

# SD 2.0 协议标准完整版

【翻译：沙贝@中科创达 2014.11】

## 1、总体描述

Sd 存储卡，是为了满足安全、容量、性能、和环境需求的新型音视频电子存储卡。

Sd 卡包含一个内容保护机制，符合 SDMI 标准，并且有更快的速度和更高的容量。

Sd 卡的安全系统采用双向认证和“新密码算法”来防止卡的内容被非法使用。也可以对用户自己的数据进行非安全访问。

SD 卡也支持基于常用标准的第二安全系统，比如 ISO-7816，这样就可以用于将 SD 卡连接到共用网络和其他系统，来支持移动电子商务和数字签名的应用。

除了 SD 卡外，还有 SDIO 卡。SDIO 卡规范在一个单独的规范中定义，命名为“SDIO 卡规范”（可以从 SD 协会得到）。SDIO 规范定义了一个 SD 卡可能包含不同的 IO 单元同 SD host 之间的接口。SDIO 卡可以包含存储功能，以及 IO 功能。SDIO 卡的存储部分应该完全兼容 SD 卡规范。SDIO 卡基于并兼容 SD 卡。这种兼容包括机械、电气、电源、信号和软件。Sdio 卡的意图是为移动电子设备在低功耗情况下提供高速数据读写。一个主要目标是一个 IO 卡插到非 SDIO 主机中，不会引起物理损坏或者设备和软件的中断。这种情况下 IO 卡应该被简单的忽略掉。一旦插入一个 SDIO 主控，卡的检测将以常规的方法描述，即带有 SDIO 规范扩展的 SD 卡规范

SD 卡通信是基于 9-pin 接口(时钟，命令，数据 x4，电源 x3)，设计在最大 50M 频率以及低电下工作。通信协议是本规范的一部分。SD 规范分为几个文件：



- 音频规范  
这个规范，以及其他应用规范，描述了一个特殊应用的规范(本文档是音频应用)，以及实施需求
- 文件系统规范  
这个规范描述了存储在 sd 卡上的数据的文件格式化结构的规范(保护和非保护区)
- 安全规范  
这个规范描述了内容保护机制和支持的特殊应用命令
- 物理层规范(本文档)  
这个规范描述了 sd 卡使用的物理接口和命令协议。这个文档的目的是定义 sd 卡以及它的环境和处理。这篇文档被分为了几个部分：  
第 3 章是关于系统概念的概述  
第 4 章描述了常见 SD 卡特点。这种描述定义了卡的整体性能，我们建议看产品文档  
第 5 章描述了 sd 卡寄存器  
第 6 章定义了 sd 卡的硬件接口的电气参数  
第 8 章描述了 sd 卡的物理和机械性能，以及卡槽或者盒子的最小建议。

这个文档中，“shall”和“will”表示一个标准的强制性规定。“should”表示一个条款，建议但不强制。“may”是指一个特征(feature)，可能存在或不存在(看使用者选择)，它的存在并不影响依存性

- Mc-EX 接口规范  
SD 卡规范的 A1 部分作为 SD 卡物理层规范的扩展，提供了所有传输移动商务扩展 (MC-EX) 命令包所需要的定义。(从 Mc-EX host 到 Mc-EX 使能 SD 卡，或者反向传输)

## 2、系统特征

- 针对便携式和固定式应用
- 存储容量：  
标准容量 SD 卡：最高达到 2GB  
大容量 SD 卡：大于 2GB(本版本规格最高 32GB)
- 电压范围：  
高电压 SD 卡-工作电压范围：2.7 -3.6V  
双电压 SD 卡-工作电压范围：低电范围(T.B.D) 和 2.7-3.6V
- 专为只读和读/写卡
- 默认模式：可变时钟频率 0-50MHz，最高 25MB/s 的接口速度(使用 4 条并行数据线)
- 切换功能命令支持高速，电子商务和未来的功能
- 存储区域错误改正
- 读取操作过程卡被移除，不会损坏内容
- 内容保护机制-符合 SDMI 标准的最高安全标准
- 卡密码保护(CMD42-LOCK\_UNLOCK)
- 机械开关的写保护功能
- 内置写保护功能(临时和永久)
- 卡检测(插入/拔出)
- 应用特殊命令
- 轻松擦写机制
- 通信信道的协议属性

SD 卡通信信道
6 线通信信道(时钟，命令，数据 x4)
错误保护数据传输
单块或多块的定向数据传输

- SD 卡形状  
标准尺寸的 SD 卡：见本规格的第 6、8 章  
Mini SD 卡：见“miniSD Memory Card Specification”  
Micro SD 卡：见“microSD Memory Card Specification”
- SD 卡标准尺寸 2.1mm 和 1.4mm

本规范的所有特征都是基于标准尺寸的 SD 卡

## 3、SD 卡系统概念

SD 卡提供给应用设计者一个低成本的存储设备(支持高安全级别的内容保护的可插拔卡)，以及一个简洁，易实现的接口。

SD 卡可以分为几个等级(class)，他们提供的功能不同(功过 D 卡系统命令的子集提供) 一个 SD 卡系统包含 SD 卡，总线，以及主机/应用。不过主机及应用的说明不在本文档中。接下来的各节提供了卡的概述，总线拓扑，SD 卡系统的通信协议。内容保护系统描述在文档“SD Memory Card Security Specification”中。

## 3.1 读写属性

以读写属性来说，有两种 SD 卡：

- 读写卡(闪存，一次可编程-OTP，多次可编程-MTP)  
这些卡一般是空白媒体卡卖出，用于存储含量数据，终端用户视频，音频或者数字图像
- 只读卡(ROM)  
这些卡是用固定内容制作的卡。他们通常用软件，音频视频的分发媒体。

## 3.2 支持电压

以电压来说，有两种 SD 卡：

- 高电压 SD 卡，可以工作在 2.7-3.6V
- 双电压 SD 卡，可以工作在低电范围(T. B. D)以及 2.7-3.6V

## 3.3 卡容量

以容量来说，有两种 SD 卡：

- 标准容量 SD 卡，支持最大 2GB 的容量。所有的物理规格文档都会定义这种
- 高容量 SD 卡，支持超过 2GB 的容量。本文档解释的规格最大为 32GB 容量。只有 Host 支持 2.0 协议才能够识别这种高容量 SD 卡。

注释：

- 1 “Part 1 物理层规格 V2.0”以及“Part2 文件系统规格 V2.0”允许标准容量 SD 卡最大 2GB，高容量 SD 卡最大 32GB。大于 32GB 的容量将在未来的版本中说明。
  - 2 Host 如果可以读写 2GB 到 32GB 的 SD 卡，那么它应该也可以读取 2GB 或更小的卡。如果 Host 只支持标准容量 SD 卡，则不能识别超过 2GB 的卡。即向下兼容。
- 高容量的 SD 卡有两种类型。类型 A(单状态卡)有一个单独的高容量存储区域。细节参考“Physical Layer Specification version2 2.00”。类型 B(多状态卡)有一个高容量存储区和标准容量存储区。在 B 类卡中，每次只能有一个存储区域能够使用。可以通过电气开关来切换哪个区域被使用。详细信息见未来的文档。**主机端不一定要区别两种类型。**

## 3.4 速度等级

我们定义了 4 个速度等级，来表示卡的最小速率：(实际上目前最高 Class10)

- Class 0 - 这种卡不定义具体性能，代表了这个规范出来之前的所有卡
- Class 2 - 最小 2MB/s 的性能
- Class 4 - 最小 4MB/s 的性能
- Class 6 - 最小 6MB/s 的性能
- Class 8 - 最小 8MB/s 的性能
- Class 10 - 最小 10MB/s 的性能

大容量 SD 卡应该支持速度等级规格，并且最小要到 Class2。

注意：性能单位表示的是 1000x1000[字节/秒]，而数据大小的 MB 单元指的是 1024x1024 字节。这是因为最大 SD 总线速度是由最大 SD 时钟频率决定的，而数据大小是基于存储范围。

### 3.5 总线拓扑

SD 卡系统定义了两种通信协议：SD 和 SPI

主机系统可以选择任意一种。当收到 reset 命令的时候，SD 卡通过主机的信息来决定使用何种模式，并且之后的通讯都会使用相同的模式。不推荐多卡槽用共同的总线信号。一个单独的 SD 总线应该连接一个单独的 SD 卡。在主机支持高速模式的情况下，单独的 SD 总线应该连接单独的 SD 卡。

#### 3.5.1 SD 总线

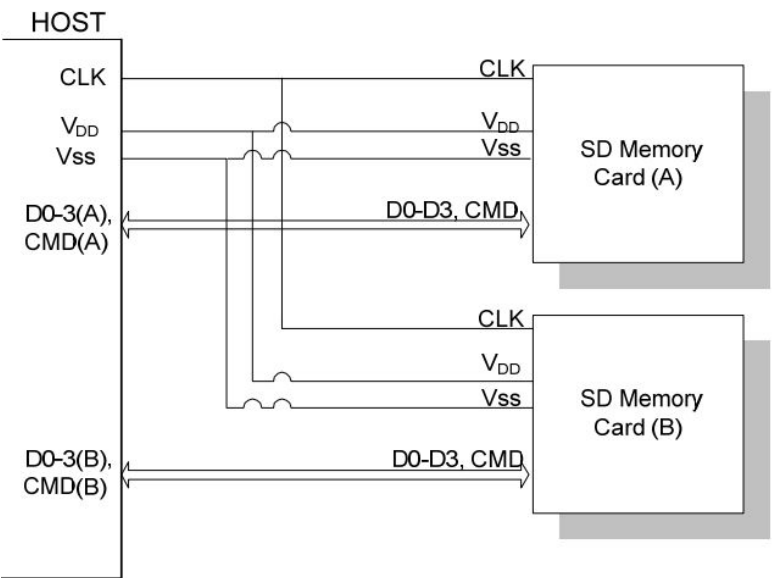


图 3-2 SD 卡总线拓扑

SD 总线包含下面的信号：

- CLK: 时钟信号
- CMD: 双向命令/响应信号
- DAT0-DAT3: 双向数据信号
- Vdd, Vss1, Vss2: 电源和地信号

SD 卡总线有一个主(应用), 多个从(卡), 同步的星型拓扑结构(图 3-2)。时钟, 电源和地信号是所有卡都有的。命令(CMD)和数据(DAT0-3)信号是根据每张卡的, 提供连续地点点对点连接到所有卡。

在初始化时, 处理命令会单独发送到每个卡, 允许应用程序检测卡以及分配逻辑地址给物理卡槽。数据总是单独发送(接收)到(从)每张卡。但是, 为了简化卡的堆栈操作, 在初始化过程结束后, 所有的命令都是同时发送到所有卡。地址信息包含在命令包中。

SD 总线允许数据线的动态配置。上电后, SD 卡默认只使用 DAT0 来传输数据。初始化之后, 主机可以改变总线宽度(使用的数据线数目)。这个功能允许硬件成本和系统性能之间的简单交换。

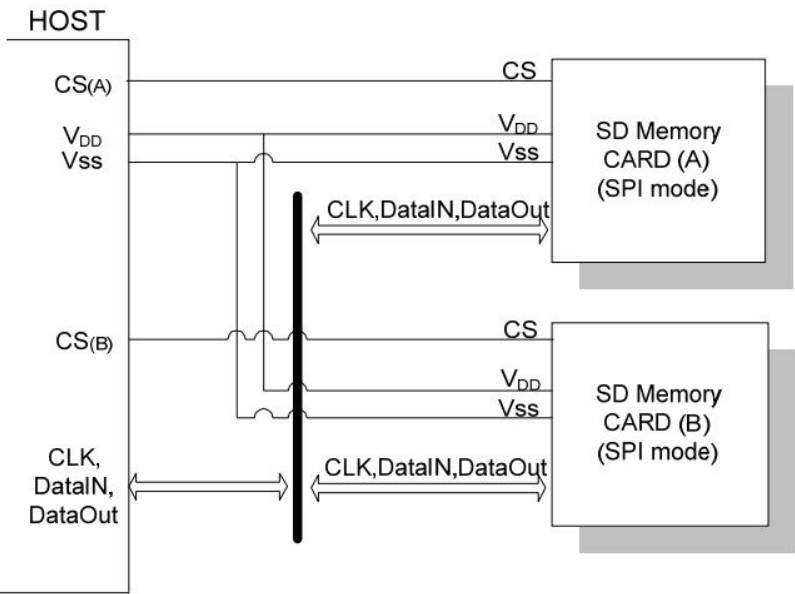
注意：当 DAT1-DAT3 没有使用的时候，相关的主机 DAT 先应该被设置为输入模式。SDIO 卡 DAT1 和 DAT2 用于信令。

### 3.5.2 SPI 总线

SD 卡的 SPI 兼容通信模式是用来同 SPI 信道通信，主要是用在市场是哪些的各种微处理器。模式选择是在上电后的第一次 reset 命令期间，并且只要不断电就不能改变。SPI 标准只是定义了物理连接，没有完成数据传输协议。SD 卡的 SPI 实现使用了 SD 模式相同的命令。从应用的角度来说，SPI 模式的优点是使用现成主机的能力，从而减小设计压力。相对于使能宽总线选项的 SD 卡来说，缺点是性能的损失。SD 卡 SPI 接口同市场上现有的 SPI 主机兼容。同其他 SPI 设备一样，SD 卡的 SPI 信道有以下 4 个信号：

- CS: 主机到卡的片选(chip select)信号
- CLK: 主机到卡的时钟信号
- DataIn: 主机到卡的数据信号
- DataOut: 卡到主机的数据信号

另一个 SPI 的通用特点是字节传输，这也是卡的实现。所有的数据都是字节(8 bit)的整数倍, 并且直接总是对齐 CS 信号。



表格 3-3 SD 卡系统 (SPI 模式) 总线拓扑

卡片的识别和寻址方法由一个硬件片选信号代替。没有广播命令。每一个命令，都会有一个从卡通过拉低片选信号来被选择(表格 3-3)。

在 SPI 活动(命令，响应，数据)期间片选信号应该是连续被拉低的。唯一的例外是在卡编程期间，此时主机能在不影响编程过程的情况下断定 CS 信号。

SPI 接口使用 SD 总线的 9 线里面的 7 根(DAT1 和 DAT2 不用，DAT3 作为 CS 信号)

## 3.6 总线协议

### 3.6.1 SD 总线

SD 总线的通信是基于命令和数据流的。由一个起始位开始，由一个停止位终止。

- 命令(Command): 命令就是一个标记, 用于发起一个操作。由主机发送到单个卡(寻址命令)或者所有卡(广播命令)。命令在 CMD 线上是连续传输的。
- 响应(Response): 响应是一个标记, 从寻址的卡或者所有卡(同步)发送给主机, 作为向前接收到的命令的回答。响应也是在 CMD 线上连续传输的。
- 数据(Data): 数据可以从主机到卡, 也可以从卡到主机。通过数据线传输。

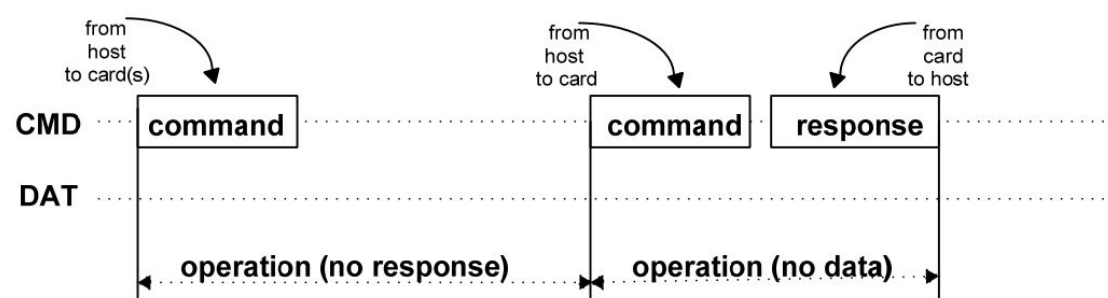


Figure 3-4: “no response” and “no data” Operations

卡片寻址通过使用会话地址来实现, 会话地址会在初始化阶段分配给卡。命令, 响应和数据块的结构在第 4 章中描述。SD 总线上的基本交互是命令/响应交互(表格 3-4)。这种总线交互直接在命令或者响应的结构里面传输他们的信息。此外, 一些操作还有数据内容。

SD 卡发送或接收的数据在块(block)中完成。数据块以 CRC 位来保证成功。目前有单块或多块操作。注意: 多块操作模式在快速写操作时更好一点。多块传输以命令线上的结束命令为结束标志。主机端可以配置单线还是多线传输。

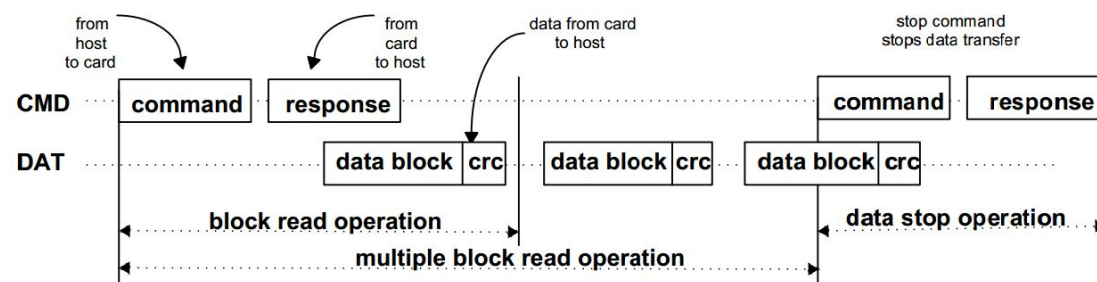


Figure 3-5: (Multiple) Block Read Operation

块写操作使用简单的 busy 来表示 DAT0 数据线上的持续写操作, 不管使用几线传输。

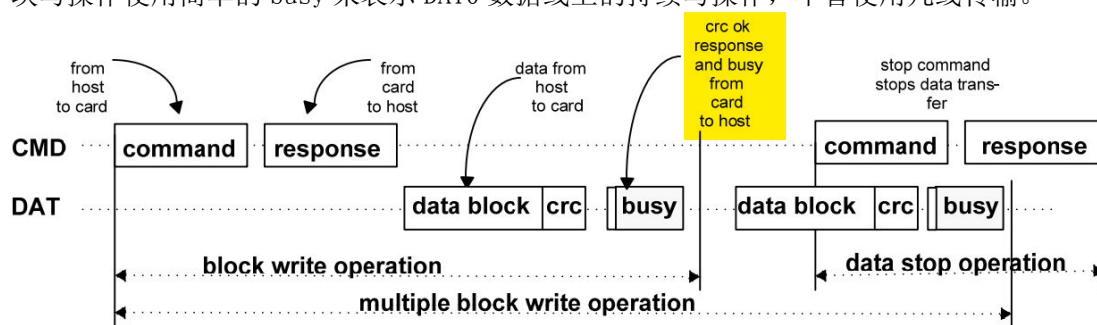


Figure 3-6: (Multiple) Block Write Operation



命令符号是以下的编码方案：

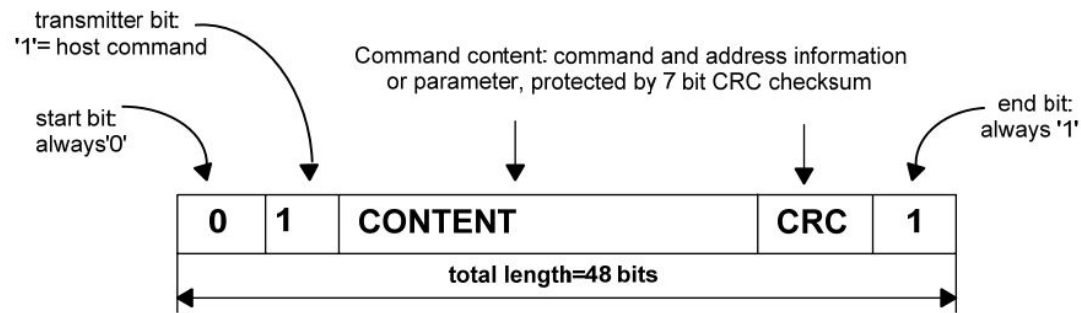


Figure 3-7: Command Token Format

注：命令内容：命令+地址信息/参数+7 位 CRC 校验(48bit)

每一个命令标记都是以起始位 bit(0) 在最开始，以结束位 bit(1) 表示成功。总长度是 48Bit。每个命令都使用 CRC 位来保护，这样可以检测到传输错误，并且再次发送命令。

响应标记以内容分，有 4 种编码方式。编码的长度可以是 48Bit 或者 136bit，详细的命令和响应见 4.7 章。数据块的 CRC 保护算法是一个 16bit 的 CCITT 多项式。所有允许的 CRC 类型见 4.5 章

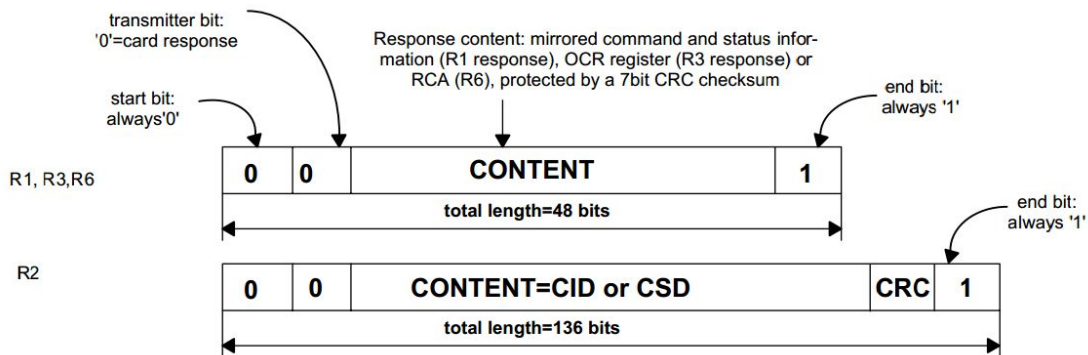


Figure 3-8: Response Token Format

注：

- R1 response: 方向位[1Bit]+命令[8Bit]+状态信息[32Bit]+CRC[7Bit] = [48Bit]
- R2 response: 方向位+命令+CID/CSD 寄存器+CRC = [136Bit]
- R3 response: 方向位+命令+OCR 寄存器+CRC 校验 = [48Bit]
- R6 response: 方向位+命令+RCA 寄存器+CRC 校验 = [48Bit]

在命令线上最高有效位(MSB)先发送，最低有效位(LSB)后发送(从高位到低位传输)。

当使用宽总线操作的时候，数据每次传 4Bit(见表格 3-10)。起始位、结束位以及 CRC 位，每条数据线上都要发送。CRC 位每条数据线都要分别检查。CRC 状态反馈和忙碌只是只会从 DAT0 发送到 host 端，DAT1-DAT3 忽略。

SD 卡有两种数据包格式。

- (1) 常规数据(8bit 宽)：常规数据发送是先低字节，再高字节的顺序，但是每个字节则是先高位后低位。
- (2) 宽位数据(SD 存储寄存器)：宽位数据从高位开始传输。(不按字节?)

## 1、常规数据包格式(8bit 宽)





### 3.6.2 SPI 总线

SPI 总线协议的详细描述见第 7 章。

### 3.7 SD 卡管脚和寄存器

SD 卡的尺寸可以分为 24mm\*32mm\*2.1mm 和 24mm\*32mm\*1.4mm

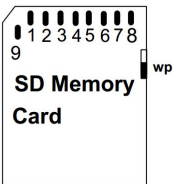


Figure 3-11: SD Memory Card Shape and Interface (Top View)

图 3-11 想死了一般的 sd 卡的形状和接口。详细的物理尺寸和力学描述在第 8 章

表 3-1 定义了卡的接口：

Pin#	SD 模式			SPI 模式		
	名称	类型①	描述	名称	类型①	描述
1	CD/DAT3②	I/O/PP③	检测/数据 3	CS	I③	片选(低有效)
2	CMD	PP	命令/响应	DI	I	数据输入
3	Vss1	S	接地电源	Vss	S	接地电源
4	Vdd	S	电源	Vdd	S	电源
5	CLK	I	时钟	SCLK	I	时钟
6	Vss2	S	接地电源	Vss2	S	接地电源
7	DAT0	I/O/PP	数据线 0	DO	O/PP	数据输出
8	DAT1④	I/O/PP	数据线 1	保留		
9	DAT2⑤	I/O/PP	数据线 2	保留		

- ① 类型解释：S-电源；I - 输入；O - 输出(使用推挽式单元?)；PP-I/O(推挽式)
- ② 扩展数据线(DAT1-DAT3)上电之后是输入。在 SET\_BUS\_WIDTH 命令之后作为数据线。当然不用的话，主机会让他们保持输入状态。
- ③上电时，这条线在卡中有一个 50K0hm 的上拉。这个电阻有两个作用：卡检测和模式选择。作为模式选择，主机可以拉高或让别人拉高这个脚来选择 SD 模式。如果主机想选择 SPI 模式，应该把管脚拉低；作为卡检测，当管脚被拉高时，认为卡插入了。这条线的上拉在数据传输期间应该由用户通过命令 SET\_CLR\_CARD\_DETECT (ACMD42) 命令来断开。
- ④DAT1 脚在 SDIO 模式下，一旦没有数据传输，可能被用于输出中断(从卡到主机)
- ⑤DAT2 脚在 SDIO 模式下，可能被用于“读等待信号”[见 SDIO Card Specification]

每个卡都有一组信息寄存器(第五章也有描述)：

名字	宽度	描述
CID	128	卡识别号；用来识别的卡的个体号码(见 5.2)。【强制的】
RCA	16	相对地址；卡的本地系统地址，初始化时，动态地由卡建议，主机核准，SPI 模式不使用(见 5.4)。【强制的】
DSR	16	驱动级寄存器；配置卡的输出驱动(见 5.5)。【可选的】
CSD	128	卡的具体数据；卡的操作条件信息(见 5.3)。【强制的】

SCR	64	SD 配置寄存器；SD 卡的特殊能力信息(见 5. 6)。【强制的】
OCR	32	操作条件寄存器(见 5. 1)。【强制的】
SSR	512	SD 状态；卡专有特征的信息(见 4. 10. 2)。【强制的】
CSR	32	卡状态；卡状态信息(见 4. 10. 1)。【强制的】

主机可以通过切换电源开关来实现卡的复位。但是每个卡都有自己的电源检测电路，用来在上电后将卡设置到一个预置状态，所以重启电源并不是必须的，通过发送 `GO_IDLE (CMD0)` 同样可以让卡复位。

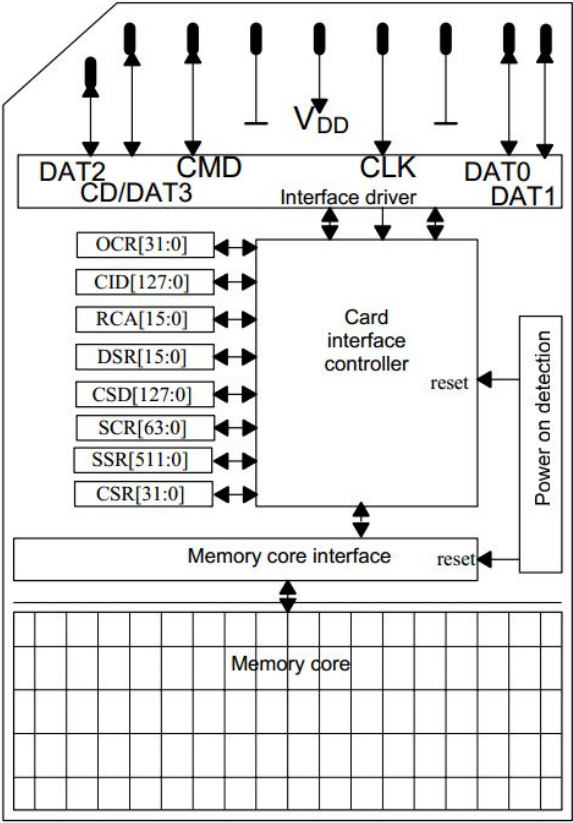


Figure 3-12: SD Memory Card Architecture

## 4、SD 卡功能描述

### 4.1 简介

主机和卡之间的交互都是由主机控制的。主机发送两种命令：广播命令，寻址(点对点)命令。

- 广播命令

广播命令的目的是所有的卡。部分命令需要响应。

- 寻址(点对点)命令

寻址命令是发送给对应地址的卡的，并且会引起这张卡的响应。命令流程的简介图，[图 4-1]是卡识别模式，[图 4-3]是数据传输模式。命令列举在命令表格中[表 4-19 ~ 表 4-28]。当前状态，收到命令和后续状态之间的关系在[表 4-29]中。后面的章节中，会首先描述各种卡的操作模式。之后，定义了控制时钟信号的限制。所有 SD 卡的命令以

及对应的响应，状态转换，错误条件以及时序都在后续章节

SD 卡系统(host &card)定义了两种操作模式：

- **卡识别模式**  
在复位后，查找总线上的新卡的时候，主机处于“卡识别模式”。卡在复位后会处于识别模式，直到收到 SEND\_RCA (CMD3) 命令。
- **数据传输模式**  
当 RCA 第一次发布后，卡会处于“数据传输模式”。主机会在总线上所有的卡都被识别后进入这个模式。

下表展示了操作模式以及卡状态之间的关系。每一个 SD 卡状态都会关联一个操作模式：

表格 4-1 卡状态 vs. 操作模式 概览

卡状态(Card state)	操作模式(Operation mode)
无效状态(Inactive State)	无效模式(Inactive)
空闲状态(Idle State)	卡识别模式(Card identification mode)
准备状态(Ready State)	
识别状态(Identification State)	
待机状态(Stand-by State)	数据传输模式(Data transfer mode)
传输状态(Transfer State)	
发送数据状态(Sending-data State)	
接收数据状态(Receive-data State)	
编程状态(Programming State)	
断开连接状态(Disconnect State)	

