外设

使用指南

文档版本　10

发布日期　2018-10-10

商标声明

 、HISILICON、海思和其他海思商标均为海思技术有限公司的商标。
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

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

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。

上海海思技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： http://www.hisilicon.com/cn/

客户服务邮箱： support@hisilicon.com

前言

概述

本文档主要是指导使用SATA、PCIe、SDIO、ETH以及USB 2.0/3.0 Host等驱动模块的相关人员，通过一定的步骤和方法对和这些驱动模块相连的外围设备进行控制，主要包括操作准备、操作过程、操作中需要注意的问题以及操作示例。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3798C	V1XX
Hi3798C	V2XX
Hi3796C	V1XX
Hi3798M	V1XX
Hi3796M	V1XX
Hi3798M	V2XX(H)
Hi3796M	V2XX
Hi3798M	V3XX(H)
Hi3716M	V43X

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

作者信息

章节号	章节名称	作者信息
01	SD/MMC操作指南	L00227819
02	ETH操作指南	L00227819
03	USB2.0操作指南	L00231238
04	3G上网卡	L00227819
05	PCIe 操作指南	J00174387
06	SATA 操作指南	J00174387
07	附录	L00227819

修订记录

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。

修订日期	版本	修订说明
2014-04-30	00B01	第1次临时版本发布。
2014-10-30	01	新增支持Hi3796MV100芯片。
2015-04-30	02	新增PCIe和SATA说明。
2015-12-16	03	新增支持Hi3798CV200芯片。
2016-7-11	04	修正SD卡操作截图和部分格式错误。
2016-11-04	05	新增支持Hi3798MV200芯片。
2017-04-05	06	新增支持Hi3796MV200芯片。
2017-08-31	07	新增支持Hi3798MV300芯片。
2017-12-22	08	第4章开头新增一个说明。 HiSTBLinuxV100R005C00SPC052不支持3G上网卡。
2018-05-21	09	新增1.2章节的步骤3。
2018-10-10	10	新增支持Hi3798MV310、Hi3798MV300H、 Hi3798MV200H、Hi3716MV430芯片。

目 录

1 SD/MMC 卡操作指南

1.1 操作准备

硬件环境：SD /MMC 卡

软件环境：SDK发布包。

1.2 操作过程

操作过程如下：

步骤1 启动单板，加载根文件系统。

步骤2 加载SDIO相关模块。

默认SDIO相关模块已经全部编入内核，不需要再执行加载命令。

内核配置对MMC/SD/SDIO卡支持的配置选项位于Device Drivers -> MMC/SD/SDIO card support，配置如图1-1、图1-2所示。(请选择红框选中的选项)

图 1-1 MMC/SD/SDIO 启动支持选项

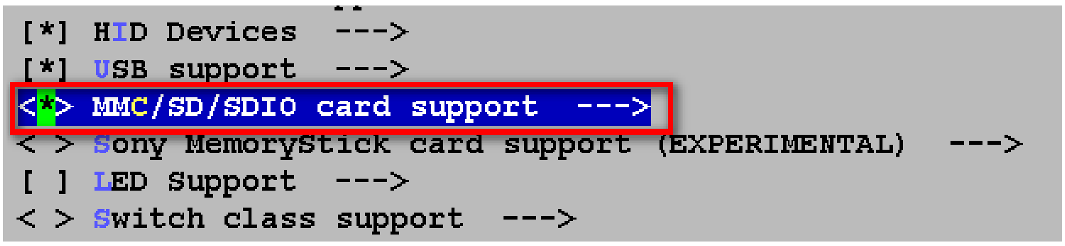
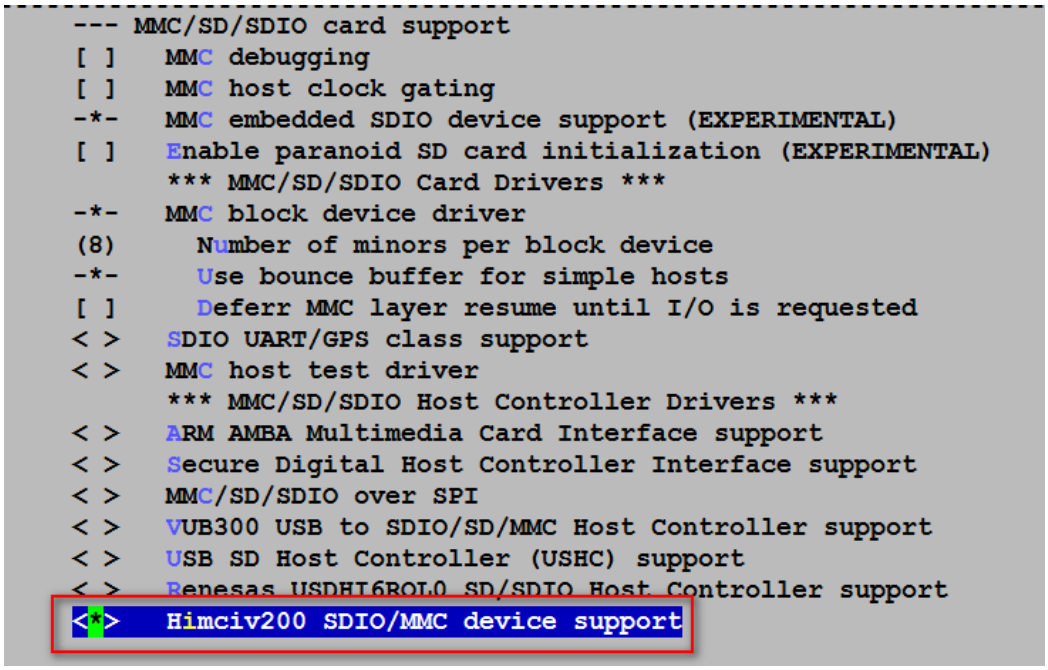


