



Peripheral
User Guide

Issue	03
Date	2015-12-29

Copyright © HiSilicon Technologies Co., Ltd. 2015. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of HiSilicon Technologies Co., Ltd.

Trademarks and Permissions



HISILICON, and other HiSilicon icons are trademarks of HiSilicon Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between HiSilicon and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

HiSilicon Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: <http://www.hisilicon.com>

Email: support@hisilicon.com



About This Document

Purpose

This document describes how to manage the peripherals connected to the modules configured with the serial advanced technology attachment (SATA), peripheral component interconnect express (PCIe), secure digital input/output (SDIO), Ethernet (ETH), or universal serial bus (USB) 2.0/3.0 Host driver program. It covers the preparations, operation procedures, precautions to be taken during operation, and operation instances.

Related Versions

The following table lists the product versions related to this document.

Product Name	Version
Hi3798C	V1XX
Hi3798C	V2XX
Hi3796C	V1XX
Hi3798M	V1XX
Hi3796M	V1XX

Intended Audience

This document is intended for:

- Technical support personnel
- Software development engineers

Change History

Changes between document issues are cumulative. Therefore, the latest document issue contains all changes made in previous issues.



Issue 03 (2015-12-29)

This issue is the third official release, which incorporates the following change:

This document is updated to support the Hi3798C V200 solution.

Issue 02 (2015-05-26)

This issue is the second official release, which incorporates the following changes:

Hi3798C V200 is supported.

Chapter 5 "Operation Guide to the PCIe" and chapter 6 "Operation Guide to the SATA" are added.

Issue 01 (2014-10-30)

This issue is the first official release, which incorporates the following change:

Hi3796M V100 is supported.

Issue 00B01 (2014-06-20)

This issue is the first draft release.



Contents

About This Document.....	i
1 Operation Guide to the SD/MMC Driver	1
1.1 Preparations.....	1
1.2 Procedure	1
1.3 Operation Instance	2
1.4 Precautions	4
2 Operation Guide to the ETH Driver.....	6
2.1 Basic Operations in Linux.....	6
2.2 Other Instances.....	6
2.2.1 Route Related Commands.....	7
2.2.2 Setting the DNS	9
2.2.3 Setting the Forwarding Function.....	9
3 Operation Guide to the USB 2.0 Driver.....	10
3.1 Preparations.....	10
3.2 Procedure	10
3.3 Operation Instances	11
3.3.1 Using the USB Flash Drive.....	11
3.3.2 Using the Keyboard	12
3.3.3 Using the Mouse	12
3.3.4 Using the USB Wi-Fi Device.....	12
3.3.5 Using the USB Serial Port	12
3.3.6 Using the USB Network Adapter.....	14
3.4 Precautions	15
4 Operation Guide to the 3G Card.....	16
4.1 Preparations.....	16
4.2 Procedure	16
4.3 Operation Instance	18
4.4 Precautions	18
4.5 Common Issue.....	18
5 Operation Guide to the PCIe	19
5.1 Preparations.....	19



5.2 Procedure	19
5.3 Precautions	19
6 Operation Guide to the SATA.....	20
6.1 Preparations	20
6.2 Procedure	20
7 Appendix	21
7.1 Partitioning a Storage Device by Using the fdisk Tool	21
7.1.1 Checking the Current Partition Status of a Storage Device.....	21
7.1.2 Creating Partitions for a Storage Device.....	21
7.1.3 Saving the Partition Information	23
7.2 Formatting a Device	23
7.3 Mounting a Directory	23
7.4 Reading/Writing to a File	24
7.5 Mapping Between a Kernel Driver and Its Kernel Option	24



Figures

Figure 1-1 Option for supporting the booting of MMC/SD/SDIO	1
Figure 1-2 MMC/SD/SDIO controller driver configuration option	2
Figure 1-3 Read/write flowchart	3
Figure 3-1 Enabling the USB network adapter.....	14
Figure 4-1 Hardware connection	16
Figure 4-2 Selecting USB Serial Converter support.....	16
Figure 4-3 Selecting USB driver for GSM and CDMA modems	17
Figure 4-4 Selecting PPP (point-to-point protocol) support.....	17
Figure 4-5 3G card support.....	17
Figure 4-6 PPPD tool support.....	17
Figure 7-1 Position of the kernel option CONFIG_SATA_AHCI.....	25



1 Operation Guide to the SD/MMC Driver

1.1 Preparations

Hardware: SD card/MMC

Software: HiSTBLinux V100R005 SDK

1.2 Procedure

To use the SD card or MMC, perform the following steps:

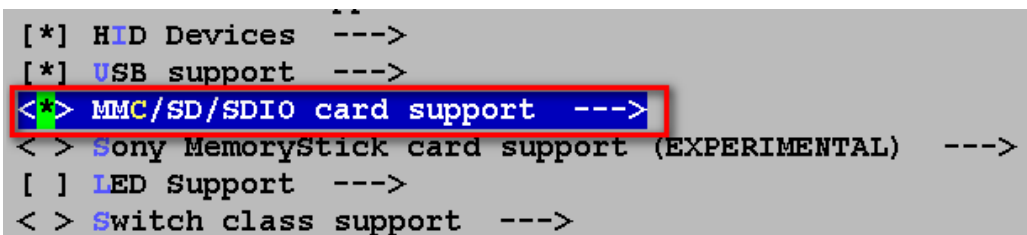
Step 1 Start the board and load the root file system.

