时钟生成模块开始工作，0表示时钟生成模块停止工作。
GPI	[7:1]	RO	保留
CLK_READY	[0]	RO	来自时钟生成模块，DWC_mobile_storage_core将输入信号打2拍后保存入该寄存器

6.2.24 TCBCNT

索引地址：0x5c。

CIU 发送到 Card 的字节数寄存器，只读。

表 6-25TCBCNT 定义

名称	范围	类型	描述
trans_card_byte_count	[31:0]	RO,32’h0	表示CIU发送到Card的字节数

6.2.25 TBBCNT

索引地址：0x60。

主机和 BIU 之间传输字节计数寄存器，只读。

表 6-26TBBCNT 定义

名称	范围	类型	描述
trans_fifo_byte_count	[31:0]	RO,32’h0	表述主机和FIFO之间传输的字节数

6.2.26 DEBNCE

索引地址：0x64。

边界计数寄存器。

表 6-27DEBNCE 定义

名称	范围	类型	描述
Reserved	[31:24]	—	—
debounce_count	[23:0]	RW,24’hff_ffff	边界过滤逻辑使用的临界值，单位是clk时钟周期

6.2.27 USRID

索引地址：0x68。

用户 ID 寄存器。

表 6-28USRID 定义

名称	范围	类型	描述

USRID	[31:0]	RW，可配置	用户可以在综合前配置

6.2.28 VERID

索引地址：0x6c。

版本 ID 寄存器，只读。

表 6-29VERID 定义

名称	范围	类型	描述
VERID	[31:0]	RO,32'h5342_270a	Synopsys版本号

6.2.29 HCON

索引地址：0x70。

硬件配置参数寄存器，只读。

表 6-30HCON 定义

名称	范围	类型	描述
Reserved	[31:27]	RO	—
AREA_OPTIMIZED	[26]	RO	是否进行面积优化
NUM_CLK_DIVIDER	[25:24]	RO	时钟分频器的个数为 NUM_CLK_DIVIDER +1
SET_CLK_FALSE_PATH	[23]	RO	是否将CLK设置位false path
IMPLEMENT_HOLD_REG	[22]	RO	是否实现保持寄存器
FIFO_RAM_INSIDE	[21]	RO	FIFO阵列实现位置 ● 0—IP外部 ● 1—IP内部
GE_DMA_DATA_PATH	[20:18]	RO	通用DMA数据总线宽度 ● 000—16bits ● 001—32bits ● 010—64bits ● 其他—保留

DMA_INTERFACE	[17:16]	RO	DMA接口 ● 00—无DMA接口 ● 01—DW_DMA接口 ● 10—通用DMA接口 ● 11—NON-DW-DMA接口
H_ADDR_WIDTH	[15:10]	RO	地址总线宽度 ● 0~7 — 保留 ● 8 — 9bits ● 9 — 10bits ● …… ● 31 — 32bits ● 32~63 — 保留
H_DATA_WIDTH	[9:7]	RO	数据总线宽度 ● 000—16bits ● 001—32bits ● 010—64bits ● 其他—保留
H_BUS_TYPE	[6]	RO	总线类型 ● 0—APB ● 1—AHB
NUM_CARDS	[5:1]	RO	存储卡个数为NUM_CARDS+1
CARD_TYPE	[0]	RO	存储卡类型 ● 0—MMC-ONLY ● 1—SD_MMC

6.2.30 UHS_REG

索引地址：0x74。

UHS-1 寄存器。

表 6-31UHS_REG 定义

名称	范围	类型	描述
DDR_REG	[31:16]	RW,16’h0	表示相应Card是否采用DDR模式
VOLT_REG	[15: 0]	RW,16’h0	表示相应Card缓冲对应的电压 ● 0—3.3v ● 1—1.8v

6.2.31 RST_n

索引地址：0x78。

复位寄存器。

表 6-32RST_n 定义

名称	范围	类型	描述
Reserved	[31:16]	—	—
CARD_RESET	[15:0]	RW,15’h1	为0时复位相应的Card

6.2.32 BMOD

索引地址：0x80。

DMA 参数配置寄存器。

表 6-33BMOD 定义

名称	范围	类型	描述
Reserved	[31:11]	—	—
PBL	[10:8]	RO,3’h0	只读，与寄存器FIFOTH中MSIZE的相同，一次读/写事务最多的传输个数
DE	[7]	RW,1’h0	是否使用IDMAC
DSL	[6:2]	RW,5’h0	对双数据缓冲的描述符有意义。表示两个描述符之间的间隔，单位是地址总线的宽度。
FB	[1]	RW,1’h0	是否使用固定长度的传输。 当该位为1时，AHB Master只能发起如下类型的传输：SINGLE/INCR4/INCR8/INCR16。当该位为0时，

			AHB Master只能发起SINGLE/INCR类型的传输。
SWR	[0]	RW,1’h0	软件复位。 当该寄存器写1时，DMA引擎复位所有内部寄存器； 1拍之后该寄存器自动清零。

6.2.33 PLDMND

索引地址：0x84。

Pull Demand 寄存器，只写。在进行 DMA 传输时，若 DMA 描述符没有准备好（DESO[31]：OWN 为 0），则 IDMA 的状态机会进入 Suspend 状态；当 DMA 描述符准备好后（DESO[31]：OWN 为 1），Host 需要向该寄存器写入一个任意值，使 IDMAC 的状态机跳出 Suspend 状态。