图 1-2 MMC/SD/SDIO 驱动配置选项



步骤3 针对Hi3798MV200/Hi3798MV300/Hi3798MV310当使用dms单板接SD卡/wifi时，要根据硬件设计情况配置控制器的能力项。

- 当硬件上没有power en电源开关通路时，只支持SD卡2.0模式，因此需要配置控制器能力项为2.0模式。
- 当硬件上有power en电源开关电路时，支持SD卡3.0模式，因此需要配置控制器能力项为3.0模式。
- 当接wifi时不受电源开关通路影响，因此需要配置控制器能力项为3.0模式。

配置控制器能力项方法：

```
/device/hisilicon/bigfish/sdk/source/kernel/linux-3.18.y.patch/arch/arm/
boot/dts/hi3798mv200.dts 或 hi3798mv310.dts 中
sd:himciv200.SD@0xf9820000 {
caps = <0x8007000f>;
};
sdio:himciv200.SD@0xf9c40000 {
caps = <0x8007000f>;
};
```

- sd:himciv200.SD@0xf9820000对应sdio0控制器
  - caps=<0x8007000f> 16-18bit配置为1代表该控制器能力项为3.0模式
  - caps=<0x8000000f> 16-18bit配置为0代表该控制器能力项为2.0模式
- sdio:himciv200.SD@0xf9c40000对应sdio1控制器
  - caps=<0x8007000f> 16-18bit配置为1代表该控制器能力项为3.0模式
  - caps=<0x8000000f> 16-18bit配置为0代表该控制器能力项为2.0模式.

早期版本代码中有强制限制控制器能力为2.0模式的代码，当dms单板sdio0接sdio wifi时要删除如下代码：

```
/device/hisilicon/bigfish/sdk/source/kernel/linux-3.18.y.patch/
drivers/mmc/host/himciv300/himciv300.c函数himciv300_pltm_probe中
```



```
#if defined(CONFIG_ARCH_HI3798MV2X)
/* not support sdio3.0 on hi3798mv200 DMS board sdio0 */
{
    void __iomem *reg_board_type;
    u32 regval;
   reg_board_type = ioremap_nocache(REG_BOARD_TYPE, sizeof(u32));
   if (!reg board type) {
   printk("%s %s iomap fail\n",  func  ,dev name(host->dev));
                                          return;
   }
   regval = readl(reg_board_type);
   regval &= (0x1<<30);
   iounmap(reg board type);
   if ((regval) &&(strcmp(dev_name(&pdev->dev), "f9820000.himciv200.SD")
== 0)) {
   mmc->caps &= ~(MMC_CAP_UHS_SDR12 | MMC_CAP_UHS_SDR25
   |MMC_CAP_UHS_SDR50 | MMC_CAP_UHS_SDR104);
  }
}
#endif
```

步骤4 插入SD/MMC卡，就可以对SD/MMC卡进行相关的操作。具体操作请参见“**1.3 操作示例1.3 操作示例**”。

说明

调试时如果对驱动有改动，需要重新编译内核。关于某个模块与其内核选项的对应关系，请参见**7 附录7.5 内核中模块与其内核选项的说明**的说明。

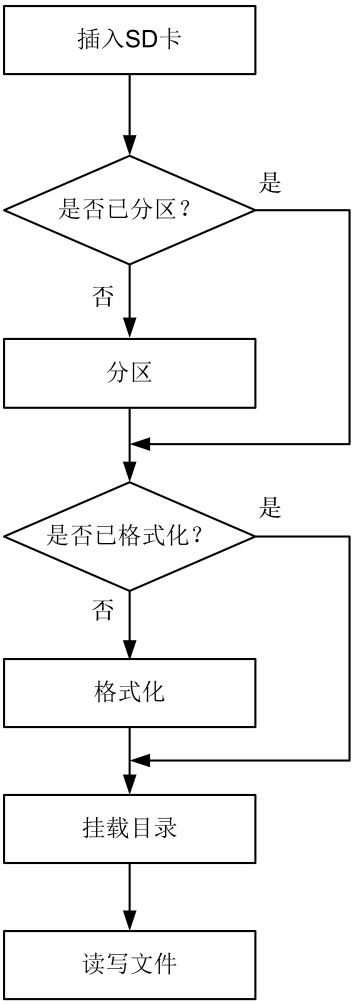
----结束

1.3 操作示例

此操作示例通过SDIO接口实现SD卡的读写操作，MMC卡的读写操作和SD卡类似，这里不再举例。

在控制台下实现读写SD卡的操作示例如**图1-3**所示。

图 1-3 在控制台下实现读写 SD 卡的操作示例



初始化及应用

模块插入完成后，进行如下操作：

说明

如无特殊说明，本文档中所有命令均为linux环境下的执行命令。如果为android环境，需要在命令前加”busybox”；

mmcblk1pX为设备节点名称，linux环境下位于/dev目录下，android环境下位于/dev/block/mmcblkY目录下。其中Y为设备号，X为分区号，由fdisk工具分区时决定。

- 命令fdisk操作的具体目录需改为：~ $ fdisk /dev/mmcblk1
- 用mkdosfs工具格式化的具体目录需改为：~ $ mkdosfs －F 32 /dev/mmcblk1pX

