



Hi3510 通过 SSP 接口连接 SD 卡

Application Notes

文档版本	01
发布日期	2006-09-30
BOM编码	N/A

深圳市海思半导体有限公司为客户提供全方位的技术支持，用户可与就近的海思办事处联系，也可直接与公司总部联系。

深圳市海思半导体有限公司

地址： 深圳市龙岗区坂田华为基地华为电气生产中心 邮编： 518129

网址： <http://www.hisilicon.com>

客户服务电话： 0755-28788858

客户服务传真： 0755-28788838

客户服务邮箱： support@hisilicon.com.

版权所有 © 深圳市海思半导体有限公司 2006。 保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



、Hisilicon、海思，均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。



目 录

前 言.....	1
1 概述.....	1-1
2 硬件接口.....	2-1
2.1 SSP 接口	2-1
2.2 SD 卡接口	2-2
2.3 SD 卡检测.....	2-3
2.4 接口示例	2-3
3 软件说明.....	3-1
3.1 块设备驱动接口层	3-1
3.2 SD 卡读写物理层	3-2
3.3 程序处理流程	3-2
4 操作示例.....	4-1
4.2 插入 SD 卡	4-2
4.3 用 fdisk 工具分区	4-2
4.3.1 查看当前状态.....	4-2
4.3.2 创建新的分区.....	4-3
4.3.3 保存分区信息.....	4-4
4.4 用 mkfs.vfat 工具格式化.....	4-4
4.5 挂载目录	4-4
4.6 读写文件	4-5
5 热插拔操作.....	5-1
5.1 无读写状态的热插拔操作	5-1
5.2 正在读写状态的热插拔操作	5-2
6 推荐 SD 卡.....	6-1



插图目录

图 2-1 SPI bus 模式下 SD 卡管脚定义	2-3
图 2-2 SPI bus 模式下 SD 卡接口示意图	2-4
图 2-3 读取 SD 卡的时序示意图.....	2-4
图 3-1 SD 卡软件接口.....	3-1
图 3-2 程序处理流程.....	3-3
图 4-1 SD 卡读写操作流程.....	4-1



表格目录

表 2-1 Hi3510 的 SSP 接口信号	2-1
表 2-2 SSP 的 SSP_CR0 寄存器定义	2-2
表 3-1 块设备驱动接口层函数	3-1
表 3-2 SD 卡读写物理层函数	3-2



前言

概述

本节介绍本文档的内容、对应的产品版本、适用的读者对象、行文表达约定、历史修订记录等。

产品版本

与本文档相对应的产品版本如下所示。

产品名称	产品版本
Hi3510 通信媒体处理器芯片（简称 Hi3510）	Hi3510 V100 Hi3510 V110
Hi3510 DVS 解决方案	Hi3510 DMS V100R001
Hi3510 Video Phone 方案	Hi3510 DMS V200R001

读者对象

本指南为程序员和系统工程师而编写，描述了基于 Hi3510 媒体解决方案平台，如何通过 SSP 接口连接 SD 卡。使用本书的程序员应该：

- 相关硬件基本知识
- 熟练掌握 C 语言
- 掌握基本的 Linux 环境编程

内容简介

本文描述了利用 Hi3510 芯片的 SSP 接口连接 SD 卡，Hi3510 芯片的 SSP 接口支持 SPI 模式，可以与 SPI bus 模式的 SD 卡连接使用。

章节	内容
1 概述	简要介绍 SD 卡特点及 Hi3510 可以支持 SD 应用。

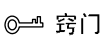



章节	内容
2 硬件接口	介绍实现 SPI Bus 模式 SD 卡的硬件接口，给出了参考示例。
3 软件接口	介绍 Linux 内核文件系统对 SD 卡块设备进行操作的软件接口及处理流程。
4 操作示例	以 128MB 的 SD 卡为例，介绍在控制台下如何对 SD 卡进行操作的详细过程。
5 热插拔操作	针对 SD 卡的特点，介绍各种情况下的热插拔操作。
6 推荐 SD 卡	描述推荐使用的 SD 卡。

约定

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	以本标志开始的文本表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害。
 警告	以本标志开始的文本表示有中度或低度潜在危险，如果不能避免，可能导致人员轻微或中等伤害。
 注意	以本标志开始的文本表示有潜在风险，如果忽视这些文本，可能导致设备或器件损坏、数据丢失、设备性能降低或不可预知的结果。
 窍门	以本标志开始的文本能帮助您解决某个问题或节省您的时间。
 说明	以本标志开始的文本是正文的附加信息，是对正文的强调和补充。