表 6-34PLDMND 定义

名称	范围	类型	描述
PD	[31:0]	WO,32’h0	Pull demand寄存器

6.2.34 DBADDR

索引地址：0x88。

描述符基址寄存器。

表 6-35DBADDR 定义

名称	范围	类型	描述
SDL	[31:0]	RW,32’h0	描述符基址

6.2.35 IDSTS

索引地址：0x8c。

DMAC 状态寄存器。

表 6-36IDSTS

名称	范围	类型	描述
Reserved	[31:17]	—	—

FSM	[16:13]	RO,4’h0	DMAC的状态机。 ● 0—DMA_IDLE ● 1—DMA_SUSPEND ● 2—DESC_RD ● 3—DESC_CHK ● 4—DMA_RD_REQ_WAIT ● 5—DMA_WR_REQ_WAIT ● 6—DMA_RD ● 7—DMA_WR ● 8—DESC_CLOSE
EB	[12:10]	RO,3’h0	总线错误类型。 ● 001—在数据发送期间，主机停止接收； ● 010—在数据接收期间，主机停止接收 ● 其他—保留
AIS	[9]	RW,1’h0	非正常中断。 包括总线错误导致的中断和描述符不可用导致的中断。
NIS	[8]	RW,1’h0	正常中断。 包括发送中断和接收中断。 写1清零。
Reserved	[7:6]	—	—
CES	[5]	RW,1’h0	存储卡错误。主要包括以下类型的错误： ● EBE—结束位错误 ● RTO—响应超时/Boot响应超时 ● RCRC—响应CRC校验错 ● SBE—开始位错误 ● DRTO—读数据超时/BDS超时 ● DCRC—数据CRC校验错 ● RE—响应错

			写1清零。
DU	[4]	RW,1’h0	描述符不可用，表示描述符由于OWN位为0导致的不可用。 写1清零。
Reserved	[3]	—	—
FBE	[2]	RW,1’h0	总线错误中断，为1时表示发生了总线错误，DMA负责停止所有的总线访问。 写1清零。
RI	[1]	RW,1’h0	接收中断，表示某个描述符的数据接收完毕。写1清零。 写1清零。
TI	[0]	RW,1’h0	发送中断，表示某个描述符的数据发送完毕。写1清零。 写1清零。

6.2.36 IDINTEN

索引地址：0x90。

IDMAC 中断使能寄存器。

表 6-37IDINTEN 定义

名称	范围	类型	描述
Reserved	[31:10]	—	—
AI	[9]	RW,1’h0	非正常中断使能位
NI	[8]	RW,1’h0	正常中断使能位
Reserved	[7:6]	—	—
CES	[5]	RW,1’h0	存储卡错误中断使能位
DU	[4]	RW,1’h0	描述符不可用中断使能位
Reserved	[3]	—	—
FBE	[2]	RW,1’h0	总线错误中断使能位

RI	[1]	RW,1’h0	接收中断使能位
TI	[0]	RW,1’h0	发送中断使能位

6.2.37 DSCADDR

索引地址：0x94。

当前主机描述符寄存器。

表 6-38DSCADDR 定义

名称	范围	类型	描述
HDA	[31:0]	RO,32’h0	当前描述符的起始起始地址

6.2.38 BUFADDR

索引地址：0x98。

当前数据缓冲地址寄存器。

表 6-39BUFADDR 定义

名称	范围	类型	描述
HBA	[31:0]	RO,32’h0	当前数据缓冲的地址

6.2.39 CardThrCtl

索引地址：0x100。

存储卡读临界值控制寄存器。

表 6-40CardThrCtl 定义

名称	范围	类型	描述
Reserved	[31:28]	—	—
CardRd hreshold	[N:16]	RW,1’h0	存储卡读临界值，其中N与FIFO的大小有关： ● N = 25，FIFO Size = 128 ● N = 24，FIFO Size = 64 ● N = 25，FIFO Size = 32

			● N = 23, FIFO Size = 8
Reserved	[15:2]	—	—
BsyClrIntEn	[1]	RW,1’h0	Busy Clear中断使能位，
CardRd ThrEn	[0]	RW,1’h0	Card读临界值使能位。为1表示使能，此时只有数据FIFO中剩余的空间大于临界值时，Host才会向MMC Card发送读命令。

6.2.40 Back_end_power

索引地址：0x100。

Back_end_power 寄存器。

表 6-41Back_end_power 定义

名称	范围	类型	描述
Reserved	[31:16]	—	—
Back End Power	[15: 0]	RW,16’h0	Back_end_power使能位，每一位对应一个存储卡

6.2.41 UHS_REG_EXT

索引地址：0x108。

UHS Register

表 6-42UHS_REG_EXT 定义

名称	范围	类型	描述
EXT_CLK_MUX_CTRL	[31:30]	RW,2’hf	cclk_in相对于clk_2x的分频计数器: 0表示2分频 1表示4分频 2表示6分频 3表示8分频
CLK_DRV_PHASE_CTRL	[29:23]	RW,7’h0	cclk_in_drv与时钟cclk_in之间的相位偏移选择:

			6’h0_0：0° 相位偏移 6’h0_1：1个clk_2x时钟周期的偏移 6’h0_2：2个clk_2x时钟周期的偏移 6’h0_3：3个clk_2x时钟周期的偏移 ……
CLK_SMPL_ PHASE_CTRL	[22:16]	RW,7’h0	cclk_in_sample与时钟cclk_in之间的相位偏移选择: 6’h0_0：0° 相位偏移 6’h0_1：1个clk_2x时钟周期的偏移 6’h0_2：2个clk_2x时钟周期的偏移 6’h0_3：3个clk_2x时钟周期的偏移 ……
MMC_VOLT_ REG	[15:0]	RW,16’h0	映射到输出biu_volt_reg_1_2[15:0]

6.2.42 EMMC_DDR_REG

索引地址：0x10c。

开始位检测控制寄存器。

表 6-43EMMC_DDR_REG 定义

名称	范围	类型	描述
Reserved	[31:16]	—	—
HALF_START_BIT	[15:0]	RW,16’h0	为1时表示开始位可以不保持一个时钟周期。