## 4.2 卡识别模式

在卡识别模式下，主机会复位所有处于“卡识别模式”的卡，确认工作电压范围，识别卡，并且要求他们发布相对卡地址(Relative Card Address)。这个操作是通过卡各自的 CMD 线完成的。卡识别模式下，所有数据通信都只通过数据线完成。

在卡识别过程中，卡应该在识别时钟频率 FOD 下的 SD 时钟频率中工作(见 6.7 章)

### 4.2.1 卡复位

GO\_IDLE\_STATE (CMD0) 是软复位命令，设置每张卡进入“Idle”状态，不管当前是什么状态。“Inactive”状态的卡不受这个命令的影响。

主机上电后，所有的卡进入“Idle”状态，包括“Inactive”状态的卡

在“上电”或者“CMD0”后，所有卡的 CMD 线都是输入模式，等待下个命令的起始位(Start bit)。卡初始化的时候，会有一个默认的相对地址(RCA=0x000)，和一个默认的驱动级寄存器设置(最低速度，最高驱动电流能力)。

### 4.2.2 操作条件确认

在主机和卡交互之初，主机可能不知道卡支持的电压，卡也可能不知道是否支持当前的电压。主机会先假设 SD 卡支持某个电压，并以这个电压发送一个复位命令 CMD0。为了验证电压，“Physical Layer Specification V2.0”又定义了一个新的命令 CMD8。

SEND\_IF\_COND(CMD8)用于验证 SD 卡接口操作条件。卡会通过分析 CMD8 的参数来检测操作条件的正确性。而主机会通过分析 CMD8 的响应来检查正确性(见 4.3.13)。支持的电压是由参数里的 VHS 区指定的。卡会假设 VHS 里面指定的电压是当前支持的电压。每一次命令 VHS 里面只有 1 位能被设置为 1。主机会通过 CRC 和检查模式来确认通信的有效性。

如果卡能够在支持电压下操作，响应会传回命令参数里设置的支持的电压和检测模式。

如果卡不能在支持电压下操作，就不会发送响应，并保持在“idle”状态。强制要求：在发送第一个 ACMD41 之前要先发送 CMD8，以便初始化大容量 SD 卡[图 4-1]。SD 卡如果收到 CMD8，就会知道主机支持 V2.0，就可以使能新的功能。

同样强制的，低电主机在发送 ACMD41 之前，也要发送 CMD8。如果双电压卡没有收到 CMD8，那么卡就会作为单独的高电压卡来工作，并且如果低电主机不发送 CMD8 的话，卡在收到 ACMD41 后会进入“inactive”状态。

**SD\_SEND\_OP\_COND(ACMD41)**是用来提供给主机一种机制来识别和拒绝那些不匹配它期望的 VDD 范围的卡。主机通过发送需求的 VDD 电压范围来完成这个命令，这个范围作为命令的参数(见 5.1)。不能支持指定电压范围的卡应该自动放弃后续的总线操作，并且进入“inactive”状态。**OCR 寄存器里面的标准应该响应的定义(见 5.1)?**。注意：ACMD41 是应用特定命令，因此 **APP\_CMD(CMD55)** 应该永远在 ACMD41 之前发送。“idle”状态下用于 CMD55 的 RCA 寄存器应该是默认的值 RCA=0x0000。

主机发送复位命令(CMD0)复位卡，主机应该在 ACMD41 之前发送 CMD8 重新初始化 SD 卡。

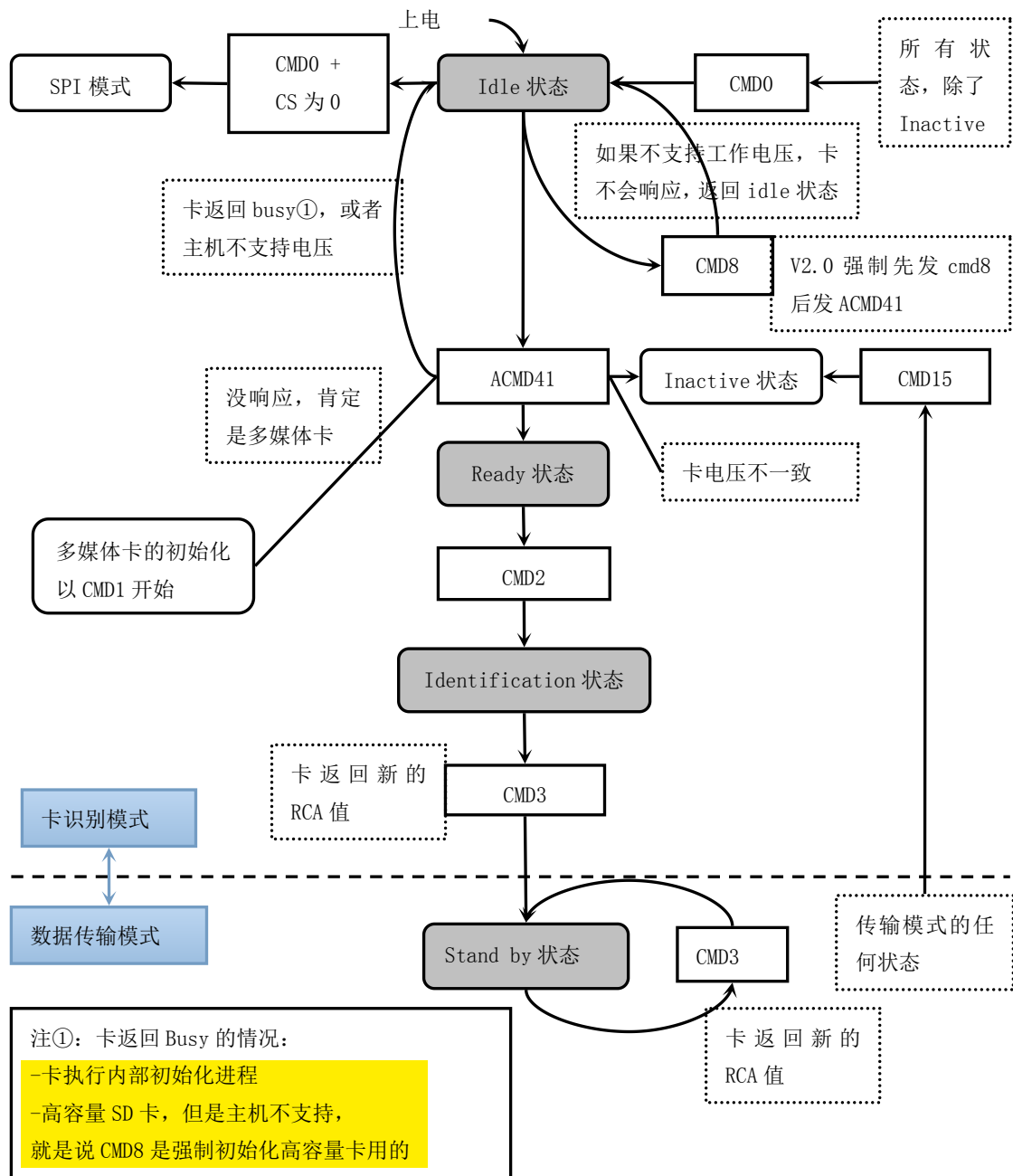


图 4-1 SD 卡识别模式状态图

通过设置 ACMD41 的参数里面 OCR(工作电压范围)的值为 0，主机可以查询每张卡，并且确定通用电压范围，进而通知超出范围的卡进入 Inactive 状态(查询模式)。但这种查询，要求主机能够选择一个通用的电压范围或者应用得到了堆栈中的不可用卡的信息。如果 ACMD41 是作为查询命令发送的，卡无法启动初始化。之后，主机可以选择一个工作电压，并重新发送 ACMD41，带着这个电压参数，这样就可以让不兼容的卡进入 Inactive 状态。

在初始化过程中，主机不能够改变工作电压。启动流程参考 6.4

### 4.2.3 卡初始化和识别过程

总线激活后，主机启动卡的初始化和识别进程(见图-2)。初始化进程以命令

SD\_SEND\_OP\_COND(ACMD41)作为开始,通过设置操作条件和 OCR 的 HCS 位来进行。HCS (High Capacity Support)位为 1,表示主机支持大容量 SD 卡。为 0 表示不支持。

CMD8 扩展了 ACMD41 的功能;参数里的 HCS 位以及响应里的 CCS (Card Capacity Status)位。HCS 会被不回应 CMD8 的卡忽视掉。然而,如果卡不回应 CMD8,主机应该设置 HCS 为 0。标准容量卡会忽略 HCS。如果 HCS 设置为 0,那么大容量 SD 卡永远都不会返回 ready 状态(保持 busy 位为 0)。卡通过 OCR 的 busy 位来通知主机 ACMD41 的初始化完成了。设置 busy 位为 0 表示卡仍然在初始化。设置 busy 位为 1,表示已经完成初始化。主机会重复发送 ACMD41,直到 busy 为被设置为 1。

卡片只在第一个 ACMD41 的命令时,检查操作条件和 OCR 里面的 HCS 位。当重复 ACMD41 的时候,除了 CMD0,主机不应该再发其他命令。

如果卡响应了 CMD8,那么 ACMD41 的响应就包括了 CCS 字段信息。当卡返回“ready”的时候,CCS 是有效的(busy 位设置为 1)。

CCS=1 表示卡是大容量 SD 卡; CCS=0 表示卡是普通 SD 卡。

在系统中,主机遵照相同的初始化顺序来初始化所有的新卡。不兼容的卡会进入“Inactive”状态。主机接着就会发送命令 ALL\_SEND\_CID(CMD2)给每一个卡,来得到他们的 CID 号。未识别的卡(处于 Ready 状态的)发送自己的 CID 作为响应。当卡发送了 CID 之后,它就进入“Identification”状态。之后主机发送 SEND\_RELATIVE\_ADDR(CMD3)命令,通知卡发布一个新的相对地址(RCA),这个地址比 CID 短,用于作为将来数据传输模式的地址。一旦收到 RCA,卡就会变为“Stand-by”状态。这时,如果主机想要分配另一个 RCA 号,它可以再发送一个 CMD3,通知卡重新发布一个 RCA 号。最后一个产生的 RCA 才是有效的。主机会重复识别进程,为系统中的每个卡循环发送“CMD2”和“CMD3”。



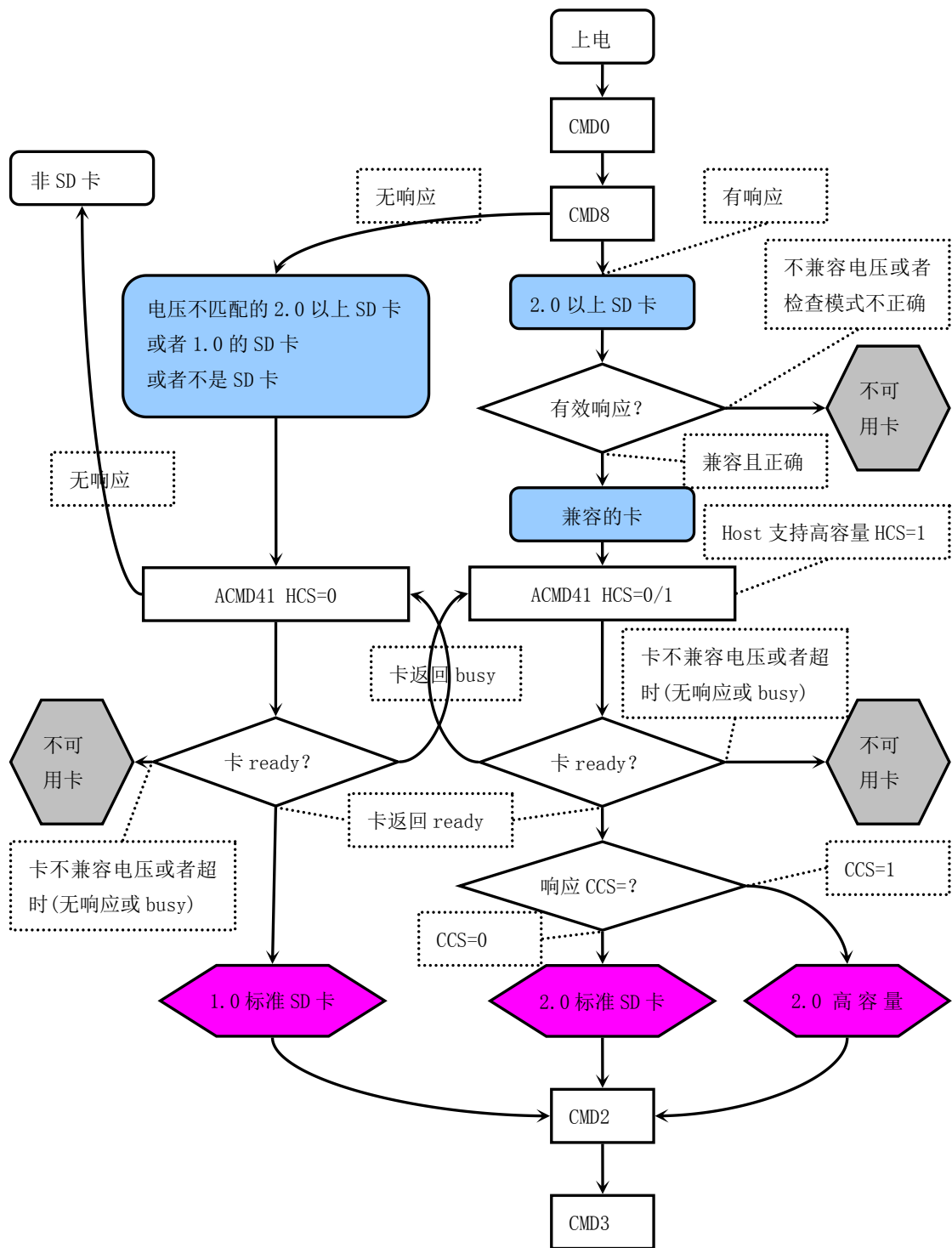


表 4-2 卡初始化和识别流程(SD 模式)

### 4.3 数据传输模式

在卡识别模式期间，主机应该保持在 Fod 频率，因为某些卡可能在卡识别模式中有频率限制。在数据传输模式，主机可以在 Fpp 频率范围（见 6.7）操作卡。主机发送命令 SEND\_CSD(CMD9)来获得“卡具体数据(Card Specific Data)”，比如“块长度”，“存储容量”

数据传输模式所有状态等。广播命令 SET\_DSR(CMD4)会配置所有已识别卡的驱动范围(电流?)。它会根据应用总线布局(长度)、总线上卡的数量以及数据传输频率来配置它们的 DSR 寄存器。同时时钟频率也从 Fod 切换到 Fpp。SET\_DSR 命令卡和一个选项。CMD7 的作用是选择一张卡,然后把它切换到传输模式,每次只能有一张卡处于传输模式。如果一张处于传输模式的卡同主机的连接被释放,那么它会回到“Stand-by”状态。当 CMD7 带着参数 RCA=0x0000 发送的时候,所有的卡都会回到“Stand-by”状态(注意:发送 RCA=0 来取消卡选择是主机的责任-参考表 4-19, CMD7)。

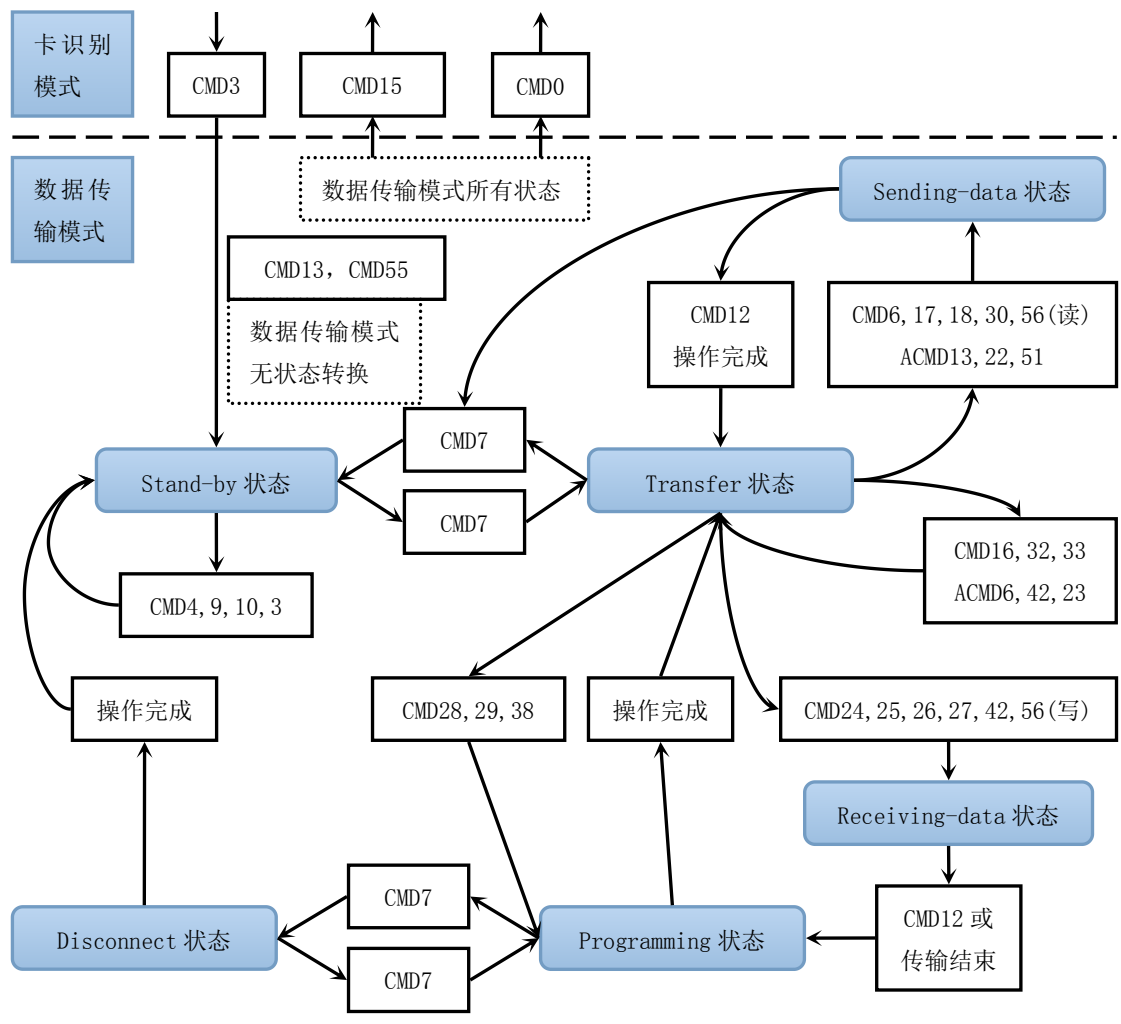


表 4-3 SD 卡状态表(数据传输模式)

不复位其他已经注册卡的情况下,在识别新卡之前这个会用到。在这种状态下,已经有 RCA 的卡对识别命令(ACMD41, CMD2, 见 4.2.3)不会响应。

**重要注意:** 如果某些卡收到 CMD7(带有不匹配的 RCA),卡会被取消选择。如果选择到另一张卡并且命令线是通用的,这个会自动发生。因此,在 SD 卡系统中,以通用命令线进行工作(初始化完成后)也是主机的责任,在这种情况下卡的取消选择会自动完成,如果命令线是单独的,那么主机应该做必要的事情来取消对卡选择

所有数据传输模式下的数据通信都是主机和被选择卡之间通过寻址命令点对点进行的。寻址命令以命令线上的响应作为应答信号。

各种数据传输模式的关系总结如下:

- 所有数据读命令可以在任何情况下通过停止命令 (CMD12) 来中止。数据传输会中止，卡会回到传输状态。读命令有：块读 (CMD17)，多块读 (CMD18)，发送写保护 (CMD30)，发送 SCR (ACMD51) 以及读模式的通用命令 (CMD56)。
- 所有数据写命令同样也可以通过 CMD12 来中止。在发送 CMD7 取消选定卡之前，应该先停止写命令。写命令有：块写 (CMD24 和 CMD25)，编程 CSD (CMD27)，锁定/解锁命令 (CMD42) 以及写模式的通用命令 (CMD56)。
- 一旦数据传输完成，卡会退出写状态，并且进入编程状态 (传输成功) 或者传输状态 (传输失败)。
- 如果“块写操作”停止，并且块长度和最后一个块的 CRC 是有效的，那么数据会被处理。
- 卡可能会提供缓存给“块写”。这就意味着当前一个块正在处理的时候，就可以发送后一个块了。如果缓存都满了，那么卡就会处于编程状态 (见图 4-3)，DAT0 会被拉低 (busy)。
- “写 CSD”，“写保护”和“擦除”的时候没有缓存操作，这就意味着当这些命令被处理的时候，其他传输命令都不接受。DAT0 线也会保持低电平，保持编程状态。  
实际上，如果 CMD 和 DAT0 线都是分开的，那么在一个卡的 DAT0 是低电平的时候，主机可以访问其他的卡。
- 当卡正在处理命令/数据的时候，是不允许发送参数设置的命令的。参数设置命令有：设置块长度 (CMD16)，擦除块开始 (CMD32) 和擦除块结束 (CMD33)
- 当卡正在处理命令/数据的时候，读命令也是不允许的。
- 将另一张卡从 Stand-by 模式转换到 Transfer 模式 (CMD7) 不会终止擦除和编程操作。卡会切换到 Disconnect 状态并且释放 DAT 线。
- 处于 Disconnect 状态的卡可以通过 CMD7 的命令重新被选定。这时，卡会进入 Programming 模式，并且重新使能 busy 指示。
- 复位卡 (CMD0 或者 CMD15) 会中止任何等候或者执行的编程操作。这可能会损坏卡的内容。主机应该尽量预防这种事情。
- CMD34-37，CMD50 和 CMD57 是 SD 命令系统的保留命令。这些命令的状态转换是单独定义的。

### 4.3.1 宽总线选择/取消

宽总线 (4Bit 宽) 操作模式可以通过命令 ACMD6 来选定和取消。上电或者 GO\_IDLE (CMD0) 命令后，默认的总线宽度是 1Bit。

如果想要改变总线宽度，需要具备下面两个条件：

- a) 卡处于 transfer 状态
- b) 卡没有被锁定 (锁定的卡会认为 ACMD6 是无效命令)

### 4.3.2 2GB 卡

要做 2GB 的卡，最大块长度应该被设置为 1024 byte。然而通过 CMD16 设置的块长度，应该最大到 512 byte，以便保持对最大 512 byte 的卡 (小于 2GB) 的兼容性。

### 4.3.3 读数据

当没有数据传输的时候，DAT 总线电平被拉高。一个传输的“数据块”包含了起始位 (1bit 或者 4bit 低)，和后面连续的数据流。数据流包含了有效数据 (如果用了卡外 ECC，还有错

误修正位)。数据流后面是**结束位**(1bit 或者 4bit 高电平)(见表 4-18 ~4-20)。数据传输是和时钟信号同时进行的。块数据传输的有效性是通过 1bit 或者 4bit 的 CRC 校验值来进行的(见 3.6)。

从 SD 卡读数据的操作有可能被断电中止,即使突然断电或者移除,SD 卡都会确保数据不被损坏,但是主机发起的写操作和擦除操作时不行。

BLOCK\_LEN\_ERROR 或者 ADDRESS\_ERROR 发生的时候,读命令是不能执行的,并且也不能数据传输。

● 块读

块读,是以块为单位的数据传输。数据传输的基本单位是一个块,最大为 512 Byte。其实和结束地址整个包含在 512byte 里的更小的块可能会被传播。

CMD16 设置的块长度可以到 512Byte,不管 READ\_BL\_LEN 的值是多少。

每个块的后面都会有一个 CRC 值,用来确保数据传输的有效性。READ\_SINGLE\_BLOCK(CMD17)代表读取一个块的内容,传输结束后,回到 transfer 状态。READ\_MULTIPLE\_BLOCK(CMD18)代表读取多个连续的块。块会连续的传输,直到 STOP\_TRANSMISSION(CMD12)命令发出。因为连续数据传输,停止命令会有些执行延迟。**数据传输会在停止命令的结束位发送之后停止。**

如果主机使用部分块的累计长度不能对齐,而又不允许不对齐,卡应该在第一个不对齐的块开始的时候就检测到这个问题,设置 ADDRESS\_ERROR 错误位到状态寄存器,停止传输,并在 Data 状态等待停止命令。

表 4-2 定义了允许部分块访问的时候卡的行为。  
如果不对齐的块是命令的第一个块(ADDRESS\_ERROR 已经在命令的响应中了),那么没有数据会被传输,卡会停留在 transfer 状态。

表 4-2 读命令块长度

CSD 值			当前块长度 ①	读命令起始地址
最大块长度 READ_BL_LEN	不对齐	局部		
512Byte	0(Disable)	1(Enable)	1-512 Bytes	可接受的任意地址②
1024Byte	0(Disable)	1(Enable)	1-512 Bytes	可接受的任意地址
2048Byte	0(Disable)	1(Enable)	1-512 Bytes	可接受的任意地址

①: 当前块长度的值可以通过 CMD16 来设置或改变。值小于等于 512Byte 就没问题  
②: 当块长度大小数据范围超过 512Byte 块边界,卡会输出数据直到 512Byte 块的边界,之后的数据就变成无效,并且可能发生 CRC 错误,卡会在下一个命令响应时发送“ADDRESS\_ERROR”。主机应该发送 CMD12 来还原。

4.3.4 写数据

写数据的数据传输格式和度数据类似。按块的写数据传输,CRC 检查为在每一个数据块后。在写之前,卡会检测每一个收到的数据块的 CRC 值。这样传输数据的错误可以避免。

如果发生 BLOCK\_LEN\_ERROR 和 ADDRESS\_ERROR 的错误,写命令就无法执行了。

● 块写

在块写(CMD24-27, 42, 56(写))期间,一个或者多个数据块从主机传递到卡,每个块都有 CRC。支持块写的卡,应该通过命令 CMD16 设置块长度为 512Byte,即使 WRITE\_BL\_LEN 的值设置为 1024Byte 或者 2048Byte。

当部分块访问关闭(WRITE\_BL\_PARTIAL=0)的时候卡的行为见下表

表 4-3 写命令块长度

CSD 值			当前块长度①	写命令起始地址
最大块长度 WRITE_BL_LEN	不对齐	局部		
512Byte	0 (Disable)	0 (Enable)	512 Bytes②	n*512Byte ③
1024Byte	0 (Disable)	0 (Enable)	512 Bytes	n*512Byte
2048Byte	0 (Disable)	0 (Enable)	512 Bytes	n*512Byte

①：CMD16 修改这个值，只要小于等于 512Byte 就行

②：如果当前的长度和这个值不一致，那么卡会在响应中加上“BLOCK\_LEN\_ERROR”。

③：如果起始地址不是这个值，那么卡会在写命令响应中加上“ADDRESS\_ERROR”

如果 WRITE\_BL\_PARTIAL=1，那么从 1-512Byte 都可以使用。如果 CRC 失败了，卡应该在数据线上标明失败；传输数据会被丢弃，后续的块也会无效。

当需要快速的写操作时，应该使用多块写命令，而不是连续的单块写命令。

如果主机使用了部分块(累计长度不是块对齐的)，而卡又不支持块的不对齐(CSD 的 WRITE\_BL\_MISALIGN 没有设置)，卡应该检查块的不对齐错误，并在第一个不对齐块开始之前停止编程。卡应该设置 ADDRESS\_ERROR 位到状态寄存器，取消后续传输，并在 receive-data 状态等待停止命令。

注意：如果写命令的第一个数据块就是不对齐的，那么卡会停留在 transfer 状态，并且不会对数据编程。

如果主机试图写数据到保护区，写操作同样会被中止。这种情况下，卡应该设置 WP\_VIOLATION 位。

CSD 寄存器的编程，不需要预置块长度设置。传输数据同样有 CRC 保护。如果 CSD 寄存器的一部分存储在了 ROM 中，那么这个不可更改的部分应该匹配接收缓存中对应的部分。如果匹配失败，那么卡会报告一个错误，并且不会改变任何寄存器的内容

有些卡可能需要较长和不定的次数来写一个数据块。在收到数据块，并检测 CRC 之后，卡会开始写，如果写缓存慢了，不能再接受新的 WRITE\_BLOCK 的数据，就会把 DAT0 拉低。主机可以在任意时间，通过发送 SEND\_STATUS (CMD13) 来轮询卡的状态，卡会报告状态。状态位 READY\_FOR\_DATA 表明了卡是否接受新的数据以及目前写进程是否仍在进行。主机可以通过 CMD7 取消卡的选定(以便选择另一个卡)，这样会使卡进入 Disconnect 状态，并且释放 DAT 线，但是不会打断写操作。当重新选定这张卡后，如果写操作还在进行并且缓存满的话，会重新激活 busy 指示。实际上主机通过交叉处理可以对几张卡同步进行写操作。交叉处理可以在其他卡忙的时候单独对每张卡进行访问。这种处理可以通过适当的 CMD 和 DAT0-3 线的操作来实现(断开忙碌的卡)。

### ● 多块写操作前的擦除设置

设置写数据块的数据来“预擦除”(ACMD23)会使多块写操作比不发送 ACMD23 更快。主机机会使用这个命令来定义下一个要写的数据块的数目。如果主机在全部数据发送到卡之前要中止写操作(通过停止传输)，那么剩下的块数据的内容会不确定(也可以被擦除或者保留旧数据)。如果主机要发送比 ACMD23 提供的数目更多的数据，那么卡会一个块一个块的擦除(收到一个，擦一个)。当多块写操作完成后，这个值会被重新设置为默认(=1)。

推荐在 CMD25 之前使用这个命令，一些卡的多块写操作会更快。注意：主机如果想使用“预擦除”功能，应该仅仅在 WRITE 命令之前发送 ACMD23。如若不然的话，“预擦除数目”



