



BSP

FAQs

Issue	11
Date	2016-08-10

Copyright © HiSilicon Technologies Co., Ltd. 2016. 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 provides the solutions to the problems that may occur when you use the board support package (BSP).

Related Versions

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

Product Name	Version
Hi3531	V100
Hi3535	V100
Hi3536	V100
Hi3516A	V100
Hi3516D	V100
Hi3521A	V100
Hi3520D	V300
Hi3531A	V100
Hi3518E	V200
Hi3518E	V201
Hi3516C	V200
Hi3519	V100
Hi3519	V101
Hi3516C	V300








Intended Audience

This document is intended for:

- Technical support engineers
- Software development engineers

Symbol Conventions

The symbols that may be found in this document are defined as follows.

Symbol	Description
 DANGER	Alerts you to a high risk hazard that could, if not avoided, result in serious injury or death.
 WARNING	Alerts you to a medium or low risk hazard that could, if not avoided, result in moderate or minor injury.
 CAUTION	Alerts you to a potentially hazardous situation that could, if not avoided, result in equipment damage, data loss, performance deterioration, or unanticipated results.
 TIP	Provides a tip that may help you solve a problem or save time.
 NOTE	Provides additional information to emphasize or supplement important points in the main text.

Change History

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

Issue 11(2016-08-10)

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

Chapter 4 PCIe

Section 4.14 is modified.

Issue 10 (2016-07-18)

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

Section 3.17 is added.

Issue 09 (2016-05-13)

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



Section 3.16 is added.

Issue 08 (2016-02-25)

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

Section 3.15 and section 5.10 are added.

Issue 07 (2015-10-10)

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

Section 3.14 is added.

Issue 06 (2015-07-16)

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

The contents related to the Hi3521A, Hi3520D V300, and Hi3531A are added.

Issue 05 (2015-06-09)

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

Chapter 2 Tools

Section 2.5 is added.

Chapter 4 PCIe

Section 4.14 is added.

Issue 04 (2014-07-10)

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

Chapter 3 Peripherals

Section 3.13 is added.

Issue 03 (2014-02-26)

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

Chapter 3 Peripherals

Section 3.12 is added.

Chapter 5 Flash

Section 5.9 is added.

Issue 02 (2013-12-25)

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

Chapter 3 Peripherals

In section 3.9, a caution is added, and the solution is updated.

Chapter 8 Kernel



Section 8.12 and section 8.13 are added.

Issue 01 (2012-09-15)

This issue is first official release.



Contents

About This Document.....	i
1 SDK Environment and Usage.....	1
1.1 What Do I Do If an Error Occurs When server_install Is Executed?	1
1.2 What Do MMZ and MMB Mean? How Do I Configure the MMZ Area and Size?.....	1
1.2.1 Concepts.....	1
1.2.2 MMZ Principle.....	2
1.2.3 MMZ Driver Parameters	2
1.3 What Do I Do When the Messages Such as MMB Leak Are Displayed?	2
1.4 How Do I Open a File Greater than 4 GB?	3
1.5 What Do I Do When the NFS Fails to Be Mounted?	3
1.6 What Do I Do When the System File Cannot Be Burnt or the Flash Memory Have Many Bad Blocks?	3
1.7 Why Is the Message "No init Found" Displayed When the File System Cannot Start?	4
1.8 Why Is a Message Indicating that the Console Cannot Be Opened Displayed When the File System Cannot Start?	5
1.9 Why Is TFTP Unavailable?	5
1.10 Why Is a Message Indicating that Scripts Cannot Be Found Displayed When Some Compilation Commands or Scripts Are Executed on the Server?.....	6
1.11 Why Is an Alarm Reported When a Cramfs File System Is Being Created, Indicating that the File Size Is Truncated to 16 MB and the File Size Is Incorrect?	6
2 Tools	8
2.1 How Do I Use the GDB on the Board?	8
2.2 How Do I Ignore Semaphore Events During Debugging?	8
2.3 How Do I Optimize the CPU Usage When It Reaches 100%?	8
2.4 How Do I View and Modify Registers?	9
2.5 How Do I Test the DDR Usage by Running the hiddrs Command	10
2.6 How Do I Reduce the Memory Occupied by udhpc?	10
2.7 Why Cannot udhpc Sometimes Obtain the IP Address?.....	11
2.8 How Do I Enable the Current udhpc to Support the DHCP+ Function?	11
3 Peripherals.....	14
3.1 How Do I Use a USB Keyboard and Mouse?	14
3.2 Why Are Socket Multicast Packets Lost?.....	14
3.3 How Do I Change the Ethernet PHY Address?	14



3.4 Why Cannot the GMAC Port Be Pinged?	15
3.5 How Do I Query the GMAC Port State?	15
3.6 How Do I Enable the TOE Mode?	17
3.7 What Precautions Should I Take When Applications Use Socket Interfaces in TOE Mode?	17
3.8 How Do I Set Parameters in Non-Block Mode When Applications Use Socket Interfaces?	17
3.9 What Precautions Should I Take While Using the Atheros 8035 PHY?	18
3.10 Why Is the Message "PHY No Link" Is Displayed Even Though the Board Is Connected by Using a Network Cable?	21
3.11 Why Does the Serial Port Sometimes Have No Response?	21
3.12 How Do I Create the U-boot and Kernel Images in RMII Mode?	22
3.13 What Do I Do If Some USB Flash Drives Are Incompatible Under the U-boot?	24
3.14 What Do I Do If the IP Address Is Null When the netstat Command is Executed to View the Link State?	25
3.15 What Do I Do If Some USB 3.0 Flash Drives Fail to Be Identified During Power-on Start After the Overcurrent Protection Chip of the USB 3.0 Port Is Removed?	25
3.16 What Precautions Should I Take During the Atomic Operation of the Kernel-Mode I ² C Interface?	26
3.17 What Do I Do If the Module Firmware Fails to Be Loaded When the USB 2.0 Port Connects to the BCM Wi-Fi Module?	28
4 PCIe	32
4.1 How Do I Compile the PCIe Controller Driver into the Kernel by Configuring Kernel Options?	32
4.2 How Do I Configure the PCIe Control Clock?	32
4.3 How Do I View the Allocation Information About the BAR Address for the PCIe Device?	33
4.4 How Do I View the Mapping Information About PCIe Addresses?	33
4.5 Why Does Not the PCIe MCC Drivers Take Effect After Being Loaded?	34
4.6 What Precautions Should I Take While Using the PCIe Network Adapter and PCIe-SATA Device?	34
4.7 Why Does the Slave Booter Fail to Boot the Slave Device over the PCIe Interface?	35
4.8 Why Is the Message "Unknown Hi-irq Triggered" Displayed Sometimes After the SDK Video Preview Service Runs?	35
4.9 Which Address Will Be Mapped to the Hi3531 or Hi3532 PCIe BAR Address by Default During Reset? What Precautions Should I Take When PCIe Window Is Moved?	36
4.10 What Are the Changes When All DMA Operations Are Initiated by the Slave Chip?	36
4.11 Why Does the Slave Chip Stop Working If It Uses the .ko Drivers Compiled in the SDK PCIe MCC When Two Hi3531s Are Cascaded?	37
4.12 What Precautions Should I Take When the Hi3532 Acts as a Plug-in Card?	38
4.13 Does the PCIe MCC Driver Allow the Master Device to Reset the Slave Device?	39
4.14 What Do I Do If the VO Bandwidth Is Low When Some PCIe-to-SATA Cards (Such as marvell9215) Connect to the SATA Disk to Read/Write Data (Taking the Hi3536 as an Example)?	39
5 Flash	42
5.1 What Do I Do When the Configured ECC Mode for the Hi3531 NAND Flash Does Not Take Effect?	42
5.2 How Do I Mark the Bad Blocks in a NAND Flash?	42
5.3 What Are the ECC Modes Supported by the Hi3531 NAND Flash Controller?	43
5.4 What Precautions Should I Take While Using a Large-Capacity NAND Flash?	44
5.5 What Are the Rules for Configuring the ECC Mode and Page Size of a NAND Flash?	44
5.6 Why Cannot the System Start After the U-boot Saves Environment Variables in the NAND Flash?	45



5.7 Why Cannot Images Be Combined into One When the NAND Flash Is Burnt?	45
5.8 Why Cannot the File System Be Written to the NAND Flash After Being Read from the NAND Flash?	45
5.9 How Do I Change the Quad-Wire Mode of the SPI Flash to Dual-Wire Mode?	47
5.10 How Do I Use the nandwrite Naked-Write Tool of mtd-utils?	48
6 File System	49
6.1 What Precautions Should I Take When a NAND Flash Is Mounted to the Cramfs File System?	49
7 Rapid Startup Optimization	50
8 Kernel	51
8.1 What Is the Difference Between PID and TGID Fields?	51
8.2 What Are the Priorities of Common Processes and Real-Time Processes?	51
8.3 How Do I Set the dmesg Buffer Size?	52
8.4 Why Is the MemTotal Value in /proc/meminfo Different from mem=xxxM Configured on the CLI?	52
8.5 How Do I Determine Whether Stacks Overflow on Linux?	52
8.6 How Do I Generate a Core Dump File for Problem Analysis When a Segmentation Fault Occurs?	56
8.7 What Precautions Should I Take While Using Caches?	56
8.8 Why Is the Load Average Value Too Large? Does It Affect Services?	56
8.9 Why Is the Message "vmap Allocation for Size 528384 Failed:1.1 Use vmalloc=<size> to Increase Size." Displayed If the Memory Size Is Set to 512 MB or Larger?	57
8.10 Can the Kernel Manage Two Inconsecutive Memories?	58
8.11 How Do I View the CPU Usage for Each Thread by Running the Top Command After the NPTL Tool Chain Is Used?	58
8.12 How Do I Display, Save, or Search the Stack Information About a Suspended Program by Using the Backtrace Function of the Gilibc Tool Chain?	59
8.13 How Do I Configure a 1 ms Timer?	59



Figures

Figure 2-1 Configuration with parameters	12
Figure 2-2 Configuration without parameters	13
Figure 3-1 GMAC TX and RX states	16
Figure 3-2 Interrupt summary	16
Figure 3-3 Start and identification process of the USB 3.0 flash drive	26
Figure 3-4 Applying for lock in the kernel-mode interface	26
Figure 3-5 Write operation over the I ² C interface	27
Figure 4-1 menconfig GUI	40
Figure 5-1 Hardware configuration	42
Figure 5-2 Block positions	45
Figure 5-3 Status of the NAND flash after it is erased	46
Figure 5-4 Data that will be written to the NAND flash	46
Figure 5-5 Status of the NAND flash after the data is written.....	46
Figure 5-6 Data in the file system after data is written to the NAND flash.....	46
Figure 5-7 Structure of blocks in the NAND flash.....	48
Figure 8-1 Stack structure	53



Tables

Table 2-1 Commands related to register operations.....	9
Table 5-1 Page sizes and ECC modes.....	43
Table 8-1 Priorities of common processes and real-time processes.....	51



1 SDK Environment and Usage

1.1 What Do I Do If an Error Occurs When server_install Is Executed?

The following error messages are displayed:

```
./server_install
\33[32m
you must use 'root' to execute this shell
\33[39m
./cross.install: 25: Syntax error: "do" unexpected (expecting "fi")
./cross.install: 28: Syntax error: "do" unexpected (expecting "fi")
./cross.install: 30: Syntax error: "do" unexpected (expecting "fi")
```

This is because the scripts in the software development kit (SDK) are based on bash, but dash or other command-line programs are installed on the used Linux server. To resolve this problem, you are advised to remove dash or change the default **sh** to **bash**. To be specific, delete the soft link pointing to the existing sh, and recreate a soft link pointing to the bash as follows:

```
cd /bin
rm -f sh
ln -s /bin/bash /bin/sh
```

1.2 What Do MMZ and MMB Mean? How Do I Configure the MMZ Area and Size?

The following describes the concepts.

1.2.1 Concepts

MMZ is the acronym of media memory zone and is an allocation pool. MMB is the acronym of media memory block.



The physical memory zone managed by the MMZ is not controlled by the Linux kernel, and is used only by media drivers such as decoder and DEMUX. The MMB is a memory block allocated from the MMZ.

1.2.2 MMZ Principle

The MMZ drives and manages the allocation pools created by users. The allocation pools from which the memory is allocated to user programs can be specified. The allocator searches for the allocation pools that meet the requirements and allocates appropriate memory blocks to applications.

1.2.3 MMZ Driver Parameters

"mmz=" is used to define the allocation pool of media-mem. The format is as follows:

```
mmz=<name>,<gfp>,<phys_start_addr>,<size>:<name>,<gfp>,<phys_start_addr>:
...
```

- **<name>**: It is a character string and is the name of the allocation pool such as **ddr**.
- **<gfp>**: It is expressed by digits and indicates the allocation pool attribute. It is used to specify the memory in which the MMZ is located (such as DDR, SDRAM, DDR2, or DDR3) for the board with multiple memories. If this parameter is set to **0**, the memory is automatically specified. Typically, this parameter is set to **0**.
- **<phys_start_addr>**: It indicates the physical start position of the allocation pool and is expressed in hexadecimal, for example, **0x86000000**. Note that the memory area of the MMZ cannot overlap the memory area of the Linux kernel, and the physical start address for the MMZ is the memory start address plus the memory size occupied by the Linux kernel. On the Hi353x platform, the memory start address is fixed at 0x80000000. Assume that bootargs of the board is '**mem=96M console=ttyAMA0,115200 root=xxxx**', and the Linux kernel occupies 96 MB memory space. In this case, set the MMZ start address to **0x86000000** (0x80000000 + 96 MB).
- **<size>**: It indicates the allocation pool size and is expressed as 0x100000 or 1 MB. Note that the sum of the allocation pool size and the memory size of the Linux kernel cannot exceed the actual size of the physical memory. For example, if the physical memory of the board is 256 MB and Linux kernel occupies 96 MB memory, the maximum MMZ size is 160 MB (256 MB – 96 MB).

The preceding parameters are mandatory. The parameters are separated with commas (.). Multiple allocation pools can be specified. The allocation pools are separated with semicolon (:). For example, modprobe mmz mmz=ddr,0,0x86000000,64M:vdec,0,0x8A000000,64M.

1.3 What Do I Do When the Messages Such as MMB Leak Are Displayed?

Typically, the following messages are displayed:

```
MMB LEAK(pid=11093): 0x880BA000, 3686400 bytes, ''
mmz_userdev_release: mmb<0x880BA000> mapped to userspace 0x408d1000 will be
force unmapped!
MMB LEAK(pid=11093): 0x884FD000, 2764800 bytes, ''
mmz_userdev_release: mmb<0x884FD000> mapped to userspace 0x40d14000 will be
```



```
force unmapped!
MMB LEAK(pid=11093): 0x8843E000, 520192 bytes, 'decctrl'
mmz_userdev_release: mmb<0x8843E000> mapped to userspace 0x40c55000 will be
force unmapped!
MMB LEAK(pid=11093): 0x884BD000, 262144 bytes, 'Hdec'
mmz_userdev_release: mmb<0x884BD000> mapped to userspace 0x40cd4000 will be
force unmapped!
```

The displayed messages do not indicate memory leakage. The SDK displays these messages when it forcibly releases resources after detecting that some resources are not released when the applications are closed. Check whether application deinitialization is complete. For example, check whether all created channels are destroyed and the enabled devices are disabled.