Step 2 Load the SDIO drivers.

By default, all SDIO drivers are compiled in the kernel. Therefore, you do not need to load the SDIO drivers.

Choose **Device Drivers > MMC/SD/SDIO card support** to configure the kernel option for supporting the MMC, SD, or SDIO card. See the options in the red rectangles in [Figure 1-1](#) and [Figure 1-2](#).

Figure 1-1 Option for supporting the booting of MMC/SD/SDIO



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




Figure 1-2 MMC/SD/SDIO controller driver configuration option

```
--- 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 --->
```

Step 3 Insert the SD card or MMC. Then you can perform operations on the SD card or MMC. For details, see section 1.3 "Operation Instance."



NOTE

Two SDIO controllers are supported. Generally, SDIO0 applies to the SD card, and SDIO1 applies to the eMMC. When the user design is inconsistent with the HiSilicon reference design, contact field application engineers (FAEs).

If drivers are modified during debugging, the kernel must be compiled again. For details about the mapping between a driver and its kernel option, see chapter 7 "Appendix."

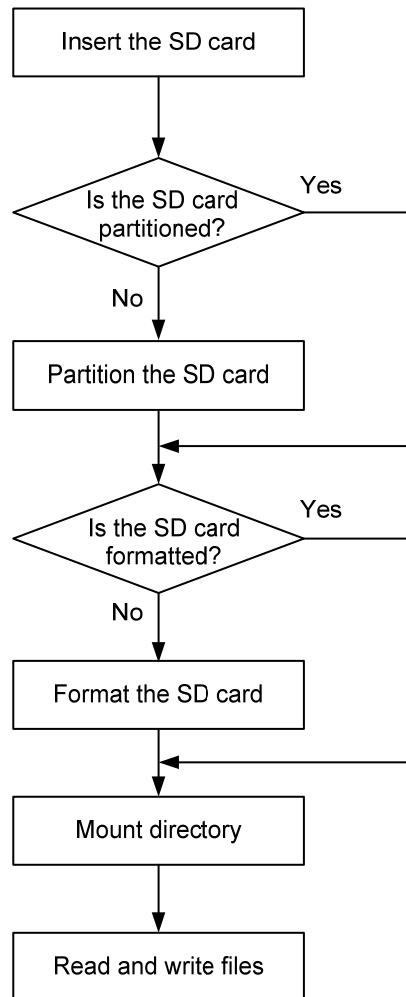
----End

1.3 Operation Instance

The following instance describes how to read/write to an SD card by using the SDIO interface. The procedure for reading/writing to an MMC is similar to that of an SD card and therefore not described in this document.

Flowchart

Figure 1-3 shows the read/write flowchart.

Figure 1-3 Read/write flowchart

Initialization

Perform the following initialization steps after the related drivers are loaded:

NOTE

In the commands mentioned below, X indicates the partition number that is assigned when you partition the SD card or MMC by running the **fdisk** command.

- Change the directory in which the **fdisk** command is executed to `~ $ fdisk /dev/mmcblk0`.
- Change the directory in which the **mkdosfs** command is executed to `~ $ mkdosfs -F 32 /dev/mmcblk0pX`.
- Change the mount directory to `~ $ mount -t vfat /dev/mmcblk0pX /mnt`.

Step 1 Check whether the SD card or MMC is partitioned.

- If **p1** is not displayed, the SD card or MMC is not partitioned. Partition it by running the `~ $ fdisk /dev/mmcblk0` command. For details, see section [7.1 "Partitioning a Storage Device by Using the fdisk Tool."](#)
- If **p1** is displayed, the SD card or MMC is detected and partitioned. Go to [Step 2](#).

Step 2 Check whether the SD card or MMC is formatted.



- If no, format the SD card or MMC by running the ~\$ **mkdosfs -F 32 /dev/mmcblk0pX** command. For details, see section 7.2 "Formatting a Device."
- If yes, go to [Step 3](#).

Step 3 Mount a directory. For details, see section 7.3 "Mounting a Directory."

Step 4 Read/write to the SD card or MMC. For details, see section 7.4 "Reading/Writing to a File."

