



外设 使用指南

文档版本 03

发布日期 2015-12-16

版权所有 © 深圳市海思半导体有限公司 2015。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、海思和其他海思商标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

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。

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总部 邮编：518129

网址： <http://www.hisilicon.com>

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

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



章节号	章节名称	作者信息
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。



目 录

前 言.....	iii
1 SD/MMC 卡操作指南.....	1-1
1.1 操作准备.....	1-1
1.2 操作过程.....	1-1
1.3 操作示例.....	1-2
1.4 操作中需要注意的问题.....	1-4
2 ETH 操作指南.....	2-1
2.1 Linux 环境下的基本操作.....	2-1
2.2 其他操作示例.....	2-1
2.2.1 Route 命令.....	2-1
2.2.2 设置 DNS.....	2-3
2.2.3 配置转发功能.....	2-3
3 USB 操作指南.....	3-1
3.1 操作准备.....	3-1
3.2 操作过程.....	3-1
3.3 操作示例.....	3-2
3.3.1 U 盘操作示例.....	3-2
3.3.2 键盘操作示例.....	3-3
3.3.3 鼠标操作示例.....	3-3
3.3.4 USB-WiFi 操作示例.....	3-3
3.3.5 USB 串口操作示例.....	3-3
3.3.6 USB 网卡操作示例.....	3-5
3.4 操作中需要注意的问题.....	3-6
4 3G 上网卡.....	4-1
4.1 操作准备.....	4-1
4.2 操作过程.....	4-1
4.3 操作示例.....	4-3
4.4 操作中需要注意的问题.....	4-3
4.5 常见问题.....	4-3



5 PCIe 操作指南.....	5-1
5.1 操作准备.....	5-1
5.2 操作过程.....	5-1
5.3 操作中需要注意的问题.....	5-1
6 SATA 操作指南	6-1
6.1 操作准备.....	6-1
6.2 操作过程.....	6-1
7 附录.....	7-1
7.1 用 fdisk 工具分区.....	7-1
7.1.1 查看当前状态.....	7-1
7.1.2 创建新的分区.....	7-1
7.1.3 保存分区信息.....	7-3
7.2 格式化设备.....	7-3
7.3 挂载目录.....	7-3
7.4 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。读写文件.....	7-4
7.5 内核中模块与其内核选项的说明	7-4



插图目录

图 1-1 MMC/SD/SDIO 启动支持选项	1-1
图 1-2 MMC/SD/SDIO 驱动配置选项	1-2
图 1-3 在控制台下实现读写 SD 卡的操作示例	1-3
图 3-1 打开 USB 网卡的配置	3-5
图 4-1 硬件连接	4-1
图 4-2 USB 转串口驱动选项	4-2
图 4-3 modem 驱动选项	4-2
图 4-4 ppp 支持选项	4-2
图 4-5 3g 组件支持选项	4-2
图 4-6 pppd 工具支持选项	4-3
图 7-1 SATA_AHCI 内核选项	7-5



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：HiSTBLinuxV100R005 的 SDK 发布包。

1.2 操作过程

操作过程如下：

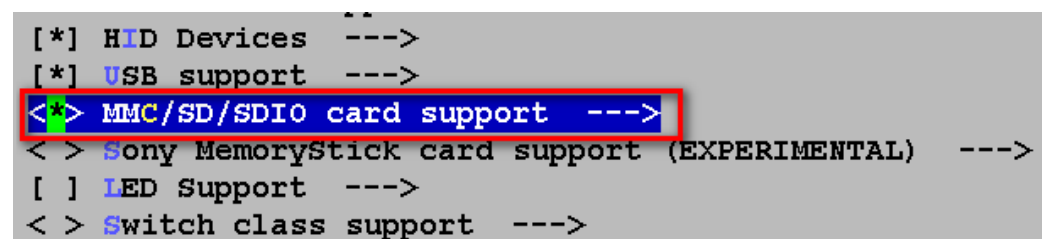
步骤 1 启动单板，加载根文件系统。

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 -> 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



```
[*] HID Devices --->
[*] USB support --->
<*> MMC/SD/SDIO card support --->
<> Sony MemoryStick card support (EXPERIMENTAL) --->
[ ] LED Support --->
<> Switch class support --->
```