6.2.43 ENABLE_SHIFT

索引地址：0x110。

相位偏移使能寄存器。

表 6-44ENABLE_SHIFT 定义

名称	范围	类型	描述
Enable_shift	[31: 0]	RW,32’ h0	每两位对应一个card，其中Bit[1:0]对应card0，Bit[3:2]对应card1，……，Bit[31:30]对应card15。含义如下： ● 00—默认相位偏移 ● 01—相位偏移在下一个时钟上升沿 ● 10—相位偏移在下一个时钟下降沿 ● 11—保留

7 软件流程

7.1 对软件的约束

1) 同一时刻只允许对一个 Card 进行数据传输，当进行数据传输时，只能接收查询状态的命令以及 stop 命令，而不能接收新的数据传输命令。因此要进行读写操作时，必须确保上一次读写操作已经结束。
2) 当要切换 card 的工作时钟频率时，必须保证此时没有数据、命令/响应传输正在进行。
3) 如果工作在 SDR 高速模式、DDR 高速模式下，cclk_in_drive 相对于 cclk_in 需要有相位偏移以满足信号的 hold time 要求，即 use_hold_reg 必须为 1。
4) 在打开中断全局使能之前，建议软件对中断状态寄存器写 32'hffff_ffff，清除残留的中断状态。
5) FIFO 的使用要求：
   a) 即使软件要写入/读取 card 的数据长度不为 4B 的整数倍，host/DMA 引擎写入/读取 FIFO 的数据长度必须满足 4B 的整数倍；
   b) 如果采用非 DMA 模式读取 card，当 DTO（Data transfer Over）中断发生时，表明数据传输结束，但此时仍有可能有部分数据留存在 FIFO 中，软件要负责读完；
   c) 当上一次传输非法结束时，应该写寄存器 CTRL[fifo_reset] 复位 FIFO，清空所有残留数据。
6) 寄存器 FIFOTH[DW_DMA_Mutiple_Transaction_Size]必须设为 16 或者以下
7) 在关闭 CIU 时钟关闭的情况下（GPIO[CLK_ENABLE]=0），无法对 CIU 进行复位。

7.2 时钟分频器的设置

这里指的时钟分频器是 IP 里的 CIU 内部的时钟分频器。配置 CIU 内部的时钟分频器的步骤如下：

1) 通过读寄存器 status[data_busy]来确保接口不处于数据传输状态

2) 写寄存器 CLKENA[0]为 0

3) 将寄存器 CLKENA[0]的值更新到 CIU 时钟域，更新之前需要确保之前发送所有命令都已经处理完毕。

   a) 写 I 寄存器 CMD[start_cmd]为 1，CMD[update_clock_registers_only]为 1，CMD[wait_previous_data_complete]为 1，然后反复读寄存器 CMD，直至寄存器 CMD[start_cmd]被硬件清 0，此时表示之前发送所有命令都已经处理完毕后，才正式关闭时钟。

4) 更新分频计数器寄存器 CLKDIV 等时钟相关寄存器，然后通过写 CMD[wait_previous_data_complete] 为 1，寄存器 CMD[start_cmd] 为 1，CMD[update_clock_registers_only]为 1，将这些值更新到 CIU 时钟域。软件通过反复读寄存器 CMD，直至寄存器 CMD[start_cmd]被硬件清 0 表示时钟更新完毕。

5) 打开时钟，写寄存器 CLKENA[0]为 1，通过写 CMD[wait_previous_data_ complete]为 1，寄存器 CMD[start_cmd]为 1，CMD[update_clock_registers_ only]为 1，将这些值更新到 CIU 时钟域。寄存器 CMD[start_cmd]被硬件清 0 表示时钟打开完毕。

7.3 时钟设置

软件首先通过写寄存器 UHS_REG_EXT 配置时钟 cclk_in 的分频数，以及时钟 cclk_in_sample 和 cclk_in_drv 的相位偏移；然后通过向寄存器 GPIO[8]写 1（默认为零），打开时钟工作使能。当寄存器 GPIO[0]为 1 时，表示时钟设置完成。

7.4 初始化过程

硬件需要确保 Rst_n 信号至少要有效 2 个 cclk/cclk_in 周期。Rst_n 信号撤销后，软件进行初始化流程如下：

1) 写 0xffff 至寄存器 RINTSTS，清除所有不干净的中断信号；

2) 设置中断屏蔽位寄存器 INTMASK，打开全局中断使能寄存器 CTRL[int_enable]；

3) 将 card 工作时钟频率设置为 400KHz 以下，比如 cclk_in 的时钟为 50MHz，50,000/(2*400)=62.5，则设寄存器 CLKDIV[clk_divider0]=63 ,打开寄存器 CLKENA 中的对应位（详见 7.2）；

4) 判断 Card 的类型：①如果是 SD 卡跳至“5)”，②如果是 SDIO（无 Memory），跳

至“6）”，③如果是 SDIO（有 Memory，即 SD_COMBO），跳至“7）”：

a) 发送 CMD5，参数为 32’h0；

b) 读响应寄存器 RESP0，将{8’hFF,RESP0[23:0]}作为参数，再次发送 CMD5；

c) ①若响应超时，说明是 SD 卡，跳转至“5）”。②若未超时，读响应寄存器 RESP。若 RESP0[27]为 1，说明是带 Memory 的 SDIO 卡，跳转至“7）”；③若未超时且若 RESP0[27]为 0，说明是不带 Memory 的 SDIO 卡，跳转至“6）”。