1.4 How Do I Open a File Greater than 4 GB?

If you fail to open a file greater than 4 GB by running **fopen**, you are advised to add the following options in the **makefile** option:

```
-D_GNU_SOURCE -D_XOPEN_SOURCE=600 -D_LARGEFILE_SOURCE -D_LARGEFILE64_SOURCE
-D_FILE_OFFSET_BITS=64
```

1.5 What Do I Do When the NFS Fails to Be Mounted?

Run the following command to mount the network file system (NFS):

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

If the following information is displayed, the **-o nolock** option is not added:

```
rpcbind: server localhost not responding, timed out
RPC: failed to contact local rpcbind server (errno 5).
rpcbind: server localhost not responding, timed out
RPC: failed to contact local rpcbind server (errno 5).
rpcbind: server localhost not responding, timed out
RPC: failed to contact local rpcbind server (errno 5).
```

If the NFS does not respond especially when large files are being read and the message such as "nfs server not responding, still trying" is displayed, the **-o tcp** option is not added.

1.6 What Do I Do When the System File Cannot Be Burnt or the Flash Memory Have Many Bad Blocks?

Typically, the following error messages are displayed:

```
hisilicon # nand write.yaffs 0x82000000 0x700000 0x1928ac0
```



```
NAND write: device 0 offset 0x700000, size 0x1928ac0
Attempt to write non page aligned data, length 26380992 4096 128
26380992 bytes written: ERROR
```

This problem occurs when the **pagesize** and **ecc** parameters specified when a YAFFS file system is being created are inconsistent with actual parameter values of the NAND flash on the board. Check the **pagesize** and **ecc** parameters for the NAND flash, and re-create a file system. Note that the inconsistency of **pagesize** and **ecc** parameters may not cause burning failures. The image of the file system may be successfully burnt, but the system cannot start and the messages such as "bad block n" are displayed. When this error occurs, re-create a system file, clear the area where the YAFFS file system is located by running **nandscrubNandFlash address length**, and then burn the image of the YAFFS file system. For example, **nand scrub 400000 1000000** indicates that a 64 MB memory is cleared from 0x400000. If the last parameter is not specified, the space is cleared from this address to the end of the NAND flash. For example, **nand scrub 400000** indicates that the entire flash memory is cleared from 0x400000.

To obtain the page size and error correcting code (ECC) type of the NAND flash, view the information displayed when the kernel starts. The following information in red indicates the page size and ECC type.

```
Hisilicon Nand Flash Controller V300 Device Driver, Version 1.00
Nand ID: 0xAD 0xDC 0x10 0x95 0x54 0xAD 0xDC 0x10
Nand(Hardware): Block:128K Page:2K Ecc:1bit Chip:512M OOB:64Byte
NAND device: Manufacturer ID: 0xad, Chip ID: 0xdc (Hynix NAND 512MiB 3,3V 8-bit)
```

If the preceding information cannot be found, the HiSilicon SDK does not support such type of flash memory.

1.7 Why Is the Message "No init Found" Displayed When the File System Cannot Start?

Typically, the following error messages are displayed:

```
ata2: failed to resume link (SControl 0)
ata2: SATA link down (SStatus 0 SControl 0)
yaffs: dev is 32505858 name is "mtdblock2" rw
yaffs: passed flags ""
VFS: Mounted root (yaffs2 filesystem) on device 31:2.
Freeing init memory: 100K
Kernel panic - not syncing: No init found. Try passing init= option to kernel.
See Linux Documentation/init.txt for guidance.
```

This problem occurs for either of the following reasons:

- The **pagesize** and **ecctype** parameters configured when a YAFFS file system is being created are inconsistent with actual parameter values of the NAND flash. As a result, the kernel cannot identify the YAFFS file system. When **make build** is executed for compiling the SDK, a file system is created based on the reference board released by the SDK by default. The used **pagesize** and **ecctype** parameters may be inconsistent with the configured values.



- The **bootargs** parameter is not properly configured.
For example, the setting **setenv bootargs ' bootargs =mem=96M console=ttyAMA0,115200 root=/dev/mtdblock2 rootfstype=YAFFS2 mtdparts=hinand:4M(boot),60M(rootfs),-(others)'** is used. In addition, the **rootfstype** parameter in **bootargs** may be incorrect. For example, **rootfstype** is set to **yaffs2**, but the image of the jffs2 file system is burnt. As a result, the kernel cannot identify the file system.

1.8 Why Is a Message Indicating that the Console Cannot Be Opened Displayed When the File System Cannot Start?

This problem occurs when the **console** file for creating the file system does not exist in **/dev** or the attribute of the **console** file is incorrect. The following shows the correct attribute of the **console** file:

```
cd SDK root directory
ls ./pub/rootbox/dev/ -l
Total 0
crw-r--r-- 1 root root 5, 1 2010-10-18 18:52 console
crw-r--r-- 1 root root 204, 64 2010-10-18 18:52 ttyAMA0
crw-r--r-- 1 root root 204, 65 2010-10-18 18:52 ttyAMA1
crw-r--r-- 1 root root 204, 64 2010-10-18 18:52 ttyS000
```

1.9 Why Is TFTP Unavailable?

Do as follows:

- Check whether the board properly connects to any Ethernet port by using a network cable.
- Use the network cable in other scenarios to ensure that the network cable works properly.
- Check whether the board directly connects to the PC by using a network cable. If the board does not directly connect to the PC, check whether the network needs to be authenticated and an agent is required. Some switches may mask the IP addresses that are not dynamically allocated by themselves. The network failure may occur if the IP addresses are configured by running **setenv ipaddr** under the boot program. The network management systems (NMSs) of some companies mask the media access control (MAC) addresses and IP addresses beyond the specified address range, which may cause network access failures. You are advised to perform the TFTP operation by directly connecting the board to the PC.
- Check whether the IPv6 protocol is enabled on the PC. If it is enabled, disable it because the board does not support the IPv6 protocol under the boot program.



1.10 Why Is a Message Indicating that Scripts Cannot Be Found Displayed When Some Compilation Commands or Scripts Are Executed on the Server?

This problem may occur if the server runs on a 64-bit operating system (OS), because the SDK requires the 32-bit C library. To resolve this problem, install a 32-bit library on the server. You are advised to search for the method of installing a 32-bit library on the server on Internet.

The following are reference commands, which are tested only on a Ubuntu 64-bit server. Note that the server must connect to the Internet.

```
apt-get install libc6-i386
apt-get install gcc-multilib g++-multilib libc6-dev-i386 libzip-dev
apt-get install ia32-libs lib32asound2 libasound2-plugins
apt-get install -y lib32nss-mdns lib32gcc1 lib32ncurses5 lib32stdc++6 lib32z1
libc6 libc6-i386 libcanberra-gtk-module
dpkg -i --force-all getlibs-all.deb
```



getlibs-all.deb

Download **getlibs-all.deb** on the server, and run it.

1.11 Why Is an Alarm Reported When a Cramfs File System Is Being Created, Indicating that the File Size Is Truncated to 16 MB and the File Size Is Incorrect?

Typically, the following error messages are displayed:

```
root@Athena:~$ mkcramfs ./tools/ root.img
Directory data: 37924 bytes
Everything: 43936 kilobytes
Super block: 76 bytes
CRC: 40e6dc31
warning: file sizes truncated to 16MB (minus 1 byte)
warning: gids truncated to 8 bits (this may be a security concern)
```

This problem occurs because the cramfs file system does not support a file greater than 16 MB. To allow the cramfs file system to support a file greater than 16 MB, do as follows:

- Ensure that the cramfs file system and memory technology device (MTD) driver are supported.
- Modify the mkcramfs source code to create special file systems. Download the cramfs source code at <http://sourceforge.net/projects/cramfs/>. Modify **#define CRAMFS_SIZE_WIDTH 24** in **cramfs/linux/cramfs_fs.h** after decompressing the



source code. The value **24** indicates 16 MB. If you require a 256 MB file, change **24** to **28**.

- Modify the cramfs file system of the kernel. To be specific, modify **cramfs_fs.h** in **include/linux** by modifying the macro **CRAMFS_SIZE_WIDTH**.



2 Tools

2.1 How Do I Use the GDB on the Board?

The GDB executable program is stored in `~/osdrv/tools/board_tools/gdb`. To use the GDB on the board, copy the GDB corresponding to the used compiler to `/usr/bin` of the board, rename it **gdb**, and grant the executable permission by running **chmod a+x gdb**. You can also run the GDB by using the GDB absolute path after mounting the NFS directory.

2.2 How Do I Ignore Semaphore Events During Debugging?

The common problem is that the GDB stops after displaying the following information until you enter **c**:

```
Program received signal SIG32, Real-time event 32.  
0x4052d940 in __rt_sigsuspend () from /lib/libc.so.0
```

The preceding information can be ignored. Enter **handle SIG32 pass noprint nostop** on the GDB command-line interface (CLI) to enable the GDB to ignore the SIG32 information. Other information is processed in the same way.

2.3 How Do I Optimize the CPU Usage When It Reaches 100%?

Perform the following steps:

- Step 1** Run **telnetd&** on the board to enable the telnet service.
- Step 2** Start applications, and log in to the board by using the telnet service when the applications are properly running. Do not press **CTRL+Z** to enable the applications to run on the background.
- Step 3** Run **top -d 1** on the telnet terminal to view the thread whose pthread identification (PID) corresponds to the highest CPU usage, and record the PID.
- Step 4** Press **q** to end the **top** command or log in to the board by using another telnet terminal.



- Step 5** Run the GDB on the telnet terminal without specifying any parameter. On the GDB CLI, enter **attach pid**. The PID is the one recorded in [Step 3](#). After the PID is attached, enter **bt full** and view the stack information. Typically, the function on the top of the stack corresponds to the thread whose CPU usage is the highest. Check whether the CPU is released in the loop body of the thread. If the CPU is not released, add **usleep(20000)** to the loop body to release the CPU.
- Step 6** Recompile programs. Verify that the CPU usage decreases. If the CPU usage does not decrease, repeat [Step 2](#) to [Step 6](#).

----End

2.4 How Do I View and Modify Registers?

View and modify registers in either of the following ways:

- Read registers by calling HI_SYS_ReadRegister or write registers by calling HI_SYS_WriteRegister.
- Run programs on the board CLI.

[Table 2-1](#) describes the commands related to register operations. These commands are stored in **/usr/sbin**.

Table 2-1 Commands related to register operations

Command	Parameter	Function	Remarks
himm	Parameter 1: address (mandatory) Parameter 2: value (optional)	Change the address defined by the address parameter to the value defined by the value parameter.	If the value parameter is not set, the system displays the value corresponding to the address parameter and asks you to enter a new value.
himd	Parameter 1: address (mandatory) Parameter 2: length (optional)	Display the content with length bytes in big endian mode starting from address .	If the length parameter is not set, 256-byte content is displayed by default.
himd.l	Parameter 1: address (mandatory) Parameter 2: length (optional)	Print the content with length bytes in little endian mode starting from address .	If the length parameter is not set, 256-byte content is displayed by default.
himc	Parameter 1: address (mandatory) Parameter 2: value (mandatory) Parameter 3: length (mandatory)	Set the content with length bytes to value starting from address .	This command is used to modify the content only in the physical memory.



The **address** parameter can be set to register address or memory address. Therefore, the commands in [Table 2-1](#) can also be used to view and modify the memory.

2.5 How Do I Test the DDR Usage by Running the hiddrs Command

The **hiddrs** command is used to collect statistics on the DDR usage in real time. The parameters are described as follows:

- **-d** indicates the DDR controller (DDRC). The value 0 indicates DDRC0, the value 1 indicates DDRC1, and the value 2 indicates DDRC0 and DDRC1. The default value is 0.
- **-f** indicates the DDR working frequency. The default value is 400 MHz.
- **-w** indicates the DDR bit width. The DDR width can be 32 bits or 16 bits, and the default width is 32 bits.
- **-i** indicates the time interval. The unit is second.
- **-h** indicates that the help information is displayed.

Example: **hiddrs -d 0 -f 400 -w 32 -i 1**

2.6 How Do I Reduce the Memory Occupied by udhpcp?

The problem symptom is that **system(udhpcp)** fails to be executed. The **system** is implemented by using fork, and the subprocesses copy the virtual memory (VM) space of the parent processes. The system fails to call udhpcp when the parent processes occupy a large amount of VM space. The root cause is that the subprocesses fail to allocate the VM space. To resolve this problem, run the following command:

```
echo 1 > /proc/sys/vm/overcommit_memory
```

A better way to resolve this problem is calling udhpcp by using **posix_spawn** but not the **system**. See the following instance:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <spawn.h>
#include <sys/wait.h>
int main(int argc, char* argv[])
{
    pid_t pid;
    int err;
    char *spawnedArgs[] = {"/bin/ls", "-l", "/home ", NULL}; /* posix_spawn
requires the full path (absolute path) of the subprocess command be
specified. */
    char *spawnedEnv[] = {NULL};
```



```
printf("Parent process id=%ld\n", getpid());
if( (err=posix_spawn(&pid, spawnedArgs[0], NULL, NULL,
    spawnedArgs, spawnedEnv)) !=0 )
{
    fprintf(stderr,"posix_spawn() error=%d\n",err), exit(-1);
}
printf("Child process id=%ld\n", pid);

/* Wait for the spawned process to exit */
(void)wait(NULL);

return 0;
}
```

For details about the usage of `posix_spawn`, search for the related information on the Internet.

2.7 Why Cannot udhcpd Sometimes Obtain the IP Address?

udhcpd initiates only three IP address requests on the board by default. As the server responds slowly, udhcpd needs to initiate more IP address requests.

To initiate more requests, run **udhcpd -t 50**.

2.8 How Do I Enable the Current udhcpd to Support the DHCP+ Function?

The following parameters can be used:

- -V: indicates the vendor information sent to the server.
- -D: indicates the vendor information expected by the client.

Take the **udhcpd -V question -D answer -f** command as an example. The following describes how the command is executed:

- The board sends questions to the server.
- The server responds answers to the board.
- The board checks whether the received data is answers. If the received data is not answers, the configuration fails. If the received data is answers, the configuration is successful.

Figure 2-1 shows the configuration with parameters.

Figure 2-1 Configuration with parameters