可能会在其他命令(比如: Security Application Commands)执行的时候被自动清除。

#### ● 发送块写的数目

使用 Pipeline 机制来进行数据缓存管理的系统, 当多块写操作的中间部分发生错误的时候, 不能确定哪个块是最后一个完好写到闪存中的。卡会通过 ACMD22 返回正确写入的块的数目。

### 4.3.5 擦除

同时擦除很多数据块有助于提高数据的吞吐量。这些写数据块的标识是通过 ERASE\_WR\_BLK\_START(CMD32), ERASE\_WR\_BLK\_END(CMD33)命令来完成的。

主机应该遵循下面的命令序列: CMD32 → CMD33 → CMD38(ERASE)。如果顺序不对, 卡就会设置 ERASE\_SEQ\_ERROR 位到状态寄存器, 并且重启整个序列。

如果收到序列外的命令(除了 SEND\_STATUS), 卡会设置 ERASE\_RESET 状态位到寄存器中, 重启擦除序列, 并且执行最后那个命令。

如果擦除范围包含了写保护的部分, 写保护的部分不会动, 而只擦除非保护的部分, 并且状态寄存器中的 WP\_ERASE\_SKIP 位应该设置。

地址设置命令里设置的地址区域是写块区域, 以 Byte 作为单位。**卡会忽略所有 LSB 的小于 WRITE\_BLK\_LEN 尺寸的部分?**

如块写部分的描述一样, 卡会 DAT0 拉低来表明正在进行擦除操作。真正的擦除时间可能会很长, 主机可能发送 CMD7 来取消卡的选定或者使卡进入 disconnect 状态。

擦除后的卡上的数据可能是 ‘0’ 或者 ‘1’, 取决于卡的厂家。SCR 寄存器的 DATA\_STAT\_AFTER\_ERASE(bit55)定义了它是 ‘0’ 还是 ‘1’。

### 4.3.6 写保护管理

SD 卡支持 3 中写保护方法:

- 物理写保护开关(主机支持)
- 卡内部写保护(卡支持)
- 密码保护卡锁定操作

#### ■ 物理写保护开关

卡的侧面有一个物理滑片(见第 8 章)被用来标记卡是否被写保护了。

#### ■ 卡内部写保护(可选)

卡的数据可以被写保护, 以免被擦除或者覆盖。整个卡可能被厂家或者内容提供者, 通过设置 CSD 的永久或临时写保护位, 而永久的写保护。对于支持通过设置 CSD 的 WP\_GRP\_ENABLE 位来实现数据写保护的卡来说, 数据的一部分可以被保护(单位是 CSD 中规定的 WP\_GRP\_SIZE 扇区), 而且写保护可以被用户改变。SET\_WRITE\_PROT 命令设置写保护区域的写保护, 而 CLR\_WRITE\_PROT 命令会清楚写保护区域的写保护。

SEND\_WRITE\_PROT 命令同单块读命令一类似。卡会发送一个数据块在 16 位 CRC 位之后有 32 位写保护位(表示了以指定的地址开始的 32 个写保护组)。写保护命令中的地址是一组以 byte 为单位的地址。卡会忽略所有 LSB 的低于组的部分。

#### ■ 密码卡锁保护在后面的章节中介绍

注意: 大容量 SD 卡不支持写保护, 也不会响应写保护命令(CMD28, CMD29, CMD30)



4.3.7 卡锁定/解锁操作[CMD42]

4.3.7.1 概述

密码保护功能，使主机能够在提供一个密码时锁定卡，这个密码也可以用来解锁。密码长度在 128Bit，长度保存在 8Bit PWD\_LEN 寄存器中。这些寄存器内容是固化的，所以不会丢失。

已经锁定的卡只会响应和执行基础命令类(class0)，ACMD41, CMD16 以及“锁定卡”命令类。因此，主机只能复位，初始化，选定，请求状态等。而不能访问卡上的数据。如果密码是预设的(PWD\_LEN 不为 0)，那么卡会在上电后就加锁。

类似于现有的 CSD 寄存器写命令，加锁/解锁命令只在“transfer”状态才能使用。就是说它是不带地址参数的，在使用前，必须已经选定卡。

卡的加锁/解锁命令有标准的单块写操作命令的结构体和总线传输类型。传输的数据块中包含了所有必须的信息(密码设置模式，密码，卡加锁/解锁等)。表 4-4 描述了命令数据块结构。注意：遵循 SD2.0 协议的主机应该在命令 CMD42 中设置保留位(Bit7-Bit4)为 0。

表格 4-4 加锁卡数据结构体

字节#	Bit7 Bit6 Bit5 Bit4	Bit3	Bit2	Bit1	Bit0
0	保留(设置为 0)	擦除	加锁/解锁	清除密码	设置密码
1	PWDS_LEN(密码长度)				
2	密码数据				
...					
PWDS_LEN+1					

- 擦除：1=强行擦除。设置为 1 的时候，其他位都是 0，
- 加锁/解锁：1=加锁卡；0=解锁卡；可以和设置密码同时发送，但不能和清除密码一起发送。
- 清除密码：1=清除密码
- 设置密码：1=设置新密码到 PWD
- 密码长度：定义了后面密码的长度(单位 byte)。如果是修改密码，那么包含了新旧密码总体的长度。密码最长 16 个字节，所以这个值最大 32。
- 密码数据：如果设置新密码，就只有新密码。如果是修改，就是旧密码+新密码。

主机在发送加锁/解锁命令钱，应该确定数据块的大小。数据块的长度应该大于或等于请求的加锁/解锁命令的数据结构体。后面的解释中，使用 CMD16 来修改块大小并非加锁解锁命令的强制要求。

下面是各种加锁解锁命令的序列：

- 设置密码
  - 选定卡(CMD7), 如果之前没选定的话
  - 定义块长度(CMD16)(由 8bit 卡加锁/解锁模式提供)，由 8Bit 的密码长度(字节为单位)和新密码的字节数组成。如果是修改密码，块长度要包含新旧密码。
  - 发送卡加锁/解锁命令(CMD42)(走数据线)，包含合适的块长度和 16bit 的 CRC。数据块应该标明模式(SET\_PWD)，长度(PWD\_LEN)和密码本身。替换密码还要带上旧密码。注意：卡要处理新密码的长度，即从 PWDS\_LEN 中减去旧密码的长度。
  - 如果旧密码不正确，那么 LOCK\_UNLOCK\_FAILED 错误为就会被设置到状态寄存器。且密码不会被更改。如果正确，新密码以及密码长度就会存到 PWD 和 PWD\_LEN 寄存

器中。

注意：密码长度寄存器(PWD\_LEN)可以看出密码是否正确设置。为0的时候，就没有密码。不等于0的时候，上电后就会加锁。通过设置 LOCK/UNLOCK 位或者发送额外的加锁命令都可以在本次开机中立即加锁。

#### ● 重置密码

- 选定卡(CMD7)
- 定义块长度(CMD16)，8Bit + 当前密码长度
- 发送加锁/解锁命令(CMD42)，数据+16bitCRC。数据块应该带有模式(CLR\_PWD)，长度(PWDS\_LEN)，以及密码。如果密码和密码长度匹配，那么就会清除 PWD 中的密码，并把 PWD\_LEN 设置为 0。如果不匹配，就会设置 LOCK\_UNLOCK\_FAILED 错误。

#### ● 加锁卡

- CMD7
- CMD16，8Bit+当前密码长度
- CMD42，数据+CRC。数据块中表明 LOCK 模式，以及密码长度和密码。

如果卡内容匹配，就会加锁，并且状态寄存器中 card-locked 位会被设置。如果不对，那么 LOCK\_UNLOCK\_FAILED 位会被设置。

注意：设置密码和加锁可同时进行。只要在设置密码的序列里设置上 LOCK 位就可以了。加锁已经加锁的卡，或者加锁没有密码的卡，都会失败，LOCK\_UNLOCK\_FAILED 会被设置，除非是在密码设置密码或者修改密码的时候加锁。

#### ● 解锁卡

- CMD7
- CMD16
- CMD42

如果密码匹配，会解锁，card-locked 位会被清除。如果不匹配，LOCK\_UNLOCK\_FAILED。

注意：注意解锁只在本次开机中有效，只要密码还在，下次开机还是会自动加锁。

解锁没加锁的卡会失败，LOCK\_UNLOCK\_FAILED，除非在密码设置或者修改操作中。

### 4.3.7.2 参数和 CMD42 的结果

块的长度必须大于等于 CMD42 需求的数据结构，否则 CMD42 的结果可能未知，而卡可能会处于预期之外的状态。表 4-5 列举了 CMD42 的行为。Bit7-Bit4 是保留的，所以不看。这里假定 CMD42 使用的新旧密码都是正确的，不正确的错误没有在表 4-5 中列举。如果密码长度是 0，或者超过 128bit，那么会报错。

如果 CMD42 发生错误，LOCK\_UNLOCK\_FAILED(卡状态寄存器的第 24Bit)会被设置为 1(没在表 4-5 中写)。CMD42 的响应中的 CARD\_IS\_LOCKED(卡状态寄存器的第 25Bit)应该同表 4-5 中的卡状态一致。卡状态中 0->1 表示加锁，1->0 表示解锁。可以从 CMD42 之后的 CMD13 的回应中看到。LOCK\_UNLOCK\_FAILED 错误可以在 CMD42 或者后面的 CMD13 的响应中看到。

表 4-5 加锁解锁功能 (CMD42 的基本流程)

CMD42 参数				当前 状态	PWD_LEN 和 PWD	功能结果	卡状态	
Bit3	Bit2	Bit1	Bit0				Bit25	Bit24
上电后					存在	加锁	1	0
擦除   加解锁   清除   设置					不存在	解锁	0	0
1	0	0	0	锁定	存在	强制擦除(表 4-6)	表 4-6	表 4-6
1	0	0	0	未锁	存在	错误	0	1
1	0	0	0	未锁	不存在	错误	0	1
0	1	0	0	锁定	存在	错误	1	1
0	1	0	0	未锁	存在	加锁卡	0→1	0
0	1	0	0	未锁	不存在	错误	0	1
0	1	0	1	锁定	存在	修改密码并保持加锁	1	0
0	1	0	1	未锁	存在	修改密码并加锁	0→1	0
0	1	0	1	未锁	不存在	设置密码并加锁	0→1	0
0	0	1	0	锁定	存在	清除密码和长度并解锁	1→0	0
0	0	1	0	未锁	存在	清除密码和长度	0	0
0	0	1	0	未锁	不存在	错误(表 4-8, Note④)	0	1
0	0	0	1	锁定	存在	替换密码并解锁	1→0	0
0	0	0	1	未锁	存在	替换密码并解锁	0	0
0	0	0	1	未锁	不存在	设置密码并解锁	0	1
0	0	0	0	锁定	存在	解锁卡	1→0	0
0	0	0	0	未锁	存在	错误	0	1
0	0	0	0	未锁	不存在	错误	0	1
其他组合				无视	无视	错误(表 4-8, Note①)	0 或者 1	1

注意：当前没有密码的情况下，主机不能发送旧密码+新密码的命令，否则结果未知。

4.3.7.3 强制擦除

如果用户忘记了密码 (PWD 的内容)，可以强制擦除所有卡的内容。

- 选择卡 (CMD7)
- 定义模块长度 (CMD16) 为 1 字节 (8Bit 的加锁解锁命令)。发送数据一个字节 (的命令。数据块中标明擦除位 (只设置这一位)。

如果擦除位不是唯一被设置的，那么 LOCK\_UNLOCK\_FAILED 位 (bit24) 被设置，且不会擦除。如果命令被执行，卡上所有数据都被擦掉，包括 PWD 和 PWD\_LEN 寄存器的内容，最终处于解锁状态。强制擦除未加锁的卡会失败，然后，LOCK\_UNLOCK\_FAILED。

4.3.7.3.1 锁定卡的强制擦除功能

表 4-6 展示了强制擦除和写保护之间的关系。强制擦除不会擦除保护区域。并且卡擦除的过程中保持加锁状态，擦除完用户区域后变为解锁状态。类似的，组写保护也是这样。当发生擦除错误的时候，卡可以继续强制擦除，如果错误数据被破坏的话。

写保护：

- PWP：永久写保护 (CSD Bit13)
- TWP：临时写保护 (CSD Bit12)
- GWP：组写保护 (CMD28, CMD29, CMD30)

表 4-6 强制擦除写保护的卡(同写保护的关系)

CMD42 的参数				PWP	TWP GWP	功能结果	卡状态	
Bit3	Bit2	Bit1	Bit0				Bit2 5	Bit2 4
1	0	0	0	是	忽略	Error(表 4-8, Note②)	1	1
1	0	0	0	否	是	强制擦除, 清除写保护③	1->0	0
1	0	0	0	否	后	执行强制擦除	1->0	0

4.3.7.4 ACMD6 和锁定/解锁状态的关系

当卡被锁定的时候 ACMD6 是无效的, 总线宽度只能在卡解锁状态才能改变。

表 4-7 ACMD6 和加锁/解锁状态之间的关系

卡状态	总线模式	结果
解锁	1Bit	接受 ACMD6
加锁	1Bit	拒绝 ACMD6, 仍然是 1bit
解锁	4Bit	接受 ACMD6
加锁	4Bit	拒绝 ACMD6, 仍然是 1bit, 但是 CMD0 能让它回到 1bit

4.3.7.5 锁定卡接受的命令

已经锁定的卡接受下面的命令, 并且响应中会设置上 CARD\_IS\_LOCKED 状态位。

- 1) Basic Class (0)
- 2) Lock card Class (7)
- 3) CMD16
- 4) ACMD41
- 5) ACMD42

所有其他命令包括安全命令, 都会被认为是无效命令。

开机后, 主机可以通过读取 CMD7 或者 CMD13 的响应中的 CARD\_IS\_LOCKED 位来判断卡的锁定状态。

4.3.7.6 锁定/解锁卡的两种类型

目前有两种类型的支持加锁/解锁功能的卡, 一种是早期的 SD 卡, 一种是 V1.1 版本之后的。表 4-8 展示了这两种卡的不同。V1.01 的卡可以同时支持 Type1 和 Type2, 而 V1.1 和之后的卡只支持 Type2。

表 4-8 不同的加锁/解锁功能

Note	Type1 (早期版本)	Type2 (新版本)
①表 4-5 中	CMD42(0011b)→0001b CMD42(0111b)→0101b CMD42(0110b)→0010b 其他错误	全部错误
②表 4-6 中	强制擦除和永久写保护。 CARD_IS_LOCKED(1→0)。强制擦除优先级高	错误 永久写保护优先级高
③表 4-6 中	执行强制擦除,但是临时写保护和组写保护不清除。需要由主机清除	执行强制擦除,并清空临时写保护和组写保护
④表 4-5 中	CMD42(0010b、0110b)不会返回错误,卡状态 Bit24 为 0	错误,卡状态 Bit24 为 1

注意：主机可以下面的方法在不检查不同的情况下，使用两种类型的卡：

- (1) 主机不应该使用表 4-5 中返回错误的 CMD42 参数。①
- (2) 如果永久写保护为 1 的时候，主机不应该发送强制擦除命令，否则 Type1 可能因为用户忘记密码而无法再使用。②
- (3) 强制擦除后，如果临时写保护没有清除，主机应该清除掉。③

4.3.8 内容保护

详细的描述见“SD Specifications Part3 Security Specification”文档。所有 SD 卡安全相关的命令都应该在“数据传输模式”下操作。

如 SDMI 文档中定义，卡上存储的数据内容已经是加密存储，并且明码读写到卡上。不会对数据做任何处理，并且任何时候都可以没有限制的读取数据。相对于每个数据包(比如歌曲)都被存储在非保护区域，有一个特殊数据应该被存储到保护区域。任何对保护区域进行读写的命令，不管是 LCM(比如 PC)还是 PD(比如 SD 播放器)，都要在卡和连接设备之间做认证操作。当认证通过后，设备就可以从卡上进行读写数据操作了。当卡处于安全操作模式的时候，读写操作的参数和数据都是加密的。当命令结束后，卡会退出安全模式。

4.3.9 特定应用命令

4.3.9.1 特定应用命令—APP\_CMD (CMD55)

当卡收到这个命令时，卡会把接下来的命令作为特定应用命令(ACMD)来解析，ACMD 命令和标准命令有着相同的结构，甚至相同的 CMD 号。只要一个命令跟在 CMD55 之后，卡就会把它当作 ACMD。

APP\_CMD 的唯一影响就是，如果紧跟着的命令序号有一个 ACMD 相对应，那么就会使用非常规(non-regular)的版本。比如，如果卡定义了 ACMD13，但是没有定义 ACMD7，那么对于紧跟在 APP\_CMD 命令后面的命令，CMD13 会被当作非常规的 ACMD13 执行，但是 CMD7 就只作为常规的 CMD7 来执行。要想使用特定厂家 ACMD 命令，主机需要按照以下流程：

- 发送 APP\_CMD。响应里有 APP\_CMD 位设置信号，告诉主机现在要接受 ACMD 命令了。

- 发送要发的 ACMD。响应里的 APP\_CMD 位表明是作为 ACMD 解析的。如果发送了非 ACMD 命令，那么卡会作为普通的 sd 卡命令响应，APP\_CMD 位会被清除。
- 如果多个 CMD55 被连续发送，那么每个响应的 APP\_CMD 位都会被设置为 1。最后一个 CMD55 后面紧跟的命令会被当作 ACMD 命令。如果超过一个命令跟在 CMD55 后 (CMD55 除外)，只有第一个命令被当作 ACMD 命令。后面的都作为常规命令。

如果发送了一个无效命令，就会返回常规的命令错误。

从 SD 卡的协议来看，ACMD 号应该由厂家参照某些限制来定义。下面的 ACMD 是保留的，任何厂家都不应该使用它们：ACMD6, ACMD13, ACMD17-25, ACMD38-49, ACMD51。

### 4.3.9.2 通用命令—GEN\_CMD (CMD56)

GEN\_CMD (CMD56) 厂家特定以及可选的命令。本文档定义了这个命令。总线上的传输，GEN\_CMD 和“单块读/写”(CMD24、CMD7)命令是一样的，也只能在“transfer”状态使用。响应的类型是 R1。不同之处在于，参数指示了数据传输的方向(而不是地址)，并且数据块不是一个有效的存储数据，但是它有厂家特定的格式和含义。发送 CMD56 之前，需要让卡进入“transfer”状态。如果是标准 SD 卡，数据块的长度是 CMD16 中定义的 BLOCK\_LEN。如果是大容量卡，数据块的大小固定为 512Byte。

参数的 Bit0 表明了数据的传输方向：0 表示写，1 表示读。厂家可以定义特定的格式到参数的 bit31-1，以及这个命令的数据块内容。然而，我们需要考虑到，卡应该预防接收到未知格式的情况。

在使用 CMD56 之前，主机应该确认 CID 信息，这样卡才能支持 CMD56 的格式。

## 4.3.10 切换功能命令 [CMD6]

### 4.3.10.1 概述

切换命令 (CMD6) ①用于切换或扩展存储卡功能。现在定义了两种功能组：

- **卡访问模式：**12.5MB/s 接口速度(默认)，25MB/s 接口速度(高速)。
- **卡命令系统：**标准命令设置(默认)，电子商务命令设置或者厂家特殊命令设置

这个是从 V1.1 版本开始定义的。因此之前的版本不支持这个命令。在发送 CMD6 之前，主机应该检查 SCR 寄存器的“SD\_SPEC”区域来检查卡支持的版本。V1.1 之后的版本是强制要求支持这个命令的。

CMD6 只在“transfer”模式下有效。一旦通过切换命令选定，在一个电源周期后所有的功能只会返回默认功能 CMD6(每个功能组中的 Function 0 的 Mode 1 操作)或者 CMD0。执行电源周期或者发送 CMD0 会蓝卡复位到“idle”状态，所有的功能会切换回默认功能。

① CMD6 是 SD 卡用的，SDIO 使用 CCCR 来切换功能。

作为 CMD6 的响应，SD 卡会从 CMD 线返回 R1 响应，从 DAT 线返回 512Bit 的状态信息。从 SD 总线传输的角度来说，这是一个标准的单块读操作。这个命令的超时时间是 100ms，和读命令一样。如果 CRC 错误发生，主机会发送一个电源周期。

CMD6 功能切换区间在状态数据的结束位之后的 8 个时钟周期内。当 CMD6 改变了总线的表现后(比如，访问模式)，在切换命令传输接受后最少 8 个时钟内，主机就可以使用新的功能(在允许的范围内增大/减小时钟频率)(图 4-4)

CMD0 的响应中，切换区间在 CMD0 结束后的 8 个时钟周期内。当 CMD6 改变了总线的表



现(比如, 访问模式)后, 在 CMD0 之后的 8 个周期, 主机就可以启动初始化进程。

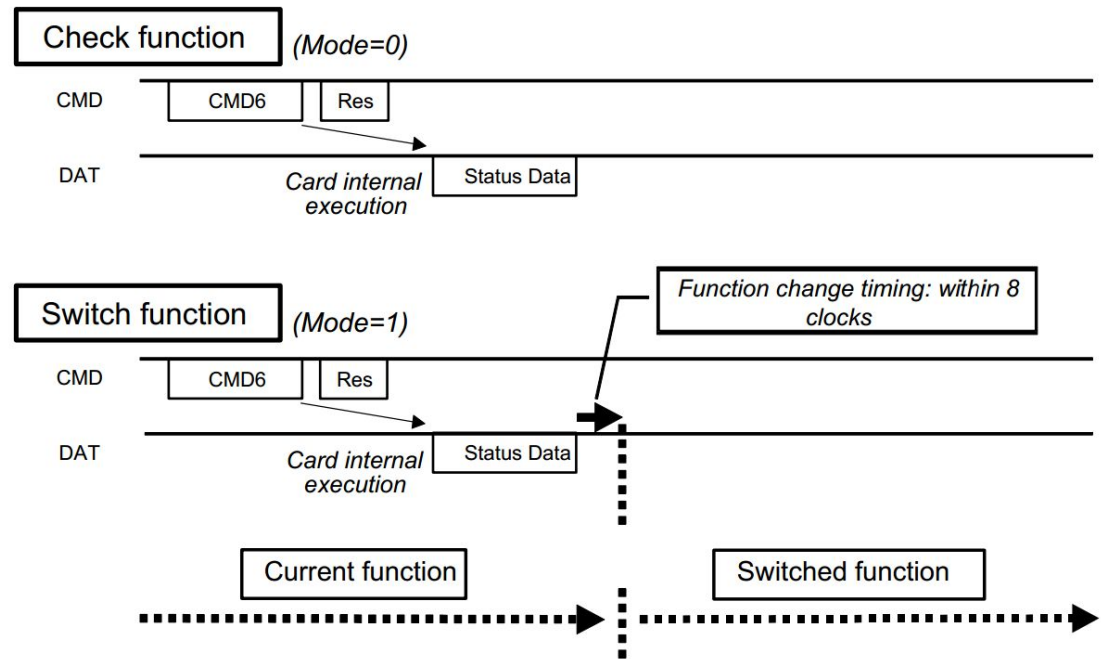


Figure 4-4: Use of Switch Command

CMD6 支持 6 个功能组(目前只用了两个), 每个功能组支持 16 种分支(功能)(目前全部加起来才有 3 个)。每个功能组只有一个功能可以被选择并生效。每个功能组的 Function 0 是默认功能。(兼容 V1.1 文档)

CMD6 可以在两种模式下使用:

- Mode 0 (查询功能) 用于检查卡是否支持特定的功能。
- Mode 1 (设置功能) 用于切换卡的功能。

#### 4.3.10.2 模式 0 运行—查询功能

CMD6 的 Mode 0 用来查询卡支持哪个功能, 以及标示在选择功能下卡的最大电流。

CMD6 的参数含义参考表 4-28: 切换功能命令(Class 10)。

查询操作通过设置命令的参数来完成, 步骤如下:

- 设置 Mode bit 为 0
- 每个功能组只选择一个功能。设置 0x0 选择默认功能。通过表 4-9 中的值来选择特定功能。选择 0xF 会保持功能组当前选定的功能。
- 当查询的功能就绪的时候, 卡会返回查询的功能的序号, 如果忙, 卡会返回当前功能的序号(见表 4-12)。

在查询的响应中, 切换功能会返回下面三种状态(见表 4-10)

- 每个功能组中支持的功能
- 每个功能组中, 卡将要切换到那个功能。这个值可主机发送有效选择时的那个参数一致, 如果参数无效, 则是 0xF。
- 在选定模式下的最大电流量。如果其中有一个功能是错误的, 那么返回值为 0

### 4.3.10.3 模式 1 运行—设置功能

CMD6 的 mode 1 用于切换卡的功能。

切换到一个新功能的步骤：

- 设置 Mode bit 为 1
- 每个功能组选择一个功能。设置 0x0 选择默认功能。除了需要改变的功能外，推荐为所有已选功能设置 0xF(没有影响)。0xF 会保持当前功能。
- 当一个功能因为 busy 不能被切换的时候。卡会返回当前的功能序号(不返回 0xF)，其他功能组的其他功能功能会继续切换。

在设置功能的响应中，切换功能会返回 3 种状态：

- 每个功能组支持的功能
- 切换命令的结果中的功能。如果一个或者多个功能无效，设置的值会被忽略，并不会做改变(就跟主机设置所有功能组 0xF 一样)。功能的无效选择会返回 0xF。
- 选择功能的最大电流。如果选择功能中有一个是错误的，那么返回值是 0。

表 4-9 有效功能

参数位	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
组序号	6	5	4	3	2	1
功能名称	保留	保留	保留	保留	命令系统	访问模式
0x0	默认 (V1.01)					
0x1	保留	保留	保留	保留	For eC	高速
0x2	保留	保留	保留	保留	保留	保留
0x3	保留	保留	保留	保留	保留	保留
0x4	保留	保留	保留	保留	保留	保留
0x5	保留	保留	保留	保留	保留	保留
0x6	保留	保留	保留	保留	保留	保留
0x7	保留	保留	保留	保留	保留	保留
0x8	保留	保留	保留	保留	保留	保留
0x9	保留	保留	保留	保留	保留	保留
0xA	保留	保留	保留	保留	保留	保留
0xB	保留	保留	保留	保留	保留	保留
0xC	保留	保留	保留	保留	保留	保留
0xD	保留	保留	保留	保留	保留	保留
0xE	保留	保留	保留	保留	厂家指定	保留
0xF	无影响					

### 4.3.10.4 切换功能状态

切换功能状态就是是返回的数据块，其中包含了功能和电流信息。块的长度被预设为 512Bit，因此 SET\_BLK\_LEN 命令也就不是必须的了。表 4-10 描述了状态数据寄存器。

响应的状态位包含了功能组的信息。最大电流也只会用于通过这个命令添加的新功能。这种情况下，当所有功能被设置到默认状态并且可被兼容 V1.01 的主机使用的时候，CSD 寄存器中的 VDD\_R\_CURR\_MIN, VDD\_W\_CURR\_MIN, VDD\_R\_CURR\_MAX 和 VDD\_W\_CURR\_MAX 的值提供了电流值。

表 4-10 状态数据结构体

位	描述	宽度
511:496	在[399:376]位的功能的最大电流。(0: 错误, 1:1mA, 2:2mA, ..., 65535mA) 对应电流的电压在 ACMD41(SD 卡)或者 CMD5(SDIO 卡)中定义。 如果功能已经切换, 那么最大电流就代表整个卡的电流(存储部分)。 主机应该检查最大电流, 并且在 Mode 1 操作前表明它支持这个电流。 最大电流平均超过 1 秒	16
495:480	功能组 6, 信息。如果 bit[i]被设置, 那么功能 i 支持	16
479:464	组 5 同上	16
463:448	组 4 同上	16
447:432	组 3 同上	16
431:416	组 2 同上	16
415:400	组 1 同上	16
399:396	Mode 0 — 组 6 中可以被切换的功能 Mode 1 — 组 6 中, 作为切换命令结果的功能。0xF 表示参数错误。	4
395:392	组 5 同上	4
391:388	组 4 同上	4
387:384	组 3 同上	4
383:380	组 2 同上	4
379:376	组 1 同上	4
375:368	数据结构版本 00h —bit511:376 定义 01h —bit511:272 定义 02h~FFh —保留	8
367:352	组 6 的功能 busy 状态。如果 bit[i]被设置, 那么功能 ibusy。 这些值可以在 Mode 0 和 Mode 1 读取	16
351:336	组 5 同上	16
335:320	组 4 同上	16
319:304	组 3 同上	16
303:288	组 2 同上	16
287:272	组 1 同上	16
271:0	保留(全 0)	272

#### 4.3.10.4.1 Busy 状态指示功能

[367:272]的每一位都代表对应功能的 busy 状态；0 表示 ready，1 表示 busy。当状态位 busy 的时候，主机不能改变当前功能。切换命令 Mode 1 只能对 ready 的功能使用。

如果 Mode 1 操作中的功能切换失败，并且在响应中返回了当前功能序号，那么就认为这个功能 busy。但是 Mode 1 的操作可能会影响功能的执行。而使用 Mode 0 操作来检查函数的 busy 状态，因为他不会影响功能行为，尤其是功能组 2。

表格 4-5 “命令系统”的 busy 状态

[illegible]