----End

1.4 Precautions

Take the following precautions during operation:

- The SD card has a write protection switch on the side of the SD card. If you want to write to the SD card, you need to set the write protection switch to the invalid mode to disable write protection. The SD card can be written only after it is mounted; otherwise, the SD card can only be read.
- Ensure that the metal sheet of the card is in good contact with the card slot; otherwise, the card fails to be detected or an error occurs in reading/writing to the card. When testing a thin MMC, hold the communication end of the slot if necessary.
- Before reading or writing to an SD card, ensure that the SD card is partitioned by running the **fdisk** command and formatted to the VFAT format by running the **mkdosfs** command. For details, see section 1.3 "Operation Instance."
- You need to run the **mount** command to mount an SD card to a file system so that you can read/write to the SD card. If an SD card has been mounted to a file system, after reading and writing to the file, you must run the **umount** command first, and then remove the card. Otherwise, an exception may occur.
- You need to run the **umount** command to release the mount node before removing an SD card. If you remove an SD card directly, data stored in the card may be lost and the card may fail to be mounted next time. Therefore, you still need to run the **umount** command to release the mount node even if an SD card is removed directly.
- You cannot perform the following operations:
 - If the current working directory is the mounted directory, you cannot run the **umount** directly. You need to go to another directory and run the **umount** command to release the mount node.
 - Do not remove an SD card during read/write. Otherwise, exception information is displayed, and the files in the SD card or the file system may be damaged.
 - If the mounted directory read/write process is still running, you cannot run the **umount** command. You can run the **umount** command after mounting the directory.

Take the following measures if exceptions occur during operation:

- If a card is removed incorrectly during a cyclic test, press **Ctrl+C** to roll back to the shell; otherwise, misoperation information is reported repeatedly.
- If you reseat a card rapidly, the card may fail to be detected. This is because the registration and deregistration processes take a period of time.
- If a card is removed incorrectly, run the **umount** command and run the **mount** command. Otherwise, the mount node directory, **/mnt** for example, cannot be read or written and misoperation information is displayed.



- If an SD card has multiple partitions, you can run the **mount** command to switch to another partition for mounting. All mounted partitions are released only if the number of times of running the **umount** command equals that of running the **mount** command.
- If the file system is damaged due to read/write operations, exceptions may occur after you reinsert the card, mount a folder, and then read/write to the card. In this case, run the **umount** command, reseal the SD card, and then run the **mount** command.



2 Operation Guide to the ETH Driver

2.1 Basic Operations in Linux

Start a terminal with serial ports, connect your PC to the target device using a serial cable. If the target device has multiple ETH ports, correctly connect the related ETH port. The following takes eth0 as an example to describes how to use the ETH port in Linux:



NOTE

The driver of the ETH port is compiled in the kernel and therefore does not need to be loaded. If the board has two ETH ports, consult HiSilicon hardware engineers about the ETH port corresponding to eth0.

- Set the MAC address for the ETH port by running the following shell command on the HyperTerminal:

```
ifconfig eth0 hw ether XX:XX:XX:XX:XX:XX
```

where **eth0** is the name of the ETH port, and XX:XX:XX:XX:XX:XX is the MAC address.

- Set the IP address for the ETH port by running the following shell command on the HyperTerminal:

```
ifconfig eth0 XXX.XXX.XXX.XXX
```

where **eth0** is the name of the ETH port, and XXX.XXX.XXX.XXX is the IP address.

- Set the subnet mask of the ETH port by running the following shell command on the HyperTerminal:

```
ifconfig eth0 netmask XXX.XXX.XXX.XXX
```

where **eth0** is the name of the ETH port, and XXX.XXX.XXX.XXX is the subnet mask.

2.2 Other Instances

In addition to the preceding functions, the ETH port provides other functions in Linux.



2.2.1 Route Related Commands

Deleting a Route

- Command

To delete a route, run the following command:

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

Where, **10.85.180.0** is the target network segment, **255.255.254.0** is the subnet mask of the target network segment and **dev eth0** is the name of the network port that is used by the target network segment for communication.

- Example

- Assume that there is a route table with the IP address 10.85.180.0 and subnet mask 255.255.254.0. In addition, the route table uses the eth0 port for communication. To delete this route table, run the following commands:

```
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

- To delete the default route, run the following commands:

```
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

Adding a Route

Before adding a route, ensure that there is a physical link between the network port and the gateway and that the link works properly.

- Command

To add a route, run the following command:

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

Where, **10.85.180.0** is the target network segment, **255.255.254.0** is the subnet mask, and **eth0** is the name of the network port that is used by the target network segment for communication.



- Example

- Assume that there is a route table with the IP address 10.85.180.0 and subnet mask 255.255.254.0. In addition, the route table uses the eth0 port for communication. To add the route table, run the following commands:

```
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 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

```
~ $
```

- Add a default route.

To enable the chip to communicate with the subnet without routing tables, set a default route by running the following command:

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

The preceding command indicates that information is transmitted from the gateway 10.85.180.1 through the eth0 port when the chip communications with the subnet without routing tables.