```
# ./busybox udhcpd -V600831STB -D600831STB
udhcpd (v0.9.9-pre) started
Jan 1 01:31:15 udhcpd[800]: udhcpd (v0.9.9-pre) started
Sending discover...
Jan 1 01:31:15 udhcpd[800]: Sending discover...
Offer vendor class: 600831STB
Client vendor class: 600831STB
Sending select for 192.168.1.19...
Jan 1 01:31:17 udhcpd[800]: Sending select for 192.168.1.19...
Sending select for 192.168.1.19...
Jan 1 01:31:19 udhcpd[800]: Sending select for 192.168.1.19...
Offer vendor class: 600831STB
Client vendor class: 600831STB
Lease of 192.168.1.19 obtained, lease time 168960
Jan 1 01:31:19 udhcpd[800]: Lease of 192.168.1.19 obtained, lease time 168960
deleting routers
route: SIOCDELRT: No such process
adding dns 192.168.1.1
#
```

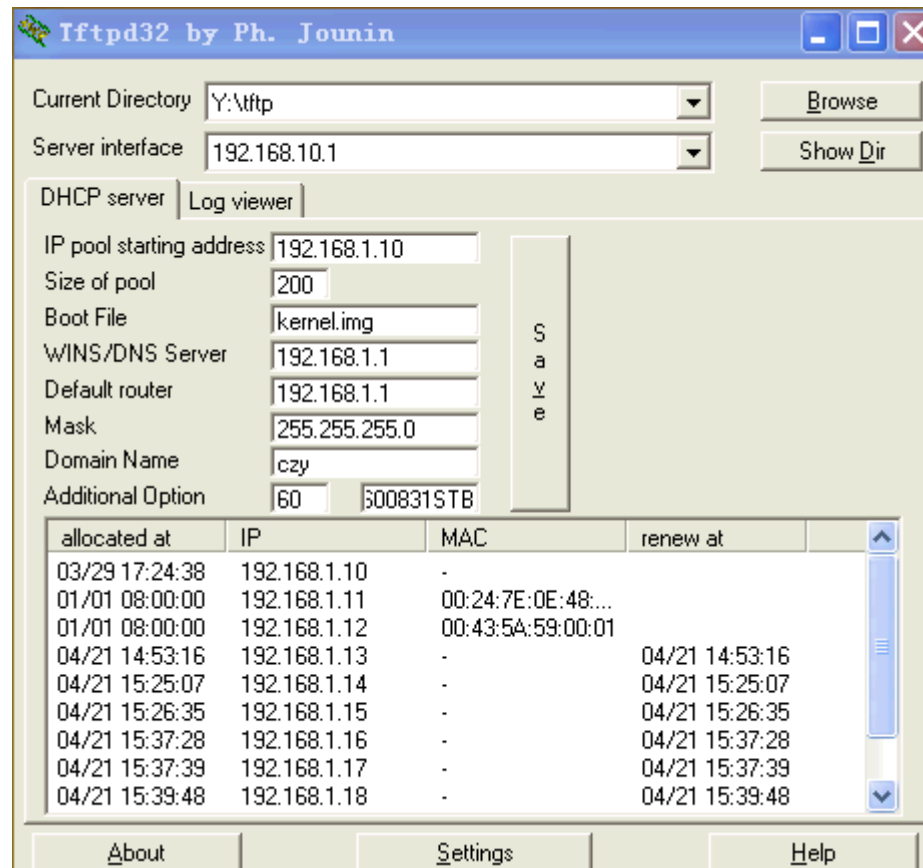


Figure 2-2 shows the configuration without parameters.



Figure 2-2 Configuration without parameters

```
# ./busybox udhcpd
udhcpd (v0.9.9-pre) started
Jan  1 01:32:58 udhcpd[810]: udhcpd (v0.9.9-pre) started
Sending discover...
Jan  1 01:32:58 udhcpd[810]: Sending discover...
Sending discover...
Jan  1 01:33:01 udhcpd[810]: Sending discover...
Sending select for 10.85.180.250...
Jan  1 01:33:01 udhcpd[810]: Sending select for 10.85.180.250...
Received DHCP NAK
Jan  1 01:33:01 udhcpd[810]: Received DHCP NAK
/usr/share/udhcpd/default.script: exec: line 7: /usr/share/udhcpd/default.nak: not found
Sending discover...
Jan  1 01:33:04 udhcpd[810]: Sending discover...
Sending discover...
Jan  1 01:33:07 udhcpd[810]: Sending discover...
Sending select for 10.85.181.24...
Jan  1 01:33:07 udhcpd[810]: Sending select for 10.85.181.24...
Received DHCP NAK
Jan  1 01:33:07 udhcpd[810]: Received DHCP NAK
/usr/share/udhcpd/default.script: exec: line 7: /usr/share/udhcpd/default.nak: not found
Sending discover...
Jan  1 01:33:10 udhcpd[810]: Sending discover...

Sending discover...
Jan  1 01:33:13 udhcpd[810]: Sending discover...
Sending select for 10.85.181.93...
Jan  1 01:33:13 udhcpd[810]: Sending select for 10.85.181.93...
Lease of 10.85.181.93 obtained, lease time 28800
Jan  1 01:33:13 udhcpd[810]: Lease of 10.85.181.93 obtained, lease time 28800
deleting routers
route: SIOCDELRT: No such process
adding dns 10.72.55.81
adding dns 10.72.255.100
adding dns 10.98.48.39
#
#
```


3 Peripherals

3.1 How Do I Use a USB Keyboard and Mouse?

The SDK kernel supports the USB keyboard and mouse by default. The USB keyboard and USB mouse can be used after they are connected. See the following sample:



usb-key-mouse.zip

3.2 Why Are Socket Multicast Packets Lost?

When receiving User Datagram Protocol (UDP) multicast data, the user-state applications are performing other delay operations, for example, writing stream data to the USB storage device. The applications delay receiving the UDP data packets, but the socket receive (RX) buffer is 108,544 bytes by default. Therefore, new UDP data packets cannot be received because the socket RX buffer is full. As a result, packets are lost.

To resolve the problem, run the following commands to increase the RX buffer size:

```
echo 20000000 > /proc/sys/net/core/rmem_max
echo 20000000 > /proc/sys/net/core/rmem_default
echo 20000000 > /proc/sys/net/core/netdev_max_backlog
```

Make modifications based on the actual stream transmit (TX) rate and the delay of RX programs.

3.3 How Do I Change the Ethernet PHY Address?

- Under the kernel

Run **make ARCH=arm CROSS_COMPILE=arm-hisiv200-linux- menuconfig** in **osdvr/kernel/linux -3.0.y**, and choose the following options on the **menuconfig** menu:

```
Device Drivers --->
[*] Network device support --->
    [*] Ethernet (1000 Mbit) --->
```



```
<M> STMicroelectronics 10/100/1000 Ethernet driver --->
(1) STMMAC MAC #0 PHY ID <<The PHY address is 1.
(2) STMMAC MAC #1 PHY ID <<The PHY address is 2.
```

- Under the U-boot:

Modify the **godnet.h** file in **include\configs** in the U-boot code.

```
#define CONFIG_NET_STMMAC
#define CONFIG_TNK
#ifdef CONFIG_NET_STMMAC
    #define STMMAC_GMACADDR      (0x101c0000)
    #define STMMAC_DMAADDR      (0x101c1000)
    #define STMMAC_IOSIZE      (0x10000)
    #define STMMAC_FRQDIV      (0)
    #define STMMAC_PHYADDR0      (1) <<The PHY address is 1.
    #define STMMAC_PHYADDR1      (2) <<The PHY address is 2.
    #define STMMAC_PHYNAME      "0:01"
    #define STMMAC_RGMII
    #define CONFIG_PHY_GIGE
#endif /* CONFIG_NET_STMMAC */
```

3.4 Why Cannot the GMAC Port Be Pinged?

The GMAC port cannot be pinged for the following reasons:

- The IP address for the board overlaps the IP addresses for other devices.
Symptom: A long delay occurs when the board is pinged at the PC end, and the board sometimes cannot be pinged.
Method: Transmit ping packets from the PC to the board, disconnect the network cable from the GMAC port corresponding to the board, and then check whether the board can be pinged.
- The MAC address for the GMAC port overlaps the MAC addresses for other devices.
Symptom: The board cannot be pinged at the PC end.
Method: Ping the board at the PC end and run the **arp -a** command at the PC end to view the arp list of the PC. Check whether the MAC address corresponding to the IP address matches the MAC address for the board.

3.5 How Do I Query the GMAC Port State?

To view the state of GMAC port 0, run **GMAC0: himd.l 0x101c1014**; to view the state of GMAC port 1, run **GMAC1: himd.l 0x101c1114**.

The upper 16 bits indicate the information about the logical state machine, and the lower 16 bits indicate the information about the reported interrupts. [Figure 3-1](#) describes important bits.

Figure 3-1 GMAC TX and RX states

22:20	TS: Transmit Process State	000	RO
These bits indicate the Transmit DMA FSM state. This field does not generate an interrupt.			
<ul style="list-style-type: none"> 3'b000: Stopped; Reset or Stop Transmit Command issued. 3'b001: Running; Fetching Transmit Transfer Descriptor. 3'b010: Running; Waiting for status. 3'b011: Running; Reading Data from host memory buffer and queuing it to transmit buffer (Tx FIFO). 3'b100: TIME_STAMP write state. 3'b101: Reserved for future use. 3'b110: Suspended; Transmit Descriptor Unavailable or Transmit Buffer Underflow. 3'b111: Running; Closing Transmit Descriptor. 			
19:17	RS: Receive Process State	000	RO
These bits indicate the Receive DMA FSM state. This field does not generate an interrupt.			
<ul style="list-style-type: none"> 3'b000: Stopped; Reset or Stop Receive Command issued. 3'b001: Running; Fetching Receive Transfer Descriptor. 3'b010: Reserved for future use. 3'b011: Running; Waiting for receive packet. 3'b100: Suspended; Receive Descriptor Unavailable. 3'b101: Running; Closing Receive Descriptor. 3'b110: TIME_STAMP write state. 3'b111: Running; Transferring the receive packet data from receive buffer to host memory. 			

Figure 3-2 describes the current register (register 5). The numbers in [] indicate bit IDs. Bit 11 and bit 12 are reserved.

Figure 3-2 Interrupt summary

16	NIS: Normal Interrupt Summary	0	R_SS_WC
Normal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in DMA Register 7:			
<ul style="list-style-type: none"> Register 5[0]: Transmit Interrupt Register 5[2]: Transmit Buffer Unavailable Register 5[6]: Receive Interrupt Register 5[14]: Early Receive Interrupt 			
Only unmasked bits affect the Normal Interrupt Summary bit.			
This is a sticky bit and must be cleared (by writing a 1 to this bit) each time a corresponding bit that causes NIS to be set is cleared.			
15	AIS: Abnormal Interrupt Summary	0	R_SS_WC
Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in DMA Register 7:			
<ul style="list-style-type: none"> Register 5[1]: Transmit Process Stopped Register 5[3]: Transmit Jabber Timeout Register 5[4]: Receive FIFO Overflow Register 5[5]: Transmit Underflow Register 5[7]: Receive Buffer Unavailable Register 5[8]: Receive Process Stopped Register 5[9]: Receive Watchdog Timeout Register 5[10]: Early Transmit Interrupt Register 5[13]: Fatal Bus Error 			
Only unmasked bits affect the Abnormal Interrupt Summary bit.			
This is a sticky bit and must be cleared each time a corresponding bit that causes AIS to be set is cleared.			



3.6 How Do I Enable the TOE Mode?

The Ethernet port is in bypass mode in the SDK by default. To change the mode to TCP/IP offload engine (TOE) mode, do as follows:

Modify the startup script `/etc/init.d/S81toe`, comment out `stmmac.ko` in `insmod/hitoe/`, and run the following commands:

```
echo 8192 > /proc/sys/vm/min_free_kbytes
echo 200 > /proc/sys/vm/vfs_cache_pressure
insmod /hitoe/stmmac.ko hitoe=1
```

For details about the settings, see the descriptions of the startup script `S81toe`. As the reserved memory increases, the value for the **MEM** parameter in `bootargs` needs to be increased. Otherwise, the memory is insufficient.

3.7 What Precautions Should I Take When Applications Use Socket Interfaces in TOE Mode?

In TOE mode, protocol processing is implemented by hardware regardless of block mode or non-block mode. When the kernel buffer is full, `EAGAIN` is returned to upper-layer applications. In bypass mode, if the kernel buffer is full in block mode, `EAGAIN` is not returned but data continues to be transmitted. In block or non-block mode, data transmission fails temporarily if the following condition is met: `send` return value < 0 && (`errno == EINTR || errno == EWOULDBLOCK || errno == EAGAIN`). In this case, retry data transmission. If the following condition is met, the connection is abnormal and needs to be disconnected: `send` return value ≤ 0 , && `errno != EINTR && errno != EWOULDBLOCK && errno != EAGAIN`.

When applications call the `send` function, wait and retry data transmission instead of calling the `close` function for disconnection if the following condition is met: `return value < 0 && (errno == EINTR || errno == EWOULDBLOCK || errno == EAGAIN)`. This avoids the preceding problems.

3.8 How Do I Set Parameters in Non-Block Mode When Applications Use Socket Interfaces?

The concepts of the block input/output (I/O) and non-block I/O for the network handle are used during network programming.

Basic Concepts

- Block I/O: Socket interfaces are returned only after I/O operations (even errors occur) are complete.
- Non-block I/O: Socket interfaces are returned no matter whether operations are complete. You need to check whether operations are successful by other means.

I/O Mode Setting

Set the I/O mode of a socket interface to block mode or non-block mode in either of the following ways:

- Method 1: Set `fcntl`, obtain flags by using `F_GETFL`, and set `flags|O_NONBLOCK` by using `F_SETFL`.

The `fcntl` function can be used to set a socket handle to non-block mode.

```
flags = fcntl(sockfd, F_GETFL, 0); //Obtain the flags value of a file.
```

```
fcntl(sockfd, F_SETFL, flags|O_NONBLOCK); //Set the mode to non-block mode.
```

After settings, all operations related to `sockfd` are non-block operations.

```
flags = fcntl(sockfd, F_GETFL, 0);
```

```
fcntl(sockfd, F_SETFL, flags & ~O_NONBLOCK); //Set the mode to non-block mode.
```

After settings, all operations related to `sockfd` are block operations.

- Method 2: Set the parameters of the `recv` and `send` functions. During read or transmission operations, `sockfd` or `filefd` is temporarily set to non-block mode.

The last **flag** parameters of the `recv` and `send` functions can be set to **MSG_DONTWAIT**.

Temporarily Set `sockfd` to non-block mode no matter whether the original mode is block mode or non-block mode.