VS: 厂家特定的 Busy 状态: 0-ready, 1-busy  
eC: eC 的 busy 状态: 0-ready, 1-busy

4.3.10.4.2 数据结构版本

数据结构版本表明了“切换功能状态”的有效 bit 区域。卡可以设置为 00h 或者 01h。当这个值设置为 01 的时候, busy 状态指示才是有效的。

表 4-11 数据结构版本

数据结构版本	数据结构的状态区域
00h	511:376
01h	511:272
02h-FFh	保留

4.3.10.4.3 切换命令的功能列表

表 4-12, 表 4-13 和表 4-14 列出了可能的功能切换。

“参数”: 切换命令 (bit23:0) 参数中指定的 4bit 码。

“Busy 状态”: 表明功能正忙

“状态码”: 状态数据结构体中的 4Bit 码

表 4-12 Mode 0 的状态码与支持的功能组

参数	Busy 状态	状态码	内容
0	忽略	0	此状态表明默认功能, 总是支持的。
支持功能	Ready	=Arg	支持参数指定的功能, 并可以切换
	Busy	当前选择	支持制定功能, 但当前 busy, 不能切换
不支持的	忽略	Fh	此状态表明指定的功能不支持
Fh	忽略	当前选择	此状态指明当前的功能。

表 4-13 Mode 1 的状态码与支持的功能组

参数	Busy 状态	状态码	内容
0	忽略	0	默认功能, 总是可以切换
支持功能	Ready	=Arg	参数指定的功能, 已经成功切换
	Busy	当前选择	退出切换, 保持当前选择的功能
不支持的	忽略	Fh	如果有一个功能组返回错误 (Fh), 所有的切换都退出。状态数据结构体中的剩余部分忽略。
Fh	忽略	当前选择	此状态指明当前选择的功能。

表 4-14 Mode0 Model 与不支持的功能组

参数	Busy 状态	状态码	内容
0	忽略	0	总是 0
Eh-1h	忽略	Fh	总是 Fh
Fh	忽略	0	总是 0

4.3.10.5 CMD6 数据和其他命令之间的关系

在 CMD6 传输期间, 卡可能收到只使用 CMD 线的命令 (CMD12, CMD13 等), 但是它的响应

和结果没有定义。

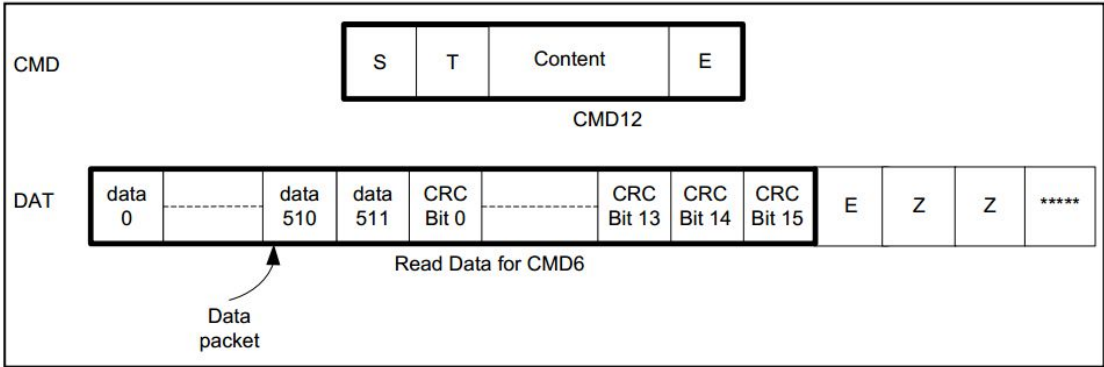
注意：主机在 CMD6 传输期间，最好是不要发送任何命令。如果主机不能获得 CMD6 的有效数据，它会发送 CMD0，并尝试重新初始化。

● CMD6 数据和 CMD12 的关系

■ Case1：没完成的状况 (卡没有输出全部的数据)

如果在 CRC bit15 之前发送了 CMD12 的结束位，那么 CMD12 会中止掉 CMD6，卡会中止掉 CMD6 的数据传输。卡的行为就会不可预料，只能通过 CMD0 来让卡恢复正常。数据线上，主机命令的最后一位后面有一个数据位和结束位。

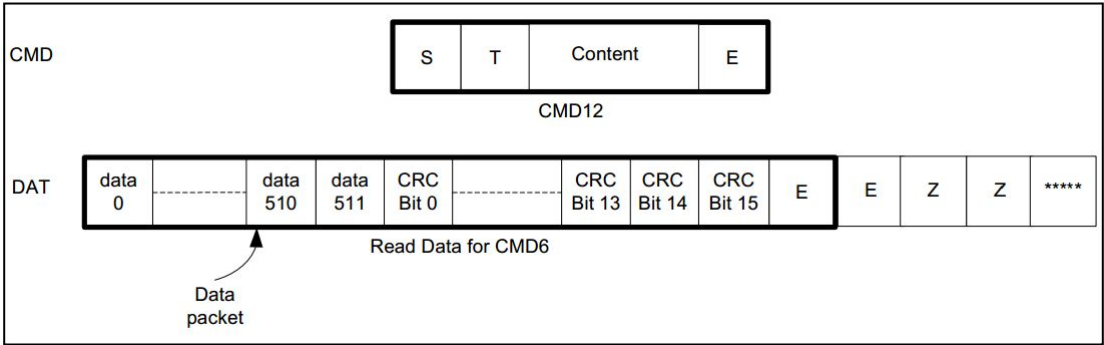
图 4-6 CMD6 期间发送 CMD12 Case 1



■ Case2：完成的状况 (卡输出了所有的数据)

此时卡应该完成 CMD6 的执行，并且这个行为是有保障的。完成的状况包含了比图 4-7 更晚的 CMD12。数据线上，主机的命令后会有一个结束位

图 4-7 CMD6 期间发送 CMD12 Case 2



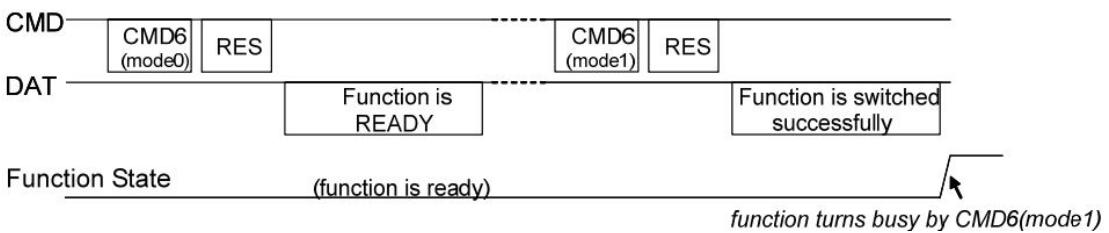
注意，主机最好不要在 CMD6 期间发送 CMD12。

4.3.10.6 切换功能流程实例

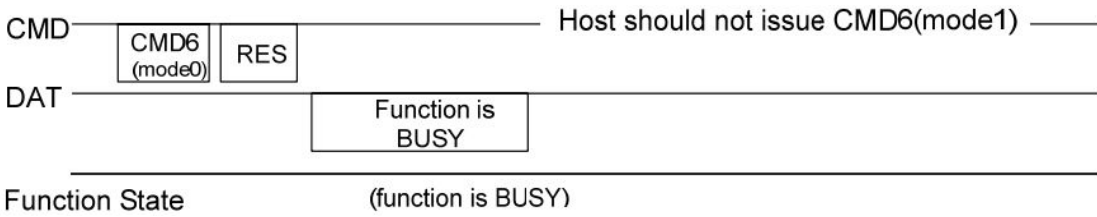
图 4-8 的 (a) (b) (c) 展示了切换功能序列的 3 种可能的情况。依据功能的 busy 状态，功能改变与 CMD6 的序列是异步进行的。主机需要能够应付这 3 种情况。

图 4-8 切换功能流程实例

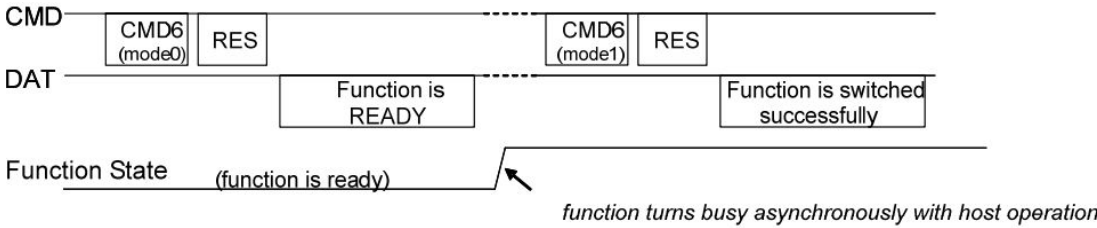
(a) 功能 Ready，切换成功



(b) 功能 Busy，不能切换

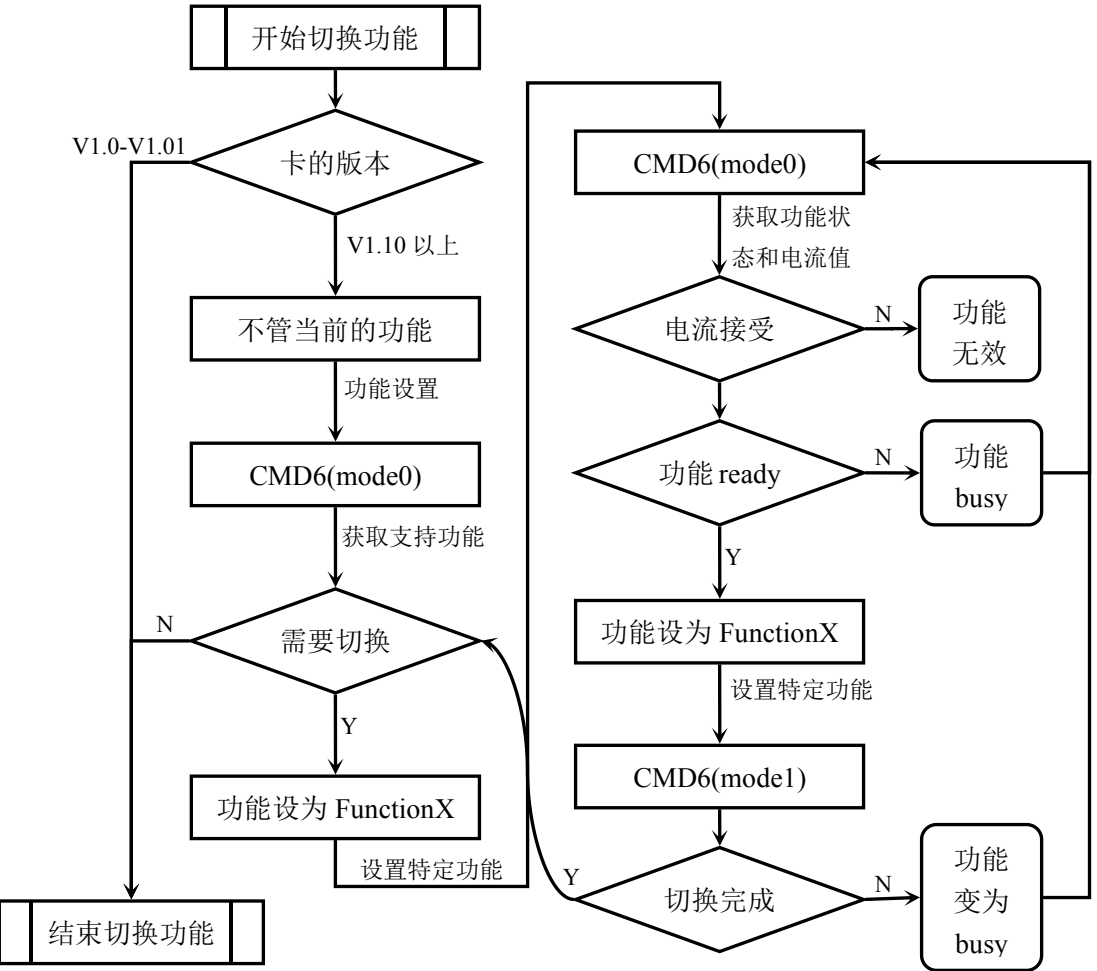


(c) 在 CMD6(mode 0)和 CMD6(mode 1)期间，功能从 Ready 到 Busy



在切换功能之前，主机应该先发送 CMD6(mode0) 来获取 busy 状态和电流值。如果电流值不能接受，那么主机应该找到符合主机电流限制的另一个值。如果功能状态是 ready，那么主机可以发送 CMD6(mode1) 来切换功能，如 (a) 和 (c)。如果功能状态是 busy，那么主机不应该发送 CMD6(mode1)，如 (b)。图 4-8 (c) 表示了，在主机从 CMD6(mode0) 响应收到 ready 状态后，功能状态又变为了 busy，因此导致 CMD6(mode1) 无法执行的情况。

图 4-9 切换功能流程图(主机应当遵循这个序列)





4. 3. 10. 7 查询实例

卡状况：  
支持的功能 = 命令系统：eC(0x1)，访问模式：高速(0x1)  
当前的功能 = 命令系统：eC(0x1)，访问模式：默认(0x0)  
切换举例：命令系统：eC => 默认，访问模式：默认 => 高速

Case1: 查询功能无误

CMD6 参数 = ‘0000 0000 1111 1111 1111 1111 0000 0001’ (参数含义，见表 4-28)

读取数据= [511:496]= ‘0000 0000 0100 0000’ (=64mA)

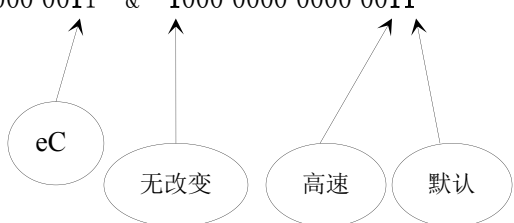
[495:400]= ‘1000 0000 0000 0001’ & ‘1000 0000 0000 0001’ & ‘1000 0000 0000 0001’  
& ‘1000 0000 0000 0001’ & ‘1000 0000 0000 0011’ & ‘1000 0000 0000 0011’

[399:376]= ‘0000 0000 0000 0000 0000 0001’

[375:368]= 数据结构版本

[367:272]= 组 1-6 的功能 busy 状态

[271:0] = 保留(全部 0)



Case2: 查询功能有错

CMD6 参数 = ‘0000 0000 1111 1000 1111 0010 0000 0001’

读取数据=

[511:496]= ‘0000 0000 0000 0000’ (代表错误)


[495:400]= ‘1000 0000 0000 0001’ & ‘1000 0000 0000 0001’ & ‘1000 0000 0000 0001’  
& ‘1000 0000 0000 0001’ & ‘1000 0000 0000 0011’ & ‘1000 0000 0000 0011’

[399:376]= ‘0000 1111 0000 1111 0000 0001’

[375:368]= 数据结构版本

[367:272]= 组 1-6 的功能 busy 状态

[271:0] = 保留(全部 0)



4. 3. 10. 8 切换实例

卡状况：  
支持的功能 = 命令系统：eC(0x1)，访问模式：高速(0x1)  
当前的功能 = 命令系统：eC(0x1)，访问模式：默认(0x0)  
切换举例：命令系统：eC => 默认，访问模式：默认 => 高速

Case3: 切换功能无误

CMD6 参数= ‘1000 0000 1111 1111 1111 1111 0000 0001’

读取数据= [511:496]= ‘0000 0000 0100 0000’ (=64mA)

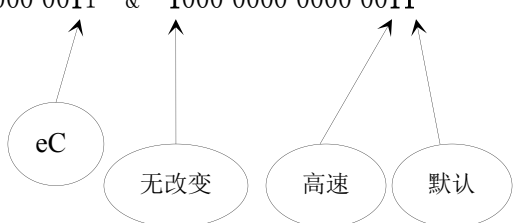
[495:400]= ‘1000 0000 0000 0001’ & ‘1000 0000 0000 0001’ & ‘1000 0000 0000 0001’  
& ‘1000 0000 0000 0001’ & ‘1000 0000 0000 0011’ & ‘1000 0000 0000 0011’

[399:376]= ‘0000 0000 0000 0000 0000 0001’

[375:368]= 数据结构版本

[367:272]= 组 1-6 的功能 busy 状态

[271:0] = 保留(全部 0)



#### Case4: 切换功能有错

CMD6 参数 = ‘1000 0000 1111 1000 1111 0010 0000 0001’

读取数据=

[511:496]= ‘0000 0000 0000 0000’ (代表错误)

[495:400]= ‘1000 0000 0000 0001’ & ‘1000 0000 0000 0001’ & ‘1000 0000 0000 0001’  
& ‘1000 0000 0000 0001’ & ‘1000 0000 0000 0011’ & ‘1000 0000 0000 0011’

[399:376]= ‘0000 1111 0000 1111 0000 0001’

[375:368]= 数据结构版本

[367:272]= 组 1-6 的功能 busy 状态

[271:0] = 保留(全部 0)

错误

### 4.3.11 高速模式(25M/s 接口速度)

尽管 V1.01 的卡已经能支持到 12.5M/s 的接口速度,但是卡的容量在不断变大,所以 25M/s 也是必须的。

既然接口速度要达到 25M/s,那么时钟需要提高到 50MHz,而 CLK/CMD/DAT 线的型号周期,电路条件也都需要考量和修改。

上电后,SD 卡处于默认速度模式,通过 CMD6, V1.10 及后续版本 SD 卡可以切换到高速模式。高速模式是访问模式组的一个功能(表 4-9)。支持高速操作。

因为同时控制两张工作在不同的时钟模式(默认和高速)的卡是不可能的。所以主机应该只驱动一张卡。CLK/CMD/DAT 信号应该一对一的连接主机和卡。

### 4.3.12 命令系统

CMD34-37, CMD50, CMD57 是为了 SD 命令系统扩展(通过切换命令)而保留的。各种命令系统功能组之间的切换会改变这些命令的解释和相关的总线传输(比如,无数据的命令,单块读,多块写,等)。支持命令系统是可选项。

- 当“标准命令设置”(默认功能 0x0)被选中的时候,这些命令不会被卡认可,会被认为是非法命令(V1.01 文档中有定义)
- 当“厂家特定设置”(功能 0xE)被选中的时候,这些命令就作为厂家特定的。这不是本标准定义的,而是由各个厂家来定义
- 当“电子商务”(功能 0x1)被选中的时候,这些命令的行为就会由“SD Specifications Part A1: Mobile commerce Extension Specification”规范。

当这些扩展命令被使用的时候,应该特别关注命令设置功能,否则,主机命令可能会被错误解析。

所有 SD 卡的其他命令(除了保留命令)总是有效的,并且会照着本标准被执行,不管当前选定何种命令设置。

### 4.3.13 发送接口条件命令(CMD8)

CMD8(发送接口条件命令)是用来依照 SD2.0 标准初始化 SD 卡的。CMD8 要在 SD 卡处于“idle”状态下使用。此命令有两个功能:

- **电压检测:**  
检测卡是否能在主机提供的电压下工作。

■ 使能已存在命令的扩展和响应：

恢复 CMD8 就能够通过从定义之前预留的 bit，为一些已存在的命令扩展新的功能。ACMD41 就是被扩展用来支持大容量卡的。

表 4-15 CMD8 的格式

位 bit	47	46	[45:40]	[39:20]	[19:16]	[15:8]	[7:1]	0
宽(bit)	1	1	6	20	4	8	7	1
值	‘0’	‘1’	‘001000’	‘00000h’	x	x	x	‘1’
描述	起始位	传输位	命令号	保留位	支持电压	检测模式	CRC7	结束位

支持电压	值的定义
0000b	未定义
0001b	2.7-3.6V
0010b	预留给低电卡
0100b	保留
1000b	保留
其他	没定义

当卡在“idle”状态时，主机应该先发送 CMD8，再发送 ACMD41。参数中“电压支持”bit 被设置为主机支持的电压，而“检测模式”bit 被设置为任意的 8bit 模式(推荐使用‘10101010b’)。

卡会检查自己能不能在主机提供的电压下工作。如果能够，就会返回 R7 响应。响应参数中卡反馈了电压范围和检测模式设置。

表 4-16 SD 模式下 CMD8 的卡操作

命令参数检查					卡的响应①					
序号	保留	VHS	模式	CRC	序号	版本	保留	VCA	模式	CRC
忽略	忽略	忽略	忽略	错误	无响应，CRC 错误标示在下一个命令中					
非 8	忽略	忽略	忽略	正确	决定于 CMD 号					
=8	忽略	不配②	忽略	正确	无响应					
=8	忽略	匹配②	忽略	正确	8	Ver=0	0	回写	回写	计算

①：这里是卡实际返回的内容(响应传输中不包含错误)

②：匹配意思是同时满足下面的 a)和 b)。不匹配是其他情况。

a) VHS(支持的电压)中只有 1 个 bit 被设置了

b) 卡支持主机提供的电压

### 4.3.14 大容量 SD 卡的命令功能区别

存储访问命令包含了块读命令(CMD17, CMD18)，块写命令(CMD24, CMD25)，块擦除命令(CMD32, CMD33)。

下面是标准卡和大容量卡之间的存储访问命令的区别：

■ 命令参数

大容量卡中，内容访问命令的 32bit 参数，使用块地址格式。块长度固定为 512Byte。标准卡中，内容访问命令的 32bit 参数使用的是字节地址格式。块长度由 CMD16 决定。

a) 标准卡的 0001h 参数表示字节地址 0001h，大容量卡表示第 0001h 个 block

b) 标准卡的 0200h 参数表示直接地址 0200h，大容量卡表示第 0200h 个 block

- 局部访问和位移访问

大容量卡因为是以块为单位的，所以不支持这种，只允许块地址访问

- 设置块长度

当以块地址模式进行存储读和写命令时，不管 CMD16 设置的块长是多少，块的长度都固定为 512Byte。就是说块长度的设置不会影响存储访问命令。CMD42(锁定/解锁)不属于存储访问命令，因此块长度可以通过 CMD16 进行设置。不管卡容量多少，只要块长度设置大于 512Byte，都会报错 BLOCK\_LEN\_ERROR。

- 写保护组

大容量卡不支持写保护组。因此 CMD28，CMD29，CMD30 会报错 ILLEGAL\_COMMAND。

## 4.4 时钟控制

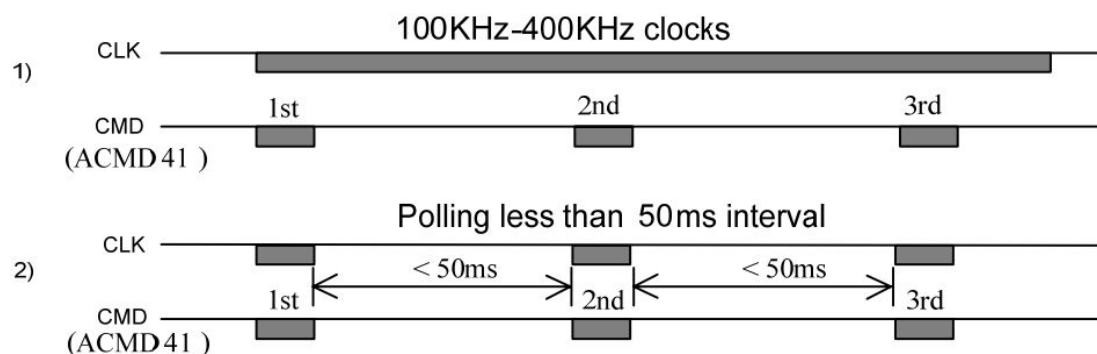
主机可以使用 SD 卡时钟信号来将卡改变到节电模式，或者控制总线上的数据流(避免不足或者超出的情况)。主机可以减小时钟频率，或者关闭时钟。比如，当主机带有 512Byte 的数据缓存，却想要向 1kByte 的卡上写块。因此，为了保持连续的数据传输，从卡这边来看，卡的时钟需要在第一个 512Byte 后关闭时钟。然后主机将另外 512Byte 数据填写到缓存中。当主机的写块的后半部分 ready 后，将会重启时钟，并继续发送数据。这样，卡就不会意识到数据传输的中断。

有一些限制，主机需要注意：

- 总线时钟可在任何时候改变(要小于最大数据传输频率和本规范定义的标示频率)
- 上面的说法有个例外，ACMD41(SD\_APP\_OP\_COND)。当 ACMD41 发送后，主机应该执行下面的 1) 和 2) 两个步骤，直到卡变为 ready 状态。

1) 发送连续的时钟，频率在 100KHz~400KHz

2) 如果主机想要停止时钟，通过 ACMD41 循环设置 busy 位，小于 50ms 间隔。



- 很明显，在输出数据或者拿到回应时，卡的时钟应该保持连续。当最后一个 SD 总线处理结束后，在关闭时钟之前，卡需要提供 8 个时钟周期，以便让卡完成操作。下面是各种总线处理的列表：

- ◆ 命令不需要响应的。在主机命令结束位后 8 个时钟
- ◆ 命令需要响应的。在响应结束位后 8 个时钟
- ◆ 读数据。最后一个数据块的结束位后 8 个时钟
- ◆ 写数据。CRC 状态得到后的 8 个时钟

- 主机可以关闭 busy 状态的卡的时钟。卡会继续完成编程操作，即使没有主机的时钟。但是主机需要提供一个时钟沿，以便让卡关闭它的 busy 型号。如果没有时钟沿，卡会永久拉低 DAT 线(除非之前已经使用 CMD7 让它进入 disconnect 状态)。

## 4.5 循环冗余码(CRC)

CRC 适用于保护 SD 总线上的 SD 卡命令，响应和数据传输免于出错的。命令线上，每个命令都会产生 CRC，而每个响应也都会检查 CRC。对于数据块来说，每一个传输的块都会产生 CRC。CRC 的产生和检查描述如下：

### ● CRC7

CRC7 的检测是用于所有的命令，所有的响应(除了 R3)，以及 CSD 和 CID 寄存器。CRC7 是一个 7bit 的数，计算方法如下：

$$G(x) = x^7 + x^3 + 1。$$

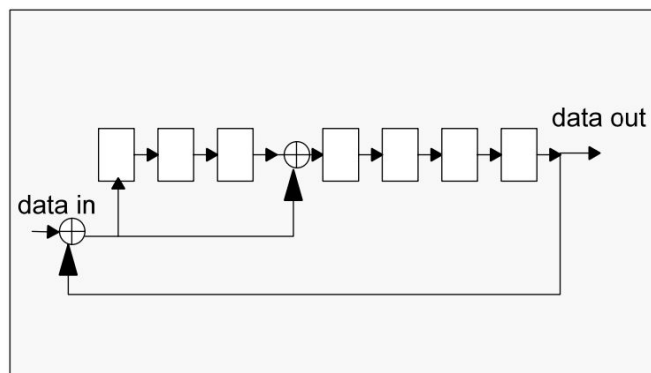
$$M(x) = (\text{第 1bit}) * x^n + (\text{第 2bit}) * x^{(n-1)} + \dots + (\text{n+1bit}) * x^0$$

$$\text{CRC}[6:0] = [(M(x) * x^7) / G(x)] \text{ 的余数}$$

第 1bit 是相对应 bit 串(命令，响应，CID，CSD)最左边的 bit。

n 是 CRC 保护的 bit 数-1。命令和响应的保护数是 40(n=39)，CSD 和 CID 是 120(n=119)。

图 4-10 CRC7 的生成器和检查器



### ● CRC7 实例

命令和响应的 CRC 断。

CMD0 (参数=0) -> 01 000000 00 "1001010" 1

CMD17(参数=0) -> 01 010001 00 "0101010" 1

CMD17 的响应 -> 00 010001 00000000000000000000000001001000000000 "0110011" 1

### ● CRC16

使用数据线的时候，CRC16 用于有效保护块传输模式。CRC 检查的是一个 16bit 的值，计算方法如下：

$$G(x) = x^{16} + x^{12} + x^5 + 1。$$

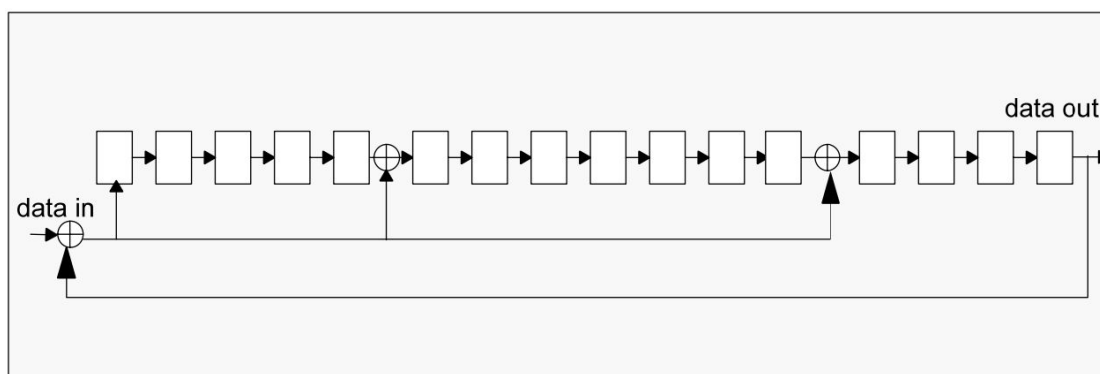
$$M(x) = (\text{第 1bit}) * x^n + (\text{第 2bit}) * x^{(n-1)} + \dots + (\text{n+1bit}) * x^0$$

$$\text{CRC}[15:0] = [(M(x) * x^{16}) / G(x)] \text{ 的余数}$$

第一个 bit 是对应块的第一 bit 数据。n 表示传输数据块的 bit 数减 1(块长 512Byte，n=4095)。G(x) 是一个标准的 CCITT 多项式。这个数有个最小值 4，最大用于 2048byte(n<16383)。

CRC16 应该单独的用在每个数据线上，即使是宽总线模式，也要每跟数据线单独校验。

图 4-11 CRC16 的生成器和检查器



## ● CRC16 实例

512 字节的 0xFF 数据 -> CRC16 = 0x7FA1

## 4.6 错误条件

### 4.6.1 CRC 和非法命令

所有的命令都通过 CRC 位来保护。如果寻址卡的 CRC 检查错误，卡不会反馈，命令也不会执行。卡不会改变自身的状态，且在状态寄存器中设置 COM\_CRC\_ERROR 位。