通用格式约定

格式	说明
宋体	正文采用宋体表示。
黑体	一级、二级、三级标题采用黑体。



格式	说明
楷体	警告、提示等内容一律用楷体，并且在内容前后增加线条与正文隔离。
“Terminal Display” 格式	“Terminal Display” 格式表示屏幕输出信息。此外，屏幕输出信息中夹杂的用户从终端输入的信息采用加粗字体表示。

命令行格式约定

格式	意义
粗体	命令行关键字（命令中保持不变、必须照输的部分）采用加粗字体表示。
<i>斜体</i>	命令行参数（命令中必须由实际值进行替代的部分）采用斜体表示。
[]	表示用“[]”括起来的部分在命令配置时是可选的。
{ x y ... }	表示从两个或多个选项中选取一个。
[x y ...]	表示从两个或多个选项中选取一个或者不选。
{ x y ... } *	表示从两个或多个选项中选取多个，最少选取一个，最多选取所有选项。
[x y ...] *	表示从两个或多个选项中选取多个或者不选。

修改记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修改日期	版本	修改说明
2006-09-30	01（第一次正式稿发布）	



1 概述

SD 卡（Secure Digital Card）是一种可广泛使用的可移动便携存储介质，具有容量大、体积小、速度快、价格低廉和可加密等特点。

SD 卡有两种连接方式：

- SD bus
- SPI bus

Hi3510 内部集成 SSP（Synchronous Serial Protocol）接口，支持 SPI（Serial Peripheral Interface）模式。因此可以通过 Hi3510 的 SSP 接口，采用 SPI 模式与 SD 卡连接，实现 SD 卡的读写操作。

关于 SD 卡的相关资料请参考：http://www.sdcard.org/sd_memorycard/index.html。



2 硬件接口

2.1 SSP 接口

Hi3510 的 SSP 接口支持 LVTTL/LVCMOS 3.3V 电平，如表 2-1 所示。

说明

在 SSP 接口的 SPI 模式下，片选信号 SSPSRM 不满足 SD 卡 SPI bus 模式的时序要求，因此需要用 GPIO 来模拟片选信号。

表2-1 Hi3510 的 SSP 接口信号

信号名	方向	类型	频率(Hz)	驱动(mA)	功能
SSPSCLK	O	LVTTL/ LVCMOS	<10M	4	SSP 总线时钟。
SSPRXD	I	LVTTL	<10M	-	SSP 总线数据接收。
SSPTXD	O	LVTTL/ LVCMOS	<10M	4	SSP 总线数据发送。
SSPSFRM	O	LVTTL/ LVCMOS	<10M	4	SSP 帧或从设备选择输出信号。