```
~ $ 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
```

For details, see the *Linux Man Pages Man Route* on a PC that runs in Linux.



2.2.2 Setting the DNS

To set the domain name server (DNS), edit the **resolv.conf** file in **/etc**, and then add the following statement:

```
nameserver dnsip
```

Where, **nameserver** is the keyword, and **dnsip** is the IP address for the DNS.

After adding the statement, you can run the **ping** www.xxxx.com command to check whether the DNS is configured successfully.

2.2.3 Setting the Forwarding Function



CAUTION

During information forwarding in standby mode, ensure that the MAC modules and ETH port physical (PHY) module are supplied with power properly and that related clocks work properly.

For details, see the procedure for setting the forwarding function for the UNF interface in standby mode.



3

Operation Guide to the USB 2.0 Driver

3.1 Preparations

Hardware: USB flash drive, USB mouse, USB keyboard, and USB Wi-Fi module.

3.2 Procedure

To use the USB flash drive, perform the following steps:

- Step 1** To compile USB_OHCI_HCD_PLATFORM, USB_EHCI_HCD_PLATFORM and USB_XHCI_HISILICON in the kernel configuration options as modules, choose **Device Drivers > USB support > Generic EHCI driver for a platform device, xHCI support for Hisilicon SoCs, and Generic OHCI driver for a platform device.**
- If you set to **[M]**, manually load USB_OHCI_HCD_PLATFORM, USB_EHCI_HCD_PLATFORM, and USB_XHCI_HISILICON in **drivers/usb/host** of the kernel source code directory.
 - If you set to **[*]**, USB_OHCI_HCD_PLATFORM, USB_EHCI_HCD_PLATFORM, and USB_XHCI_HISILICON are compiled into the kernel. In this case, you do not need to manually load related drivers, because the kernel automatically loads the drivers after it starts.

Run the following commands for loading the USB driver module:

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

- Step 2** Access the file system, insert a USB device such as the USB flash drive, mouse, or keyboard, and then perform operations on the USB device. For details, see section 3.3 "[Operation Instances](#)."



NOTE

If drivers are modified during debugging, the kernel must be compiled again. For details about the mapping between a driver and its kernel option, see section 7.5 "[Mapping Between a Kernel Driver and Its Kernel Option](#)."

----End



3.3 Operation Instances

3.3.1 Using the USB Flash Drive

Inserting and Detecting a USB Flash Drive

Insert a USB flash drive, and then check whether it can be detected.

If the USB flash drive is detected, the following information is displayed:

```
~ $ 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
```

This instance assumes that the USB flash drive has only one partition **sda1**; therefore, only **sda1** is displayed. If there are multiple partitions, texts such as **sda1**, **sda2**, and **sda3** are displayed. If the USB flash drive is not partitioned, only the text **sda** but not **sda1** or **sda2** is displayed.

Using the USB Flash Drive

After the USB flash drive is inserted properly, perform the following steps:



NOTE

In the commands mentioned below, X indicates the partition number that is assigned when you partition a USB flash drive by running the **fdisk** command.

- Change the directory in which the **fdisk** command is executed to **~ \$ fdisk /dev/sda**.
- Change the directory in which the **mkdosfs** command is executed to **~ \$ mkdosfs -F 32 /dev/sdaX**.
- Change the mount directory to **~ \$ mount -t vfat /dev/sdaX /mnt**.
- Format the USB flash drive into FAT32, NTFS, EXT3, or EXT4 based on the specific application scenarios.
 - To format the USB flash drive into NTFS, run the **~ \$ mkntfs /dev/sdaX** command. To restore the NTFS file system, run the **~ \$ chknfs -a -f /dev/sdaX** command.
 - For details about how to format the USB flash drive into EXT3 or EXT4, see the usage of EXT3 or EXT4 for the eMMC. Details are not provided in this document. By default, the USB flash drive is formatted into FAT32.

Step 1 Check whether the USB flash drive is partitioned.

- If **sda1** is not displayed, the USB flash drive is not partitioned. Partition it by running the **fdisk** command. For details, see section 7.1 "[Partitioning a Storage Device by Using the fdisk Tool](#)."
- If **sda1** is displayed, the USB flash drive is partitioned. Go to [Step 2](#).



Step 2 Check whether the USB flash drive is formatted.

- If no, format the USB flash drive. For details, see section 7.2 "Formatting a Device."
- If yes, go to [Step 3](#).

Step 3 Mount a directory. For details, see section 7.3 "Mounting a Directory."

Step 4 Read/write to the USB flash drive. For details, see section 7.4 "Reading/Writing to a File."

----End

3.3.2 Using the Keyboard

To use the keyboard, perform the following steps:

Step 1 Load the keyboard drivers.

After the keyboard drivers are loaded, the **event0** node is generated in **/dev/input/**.

Step 2 Receive the keyboard inputs by running the following command:

```
cat /dev/input/event0
```

This command is used to display the contents input through the USB keyboard of the target board on the terminal.

Step 3 Press any keys on the keyboard.

If no error occurs, the content that you entered is displayed on the screen.

----End

3.3.3 Using the Mouse

To use the mouse, perform the following steps:

Step 1 Load the mouse drivers.

After the mouse drivers are loaded, the **mouse0** node is generated in **/dev/input/**.

Step 2 Run a standard test program (**mev** recommended) of the **gpm** tool.

Step 3 Click randomly on the screen or move the pointer.

If the mouse functions properly, the corresponding code value is displayed.

----End

3.3.4 Using the USB Wi-Fi Device

For details about how to use the USB Wi-Fi device, see the *HMS Development Guide*.

3.3.5 Using the USB Serial Port

3.3.5.1 Adding the Driver for the USB-to-Serial Adapter in the Kernel

To add the driver for the USB-to-serial adapter in the kernel, perform the following steps:



Step 1 Go to the Linux kernel directory and configure the kernel by running the following commands (take Linux kernel 3.18.y as an example):

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

Step 2 Configure the kernel as follows:

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

**NOTE**

The kernel supports various USB-to-serial adapters. Prolific 2303 Single Port Serial is commonly used. If another USB-to-serial adapter is used, select the driver accordingly.

Step 3 Save the configuration and recompile the kernel image.

After the configuration, you can find a device similar to **ttyUSB0** (name of the USB-to-serial adapter) in the **/dev** directory. Then applications can operate the USB-to-serial adapter (for example, **ttyUSB0**) in the same way they operate common serial ports.

----End

3.3.5.2 Redirecting the Serial Port to the USB-to-Serial Adapter (ttyUSB0)

To output the kernel and file system information displayed when the system is started to ttyUSB0 and use ttyUSB0 as a system control port, perform the following steps:

Step 1 Change **console=ttyAMA0** to **console=ttyUSB0** in bootargs.

Step 2 Delete **::respawn:-/bin/sh** and add **::respawn:/sbin/getty ttyUSB0 115200 vt100** in the boot initialization script of the board file system.

Step 3 Save the modifications and restart the board. Connect the USB-to-serial cable to the serial port on the PC and restart the board. The system control end changes to the USB-to-serial device, and the information displayed when the kernel boots is output through the USB-to-serial device. You can also operate the board (for example, issue a **shell** command) by using the USB-to-serial device.

----End

3.3.5.3 Notes

- After the system console is redirected to the USB-to-serial adapter, the system displays a dialog box asking you to enter the user name and password when the system boots. You can follow the instructions.
- If you do not want to enter the password each time you log in to the system, you can modify **/etc/passwd** of the file system and left the corresponding user password blank. Take the user **root** as an example.

Before modification:



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

After modification