步骤1 挂载的具体目录需改为：~ $ mount -t vfat /dev/mmcblk1pX /mnt查看分区信息。

- 若没有显示出p1，表示还没有分区，请参见“**7.1 用fdisk工具分区7.1 用fdisk工具分区**”进行分区后，进入**步骤2**。
- 若有分区信息p1，则SD/MMC卡已经检测到，并已经进行分区，进入**步骤2**。

步骤2 查看格式化信息。

- 若没有格式化，请参见“7.2 格式化设备7.2 格式化设备”进行格式化后，进入步骤3。
- 若已格式化，进入步骤3。

步骤3 挂载目录，请参见“7.3 挂载目录7.3 挂载目录”。

步骤4 对SD/MMC卡进行读写操作，请参见“7.4 读写文件7.4 读写文件”。

----结束

1.4 操作中需要注意的问题

在正常操作过程中需要遵守的事项：

- SD卡有写保护开关（位于SD卡侧边），如果要对SD卡进行写操作，需要将写保护拨至无效以关闭写保护，此后mount SD卡之后才能写，否则SD卡只读。
- 保证卡的金属片与卡槽硬件接触充分良好（如果接触不好，会出现检测错误或读写数据错误），测试薄的MMC卡，必要时可以用手按住卡槽的通讯端测试。
- 每次需要读写SD卡时，必须确保SD卡已经创建分区，并且已经正确格式化为某种格式，比如vfat格式（通过fdisk和mkdosfs命令，具体过程参见1.3 操作示例1.3 操作示例）。
- 每次插入SD卡后，需要做一次mount操作挂载文件系统，才能读写SD卡；如果SD卡已经挂载到文件系统，读写文件完毕后，正常的操作顺序是先执行umount操作，然后才能拔卡，否则可能出现异常。
- 异常拔卡（未执行umount就直接拔卡）有可能会造成SD卡数据的丢失，请慎重，同时下次挂载SD卡时也有可能会出现问题，所以不建议这样操作，如果不小心这样操作后，也要再执行umount挂载点的操作。

在正常操作过程中不能进行的操作：

- 读写SD卡时不要拔卡，否则会打印一些异常信息，并且可能会导致卡中文件或文件系统被破坏。
- 当前的工作目录位于挂载目录时，不能执行umount操作，必须转到挂载目录以外才能umount操作。
- 系统中读写挂载目录的进程没有完全退出时，不能执行umount操作，必须完全结束操作挂载目录的任务才能正常执行umount操作。

在操作过程中出现异常时的操作：

- 如果在循环测试过程中异常拔卡，如果出现一直不停的打印异常操作信息，需要按ctrl+c回退到shell下。
- 拔卡后，再极其快速地再次插入卡时可能会出现检测不到卡的现象，因为卡的检测注册/注销过程需要一定的时间。
- 异常拔卡后，必须先执行umount操作，再执行mount操作，否则不能读写挂载点目录如/mnt，并会打印异常信息。
- SD有多分区时，可以通过mount操作切换挂载不同的分区，但最后umount操作次数与mount操作次数相等时，才会完全umount所有的挂载分区。
- 如果在读写SD卡时直接拔卡，导致SD卡文件系统破坏，重新插卡并挂载，读写卡时可能会出现异常，这时，需要umount操作，拔卡，再次插卡并mount，才能正常读写SD卡。

2 ETH 操作指南

2.1 Linux 环境下的基本操作

启动串口终端，将目标机通过串口与PC连接，对于目标机有多网口的情况，正确连接目标机相应网口。在Linux环境下，网口基本使用方法如下（以eth0为例，对于双网口的单板，具体哪个网口对应eth0请咨询硬件相关人员）：

- 由于网口驱动已编入内核，因此不需要额外加载驱动
- 设置MAC地址

  在超级终端上执行shell命令：**ifconfig eth0 hw ether XX:XX:XX:XX:XX:XX**

  其中：**eth0**是网口名，**XX:XX:XX:XX:XX:XX**是MAC地址。
- 设置网口IP地址

  在超级终端上执行shell命令：**ifconfig eth0 XXX.XXX.XXX.XXX**

  其中：**eth0**是网口名，**xxx.xxx.xxx.xxx**是IP地址。
- 设置子网掩码

  在超级终端上执行shell命令：**ifconfig eth0 netmask XXX.XXX.XXX.XXX**

其中：**eth0**是网口名，**Netmask**后面的**XXX.XXX.XXX.XXX**为子网掩码。

2.2 其他操作示例

在Linux环境下，除了前面描述的基本使用外，网口还具有其他的用途以及使用方法。

2.2.1 Route 命令

删除路由

- 命令

  删除路由的命令如下：

  ```
  route del -net 10.85.180.0 netmask 255.255.254.0 dev eth0
  ```

  其中：**10.85.180.0**是目标网段，**255.255.254.0**是目标网段的子网掩码，**dev eth0**是目标网段经由的网口名。
- 举例

- 删除子网为10.85.180.0，掩码为255.255.254.0通过eth0的路由表：

```
route del -net 10.85.180.0 netmask 255.255.254.0 dev eth0
~ $ route
Kernel IP routing table
Destination  Gateway      Genmask    Flags  Metric Ref    Use
Iface
default      10.85.180.1  0.0.0.0    UG      0       0
0    eth0
```

