SD 卡和 FATFS 文件系统

版本 ： V1.0

一、 SD 卡模块

1. SD 卡：

安全数码卡， 它是在 MMC 的基础上发展而来， 是一种基于半导体快闪记忆器的新一代记忆设备。

按容量分类，可以将 SD 卡分为 3 类： SD 卡、 SDHC 卡、 SDXC 卡。

SD 卡(SDSC)：0~2G

SDHC 卡：2~32G

SDXC 卡：32G~2T。

2. SD 卡一般支持 2 种操作模式：

SD 卡模式（通过 SDIO 通信）：

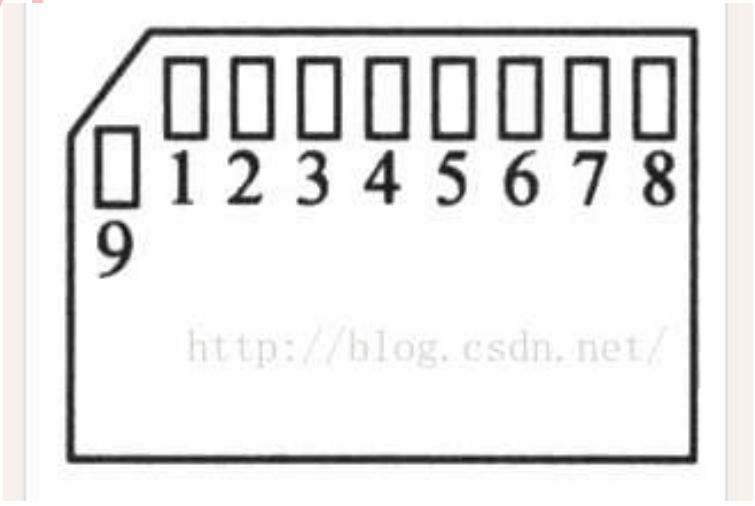
允许 4 线的高速数据传输，只能使用 3.3V 的 IO 电平，所以， MCU 一定要能够支持 3.3V 的 IO 端口输出。

SPI 模式：

同 SD 卡模式相比就是丧失了速度，在 SPI 模式下， CS/MOSI/MISO/CLK 都需要加 10~100K 左右的上拉电阻。

SD 卡引脚功能表：

针脚	1	2	3	4	5	6	7	8	9
SD 卡模式	CD/DAT3	CMD	VSS	VCC	CLK	VSS	DAT0	DAT1	DAT2
SPI 模式	CS	MOSI	VSS	VCC	CLK	VSS	MISO	NC	NC



3. SD 卡的 5 个寄存器

名称	宽度	描述
CID	128	卡标识寄存器
RCA	16	相对卡地址寄存器：本地系统中卡的地址，动态变化，在卡的初始化时确定。（SPI模式中没有）
CSD	128	卡描述数据寄存器：卡操作条件相关的信息数据。
SCR	64	SD配置寄存器：SD卡特定信息数据
OCR	32	操作条件寄存器

4. **SD 卡初始化过程（知道 SD 卡的类型 V1、 V2、 V2HC 或者 MMC）：**

- 初始化与 SD 卡连接的硬件条件（MCU 的 SPI 配置， IO 口配置）；
- 上电延时（>74 个 CLK）（因为 SD 卡内部有个供电电压上升时间，大概为 64 个 CLK，剩下的 10 个 CLK 用于 SD 卡同步，之后才能开始 CMD0 的操作）；
- 复位卡（CMD0），进入 IDLE 状态；
- 发送 CMD8，检查是否支持 2.0 协议；
- 根据不同协议检查 SD 卡（命令包括： CMD55、 CMD41、 CMD58 和 CMD1 等）；
- 取消片选，发多 8 个 CLK（提供 SD 卡额外的时钟，完成某些操作），结束初始化；

5. **SD 卡读取数据（CMD17）：**

- 发送 CMD17；
- 接收卡响应 R1；
- 接收数据起始令牌 0XFE；
- 接收数据；
- 接收 2 个字节的 CRC，如果不使用 CRC，这两个字节在读取后可以丢掉。
- 禁止片选之后，发多 8 个 CLK；

6. **SD 卡写数据（CMD24）：**

- 发送 CMD24;
- 接收卡响应 R1；
- 发送写数据起始令牌 0XFE；
- 发送数据；
- 发送 2 字节的伪 CRC；
- 禁止片选之后，发多 8 个 CLK；