类似的，如果收到无效命令，卡也不会改变状态，不会响应，并且在状态寄存器中设置 ILLEGAL\_COMMAND 位。只有正确的状态分支会显示在状态图中(图 4-1 和图 4-3)。

表 4-29 包含了一个完成的状态转换描述。

有几种不同的非法命令：

- 卡不支持的组中的命令(比如只读卡里的写命令)
- 当前状态不允许的命令(Transfer 模式下的 CMD2)
- 没有定义的命令(CMD5)

### 4.6.2 读，写和擦除超时情况

卡应该在规定的时间内完成命令的处理，或者放弃并返回错误。如果主机没有在规定的时间收到任何回应，那么会认为这张卡不会回应，然后尝试恢复(复位，重新上电，放弃)。

#### 4.6.2.1 读

对于标准卡来说，读操作的超时时间是 100 倍的标准访问时间或者是 100ms(取较小的)。读访问时间是 CSD 的参数 TAAC 和 NSAC 的和(见 5.3)。如果是单独的读操作，这些卡的参数定义了读命令的结束位和数据块的起始位之间的延时。如果是多块的读操作，他们也定义了块与块之间的标准延迟。

对于高速卡来说，TAAC 和 NSAC 是写死的值。主机应该使用 100ms(最小)作为超时时间，而不是使用 TAAC 和 NSAC 的和。

#### 4.6.2.2 写

对于标准卡来说，超时时间应该是 100 倍的标准操作时间，或者是 250ms(取较小的)。CSD 中的 R2W\_FACTOR 区域用来指示标准的操作时间，这个值乘以读访问时间就是写访问时



间。所有的写操作都是这个时间 (SET (CLR) \_WRITE\_PROTECT, PROGRAM\_CSD, 以及块读命令)。

对于高速卡来说, R2W\_FACTOR 也是一个写死的值。所有写操作 Busy 的最大值是 250ms。主机应该使用 250ms 作为单块或多块写操作的超时时间, 而不使用 R2W\_FACTOR。

### 4.6.2.3 擦除

如果卡在 SD 状态支持擦除超时参数, 主机应该使用他们来确定擦除超时 (见 4.10.2)。如果卡不支持这些参数, 擦除超时可以通过块写操作延迟来估算。

擦除命令的持续时间, 可以通过写的块的数目 (WTITE\_BL) 来估算, 乘以 250ms 就行。

## 4.7 命令

### 4.7.1 命令类型

Sd 卡定义了 4 种命令类型:

- **广播命令 (bc), 无响应** — 广播命令只有当所有的 CMD 线都一起连到主机上时才会用。如果是分开的, 那么每张卡单独处理命令。
- **广播命令, 带响应 (bcr)** — 所有卡同时响应, 既然 SD 卡没有开漏模式, 这种命令应该是所有的命令线都是分开的, 命令也会每张卡分开接收和响应。
- **寻址命令 (ac, 点对点)** — 没有数据在 DAT 线上
- **寻址数据传输命令 (adtc)** — 有数据在 DAT 线上

所有命令和响应都是通过 SD 卡的 CMD 线发送的。命令的发送总是从最左边的那一位开始。

### 4.7.2 命令格式

所有的命令都有一个固定的长度 48bit, 在 25MHz 的频率下需要 1.92us, 在 50Mhz 的频率下需要 0.96us。

表 4-17 命令格式

Bit 位	47	46	[45:40]	[39:8]	[7:1]	0
宽度 (bit)	1	1	6	32	7	1
值	'0'	'1'	x	x	x	'1'
描述	起始位	传输位 ✓	命令号 ✓	参数 ✓	CRC7 ✓	结束位

命令总是以起始位开始 (0), 第 2bit 是传输方向 (host=1)。之后 6bit 是命令的序号, 这是个二进制的数 (0~63)。有些命令需要参数 (比如地址), 这个是 32bit, 因具体命令而不同。所有的命令都用 CRC 进行保护 (见 4.5)。最后是结束位 (1)。所有命令和参数都在 [表 4-19] ~ [表 4-28] 中。

### 4.7.3 命令类

Sd 卡系统的命令设置被分为多个类 (见表 4-18)。

每个类支持一种“卡的功能设置”。

表 4-18 定义了卡支持命令的 Card Command Class (CCC) 的设置。CCCbit, 对应一个命令序号, 被设置为 1。CCC 中的类 (包括强制命令) 总是被设置为 1。带有特定功能的卡可能需要支持某些可选命令。比如 Combo 卡应该支持 CMD5。

Class 0, 2, 4, 5, 7, 8 是强制要求所有 sd 卡都要支持的。其他的类可选。每个卡支持的卡命令类 (CCC) 写在 CSD 的参数中, 用来告诉主机如何访问卡。

表 4-18	CCC	0	1	2	3	4	5	6	7	8	9	10	11
支持命令	类描述	基础	保留	块读	保留	块写	擦除	写保护	锁卡	特定应用	I/O 模式	切换	保留
CMD0	强制	+											
CMD2	强制	+											
CMD3	强制	+											
CMD4	强制	+											
CMD5	可选										+		
CMD6	强制											+	
CMD7	强制	+											
CMD8	强制	+											
CMD9	强制	+											
CMD10	强制	+											
CMD12	强制	+											
CMD13	强制	+											
CMD15	强制	+											
CMD16	强制			+		+			+				
CMD17	强制			+									
CMD18	强制			+									
CMD24	强制					+							
CMD25	强制					+							
CMD27	强制					+							
CMD28	可选							+					
CMD29	可选							+					
CMD30	可选							+					
CMD32	强制						+						
CMD33	强制						+						
CMD34-37	可选											+	
CMD38	强制						+						
CMD42	强制								+				
CMD50	可选											+	
CMD52	可选										+		
CMD53	可选										+		
CMD55	强制									+			
CMD56	强制									+			
CMD57	可选											+	
ACMD6	强制									+			
ACMD13	强制									+			
ACMD22	强制									+			
ACMD23	强制									+			
ACMD41	强制									+			
ACMD42	强制									+			
ACMD51	强制									+			

注意:

- (1): 写命令的强制性只是针对可写的卡
- (2): CMD6, CMD34-37, CMD50, CMD57 是在 1.1 版本中定义的
- (3): CMD8 是 2.0 版本新定义的
- (4): CMD42 在 V1.01 和 V1.10 版本中是可选的, 在 2.0 版本中是强制的

#### 4.7.4 详细命令描述

下表详细描述了 SD 卡总线命令。响应 R1-R3, R6 在 4.9 章中定义。CID, CSD, DSR 寄存器在第 5 章中定义。卡应该忽视参数中的填充位以及保留位。

表 4-19 基础命令(Class 0)

命令序号	类型	参数	响应	缩写	描述
CMD0	bc	[31:0]填充位	—	GO_IDLE_STATE	复位所有的卡到 idle 状态
CMD1	保留				
CMD2	bcr	[31:0]填充位	R2	ALL_SEND_CID	通知所有卡通过 CMD 线返回 CID 值
CMD3	bcr	[31:0]填充位	R6	SEND_RELATIVE_ADDR	通知所有卡发布新 RCA
CMD4	bc	[31:16]DSR [15:0]填充位	—	SET_DSR	编程所有卡的 DSR
CMD5	为 I/O 卡保留(参考 SDIO Card Specification)				
CMD7	ac	[31:16]RCA [15:0]填充位	R1b (已选定卡)	SELECT/DESELECT_CARD	这个命令在“stand-by”和“transfer”状态之间使用, 以及“programming”和“disconnect”状态之间使用。在这两种情况下, 地址匹配就被选中, 不匹配就被取消选中。address 0, 取消所有卡的选中。当 RCA=0 时, 主机可能做下面的事情: -使用其他的 RCA 号来取消卡 -重新发送 CMD3 来改变卡 RCA 号, 使它不为 0。然后在用 RCA=0 来取消选择。
CMD8	bcr	[31:12]保留位 [11:8]VHS [7:0]检查模式	R7	SEND_IF_COND	发送 SD 卡接口条件, 包含了主机支持的电压信息, 并询问卡是否支持。保留位应该设置为 0。
CMD9	ac	[31:16]RCA [15:0]填充位	R2	SEND_CSD	选定的卡发通过命令线送卡的特殊数据(card-specific data 即 CSD)。
CMD10	ac	[31:16]RCA [15:0]填充位	R2	SEND_CID	选定的卡通过命令线发送卡标识(CID)
CMD11	保留				
CMD12	ac	[31:0]填充位	R1b	STOP_TRANSMISSION	强制卡停止传输
CMD13	ac	[31:16]RCA [15:0]填充位	R1	SEND_STATUS	选定的卡发送状态寄存器

CMD14	保留				
CMD15	ac	[31:16]RCA [15:0]保留位	—	GO_INACTIVE_STATE	使选定的卡进入“inactive”状态。这个命令用于当主机明确的想停用这张卡的时候。保留位要设置为‘0’

表 4-20 面向块的读操作(class 2)

命令序号	类型	参数	响应	缩写	描述
CMD16	ac	[31:0]块长度	R1	SET_BLOCKLEN	<p>对于标准 SD 卡来说，这个命令会设置所有块命令的长度(字节)。默认的块长度是 512Byte。只有当 CSD 允许部分块读取操作，设置的长度才对存储访问命令有效。</p> <p>对于大容量 SD 卡来说，CMD16 设置的块长度对于读写命令来说没有硬性，因为块长度是固定的 512Byte。这个命令只对加锁/解锁命令有效。</p> <p>不管哪种，只要块长度设置大于 512Byte，就报错 BLOCK_LEN_ERROR。</p>
CMD17	adtc	[31:0]数据地址 ②	R1	READ_SINGLE_BLOCK	<p>对于标准卡来说，这个命令读取 SET_BLOCK_LEN 命令①所规定的长度的一个块。</p> <p>对于大容量卡来说，直接读取 512 字节的块。</p>
CMD18	adtc	[31:0]数据地址 ②	R1	READ_MULTIPLE_BLOCK	<p>连续数据块传输(从卡到主机)，直到被 STOP_TRANSMISSION 命令停止。</p> <p>块长度和 CMD17 一样</p>
CMD19... CMD23	保留				

注意：

- ①:数据传输不应该穿过物理块的边界，除非 CSD 中设置了 READ\_BLK\_MISALIGN。
- ②:数据地址在标准卡中是以字节为单位的，而大容量卡中，是以块(512byte)为单位的。

表 4-21 面向块的写操作(class 4)

命令序号	类型	参数	响应	缩写	描述
CMD16	ac	[31:0]块长度	R1	SET_BLOCKLEN	见表 4-20
CMD24	adtc	[31:0]数据地址②	R1	WRITE_BLOCK	<p>标准卡来说，写 SET_BLOCKLEN 命令定义的长度的块。</p> <p>大容量卡，写 512 字节长度的块，不管 SET_BLOCKLEN。</p>
CMD25	adtc	[31:0]数据地址②	R1	WRITE_MULTIPLE_BLOCK	<p>连续写数据块直到 STOP_TRANSMISSION 命令被发送。</p> <p>块长度和 WRITE_BLOCK 一致</p>
CMD26	强制保留				
CMD27	adtc	[31:0]填充位	R1	PROGRAM_CSD	对 CSD 的可编程位进行编程

注意：

- ①:数据传输不应该穿过物理块的边界,除非 CSD 中设置了 READ\_BLK\_MISALIGN。如果部分块写操作不支持,那么块长度=默认块长度(CSD 中定义了)。
- ②:数据地址在标准卡中是以字节为单位的,而高容量卡中,是以块(512byte)为单位的。

表 4-22 面向块的写保护命令(class 6)

命令序号	类型	参数	响应	缩写	描述
CMD28	ac	[31:0]数据地址②	R1b	SET_WRITE_PROT	如果卡支持写保护功能,这个命令设置地址组中的写保护位。写保护的目的是编码卡上的特殊数据。高容量卡不支持这个功能
CMD29	ac	[31:0]数据地址②	R1b	CLEAR_WRITE_PROT	如果卡支持写保护,这个命令清除寻址组的写保护位。高容量卡不支持这个功能
CMD30	adtc	[31:0]写保护数据地址②	R1	SED_WRITE_PROT	如果卡支持写保护,这个命令要求卡返回写保护位的状态。① 高容量卡不支持这个功能
CMD31	保留				

- ①:32bit 的写保护位会通过数据线发送,后面跟着 16CRC。写保护位的最后 1bit(最小位)对应着第 1 个地址组。如果最后一个组的地址超过了有效范围,那么对应的写保护位就会设置为 1。
- ②:标准卡的数据地址以字节为单位。

表 4-23 擦除命令(class 5)

命令序号	类型	参数	响应	缩写	描述
CMD32	ac	[31:0]数据地址①	R1	ERASE_WR_BLK_START	设置要擦除的第一个块的地址
CMD33	ac	[31:0]数据地址①	R1	ERASE_WR_BLK_END	设置要擦除的最后一个块的地址
CMD38	ac	[31:0]填充位	R1b	ERASE	擦除所有预先选定的写块
CMD39	保留				
CMD40					SD 卡无效,保留给 SDIO
CMD41	保留				

- ①:标准卡是字节为单位,高容量卡是块为单位(512Byte)

表 4-24 加锁命令(class 7)

命令序号	类型	参数	响应	缩写	描述
CMD16	ac	[31:0]块长度	R1	SET_BLOCK_LEN	见表 4-20
CMD42	adtc	[31:0]保留全 0	R1	LOCK_UNLOCK	用来设置/复位密码,或者加锁/解锁卡。数据块的长度是通过 SET_BLOCK_LEN 命令设置的。 参数以及锁卡数据结构里的保留位应该设置为 0
CMD43-49 CMD51	保留				

表 4-25 特定应用命令(class 8)

命令序号	类型	参数	响应	缩写	描述
CMD55	ac	[31:16]RCA [15:0]填充位	R1	APP_CMD	告诉卡，下个命令是特定应用命令，而不是标准命令。
CMD56	adtc	[31:1]填充位 [0]读/写	R1	GEN_CMD	通用命令，或者特定应用命令中，用于传输一个数据块到卡，或者卡获取一个数据块。通用卡数据块长度由 SET_BLOCK_LEN 命令设置。高容量卡为固定 512Byte。 [Bit0]1 为读数据，0 表示写数据到卡。
CMD58-59	保留				
CMD60-63	强制保留				

如果支持 Class 8，那么所有表 4-25 中的命令都支持。

表 4-26 I/O 模式命令(class 9)

命令序号	类型	参数	响应	缩写	描述
CMD52 CMD53 CMD54					保留给 I/O 模式使用。(SDIO Card Specification)

所有保留的命令都应该有一个 48bit 长的代码，响应也是(如果有的话)。

所有的 ACMD 命令都要跟在 CMD55 之后。

表 4-27 SD 卡使用和保留的特定应用命令

命令序号	类型	参数	响应	缩写	描述
ACMD6	ac	[31:2]填充位 [1:0]总线宽度	R1	SET_BUS_WIDTH	定义数据总线的宽度(‘00’=1bit, ‘10’=4bit)。接受的数据总线定义在 SCR 寄存器中。
ACMD13	adtc	[31:0]填充位	R1	SD_STATUS	发送 SD 状态，状态区域见表 4-38。
ACMD17	保留				
ACMD18	—	—	—	—	保留给 SD 安全应用①
ACMD19 ACMD21	保留				
ACMD22	adtc	[31:0]填充位	R1	SEND_NUM_WR_BLOCK	发送已经写入的块(没有错误的)的数目。响应 32bit + CRC 数据块。 如果 WRITE_BL_PARTIAL=0，ACMD22 的单位总是 512Byte。如果=1，ACMD22 的单位就是写命令使用的单块的长度。
ACMD23	ac	[32:23]填充位 [22:0]块数	R1	SET_WR_BLK_ERASE_COUNT	设置需要擦除的块的数目(提高速度)。默认值=1。②
ACMD24	保留				
ACMD25	—	—	—	—	保留给 SD 安全应用①
ACMD26	—	—	—	—	保留给 SD 安全应用①



ACMD38	—	—	—	—	保留给 SD 安全应用①
ACMD39 ACMD40	保留				
ACMD41	bcr	[31]保留位 [30]HCS (OCR30) [29:24]保留位 [23:0]Vdd 电 压 (OCR[23:0])	R3	SD_SEND_OP_COND	<u>发送卡的支持信息 (HCS)，并要求卡通过命令线返回 OCR 寄存器内容。</u> 当卡收到 SEND_IF_COND 时，HCS 是有效的。保留位设为 0。CCS 位对应 OCR[30]
ACMD42	ac	[31:1]填充位 [0]设置卡检测	R1	SET_CLR_CARD_DETECT	1-connect, 0-discon, 50k 欧姆上拉电阻，DAT3/CD
ACMD43 ACMD49	—	—	—	—	保留给 SD 安全应用①
ACMD51	adtc	[31:0]填充位	R1	SEND_SCR	读取配置寄存器 SCR

①: SD Specifications Part3 Security Specification  
 ②:STOP\_TRAN(CMD12)用来停止多块的数据传输，不管 ACMD23 功能是否使用。

表 4-28 切换功能命令(Class 10) (V1.10 版本添加)

命令序号	类型	参数	响 应	缩写	描述
CMD6	adtc	[31]模式 0: 查询，1: 切换 [30:24]保留，全 0 [23:20]保留给组 6 (0h 或 Fh) [19:16]保留给组 5 [15:12]保留给组 4 [11:8]保留给组 3 [7:4] 功能组 2-命令系统 [3:0] 功能组 1-访问模式	R1	SWITCH_FUNC	具体描述见 4.3.10
CMD34-37	为切换功能设置的每一个命令系统保留。详细内容见每一个特定系统				
CMD50					
CMD57					

## 4.8 卡状态转换表

表 4-29 卡状态转换表

	当前状态									
	idle	ready	ident	stby	tran	data	rcv	prg	dis	ina
触发状态改变	下个状态									
单独类										
“操作完成”	—	—	—	—	rcv	—	—	—	—	—
Class 0	基础命令									
CMD0	idle	idle	idle	idle	idle	idle	idle	idle	idle	—
CMD2	—	ident	—	—	—	—	—	—	—	—
CMD3	—	—	stby	stby	—	—	—	—	—	—



ACMD41, OCR 检查 fails	ina	—	—	—	—	—	—	—	—	—
ACMD41, 查询模式	idle	—	—	—	—	—	—	—	—	—
ACMD42	—	—	—	—	tran	—	—	—	—	—
ACMD51	—	—	—	—	data	—	—	—	—	—
<b>Class 9</b>	<b>SDIO 保留</b>									
CMD52–CMD54										
<b>Class 10</b>	<b>切换功能</b>									
CMD6	—	—	—	—	data	—	—	—	—	—
CMD34–37, 50, 57										
<b>Class 11</b>	<b>保留</b>									
CMD41, CMD43–49, CMD58–59	保留									
CMD60–CMD63	强制保留									

①: 卡返回 busy 的情况:

1) 卡执行初始化处理 ; 2) 卡是高容量卡, 但是 host 不支持。

## 4.9 响应

所有的响应都是通过 CMD 线发送的。响应总是从 bit 串最左边的 bit 开始发送, 对应响应码。响应的长度取决于响应的类型。

响应总是以起始位开始(0), 跟着是传输方向(card=0)。下表中用 x 代替的表明是可变的。除了 R3 类型之外, 所有响应都用 CRC7 来保护。所有命令以结束位结束(1)。

SD 卡有 5 中响应类型。SDIO 卡支持增加的 R4, R5 类型的响应。

### 4.9.1 R1(正常响应命令)

长度 48bit。bit [45:40]代表响应的命令码(0–63)。卡的状态存储在 bit [39:8]。注意: 如果有传输到卡的数据, 那么每个块传输完成后, 数据线上会有 busy 信号。主机在发送完数据后, 应该检查 busy 信号。状态描述见 4.10。

表 4-30 Response R1

Bit 位	47	46	[45:40]	[39:8]	[7:1]	0
位宽	1	1	6	32	7	1
值	‘0’	‘0’	x	x	x	‘1’
描述	起始位	传输位	命令号	卡状态	CRC7	结束位

### 4.9.2 R1b

R1b 就是 R1 响应命令, 同时数据线上有 busy 信号。卡在收到这些命令后可能会变为 busy。主机应该在响应中检查 busy。时序图见 4.12.3 章。

### 4.9.3 R2(CID, CSD 寄存器)

长度为 136bit。CID 寄存器的内容作为 CMD2 和 CMD10 的响应发送。CSD 寄存器的内容作为 CMD9 的响应发送。只传输 CID 和 CSD 寄存器的[127:1]位, 这些寄存器的第[0]位被响应的结束位替代了。

表 4-31 Response R2

Bit 位	135	134	[133-128]	[127:1]	0
位宽	1	1	6	127	1
值	‘0’	‘0’	‘111111’	x	‘1’
描述	开始位	传输位	保留	CID 或者 CSD 寄存器的值，内部 CRC7	结束位

#### 4.9.4 R3(OCR 寄存器)

长度 48bit，OCR 寄存器的值作为 ACMD41 的响应发送。

表 4-32 Response R3

Bit 位	47	46	[45:40]	[39:8]	[7:1]	0
位宽	1	1	6	32	7	1
值	‘0’	‘0’	‘111111’	x	‘111111’	‘1’
描述	开始位	传输位	保留	OCR 寄存器	保留	结束位

#### 4.9.5 R6(发布的 RCA 寄存器响应)

长度 48bit。[45:40]是响应的命令号，这里就是 ‘000011’，即 CMD3。参数中的 16 位 MSB 用于产生 RCA 号。

表 4-33 Response R6

Bit 位	47	46	[45:40]	[39:8]		[7:1]	0
位宽	1	1	6	16	16	7	1
值	‘0’	‘0’	x	x	x	x	‘1’
描述	开始位	传输位	命令号 ‘000011’ CMD3	新发布卡的 RCA[31:16]	[15:0]卡的状态位: 23, 22, 19, 12:0 见表 4-36	CRC7	结束位

#### 4.9.6 R7(卡接口条件)

长度 48bit。卡支持的电压信息通过 CMD8 的响应发送。Bit[19:16]表明卡支持的电压范围。卡接受提供的电压范围就返回 R7 响应。卡会在响应的参数中返回电压范围和检查模式。

表 4-34 Response R7

Bit 位	47	46	[45:40]	[39: 20]	[19:16]	[15:8]	[7:1]	0
位宽	1	1	6	20	4	8	7	1
值	‘0’	‘0’	‘001000’	‘00000h’	x	x	x	‘1’
描述	开始位	传输位	命令号 CMD8	保留位	接受电压	检查模式	CRC7	结束位

表 4-35 R7 中接受的电压

接受电压	值的定义
0000b	没定义
0001b	2.7V-3.6V
0010b	保留给低电
0100b	保留
1000b	保留
其他	没定义

## 4.10 SD 卡的两种状态信息

SD 卡支持两种状态区域：

- ‘卡状态’：执行命令的错误和状态信息，在响应中标示。
- ‘SD 状态’：512 位的扩展状态信息，支持 SD 卡的特定功能以及将来的应用特定功能。

### 4.10.1 卡状态

响应 R1 包含一个 32bit 的“卡状态”区域。这个区域用来传输卡的状态信息(可以被存在本地状态寄存器中)给主机。如果没有特别说明，这个状态总是同前一个命令相关。

表 4-36 定义了不同的状态信息。类型和清除条件区域的缩写说明如下：

- 类型：
  - ◆ E：错误位
  - ◆ S：状态位
  - ◆ R：检测和设置实际的命令响应。
  - ◆ X：在命令执行期间，检测和设置。主机可以通过发送带有 R1 响应的命令得到状态。
- 清除条件：
  - ◆ A：对应卡当前状态
  - ◆ B：总是与上一条命令相关。接收到有效命令会清除它(有一个命令的延迟)
  - ◆ C：读之后就清除

表 4-34 卡状态

位	标识	类型	值	描述	清除条件
31	OUT_OF_RANGE	E R X	‘0’ = no error ‘1’ = error	命令的参数超出卡的接受范围	C
30	ADDRESS_ERROR	E R X	‘0’ = no error ‘1’ = error	没对齐的地址，同命令中使用的块长度不匹配	C
29	BLOCK_LEN_ERROR	E R X	‘0’ = no error ‘1’ = error	卡不接受的块长度， 或者传输的长度同块长度不匹配	C
28	ERASE_SEQ_ERROR	E R	‘0’ = no error ‘1’ = error	擦除命令序列有错	C
27	ERASE_PARAM	E R X	‘0’ = no error ‘1’ = error	要擦除的“写模块”的选择无效	C
26	WP_VIOLATION	E R X	‘0’ = not protected ‘1’ = protected	对保护模块进行写操作。 对写保护卡进行操作	C
25	CARD_IS_LOCKED	S X	‘0’ = card unlocked ‘1’ = card locked	表明卡被主机加锁了	A
24	LOCK_UNLOCK_FAILED	E R X	‘0’ = no error	加锁解锁命令发生错误	C

			‘1’ = error		
23	COM_CRC_ERROR	E R	‘0’ = no error ‘1’ = error	前一个命令的 CRC 检查错误	B
22	ILLEGAL_COMMAND	E R	‘0’ = no error ‘1’ = error	卡当前状态不接受的命令	B
21	CARD_ECC_FAILED	E R X	‘0’ = success ‘1’ = failure	内部 ECC 收到但是数据不对	C
20	CC_ERROR	E R X	‘0’ = no error ‘1’ = error	内部的卡控制器错误	C
19	ERROR	E R X	‘0’ = no error ‘1’ = error	通用或者未知错误	C
18	保留				
17	保留				
16	CSD_OVERWRITE	E R X	‘0’ = no error ‘1’ = error	— CSD 的只读部分同卡的内容不匹配 — 试图方向拷贝，或者永久写保护位	C
15	WP_ERASE_SKIP	E R X	‘0’ = not protected ‘1’ = protected	因为保护块，或者是写保护卡，只擦除了部分指定的地址。	C
14	CARD_ECC_DISABLED	S X	‘0’ = enabled ‘1’ = disabled	命令已经执行，没使用内部 ECC	A
13	ERASE_RESET	S R	‘0’ = cleared ‘1’ = set	因为收到退出擦除序列命令，在执行之前，擦除序列被清除	C
12:9	CURRENT_STATE	S X	0 = idle 1 = ready 2 = ident 3 = stby 4 = tran 5 = data 6 = rcv 7 = prg 8 = dis [9-14]保留 [15]IO 保留	收到命令时的卡的状态。如果命令引起状态改变，那么会在下一条命令的响应中反馈。这 4Bit 数被作为 2 进制数进行解读	B
8	READY_FOR_DATA	S X	‘0’ = not ready ‘1’ = ready	总线上表示缓存空的状态，即可以接受数据	A
7:6					
5	APP_CMD	S R	‘0’ = disabled ‘1’ = enabled	卡会接收到 ACMD 命令。或者说命令已经被作为 ACMD 解读	C
4	保留给 SDIO 卡				
3	AKE_SEQ_ERROR	E R	‘0’ = no error ‘1’ = error	认证过程序列发生错误	C
2	保留给应用特定命令				
1:0	保留给厂商测试模式				



针对每个响应 R1 的命令，下表定义了状态位受影响的位。x 表示错误/状态位可能分别设置在命令的响应中。

表 4-37 卡状态区域/命令 - 交叉相关

CMD#	Response Format 1 Status bit #																					
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12:9	8	5
3①									x	x			x							x		
6②	x						x		x	x	x	x	x	x	x					x		
7					x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
12	x	x				x	x		x	x	x	x	x	x	x			x		x		
13	x	x			x	x	x	x	x	x	x	x	x	x	x	x	x	x		x	x	
16			x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
17	x	x			x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
18	x	x			x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
24	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
25	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
26					x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
27					x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
28	x				x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
29	x				x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
30	x				x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
32	x			x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
33	x			x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
38				x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
42					x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
55					x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x
56					x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
ACMD6	x				x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x
ACMD13					x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x
ACMD22					x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x
ACMD23					x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x
ACMD42					x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x
ACMD51					x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x

①:CMD3 的响应是 R6，只包含卡状态的 bit 23, 22, 19 和 12:9

②:CMD6 是 V1.1 之后定义的

4. 10. 2 SD 状态

SD 状态包含了与 sd 卡属性功能相关的状态位，可能会用于将来特定的应用命令。SD 状态的大小是一个 512bit 的数据块。这个寄存器的内容会通过 DAT 总线传递给主机，CRC16。SD 状态会通过 DAT 总线发送给主机，作为 ACMD13(前面是 CMD55)的响应。ACMD13 只能在“transfer”模式发送给已选定的卡。卡状态结构体在下表描述。“类型”和“清除条件”使用的缩写和卡状态一样。

表 4-38 SD 状态

位	标识	类型	值	描述	清除条件
511:510	DAT_BUS_WIDTH	SR	‘00’ =1 默认 ‘01’ =保留 ‘10’ =4 位宽 ‘11’ =保留	SET_BUS_WIDTH 定义的当前总线宽度	A
509	SECURED_MODE	SR	‘0’ =不是担保模式 ‘1’ =担保模式	参考” SD Security Specification”	A
508:496	保留				
495:480	SD_CARD_TYPE	SR	‘00xxh’ =V1.01-2.0 ‘0000h’ =常 规 读 写 卡 ‘0001h’ =SD Rom 卡	在将来，8 位 LBS 将会被用来定义 SD 卡的不同的变化(每一位会定义不同的 SD 类型)。8 位 MSB 会被用来定义不遵循当前 Sd 物理协议的 SD 卡	A
479:448	SIZE_OF_PROTECTED_AREA	SR	保护区域尺寸	见后面	A
447:440	SPEED_CLASS	SR	卡的速度等级	见后面	A
439:432	PERFORMANCE_MOVE	SR	指示性能 1[MB/s]	见后面	A
431:428	AU_SIZE	SR	AU 尺寸	见后面	A
427:424	保留	SR			
423:408	ERASE_SIZE	SR	一次擦除的 AU 数	见后面	A
407:402	ERASE_TIMEOUT	SR	擦写超时 UNIT_OF_ERASE_AU	见后面	A
401:400	ERASE_OFFSET	SR	擦除时间的固定偏移	见后面	A
399:312	保留				
311:0	厂家保留				