5) SD 卡的枚举过程：

a) 发送 CMD0（寄存器 CMD 的值为{{16'h8000 +cardnum, 16'h0000}}，寄存器 CMDARG 的值为 32’h0）；

b) 发送 CMD8（寄存器 CMD 的值为{{16'h8000 +cardnum, 16'h0148}}，寄存器 CMDARG 的值为 32’h1aa）；

c) 发送 ACMD41，其中 CMD55（寄存器 CMD 的值为{{16'h8000 +cardnum, 16'h0177}}，寄存器 CMDARG 的值为 32’h0），CMD41（寄存器 CMD 的值为{{16'h8000 +cardnum, 16'h0169}}，寄存器 CMDARG 的值为 4'hf,3'b0,S18R_bit,24'hff_fff0），其中对于 SD_MEM 3.0， S18R_bit 为 1，其含义是 Supports voltage switching for 1.8V。

d) 读 ACMD41 的响应，①若 RESP0[24]为 1，且对应 SD 卡是 SD_MEM v3.0，则进行电压转换，发送 CMD11（寄存器 CMD 的值为{16'h9000+cardNum, 16'h014b}，寄存器 CMDARG 的值为 32’h0）；②否则转至下一步；

e) 发送 CMD2（寄存器 CMD 的值为{16'h8000+cardNum, 16'h01c2}，寄存器 CMDARG 的值为 32’h0）；

f) 发送 CMD3（寄存器 CMD 的值为{16'h8000+cardNum, 16'h0143}，寄存器 CMDARG 的值为 32’h0）

g) 读 CMD3 的响应，其中 RESP0[31:16]即是 SD 卡的 RCA。

6) SDIO（无 Memory）的枚举过程：

a) 发送 CMD5，参数为 32’h0；

b) 读响应寄存器 RESP0，将{8’hFF,RESP0[23:0]}作为参数，再次发送 CMD5；

c) 发送 CMD3（寄存器 CMD 的值为{16'h8000+cardNum, 16'h0143}，寄存器 CMDARG 的值为 32’h0001_0000）

d) 读 CMD3 的响应，其中 RESP0[31:16]即是 SD 卡的 RCA。

7) SDIO（有 Memory）的枚举过程：
   a) 发送 CMD5，参数为 32’h0；
   b) 读响应寄存器 RESP0，将{8’hFF,RESP0[23:0]}作为参数，再次发送 CMD5；
   c) 发送 CMD8（寄存器 CMD 的值为{{16'h8000 +cardnum, 16'h0148}}，寄存器 CMDARG 的值为 32’h1aa）；
   d) 发送 ACMD41，其中 CMD55（寄存器 CMD 的值为{{16'h8000 +cardnum, 16'h0177}}，寄存器 CMDARG 的值为 32’h0），CMD41（寄存器 CMD 的值为{{16'h8000 +cardnum, 16'h0169}}，寄存器 CMDARG 的值为 4'hf,3'b0,S18R_bit,24'hff_fff0），其中对于 SD_MEM 3.0， S18R_bit 为 1，其含义是 Supports voltage switching for 1.8V。
   e) 读 ACMD41 的响应，①若 RESP0[24]为 1，且对应 SD 卡是 SD_MEM v3.0，则进行电压转换，发送 CMD11（寄存器 CMD 的值为{16'h9000+cardNum, 16'h014b}，寄存器 CMDARG 的值为 32’h0）；②否则转至下一步；
   f) 发送 CMD2（寄存器 CMD 的值为{16'h8000+cardNum, 16'h01c2}，寄存器 CMDARG 的值为 32’h0）；
   g) 发送 CMD3（寄存器 CMD 的值为{16'h8000+cardNum, 16'h0143}，寄存器 CMDARG 的值为 32’h0）
   h) 读 CMD3 的响应，其中 RESP0[31:16]即是 SD 卡的 RCA。
8) 设备枚举结束后，通过修改寄存器 CLKDIV 和寄存器 CLKENA 将 card 工作时钟切换至至 50MHz。
9) 设置其他基本参数
   a) 寄存器 TMOUT[response_timeout]=0x64
   b) 寄存器 TMOUT[data_timeout]=max{ (10 * ((TAAC*Fop) + (100*NSAC)), Host FIFO read/write latency from FIFO empty/full }
   c) FIFOTH[RX_WMark] = (FIFO_DEPTH/2) – 1 = 63;
   d) FIFOTH[TX_WMark] = FIFO_DEPTH/2 = 64

7.5 设置数据总线的宽度

7.5.1 SD 卡和 SDIO（有 Memory）

首先，通过CMD7选择对应的Card，其寄存器CMD的值为{16'h8000+cardNum,16'h0047}，寄存器 CMDARG 的值为{rca,16'h0}；

其次，通过 ACMD6 来切换 Card 的模式，其中 CMD55（寄存器 CMD 的值为{16'h8000+cardNum,16'h0077}，寄存器 CMDARG 的值为{rca,16'h0}），CMD6（寄存器 CMD 的值为{16'h8000+cardNum,16'h0046}，寄存器 CMDARG 的值为{24'h0,dataBusWidth}）；

最后，通过写寄存器 CTYPE 来配置 Host 段的位宽。

7.5.2 SDIO（无 Memory）

首先，通过CMD7选择对应的Card，其寄存器CMD的值为{16'h8000+cardNum,16'h0047}，寄存器 CMDARG 的值为{rca,16'h0}；

其次，通过 CMD52 来切换 Card 的模式，其中，寄存器 CMD 的值为{16'h8000+cardNum,16'h0574}，寄存器 CMDARG 的值为{24'h8000_0e,datBusWidth}。

7.6 模式切换

7.6.1 SD 卡和 SDIO（有 Memory）

使用 CMD6 进行模式切换，其中寄存器 CMD 的值为{16'h8000+cardNum,16'h0346}，寄存器 CMDARG 的值为{28'h80ffff0, Mode}。