二、 FATFS 文件系统

1. FATFS 是一个完全免费开源的 FAT 文件系统模块，专门为小型的嵌入式系统而设计。可以移植到 8051、 PIC、 AVR、 SH、Z80、 H8、 ARM 等系列单片机上而只需做简单的修改。它支持 FAT12、 FAT16 和 FAT32，支持多个存储媒介；有独立的缓冲区，可以对多个文件进行读 / 写，并特别对 8 位单片机和 16 位单片机做了优化。

2. FATFS 的特点有：

- Windows 兼容的 FAT 文件系统（支持 FAT12/FAT16/FAT32）
- 与平台无关，移植简单
- 代码量少、效率高
- 多种配置选项
- 支持多卷（物理驱动器或分区，最多 10 个卷）
- 多个 ANSI/OEM 代码页包括 DBCS
- 支持长文件名、 ANSI/OEM 或 Unicode
- 支持 RTOS
- 支持多种扇区大小
- 只读、最小化的 API 和 I/O 缓冲区等

3. 与平台无关的是：

ffconf.h	FATFS 模块配置文件
ff.h	FATFS 和应用模块公用的包含文件
ff.c	FATFS 模块
diskio.h	FATFS 和 disk I/O 模块公用的包含文件
interger.h	数据类型定义
option	可选的外部功能（比如支持中文等）

4. 与平台相关的是：

diskio.c	FATFS 和 disk I/O 模块接口层文件

FATFS 模块在移植的时候，我们一般只需要修改 2 个文件，即 ffconf.h 和 diskio.c。

5. diskio.c 和 diskio.h 是硬件层，需要根据存储介质来修改

ff.c 和 ff.h 是 FATFS 的文件系统层和文件系统的 API 层

6. 移植步骤：
   1) 数据类型：在 integer.h 里面去定义好数据的类型。这里需要了解你用的编译器的数据类型，并根据编译器定义好数据类型。
   2) 配置：通过 ffconf.h 配置 FATFS 的相关功能，以满足你的需要。
   3) 函数编写：打开 diskio.c，进行底层驱动编写，一般需要编写 6 个接口函数

7. FATFS 给用户提供了大量的 API 函数，可以满足我们对文件的各种操作。

f_mount - 注册/注销一个工作区域（Work Area）
f_open - 打开/创建一个文件
f_close - 关闭一个文件
f_read - 读文件
f_write - 写文件
f_lseek - 移动文件读/写指针
f_truncate -截断文件
f_sync - 冲洗缓冲数据 Flush Cached Data
f_forward - 直接转移文件数据到一个数据流
f_stat - 获取文件状态
f_opendir - 打开一个目录

f_closedir -关闭一个已经打开的目录
f_readdir - 读取目录条目
f_mkdir - 创建一个目录
f_unlink -删除一个文件或目录
f_chmod - 改变属性（Attribute）
f_utime -改变时间戳（Timestamp）
f_rename - 重命名/移动一个文件或文件夹
f_chdir - 改变当前目录
f_chdrive - 改变当前驱动器
f_getcwd - 获取当前工作目录
f_getfree - 获取空闲簇 Get Free Clusters

f_getlabel - Get volume label
f_setlabel - Set volume label
f_mkfs - 在驱动器上创建一个文件系统
f_fdisk - Divide a physical drive
f_gets - 读一个字符串
f_putc - 写一个字符
f_puts - 写一个字符串
f_printf - 写一个格式化的字符串
f_tell - 获取当前读/写指针
f_eof - 测试文件结束
f_size - 获取文件大小
f_error - 测试文件上的错误

8. 几个重要结构体：

- 文件对象结构体（FIL 类型）：存放文件的相关信息，打开关闭读写文件等操作时需要使用其指针
- 目录对象结构体（DIR 类型）：存放目录的相关信息，对目录操作时需要其指针
- 文件状态结构体（FILINFO 类型）：存放文件的大小属性文件名等信息
- 文件系统对象结构体（FATFS 类型）

附录：

1.SD卡的命令格式

SD卡的指令由6字节(Byte)组成，如下：

Byte1：0 1 x x x x x x(命令号，由指令标志定义，如CMD39为100111即16进制0x27，那么完整的CMD39第一字节为01100111，即0x27+0x40)

Byte2-5:Command Arguments,命令参数，有些命令没有参数

Byte6:前7位为CRC(Cyclic Redundacy Check，循环冗余校验)校验位，最后一位为停止位0