- 删除默认路由：

```
route del default
~ $ route
Kernel IP routing table
Destination   Gateway  Genmask       Flags  Metric  Ref     Use
Iface
192.168.0.0   *         255.255.254.0   U     0      0
0     eth0
default      10.85.180.1 0.0.0.0      UG      0      0
0     eth0
~ $ route del default
~ $ route
Kernel IP routing table
Destination   Gateway   Genmask         Flags Metric Ref    Use
Iface
192.168.0.0   *          255.255.254.0  U      0        0
0    eth0
```

添加路由

注意

添加路由需要保证网口与gateway之间存在物理链路，并且能通信。

- 命令

  添加路由的命令如下：

```
route add -net 10.85.180.0 netmask 255.255.254.0 dev eth0
```

  其中：**10.85.180.0**是目标网段，**255.255.254.0**是子网掩码，**eth0**是该网段通过哪个网口通信。

- 举例

  - 使子网10.85.180.0 掩码为255.255.254.0 经由eth0通信

```
route add -net 10.85.180.0 netmask 255.255.254.0 dev eth0
~ $ route
Kernel IP routing table
Destination  Gateway       Genmask    Flags  Metric  Ref    Use
Iface
default       10.85.180.1  0.0.0.0     UG     0        0
0     eth0
~ $ route add -net 10.85.180.0 gateway 10.85.180.1 netmask
255.255.254.0 dev eth0
~ $ route
Kernel IP routing table
Destination  Gateway       Genmask   Flags  Metric   Ref  Use
Iface
default      10.85.180.1   0.0.0.0   UG      0         0
0      eth0
10.85.180.0  *       255.255.254.0   U       0          0
```

```
0 eth0
~ $
```

- 添加默认路由

  当需要与不存在路由表的子网通信时，可设置默认路由：

  ```
  route add default gw 10.85.180.1 dev eth0
  ```

  指明当需要与不存在路由表的子网通信时，将通过eth0，网关10.85.180.1转发。

  ```
  ~ $ route add default gw 10.85.180.1 dev eth0
  ~ $ route
  Kernel IP routing table
  Destination     Gateway         Genmask         Flags   Metric   Ref  Use
  Iface
  10.85.180.0     *               255.255.254.0   U         0          0      0
  eth0
  default         10.85.180.1     0.0.0.0         UG        0          0      0
  eth0
  ~ $
  ~ $ ping 10.110.10.1
  PING10.110.10.1 (10.110.10.1): 56 data bytes
  84 bytes from 10.110.10.1: icmp_seq=0 ttl=244 time=32.1 ms
  84 bytes from 10.110.10.1: icmp_seq=1 ttl=244 time=32.0 ms

  --- 10.110.10.1 ping statistics ---
  2 packets transmitted, 2 packets received, 0% packet loss
  round-trip min/avg/max = 32.0/32.0/32.1 ms
  ```

  更多详细信息请参见Linux PC上帮助手册《Linux man pages man route》。

2.2.2 设置 DNS

编辑/etc/resolv.conf，加入语句：

```
nameserver dnsip
```

其中：**nameserver**是关键字，**dnsip**是DNS服务器的IP地址。

完成之后，可以使用**ping www.xxxx.com**测试是否成功。

2.2.3 配置转发功能

注意

待机转发，需要保证MAC和网口PHY正常供电以及时钟正常工作。

请参考待机的UNF接口配置网口转发。

3 USB 操作指南

3.1 操作准备

硬件环境：U盘、USB鼠标、USB键盘和USB-WiFi模块

3.2 软件环境：SDK 发布包操作过程

操作过程如下：

步骤1 如果在内核配置选项中把USB_OHCI_HCD_PLATFORM、USB_EHCI_HCD_PLATFORM和USB_XHCI_HISILICON编译成模块，方法为将Device Drivers --->USB support --->Generic EHCI driver for a platform device,xHCI support for Hisilicon SoCs和Generic OHCI driver for a platform device

- 配置为[M]，则需要手动加载下面三个模块（位于内核源码目录drivers/usb/host）；
- 配置成[*]，则是将USB_OHCI_HCD_PLATFORM、USB_EHCI_HCD_PLATFORM和USB_XHCI_HISILICON编译进内核，此时无需手动加载，内核在启动之后自动就加载相应驱动。

加载USB驱动模块命令：

```
insmod ohci-platfrom.ko
insmod ehci-platfrom.ko
insmod xhci-plat-hcd.ko
```

步骤2 进入文件系统后，插入设备，然后就可以对U盘、鼠标或者键盘进行相关的操作了。具体操作请参见“3.3 操作示例3.3 操作示例”。

说明

调试时如果对驱动有改动，需要重新编译内核。关于某个模块与其内核选项的对应关系，请参见7 附录7.5 内核中模块与其内核选项的说明的说明。

SDK发布版本默认会配置USB相关配置项，而且系统启动时会自动加载相应的.ko文件，无需手动加载。

----结束

3.3 操作示例

3.3.1 U 盘操作示例

插入检测

直接插入U盘，观察是否枚举成功。

以Linux为例，正常情况下串口打印为：

```
~ $ usb 2-2: new high speed USB device using hi godbox and address 2
        scsi0 : SCSI emulation for USB Mass Storage devices
        Vendor: Generic   Model: USB Flash Disk    Rev: 0.00
        Type:   Direct-Access                      ANSI SCSI revision: 02
        SCSI device sda: 32243711 512-byte hdwr sectors (16509 MB)
        sda: Write Protect is off
        sda: assuming drive cache: write through
        SCSI device sda: 32243711 512-byte hdwr sectors (16509 MB)
        sda: Write Protect is off
        sda: assuming drive cache: write through
        sda: sda1
        Attached scsi removable disk sda at scsi0, channel 0, id 0, lun 0
```