其中 Mode 的含义如下：

- 为 0 表示 SDR12；
- 为 1 表示 SDR25；
- 为 2 表示 SDR50；
- 为 3 表示 SDR104；
- 为 4 表示 DDR50；
- 为 5 表示 HS200Mode。

对于 DDR 传输传输，还要配置寄存器 UHS_REG 中对应的位为 1。

7.6.2 SDIO（无 Memory）

使用 CMD52 进行模式切换，其中寄存器 CMD 的值为{16'h8000+cardNum,16'h0574}，寄存器 CMDARG 的值为{ 1'b1,fnNum,RAW,1'b0,reg_addr,1'b0, 4'h0, Mode[2:0],1'b1}。

其中，Mode 的含义同上。

对于 DDR 传输传输，还要配置寄存器 UHS_REG 中对应的位为 1。

7.7 设置 BlockLength 和 BlockCount

7.7.1 SD 卡和 SDIO（有 Memory）

首先，向寄存器 BLKSIZE 和 BYTCNT 中分别写入 Block 的大小和要传输的字节数；

其次，通过 CMD16 配置 Card 端 BlockLen；其中，寄存器 CMD 的值为{16'h8000+cardNum,16'h0050}，寄存器 CMDARG 的值为 blkLength。

最后，通过 CMD23 配置 BlockCount；其中，寄存器 CMD 的值为{16'h8000+cardNum,16'h0057}，寄存器 CMDARG 的值为 BlkCnt。

7.7.2 SDIO（无 Memory）

首先，向寄存器 BLKSIZE 和 BYTCNT 中分别写入 Block 的大小和要传输的字节数；

其次，发送 CMD52，其中，寄存器 CMD 的值为{16'h8000+cardNum,16'h0574}，寄存器 CMDARG 的值为{1'b1,fnNum,RAW,1'b0,{6'h0,fnNum,8'h10},1'b0,blkLength[7:0]};

最后，发送 CMD52，其中，寄存器 CMD 的值为{16'h8000+cardNum,16'h0574}，寄存器 CMDARG 的值为{ 1'b1,fnNum,RAW,1'b0,{6'h0,fnNum,8'h11},1'b0,blkLength[15:8]}。

7.8 非数据传输命令的发送

1) 软件通过写寄存器 CMD 和寄存器 CMDARG，触发硬件向 card 发送命令。寄存器 CMD 的编程如下表所示。

参数名	设置值	含义

start_cmd	1	表示命令正在发送，CIU 接收命令后，清零该位。
use_hold_reg	0/1	CIU 向存储卡发送 CMD/DATA 时是否数据 HOLD 寄存器
Update_clk_regs_only	0	为 1 表示该命令只更新 CLK 相关寄存器
data_expected	0	为 1 表示该命令带数据传输
card_number	CardNo	存储卡的实际编号
cmd_index	command index	相应命令的索引值
send_initialization	0	只有命令 CMD0 才将该域置 1
stop_abort_cmd	1	表示该命令是 STOP 命令
response_length	0	为 1 表示该命令对应长响应
response_expect	1	为 1 表示该命令期望收到响应
wait_prvdata_complete	1	为 1 表示只有之前的数据传输完成以后，才会发送该命令
check_response_crc	1	表示 host 收到响应之后是否进行 CRC 校验

2) 硬件收到 card 返回的响应，会登记寄存器 RINTSTS[command_done]，同时把响应记录在寄存器 RES0~3 中，相关的错误也记录在寄存器 RINTSTS 中。软件可通过读寄存器 RINTSTS 检查是否发生响应超时错误、响应 CRC 错误、响应错误。

3) 注意：在主机控制器 DWC_mobile_storage 中有个单条目的数据缓冲，当命令缓冲中已经有一个未发送的命令时，置寄存器 CMD[31]：start_cmd 为 1，或者写相关的寄存器都会产生 HLE 错误。

7.9 数据传输命令的发送

1) 发送命令前，软件应首先通过发送 CMD13（SEND_STATUS）检查 card 是否正忙，并通过发送 CMD7（SELECT/DESELECT_C）来确保 card 进入传输 transfer 状态；

2) SD 卡和 SDIO（with Memory）使用如下命令进行数据传输：

命令	含义	寄存器设置
CMD17	Read Single Block	CMD：{16'h8000+cardNum,16'h0351} CMDARG：读地址
CMD18	Read Multiple Block	CMD：{16'h8000+cardNum,16'h0352} CMDARG：读地址
CMD24	Write Single Block	CMD：{16'h8000+cardNum, 16'h0758} CMDARG：写地址
CMD25	Write Multiple Block	CMD：{16'h8000+cardNum,16'h0759} CMDARG：写地址

3) SDIO（no Memory）使用如下命令进行数据传输：

读写均是用 CMD53，参数不同含义不同，具体如下：

命令	寄存器设置	含义
CMD53	CMD：{16'h8000+cardNum,16'h0375} CMDARG：{**1'b0**,fnNum, blockMode,1'b1,reg_addr,bytCnt}	Read Single Block
CMD53	CMD：{16'h8000+cardNum,16'h0375} CMDARG：{**1'b0**,fnNum, blockMode,1'b1,reg_addr,byteCount/blkLength}	Read Multiple Block
CMD53	CMD：{16'h8000+cardNum,16'h0375} CMDARG：{**1'b1**,fnNum, blockMode,1'b1,reg_addr,bytCnt}	Write Single Block
CMD53	CMD：{16'h8000+cardNum,16'h0375} CMDARG：{**1'b1**,fnNum, blockMode,1'b1,reg_addr,byteCount/blkLength}	Write Multiple Block