Hi3510 的 SSP 接口可以工作在多种模式下：

- SPI
- Microwire
- TI synchronous serial。

SSP 的 SPI 模式选择由 SSP_CR0 寄存器确定，如表 2-2 所示。



表2-2 SSP 的 SSP_CR0 寄存器定义

比特	名称	描述																
[15:8]	SCR	<p>串行时钟率。SCR 的值用来产生 PrimeCell SSP 发送和接收的比特率（即 SPICK 频率）。</p> <p>比特率由下面公式计算： $FSSPCLK / (CPSDVR \times (1 + SCR))$ FSSPCLK: SSP 主时钟输入。 CPSDVR: Clock prescale divisor, 时钟预分频系数。 CPSDVR 通过配置寄存器 SSP_CPSR 编程, 当 CPSDVR 是一个 2~254 之间的偶数时候, SCR 是一个在 0~255 之间的值。</p>																
[7]	SPH	SSPCLKOUT 相位（只对于 Motorola SPI 帧格式适用），在本例中设置为 1。																
[6]	SPO	SSPCLKOUT 电平（只对于 Motorola SPI 帧格式适用），在本例中设置为 1。																
[5:4]	FRF	<p>帧格式。</p> <p>00: Motorola SPI 帧格式; 10: National Microwire 帧格式; 01: TI 同步串行帧格式; 11: 保留。</p>																
[3:0]	DSS	<p>数据大小选择。</p> <table border="0"> <tr> <td>0000: 保留;</td> <td>1000: 9 bit data;</td> </tr> <tr> <td>0001: 保留;</td> <td>1001: 10 bit data;</td> </tr> <tr> <td>0010: 保留;</td> <td>1010: 11 bit data;</td> </tr> <tr> <td>0011: 4 bit data;</td> <td>1011: 12 bit data;</td> </tr> <tr> <td>0100: 5 bit data;</td> <td>1100: 13 bit data;</td> </tr> <tr> <td>0101: 6 bit data;</td> <td>1101: 14 bit data;</td> </tr> <tr> <td>0110: 7 bit data;</td> <td>1110: 15 bit data;</td> </tr> <tr> <td>0111: 8 bit data;</td> <td>1111: 16 bit data.</td> </tr> </table>	0000: 保留;	1000: 9 bit data;	0001: 保留;	1001: 10 bit data;	0010: 保留;	1010: 11 bit data;	0011: 4 bit data;	1011: 12 bit data;	0100: 5 bit data;	1100: 13 bit data;	0101: 6 bit data;	1101: 14 bit data;	0110: 7 bit data;	1110: 15 bit data;	0111: 8 bit data;	1111: 16 bit data.
0000: 保留;	1000: 9 bit data;																	
0001: 保留;	1001: 10 bit data;																	
0010: 保留;	1010: 11 bit data;																	
0011: 4 bit data;	1011: 12 bit data;																	
0100: 5 bit data;	1100: 13 bit data;																	
0101: 6 bit data;	1101: 14 bit data;																	
0110: 7 bit data;	1110: 15 bit data;																	
0111: 8 bit data;	1111: 16 bit data.																	

例如选择 SSP 接口的 SPI 模式，8 位数据位，只需要配置寄存器 SSP_CR0 中的 SPH=1，SPO=1，FRF=0，DSS=7。

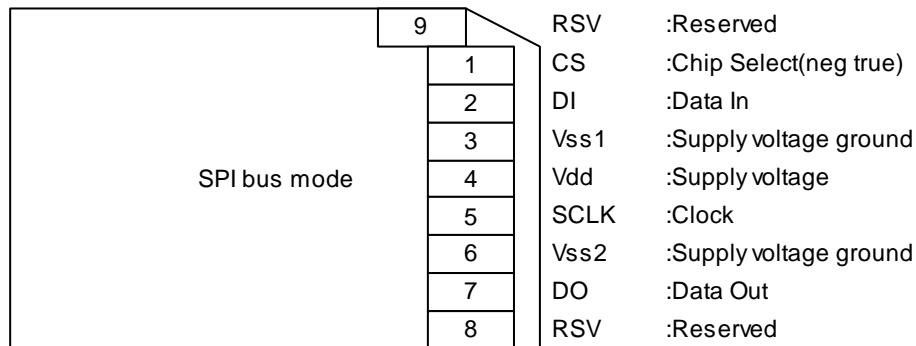
关于 SSP 模块工作的 SPI 模式和其他相关寄存器的详细信息，请参考《Hi3510 V100 通信媒体处理器芯片 用户指南》。

2.2 SD 卡接口

SPI bus 采用 4 线制：CS、SCLK、DI、DO。SPI bus 模式下工作的 SD 卡的管脚定义如图 2-1 所示。



图2-1 SPI bus 模式下 SD 卡管脚定义



2.3 SD 卡检测

在 Hi3510 VSEVB (Video Solution Evaluation Board) 上, SD 卡工作在 SPI bus 模式。在评估板上电或热插拔等情况下, 都能自动对 SD 卡进行检测。

- 插卡操作
 - 评估板上 Hi3510 芯片的 GPIO3_3 管脚由高电平变为低电平, 表明检测到 SD 卡, 产生中断通知 Hi3510 芯片。
 - Hi3510 芯片使能电源管理模块程序, 开始给 SD 卡供电, 以使 SD 卡可以开始工作。
- 拔卡操作
 - 评估板上 Hi3510 芯片的 GPIO3_3 管脚由低电平变为高电平, 表明未检测到 SD 卡, 产生中断通知 Hi3510 芯片。
 - Hi3510 芯片不使能电源管理模块程序, 关闭 SD 卡电源。