其中：sda1表示U盘或移动硬盘上的第一个分区（此例假定对U盘只分了一个区），当存在多个分区时，会出现sda1 sda2 sda3的字样。特别地，如果U盘没有分区，则不会出现sda1、sda2等信息，只有sda的信息。

初始化及应用

模块插入完成后，进行如下操作：

说明

mmcblk1pX为设备节点名称，linux环境下位于/dev目录下，android环境下位于/dev/block/目录下。其中X为分区号，由fdisk工具分区时决定。如无特殊说明，本文档中android环境下不支持命令行格式化NTFS/ext3/ext4的工具。

- 命令fdisk操作的具体目录需改为：~ $ fdisk /dev/sda
- 用mkdosfs工具格式化的具体目录需改为：~ $ mkdosfs -F 32 /dev/sdaX
- 挂载的具体目录需改为：~ $ mount -t vfat /dev/sdaX /mnt
- U盘分区可以格式化成fat32格式，还可以格式化成ntfs格式或者ext3/ext4格式，实际应用中可以根据场景选择实际的文件系统格式。
  - 对于ntfs格式，相应的格式化命令为：~ $ mkntfs /dev/sdaX；如果ntfs文件系统格式损坏，还可以使用命令进行修复：~ $ chkntfs -a -f /dev/sdaX。
  - 对于ext3/ext4格式，使用方法与在eMMC使用方法类似。以下如无特殊说明，默认将U盘格式化成fat32格式。

步骤1 查看分区信息。

- 若没有分区信息sda1，表示还没有分区，请参见“**7.1 用fdisk工具分区7.1 用fdisk工具分区**”进行分区后，进入**步骤2**。
- 若有分区信息sda1，则已经检测到U盘，并已经进行分区，进入**步骤2**。

步骤2 查看格式化信息。

- 若没有格式化，请参见“**7.2 格式化设备7.2 格式化设备**”进行格式化后，进入**步骤3**。
- 若已格式化，进入**步骤3**。

步骤3 挂载目录，请参见“7.3 挂载目录7.3 挂载目录”。

步骤4 对硬盘进行读写操作，请参见“7.4 读写文件7.4 读写文件”。

----结束

3.3.2 键盘操作示例

键盘操作过程如下：

步骤1 插入模块。

插入键盘相关模块后，键盘会在/dev/input/目录下生成event0节点。

步骤2 接收键盘输入。

执行命令：cat /dev/input/event0

该命令就是将目标板上的USB键盘输入打印到终端上：在USB键盘上敲击，可以看到屏幕有输出。

----结束

3.3.3 鼠标操作示例

鼠标操作过程如下：

步骤1 插入模块。

插入鼠标相关模块后，鼠标会在/dev/input/目录下生成mouse0节点。

步骤2 运行gpm中提供的标准测试程序（建议使用mev）。

步骤3 进行鼠标操作（点击、滑动等），可以看到串口打印出相应码值。

----结束

3.3.4 USB-WiFi 操作示例

USB-WiFi的操作方法请参见《HMS 开发指南》的WiFi章节。

3.3.5 USB 串口操作示例

3.3.5.1 在内核下添加 USB 转串口驱动

在内核下添加USB转串口驱动的操作步骤如下：

步骤1 进入Linux内核目录，配置内核，以3.18.y版本内核为例：

```
cd source/kernel/linux-3.18.y
make ARCH=arm CROSS_COMPILE=arm-histbv310-linux- menuconfig
```

步骤2 按照如下配置：

```
Device Drivers  --->
       [*] USB support  --->
              <*>   USB Serial Converter support  --->
                    [*]   USB Serial Console device support
                    [*]   USB Generic Serial Driver
                    <*>   USB Prolific 2303 Single Port Serial Driver
```

说明

内核支持多种USB串口设备，Prolific 2303 Single Port Serial是一种常用USB串口设备，如果使用其它串口设备，请选择相应的驱动。

步骤3 保存配置，重新编译内核镜像。

这样配置之后，在/dev目录下可以看到出现类似ttyUSB0的设备，此设备便是USB转串口的设备名，应用可以像操作普通串口一样，操作这个USB转串口设备(ttyUSB0)。

----结束

3.3.5.2 将串口重定向到 USB 串口(ttyUSB0)

为了将系统启动时的内核打印和文件系统打印都重定向到ttyUSB0，并且将ttyUSB0作为系统控制端口，需要作如下修改：

步骤1 修改bootargs，将console=ttyAMA0修改为console=ttyUSB0。

步骤2 修改单板文件系统的启动初始化脚本，去掉“::respawn:-/bin/sh”，增加一行：“::respawn:/sbin/getty ttyUSB0 115200 vt100”。

步骤3 保存并重启单板。将USB转串口线与PC端的串口连接，重启单板可以看到，系统控制终端变成了USB转串口设备，内核启动的打印信息都通过USB转串口设备输出。并且可以通过USB转串口设备操作单板，比如向单板下发shell命令。

----结束

3.3.5.3 注意事项

- 系统控制台重定向到USB转串口设备之后，系统启动时会弹出提示框提示输入用户名和密码，按照提示操作即可。
- 如果用户想不需要输入密码即可登录系统，可以更改单板文件系统的/etc/passwd文件，将对应用户名的密码设为空即可，例如，当root用户登录时不需要输入密码，修改方法如下：

