

CMD55：定义下一条指令为ACMD指令

OCR: 卡操作电压寄存器 32位， 只读，每隔0.1V占1位， 第31位卡上电过程是否完成

CID: 卡身份识别寄存器 128bit,只读， 厂家号，产品号，串号，生产日期

SCR: 卡配置寄存器， 可写的 64bit 是否用Security特性(LINUX不支持)，以及数据位宽（1bit或4bit）

RCA: 卡地址寄存器： 可写的 16bit SD host和卡进行协商的一个地址，内核中会在代码里面记录这个地址，卡这      边则写到RCA寄存器

CSD: 卡专有数据寄存器 部分可读写 128bit， 卡容量，最大传输速率，读写操作的最大电流、电压，读写擦出块的最大长度等

SD卡的初始化

1. 设置SPI速度，SD卡的类型先不管
2. 片选SD卡
3. 提供时钟为SD卡内部实现自初始化
4. 发送CMD0 复位指令，判断响应
5. 发送CMD8 电压检测指令，判断响应
6. (Ⅰ) 若为标准容量或更大容量的2.0卡，且电压处于工作范围内，

发送CMD55 指令，判断响应

(Ⅱ) 若为1.0卡或其他，自己看代码吧，不想写了。

7）(Ⅰ) 发送ACMD41指令，执行初始化，判断响应

8）发送CMD58指令，读取OCR寄存器，判断响应（该寄存器此处用来判别卡的类型）

9）定义卡的类型，重设SPI速度

发送指令→SD卡

1. 片选SD卡前，对SPI总线进行清空
2. 片选SD卡，发送指令。指令为6Bytes，格式详情、格式响应自行百度或查看代码。
3. 接收响应
4. （放SPI 总线）
5. 返回响应

从SD卡中读取 n块数据

1. if 1 == n，则正常读取（发送CMD 17指令）
2. if 1 <= n，则每次一块读取（发送CMD 18指令）
3. 取消片选，发送CMD 12指令（结束读取）

向SD卡写 n块数据

Ⅰ1）写入 1块数据：发送CMD 24指令

Ⅰ2）写入数据标识符、512 B数据内容、2 B的CRC校验码

Ⅰ3）判断响应

Ⅱ1）写入多块数据：发送CMD 25指令

Ⅱ2）多次 写入数据标识符、512 B数据内容、2 B的CRC校验码

Ⅱ3）判断响应

读CSD寄存器

1. 发送CMD 9指令，判断响应

读SD卡容量

1）读CSD寄存器