```
recv(sockfd, buff, buff_size, MSG_DONTWAIT); //Receive messages in non-block mode.
```

```
send(sockfd, buff, buff_size, MSG_DONTWAIT); //Send messages in non-block mode.
```

3.9 What Precautions Should I Take While Using the Atheros 8035 PHY?



CAUTION

The delay on the `GTX_CLK` pin for the Atheros 8035 PHY is implemented by configuring the debug register but not by connecting components on the hardware link for the common PHY.

See the following register description.



4.2.25 rgmii rx clock delay control

Offset: 0x00

Bit	Name	Type		Description
15	Sel_clk125m_dsp	Mode	R/W	Control bit for rgmii interface rx clock delay: 1 = rgmii rx clock delay enable 0 = rgmii rx clock delay disable
		HW Rst.	1	
		SW Rst.	1	
14:0	Reserved	Mode	RO	
		HW Rst.	2EE	
		SW Rst.	2EE	

The debug interface can be read or written by configuring the following registers over the management data input/output (MDIO) interface:

4.1.23 Debug Port Address Offset

Offset: 0x1D

Mode: Read/Write

Hardware Reset: 0

Software Reset: 0

Bit	Name	Description
15:6	RES	Reserved
5:0	ADDRESS_OFFSET	Address index to access the debug registers

4.1.24 Debug Port Data

Offset: 0x1E

Mode: Read/Write

Hardware Reset: 0x82EE

Software Reset: 0x82EE

Bit	Name	Description
15:0	DATA	Data contents of the debug registers as addressed by the "Debug Register Summary" register

The following are operation instances:

- Under the U-boot:

```
mw 0x101c0014 0x5
mw 0x101c0010 0x1743
mw 0x101c0014 0x0100
mw 0x101c0010 0x1783
```

- Under the kernel:

Add the following operations to the stmmac_open function in **stmmac_main.c** in **drivers/net/stmmac/**:

```
priv->mii->write(priv->mii, 0x2, 0x1d, 0x5);
priv->mii->write(priv->mii, 0x2, 0x1e, 0x0100);
```



CAUTION

The PHY fails to be communicated under the U-boot on the Fast Ethernet. When the related register is read, it is found that the network is adapted to Gigabit Ethernet. This is because the read value of the PHY register whose offset address is 0F should be 0x2000, but the actual read value is 0xA000. As a result, the rate is considered as a gigabit rate based on the driver algorithm.

To resolve this issue, comment out the statements in red in the **miiphyutil.c** file in the **common** directory under the U-boot:

```
int miiphy_speed (char *devname, unsigned char addr)
{
    u16 bmcrr, anlpar;
    u16 btsr, val;

    miiphy_read(devname, addr, PHY_BMSR, &val);

    if (val & BMSR_ESTATEN) {
        #if 0
            if (miiphy_is_1000base_x (devname, addr)) {
                return _1000BASET;
            }
        #endif
    }
}
```

3.10 Why Is the Message "PHY No Link" Is Displayed Even Though the Board Is Connected by Using a Network Cable?

When a network-related operation is performed on the Hi3531 demo board for the first time, the following messages are displayed even though the board is connected by using a network cable:

[illegible]

It takes the PHY on the demo board a period of time to negotiate the working mode and rate with the PHY at the peer end. After negotiation, it also takes a period of time to reset the PHY. If a network-related operation (such as ping operation or TFTP operation) is performed immediately after the board is connected by using a network cable, the operation may fail. You are advised to perform a network-related operation after the power indicator for the PHY on the board is on (typically in green).

3.11 Why Does the Serial Port Sometimes Have No Response?

The terminal program on the PC does not respond to keyboard inputs and display output information frequently during code debugging. This problem is caused for the following reasons:



- Reason 1: The program or kernel is suspended.
- Reason 2: The program is running, but the serial port has no response.

To determine the exact reason, choose either of the following methods:

- Enable the telnet service after the kernel starts. When the kernel is suspended, use telnet to connect the board. If the board fails to be connected, this problem is caused for reasons 1. If the board can be connected, this problem is caused for reason 2.
- After the serial port has no response, view internal serial registers by using the simulator or telnet. The following uses the Hi3531 as an example. If bit 11 of a serial port register (base address 0x20080000) or bit 3 of a serial port register (base address 0x20080004) is 1, this problem is caused by the serial port overflow error.

For reason 2, modify the serial port driver, and reinitialize the serial port if the overflow error occurs. This ensures that the serial port always responds. Replace **amba-pl011.c** in `~/osdrv/kernel/linux-3.0.y/drivers/tty/serial/` with the file attached below, and recompile the kernel.

Note that the file attached is based on Linux-3.0.y.



amba-pl011.c

3.12 How Do I Create the U-boot and Kernel Images in RMII Mode?

To create the U-boot and kernel images in RMII mode, perform the following steps:

Step 1 Modify the U-boot table.

1. Set the ETH mode to RMII mode by setting bit 3 of the CRG register (0x200300CC), set the RMII clock source to PAD input clock (the external clock source is selected to support the demo board) by setting bit 2 of the CRG register (0x200300CC), and then set the initial value to **0xE**, see the row in red in the following table.

[3]	RW	mii_rmii_mode	ETH mode. 0: MII mode 1: RMII mode
[2]	RW	eth_rmick_sel	ETH RMII clock source select. 0: internal CRG clock 1: PAD input clock



Module	pll							
Base Address	0x20030000	Add module	Add register					
Priority	1							
Whether Operations Required for Resuming from the Standby Mode	N							
Whether Operations Required for Normal Booting	Y							
Register	Offset Address	Written Value or Value Compared with the Read Value	Delay	Read or Write	Number of Bits to Be Read or Written	Start Bit to Be Read or Written	Read or Write Attribute	
PERI_CRG0	0x0	0x11000000	0	Write	31	0	0x000000FD	
PERI_CRG1	0x4	0x0068306E	0	Write	31	0	0x000000FD	440 MHz
PERI_CRG2	0x8	0x12000000	0	Write	31	0	0x000000FD	
PERI_CRG3	0xc	0x007C2063	0	Write	31	0	0x000000FD	
PERI_CRG4	0x10	0x11000000	0	Write	31	0	0x000000FD	
PERI_CRG5	0x14	0x00683064	0	Write	31	0	0x000000FD	400 MHz
PERI_CRG8	0x20	0x1B000000	0	Write	31	0	0x000000FD	
PERI_CRG9	0x24	0x007C40E1	0	Write	31	0	0x000000FD	
PERI_CRG10	0x28	0x00000010	0	Write	31	0	0x000000FD	
PERI_CRG58	0xe8	0xf	0	Read	3	0	0x001D0000	
PERI_CRG51	0xcc	0x0000000e	0	Write	31	0	0x000000FD	RMII

- Set the multiplexing function of the MII_TXCK pin to RMII_CLK by setting the pin multiplexing register (0x200F005C), and set the initial value to **0x3**, see the row in red in the following table.

[1:0]	RW	muxctrl_reg23	Multiplexing for the MII_TXCK pin. 00: GPIO3_3 01: MII_TXCK 10: VOU1120_DATA7 11: RMII_CLK
-------	----	---------------	--

Module	standby_pin_ctrl							
Base Address	0x200F0000	Add module	Add register					
Priority	5							
Whether Operations Required for Resuming from the Standby Mode	N							
Whether Operations Required for Normal Booting	Y							
Register	Offset Address	Written Value or Value Compared with the Read Value	Delay	Read or Write	Number of Bits to Be Read or Written	Start Bit to Be Read or Written	Read or Write Attribute	
muxctrl_reg12	0x30	0x01	0	Write	1	0	0x0000000D	
muxctrl_reg13	0x34	0x01	0	Write	1	0	0x0000000D	
muxctrl_reg14	0x38	0x01	0	Write	1	0	0x0000000D	
muxctrl_reg15	0x3c	0x01	0	Write	1	0	0x0000000D	
muxctrl_reg16	0x40	0x01	0	Write	1	0	0x0000000D	
muxctrl_reg17	0x44	0x01	0	Write	1	0	0x0000000D	
muxctrl_reg18	0x48	0x01	0	Write	1	0	0x0000000D	
muxctrl_reg19	0x4c	0x01	0	Write	1	0	0x0000000D	
muxctrl_reg20	0x50	0x01	0	Write	1	0	0x0000000D	
muxctrl_reg21	0x54	0x01	0	Write	1	0	0x0000000D	
muxctrl_reg22	0x58	0x01	0	Write	1	0	0x0000000D	
muxctrl_reg23	0x5c	0x03	0	Write	1	0	0x0000000D	RMII_CLK

Step 2 Set the mode to RMII mode in the U-boot command line.

- Create the U-boot image by using the U-boot table described in step 1.
- Enter the following command after the board boots under the U-boot:
setenv mdio_intf rmii
- Run **saveenv** to save settings. The mode is set to the RMII mode, see the last row in the following figure.



```
hisilicon # pri
bootdelay=1
baudrate=115200
ethaddr=00:00:23:34:45:66
netmask=255.255.254.0
bootfile="uImage"
bootargs=mem=256M console=t
2M(rootfs),16M(test)
serverip=192.168.0.234
ipaddr=192.168.0.123
bootcmd=nand read 0x8200000
stdin=serial
stdout=serial
stderr=serial
verify=n
ver=U-Boot 2010.06 (Feb 13
mdio_intf=rmi
```

Step 3 Open **menuconfig**, and set the mode to RMI mode by configuring the options in the following figure. Then the kernel image **uImage** in RMI mode is created.

```
Device Drivers --->
[*] Network device support --->
[*] Ethernet (10 or 100Mbit) --->
<*> hieth(switch fabric) family network device support --->
(1) hieth mii/rmii mode for up port (mii:0/rmii:1)
(1) hieth mii/rmii mode for down port (mii:0/rmii:1)
```

----End

3.13 What Do I Do If Some USB Flash Drives Are Incompatible Under the U-boot?

Currently, the USB 2.0 open host controller interface (OHCI) protocol and USB 3.0 eXtensible host controller interface (XHCI) protocol are supported under the U-boot. When the U-boot is being initialized, some USB 3.0 flash drives can be identified over the USB 2.0 port but not the USB 3.0 port.

The USB flash drive compatibility test under the U-boot shows that the time during which some USB flash drives are powered on and then identified by the controller is different. In the U-boot code, the current value of the **CONFIG_USB_HUB_MIN_POWER_ON_DELAY** parameter is **3000** ms, which is supported by most USB flash drives. However, some USB flash drives (such as Kingston DTG3, Kingston DataTraveler 111, EAGET F30, Apacer, and SSK SFD201) cannot be identified over the USB 3.0 port when **CONFIG_USB_HUB_MIN_POWER_ON_DELAY** is **3000** ms.



When the **CONFIG_USB_HUB_MIN_POWER_ON_DELAY** parameter is set to **1000** ms, the preceding USB flash drives can be identified, but other USB flash drives cannot be identified.

Therefore, when USB flash drives cannot be identified, change the value of **CONFIG_USB_HUB_MIN_POWER_ON_DELAY** in **common/usb.c** as required.

3.14 What Do I Do If the IP Address Is Null When the netstat Command is Executed to View the Link State?

If the glibc system is used, the IP address may be null when the **netstat** command is executed to view the link state. When the **netstat** command is executed, the **getnameinfo** interface of the **libc** is called to obtain the IP address of the link. As the implementation of the **getnameinfo** interface differs according to the glibc system and uClibc system, null is returned when the **getnameinfo** interface of the glibc system is called by running the **netstat** command. The IP address is normally displayed if the **-n** parameter is added at the end of the **netstat** command. You can also solve this issue by modifying the code of the **ip_port_str()** function in **netstat.c** under the **networking** directory of the BusyBox as follows:

Before modification

```
host = numeric ? xmalloc_sockaddr2dotted_noport(addr)
                : xmalloc_sockaddr2host_noport(addr);
```

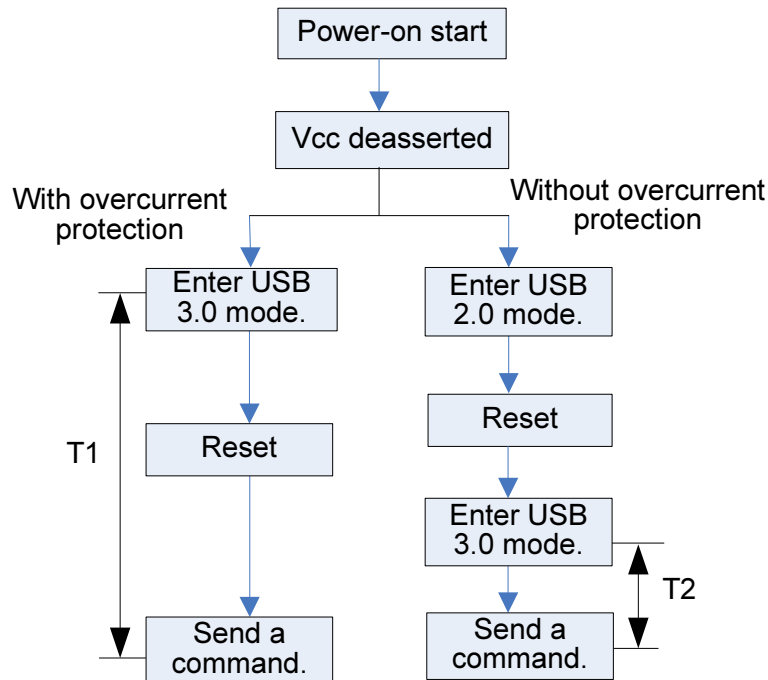
After modification

```
host = NULL;
if (!numeric)
    host = xmalloc_sockaddr2host_noport(addr);
if (!host)
    host = xmalloc_sockaddr2dotted_noport(addr);
```

3.15 What Do I Do If Some USB 3.0 Flash Drives Fail to Be Identified During Power-on Start After the Overcurrent Protection Chip of the USB 3.0 Port Is Removed?

If the overcurrent protection chip of the USB 3.0 port is removed, one Teclast USB 3.0 flash drive fails to be identified after it is inserted and powered on. The cause is that the period during which the controller stays in USB 3.0 mode is too short and therefore the USB 3.0 flash drive is not initialized. See [Figure 3-3](#).

Figure 3-3 Start and identification process of the USB 3.0 flash drive



After the overcurrent protection chip is removed from the USB 3.0 controller, the controller enters USB 2.0 mode first, and then enters USB 3.0 mode after reset. In this way, the time (T2) during which the controller is in USB 3.0 mode is much less than the time (T1) used by the controller to directly enter USB 3.0 mode when the overcurrent protection chip is used. As a result, the USB flash drive is not ready and the host fails to send the command. To solve this issue, increase the value of **HUB_ROOT_RESET_TIME** to **100** in **hub.c** under **drivers/usb/core**.