当插入电源管理模块程序时, 在模块的初始化过程中会首先检测是否存在 SD 卡。

- 若检测到 SD 卡, 使能电源管理模块, 开始给 SD 卡供电。
- 若检测不到 SD 卡, 不使能电源管理模块, 关闭 SD 卡电源。

在插、拔卡操作触发中断后, 中断处理程序先屏蔽中断, 并延迟 10ms 去抖, 然后检测是否存在 SD 卡, 选择是否给 SD 卡供电, 并打开中断。

说明

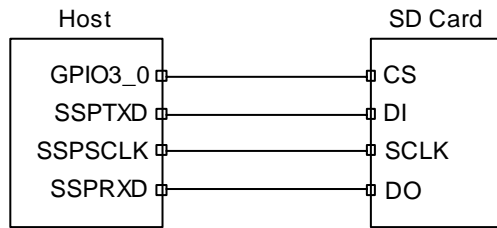
如果没有插入电源管理模块程序, SD 卡不能工作。

2.4 接口示例

当 SD 卡工作在 SPI bus 模式下, 一般连接如图 2-2 所示。片选信号利用 GPIO3_0 模拟。

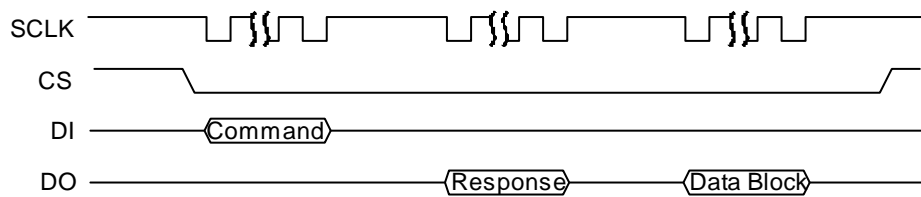


图2-2 SPI bus 模式下 SD 卡接口示意图



读取 SD 卡数据的时序如图 2-3 所示。

图2-3 读取 SD 卡的时序示意图

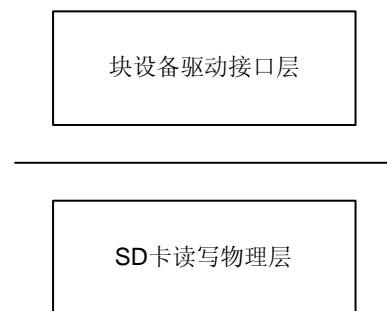




3 软件说明

通过 SPI 模式实现 SD 卡的软件接口分为两层，如图 3-1 所示。

图3-1 SD 卡软件接口



- 块设备驱动接口层
提供 Linux 内核文件系统访问接口，通过块设备驱动层，内核对块设备进行读写访问。
- SD 卡读写物理层
提供块设备驱动访问 SD 卡的接口。

3.1 块设备驱动接口层

按照 Linux 下标准的块设备驱动结构，在块设备驱动接口层都是采用 Linux 下块设备的标准函数，如表 3-1 所示。

表3-1 块设备驱动接口层函数

函数	描述
static int __init sd_init (void)	模块初始化函数
static void __exit sd_exit (void)	模块卸载函数



函数	描述
static int sd_open (struct inode *inode, struct file *filp)	打开函数
static int sd_release (struct inode *inode, struct file *filp)	关闭函数
static void sd_request (request_queue_t *q)	请求处理函数
static int sd_ioctl (struct inode *inode, struct file *filp, unsigned int cmd, unsigned long arg)	其他命令处理函数

3.2 SD 卡读写物理层

块设备驱动接口函数调用 SD 卡读写物理层函数，实现 SD 卡的读写操作。SD 卡读写物理层函数如表 3-2 所示。

表3-2 SD 卡读写物理层函数

函数	描述
unsigned int sdcard_init (void)	SD 卡初始化函数
unsigned int sdcard_read (unsigned int addr, unsigned int off, unsigned int block_num)	SD 卡读函数
unsigned int sdcard_write (unsigned int addr, unsigned int off, unsigned int block_num)	SD 卡写函数

3.3 程序处理流程

Linux 内核文件系统对块设备的读写操作（包含对分区表、文件的操作），通过请求函数完成。

驱动程序只负责处理由内核文件系统发出对某个扇区的读写请求，以及返回磁头、扇区、柱面数等信息。为降低 CPU 资源的占用率，驱动程序在初始化 sd_init () 函数时创建一个内核线程来处理请求，而在 sd_request () 函数中唤醒这个内核线程，以最大程度释放 CPU 资源。