4) 数据传输完毕检查寄存器 RINTSTS 中各种错误状态。

说明：数据传输可能产生的中断如下：

- DTO 中断：数据传输结束或者被中止；
- TXDR 中断：当向存储卡写数据时，数据 FIFO 中的数据低于临界值；
- RXDR 中断：当从存储卡读数据时，数据 FIFO 中的数据高于临界值；

- HTO 中断：当向存储卡写数据时，数据 FIFO 为空；或者从存储卡中读数据时，数据 FIFO 满；
- DRTO 中断：当从存储卡中读数据时，数据返回超时；
- DCRC 中断：存储卡返回的数据存在 CRC 校验错；
- SBE 中断：从存储卡中读数据时，没有收到 Start-Bit；
- EBE 中断：读/写存储卡时，没有收到 End-Bit；
- BCI 中断：数据传输后，存储卡驱动的 Busy 信号结束。

5) 当产生 DTO 中断时，软件需要把数据 FIFO 中残留的数据读出。

7.9.1 Card Read Threshold

当 Round Trip Delay 大于 0.5cclk_in 的时候，寄存器 CardThrCtl [CardRdThrEnable]必须使能。需要满足的条件是：

- 选择 Block Size：Block Size 必须是 128B 的整数倍，且小于 FIFO 容量；
- RX_WMark 必须是 128B 的整数倍；
- CardRdThreshold 为 512B。

7.9.2 错误处理

1) 响应/数据超时：对于响应超时，软件可以重试相关命令；对于数据超时，软件可以重新传输整个数据块，也可以只传输没有收到响应的数据块。
2) 响应错：软件可以重试出错的命令。
3) 数据错：发生数据错以后，软件可以发送 STOP/ABORT 命令，并重试整个数据块或者是出错和剩下的数据块。
4) HLE 错：软件需要重新配置出错命令的相关参数。
5) FIFO 上溢/下溢：在读/写数据 FIFO 之前时，软件需要先读寄存器 STATUS 中的相应位，确定 FIFO 的空满；
6) 主机超时导致数据“饥饿”：当 DWC_mobile_storage 等待软件对数据 FIFO 读/写期间，在规定的时间内（Timeout Period）,软件没有发起请求，则认为发生了 Data Starvation Interrupt。
7) 命令 CRC 校验错：当发生命令 CRC 校验错时，存储卡不会返回对应的响应，进而

会导致响应超时中断发生。

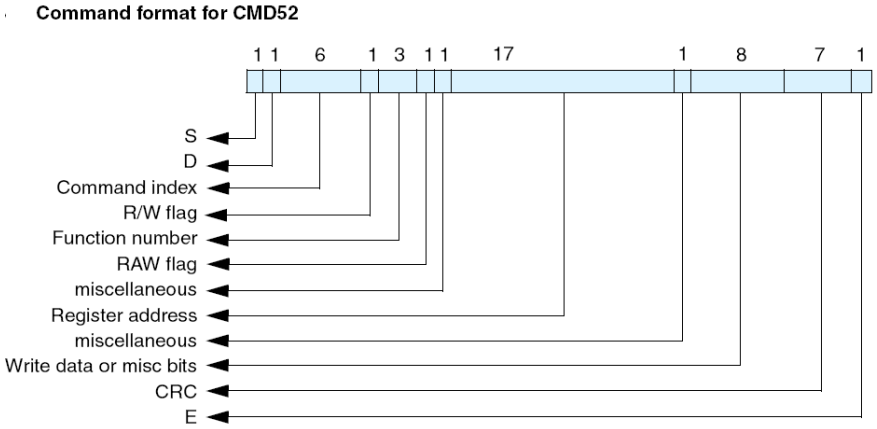
7.10 Stop 命令

7.10.1 SD 卡和 SDIO（有 Memory）

使用命令 CMD12；其中，寄存器 CMD 的值为{16'h8000+cardNum,16'h4c4c}，寄存器 CMDARG 的值为 32'h0。

7.10.2 SDIO（无 Memory）

使用命令 CMD52；其中，寄存器 CMD 的值为{ 16'h8000+cardNum,16'h4174}，寄存器 CMDARG 的值为 32'h80000c00。



CMDARG Bits	Contents	Value
31	R/W flag	1
30-28	Function Number	0, for CCCR access
27	RAW flag	1, if needed to read after write
26	Don't care	–
25-9	Register address	0x06
8	Don't care	–
7-0	Write Data	Function number to be aborted

7.11 DMA 引擎

eMMC 接口的 DMA 引擎基于描述符机制实现，本节内容对实现数据传输至关重要。

7.11.1 描述符

描述符用于保存 DMA 引擎要访问的数据缓冲的相关信息，描述符也位于主存中，**其存放地址必须 64 位对界。**第一个描述符的地址记录在寄存器 DBADDR（描述符基址寄存器）中。具体有如下 2 种描述符组织方式：