  修改前：

  root:x:0:0:root:/root:/bin/sh

  修改后：

  root::0:0:root:/root:/bin/sh
- 重启单板。

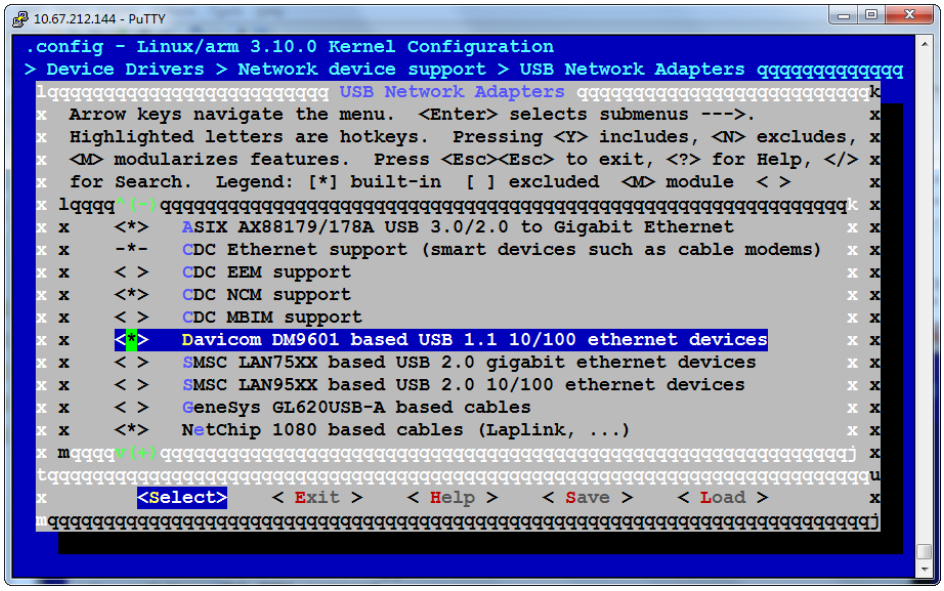
3.3.6 USB 网卡操作示例

以Hi3798C V200平台为例，USB网卡型号以沐阳JP1081（芯片型号为DM9601）为例。

步骤1 首先需要在内核下打开USB网卡驱动，先执行cd source/kernel/linux-3.18.y;make hi3798cv200_defconfig ARCH=arm CROSS_COMPILE=arm-histbv310-linux-;make menuconfig ARCH=arm CROSS_COMPILE=arm-histbv310-linux-

步骤2 选中Device Drivers -> Network device support -> USB Network Adapters -> Multi-purpose USB Networking Framework -> Davicom DM9601 based USB 1.1 10/100 ethernet devices。

图 3-1 打开 USB 网卡的配置



步骤3 退出并保存配置，然后执行cp .config arch/arm/configs/hi3798cv200_defconfig,注意必须要执行这一步后，才能将修改保存到hi3798cv200_defconfig。

步骤4 执行cd ../../..，退回到SDK根目录，并执行make linux。

步骤5 重新烧录内核，单板上电，插入USB网卡，此时会生成一个ethX设备，可以通过ifconfig命令查看、配置ip，与通常的以太网设备使用方法类似。注意，默认ethX设备可能没有up，需要在ifconfig加上-a参数查看。

----结束

3.4 操作中需要注意的问题

- 对于USB设备的操作请参见地址：http://www.usb.org/developers/compliance/。
- 在操作时请尽量按照完整的操作顺序进行操作（mount→操作文件→umount），以免造成文件系统的异常。
- 目前键盘和鼠标的驱动要和上层结合使用，比如鼠标事件要和上层的GUI结合。对键盘的操作只需要对/dev下的event节点读取即可，而鼠标则需要标准的库支持。
- 在Linux系统中提供了一套标准的鼠标应用接口libgpm，如果需要是用鼠标客户可自行编译此库。在使用时建议使用内核标准接口gpm。

  已测试通过的标准接口版本：gpm-1.20.5。

  另外在gpm中还提供了一整套的测试工具源码（如：mev等），用户可根据这些测试程序进行编码等操作，降低开发难度。

4 3G 上网卡

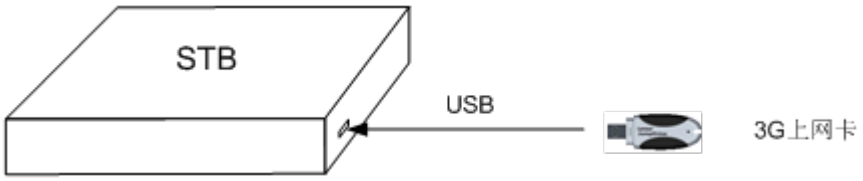
4.1 操作准备

使用3G上网卡前的环境准备如下：

- 硬件环境：3G上网卡
- 软件环境：SDK发布包

硬件连接图如图4-1所示，3G上网卡成功连接到网络后，机顶盒即可以通过3G上网卡上网。

图 4-1 硬件连接



4.2 操作过程

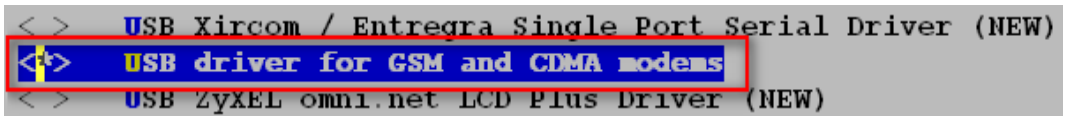
操作过程如下：

步骤1 配置内核支持3G上网卡，在内核目录下，使用make menuconfig命令。

选择usb转串口驱动“USB Serial Converter support”，同时选择modem串口驱动“USB driver for GSM and CDMA modems”，如图4-2、图4-3所示。