在 Linux 内核发出的请求队列中，每个单独请求都包含具体读写信息（读/写、扇区地址、扇区数、数据），根据这些信息去调用 SD 卡读写函数（sdcard_read ()，sdcard_write ()），完成请求操作。

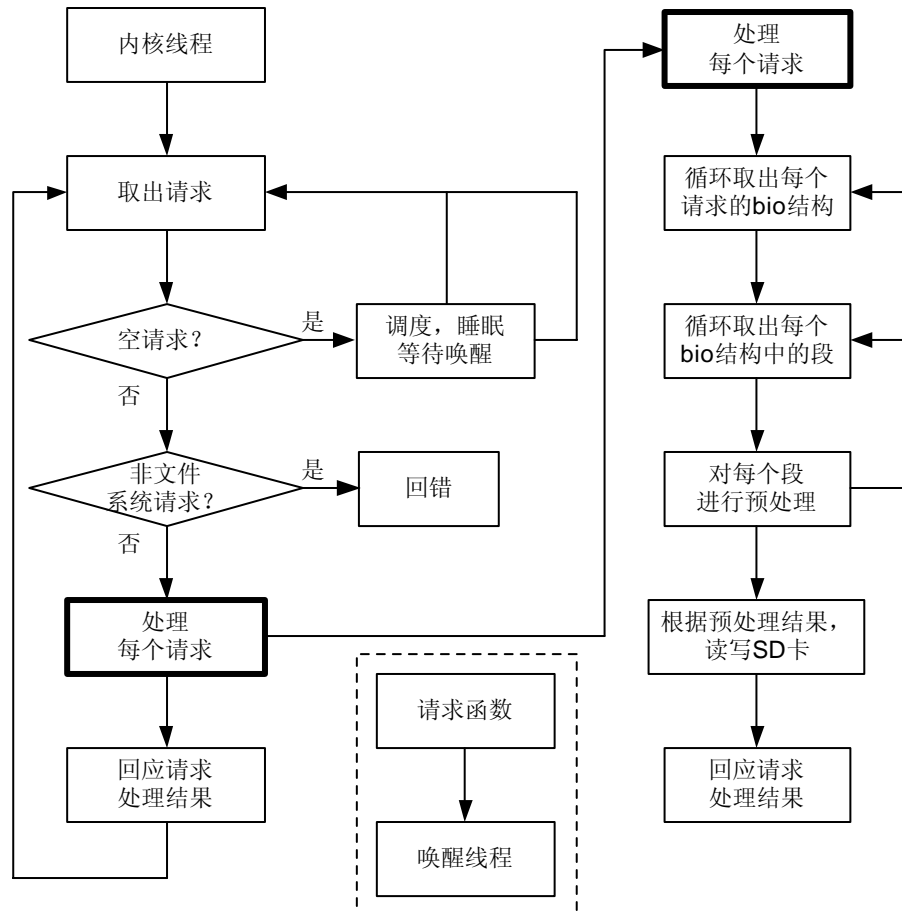
在处理每个单独请求时，分为两步：

1. 分析读写请求，并根据扇区地址和扇区数进行合并处理。
2. 调用 SD 卡读写函数，一次性完成读写请求。



实际测试证明：这样可以提高读写速度和稳定性，并降低了 CPU 资源的占用率。
Linux 内核文件系统对块设备的读写操作处理流程如图 3-2 所示。

图3-2 程序处理流程

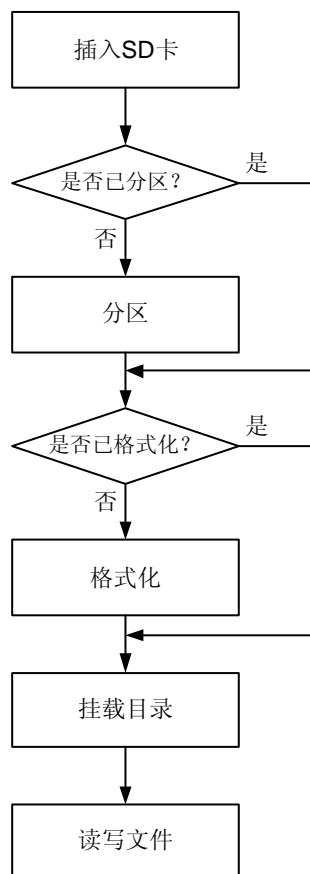




4 操作示例

以 128MB 的 SD 卡为例，通过 SSP 接口的 SPI 模式实现 SD 卡的读写操作，如图 4-1 所示。本例是在控制台下实现读写 SD 卡操作的。

图4-1 SD 卡读写操作流程





4.2 插入 SD 卡

在控制台上手动输入以下命令（以下模块如果已经插入了，就不需要再插入）：

```
insmod hi_gpio.ko
insmod misc_gpio.ko
insmod hi_ssp.ko
insmod sd.ko
```

控制台显示如下提示信息：

```
SD128 127139840 bytes
/dev/sd/0: p1
$
```

- 若显示出 p1，表示已经进行分区。
- 若没有显示出 p1，表示还没有分区，需要对 SD 卡进行分区。

4.3 用 fdisk 工具分区



说明

如果已经有分区，本操作可以跳过。

4.3.1 查看当前状态

本例是在一片没有分区的 SD 卡上进行操作。

1. 选择 fdisk 工具进行分区

在控制台的提示符下，输入命令 **fdisk** 进行分区：

```
~ $ fdisk /dev/sd/0/disc
```

2. 查看当前分区状态

在控制台的提示符下，输入命令 **m**，

在控制台的提示符下，输入命令 **p**，查看当前分区状态：

```
Command (m for help): p
```



说明

输入命令 **m**，可以查看 **fdisk** 所有命令的详细信息。

控制台显示出分区状态信息：

```
Disk /dev/sd/0/disc: 127 MB, 127139840 bytes
8 heads, 32 sectors/track, 970 cylinders
Units = cylinders of 256 * 512 = 131072 bytes
```

Device	Boot	Start	End	Blocks	Id	System
--------	------	-------	-----	--------	----	--------

上面信息表明 SD 卡没有分区，需要对 SD 卡进行分区。



4.3.2 创建新的分区

1. 创建新的分区

在提示符下输入命令 **n**，创建新的分区：

```
Command (m for help): n