1) 双缓冲方式：每个描述符记录了连续 2 个要访问的数据缓冲的相关信息，描述符在主存中的分布为连续或者固定步长。步长记录在寄存器 BMOD[DSL]中；

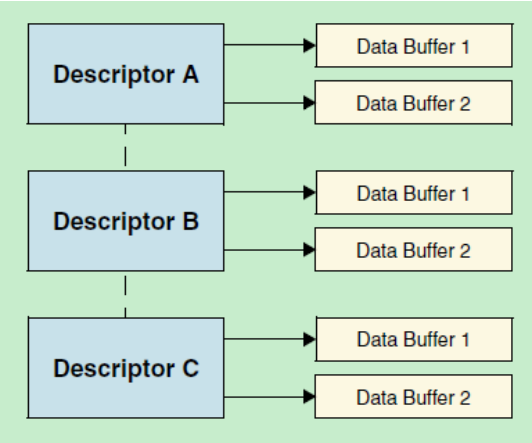


图 7-1DMA 描述符—双缓冲方式

2) 链表方式：每个描述符记录了 1 个数据缓冲相关信息，并包含指针指向下一个要访问的描述符地址。

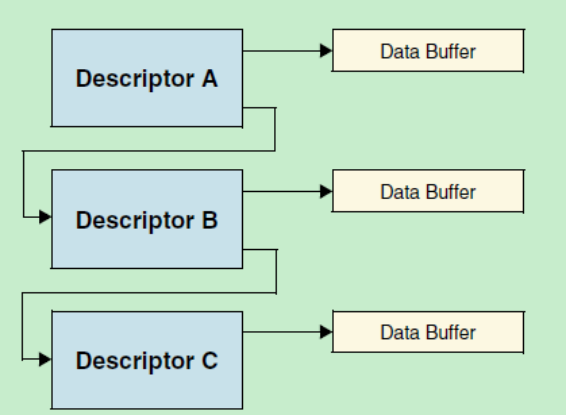


图 7-2DMA 描述符—链表方式

每个描述符长度为 16B，由 4 个连续的 32 位的字 DES0、DES1、DES2、DES3 组成。具体格式如下所示：

表 7-1DES0 结构

名称	范围	描述

OWN	[31]	表示当前描述符由 IDMAC 控制；完成数据传输后，IDMAC 需要清零该位。
CES	[30]	表示 DMA 传输发生了以下错误中的一个或多个： ● End_bit 错误 ● 响应超时 ● 响应 CRC 校验错 ● Start_bit 错误 ● 读数据超时 ● 接收数据 CRC 校验错 ● 响应错
Reserved	[29:6]	—
ER	[5]	只对双数据缓冲的描述符有意义。表示当前描述符是最后一个描述符。
CH	[4]	为 1 表示该描述符指向下一个描述符而不是第二个数据缓冲
FS	[3]	为 1 表示当前描述符指向第一个数据缓冲，如果当前描述符所指的数据缓冲大小为 0，则顺推指下一个描述符指向第一个数据缓冲。
LD	[2]	为表示当前描述符指向最后一个数据缓冲。
DIC	[1]	Disable Interrupt onCompletion；如果置为 1 时，则当前描述符所指缓冲中的数据完成传输时，将不再写寄存器 IDSTS 的 TI/RI 位。
Reserved	[0]	—

表 7-2DES1 结构

名称	范围	描述
RSV	[31:26]	保留
BS2	[25:13]	缓冲 2 大小，必须是 4B 的整数倍； 注意：①为 0 时直接忽略该域；②该域只对双数据缓冲的描述符由意义。
BS1	[12:0]	缓冲 1 大小，必须是 4B 的整数倍； 注意：该域不可以为 0。

表 7-3DES2 结构

名称	范围	描述
BAP1	[31:0]	表示第一个数据缓冲的物理地址；

表 7-4DES3 结构

名称	范围	描述
BAP2	[31:0]	对于链式管理的描述符，表示下一个描述符的物理地址； 对于双数据缓冲的描述符，表示第二个数据缓冲的物理地址。

7.11.2 DMA 初始化流程

1) 软件写寄存器 BMOD 配置 DMA 访问的相关参数；
   a) 为保证效率，寄存器 BMOD[FB]应设置为 1，确保 AHB burst 类型为 SINGLE 或者 INCR4 或者 INCR8 或者 INCR16；否则 AHB burst 类型为 SINGLE 或者 INCR，AHB-AXI 桥的效率会很低；
   b) 寄存器 BMOD[PBL]应显示为 3’b011，即每次 AHB 交易最多有 16 个 beat。
2) 软件写寄存器 IDINTEN64 来屏蔽不需要的中断源；
3) 软件往主存中写入描述符（DES0-DES3）列表（可以是双缓冲方式的，也可以使链表方式的），并写寄存器 DBADDR 记录描述符列表的头地址；
   a) 描述符的地址必须是 32 位对齐的。
   b) 描述符中 DES0[OWN]应该设置为 1；
   c) 缓冲的大小（DES1[BS1]、DES1[BS2]）必须是 4B 的整数倍；
   d) 如果是 DMA 读，软件应根据描述符列表同时准备好相应数据；
4) DMA 引擎会尝试获取描述符信息；
5) DMA 引擎只有在 FIFO 中有足够多的空闲数据来保存 16 个 burst 长度时，才会启动一次 DMA 读；只有在 FIFO 中有足够多有效的数据（16 个 burst 长度时），才会启动一次 DMA 写；

7.11.3 DMA 读操作流程

1) 初始化结束后，软件写寄存器 FIFOTH[TX_WMark]，配置 FIFO 传送阈值；
   a) 寄存器 FIFOTH[TX_WMark]必须大于等于 16，建议设置为 128。

2) 软件写寄存器 CMD（软件写，为何不是 DMA 引擎自行控制？CMD 和 CMD_ARG 分别设置何值？），通知 card 将启动一次写操作（同时也是 DMA 读操作）；
   a) CMD[data_expected] = 1；
   b) 此时，DMA 引擎根据寄存器 DBADDR 从主存中取第一个描述符，并检查 DES0[OWN]是否为 1。如果不为“1”，DMA 引擎进入挂起状态，并触发 Descriptor Unavailable 中断，即设置 IDSTS[DU]为 1。此时，需要软件将描述符 DES0[OWN] 设置为 1，并通过写寄存器 PLDMND 使 DMA 引擎从挂起状态下退出，并重新取描述符。
3) 当 Command Done 信号有效（即 BIU 收到了 card 写命令的响应），并且没有发生任何错误时，表明此时可以开始进行数据传输。BIU 会根据 FIFO 的空满情况，指示 DMA 引擎从主存中取数。DMA 引擎将取到的数据写入 FIFO。CIU 会从 FIFO 中读取相应数据，发送给 card；
4) 当一个描述符所指示的数据传输完毕后，DMA 引擎会往主存中写该描述符 DES0[OWN]为 0，然后自动读取下一个描述符，检查该描述符 DES0[OWN]是否为 1；并根据取得的描述符从主存中取数。上述过程直至最后一个描述符（即 DES0[LD]为 1）；
5) 数据传送完毕，DMA 引擎会置寄存器 IDSTS[TI]为 1，表示 DMA 读结束。