● SIZE\_OF\_PROTECTED\_AREA

这个设置在标准和高容量卡之间是有不同的。

标准卡，保护区域的容量是这么计算的：

保护区域 = SIZE\_OF\_PROTECTED\_AREA \* MULT\*BLOCK\_LEN

SIZE\_OF\_PROTECTED\_AREA 是以 MULT\*BLOCK\_LEN 为单位。

高容量卡，保护区域的容量是这么计算的：

保护区域= SIZE\_OF\_PROTECTED\_AREA

SIZE\_OF\_PROTECTED\_AREA 以 byte 为单位

● SPEED\_CLASS

8bit 的数据，表示速度等级，数值是 2 的整数倍

表 4-39 速度等级表

速度等级	数值定义
00h	Class0
01h	Class2
02h	Class4
03h	Class6
04h-FFh	保留

● PERFORMANCE\_MOVE

8 位区域表明性能，以及可以设置的值[MB/s]。如果卡不使用 RU 搬迁数据，那么 Pm 是无穷大的。设置 FFh 表示无穷大。最小值定义在[表 4-49]中。

表 4-40 数据移动性能

速度等级	数值定义
00h	没定义
01h	1 [MB/s]
02h	2 [MB/s]
...	... [MB/s]
FEh	254 [MB/s]
FFh	无穷大

● AU\_SIZE

4bit 数值，AU 的长度，值是  $16 \times (2^n)$

表 4-41 AU 大小

AU_SIZE	数值定义
0h	没定义
1h	16KB
2h	32KB
3h	64KB
4h	128KB
5h	256KB
6h	512KB
7h	1MB
8h	2MB
9h	4MB
Ah-Fh	保留

最大的 AU 长度，取决于卡的容量，表 4-42 中定义。卡可以设置任意 AU 长度，要介于 RU 长度和最大 AU 长度之间。

容量	16MB-64MB	128MB-256MB	512MB	1GB-32GB
最大 AU 长度	512KB	1MB	2MB	4MB

注意：主机应该使用最大 AU 长度 (4MB) 来决定主机的缓存大小。主机可以协调多个 AU 使用同一个单元。

● ERASE\_SIZE

16bit 长度，代表 N.erase。当 N.erase 个 AU 被擦除后，超时值由 ERASE\_TIMEOUT 指定。主机应该决定一次操作中要擦除的 AU 数，这样主机就能够表明擦除操作的进程。如果值是 0，那么不支持擦除超时的计算

表 4-43 擦除大小

擦除大小	数值定义
000h	不支持擦除超时计算
0001h	1AU
0002h	2AU
...	...
FFFFh	65535 AU

● ERASE\_TIMEOUT

6bit 长度，表明 T.erase，值表明了当 ERASE\_SIZE 定义的多 AUs 被擦除时，偏移的擦除超时。ERASE\_TIMEOUT 的范围被定义到 63 秒，而且卡厂家可以根据情况改变 ERASE\_SIZE 和 ERASE\_TIMEOUT 的组合。一旦 ERASE\_TIMEOUT 确定了，那么 ERASE\_SIZE 也确定了。主机可以通过方程式 (6) 确定任意数目的擦除超时。参考 4.14 中的擦除超时计算方法。如果 ERASE\_SIZE 被设置为 0，这个值也是 0。

表 4-44 擦除超时

擦除大小	数值定义
00	不支持擦除超时计算
01	1 秒
02	2 秒
...	...
63	63 秒

● ERASE\_OFFSET

2bit 数据，表明 T.offset，有 4 种选择。擦除偏移通过上位平移来调整线。参考表 4-33 以及 4.14 的公式 (6)。如果 ERASE\_SIZE 和 ERASE\_TIMEOUT 是 0 的话，这个值无意义。

表 4-44 擦除偏移

擦除偏移	数值定义
0	0 秒
1	1 秒
2	2 秒
3	3 秒

4.11 存储阵列分区

数据传输的基本单位是字节。而以块进行传输的，块的长度也是直接的整数倍。一些特殊的功能需要其他的划分尺寸。

- **块：**是块读写操作的单位。它的值是主机发送一个块命令时需要传输的字节数。块的尺寸是可编程的或者是固定的。允许的块尺寸和可编程性存储在 CSD 中。
- 对于有可擦除单元的设备来说，定义了特殊的擦除命令。可擦除单元的尺寸一般情况下和块操作命令不一样。
- **扇区，**擦除命令的单位。它的值是擦除操作要擦除的块的数目。每个设备的扇区尺寸是固定的。扇区尺寸信息存储在 CSD 中。注意，如果卡指定了 AU 尺寸，那么扇区尺寸就会被忽略。
- **AU(分配单元)**，卡的物理边界，包括一个或多个块，取决于每张卡。卡的容量决定最大的 AU 大小。此外，AU 是带有速度等级的卡表明设备性能的最小单位。尺寸信息和速度等级存储在 SD 状态中。AU 还用于计算擦除超时。
- **保护组(WP-Group)**，对支持写保护组的设备进行单独写保护的最小单位。它的值是将要按字节进行写保护的组的数目。每个设备的值是固定的。存储在 CSD 中。高容量卡不支持写保护组命令。

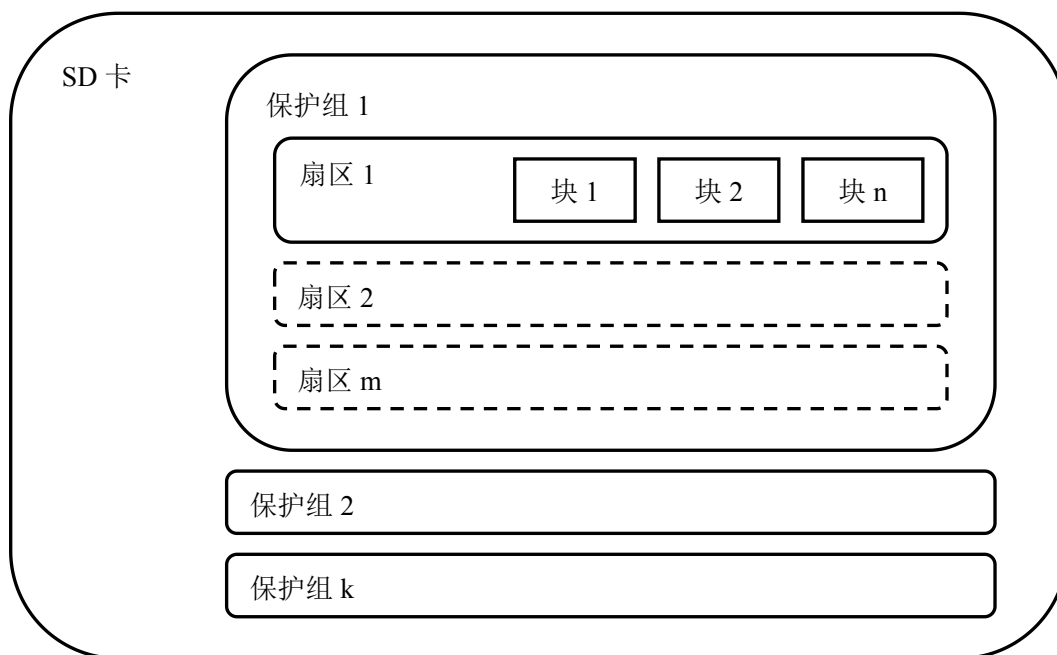


图 4-12 写保护层次

每个保护组可以有一个额外的写保护位。写保护位可以通过特殊的命令编程(见 4.7.4)。功能都是可选的，只对可写/可擦除设备有效。写保护同样可以用户多种卡(ROM-Flash 组合)。关于可用性信息存储在 CSD 中。

## 4.12 时序

所有时序图都采用下面的图例和缩写：

S	起始位(= ‘0’)
T	传输位(Host=‘1’, Card=‘0’)
P	一个周期上拉(= ‘1’)
E	结束位(= ‘1’)
Z	高阻抗状态(->= ‘1’)
D	数据位
X	不关心数据位(From card)
*	重复
CRC	CRC7
	卡活动的(active)
	主机活动的(active)

P 位和 Z 位的不同之处就是，P 位是被主机输出驱动的各个卡主动拉高的，Z 位是分别被上拉电阻 Rcmd 和 Rdat 拉高。P 位受噪音影响更小。所有的时序值都在表 4-47 中定义。

### 4.12.1 命令和响应

主机命令和卡响应都是动过 6.8 章(6.9 高速卡)中指定的时序计时。

#### ● 卡识别和卡操作条件时序

CMD2 和 ACMD41 的时序如下。命令后跟着两个 Zbit(允许总线切换方向的时间)，接着是响应卡发出的 P bit。卡响应主机命令的起始在 Nid 时钟周期后。



图 4-13 识别时序(卡识别模式)

● 分配卡相对地址 RCA

SEND\_RELATIVE\_ADDR (CMD3) 命令时序如下。主机命令和响应之间的最小延迟是 Ncr 个时钟周期。



图 4-14: 发送相对地址(CMD3)时序

● 数据传输模式

当卡发布自己的 RCA 后。将会切换到数据传输模式“transfer”。命令后跟着两个 Zbit (允许总线上切换方向的时间)，然后响应卡发送的 Pbit。这个响应时序是除了 ACMD41 和 CMD2 之外，所有主机命令响应的时序。



图 4-15: 命令响应时序(数据传输模式)

● 上一个卡响应-下一个主机命令时序

当上一个命令发送后，主机在最少 Ncc 个时钟周期后可以继续发送下一个命令。

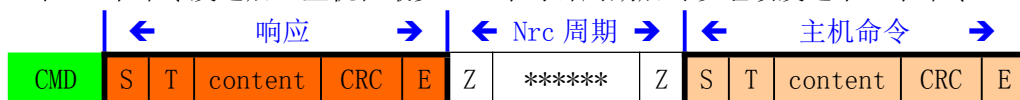


图 4-16: 响应结束-下个命令开始时序(数据传输模式)

● 上一个卡命令-下一个主机命令时序

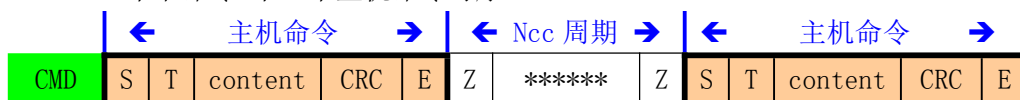


图 4-17: 命令时序(所有模式)

## 4.12.2 读数据

注意：数据线代表着数据总线(不管是 1bit 还是 4bit 宽)

● 单块读

主机通过 CMD7 来选择一个卡进行数据读取操作，通过 CMD16 来设置需要传送数据的有效块长度。读操作的基本总线时序见图 4-18。序列以单块读命令 (CMD17) 开始，CMD17 在参数中指出了起始地址。响应也通过 CMD 线发送。

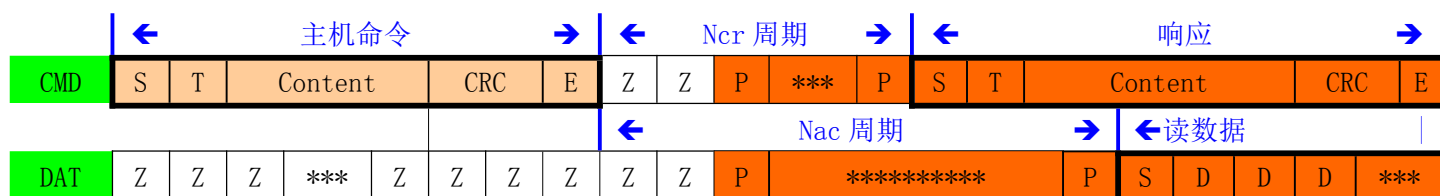


图 4-18: 单块读命令时序



如果卡没有空闲的缓存，卡会把 DAT0 拉低，通知主机。只要缓存空出了能够接受一个数据块长度的空间，就会立刻停止拉低 DAT0。不过这个信号不会给出任何写状态信息，这个信息应该由主机来查询。

● 多块写

多块写模式，卡会在主机发送写命令之后收到连续的数据块。

同单块写类似，数据后跟着 CRC 校验位，允许卡检查数据正确性。卡会通过 DAT0 发送回 CRC 检查结果，即 CRC 状态。如果有传输错误，CRC 状态= ‘101’。如果没错误，CRC 状态= ‘010’，并且启动数据编程。如果出现错误，卡会忽略后续的数据块。这种情况下 CRC 响应不会发送给主机，因此，不会收到 CRC 起始位，CRC 状态= ‘111’。

数据传输流程会被 CMD12 停止。表 4-22 描述了没有 busy 信号的多块写操作。

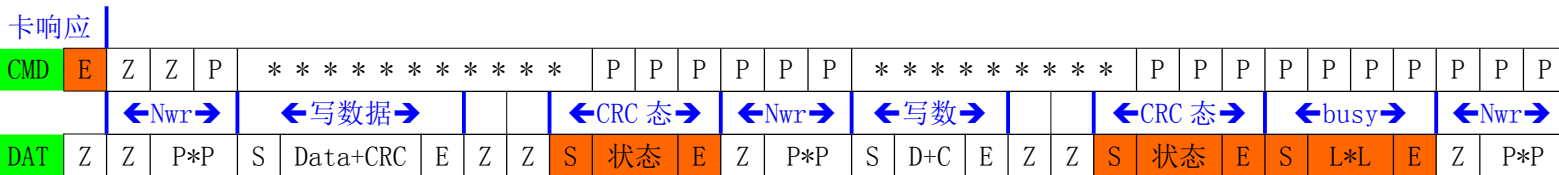


图 4-22：多块写命令时序

停止命令的工作机制同读操作类似。图 4-23 到图 4-26 描述了不同卡状态的停止命令

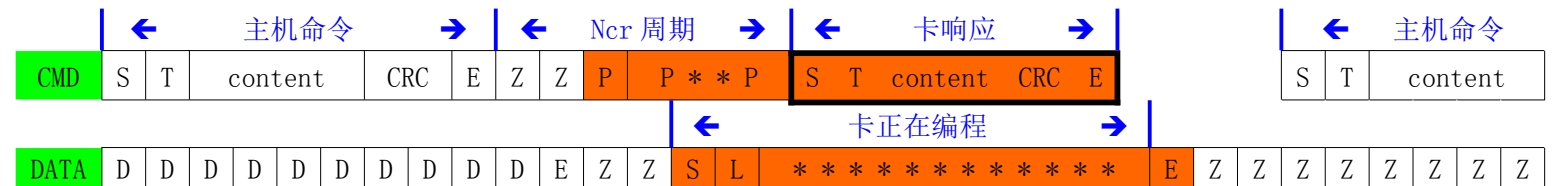


图 4-23：主机数据传输期间停止传输

只有在块的 CRC 数据正确，并且已经把 CRC 状态返回给主机之后，才会认为一个数据块成功收到，并可以编程。图 4-24 的例子，是在卡返回 CRC 状态时，收到主机的停止命令。这个序列和其他的停止传输例子完全一样。主机命令的后面跟着一个数据位和 busy 信号的开始位(数据线上)。这种情况下，没有 Z 时钟来切换总线方向，因为总线方向已经是向主机方向了。这种情况下，收到的数据块会被认为是没有完成，并且不会被编程。

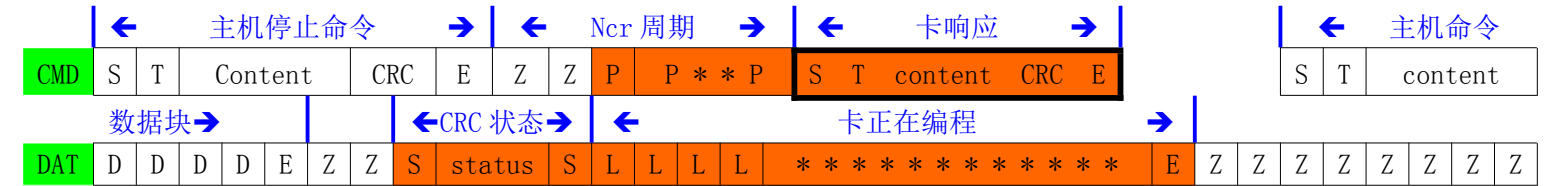


图 4-24：卡传输 CRC 状态期间停止传输

之前的例子都是传输数据期间，主机发起的停止操作。下面两种情况是在数据块中接收到停止操作。第一种情况，当卡处于 idle 状态的时候，卡处于 busy 状态，正在编程最后一个块，。但是仍旧有没有编程的数据块在输入缓存中。这些块将会在停止信号收到的同时编程，卡会显示忙碌信号。

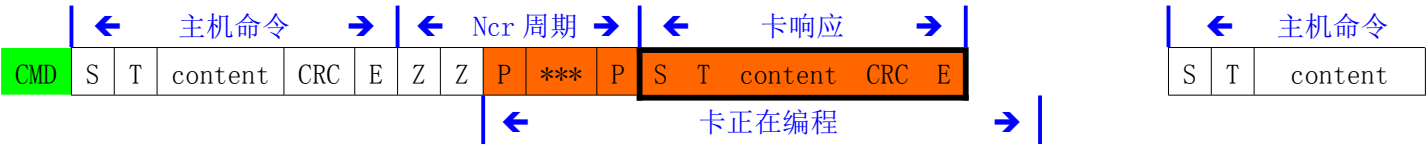




图 4-26: 最后数据块之后，停止传输。卡变为 busy

### ● 擦除，设置/清除写保护时序

主机应该先标识要擦除的开始 (CMD32) 和结束 (CMD33) 地址。擦除命令 (CMD38) 一旦使用，就会擦除选定的块。同样，设置和清除写保护命令也会开始编程操作。擦除或者编程期间，卡会持续发出 busy 信号 (拉低 DAT0)。总线时序和图 4-26 中的一致。

### ● 重选一个 busy 卡

当一个正处于 disconnect 状态的 busy 卡 Bei 选中的时候，它会重新发出 busy 信号。这个命令，响应，busy 的时序图和图 4-26 中一致。

## 4.12.4 时序值

表 4-47 定义了所有的时序值

参数	最小	最大	单位
Ncr	2	64	时钟周期
Nid	5	5	时钟周期
Nac <sup>①</sup>	2	—	时钟周期
Nrc	8	—	时钟周期
Ncc	8	—	时钟周期
Nwr	2	—	时钟周期

表 4-47 时序值

①: 标准 SD 卡的最大读访问时间应该这么计算:

$$Nac(max) = 100 * ((TAAC \times fpp) + (100 \times NSAC));$$

fpp 是时钟速率接口，TAAC 和 NSAC 是 CSD 提供的 (5.3 章)，具体超时见 4.6.2 章  
高容量卡的，最大读超时固定为 100ms。

## 4.13 速度等级规范

速度等级规范依照速度等级号来表现卡的性能，并提供了计算性能的办法。这个规范让主机能够支持 AV 应用能够实时记录到 SD 卡上。下面的文档描述了卡的速度等级规范。以一个应用说明来作为主机实现的例子。

### 4.13.1 分配单元(AU)

用户区域被分为多个单元，被称为 “Allocation Unit (AU)” (见图 4-27)。每张卡都有自己固定的 AU 尺寸 (Szu) 以及卡的残疾定义的最大 AU 尺寸。主机应该以 AU 为单位管理数据区域。AU1 之后的几个 AU 应该被用于实时记录，因为他们可能包含系统信息。AV 应用应该

从第一个完成的 AU 之后启动记录，只有用户数据能被记录到这个 AU。注意，这个规范不支持写保护。

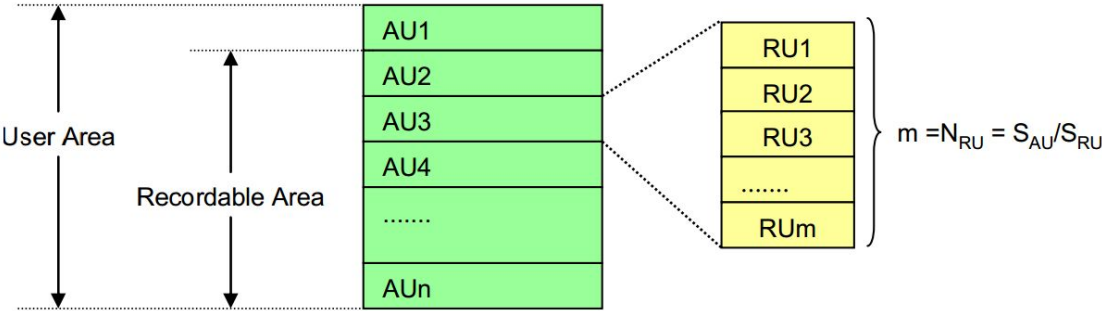


图 4-27 分配单元定义 (AU)

### 4. 13. 2 记录单元 (RU)

每个 AU 被分为多个单元，称为“Recording Unit (RU)”，见上图，RU 尺寸应该是 SD 文件系统规范定义的一个或者多个簇。一个 AU 里的 RU 数目  $(N_{ru}) = S_{au} / S_{ru}$ 。

### 4. 13. 3 写性能

图 4-28 展示了当主机写一个 AU 里的 RU 时，卡的典型数据管理。当主机写内容到一个碎片 AU 中时，卡会拷贝那个 AU。A 位置是 AU 的起始边界，B 位置是 AU 的结束边界。从 A 到 B，主机应该连续写数据到空的 RU 中，跳过已经使用的 RU (不能跳过空的 RU)。当卡控制器正在写或者移动数据的时候，卡会发送 busy 信号给主机让它等待。从 A 到 B 的整个时间可以就是写空 RU 和移动已使用 RU 的时间和。已使用的 RU 数  $(N_u)$  是可以用于 AU 的计算的，空 RU 数  $= N_{ru} - N_u$ 。

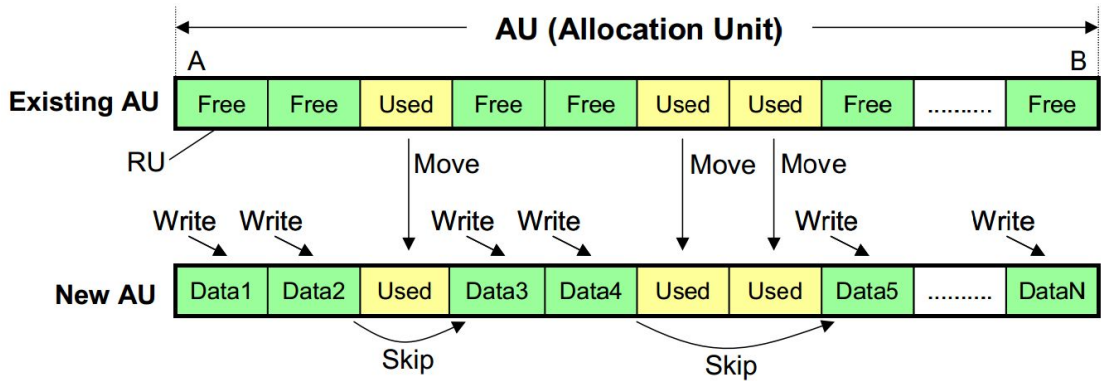


图 4-28 写碎片 AU 实例

碎片化的 AU 的平均性能可以通过总体执行时间，划分空 RU 的总数来计算。通过写性能  $(P_w)$  和移动性能  $(P_m)$  来表达。

$$\begin{aligned} \text{碎片 AU 的性能 } P(N_u) &= S_{ru} \times (N_{ru} - N_u) / (S_{ru} \times (N_{ru} - N_u) / P_w + S_{ru} \times N_u / P_m) \\ &= (N_{ru} - N_u) \times P_m \times P_w / ((N_{ru} - N_u) \times P_m + N_u P_w) \quad \text{【公式 1】} \end{aligned}$$

$P_w$  代表一个 AU 上写性能的最小平均数。它是通过完成一个 AU 的连续 RU 写操作的平均数算出来的，这个 AU 不是碎片化的。

$P_m$  代表移动性能的最小平均数。它是通过完成一个 AU 的连续 RU 移动操作的平均数算出来的。移动是卡内部的操作，所以 SD 的时钟频率不会影响移动操作。如果卡没必要移动 RU 的话， $P_m$  应该认为是无穷大 ( $1/P_m = 0$ )。表 4-49 是每个速度等级的值。

### 4.13.4 读性能

我们定义了两种读性能。一种类型的读操作可以直接在写操作时插入。即使是读操作(任何地址和大小)完成了,这个性能规范也应该被保证。

(1)流数据读性能

这个性能简称为读性能(Pr)。Pr 代表随机 RU 读性能的最小平均数。是 256 个随机单 RU 读操作的平均数。每个 RU 通过多块读命令来进行。Pr 应该大于等于 Pw

(2)FAT 和目录项读时间

Tfr(4KB)是读 4KB FAT 和目录项的最大时间。FAT 和目录项读时间(Sfr[KB])使用 CEIL 函数定义:

**Sfr(KB):  $Tfr(Sfr)=(Sfr/4KB) \times Tfr(4KB)$  【公式 2】**

CEIL 函数。小数 x 转换为最小整数,大于或等于 x。四舍五入?

表 4-49 是每个速度等级的值。

### 4.13.5 性能曲线定义

图 4-29 展示了写性能的 P(Nu) 柱状图。在本例中一个 AU 由 16 个 RU 组成。每个柱子的点的连线就暂时了性能曲线。这个可以通过两个参数计算 Pw 和 Pm。

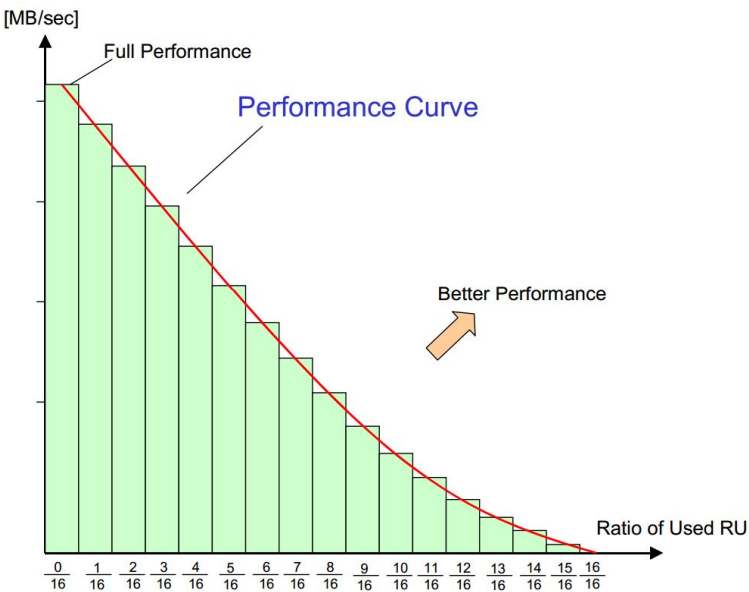


表 4-29 16RU 的卡性能

已用 RU 的比值定义如下:

$$r = Nu/Nru, Nu=r \times Nru$$

r 的取值范围是 0~1。(1-r)代表空 RU 的比例, r=0 代表所有的 RU 都是空的。r=1 代表所有的 RU 都被使用了,这种情况下性能值为 0。通过使用 r, 公式 1 可以换成公式 3

**$P(r)=((1-r) \times Pw \times Pm)/(r \times Pw+(1-r) \times Pm)$  (0≤r≤1) 【公式 3】**

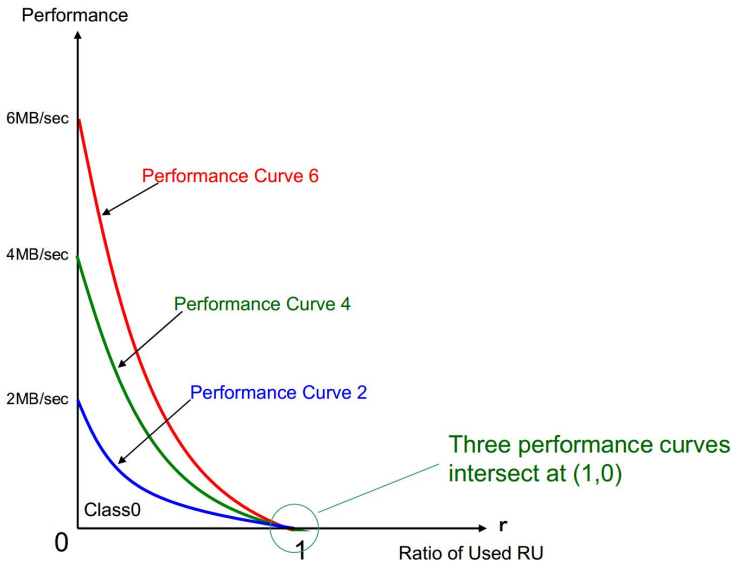
P(Nu)在公式 1 中是碎片功能,但是 P(r)是作为一个连续的功能。

### 4.13.6 速度等级定义

图 4-30 展示了 3 中性能曲线。Pw 表示了 R=0 的性能, Pm 决定了曲线的形状。所有的性能曲线最终聚集到点(1, 0)。因此当 r 接近 1 时,性能有一点不同。这三种曲线把性能分为 4 个等级: Class 0, Class 2, Class 4, Class 6。

**Class 0** 的卡不保证能够符合速度等级规范。即使卡能够达到更高等级的速度，它也不会报告性能参数。**Class 0** 也包括了所有本文档之前的所有 SD 卡产品。定义些等级为的就是 AV 应用(比如 MPEG2)可以支持 SD 卡设备。**Class 2** 卡应该高于曲线 2 的性能，它被定义为标准 TV 图像质量，大约需要 2MB/s 的性能。而 **Class 4** 的卡应该比曲线 4 的性能高，它是用于 HD 视频质量，大约需要 4MB/s 的性能。未来会有更高的等级。但是主机应该总是支持达到最小速度等级性能的卡。