```

控制台显示出如下信息：

```
Command action
  e  extended
  p  primary partition (1-4)
```

2. 建立主分区

输入命令 **p**，选择主分区：

```
p
```

3. 选择分区数

本例中选择为 1，输入数字 1：

```
Partition number (1-4): 1
```

控制台显示出如下信息：

```
First cylinder (1-970, default 1):
```

4. 选择起始柱面

本例选择默认值 1，直接回车：

```
Using default value 1
```

5. 选择结束柱面

本例选择默认值 970，直接回车：

```
Last cylinder or +size or +sizeM or +sizeK (1-970, default 970):
```

```
Using default value 970
```

6. 选择系统格式

由于系统默认为 Linux 格式，本例中选择 Win95 FAT 格式，输入命令 **t** 进行修改：

```
Command (m for help): t
```

```
Selected partition 1
```

输入命令 **b**，选择 Win95 FAT 格式：

```
Hex code (type L to list codes): b
```

说明

输入命令 **l**，可以查看 **fdisk** 所有分区的详细信息。

```
Changed system type of partition 1 to b (Win95 FAT32)
```

7. 查看分区状态

输入命令 **p**，查看当前分区状态：

```
Command (m for help): p
```



控制台显示出当前分区状态信息，表示成功分区：

```
Disk /dev/sd/0/disc: 127 MB, 127139840 bytes
8 heads, 32 sectors/track, 970 cylinders
Units = cylinders of 256 * 512 = 131072 bytes

Device Boot      Start      End  Blocks  Id System
/dev/sd/0/part1   1          970   124144   b  Win95 FAT32
```

4.3.3 保存分区信息

输入命令 **w**，写入并保存分区信息到 SD 卡：

```
Command (m for help): w
```

控制台显示出当前 SD 卡信息，表示成功写入分区信息到 SD 卡：

```
The partition table has been altered!

Calling ioctl() to re-read partition table.
/dev/sd/0: p1
/dev/sd/0: p1

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
~ $
```