7.11.4 DMA 写操作流程

1) 初始化结束后，软件写寄存器 FIFOTH[RX_WMark]，配置 FIFO 接收阈值；
   a) 寄存器 FIFOTH[RX_WMark]必须大于等于 16，建议设置为 127
2) 软件写寄存器 CMD，通知 card 将启动一次读操作；
   a) CMD[data_expected]=1；
   b) 此时，DMA 引擎根据寄存器 DBADDR 从主存中取第一个描述符，并检查 DES0[OWN]是否为 1。如果不为“1”，DMA 引擎进入挂起状态，并触发 Descriptor Unable 中断，即设置 IDSTS[DU]为 1。此时，需要软件将描述符 DES0[OWN] 设置为 1，并通过写寄存器 PLDMND 使 DMA 引擎从挂起状态下退出，并重新取描述符。
3) 当 Command Done 信号有效（即 BIU 收到了 card 写命令的响应），并且没有发生任

何错误时，表明此时可以开始进行数据传输。此时 CIU 会从 card 处读取数据，并写入 FIFO，BIU 会根据 FIFO 的空满情况，指示 DMA 引擎从 FIFO 中取数，并写入主存中。

4) 当一个描述符所指示的数据传输完毕后，DMA 引擎会往主存中写该描述符 DES0[OWN]为 0，然后自动读取下一个描述符，检查该描述符 DES0[OWN]是否为 1；并根据取得的描述符把后续 FIFO 中的数据写入主存。上述过程直至最后一个描述符（即 DES0[LD]为 1）；

5) 数据传送完毕，DMA 引擎会置寄存器 IDSTS[RI]为 1，表示 DMA 写结束。

7.11.5 Stop 命令处理

当 DMA 正在进行时，软件发送 CMD12 给 card，即 stop 命令。DMA 引擎会完成当前正在处理的描述符所指示的数据传输，直至 DTO（Data Transfer Over）信号有效。但后续描述符不做处理。

7.11.6 AHB 总线错误处理

由于编程错误，比如 FIFOTH[TX_WMark]和 FIFOTH[RX_WMark]没有大于等于 16 或者描述符中缓冲大小没有设置为 8B 的整数倍，则会引发 AHB 总线错误 FEB。此时软件需要写寄存器 CTRL[0]来复位整个 eMMC 控制器。

发生 AHB 总线错误时，正在处理的描述符以及后续的描述符都不会被关闭（即 DES0[OWN]为 0）。

7.12 DDR 流程

进入 DDR 模式：软件将 UHS_REG 中对应位置 1；

退出 DDR 模式：发射 CMD0，更新寄存器 CLKDIV，将 UHS_REG 对应位清 0。

8 附录 1：参数配置

表 8-1 eMMC IP 参数定义

参数名称	含义	可选值	配置值
CARD_TYPE	card 的类型。 ● 0—版本为 3.3 的 MMC 芯片控制器； ● 1—SD_MMC 芯片控制器，支持 SD 存储卡、SDIO、MMC 芯片及 CE_ATA 等。	0/1	1
NUM_CARDS	表示最多支持的芯片数量，当选择模式 MMC-Ver3.3-only 时，可选范围是 1~30；当选择 SD_MMC_CE-ATA 模式是，可选范围是 1~16.	1~30（默认值是 3）	2
H_BUS_TYPE	主机总线类型 ● 0—APB； ● 1—AHB。	0/1	1
H_DATA_WIDTH	主机数据总线的宽度	16/32/64	32
H_ADDR_WIDTH	主机地址总线的宽度。 注意：在 DWC_mobile_storage 中，地址 0x0~0x1ff 映射到内部 IO 寄存器，地址 0x200 及以上的地址空间映射数据 FIFO。	10~28（默认值是 20）	20
INTERNAL_DMAC	是否使用内置的 DMA 引擎。 ● 0—不使用； ● 1—使用。	0/1	1
DMA_INTERFACE	-	0/1/2/3	-
GE_DMA_DATA_WIDTH	通用 DMA 接口的数据总线宽度。	16/32/64	-

FIFO_DEPTH	数据 FIFO 的深度。	8~4096（默认值是 16）	128
FIFO_RAM_INSIDE	FIFO 的放置位置。 ● 1：FIFO 实现时放置在 IP 内； ● 0：FIFO 实现时放置在 IP 外。	0/1	1
NUM_CLK_DIVIDERS	时钟分频器的个数	1/2/3/4	2
UID_REG	用于 ID 寄存器的值	0x0~0xffffffff（默认值是 0x7967797）	0x7967797
SET_CLK_FALSE_PATH	以下几组时序路径设为 false path：clk to cclk_in, cclk_in to clk, reset_n to cclk_in	0/1	1
AREA_OPTIMIZED	是否优化面积。优化造成的影响： ● 删除通用 IO 端口 ● 不再实现用户自定义寄存器 ● 在数据传输期间，不可以对读寄存器 TCBCNT。 面积优化大概可以节省 1000 个门。	0/1	0
IMPLEMENT_SCAN_MUX		0/1	0
M_ADDR_WIDTH	使用内置 DMA 引擎时，AHB Master 的地址总线宽度	32/64	32