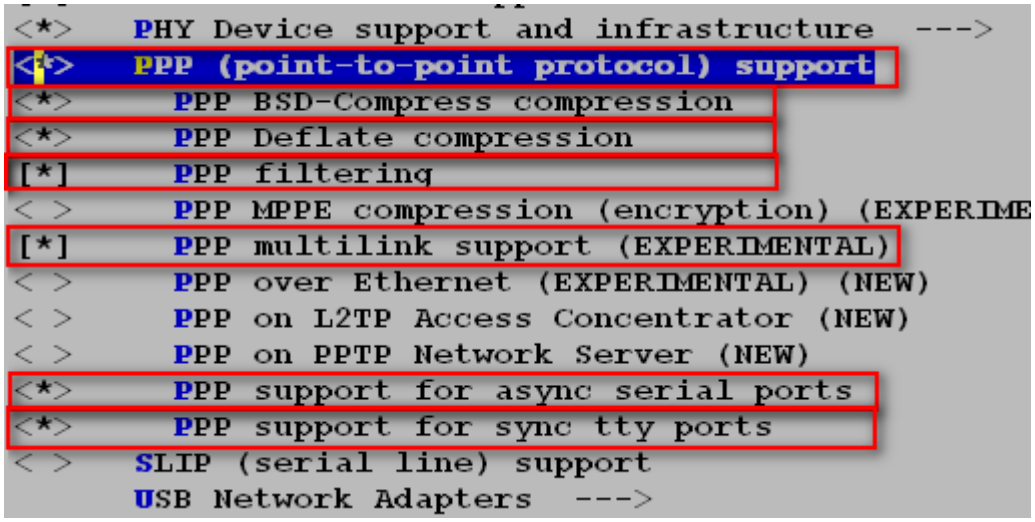
图 4-2 USB 转串口驱动选项

图 4-3 modem 驱动选项



选择ppp协议，位于Device Drivers ---> Network device support ---> PPP (point-to-point protocol) support，如图4-4所示。

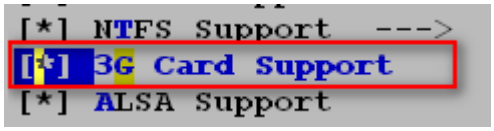
图 4-4 ppp 支持选项



步骤2 配置SDK以支持3G上网卡

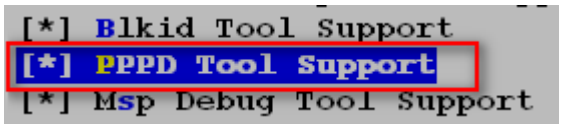
选择Component ---> 3G Card Support，编译3G组件，如图4-5所示。

图 4-5 3g 组件支持选项



选择Rootfs---> Board Tools Config ---> PPPD Tool Support，编译拔号程序，如图4-6所示。

图 4-6 pppd 工具支持选项



步骤3 编译内核和文件系统，烧写内核和文件系统。

步骤4 插入3G上网卡，就可以使用sample下的示例程序对3G上网卡进行相关的操作。具体操作请参见“1.3 操作示例1.3 操作示例”。

----结束

4.3 操作示例

在sample(SDK/sample/3g)下有3G上网卡的示例程序，演示3G上网卡的扫描、初始化及连接操作。详细请参考文档《HMS开发指南.pdf》中的“3G上网卡”章节。

4.4 操作中需要注意的问题

需要使用在兼容性列表中的上网卡，不在该列表中的器件未经过调试。

4.5 常见问题

为什么会出现扫描失败，未找到3G上网卡的现象？

问题描述

已经插入3G上网卡，但是程序执行时提示未扫描到设备。

问题分析

如果确定设备正常接入，可能是该设备不在支持列表中。

解决方法

将该设备型号提供给FAE，将其添加到兼容性列表中。

5 PCIe 操作指南

5.1 操作准备

硬件环境：PCIe WiFi模块

5.2 操作过程

操作过程如下：

1. 内核配置选项中勾选PCIe驱动。选择Bus support ---> PCI support，PCI Express Port Bus support，PCI host controller drivers->Hisilicon PCIe controller，将PCIe驱动编译进内核。
2. 编译内核，重新烧写内核。
3. 插入PCIe WiFi模块，系统启动后，加载WiFi驱动，使用ifconfig －a命令可以看到wlan0设备。

----结束

5.3 操作中需要注意的问题

需要使用在兼容性列表中的PCIe WiFi模块。

6 SATA 操作指南

6.1 操作准备

硬件环境：SATA硬盘

6.2 操作过程

操作过程如下：

1. 内核配置选项中勾选SATA驱动。选择Device Drivers ---> PHY Subsystem --->PHY Core ， Serial ATA and Parallel ATA drivers ---> <M> Platform AHCI SATA support, Serial ATA and Parallel ATA drivers ---> [*]AHCI SATA support。
2. 编译内核，重新烧写内核。
3. 插入SATA硬盘，系统启动后，加载SATA驱动libahci.ko、libahci_platform.ko和ahci_platform.ko，系统会识别到sda设备。

----结束

7 附录

7.1 用 fdisk 工具分区

说明

如无特殊说明，以下命令均为linux环境下的执行命令。如果为android环境，需要在命令前加”busybox”；

通过7.1.1 查看当前状态7.1.1 查看当前状态，对应以下情况选择操作：

- 若已有分区，本操作可以跳过，直接到“7.2 格式化设备7.2 格式化设备设备”。
- 若没有分区，则在控制台的提示符下，输入命令fdisk，具体格式如下：

```
~ $ fdisk 设备节点
```