```
#define HUB_ROOT_RESET_TIME    100;
```

3.16 What Precautions Should I Take During the Atomic Operation of the Kernel-Mode I²C Interface?

The kernel-mode interface functions (`i2c_master_send`, `i2c_master_recv`, and `i2c_transfer`) that are used in the I²C driver internally apply for various locks based on atomic operation or non-atomic operation, as shown in [Figure 3-4](#).

Figure 3-4 Applying for lock in the kernel-mode interface

```

if (in_atomic() || irqs_disabled()) {
    ret = i2c_trylock_adapter(adap);
    if (!ret)
        /* I2C activity is ongoing. */
        return -EAGAIN;
} else {
    i2c_lock_adapter(adap);
}
  
```

If the atomic lock is used before the I²C interface function is called or the I²C interface function is called in the interrupt, the current operation is in the atomic operation. In this case, the if branch in the upper part of [Figure 3-4](#) will be run; otherwise, the else branch in the lower part will be run.



CAUTION

In the atomic operation, the lock is applied for by using `i2c_trylock_adapter(adap)`. If error occurs and `-EAGAIN` is returned, the lock is not obtained and the I²C communication is normal. In this case, read and write operations are not performed and values are not written and do not need to be read. Therefore, check whether the returned value is equal to `-EAGAIN`. If yes, call the interface function again as required. [Figure 3-5](#) describes the write operation over the I²C interface.

Figure 3-5 Write operation over the I²C interface

```
/*
 * i2c_write_foo() - write i2c sensor register sample
 * @client: i2c slave
 * @reg: sensor register address
 * data: sensor register data to written
 *
 * Before the call, you need to do the following,
 * and i2c_lock is global in this file.
 *     spinlock_t i2c_lock;
 *     spin_lock_init(&i2c_lock);
 *
 * Return: zero on success, else a error code.
 */
int i2c_write_foo(struct i2c_client* client, unsigned char reg,
                 unsigned char data)
{
    int ret;
    unsigned char buf[2];
    unsigned long flags;

    buf[0] = reg;
    buf[1] = data;
    do {
        spin_lock_irqsave(&i2c_lock, flags);
        ret = i2c_master_send(client, buf, 2);
        spin_unlock_irqrestore(&i2c_lock, flags);
        if (ret == 2) {
            break;
        } else if (ret == -EAGAIN) {
            continue;
        } else {
            printk("[%s %d]i2c_master_send error, ret = %d",
                  __func__, __LINE__, ret);
            return ret;
        }
    } while (1);

    return 0;
}
```



3.17 What Do I Do If the Module Firmware Fails to Be Loaded When the USB 2.0 Port Connects to the BCM Wi-Fi Module?

When the Hi3518E V200 series chips (including Hi3518E V200, Hi3518E V201, and Hi3516C V200) connect to the BCM43143 Wi-Fi module, the firmware fails to be loaded (this issue has not occurred yet when the Wi-Fi module of any other brands is connected). This module requires one simulated disconnection. If the module firmware fails to be disconnected within the specified count value (configurable), the driver fails to be loaded. The following log is displayed:

```
~ # ./bcm43143 -n nvram_wubb-738gn.nvm brcm43143.bin.trx -C 10
version: 0.2
argv=-n
nvfn=nvram_wubb-738gn.nvm
argv=brcm43143.bin.trx
fwfn=brcm43143.bin.trx
argv=-C
cnt=10
Vendor 0xa5c ID 0xbd1e
claiming interface 0
Found device: vend=0xa5c prod=0xbd1e
ID : Chip 0xa887 Rev 0x2 RamSize 458752 RemapBase 0x60000000 BoardType 0
BoardRev 0
Final fw_path=brcm43143.bin.trx
Final nv_path=nvram_wubb-738gn.nvm
File Length: 358596
start
rdl.state 0x4
elapsed download time 0.214301
Vendor 0xa5c ID 0xbd1e
Vendor 0x1d6b ID 0x2
Vendor 0x1d6b ID 0x1
No devices found
Error: usbdev_find ... cnt=0
Vendor 0xa5c ID 0xbd1e
Vendor 0x1d6b ID 0x2
Vendor 0x1d6b ID 0x1
No devices found
Error: usbdev_find ... cnt=1
Vendor 0xa5c ID 0xbd1e
Vendor 0x1d6b ID 0x2
Vendor 0x1d6b ID 0x1
No devices found
Error: usbdev_find ... cnt=2
```



```
Vendor 0xa5c ID 0xbd1e
Vendor 0x1d6b ID 0x2
Vendor 0x1d6b ID 0x1
No devices found
Error: usbdev_find ... cnt=3
Vendor 0xa5c ID 0xbd1e
Vendor 0x1d6b ID 0x2
Vendor 0x1d6b ID 0x1
No devices found
Error: usbdev_find ... cnt=4
Vendor 0xa5c ID 0xbd1e
Vendor 0x1d6b ID 0x2
Vendor 0x1d6b ID 0x1
No devices found
Error: usbdev_find ... cnt=5
Vendor 0xa5c ID 0xbd1e
Vendor 0x1d6b ID 0x2
Vendor 0x1d6b ID 0x1
No devices found
Error: usbdev_find ... cnt=6
Vendor 0xa5c ID 0xbd1e
Vendor 0x1d6b ID 0x2
Vendor 0x1d6b ID 0x1
No devices found
Error: usbdev_find ... cnt=7
Vendor 0xa5c ID 0xbd1e
Vendor 0x1d6b ID 0x2
Vendor 0x1d6b ID 0x1
No devices found
Error: usbdev_find ... cnt=8
Vendor 0xa5c ID 0xbd1e
Vendor 0x1d6b ID 0x2
Vendor 0x1d6b ID 0x1
No devices found
Error: usbdev_find ... cnt=9
Vendor 0xa5c ID 0xbd1e
Vendor 0x1d6b ID 0x2
Vendor 0x1d6b ID 0x1
No devices found
Error: usbdev_find ... cnt=10
Run out of cnt
Error: can not find bdc device
```

[Solution]



The recommendations are as follows:

1. Enable the EOP pre-emphasis function.
2. Add a judgment process to the BCM43143 USB 2.0 Wi-Fi firmware. When the firmware fails to be loaded, automatically adapt to the BCM Wi-Fi by dynamically decreasing the threshold for one or multiple times (four levels are recommended: 600 mV, 575 mV, 550 mV, and 525 mV) to ensure that the firmware is successfully loaded before timeout occurs.
3. If the power consumption is not specially considered, adopt the fixed board voltage for power supply instead of the SVB solution. For details about the voltage, see the design requirements in the chip data sheet.
4. The methods of switching the threshold and enabling/disabling the EOP pre-emphasis are as follows:

```
//Enable EOP pre-emphasis.
```

```
himm 0x20120080 0x1900
```

```
himm 0x20120080 0x1920
```

```
//Disable EOP pre-emphasis.
```

```
himm 0x20120080 0x1c00
```

```
himm 0x20120080 0x1c20
```

```
//disconnect threshold = 650mV
```

```
himm 0x20120080 0x1b0a
```

```
himm 0x20120080 0x1b2a
```

```
//disconnect threshold = 625mV
```

```
himm 0x20120080 0x190a
```

```
himm 0x20120080 0x192a
```

```
//disconnect threshold = 612mV
```

```
himm 0x20120080 0x1d0a
```

```
himm 0x20120080 0x1d2a
```

```
//disconnect threshold = 600mV
```

```
himm 0x20120080 0x110a
```

```
himm 0x20120080 0x112a
```

```
//disconnect threshold = 587mV
```

```
himm 0x20120080 0x150a
```

```
himm 0x20120080 0x152a
```

```
//disconnect threshold = 575mV
```

```
himm 0x20120080 0x1f0a
```

```
himm 0x20120080 0x1f2a
```

```
//disconnect threshold = 562mV
```



```
himm 0x20120080 0x050a
himm 0x20120080 0x052a

//disconnect threshold = 550mV
himm 0x20120080 0x170a
himm 0x20120080 0x172a

//disconnect threshold = 537mV
himm 0x20120080 0x0d0a
himm 0x20120080 0x0d2a

//disconnect threshold = 525mV
himm 0x20120080 0x070a
himm 0x20120080 0x072a

//disconnect threshold = 500mV
himm 0x20120080 0x0f0a
himm 0x20120080 0x0f2a
```

This solution increases the delay caused by Wi-Fi connection. Therefore, select the level and count value as required.



4 PCIe

4.1 How Do I Compile the PCIe Controller Driver into the Kernel by Configuring Kernel Options?

In root complex (RC) mode, the kernel must run the peripheral component interconnect express (PCIe) controller driver after starting to initialize the controller and enumerate PCIe endpoint (EP) devices. In EP mode, the kernel does not run the PCIe controller driver. The logic configures the EP mode by default. In RC mode, run **make ARCH=arm CROSS_COMPILE=arm-hisiv100-linux- menuconfig**, and choose the following options on the **menuconfig** menu:

```
Bus Support --->
    [*] PCI support --->
    [*] PCI Express support --->
Bus Support --->
    [*] PCI support --->
    [*] Hisilicon PCI Express support --->
```

In EP mode, you are advised to deselect **PCI support**.

```
Bus Support --->
    [] PCI support --->
```

In EP mode, note that the **PCI support** option must be deselected, and the board cannot connect to other device over the PCIe interface.

4.2 How Do I Configure the PCIe Control Clock?

The working clocks of the PCIe module are derived from the chip (called internal clocks) or peripherals (called external clocks). Typically, when the PCIe module works in RC mode, internal clocks are used. To use external clocks, the clock signal must connect to a dedicated clock source. When the PCIe module works in EP mode, the output clock of the master device is used.

The master device (RC) clock is controlled by bit 0 of the PCIe0 PHY control register (base address 0x200500B4). Clock control must be complete before the PCIe system controller is enabled.



The following uses the Hi3531 as an example.

- Run the following command to view the state of the current clock:
`himd.l 0x200500B4`
- Run the following command to switch the PCIe working clock to external clock:
`himd.l 0x200500B4 0x05605000`
- Run the following command to switch the PCIe working clock to internal clock:
`himd.l 0x200500B4 0x05605001`

When the PCIe module works in RC mode, the clock must be output for the slave device. The slave device can be identified only when bit 7 and bit 8 of PERI_CRG30 (base address 0x20030078) are 0. It is not recommended that the clocks of the master device and slave device be derived from different sources unless the clock of the slave device is not provided by the master device.

4.3 How Do I View the Allocation Information About the BAR Address for the PCIe Device?

The BAR address for the PCIe device is allocated when the system starts. The allocation information is stored in the PCIe configuration space. The following uses the Hi3531 as an example. In the configuration space, the offset address 0x10, 0x14, and 0x18 store the address information about BAR0, BAR1, and BAR2 respectively. BAR0 is 8 MB and prefetchable. BAR1 is 64 KB and BAR2 is 1 MB. Both BAR1 and BAR2 are not prefetchable.

For the first device connected to PCIe controller 0:

```
himd.l 0x40100000
0000: 353219e5 00100140 04800001 00000008
0010: 30800008 31500000 31400000 37800001
0020: 00000000 00000000 00000000 00000000
0030: 00000000 00000040 00000000 00000149
0040: 5bc35001 00000008 00000000 00000000
0050: 00807005 00000000 00000000 00000000
0060: 00000000 00000000 00000000 00000000
0070: 00020010 00008702 00002010 00423c11
```

The preceding message in blue is the BAR address for the slave device.

For the first device connected to PCIe controller 1, the base address for the configuration space is 0x70300000.

4.4 How Do I View the Mapping Information About PCIe Addresses?

The mapping information about PCIe addresses is stored in IATU register groups in the configuration space. There are six IATU register groups. Each group supports both input and output. The register group and register direction are selected by configuring the Viewport



register. The offset of the PCIe configuration space is 0x900. The following describes how to use register groups by using the Hi3531 as an example.

Run the following command to select IATU register group 0 and set the register direction to input (from the slave device to the master device):

```
himm 0x40100900 0x0
```

Run the following command to select IATU register group 1 and set the register direction to output (from the master device to the slave device):

```
himm 0x40100900 0x800000001
```

To view the address mapping information about the selected register group, run the following command:

```
himd.l 0x40100900
0000: 00000000 00000000 00000000 00000000
0010: 00000000 0000ffff 00000000 00000000
0020: 00000000 00000000 00000000 00000000
0030: 00000000 00000000 00000000 00000000
0040: 00000000 00000000 00000000 00000000
0050: 00000000 00000000 00000000 00000000
0060: 00000000 00000000 00000000 00000000
```

The preceding information shows that IATU register group 0 is not configured in input mode.

4.5 Why Does Not the PCIe MCC Drivers Take Effect After Being Loaded?

Note the following:

- The **MCC** .ko drivers for the master end and slave end must be separately compiled. The PCIe controller driver is required for compiling the **MCC** .ko driver for the master end. Ensure that the PCIe controller driver is compiled into the kernel (the PCIe compilation option is selected on the **menuconfig** menu), and the kernel is compiled. There are no such requirements for compiling the **MCC** .ko drivers for the slave end.
- The **MCC** .ko drivers for the master end can run only in the image of the kernel into which the PCIe controller driver has been compiled.

If any of the preceding requirements is not met, the **MCC** .ko drivers for the master do not take effect.

4.6 What Precautions Should I Take While Using the PCIe Network Adapter and PCIe-SATA Device?

The following uses the Hi3531 as an example. Before using the PCIe network adapter and PCIe-SATA device, ensure that the related configuration options are selected.

PCIe network adapter:



```
Device Drivers --->
[*] Network device support --->
[*] Ethernet (1000 Mbit) --->
<*> SysKonnect Yukon2 support
```

PCIe-SATA:

For Silicon Image 3124/3132:

```
Device Drivers --->
-*- Serial ATA and Parallel ATA drivers --->
<*> Silicon Image 3124/3132 SATA support --->
```

For JMB 362:

```
Device Drivers --->
-*- Serial ATA and Parallel ATA drivers --->
<*> AHCI SATA support --->
```

4.7 Why Does the Slave Booter Fail to Boot the Slave Device over the PCIe Interface?

Note the following:

- When compiling the kernel of the slave chip, ensure that the following option on the **menuconfig** menu is selected:

```
General setup --->
[*] Initial RAM filesystem and RAM disk
(initramfs/initrd) support
```

- To support the cramfs file system greater than 4 MB, change the macro definition to **CONFIG_BLK_DEV_RAM_SIZE=65536** in the .config file.
- In the current cascade scenario, the file loaded to the slave chip must be less than or equal to 7 MB.
- To boot the slave device by using the slave booter provided in the SDK, set the U-boot environment variables as follows:

```
setenv bootargs 'mem=64M console=ttyAMA0,115200'
setenv bootcmd 'bootm 0x81000000 0x82000000'
```

4.8 Why Is the Message "Unknown Hi-irq Triggered" Displayed Sometimes After the SDK Video Preview Service Runs?

This message is only a reminder but not an error.

If a chip is cascaded with another one, the message communication between them is as follows:



The slave end writes a message and triggers an interrupt for the master end. The master end responds to the interrupt and obtains the message from the shared memory. The master chip may send a response message to the slave chip in the same way. That is, the master chip writes the response message to the shared memory and triggers an interrupt for the slave chip. The slave chip enters the interrupt service routine (ISR) to handle the message in the shared memory.

Typically, the local end checks the interrupt status of the peer end before triggering an interrupt for the peer end. If the interrupt status is not cleared, the local end triggers an interrupt only after the previous interrupt of the peer end is handled. This ensures that a message corresponds to an interrupt. However, the peer end frequently enters the ISR, which affects the message interaction efficiency. You are advised to handle the messages that do not have high real-time requirements at a time by using the timer to improve the efficiency. After sending a message, the local end can trigger an interrupt even though the interrupt status of the peer end is not cleared. The following situation may occur: After the local end writes an interrupt for the peer end after sending a message, if the interrupt is not triggered and the previous interrupt for the peer end is being handled, the message is also handled. In this case, the interrupt status of the peer end is cleared. When the local end detects the interrupt status for triggering the interrupt for the peer end, the local end cannot find the interrupt status flag. Then the message "unknown Hi-irq triggered" is displayed.

A comprehensive analysis shows that this problem does not cause message loss or other exceptions.

4.9 Which Address Will Be Mapped to the Hi3531 or Hi3532 PCIe BAR Address by Default During Reset? What Precautions Should I Take When PCIe Window Is Moved?

After the system is reset, the address mapping function of the Hi3531 or Hi3532 is disabled. That is, the window is not mapped to the address space of the slave device. The address mapping function is enabled only after the window is configured.

If the PCIe window is moved, the window addresses must be aligned.

The Hi3531 or Hi3532 PCIe controller has three BAR addresses whose sizes are 8 MB, 64 KB, and 1 MB respectively. If you use the window whose BAR address is 8 MB to map a region at the slave end, the start address for the region must be an integral multiple of 8 MB. For example, you can map the window to the address 0x84200000 but not 0x84270000 at the slave end, because 0x84270000 is not aligned by 8 MB. To use the address 0x84270000, map the window start address to 0x84200000 and access 0x84270000 after 0x70000 offset.

For the window whose BAR address is 1 MB, the start address must be aligned by 1 MB. For the window whose BAR address is 64 KB, the start address must be aligned by 64 KB.

4.10 What Are the Changes When All DMA Operations Are Initiated by the Slave Chip?

To enable a Hi3531 to cascade with another Hi3531, the PCIe MCC drivers do not allow the master chip to write data to the slave chip in direct memory access (DMA) mode. All DMA operations are initiated by the slave chip. The original DMA write operation from the master



chip to the slave chip is replaced with the DMA read operation from the slave chip to the master chip.

The SDK earlier than SPC070 supports DMA write operations between the master chip and the slave chip. The tests for a long period of time show that the DMA operations and non-DMA data transfer between the master chip and the slave chip may cause unexpected results. Therefore, the SDK SPC070 does not allow the master chip write data to the slave chip in DMA mode. Such DMA write operation is replaced with the DMA read operation from the slave chip to the master chip. The system runs stably when the DMA read/write operations from the slave chip to the master chip and non-DMA data transfer are performed, which is tested on a demo board for a long period of time.

The DMA read/write operations from the slave chip to the master chip are managed by using two task linked lists (only one task linked list is provided before). In this way, the PCIe TX and RX channels can work at the same time. The PCIe channel utilization when the slave chip initiates DMA write operations is the same as that when the master chip and slave chip initiate DMA write operations at the same time.

4.11 Why Does the Slave Chip Stop Working If It Uses the .ko Drivers Compiled in the SDK PCIe MCC When Two Hi3531s Are Cascaded?

In the SDK, the Hi3531 acts as the master chip and the Hi3532 acts as the slave chip by default. See the configurations in **config.h** in **osdrv/drv/pcie/hi35xx_dev/slave/**:

```
//#define CONFIG_GODNET 1 /* Hi3531 */
#define CONFIG_GODCUBE 1 /* Hi3532 */

#define PCIE_SLAVE_CONNECTOR0 1
//#define PCIE_SLAVE_CONNECTOR1 1
Note: The macro CONFIG_GODXXX specifies the slave chip. The Hi3532 acts as
the slave chip by default (CONFIG_GODCUBE).
```

If the Hi3531 acts the slave chip, modify **config.h** in **osdrv/drv/pcie/hi35xx_dev/slave/** as follows:

```
#define CONFIG_GODNET 1
//#define CONFIG_GODCUBE 1

//Configure PCIE_SLAVE_CONNECTORX as required:
//#define PCIE_SLAVE_CONNECTOR0 1 /*Controller 0*/
#define PCIE_SLAVE_CONNECTOR1 1 /*Controller 1*/
```

Disable **CONFIG_GODCUBE** and enable **CONFIG_GODNET (Hi3531)**. The Hi3531 has two PCIe controllers. You can choose one to connect it to the master chip, which is determined by hardware. The actual configuration determines the controller used by the slave chip. **PCIE_SLAVE_CONNECTOR0** indicates that the slave chip Hi3531 uses controller 0, and **PCIE_SLAVE_CONNECTOR1** indicates that the slave chip Hi3531 uses controller 1.

The manual configuration is required because software cannot know the controller used by the slave chip Hi3531. That is, software cannot identify the valid address space.

4.12 What Precautions Should I Take When the Hi3532 Acts as a Plug-in Card?

- Problem

The Windows OSs earlier than Windows 7 do not support access to the PCIe extended configuration space. The key functions of the Hi3532 PCIe such as the DMA operations and window mapping are implemented only when the PCIe extended configuration space is accessible. PCIe window mapping is disabled by default after the Hi3532 is reset. The PCIe device (Hi3532) supports address mapping and DMA operations only when PCIe window mapping is available. How to access the PCIe extended configuration space on all Windows OSs?

- Solution

When the PC is cascaded with the Hi3532, map the address BAR2 of the three BAR address of the Hi3532 to the Hi3532 PCIe configuration space (4 KB) by using the Hi3532 U-boot. The PC can access the entire PCIe configuration space by using the address BAR2.

Before the PC performs any operation on the Hi3532, the Hi3532 must map the address BAR2 to its PCIe configuration space. This function requires that a small-capacity flash memory (such as 2 MB) is added to the Hi3532 for storing the U-boot. The U-boot is used to map the address BAR2 to the Hi3532 PCIe configuration space. After the PC and Hi3532 work, the PC can access the entire Hi3532 PCIe configuration access by using the address BAR2

- Modification details

For details about the modifications to source code of the U-boot running on the Hi3532 SPI flash, see **start.S** in **arch/arm/cpu/godcube/**.



start.S

Perform the following steps after the U-boot starts but before the DDR is initialized:

- Step 1** Disable the Hi3532 PCIe controller.
- Step 2** Use the internal clock as the Hi3532 working clock.
- Step 3** Map the address BAR2 to the Hi3532 configuration space.
- Step 4** Use the external clock as the Hi3532 working clock.
- Step 5** Enable the Hi3532 PCIe controller.
- Step 6** Wait and check the register whose base address is 0x20030028. If the PC changes the register value from 0x11 to 0x1, skip DDR initialization (DDR initialization is performed by the PC) and start the U-boot. Otherwise, continue to check the register value.

For the PCIe MCC driver running on the PC, all configuration operations related to PCIe window mapping are performed by using the Hi3532 address BAR2. The code is stored in **hi35xx_dev_host.c** in **~/drv/pcie_mcc/hi35xx_dev/host/**.

`hi35xx_dev_host.c`

----End

- Others
 - Map the Hi3532 address BAR2 only to the Hi3532 PCIe configuration space (4 KB).
 - Start the Hi3532 from the SPI flash.
 - Retain other operations.

4.13 Does the PCIe MCC Driver Allow the Master Device to Reset the Slave Device?

The PCIe MCC driver allows the master device to reset the slave device. Note the following:

- The previous states such as the device function states and address mapping state (excluding the states of DMA-related registers) are retained after the slave device is reset.
- The master device can continuously reset the slave device. Reserve sufficient time for starting the slave device during reset.

After the driver for the master chip is compiled, the executable file **booter** is generated in **drv/pcie_mcc/koes**. This file is a sample for starting and resetting the slave device, and is compiled by using the code in **drv/pcie_mcc/multi-boot/example**. The following describes the usage:

To start the slave device, run the following command:

```
$/booter start_device
```

To reset the slave device, run the following command:

```
$/booter reset_device
```

For details, read **boot_test.c** in **~/pcie_mcc/multi_boot/example/** and driver code.

Note that only the SDK SPC70 allows the master device to reset the slave device.

4.14 What Do I Do If the VO Bandwidth Is Low When Some PCIe-to-SATA Cards (Such as marvel9215) Connect to the SATA Disk to Read/Write Data (Taking the Hi3536 as an Example)?

- Cause:

The default size of requested data of the PCIe-to-SATA card is 512 bytes, which results in high PCIe bandwidth usage.



- Solution:

The Max_Read_Request_Size register of the PCIe-to-SATA card can be modified to limit the maximum size of the requested read data to 256 bytes and reduce the PCIe bandwidth usage.

Perform the following steps:

Step 1 Run `cp arch/arm/configs/hi3536_full_defconfig .config`.

Step 2 Run `make ARCH=arm CROSS_COMPILE=arm-hisiv300-linux- menuconfig`.

Step 3 Configure the option **Bus support** ---> **Hisilicon PCI Express support**---> **PCI Express configs**---> **limit pcie max read request size** on the menconfig GUI, as shown in [Figure 4-1](#).

Figure 4-1 menconfig GUI

```
[*] Patch physical to virtual translations at runtime
    General setup --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
    System Type --->
    Bus support --->
        Kernel Features --->
        Boot options --->
        CPU Power Management --->
        Floating point emulation --->
        Userspace binary formats --->
        Power management options --->
    [*] Networking support --->
        Device Drivers --->
        File systems --->
        Kernel hacking --->
        Security options --->
    -* Cryptographic API --->
        Library routines --->
    [ ] Virtualization --->
```



```
[*] PCI support
[ ] PCI Debugging
[ ] Enable PCI resource re-allocation detection
< > PCI Stub driver
[ ] PCI IOV support
[ ] PCI PRI support
[ ] PCI PASID support
[*] PCI Express support
[*]   Root Port Advanced Error Reporting support
[ ]   PCI Express ECRC settings control
< >   PCIe AER error injector support
[*]   PCI Express ASPM control
[ ]   Debug PCI Express ASPM
      Default ASPM policy (BIOS default) --->
< > Support for PCI Hotplug --->
< > PCCard (PCMCIA/CardBus) support --->
[*] Hisilicon PCI Express support --->
```

```
--- Hisilicon PCI Express support
[*] PCI Express configs --->
```

```
[*] PCI Express controller 0 sel
(0x10000000) Total memory size of PCI Express EP devices
(0x800000) Sum of configuration header size mapped for all PCIe EP devices
[*] limit pcie max read request size
```

Step 4 Save the settings and exit.

Step 5 Run **make ARCH=arm CROSS_COMPILE=arm-hisiv300-linux- uImage -j 128**.

Step 6 Store the compiled kernel image in **arch/arm/boot/uImage**.

----End



5 Flash

5.1 What Do I Do When the Configured ECC Mode for the Hi3531 NAND Flash Does Not Take Effect?

The Hi3531 NAND flash driver allows you to view the ECC mode in either of the following ways:

- Configure the ECC mode by using hardware, and then read it.
- Automatically check the NAND flash type and select the optimal ECC mode.

If the ECC mode is configured by using hardware, the message "Nand(Hardware)" is displayed during startup, as shown in [Figure 5-1](#).

Figure 5-1 Hardware configuration

```
NAND: Special nand id table version 1.21  
Nand ID: 0xAD 0xDC 0x10 0x95 0x54 0xAD 0xDC 0x10  
Nand(Hardware): Block:128K Page:2K Ecc:1bit Chip:512M 00B:64Byte  
512 MiB
```

If the ECC mode is automatically selected, the message "Nand(Auto)" is displayed.

If the ECC mode is automatically selected, the board cannot be booted from the NAND flash.

5.2 How Do I Mark the Bad Blocks in a NAND Flash?

In the SDK, the read and write functions for the NAND flash have the policy of processing bad blocks by default. You can ignore them. The following methods are provided to forcibly mark some bad blocks for tests. Typically, the methods are not used.

- Marking bad blocks under u-boot-2010.06

To mark bad blocks, run the following command:

```
nand markbad offset
```

The preceding command is used to mark the bad blocks that are located in the position specified by **offset**. For example, to mark the bad blocks that are located in the 1 MB position, run the following command:



```
nand markbad 0x100000
```

The value specified by **offset** should be an integral multiple of the NAND flash block size. To view marked NAND bad blocks, run the following command:

```
nand bad
```

- Marking bad blocks under the kernel

The related code is as follows:

```
#define MEMSETBADBLOCK      _IOW('M', 12, __kernel_loff_t)
int fd;
unsigned long long offset;
fd = open("/dev/mtd1", O_RDWR);
offset = 0x100000;
if (ioctl(fd, MEMSETBADBLOCK, &offset))
{
    printf("Mark bad block 0x%llx failed!\n", offset);
}
```

The preceding program marks the NAND block in the position where the offset value is **offset** in the **mtd1** partition as a bad block.

To mark the block with 1 MB offset in the mtd1 partition as bad blocks, specify mtd1 partition (byte device node of the specific partition) and set **offset** to **0x10000** when the open function is called, as shown in the preceding code.

Note that the value specified by **offset** is the offset position relative to the NAND flash under u-boot-2010.06, and is the offset position relative to partition enabled by calling the open function under the kernel.

5.3 What Are the ECC Modes Supported by the Hi3531 NAND Flash Controller?

Table 5-1 lists the ECC modes supported by the Hi3531 NAND flash controller.

Table 5-1 Page sizes and ECC modes

Page Size	ECC Mode	Support
2 KB	1 bit	Supported
	4 bytes	Supported (UBI or UBIFS)
	8 bytes	Not supported
	24 bits/1 KB	Not supported
4 KB	1 bit	Supported
	4 bytes	Supported
	8 bytes	Not supported
	24 bits/1 KB	Supported



Page Size	ECC Mode	Support
8 KB	1 bit	Not supported
	4 bytes	Not supported
	8 bytes	Not supported
	24 bits/1 KB	Supported

**NOTE**

- The 4-byte ECC mode can be used as 4-bit ECC mode and no exception will occur.
- The 1-bit, 4-byte, and 8-byte ECC modes respectively indicate that at most 1-bit, 4-byte, and 8-byte errors are corrected in 512-byte data (or several bytes more). The 24 bits/1 KB ECC mode indicates that at most 24-bit errors are corrected in 1 KB data.
- The 2 KB page size and 4-byte ECC mode are supported at the same time only for the UBI or UBIFS file system.
- "Not supported" indicates that no such NAND flash exists currently. For example, there is no such NAND flash with 2 KB page size and 24 bit/1 KB ECC mode.

5.4 What Precautions Should I Take While Using a Large-Capacity NAND Flash?

The capacity of a large-capacity NAND flash is greater than 4 GB. 4 GB is the maximum 32-bit unsigned integer. If the capacity is greater than 4 GB, the 32-bit variable is wrapped. The maximum capacity of the tested NAND flash is 8 GB. If a NAND flash whose capacity is greater than 4 GB is used, ensure that the partition size of the file system (UBI or YAFFS2) cannot be greater than 4 GB. Otherwise, the file system is wrapped.

5.5 What Are the Rules for Configuring the ECC Mode and Page Size of a NAND Flash?

The ECC mode and page size of a NAND flash are configured by the logic, hardware and software in the following sequence:

- When the logic is powered on, the logic reads the ECC mode and page size to the NFC_CON register by using the pull-up or pull-down resistor.
- If the system boots from the NAND flash, the logic reads the NAND flash based on the NFC_CON configuration.
- Software initializes the DDR and all the configured registers. If NFC_CON has been configured, software overwrites it.
- After software starts, the driver reads the ECC mode and page size from the NFC_CON register.

5.6 Why Cannot the System Start After the U-boot Saves Environment Variables in the NAND Flash?

The system can start after some users burn U-boot in the NAND flash, but cannot start after the environment variables are saved.

As shown in [Figure 5-2](#), the U-boot occupies three good blocks, and there are three bad blocks (in red) at the start address for the NAND flash. The U-boot content is erased after the environment variables are saved. As a result, the system cannot start. To resolve this problem, move environment variable address backwards.

Figure 5-2 Block positions



5.7 Why Cannot Images Be Combined into One When the NAND Flash Is Burnt?

For details, see section [5.8 "Why Cannot the File System Be Written to the NAND Flash After Being Read from the NAND Flash?"](#)

5.8 Why Cannot the File System Be Written to the NAND Flash After Being Read from the NAND Flash?

The NAND flash can be written only once after it is erased. Before writing data to the NAND flash again, re-erase it.

The data in the NAND flash is 0xFF after it is erased. If you write 0xFF to the NAND flash for the first time and write other data for the second time, an error may occur, because data is written to the NAND flash for several times after it is erased.

The following describes the meanings of "empty" and "non-empty" in [Figure 5-3](#) to [Figure 5-6](#).

- **Empty**: It indicates the status of the NAND flash after it is erased.
- **Non-empty**: It indicates that the NAND flash is written after it is erased.

[Figure 5-3](#) shows the status of the NAND flash after it is erased.

**Figure 5-3** Status of the NAND flash after it is erased

0xFF·0xFF·0xFF·0xFF _⌞	0xFF·0xFF·0xFF·0xFF _⌞	0xFF·0xFF·0xFF·0xFF _⌞
0xFF·0xFF·0xFF·0xFF _⌞	0xFF·0xFF·0xFF·0xFF _⌞	0xFF·0xFF·0xFF·0xFF _⌞
0xFF·0xFF·0xFF·0xFF _⌞	0xFF·0xFF·0xFF·0xFF _⌞	0xFF·0xFF·0xFF·0xFF _⌞
0xFF·0xFF·0xFF·0xFF _⌞	0xFF·0xFF·0xFF·0xFF _⌞	0xFF·0xFF·0xFF·0xFF _⌞
Page 1·Empty·(all 0xFF) _⌞	Page 2·Empty·(all 0xFF) _⌞	Page 3·Empty·(all 0xFF) _⌞

Figure 5-4 shows the data that will be written to the NAND flash.

Figure 5-4 Data that will be written to the NAND flash

0x11·0x22·0xFF·0xFF _⌞	0xFF·0xFF·0xFF·0xFF _⌞	0xFF·0x11·0xFF·0xFF _⌞
0x33·0x44·0xFF·0xFF _⌞	0xFF·0xFF·0xFF·0xFF _⌞	0xFF·0xFF·0x33·0xFF _⌞
0xFF·0x55·0xFF·0xFF _⌞	0xFF·0xFF·0xFF·0xFF _⌞	0xFF·0xFF·0x44·0xFF _⌞
0xFF·0xaa·0xFF·0xFF _⌞	0xFF·0xFF·0xFF·0xFF _⌞	0xFF·0xFF·0xaa·0xFF _⌞
Page 1·Data _⌞	Page 2·Data·(all 0xFF) _⌞	Page 3·Data _⌞

Figure 5-5 shows the status of the NAND flash after the data shown in Figure 5-4 is written to the NAND flash.

Figure 5-5 Status of the NAND flash after the data is written

0x11·0x22·0xFF·0xFF _⌞	0xFF·0xFF·0xFF·0xFF _⌞	0xFF·0x11·0xFF·0xFF _⌞
0x33·0x44·0xFF·0xFF _⌞	0xFF·0xFF·0xFF·0xFF _⌞	0xFF·0xFF·0x33·0xFF _⌞
0xFF·0x55·0xFF·0xFF _⌞	0xFF·0xFF·0xFF·0xFF _⌞	0xFF·0xFF·0x44·0xFF _⌞
0xFF·0xaa·0xFF·0xFF _⌞	0xFF·0xFF·0xFF·0xFF _⌞	0xFF·0xFF·0xaa·0xFF _⌞
Page 1·Non-empty _⌞	Page 2·Non-empty·(all 0xFF) _⌞	Page 3·Non-empty _⌞

Figure 5-6 shows the data in the file system after data is written to the NAND flash.

Figure 5-6 Data in the file system after data is written to the NAND flash

0x11·0x22·0xFF·0xFF _⌞	0xFF·0xFF·0xFF·0xFF _⌞	0xFF·0x11·0xFF·0xFF _⌞
0x33·0x44·0xFF·0xFF _⌞	0xFF·0xFF·0xFF·0xFF _⌞	0xFF·0xFF·0x33·0xFF _⌞
0xFF·0x55·0xFF·0xFF _⌞	0xFF·0xFF·0xFF·0xFF _⌞	0xFF·0xFF·0x44·0xFF _⌞
0xFF·0xaa·0xFF·0xFF _⌞	0xFF·0xFF·0xFF·0xFF _⌞	0xFF·0xFF·0xaa·0xFF _⌞
Page 1·Non-empty _⌞	Page 2·Empty·(all 0xFF) _⌞	Page 3·Non-empty _⌞

Even though you do not write data to the NAND flash, a non-read-only file system also performs write operations such as recycling trash periodically and synchronizing data.



The file system cannot identify page 2. The reason is that the data on page 2 is 0xFF after it is erased (see [Figure 5-3](#)) and the data is written to the NAND flash (see [Figure 5-5](#)). The file system determines that page 2 is empty. If page 2 is written again (0xFF is written to page 2 for the first time), an error occurs.

**NOTE**

- The preceding problems do not occur in read-only file systems such as cramfs and squashfs.
- The kernel data and U-boot data are not overwritten again, and the preceding problems do not occur.

5.9 How Do I Change the Quad-Wire Mode of the SPI Flash to Dual-Wire Mode?

Under the U-boot, search for the ID list of the corresponding component in the **hisfc350_spi_ids.c** file in `~/drivers/mtd/spi/hisfc350`, and disable the quad capability of the component (both the write and read functions are disabled). When the driver detects that the component has no quad capability, the driver does not enable the quad capability, and the component works in dual-wire mode. The change method under the kernel is similar to that under the U-boot.

```
{  
    "MX25L25635E/735E/635F",  
    {0xc2, 0x20, 0x19}, 3, _32M, _64K, 4,  
    {  
        &READ_STD(0, INFINITE, 40/*50*/),  
        &READ_FAST(1, INFINITE, 104),  
        &READ_DUAL(2, INFINITE, 104),  
        &READ_DUAL_ADDR(1, INFINITE, 84),  
        // &READ_QUAD_ADDR(3, INFINITE, 75),  
        0  
    },  
  
    {  
        &WRITE_STD(0, 256, 75),  
        0  
    },  
  
    {  
        &ERASE_SECTOR_64K(0, _64K, 80),  
        0  
    },  
    &spi_driver_mx25l25635e,  
},
```

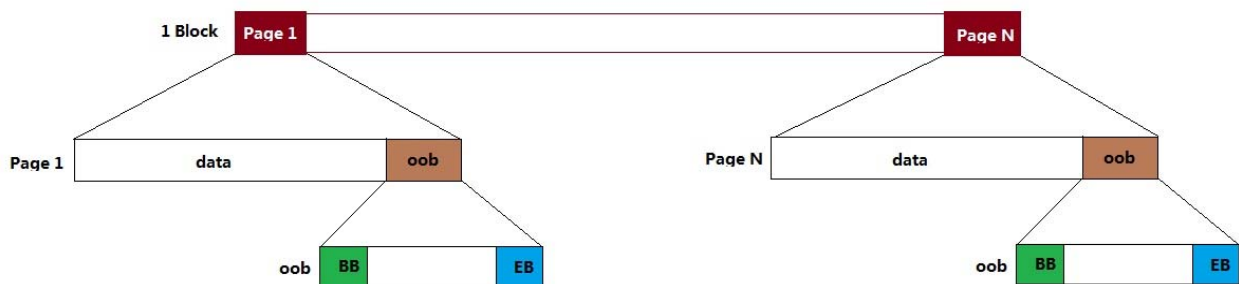
5.10 How Do I Use the nandwrite Naked-Write Tool of mtd-utils?

When the nandwrite naked-write tool of mtd-utils is used, if the u-boot.bin image is written and the image size is greater than the size of a block in the NAND flash, ensure that the data of the u-boot.bin image is aligned and filled by block. Otherwise, the written u-boot.bin image cannot boot normally.

The reason is as follows:

The bad block (BB) flag bits of some NAND flash are marked as non-all-0 values such as 0xFE upon delivery. As a result, during BB judgment, the FMC easily corrects the BB flag bits to 0xFF by using ECC correction (because the ECC algorithm of the controller considers 0xFF to be valid and correctable). Therefore, the empty block (EB) flag bits are set at the last two bytes of the OOB information on each page. As shown in [Figure 5-7](#), when the U-boot is started, the prerequisite for the logic to consider a block as a good block is as follows: BB is 0xFF and EB is 0x00 on page 1 and page N.

Figure 5-7 Structure of blocks in the NAND flash



The nandwrite tool writes data by page based on the size of the image file, and automatically sets the EB flag bits of the current page to 0x00 during page writing. If the last page to be written is not the last page of the block, the logic considers this block to be empty and does not read because the EB flag bits on the last page of the block are 0xFF. As a result, the U-boot fails to be started.

Note that the first block is guaranteed to be a good block during the delivery of the NAND flash, and the logic does not judge the first block. Therefore, if the size of the u-boot.bin image is less than the size of the first block, the U-boot is started normally.

In addition, once the U-boot is started normally, the EB bits are not judged. This is why the image size does not need to be block-aligned when the nandwrite tool is used to write the kernel image and file system image.



6 File System

6.1 What Precautions Should I Take When a NAND Flash Is Mounted to the Cramfs File System?

When the cramfs file system is used, the NAND flash must be mounted to the ROM block but not the MTD block.

Cramfs is not designed for the NAND flash and cannot skip bad blocks. The block device ROM block in the kernel can skip bad blocks.



7

Rapid Startup Optimization

Note the following:

- Set the environment variable *bootdelay* under the boot to **0**.
 - Usage: Enter **setenv bootdelay 0;saveenv** on the boot CLI.
 - Description: The variable *bootdelay* under the boot is set to **1** by default to display the boot CLI easily. If *bootdelay* is set to **0**, the system starts 1s faster than before. The default value of *bootdelay* is changed to **0** in the code.
- Disable kernel check when the boot is running.
 - Usage: Enter **setenv verify n;saveenv** on the boot CLI.
 - Description: If an error occurs in the kernel, the system is suspended no matter whether the kernel is checked. The system is not affected and the startup time is reduced by 1s if you disable kernel check. The kernel is not checked by default, which is implemented by modifying the code under the boot.
- Cancel the BogoMIPS calculation when the kernel boots.
 - Usage: Configure **bootargs** and add **lpj=5996544** to **bootargs**.
 - Description: **BogoMIPS** is used to measure the CPU running speed. You can cancel the calculation by setting **lpj** to **5996544**. This setting reduces the startup time by 0.2s.
- Run the **bootcmd** command.
 - Usage: Run **setenv bootcmd 'nand read 0x807ffc0 0x100000 0x400000;bootm 0x807ffc0'**.
 - Description: After the boot command is configured, the boot reads the kernel image from the flash memory to 0x807FFFC0 and the kernel boots from 0x807FFFC0. If the default command **nand read 0x82000000 0x100000 0x400000;bootm 0x82000000** is executed, the kernel image is read from the flash memory to 0x82000000 when the boot is running, and copied from 0x82000000 to 0x807FFFC0. Then the kernel boots from 0x807FFFC0.



8 Kernel

8.1 What Is the Difference Between PID and TGID Fields?

The data structure `task_struct` has two fields: PID and thread group identification (TGID).

The PID is a unique ID of `task_struct`. The TGID is the ID of the thread group to which a task belongs. If a process has no subprocess, its TGID is the same as the PID. The TGID is used to ensure POSIX standard compatibility.

```
getpid() return tgid
```

In normal processes, the TGID is the same as the PID.

With threads, the TGID is the same for all threads in a thread group. This enables the threads to call `getpid()` and obtain the same PID.

The POSIX 1003.1c standard states that all threads of a multithreaded application must have the same PID.

8.2 What Are the Priorities of Common Processes and Real-Time Processes?

For details, see [Table 8-1](#).

Table 8-1 Priorities of common processes and real-time processes

Process Priority Range	Priority Range of a Real-Time Process	Priority Range of a Common Process	Nice Priority Corresponding to a Common Process Priority	Default Process Priority in the Kernel
0–139	0–99	100–139	–20 to +19 (corresponding to 100 to 139 respectively)	120 (indicating nice = 0)



8.3 How Do I Set the dmesg Buffer Size?

To set the dmesg buffer size, select the following option in the **menuconfig** menu:

```
General setup ---> Kernel log buffer size
CONFIG_LOG_BUF_SHIFT (The default value is 18 in the kernel, which indicates
256 KB offset.)
```

8.4 Why Is the MemTotal Value in /proc/meminfo Different from mem=xxxM Configured on the CLI?

When the kernel starts, some memories are reserved, such as the memory occupied by the kernel code and reserved memory in pmem. The reserved memory size is subtracted from the **MemTotal** value.

For details about the code, read **init.c** in **arch/arm/mm/**.

```
totalram_pages += free_all_bootmem();
```

```
totalram_pages += totalhigh_pages;
```

The reserved memory and bootmem are allocated from the memblock.

totalram_pages is the total memory after the zone (buddy allocator) takes over memory management.

For details about each memory mentioned in meminfo, read **proc.txt** in **./Documentation/filesystems/**.