图1-2 MMC/SD/SDIO 驱动配置选项

```

--- MMC/SD/SDIO card support
[ ]   MMC debugging
-*-   Assume MMC/SD cards are non-removable (DANGEROUS)
[ ]   MMC host clock gating (EXPERIMENTAL)
-*-   MMC embedded SDIO device support (EXPERIMENTAL)
[ ]   Enable paranoid SD card initialization (EXPERIMENTAL)
***   MMC/SD/SDIO Card Drivers ***
-*-   MMC block device driver
(8)   Number of minors per block device
-*-   Use bounce buffer for simple hosts
[ ]   Deferr MMC layer resume until I/O is requested
< >   SDIO UART/GPS class support
< >   MMC host test driver
***   MMC/SD/SDIO Host Controller Drivers ***
< >   ARM AMBA Multimedia Card Interface support
< >   Secure Digital Host Controller Interface support
< >   Marvell MMP2 SD Host Controller support (PXAV3)
< >   Marvell PXA9XX SD Host Controller support (PXAV2)
< >   Synopsys DesignWare Memory Card Interface
< >   VUB300 USB to SDIO/SD/MMC Host Controller support
< >   USB SD Host Controller (USHC) support
<*>  himci v200 sdio/mmc device support --->

```

步骤3 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”。



说明

芯片支持两个 SDIO 控制器，通常 SDIO0 用于 SD 卡，SDIO1 用于 eMMC 器件，如果用户设计与海思参考设计不一致，请联系 FAE 支持。

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，请参见附录 7.5 的说明。

----结束

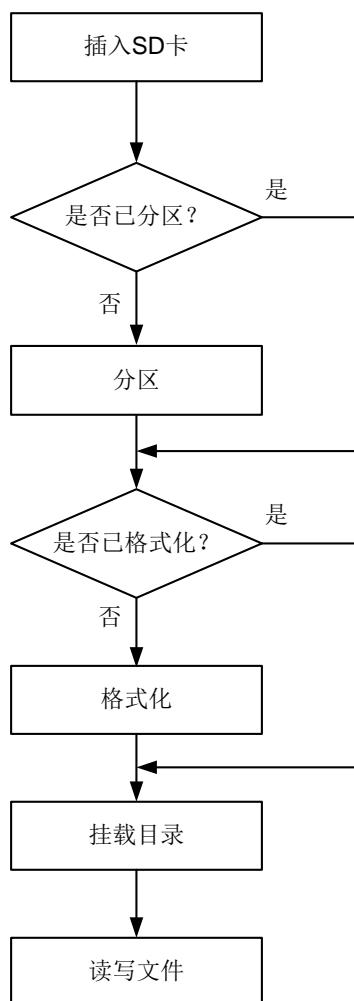
1.3 操作示例

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

其中 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/mmcblk0
- 用 mkdosfs 工具格式化的具体目录需改为：~\$ mkdosfs -F 32 /dev/mmcblk0pX
- 挂载的具体目录需改为：~\$ mount -t vfat /dev/mmcblk0pX /mnt

步骤 1 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”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”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”。

步骤 4 对 SD/MMC 卡进行读写操作，请参见“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）。
- 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异常，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
PING 10.110.10.1 (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 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”。



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，请参见附录 7.5 的说明。

----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。

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

其中 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/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：~\$ chknfs -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](#)”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](#)”进行格式化后，进入[1.3 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](#)”。