```
root::0:0:root:/root:/bin/sh
```

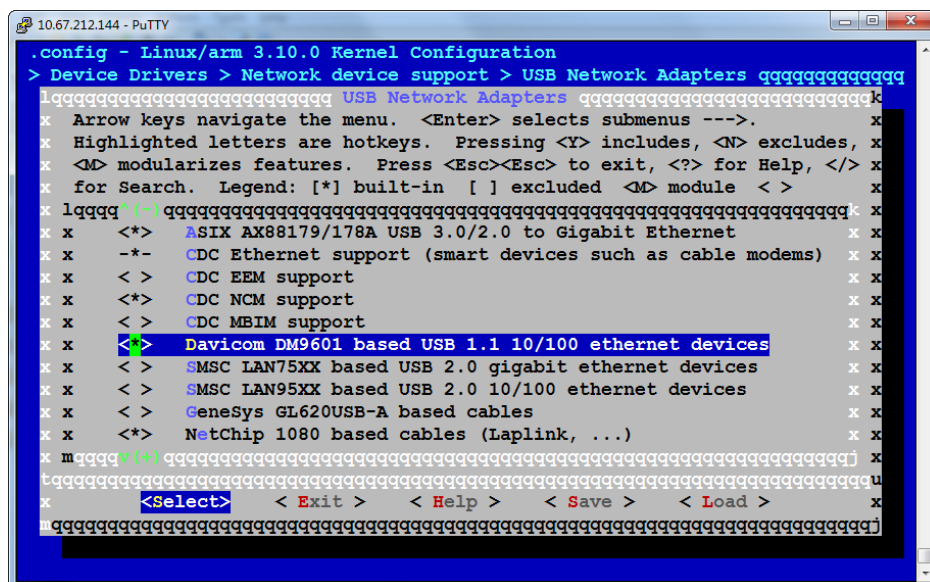
- Restart the board after modification.

3.3.6 Using the USB Network Adapter

To use the USB network adapter, perform the following steps (use the USB network adapter Mu Yang JP1081, chip model DM9601, on the Hi3798C V200 platform as an example):

- Step 1** Enable the USB network adapter driver under the kernel. Run `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-`.
- Step 2** Choose **Device Drivers > Network device support > USB Network Adapters > Multi-purpose USB Networking Framework**, and select **Davicom DM9601 based USB 1.1 10/100 ethernet devices**.

Figure 3-1 Enabling the USB network adapter



- Step 3** Save the configuration and exit, and then run `cp .config arch/arm/configs/hi3798CV200_defconfig`. Note that the configuration is saved to `hi3798CV200_defconfig` only after this command is executed.
- Step 4** Run `cd ../../..` to return to the root directory of the SDK, and then run `make linux;make linux_install`.
- Step 5** Re-burn the kernel. Power on the board, and insert the USB network adapter. An ethX device is generated. You can query the device or configuration its IP address by running the `ifconfig` command. Note that the ethX device is not enabled by default. You need to add the `-a` parameter before `ifconfig`.

----End



3.4 Precautions

Take the following precautions during operation:

- The descriptions about the operations related to USB devices are available at <http://www.usb.org/developers/compliance/>.
- You need to run the **mount** command, operate a file, and then run the **umount** command in sequence each time. This avoids exceptions in the file system.
- The drivers of the keyboard and mouse must work with the upper layer. For example, mouse events are displayed on the graphical user interface (GUI) of the upper layer. You only need to operate the keyboard by accessing the event node in **/dev**. In addition, standard libraries are required for mouse operations.
- Mouse application libgpm libraries are provided in Linux. If you need to use the mouse, these libraries must be compiled. You are advised to use the standard kernel interface gpm-1.20.5 that has passed the test.

In addition, a set of test programs (such as mev) are provided in the gpm tool. You can perform encoding by using the test programs, making the development easier.



4 Operation Guide to the 3G Card

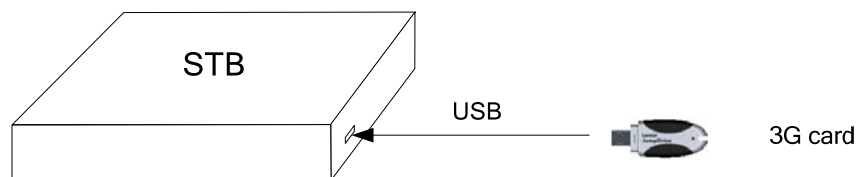
4.1 Preparations

Before using the 3G card, prepare the environment as follows:

- Hardware environment: 3G card
- Software environment: HiSTBLinux V100R005 SDK

Figure 4-1 shows the hardware connection. After the 3G card is successfully connected to the network, the STB can connect to the network through the 3G card.

Figure 4-1 Hardware connection



4.2 Procedure

To use the 3G card, perform the following steps:

Step 1 Configure the kernel to support the 3G card.

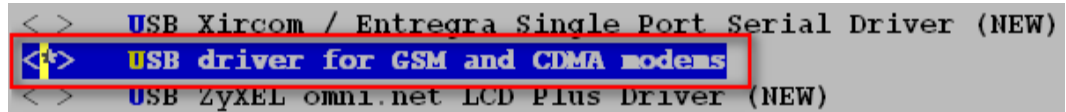
1. Run **make menuconfig** in the kernel directory.
2. Select **USB Serial Converter support** and **USB driver for GSM and CDMA modems**, as shown in Figure 4-2 and Figure 4-3.

Figure 4-2 Selecting USB Serial Converter support