速度等级应该定义为 SD 总线接口的性能，虽然性能曲线是通过 4.13.3 章中的后端性能分析来得到。SD 卡时钟频率和 RU 大小是用来作为速度等级的测量条件的。见 4.13.8 章。



注意：为了方便较早的卡用户，主机应该尽量使用能够允许的较小的性能。当一个模式只提供了为特殊速度等级的卡进行操作，那么其他模式就要支持较低速度的卡包括 Class0。

#### 4.13.7 记录期间插入 FAT 更新的考量

图 4-32 暂时了实时记录的 FAT 更新周期的典型序列。FAT 更新可以插入到任意 RU 访问的中间。FAT 更新周期包含了 3 个 FAT 写操作。FAT1 和 FAT2 意味着两个 FAT 表，每个表的写操作使用了一个多块写命令。FAT 表写操作可以启动任意 512 字节边界地址，和 16Kbyte 以内的任意大小。只有 FAT 的修改部分需要写。DIR 是用于目录条目写操作。一个目录条目应该在记录开始之前就被创建，只有修改的部分需要写到目录条目(512 字节)中。“FAT Write Time” (T<sub>fw</sub>)代表 3 个 FAT 写时序的全部时间。主机应该使用这个时序，这样就能通过插入 FAT 更新周期来计算性能的降低幅度。卡请求需要比应用性能(P<sub>a</sub>)更高的卡性能(P<sub>c</sub>)来插入 FAT 更新周期。

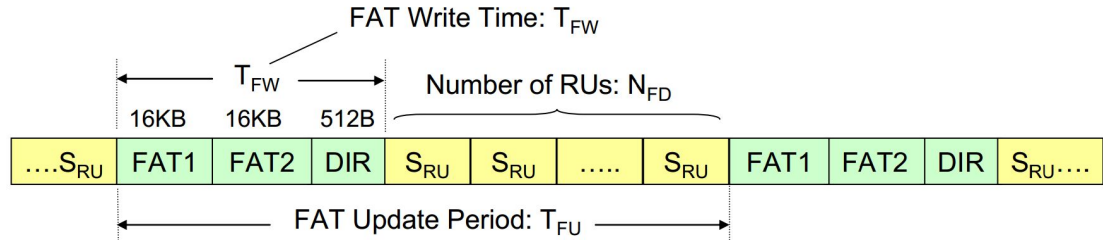


图 4-31 典型 FAT 更新序列



4.13.7.1 确定条件来结束平均 TFW

公式 4 定义了平均 FAT 写操作时间(Tfw(ave)), 8 次 FAT 写周期平均移动值的最大值。

$$T_{FW}(ave.) = \frac{\max(\sum_{i=1}^8 T_{FW}(i))}{8}$$

【公式 4】

4.13.7.2 最大 FAT 写时间

在 FAT 更新期间, 主机不能写数据到卡中。因此, 主机应该准备足够大的缓存来临时存储数据。最大 FAT 写时间 Tfw(max.) 是决定主机缓存大小的一个因素。在 8 次 FAT 写周期中, Tfw(max.) 出现不应该超过 1 次。主机缓存大小估值的方法, 参考 “Implementation Guideline of the Speed Class Specification”。

Tfw(max.) ≤ 750ms 【公式 5】

4.13.8 速度等级的测量条件和需求

4.13.8.1 测量条件

表格 4-48 列出了每个速度等级的测量条件。Fsd 是最小时钟频率, Sru 是最小 RU 长度。Class 6 卡应该两个测试条件, 并且支持高速模式。更高的速度等级需要更高的频率和更大的 RU 长度。这些值提供了主机应用最大速度运行的边界。

表 4-48 速度等级测量条件

	最小 SD 时钟频率 Fsd[MHz]	RU 长度 Sru[Kbyte]
Class 2	20	16
Class 4	20	16
Class 6	40	16
	20	64

注意: 主机可以在 Class6 的条件中选择一个。一种是 16KB 的 RU, 支持高速 40MHz 的时钟, 一种是 64KB 的 RU, 支持普通的 20MHz。最小性能是测量 SD 总线的 100%利用率, 因此, 向 SD 卡写, 会间歇性降低性能。

4.13.8.2 每个速度等级的性能参数需求

表 4-49 列举了在测试条件下, 每个速度等级的参数需求。每个速度等级要同时满足所有条件。任何有速度等级的卡都应该满足更低等级的需求和条件。比如 Class6 在 Class4 的条件下应该满足 Class4 的性能。

表 4-49 每个等级的性能需求

	Pw min.	Pm min.	Pr min.	Tfw(ave.) max.	Tfw(max.) max.	Tfr(4KB) max.
Class2	2 [MB/s]	1 [MB/s]	2 [MB/s]	100 [ms]	750 [ms]	4 [ms]
Class4	4 [MB/s]	2 [MB/s]	4 [MB/s]	100 [ms]	750 [ms]	4 [ms]
Class6	6 [MB/s]	3 [MB/s]	6 [MB/s]	100 [ms]	750 [ms]	4 [ms]



### 4.13.8.3 SD 文件系统的需求

这个规范只支持 SD 协议 2.0 定义的 SD 文件系统格式的卡。将来隐藏扇区的数目将会被接受为最小数，这个数满足数据区域推荐单元边界。

## 4.14 擦除超时计算

这一章提供了长擦除的指导方针，以及一个计算擦除超时值的办法。

### 4.14.1 擦除单元

速度等级规范定义了一个新的 AU 管理单元。擦除超时的计算也作为 AU 的基础。SD 卡支持块擦除，但是擦除 AU 的一部分需要花费更多的时间。这种情况下，主机应该增加 250ms 的超时到 AU 基础计算出来的超时。当开始和结束块在擦除 AU 的相同的部分，应该增加 500ms 的时间。

### 4.14.2 擦除时间特性的案例分析

图 4-32 展示了擦除特性，相对擦除时间擦除的 AU 数目的例子。擦除时间是通过一次擦除命令擦除特定数目 AU 来计算的。假设擦除在 AU 底部执行，而且擦除特点接近线性。图 4-32 中的线 A 就是一个示例特征。

红线表明主机应该使用的擦除超时值。超时值可以由线 A 来决定，如果擦除超时小于 1 秒，那么主机应该使用 1 秒作为超时时间。如果超时大于 1 秒，主机应该使用线 A 来决定使用的超时时间。

寄存器参数  $N_{erase}$ ,  $T_{erase}$ ,  $T_{offset}$  定义了线的形状。 $T_{erase}$  表明从  $T_{offset}$  擦除  $N_{erase}$  个 AU 的超时时间。 $T_{erase}$  和  $N_{erase}$  决定了线的斜度。 $T_{offset}$  通过在线的上部平移来调整线。卡的厂家应该决定这些参数，这样先应该总是大于任何 AU 的擦除时间。

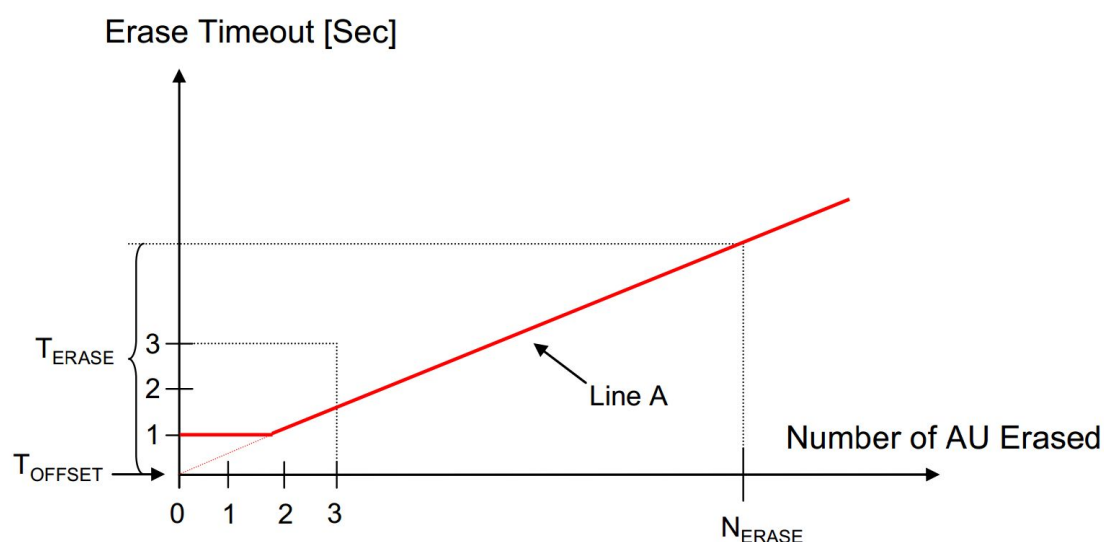


图 4-32 擦除特性示例 ( $T_{offset}=0$ )

图 4-33 中的 B 线是另一种擦除特性的例子，红线表明主机应该使用的擦除超时。既然超时大于 1 秒，那么就应该是使用 B 线。

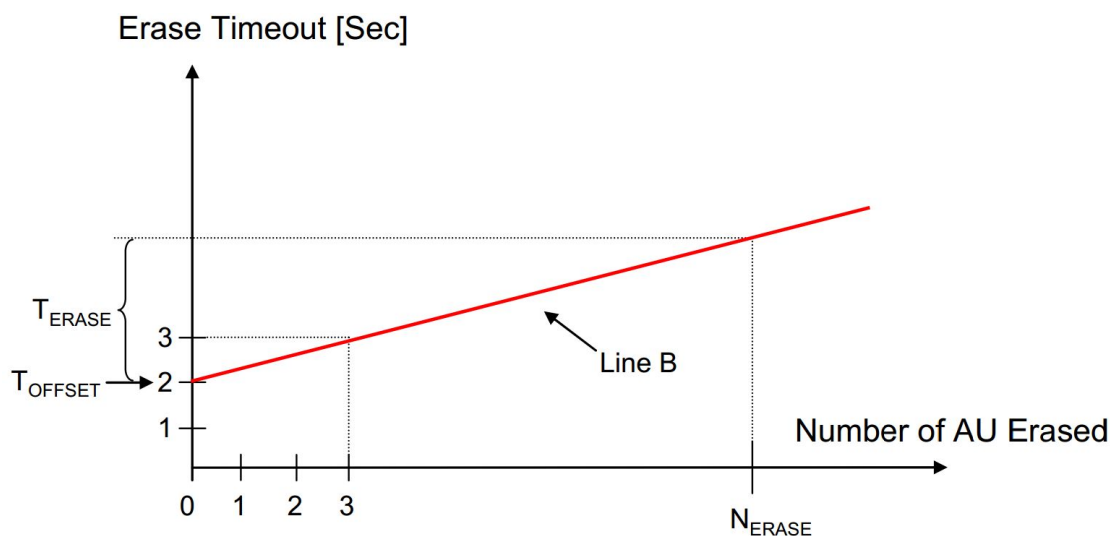


图 4-33 擦除特性示例 (Toffset=2)

#### 4.14.3 大面积擦除的方法

计算的多 AU 擦除超时同实际的相比可能太大。不精确的原因是计算的值包含了边界空白区。每个 AU 的边界空白区积累起来，最终计算出来的超时值就会包含一个很大的边界空白值。这种计算是无意义的，因为边界空白值可能会达到几分钟。因此每次擦除的 AU 应该很少。这样主机计算的超时值就会误差较小。

注意，当一个大区域被擦除时，主机应该以 AU 为界把它分割成小的区域，然后连续擦除小的区域，使用小的擦除超时值，否则用户可能会取消擦除操作。

#### 4.14.4 使用参数寄存器计算擦除超时值

X 个 AU 的擦除超时 =  $(T_{\text{erase}}/N_{\text{erase}}) \times X + T_{\text{offset}}$  【公式 6】

擦除超时通过以下步骤确定：

- (1) 【公式 6】
- (2) 如果【公式 1】结果小于 1 秒，超时设置为 1 秒
- (3) 每个部分擦除 AU 的【公式 2】计算结果都要加上 250ms。当开始和结束都在同一个要擦除的 AU 时，增加 500ms 到【公式 2】的结果上

# 5、SD 卡寄存器

卡的接口中定义了 6 个寄存器：OCR，CID，CSD，RCA，DSR，SCR。这些寄存器只能通过对应的命令访问(见 4.7 章)。OCR，CID，CSD，SCR 寄存器保存了卡/内容的特定信息，RCA 和 DSR 寄存器是配置寄存器，存储目前的配置参数。

为了使能扩展功能，寄存器的保留位卡应该返回 0。

## 5.1 OCR 寄存器

32 位的操作条件寄存器(OCR)存储了卡的 VDD 电压描述。另外，还包括了状态信息位。如果卡的上电程序完成，一个状态位会被设置。寄存器还包括另一个状态位，在设置上电状态位后，用来表明卡的容量状态。SD 卡应该有 OCR 寄存器。

OCR 的 bit7 是新定义的，用于双电压卡，默认设置为 0，。如果双电压卡没有收到 CMD8，那么 OCR 的 bit7 就会显示 0，如果收到 CMD8 了，那么 bit7 就会显示 1。

另外，这个寄存器还包含 2 个状态信息为。

Bit31 - 卡上电状态位，这个状态位在卡的上电流程完成后设置。

Bit30 - 卡容量状态位，如果是高容量卡，设置为 1，如果是标准卡，设置为 0。卡容量状态位只有在上电流程完成，且 Bit31 设置为 1 之后才有效。主机应该读取这个状态位来判断卡的种类。

表 5-1 OCR 寄存器定义

OCR bit 位	OCR 字段描述	说明
0-3	保留	
4、5、6、	保留	
7	保留给低电	
8、9、10、11、12、13、14、	保留	
15	2.7-2.8	设为 1 支持，设为 0 不支持
16	2.8-2.9	
17	2.9-3.0	
18	3.0-3.1	
19	3.1-3.2	
20	3.2-3.3	
21	3.3-3.4	
22	3.4-3.5	
23	3.5-3.6	
24-29	保留	
30	卡容量状态 (CCS)	只有 bit31 设置了才有效
31	卡上电状态 (busy)	上电没完成时设置为 0

## 5.2 CID 寄存器

卡识别 (CID) 寄存器是一个 128 位的寄存器。包含了卡的识别信息，用于卡识别解析。每个读/写卡都有一个特殊的识别号。CID 寄存器的结构如下：

表 5-2 CID 区域表

名称	区域	宽度	CID 位
制造商 ID	MID	8	[127:120]
<u>OEM/应用 ID</u>	OID	16	[119:104]
<u>产品名称</u>	PNM	40	[103:64]
<u>产品版本</u>	PRV	8	[63:56]
<u>产品序列号</u>	PSN	32	[55:24]
保留	--	4	[23:20]
生产日期	MDT	12	[19:8]
CRC7 校验码	CRC	7	[7:1]
不使用, 总是 1	--	1	[0:0]

#### ● MID

8bit 的二进制数, 表示卡的制造商。MID 号, 由 SD-3C, LLC 来控制、定义、以及分配给 SD 卡制造商。这个程序是用来保证 CID 寄存器的唯一性。

#### ● OID

2 个字符的 ASCII 码, 表明卡的 OEM 和/或者卡的内容(当用于分发媒体的时候)。OID 同样是由 SD-3C, LLC 来控制、定义和分配的。也是为了保证 CID 的唯一性

注: SD-3C, LLC 授权给厂家来生产或者销售 SD 卡, 包含但不限于 Flash 存储, ROM, OTP, RAM 和 SDIO 卡。

SD-3C, ALL 是由松下电子工业, SanDisk 公司和东芝公司成立的有限责任公司。

#### ● PNM

产品名称, 5 个字符的 ASCII 码

#### ● PRV

产品版本, 由两个十进制数组成, 每个数 4 个 bit, “n.m”, n 表示大版本号, m 表示小版本号。比如, 产品版本号是 “6.2”, 那么 PRV = “0110 0010b”

#### ● PSN

序列号, 32 位二进制数

#### ● MDT

制造日期由两个 16 进制数组成, 一个是 8bit 的年(y), 一个是 4bit 的月(m)。

m=bit[11:8], 1= 1 月。

n=bit[19:12], 0=2000

比如 2001 年 4 月, MDT= “0000 0001 0100”

#### ● CRC

7 位 CRC 校验码, CID 内容的校验码, 见 4.5 章。

## 5.3 CSD 寄存器

卡特定数据寄存器提供关于访问卡内容的信息。CSD 定义了卡的数据格式, 错误修改类型, 最大数据访问时间, 不管 DSR 寄存器时候可用。寄存器的可编程部分(标记了 W 或者 E 的), 可以通过 CMD27 来修改。下表中的条目类型定义: R = 可读, W(1)=可写 1 次, W=可反复写。

5.3.1 CSD\_STRUCTURE

CSD 寄存器的区域结构体因不同的规范版本和卡的容量而不同。  
CSD 寄存器中的 CSD\_STRUCTURE 表明了结构体版本。

表 5-3CSD 寄存器结构体

CSD_STRUCTURE	CSD 结构体版本	有效的 SD 版本/容量
0	CSD V1.0	版本 1.01-1.10 版本 2.0/标准卡
1	CSD V2.0	版本 2.0/高容量卡
2-3	保留	

5.3.2 CSD 寄存器(1.0)

略

5.3.3 CSD 寄存器(2.0)

表 5-16 暂时了高容量卡的 CSD 寄存器定义。后面章节描述了 CSD 区域，以及高容量卡的相关数据类型。  
CSD V2.0 只适用于高容量 SD 卡。插入的区域名字是固定值，并且主机不一定要求涉及这些区域。这些区域的固定值使主机保持对 CSD V1.0 的兼容。单元类型区域中，R=可读，W(1)=写一次，W=反复写。

表 5-16 CSD 寄存器区域(V2.0)

名称	区域	宽度	值	单元类型	CSD 位
CSD 结构体	CSD_STRUCTURE	2	01b	R	[127:126]
保留	-	6	00 0000b	R	[125:120]
数据读访问时间	(TAAC)	8	0Eh	R	[119:112]
时钟周期 (NSAC*100) 中的数据读访问时间	(NSAC)	8	00h	R	[111:104]
最大数据传输速率	(TRAN_SPEED)	8	32h/5Ah	R	[103:96]
卡命令类	CCC	12	01x110110101b	R	[95:84]
读数据块最大长度	(READ_BL_LEN)	4	9	R	[83:80]
允许块部分读	(READ_BL_PARTIAL)	1	0	R	[79]
写块不对齐	(WRITE_BLK_MISALIGN)	1	0	R	[78]
读块不对齐	(READ_BLK_MISALIGN)	1	0	R	[77]
执行的 DSR	DSR_IMP	1	x	R	[76]
保留	-	6	00 0000b	R	[75:70]
设备尺寸	C_SIZE	22	00 xxxxh	R	[69:48]
保留	-	1	0	R	[47]
擦单块使能	(ERASE_BLK_EN)	1	1	R	[46]
擦扇区尺寸	(SECTOR_SIZE)	7	7Fh	R	[45:39]

写保护组尺寸	(WP_GRP_SIZE)	7	0000000b	R	[38:32]
写保护组使能	(WP_GRP_ENABLE)	1	0	R	[31]
保留		2	00b	R	[30:29]
写速度因素	(R2W_FACTOR)	3	010b	R	[28:26]
最大写数据块长度	(WRITE_BL_LEN)	4	9	R	[25:22]
允许块部分写	(WRITE_BL_PARTIAL)	1	0	R	[21]
保留	–	5	00000b	R	[20:16]
文件格式组	(FILE_FORMAT_GRP)	1	0	R	[15]
拷贝标志 (OTP)	COPY	1	x	R/W(1)	[14]
永久写保护	PERM_WRITE_PROTECT	1	x	R/W(1)	[13]
临时写保护	TMP_WRITE_PROTECT	1	x	R/W	[12]
文件格式	(FILE_FORMAT)	2	00b	R	[11:10]
保留	–	2	00b	R	[9:8]
CRC	CRC	7	xxxxxxx	R/W	[7:1]
无用，总是 1	–	1	1	–	[0]

● TAAC

这个值固定为 0Eh，代表 1ms。主机不应该使用 TAAC，NSAC，和 R2W\_FACTOR 来计算超时，而应该使用固定的超时值来进行读写操作。（见 4.6.2）

● NSAC

这个值固定是 00h。NSAC 也不应该用于超时值计算。

● TRAN\_SPEED

下表定义了每条数据线的最大数据传输速率-TRAN\_SPEED：

表 5-6 最大数据传输速率

TRAN_SPEED 位	代码
2:0	传输速率单元 0=100Kbit/s, 1=1Mbit/s, 2=10MBit/s 3=100Mbit/s, 4…7=保留
6:3	时间值 0=保留, 1=1.0, 2=1.2, 3=1.3, 4=1.5, 5=2.0, 6=2.5, 7=3.0, 8=3.5, 9=4.0, A=4.5 B=5.0, C=5.5, D=6.0, E=7.0, F=8.0
7	保留

注意，对于标准 SD 卡，这个区域值应该总是 0 0110 010b(32h)，这个等于 25MHz，是强制的 SD 卡最大操作频率。

对于高速卡来说，这个值应该是 0 1011 010b(5Ah)，这个值代表 50MHz，当时序通过 CMD6 和 CMD0 命令回到默认的时候，这个值会重新设置为 032h。

● CCC

SD 卡命令设置被分为一些子合集(命令类)。卡命令类寄存器 CCC 定义了这张卡支持哪些命令类。CCC 的某 1bit 被设置为 1，表明支持对应的命令类。命令类的定义，参考表 4-18。

表 5-7 支持的卡命令类

CCC 位	支持的卡的命令类
0	Class0
1	Class1
...	...
11	Class11

#### ● READ\_BL\_LEN

这个值固定为 9h，代表 READ\_BL\_LEN=512byte

#### ● READ\_BL\_PARTIAL

这个值固定为 0，表明不允许部分块读操作，只能按块进行访问。

#### ● WRITE\_BLK\_MISALIGN

这个值固定为 0，表明高容量卡中，写操作不允许越过物理块边界

#### ● READ\_BLK\_MISALIGN

这个值固定为 0，表明高容量卡中，读操作不允许越过物理块边界

#### ● DSR\_IMP

定义是否可配置驱动阶段在卡上整合了。如果设置了，驱动阶段寄存器 (DSR) 就应该使用，参考 5.5 章。0 — 不实现 DSR，1 — 实现 DSR。

#### ● C\_SIZE

这个值有 22bit，可以支持最大到 2Tbyte(和通过 32Bit 块地址指定的最大存储空间一致)  
这个参数是用来计算 sd 卡的用户数据区域容量的(不包括保护区)。公式如下：

$$\text{存储容量} = (\text{C\_SIZE} + 1) \times 512\text{K byte}$$

鉴于 V2.0 支持的最大容量是 32GB，超过 6bit 的部分都设置为 0。

#### ● ERASE\_BLK\_EN

这个值固定为 1，代表着主机可以擦除 1 个或者多个 512 字节单位。

#### ● SECTOR\_SIZE

这个值固定是 7Fh，这个代表 64 字节。这个值同擦除操作没关系。V2.0 的卡通过 AU 尺寸来表明存储边界，不用这个值。

#### ● WP\_GRP\_SIZE

这个值固定是 0，高容量 SD 卡不支持写保护组。

#### ● WP\_GRP\_ENABLE

这个值固定为 0，高容量卡不支持写保护组。

#### ● R2W\_FACTOR

这个值固定是 2h，代表 4 倍。写超时可以通过读访问时间和 R2W\_FACTOR 的乘积计算。但是，对于写超时，主机不应该用这个参数，而是应该用固定的 250ms。

#### ● WRITE\_BL\_LEN

这个值固定是 9h，代表 WRITE\_BL\_LEN=512Byte。

#### ● WRITE\_BL\_PARTIAL

这个值固定是 0，表明部分块写不支持，值支持块整数倍的写操作。

#### ● FILE\_FORMAT\_GRP

这个值固定是 0，主机不应该使用这个值。

#### ● COPY

定义是否内容是原始的(0)，或者是拷贝的(1)。拷贝位对于销售到终端用户的 OTP 和 MTP 设备来说是设置为 1 的，这表明卡的内容是拷贝的。拷贝位是一次性编程位。

#### ● PERM\_WRITE\_PROTECT

永久保护整个卡内容，不允许写和擦除(所有相关命令都无效)。默认值是 0，非永久写保护。



● **TMP\_WRITE\_PROTECT**

临时保护卡的内容，不允许写和擦除(所有相关命令都临时无效)。这个值可以设置和复位，默认值是 0，非写保护。

● **FILE\_FORMAT**

这个值固定为 0，主机不应该使用这个值。

● **CRC**

CRC 是 CSD 内容的校验码，具体参考 4.5 章。任何 CSD 内容的改变，主机都必须重新计算 CRC。默认值与 CSD 初始内容相关。

5.4 RCA 寄存器

可写的 16 位卡相对地址寄存器，在卡的初始化期间，由卡向外发布的卡地址。这个地址用于卡初始化进程之后，主机同卡之间的交互寻址。默认 RCA 寄存器值是 0x0000，这个值保留着，用来通过 CMD7 设置所有卡到 stand-by 状态

5.5 DSR 寄存器(可选)

16 位驱动阶段寄存器，在 6.5 章详细描述。是可选的，可以用来在扩展操作条件中，提高总线性能(受总线长度，传输速率和卡数目的影响)。CSD 寄存器中有 DSR 寄存器是否使用的标志。DSR 默认值是 0x404。

5.6 SCR 寄存器

作为 CSD 寄存器的补充，另一个配置寄存器称为 SD 卡配置寄存器(SCR)。SCR 提供了 SD 卡的特殊功能的信息。SCR 是一个 64bit 的寄存器，这个寄存器应该由 SD 厂家设置。

表 5-17 SCR 值

描述	区域	宽度	类型	SCR 位
SCR 结构体	SCR_STRUCTURE	4	R	[63:60]
SD 卡规范版本	SD_SPEC	4	R	[59:56]
擦除后数据状态	DATA_STAT_AFTER_ERASE	1	R	[55]
SD 安全支持	SD_SECURITY	3	R	[54:52]
DAT 总线宽度	SD_BUS_WIDTHS	4	R	[51:48]
保留	-	16	R	[47:32]
保留给厂家使用	-	32	R	[31:0]

表 5-18 SCR 寄存器结构体版本

SCR 结构体	SCR 结构体版本	SD 卡版本
0	SCR V1.0	V1.01-V2.0
1-15	保留	

● **SD\_SPEC**

描述卡支持的物理层规范版本

表 5-19:物理层规范版本

SD_SPEC	物理层规范版本号
---------	----------

0	V1.0-1.01
1	V1.10
2	V2.0
3-15	保留

#### ● DATA\_STAT\_AFTER\_ERASE

定义了擦除之后的数据状态，可能是 0 或者 1，由厂家定义。

#### ● SD\_SECURITY

表 5-20:SD 支持安全算法

SD_SECURITY	安全规范版本
0	不支持安全
1	未使用
2	V1.01
3	V2.0
4-7	保留

注意，一个正规的可写 SD 卡是强制要求支持 Security 协议。而对于 ROM(只读)和 OTP(一次性编程)的卡来说，这是个可选项。标准容量 SD 卡，这个值设置为 2(V1.01)，高容量 SD 卡，设置为 3(V2.00)。

#### ● SD\_BUS\_WIDTHS

描述了卡支持的所有 DAT 总线宽度

表 5-12 SD 卡支持总线宽度

SD_BUS_WIDTHS	支持的总线宽度
Bit0	1bit(DAT0)
Bit1	保留
Bit2	4bit(DAT0-3)
Bit3	保留

SD 卡最少应该支持 1bit 和 4bit 的宽度，所以 SD 卡最少应该设置 bit0 和 bit2 为 1。

## 6、SD 卡硬件接口

SD 卡有 6 条交互线以及 3 条供电线：

- CMD：命令线，主机和卡都可以驱动它
- DAT0-3：数据线，主机和卡都可以驱动
- CLK：时钟线，主机发送到卡的
- Vdd：供电线
- Vss1, Vss2：两条地线

除了这些卡内部电路的线之外，还有两个写保护和卡检测切换。这俩不是强制的，但是如果有，那他们就应该像下图这么连接。当 DAT3 作为卡检测功能时，DAT3 的 Rdat 就不应该连接，而另一个电阻也应该接地。