步骤 4 对硬盘进行读写操作，请参见“[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
```

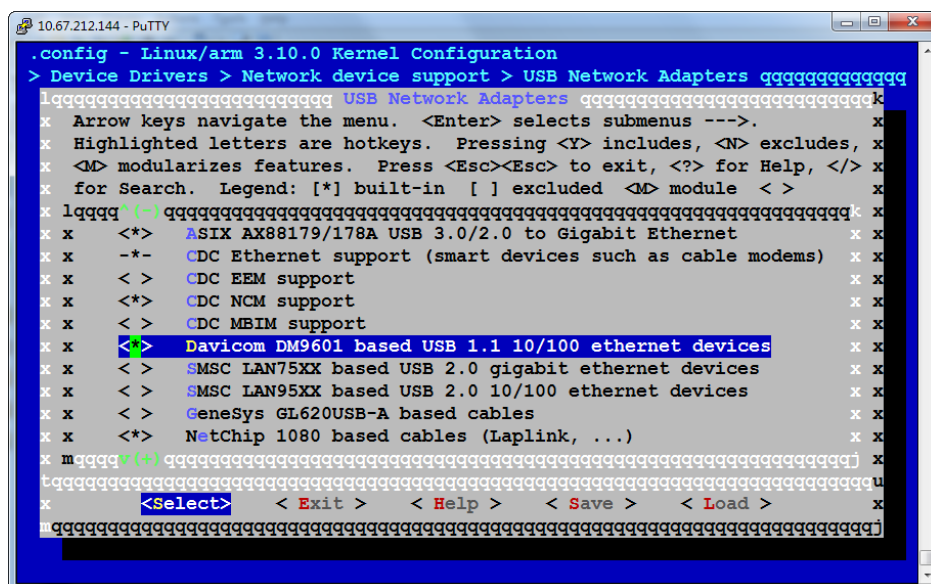
- 重启单板。

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;make linux_install`
- 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

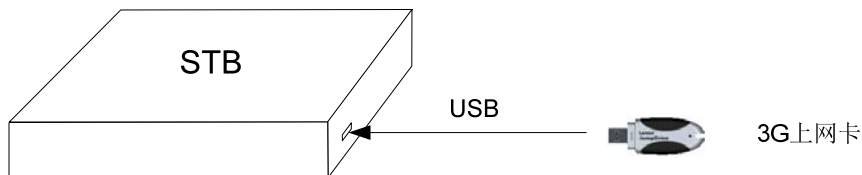
4.1 操作准备

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：HiSTBLinuxV100R005 的 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。

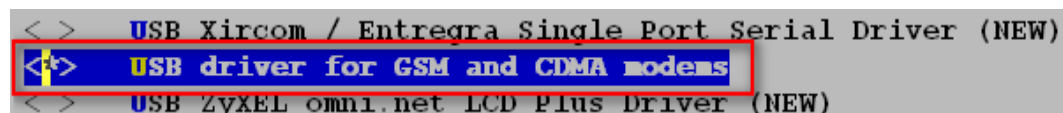
选择 usb 转串口驱动 “USB Serial Converter support”，同时选择 modem 串口驱动 “USB driver for GSM and CDMA modems”，如[图 4-2](#)、[图 4-3](#) 所示。



图4-2 USB 转串口驱动选项

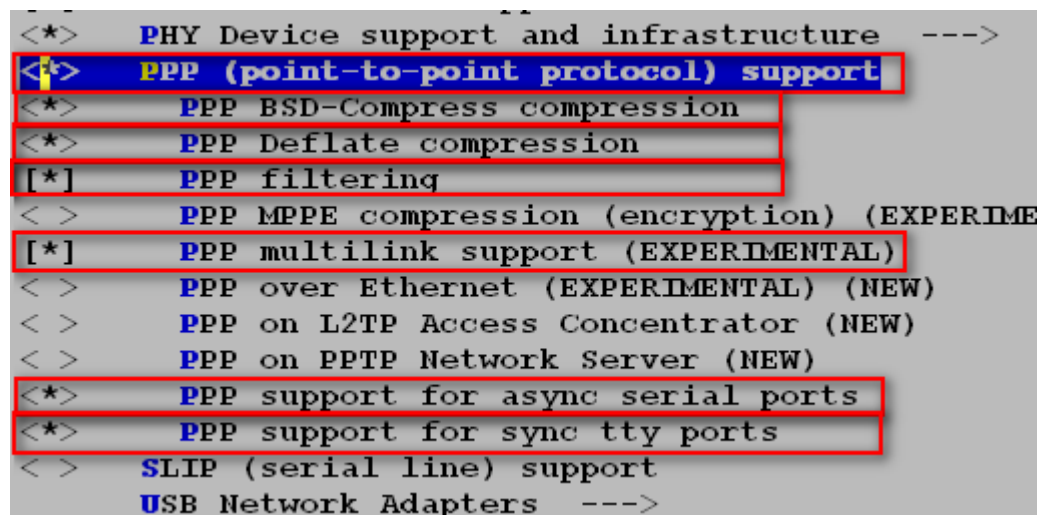


图4-3 modem 驱动选项



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

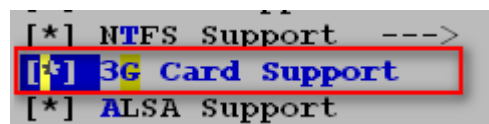
图4-4 ppp 支持选项



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。

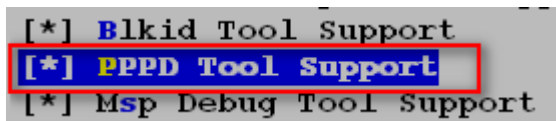
图4-5 3g 组件支持选项



选择 Rootfs---> Board Tools Config ---> PPPD Tool Support，编译拨号程序，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，炼写内核和文件系统。

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”。

----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：

- 步骤 1 内核配置选项中勾选 PCIe 驱动。选择 Bus support ---> PCI support, PCI Express Port Bus support, Hisilicon PCIe controller, 将 PCIe 驱动编译进内核。
- 步骤 2 编译内核，重新烧写内核。
- 步骤 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：

步骤 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。

步骤 2 编译内核，重新烧写内核。

步骤 3 插入 SATA 硬盘，系统启动后，加载 SATA 驱动 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

通过 [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](#)”。
- 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](#)和 [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

格式化 fat 格式的设备

- 若已格式化，本操作可以跳过，直接到“[7.3 挂载目录](#)”。
- 若没有格式化，则输入命令 mkdosfs 进行格式化：

```
~ $ 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
```

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 ntfs 设备分区名 /mnt
```

7.4 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。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。

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])
```

----结束