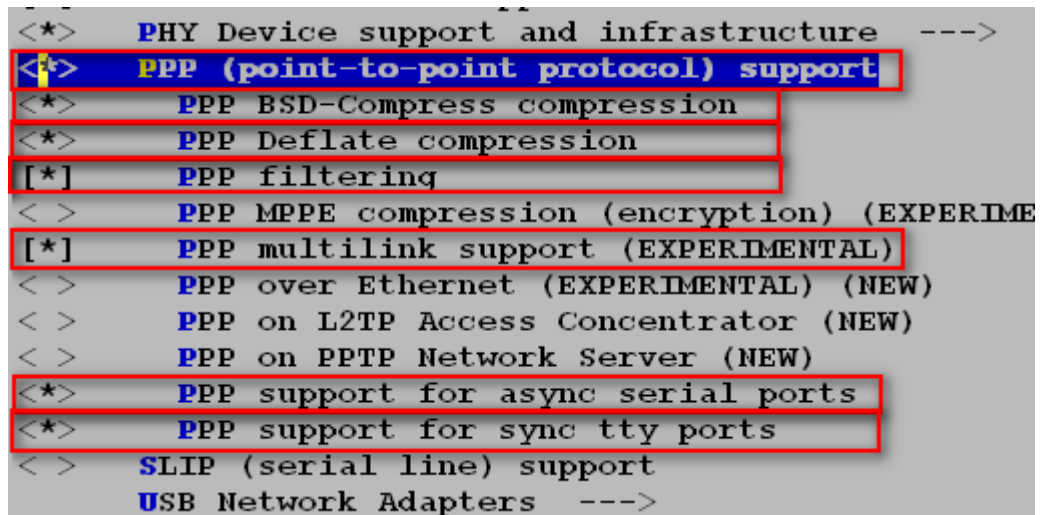


Figure 4-3 Selecting USB driver for GSM and CDMA modems



3. Choose **Device Drivers > Network device support**, and select **PPP (point-to-point protocol) support**, as shown in [Figure 4-4](#).

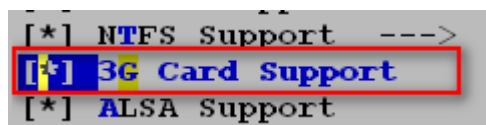
Figure 4-4 Selecting PPP (point-to-point protocol) support



Step 2 Configure the SDK to support the 3G card.

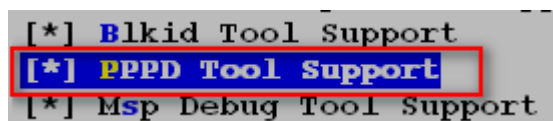
1. Choose **Component > 3G Card Support**, and compile the 3G component, as shown in [Figure 4-5](#).

Figure 4-5 3G card support



2. Choose **Rootfs > Board Tools Config > PPPD Tool Support**, and compile the dialing program, as shown in [Figure 4-6](#).

Figure 4-6 PPPD tool support



Step 3 Compile and burn the kernel and file system images.



- Step 4** Insert the 3G card, and perform related operations. For details, see section [1.3 "Operation Instance."](#)
- End

4.3 Operation Instance

The 3G card sample is in **SDK/sample/3g**, which demonstrates operations such as detecting, initializing, and connecting the 3G card. For details, see the "3G Card" chapter in the *HMS Development Guide*.

4.4 Precautions

You must use the 3G card in the compatibility list because the components that are not in the list are not debugged.

4.5 Common Issue

What do I do if the 3G card is inserted but cannot be detected?

Problem Description

The 3G card is inserted but cannot be detected.

Analysis

If the device is properly connected, this problem may occur because the device is not in the list of compatible devices.

Solution

Ask the FAEs to add the device model to the compatibility list.



5 Operation Guide to the PCIe

5.1 Preparations

Prepare a PCIe Wi-Fi module.

5.2 Procedure

To use the PCIe Wi-Fi module, perform the following steps:

- Step 1** Run **make menuconfig**. Choose **Bus support** and select **PCI support**, **PCI Express Port Bus support**, and **Hisilicon PCIe controller** to compile the PCIe driver into the kernel.
 - Step 2** Compile the kernel and reburn the kernel image.
 - Step 3** Insert a PCIe Wi-Fi module. After the system is booted, load the Wi-Fi driver. Run **ifconfig -a** to check the wlan0 device.
- End

5.3 Precautions

You must use the PCIe Wi-Fi module in the compatibility list.



6 Operation Guide to the SATA

6.1 Preparations

Prepare a SATA hard disk.

6.2 Procedure

To use the SATA hard disk, perform the following steps:

Step 1 Run **make menuconfig**. Choose **Device Drivers > PHY Subsystem --->PHY Core**. Select **Serial ATA and Parallel ATA drivers** and then **Platform AHCI SATA support**, Serial ATA and Parallel ATA drivers ---> [*]AHCI SATA support.

Step 2 Compile the kernel and reburn the kernel image.

Step 3 Insert a SATA hard disk. After the system is booted, load the SATA driver **libahci_platform.ko** and **ahci_platform.ko**. The system automatically identifies the sda device.

----End



7 Appendix

7.1 Partitioning a Storage Device by Using the fdisk Tool

Perform the following operations based on the current partition status:

- If partitions exist, skip these operations and go to section [7.2 "Formatting a Device."](#)
- If no partition exists, enter the **fdisk** command at the command prompt of the console:

```
~ $ fdisk device node
```

Then run the **m** command. Continue with the operations based on the help information.

The device node depends on the type of the actual storage device. For details, see the operation instances in preceding chapters.

7.1.1 Checking the Current Partition Status of a Storage Device

To check the current partition status of a storage device, enter the following command at the command prompt of the console:

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

If the following information is displayed on the console, the storage device is not partitioned:

```
Disk /dev/mmcblk1/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
```

In this case, you need to partition the device by following the description in section [7.1.3 "](#)and save the partition information by following the description in section [7.1.3 "Saving the Partition Information."](#)

7.1.2 Creating Partitions for a Storage Device

To create partitions for a storage device, perform the following steps:

Step 1 Create partitions by entering the following command at the command prompt:

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

The following information is displayed on the console:

```
Command action
```



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

Step 2 Create the primary partition by running the following command:

```
p
```

To create an extension partition, run the following command:

```
e
```

Step 3 Set the number of partitions by entering a number (for example, **1**).

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

The following information is displayed on the console:

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

Step 4 Select the start cylinder by entering a number, and then press **Enter**. In this example, the default value **1** is retained.

```
Using default value 1
```

Step 5 Select the end cylinder by entering a number, and then press **Enter**. In this example, the default value **970** is retained.

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

```
Using default value 970
```

Step 6 Select a data format for the storage device.

The default file system is Linux. The Win95 FAT file system is to be used in this example. You need to run the following commands:

```
Command (m for help): t
Selected partition 1
```

Then run the following command:

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

To view the details about all partitions of the Win95 FAT file system, run the following command:

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

Step 7 Check the information about partitions by running the following command:

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

If the partition information is displayed on the console, the partitioning is successful.

----End



7.1.3 Saving the Partition Information

To save the partition information, run the following command:

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

If the following information is displayed on the console, the partition information is successfully saved:

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

7.2 Formatting a Device

- You can format a FAT device by running the following command:

```
~ $ mkdosfs -F 32 Partition name
```



NOTE

The name of a partition depends on the type of the actual storage device. For details, see the operation instances in preceding chapters.

If the following information is displayed on the console, a partition is formatted successfully:

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

- You can format a HiFAT device by running the following command:

```
~ $ mkdosfs -F 32 Partition name
```

7.3 Mounting a Directory

Before reading/writing to a file, you must mount a partition to the **mnt** directory by running the following command:

- Mount a VFAT file system by running the following command:

```
~ $ mount -t vfat Partition name /mnt
```

- Mount a HiFAT file system by running the following command:

```
~ $ himount Partition name/mnt
```

- Mount an NFS file system by running the following command (the services of the NFS server must be enabled on the host):

```
mount -t nfs -o nolock xxx.xxx.xxx.xxx:/xx/xx /mnt
```

Where, **xxx.xxx.xxx.xxx** is the IP address for the NFS server, and **/xx/xx** is the shared directory of the NFS service.

- Mount a jffs file system by running the following command:

```
~ $ mount -t jffs2 Partition name/mnt
```



- Mount a yaffs2 file system by running the following command:
~ \$ mount -t yaffs2 Partition name/mnt
- Mount an ntfs file partition by running the following command:
~ \$ mount -t ntfs device partition name /mnt



NOTE

The name of a partition depends on the type of the actual storage device. For details, see the operation instances in preceding chapters.

7.4 Reading/Writing to a File

You can read/write to a file in various manners. In this section, the **cp** command is used to read/write to a file.

To write to a file, copy the **test.txt** file in the current directory to the **mnt** directory of a storage device by running the following command:

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

You can test the read/write speed by running the corresponding **dd** command. The parameters in the **dd** command are described as follows:

- The **dd if= parameter** indicates the source file.
- The **of=** parameter indicates the destination file.
- The **bs=** parameter indicates the size of data that is read or written each time.
- The **count=** parameter indicates the number of read/write times.
- The **bs*count** parameter indicates the total size (in bytes) of data that is read or written.

The following examples show how to test the write speed and read speed:

- To test the write speed, run the following command:
~ \$dd if=/dev/zero of=/mnt/test_file bs=1024 count=1024
- To test the read speed, run the following command:
~ \$dd if=/mnt/test_file of=/dev/null bs=1024 count=1024

7.5 Mapping Between a Kernel Driver and Its Kernel Option

You can determine whether to compile the drivers and whether to compile the drivers to the kernel or other modules by setting kernel options. The following uses the **ahci.ko** driver as an example to describe how to search for a kernel option configuration name and how to compile them in the kernel, how to compile them to modules, or whether to compile them based on module name. The kernel options of other drivers can be found in the same way.

To search the kernel option corresponding to **ahci.ko**, perform the following steps:

Step 1 Run the **grep -Rn "ahci.o" *** command on the HyperTerminal.

Then the following information is displayed:



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



NOTE

The compilation rules of the modules are defined in **Makefile**, but the compilation rules in **Makefile** is determined by the kernel configuration options related to the modules. The kernel options related to the modules are defined in the corresponding **Kconfig**. Therefore, a kernel option name corresponding to a particular module can be searched by running the **grep** command. Assume that **CONFIG_SATA_AHCI** is the kernel option corresponding to the **ahci.ko** driver.

Step 2 Run the **make menuconfig** command.

Step 3 Press / on the keyboard. The search user interface is displayed.

Step 4 Type the kernel option name **SATA_AHCI**.

Step 5 Press **Enter**.

Then all the options and descriptions containing **SATA_AHCI** are displayed. As shown in [Figure 7-1](#), **Location** indicates the position of the kernel option **CONFIG_SATA_AHCI**.

Figure 7-1 Position of the kernel option **CONFIG_SATA_AHCI**

```
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])
```

----End