4.4 用 mkfs.vfat 工具格式化



说明
如果已经格式化，本操作可以跳过。

输入命令 **mkfs.vfat** 进行格式化：

```
~ $ mkfs.vfat /dev/sd/0/part1
```

控制台显示出如下提示信息，表示成功格式化：

```
mkfs.vfat 2.11 (12 Mar 2005)
~ $
```

4.5 挂载目录

使用命令 **mount** 挂载到 **mnt** 目录下，就可以进行读写文件操作：

```
~ $ mount -t vfat /dev/sd/0/part1 /mnt
```



4.6 读写文件

读写操作的具体情况很多，在本例中使用命令 **cp** 实现读写操作。

使用命令 **cp** 拷贝当前目录下的 **test.txt** 文件到 **mnt** 目录下，即拷贝至 SD 卡，实现写操作：

```
~ $ cp ./test.txt /mnt
```



5 热插拔操作

通过 SSP 接口的 SPI 模式，可支持 SD 卡在带电状态下插入或拔出的热插拔操作，也支持 SD 卡正在读写操作状态下进行热插拔操作。

5.1 无读写状态的热插拔操作

取卡操作

在控制台中输入 **umount** 命令进行卸载：

```
umount /mnt
```

系统显示如下提示信息，表示成功卸载：

```
close SD power!
```

插卡操作

插入 SD 卡后，系统显示如下提示信息，表示成功识别 SD 卡：

```
open SD power!  
SD128 127139840 bytes
```

使用 **mount** 命令进行挂载：

```
mount -t vfat /dev/sd/0/part1 /mnt
```

挂载完成后，即可对 SD 卡进行读写操作。



5.2 正在读写状态的热插拔操作



注意

热插拔的存储类设备（SD 卡，U 盘等），在拔除的时候都应该进行 `umount` 操作。正在读写 SD 卡，如果进行热插拔操作，可能会破坏文件的完整性。

- 占用 SD 卡容量。
- 文件名可能存在，也可能不存在。
- 如果删除文件，系统会设置 SD 卡为只读，此时需要重新挂载才可写 SD 卡。

当 SD 卡正处于读写操作时，进行取卡操作，系统显示如下提示信息和读写错误信息，并回到命令行状态（以下显示的出错信息为标准错误输出信息，仅供参考）：

```
close SD power!  
  sdcard_write error offset=4649 num=1024  
end_request: I/O error, dev sd0, sector 5673  
Buffer I/O error on device sd0p1, logical block 5641  
lost page write due to I/O error on sd0p1  
Buffer I/O error on device sd0p1, logical block 5642  
lost page write due to I/O error on sd0p1  
Buffer I/O error on device sd0p1, logical block 5643  
lost page write due to I/O error on sd0p1  
Buffer I/O error on device sd0p1, logical block 5644  
lost page write due to I/O error on sd0p1  
Buffer I/O error on device sd0p1, logical block 5645  
lost page write due to I/O error on sd0p1  
Buffer I/O error on device sd0p1, logical block 5646  
lost page write due to I/O error on sd0p1  
Buffer I/O error on device sd0p1, logical block 5647  
lost page write due to I/O error on sd0p1  
Buffer I/O error on device sd0p1, logical block 5648  
lost page write due to I/O error on sd0p1  
Buffer I/O error on device sd0p1, logical block 5649  
lost page write due to I/O error on sd0p1  
Buffer I/O error on device sd0p1, logical block 5650  
lost page write due to I/O error on sd0p1  
end_request: I/O error, dev sd0, sector 6697  
end_request: I/O error, dev sd0, sector 7721  
end_request: I/O error, dev sd0, sector 33  
end_request: I/O error, dev sd0, sector 275  
end_request: I/O error, dev sd0, sector 517  
/mnt $
```

重新插入 SD 卡（如果插入另外一张 128MB SD 卡，操作相同），系统显示如下提示信息，表示成功识别 SD 卡。

```
open SD power!  
SD128 127139840 bytes
```

1. 用 `umount` 命令进行卸载：



```
umount /mnt
```

2. 使用 **mount** 命令进行挂载:

```
mount -t vfat /dev/sd/0/part1 /mnt
```

3. 完成挂载后, 即可对 SD 卡进行读写操作。

如果是插入一张没有分区的 SD 卡, 具体操作请参见“[4 操作示例](#)”。



6 推荐 SD 卡

经过充分测试，证明 SanDisk、Kingston、KingMax 三个品牌的 1G 和 2G 容量的 SD 卡运行性能良好，推荐使用。