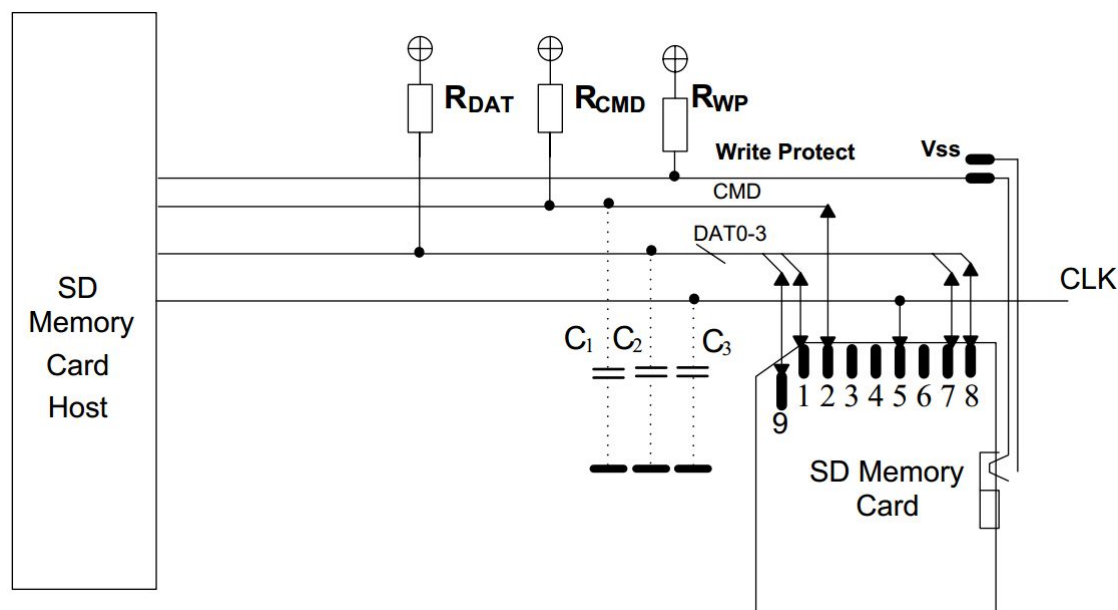


图 6-1 总线电路图

Rdat 和 Rcmd 是上拉电阻，保护 CMD 和 DAT 线，以免受到总线浮动影响，当没有卡插入，或者所有卡驱动都处于高阻抗模式。

主机应该通过 Rdat 上拉 DAT0-3 线，即使主机使用 1bit 传输模式。同样，在 SPI 模式，主机也应该上拉所有的“RSV”线，即使没有使用。

Rwp 用于写保护，卡检测切换。

具体的值参考 6.6 章。

### 6.1 热插拔

为了在热插入期间保证正确的卡管脚连接序列，强制要求使用特定热插入功能卡连接器，或者是主机侧自动轮训检测。

即使在 Vdd 正在上电的时候，可也不应该因为热插拔被损坏。数据传输操作由 CRC 进行保护，因此任何的数据改变都会被检测到，即使是热插拔。

当 clk 走了一个时钟周期 Fpp，插入的卡也应该被正确的复位。每张卡都应该有电源保护，防止卡损坏。数据传输失败由主机来检测。应用会通过重新发送来进行校正。

### 6.2 卡检测(插入/拔出)

为了给用户一个提示信息，SD 卡系统应该检测到卡的插入和拔出。一个方法是检测 SD 卡的

管脚 1，并且检测上拉电阻。详细描述在“Application Notes Relating to SD Physical Specification”。

### 6.3 电源保护(插入/拔出)

卡应该无损的进行热插拔。如果一个电源脚( $V_{DD}$  或者  $V_{SS}$ )没有正确的连接，那么电流会从一条数据线提供给卡。

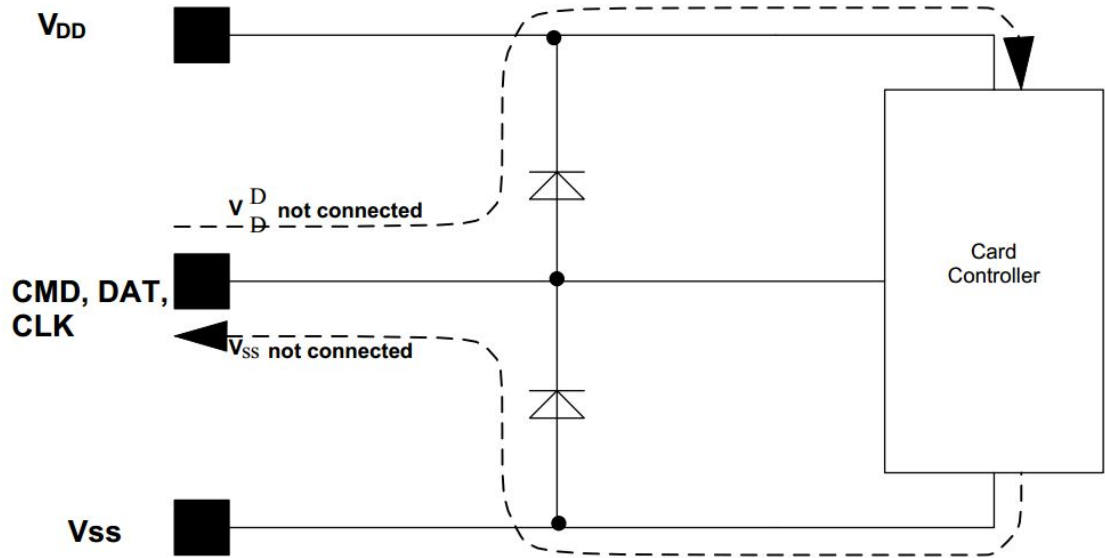


图 6-2 错误的供电

每张卡的输出都应该能够抵挡短路。

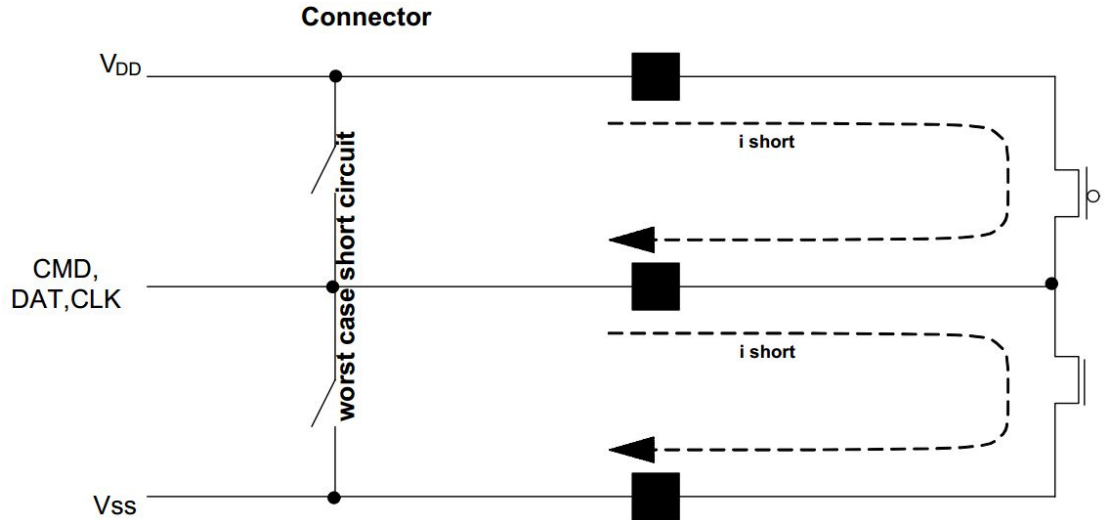


图 6-3 短路保护

如果主机支持热插拔功能，那么就应该能够抵挡瞬间的  $V_{DD}$  和  $V_{SS}$  短路，而不会损坏 SD 卡。

### 6.4 电源方案

SD 卡的电源方案是由总线和卡来决定的。

### 6.4.1 上电

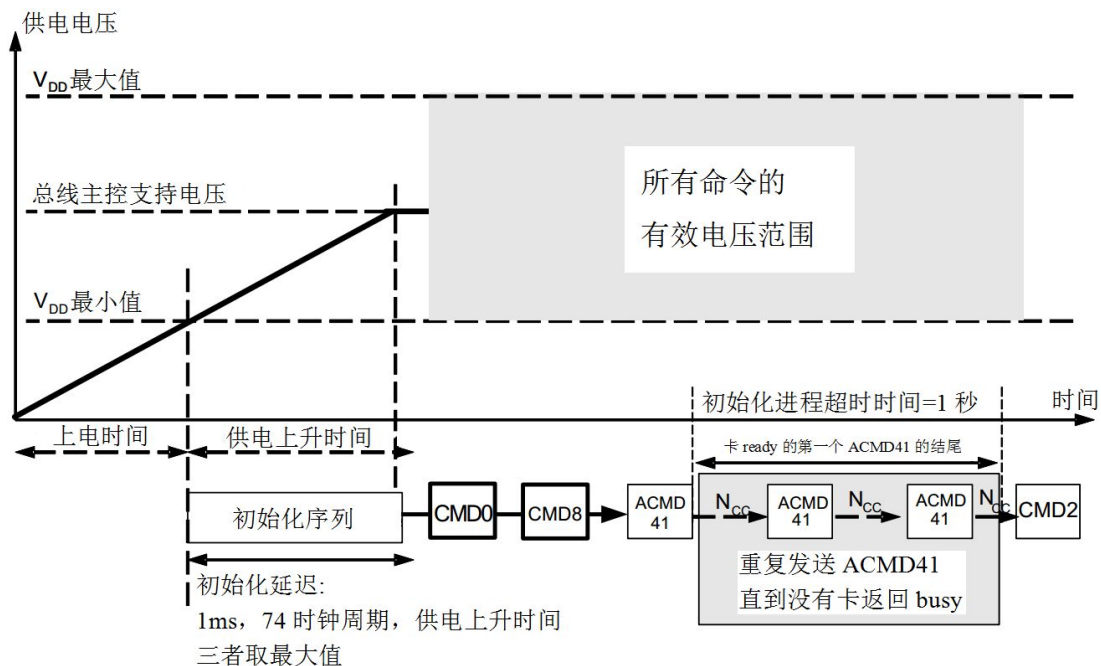


图 6-4 上电流程

- “上电时间”，指的是供电从 0V 上升到  $V_{DD}$  最小值的时间，收到应用参数影响，比如 SD 卡的最大数目，总线长度，以及供电单元的特性。
- “供电上升时间”，是电压上升到操作的电平(总线支持的)，以及等到 SD 卡能收到第一条命令的时间。
- 主机应该供电给卡，电压要在 250ms 之内达到  $V_{DD}$  最小值，并且开始提供最少 74 个 SD 卡时钟给 SD 卡，期间，保持 CMD 线为高。如果是 SPI 模式，那么 CS 应该在 74 个时钟周期内保持高电平。
- 上电之后，卡会进入 idle 状态，如果是 SD 主机，CMD0 非必需；而 SPI 主机，第一条命令就应该发送 CMD0，让卡进入 SPI 模式
- CMD8 是 V2.0 规范新添加的，为的是支持多电压范围，已经用于检查卡是否支持某个电压。V2.0 的主机在初始化前，应该发送 CMD8 并表明支持电压。不支持 CMD8 的主机要支持高电压范围。
- ACMD41 是一个同步命令，用于交涉操作电压范围，并轮询卡，直到上电时序完成。如果主机系统连接了多个卡，那么主机应该检查所有卡满足的电压。否则，主机就应该选择其中之一，然后初始化。

### 6.4.2 下电和电源周期

- 当主机下电的是否，卡  $V_{DD}$  应该减小到 0.5V 以下，并保持最少 1ms 的时间。在下电期间，DAT，CMD 和 CLK 应该断开，或者拉到 0，以防止操作电流通过信号线传递。
- 如果主机需要改变工作电压，需要一个电压周期，电压周期的意思是，电源被关闭，然后再次打开。电源周期也是范围 Inactive 状态的卡所必须的。为了创建一个电源周期，在上电之前，主机应该遵循下电描述。(比如，卡  $V_{DD}$  应该拉低小于 0.5V，并维持 1ms)。

## 6.5 可编程卡输出驱动(可选)

SD 总线上每条线的总线容量是总线主控容量，总线自身容量以及每张插入卡的容量之和。这个值对于某个应用来说是固定的，但是不同的应用是不同的。同一个应用，每张卡插入后也是不同的。

在后面，推拉模式的可编程卡输出驱动是一个可选的方法，用于保证定义的最大时钟速率，独立于拓扑结构和插入可数量之外

推挽模式的数据和命令驱动阶段有可编程的峰值电流驱动能力和可编程的上升和下降时间。驱动阶段寄存器(DSR)由两个 8bit 组成。门线线路的内容是通过接口需求的传输速度和总线加载来计算的。

CMD 和 DAT 总线驱动由一个预驱动阶段和一个补充驱动三极管组成(图 6-5)。预驱动阶段输出上升和下降时间，通过 DSR1 寄存器来设置，并决定了驱动阶段的速度。补充驱动三极管尺寸由 DSR2 寄存器来设置，决定了驱动阶段的电流驱动能力，同时也影响总线驱动的功耗性能。这二者的正确组合可以让总线性能达到最佳。表 6-1 定义了 DSR 寄存器内容：

DSR1	7	6	5	4	3	2	1	0
t(switch-on max)	保留				5ns	20ns	100ns	500ns
t(switch-on min)					2ns	10ns	50ns	200ns

DSR2	7	6	5	4	3	2	1	0
i(peak min)	保留				100mA	20mA	5mA	1mA
i(peak max)					200mA	50mA	10mA	2mA
t(rise typ)					5ns	20ns	100ns	500ns

表 6-1 DSR 寄存器内容

DSR1 里的时间表明了输出驱动三极管的切换时间。在扩展接口中，在时钟和驱动阶段输出信号之间的延迟是可以测量的。

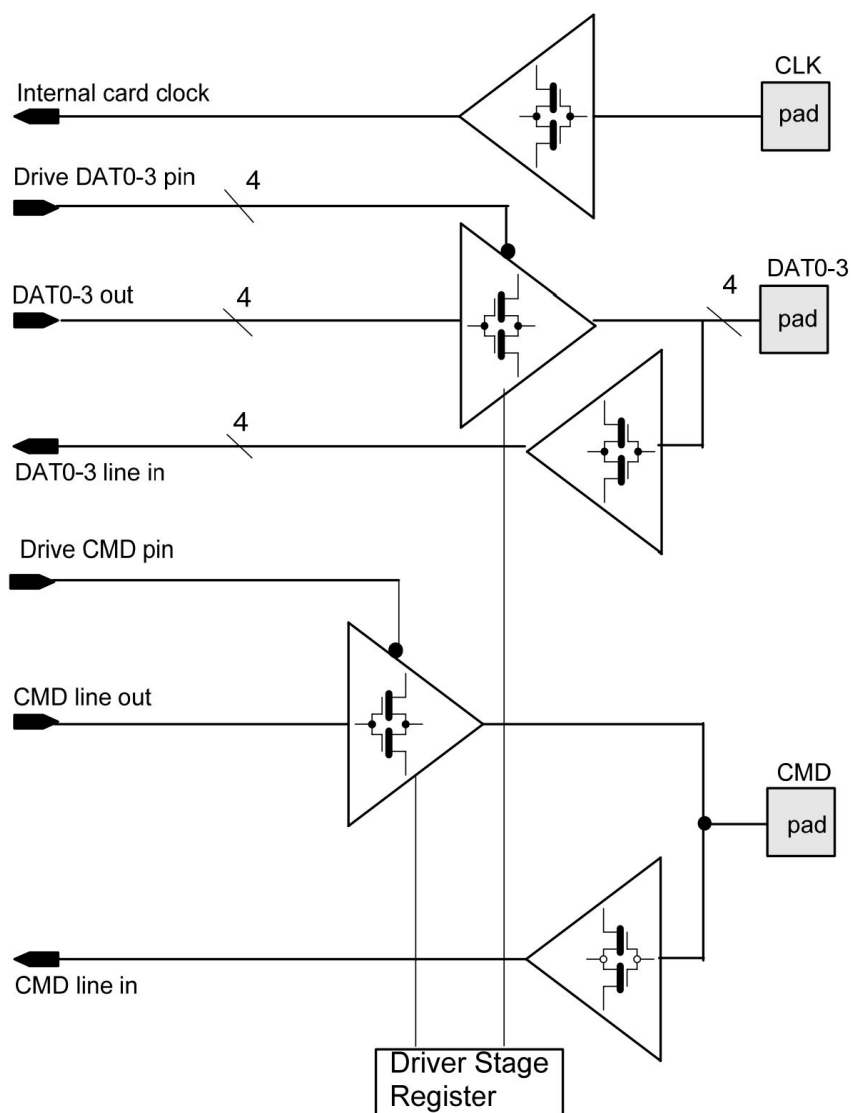


图 6-5 SD 卡总线驱动

在指定的操作范围内 (电压, 温度) 所有数据都是有效的。DSR1 和 DSR2 的任意组合都是可以的。DSR1 必须在请求的时钟频率下编程  $f_{clock} = 1/(2 \times t(\text{switch-on max}))$ 。DSR2 寄存器应该根据请求的驱动尺寸来编程。合适驱动范围选择的提示, 是未来应用注意的一部分。

## 6.6 总线操作条件

### 6.6.1 高电压范围的阈值

参数	符号	最小值	最大值	单位	注释
支持电压	Vdd	2.7	3.6	V	
输出高电压	Voh	$0.75 \times V_{dd}$		V	$I_{oh} = -100\mu A$ Vdd min
输出低电压	Vol		$0.125 \times V_{dd}$	V	$I_{ol} = 100\mu A$ Vdd min
输入高电压	Vih	$0.625 \times V_{dd}$	$V_{dd} + 0.3$	V	
输入低电压	Wil	$V_{ss} - 0.3$	$0.25 \times V_{dd}$	V	
上电时间			250	ms	$0V \sim V_{dd}$ min

表 6-2: 高电压阈值水平



### 6.6.2 低电压范围的阈值

T. B. D

### 6.6.3 通用

参数	符号	最小值	最大值	单位	注释
所有线上峰值电压		-0.3	Vdd+0.3	V	
所有输入					
输入漏电		-10	10	uA	
所有输出					
输出漏电		-10	10	uA	

表 6-3 总线操作条件-通用

### 6.6.4 电流消耗

电流消耗是以秒为单位来进行测量的

- 第一条命令之前：最大 15mA
- 初始化期间：最大 100mA
- 默认模式下操作：最大 100mA
- 高速模式下操作：最大 200mA
- 其他功能操作：最大 500mA

### 6.6.5 总线信号线加载

SD 卡总线的整体容量是，总线主控容量 Chost，总线容量 Cbus 自身，以及线上每张卡的容量 Ccard 之和

总线整体容量=Chost + Cbus + N\*Ccard

参数	符号	最小	最大	单位	注释
上拉电阻	Rcmd Rdat	10	100	k Ω	避免总线抖动
每条信号线总线容量和	C1		40	pF	1 张 Chost+Cbus <30pF
每个信号管脚的卡的容量	Ccard		10	pF	
最大信号线电感			16	nH	fpp≤20MHz
卡内上拉电阻	Rdat3	10	90	k Ω	可能用于卡检测

表 6-4：总线操作条件-信号线加载

注意，CMD 和 DAT 线的总容量包含了 Chost，Cbus 和一张卡，因为他们是分别连到主机的。主机应该为每个信号考虑总线容量和 Chost，Cbus 和 Ccard。这些参数是每条信号分别定义的。主机可以决定 Chost 和 Cbus，所以容量和和卡的关系不是太大，(C1=40pF)。当总体容量小于 C1 (40pF) 的时候 SD 卡保证他的总线时序。

### 6.6.6 总线信号电平

总线可以提供一个可变的供电电压，所有的信号电平都同供电电压有关。

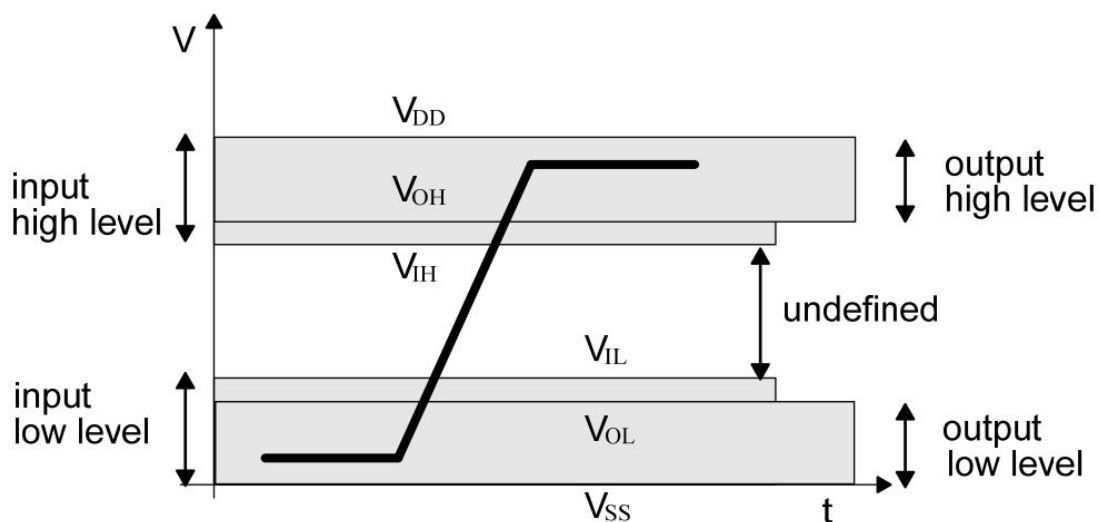


图 6-6 总线信号电平

为了满足 JEDEC 规范的需求文档 JESD8-1A 和 JESD8-7 文档,卡输入和输出电压应该在表 6-2 的范围之内。

## 6.7 总线时序(默认)

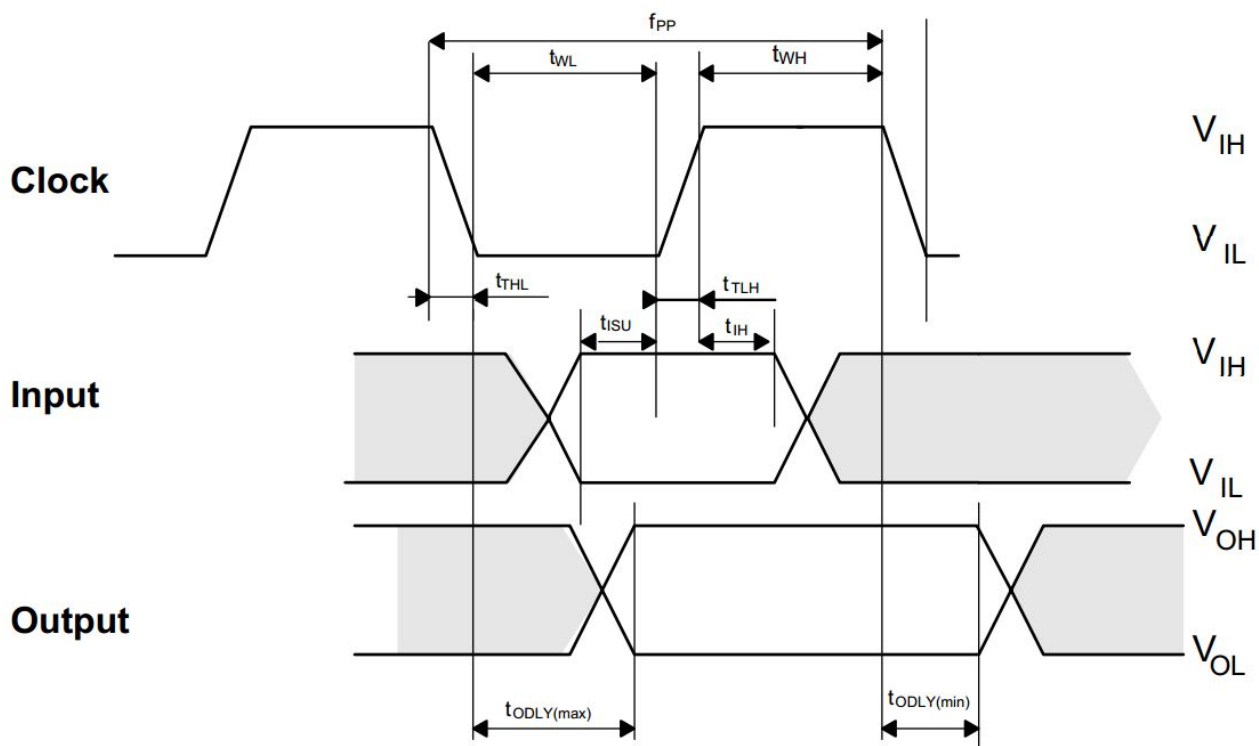


图 6-7 数据输入输出同时钟的关系图

参数	符号	最小值	最大值	单位	注释
时钟 CLK					
数据传输模式时钟频率	$f_{pp}$	0	25	MHz	$C_{card} \leq 10pF$ (一张卡)
识别模式时钟频率	$f(OD)$	0/100 ①	400	kHz	$C_{card} \leq 10pF$ (一张卡)

时钟低时间	t (WL)	10		ns	Ccard≤10pF (一张卡)
时钟高时间	t (WH)	10		ns	Ccard≤10pF (一张卡)
时钟上升时间	t (TLH)		10	ns	Ccard≤10pF (一张卡)
时钟下降时间	t (THL)		10	ns	Ccard≤10pF (一张卡)
输入命令和数据					
输入组合时间	t (ISU)	5		ns	Ccard≤10pF (一张卡)
输入持续时间	t (IH)	5		ns	Ccard≤10pF (一张卡)
输出命令数据					
数据传输模式输出延迟	t (ODLY)	0	14	ns	C1≤40pF (一张卡)
识别模式输出延时	t (ODLY)	0	50	ns	C1≤40pF (一张卡)

表 6-5 总线-时序参数值(默认)

①： 0Hz 代表停止时钟，给出最小频率范围是因为连续时钟的需要，参考 4.4 时钟控制。

6.8 总线时序(高速模式)

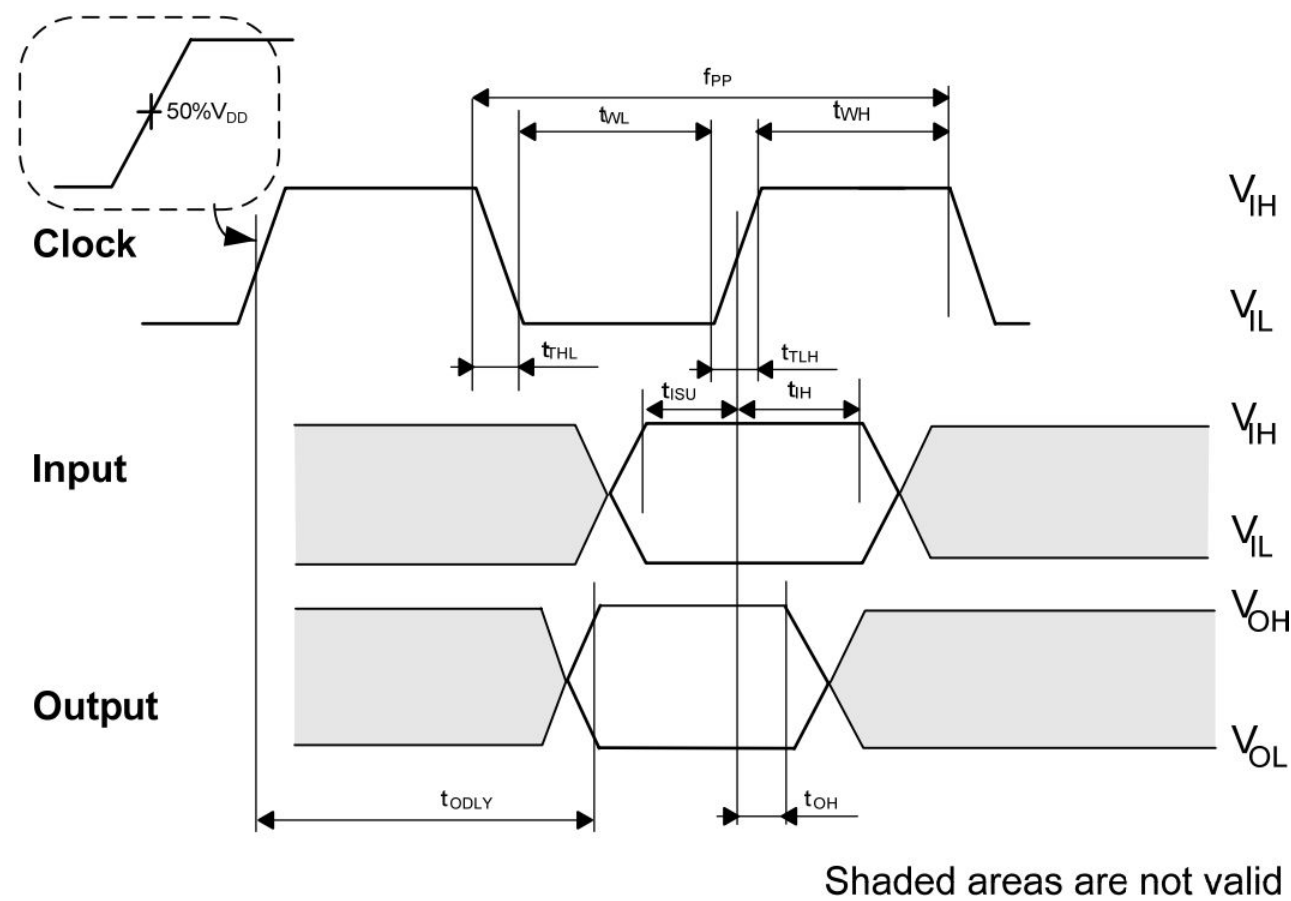


图 6-8 时钟与数据输入输出时序图(高速)

参数	符号	最小值	最大值	单位	注释
时钟 CLK(所有的值都和 minV (IH) 和 max V (IL) 相关)					
数据传输模式时钟频率	fpp	0	50	MHz	Ccard≤10pF (一张卡)
时钟低时间	t (WL)	7		ns	Ccard≤10pF (一张卡)
时钟高时间	t (WH)	7		ns	Ccard≤10pF (一张卡)

时钟上升时间	t (TLH)		3	ns	Ccard≤10pF(一张卡)
时钟下降时间	t (THL)		3	ns	Ccard≤10pF(一张卡)
输入命令和数据					
输入组合时间	t (ISU)	6		ns	Ccard≤10pF(一张卡)
输入持续时间	t (IH)	2		ns	Ccard≤10pF(一张卡)
输出命令数据					
数据传输模式输出延迟	t (ODLY)		14	ns	C (L) ≤40pF(一张卡)
输出持续时间	t (OH)	2.5		ns	C (L) ≥15pF(一张卡)
每条线的总体系统容量	C1		40	pF	一张卡

为了满足时序，主机应该只驱动一张卡

## 7、SPI 模式

(略)

## 8、SD 卡电气规格

(略)

## 附录 A

(略)

## 附录 B

(略)