回车后，输入命令m，根据帮助信息继续进行以下的操作。

其中设备节点与实际接入的设备类型有关，具体名称在以上各章节的“操作示例”中均有说明。

7.1.1 查看当前状态

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
```

控制台显示出分区状态信息：

```
Disk /dev/mmc/blk1/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
```

上面信息表明设备没有分区，需要按照7.1.2 创建新的分区7.1.2 创建新的分区和7.1.3 保存分区信息7.1.3 保存分区信息的描述对设备进行分区。

7.1.2 创建新的分区

创建新的分区步骤如下：

步骤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)
```

步骤2 建立主分区。

输入命令p，选择主分区：

```
P
如果需要建立扩展分区，输入：
e
```

步骤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):
```

步骤4 选择起始柱面。

本例选择默认值1，直接回车：

```
Using default value 1
```

步骤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
```

步骤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
```

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)
```

步骤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。

----结束

7.1.3 保存分区信息

输入命令w，写入并保存分区信息到设备：

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

控制台显示出当前设备信息，表示成功写入分区信息到设备：

```
The partition table has been altered!
Calling ioctl() to re-read partition table.
.....
~ $
```

7.2 格式化设备

- 若已格式化，本操作可以跳过，直接到“7.3 挂载目录7.3 挂载目录”。
- 若没有格式化，则可使用命令mkdosfs格式化FAT格式设备：

  ```
  ~ $ mkdosfs  -F 32 设备分区名
  ```

  其中设备分区名与实际接入的设备类型有关，具体名称在以上各章节的“操作示例”中均有说明。

  控制台显示出如下提示信息，表示成功格式化：

  ```
  mkdosfs 2.11 (12 Mar 2005)
  ~ $
  ```

- 格式化Hifat格式的设备

  同样执行标准的fat格式化工具：

  ```
  ~ $ mkdosfs  -F 32 设备分区名
  ```

- 格式化NTFS格式设备

  ```
  ~ $ mkntfs   设备分区名
  ```

7.3 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：

- 挂载vfat格式的文件系统

  ```
  ~ $ mount -t vfat 设备分区名 /mnt
  ```

- 挂载hifat格式的文件系统

  ```
  ~ $ himount 设备分区名 /mnt
  ```

- 挂载NFS格式的文件分区

  需要主机上开通NFS server的服务：

  ```
  mount -t nfs -o nolock xxx.xxx.xxx.xxx:/xx/xx /mnt
  xxx.xxx.xxx.xxx为NFS服务器的IP地址，/xx/xx为NFS服务器的共享目录
  ```

- 挂载jffs格式的文件分区

  ```
  ~ $ mount -t jffs2 设备分区名 /mnt
  ```

- 挂载yaffs2格式的文件分区

  ```
  ~ $ mount -t yaffs2 设备分区名 /mnt
  ```

- 挂载ntfs格式的文件分区

  ```
  ~ $ mount -t tntfs 设备分区名 /mnt
  ```

其中：设备分区名与实际接入的设备类型有关，具体名称在以上各章节的“操作示例”中均有说明。

7.4 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，即拷贝至设备，实现写操作，如：

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

也可以使用DD命令测试读写速度，该命令使用参数如下：

dd if=源文件，of=目标文件，bs=每一次读写大小，count=总共读写次数；其中bs*count表示读/写的总大小，单位为Byte。

测试写速度举例：

```
~ $dd if=/dev/zero of=/mnt/test_file bs=1024 count=1024
```

测试读速度举例：

```
~ $dd if=/mnt/test_file of=/dev/null bs=1024 count=1024
```

7.5 内核中模块与其内核选项的说明

对于内核模块，可以通过设置内核选项，来选择是否将其编入内核、编译成模块或者不编译。以下以ahci.ko为例说明如何根据模块名称查找其对应的内核选项配置名称并配置其编译进内核、编译成模块或者不编译。其它模块所对应的内核选项的查找方法类似。

要查找模块ahci.ko所对应的内核选项，请执行以下步骤：

步骤1 在超级终端上执行shell命令：grep -Rn “ahci.o” *。

此时屏幕显示以上信息：

```
./drivers/ata/Makefile:4:obj-$(CONFIG_SATA_AHCI)            += ahci.o
```

说明

模块的编译规则在Makefile文件中定义，而Makefile文件中的编译规则就是由模块相关的内核配置选项决定，模块相关的内核选项都是在对应的Kconfig文件中定义，因此通过grep命令可以查找到某一模块对应的内核选项的名称。例如ahci.ko对应的Kconfig文件中的config项名称就是SATA_AHCI。

步骤2 进入内核选项make menuconfig。

步骤3 在键盘上按下“/”按键，进入搜索界面。

步骤4 输入内核选项名称SATA_AHCI 。

步骤5 回车确认。

----结束

会打印内核所有名称中包含“SATA_AHCI”的选项及相关说明。如图7-1所示。其中“Location”就是其内核选项的具体位置。

图 7-1 SATA_AHCI 内核选项

```
Search Results
Symbol: SATA_AHCI [=n]
Prompt: AHCI SATA support
  Defined at drivers/ata/Kconfig:50
  Depends on: ATA
  Location:
    -> Device Drivers
      -> Serial ATA (prod) and Parallel ATA (experimental) drivers (ATA [=n])
```