```
MemTotal: Total usable ram (i.e. physical ram minus a few reserved
          bits and the kernel binary code)
MemFree: The sum of LowFree+HighFree
Buffers: Relatively temporary storage for raw disk blocks
          should not get tremendously large (20MB or so)
Cached: in-memory cache for files read from the disk (the
        pagecache). Does Not include SwapCached
```

...

8.5 How Do I Determine Whether Stacks Overflow on Linux?

The following four types of stacks are used on Linux:

- Stacks that are temporarily used when the system is initialized
- Stacks for kernel program initialization in protection mode. This type of stacks is always in the kernel code address space, and is the user-state stacks used by task 0.

- Stacks that are called by each task over the system for running kernel programs. This type of stacks is named kernel-state stacks for tasks. Each task has an independent kernel-state stack.
- Stacks that are executed in user state by tasks. These stacks are located at the end of task or process logic address space.

Multiple stacks are used for the following two reasons:

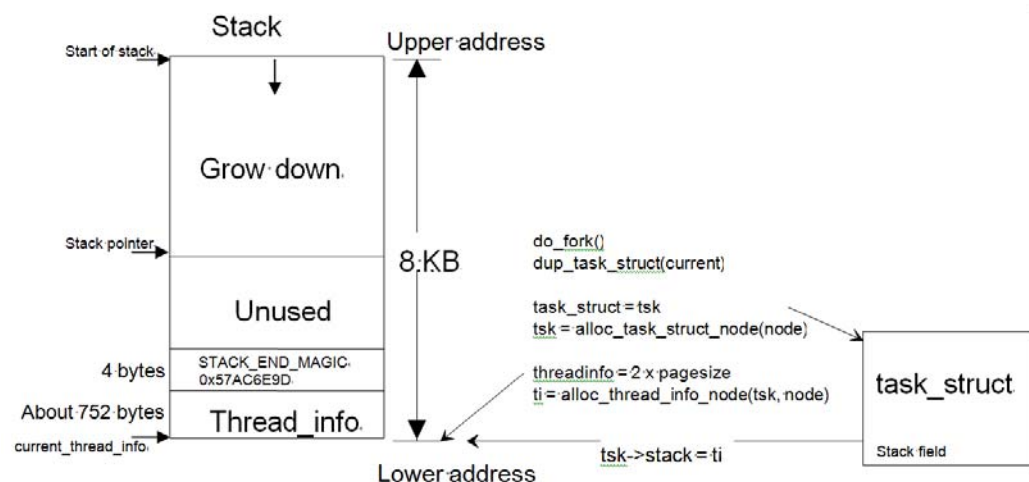
- When the real mode is changed to protection mode, the memory addressing mode of the CPU changes. The stack areas must be adjusted again.
- If a stack is shared in various CPU privilege levels, the data may be protected. Different stacks must be used for executing level-0 and level-3 kernel code.

When a task is in the kernel state, the level-0 stack pointers `tss.ss0` and `tss.esp0` (kernel stacks) provided by the TSS segment are used. The original user stack pointers are stored in kernel stacks. After the task state switches from the kernel state to user state, the user-state stacks are used.

When a process runs in the kernel state (for example, the process is called by the system), it uses its kernel stack. If the kernel stack size is 8 KB, the triggered interrupt handler uses the same kernel stack. If the kernel stack size is 4 KB, the triggered interrupt handler uses another kernel stack.

As the kernel stack size is small and fixed, the stack may overflow easily. Therefore, do not use the recursive calling mode in the kernel code. You are advised to change the recursive calling mode to cyclic calling mode even though you are familiar with the recursive hierarchy, because the recursive hierarchy may change. In addition, large or unknown stack variables (such as dynamic arrays) are not recommended. As the kernel stack shares a memory with the data structure `task_struct`, `task_struct` is destroyed if the kernel stack overflows. `task_struct` is important on Linux and is also called process control block (PCB). It defines and controls all processes. In kernel 2.6.x or later, the kernel stack shares a memory with the data structure `thread_info` but not `task_struct`. Because the first field in `thread_info` is the `task_struct` pointer, `task_struct` is also destroyed if the kernel stack overflows. In this case, the `task_struct` pointer points to an unpredictable address, and therefore the data in all fields of `task_struct` is trash data. See [Figure 8-1](#).

Figure 8-1 Stack structure





When the `do_fork()` function is called, the kernel allocates the `task_struct` data structure `tsk` by using the slab.

No space needs to be separately allocated to `threadinfo` because `threadinfo` is stored at the bottom of the thread stack space. That is, the space for storing `threadinfo` is reserved after a stack space is allocated.

`ti = alloc_pages(GFP_KERNEL, 1);` //order=1 indicates 8 KB space. The `ti` parameter indicates that 8 KB space is allocated, and `threadinfo` occupies the start part of the space. `threadinfo` occupies about 750-byte space. When the `do_fork` function is called, a magic value is placed after `threadinfo` to check whether `threadinfo` is overwritten due to stack overflow. `task_struct` has a `stack` field (`void * stack`) that points to the start address for its stack.

`tsk.stack = ti;` //The start address (pointer) for `threadinfo` is found by using the `stack` of `task_struct`. `threadinfo` has a `task` field (`struct task_struct *task`) that points to its `task_struct`. That is, there is a cross reference between `threadinfo` and `task_struct`.

The following part describes how to determine whether stacks overflow. The following is the end address where the kernel checks the stack:

```
static inline unsigned long *end_of_stack(struct task_struct *p)
{
    return (unsigned long *) (task_thread_info(p) + 1);
}
```

As shown in [Figure 8-1](#), the part above `thread_info` indicates the end of stack. The value plus 1 indicates that the address offset is `struct threadinfo`.

The kernel stores a `STACK_END_MAGIC` value at the end of stack.

After the address for the end of stack is obtained, whether the stacks overflow is determined by checking the read value. If the read value is not the `STACK_END_MAGIC` value, stacks overflow. The following example shows how to modify the code.

Add the code in red to the `asmlinkage void __do_softirq(void)` function in **kernel/softirq.c**.

```
...
do {
    if (pending & 1) {
        int printkflg = 0;
        static int ncount = 0;
        unsigned long *sp_magic = NULL;
        unsigned int tmp=current_thread_info();

        unsigned int vec_nr = h - softirq_vec;
        int prev_count = preempt_count();

        kstat_incr_softirqs_this_cpu(vec_nr);

        sp_magic = (unsigned long *) ((struct thread_info *)tmp +1);
        if(current_thread_info()->task->pid != 0)
        {
            register unsigned long sp asm ("sp");

```



```
        unsigned int tmp2=sp;
        static int minstack=0xFFFFFFFF;

        if((*sp_magic != 0x57AC6E9D)) ncount++;

        if( (tmp2 - tmp < minstack) || ((*sp_magic != 0x57AC6E9D) &&
(ncount < 2)))
        {
            minstack = tmp2 - tmp;
            printkflg = 1;
            printk("-----sp_magic = 0x%x, 0x%x,
0x%x\n",*sp_magic,0x57AC6E9D,sp_magic);
            printk("-----Dump At %s %d,
StackS %p(id=%d)-- %p ?= %08x -----\n",
                __FUNCTION__, __LINE__, current_thread_info(),
current_thread_info()->task->pid, &tmp, sp);
        }
    }

    trace_softirq_entry(vec_nr);
    h->action(h);
    trace_softirq_exit(vec_nr);

    if(current_thread_info()->task->pid != 0)
    {
        if( (printkflg) || ((*sp_magic != 0x57AC6E9D) && (ncount < 2)))
        {
            register unsigned long sp asm ("sp");
            unsigned int tmp2=sp;

            printkflg = 0;
            printk("-----sp_magic = 0x%x, 0x%x,
0x%x\n",*sp_magic,0x57AC6E9D,sp_magic);
            printk("-----Dump At %s %d,
StackS %p(id=%d)-- %p ?= %08x -----\n",
                __FUNCTION__, __LINE__, current_thread_info(),
current_thread_info()->task->pid, &tmp, sp);
            dump_stack();
        }
    }
    ...

```



8.6 How Do I Generate a Core Dump File for Problem Analysis When a Segmentation Fault Occurs?

Run the following command under the shell to generate a core dump file:

```
ulimit -S -c unlimited > /dev/null 2>&1
```

To ensure that error information is displayed properly in the core dump file, use the **-g** debugging option when you compile executable files.

8.7 What Precautions Should I Take While Using Caches?

The Hi3531 CPU is ARM Cortex-A9 that has L1 and L2 caches. Pay attention to the caches of the ARM architecture during encoding to avoid data inconsistency among the L1 cache, L2 cache, and memory. The following describes the precautions to be taken in two scenarios.

DMA Synchronous Processing

In this scenario, the processor and peripherals directly operate the physical memory. For example, the network controller transfers data packets to the physical memory because it cannot identify the caches. The processor also processes these data packets but it can identify the caches. If caches are hit, the processor directly operates the data in the caches but does not access the physical address. As a result, data inconsistency occurs.

In this case, the cache-related interfaces must be called to synchronize data.

The L1 cache or L2 cache is operated by cache line (32 bytes). The start address must be 32-byte-aligned and the address length must be an integral multiple of 32 bytes when the physical memory is allocated. If the physical memory areas are not aligned by cache line, the memory area positions are moved into the cache line range. If the areas are used, data inconsistency occurs.

Synchronous Processing at the Processor Side

In this scenario, only the processor can identify the address that requires data synchronization, which is irrelevant to the peripherals. Assume that a physical memory address is mapped into two virtual address spaces in cache mode and the two spaces belong to two processes. The processes use the memory for communication. The virtual memory address for process 1 is hit by caches, and the virtual memory address for process 2 is not hit by caches. In this case, data must be synchronized at the processor side. The standard kernel interface for operating the L1 cache is used in this scenario.

8.8 Why Is the Load Average Value Too Large? Does It Affect Services?

When a simple program is running, the **Load Average** value contained in the top information is 2.95 but the CPU usage is low. The **Load Average** value indicates the task queuing status of the CPU.



In the top command, the **load average** information indicates the average system loads within 1 minute, 5 minutes, and 15 minutes. The average system load is the number of average process trees in the queues (running processes or waiting processes) in a specific period. A process is in a running queue if it meets any of the following conditions:

- The process is not waiting for I/O operation results.
- The process does not proactively enter the wait state, that is, the 'wait' function is not called.
- The process is not stopped. For example, the process is waiting for end.

Processes are classified into blocked processes, runnable processes, and running processes by status on Linux. When a process is blocked, it waits for I/O device data or system calling.

When a process is runnable, it is in a running queue and asks for the CPU time with other runnable processes. The system load is the sum of the running processes and runnable processes. For example, if there are two running processes and three runnable processes, the system load is 5. The **load average** value is the load quantity during a period of time.

For example:

[\[Copy to clipboard\]](#)[View Code](#) BASH

```
1
2      # uptime
3      7:51pm up 2 days, 5:43, 2 users, load average: 8.13, 5.90, 4.94
```

The last three numbers in the command output indicate the average numbers of processes within 1 minute, 5 minute, and 15 minutes.

Typically, the system performance is good if the number of active processes is less than or equal to 3 for each CPU. If the number of processes is greater than 5 for each CPU, the system performance is poor. In the preceding example, if the system has two CPUs, the number of current processes is 4.065 (8.13/2) for each CPU. This indicates that the system performance is acceptable.

Therefore, it is normal that the number of current processes is 1.475 (2.95/2) for each CPU.

8.9 Why Is the Message "vmap Allocation for Size 528384 Failed:1.1 Use vmalloc=<size> to Increase Size." Displayed If the Memory Size Is Set to 512 MB or Larger?

The kernel occupies the virtual space ranging from the 3 GB position to the 4 GB position. That is, the kernel occupies only 1 GB virtual space. Except the space for exception vectors, there is only 992 MB available space. However, the malloc area allocated to services is greater than 512 MB. When the memory size is set to 512 MB, the malloc area is insufficient. As a result, the message "**vmap allocation for size 528384 failed:1.1 use vmalloc=<size> to increase size.**" is displayed. To resolve this problem, change the space occupied by the kernel from 1 GB to 2 GB as follows:

Change the duty ratio of the kernel-state address to the user-state address to 2:2.



Change the **Memory split** option on the **menuconfig**:

```
Kernel Features --->
Memory split (2G/2G user/kernel split) --->
| |      ( ) 3G/1G user/kernel split
      | |      (X) 2G/2G user/kernel split
            | |      ( ) 1G/3G user/kernel
```

To increase the vmalloc area size to 0x7e000000 (1912 MB), modify the **vmalloc.h** file in **arch/arm/mach-godnet/include/mach/**.

8.10 Can the Kernel Manage Two Inconsecutive Memories?

The kernel can manage two inconsecutive memories after configurations performed in **bootargs**. For example, in the configuration information "mem=256M@0x80000000 mem=256M@0xCFF00000", **256M** indicates the size of each memory, and **0x80000000** and **0xCFF00000** indicate the start address and end address for each memory respectively. As the address range from 0x80000000 to 0xC0000000 is too wide, the spaces after 0xC0000000 cannot be mapped. In this case, change the duty ratio of the kernel space to the user space from 1:3 to 2:2.

In addition, modify the MMZ code as follows:

In **media-mem.c** under **~/drv/mmz/**, change the judgment conditions in the **_check_mmz** function. The following are the judgment conditions before modification:

```
if(!((new_start>=__pa(high_memory)) || (new_start<PHYS_OFFSET &&
new_end<=PHYS_OFFSET))) {
    printk(KERN_ERR "ERROR: Conflict MMZ:\n");
    printk(KERN_ERR HIL_MMZ_FMT_S "\n", hil_mmz_fmt_arg(zone));
    printk(KERN_ERR "MMZ conflict to kernel memory (0x%08lX,
0x%08lX)\n", (long unsigned int)PHYS_OFFSET, __pa(high_memory)-1);
The following are the judgment conditions after modification:
if (pfn_valid(new_start>>PAGE_SHIFT) || pfn_valid(new_end >> PAGE_SHIFT))
{
    printk(KERN_ERR "ERROR: Conflict MMZ:\n");
    printk(KERN_ERR HIL_MMZ_FMT_S "\n", hil_mmz_fmt_arg(zone));
}
```

8.11 How Do I View the CPU Usage for Each Thread by Running the Top Command After the NPTL Tool Chain Is Used?

To view the CPU usage for each thread, run the **top** command and press **Shift+h**.

To ensure that the thread names displayed in the **top** command are the ones changed by using **prctl**, use the following attachment:



top

8.12 How Do I Display, Save, or Search the Stack Information About a Suspended Program by Using the Backtrace Function of the Gilibc Tool Chain?

The backtrace function of the gilibc tool chain is disabled by default. It is available only when the **-funwind-tables** parameter is added to the application compilation command. Otherwise, 0 is always returned when the backtrace function is used.

For example, to compile **test.c**, run the following command:

```
arm-hisiv200-linux-gcc -g -funwind-tables -rdynamic test.c
```

8.13 How Do I Configure a 1 ms Timer?

The chip provides multiple timers. For example, the Hi3518 has two dual-timers (dual-timer 0 and dual-timer 1). The operating system uses dual-timer 0 as the system clock. The following describes how to configure a 1 ms timer (3 MHz clock source) by using timer 1 of dual-timer 1 as an example:

Step 1 Select the clock source of timer 1. You can select the system bus clock or crystal oscillator clock. In this example, the external 3 MHz crystal oscillator clock is selected. Therefore, bit 19 and bit 20 of the system controller register whose offset address is 0x20050000 need to be set to 0.

Step 2 Write an initial value to **TIMERx_LOAD**.

Because the 3 MHz clock source is selected and the required interval of generating clock interrupts is 1 ms, the initial value to be written to **TIMERx_LOAD** is 3000 (3000000/1000).

Step 3 Write an initial value to **TIMERx_VALUE**.

The value should be the same as that written to **TIMERx_LOAD**. Otherwise, the first time of generating interrupt is not 1 ms.

Step 4 Configure timer 1 by setting **TIMERx_CONTROL** as follows:

- Timer count mode (periodic mode or one-shot mode)
- Clock divider (1, 16, or 256 for the Hi3518)
- Counter working mode (16-bit or 32-bit mode)
- Timer enable status
- Interrupt mask status



In this example, set bit 0 to **0** (periodic count mode), bit 1 to **1** (32-bit mode), bit[3:2] to **0** (no clock division), bit 5 to **1** (timer interrupt handled), and bit 6 to **1** (periodic count mode but not free-running mode). For details, see the *Hi3518 720p IP Camera SoC Data Sheet*.

Step 5 Register the interrupt service routine (ISR) of timer 1.

Step 6 Enable timer 1 by setting bit 7 to **1**.

----**End**

For details about the code, see the following motor drive implementation:



hisi_motor.rar