Hikey970

Development Guide

Issue	01
Date	2018-03-15

Trademarks and Permissions

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/cn/

Email: support@hisilicon.com

About This Document

Purpose

This document instructs developers to compile and upgrade the Hikey970 native kernel image.

Intended Audience

This document is intended for:

- Instruct developers to compile native kernel images;
- Guide the developer to upgrade kernel images.

Change History

Changes between document issues are cumulative. The latest document issue contains all the changes made in earlier issues.

Issue 02 (2018-05-10)

Add explain for UEFI build.

Issue 01 (2018-03-15)

This issue is used for first office application (FOA).

Contents

1 Compiling UEFI Image

1.1 Step 1: Download the Android Source Tree

```
repo init -u https://android.googlesource.com/platform/manifest -b master
git clone https://github.com/96boards-hikey/android-manifest.git -b
hikey970_v1.0 .repo/local_manifests
repo sync --force-sync -j8
```

1.2 Step 2: Download tools and images for HiKey970

```
git clone https://github.com/96boards-hikey/tools-images-hikey970.git
```

1.3 Step 3: Build UEFI Image

Execute follow command to compile the image:

```
./device/linaro/hikey/l-loader/build_uefi.sh hikey970
```

After the compilation is completed, the “fip.bin” “l-loader.bin” and “prm_ptable.img” can be generated under “device/linaro/hikey/l-loader”.

NOTE:

1. The ptable for hikey970 is ptable-aosp-32g.img.
2. When Build UEFI,You need install Linaro GCC 7.1.1 in path:
   ```
   /opt/toolchain/gcc-linaro-7.1.1-2017.08-x86_64_aarch64-linux-gnu/bin/
   ```
   You can get it from:
   http://releases.linaro.org/components/toolchain/binaries/
3. When Build UEFI, If print error information about “uuid” or “ssl”,Maybe you need install uuid-dev and openssl libssl-dev in your PC,you can follow:
   ```
   sudo apt-get install uuid-dev
   sudo apt-get install openssl libssl-dev
   ```

2 Compiling Boot Image

2.1 Step 3: Copy “build_kernel.sh”

Copy the “build_kernel.sh” from the tools-images-hikey970 directory to root directory of the Android Source Tree. Like this:

```
./system
./kernel
./device
./bootable
…
./build_kernel.sh
```

2.2 Step 4: Build “boot.img”

Execute "./build_kernel.sh" in the tools directory to compile the image.

If the error information is like this：

```
Kernel Image build success!
  CHK     scripts/mod/devicetable-offsets.h
 Hikey970 dtb build success!
 /home/xxx/AOSP/out/target/product/hikey970/ramdisk.img is not exist!
please build ramdisk first.
 . ./build/envsetup.sh && lunch hikey970-userdebug && make ramdisk
```

Please use the following command to compile the ramdisk image：

```
. ./build/envsetup.sh && lunch hikey970-userdebug && make ramdisk
```

After the compilation is completed, the boot.img can be generated under “out/target/product/hikey970”.

3 Compiling Userspace

3.1 Build System Image

```
source ./build/envsetup.sh
lunch hikey970-userdebug
make systemimage -j32
```

If print error information about “dtb”.

Please use the following command in the “/kernel/hikey-linaro” directory to compile the “dtb” image:

```
make hisilicon/kirin970-hikey970.dtb
cp arch/arm64/boot/dts/hisilicon/kirin970-hikey970.dtb 
${LOCAL_DIR}/device/linaro/hikey-kernel
```

After the compilation is completed, the “system.img” can be generated under “out/target/product/hikey970”. And “userdata.img” “cache.img” “ramdisk.img” can be generated under same path.

4 Base Firmware Files and Installation

This section shows how to install all base firmware components for the HiKey970. Once finished with these instructions, please continue to the HiKey970 documentation landing page to flash an operating system.

- **Step 1**: Material and preparation
- **Step 2**: Dependencies
- **Step 3**: Enter recovery/forced-download mode on HiKey970
- **Step 4**: Flash base firmware
- **Step 5**: Explore other modes, proceed to OS installation
- **Troubleshooting**

You can get the base firmware for the HiKey970 from:
https://github.com/hihope/hiley970_aosp_o_fw

4.1 Step 1: Material and Preparation

- HiKey970
- USB Type-A (Host machine) to USB Type-C (96Boards) cable
- 96Boards compliant power supply
- To boot into fastboot mode everytime set switch 1 & 3 to ON state and switch 2 to OFF state.
- To boot into fastboot mode at every alternate reboot set switch 1 to ON and switch 2 & 3 to OFF state.
- To boot into recovery mode set switch 1 & 2 & 3 to ON state

4.2 Step 2: Dependencies

Host Linux Machine

- Remove modem manager. At least in Ubuntu 14.04 and 16.04 version, we found a conflicting issue if modem manager is installed and active. Modem manager monitors ttyUSBx's incoming data, when it reads some given pattern, it will send some bytes back into the tty as response. And those bytes sent by modem manager can make board side recovery flashing tool confuse and fail. Solution is to uninstall this service. If you have a doubt whether you are safe to remove it or not, double confirm here: ModemManager homepage.

```
$ sudo dpkg -s modemmanager
```

```
$ sudo apt-get remove modemmanager
```

- Android SDK “Platform-Tools” for Linux can be downloaded here
- Use terminal to clone this repository into desired folder and cd into tools-images-HiKey970

```
$ git clone https://github.com/96boards-hikey/tools-images-hikey970.git
```

```
$ cd tools-images-hiKey970
```

4.3 Step 3: Enter recovery/forced-download Mode on HiKey970

- Remove power from the board
- Change Jumper/DIP switch settings, to enter recovery/forced-download mode:

Name	Switch	State
Auto Power up	Switch 1	ON
Recovery	Switch 2	ON
Fastboot	Switch 3	ON

- Apply power to the board using 96Boards compliant power supply
- Insert USB Type-C cable (OTG port) to the board, and connect the other end to your Linux PC
- Check whether there is a device node "/dev/ttyUSBx". If there is, it means your PC has detected the target board; If there is not, try to repeat previous steps.

4.4 Step 4: Flash Base Firmware

Once again using the terminal on your host machine, execute the following command. Be sure to replace /dev/ttyUSBx with the USB value detected by your machine.

```
$ sudo ./recovery-flash.sh /dev/ttyUSBx
```

After it completes, the base firmware will be flashed to the device, this does not mean OS.

The board will then be in fastboot mode.

4.5 Step 5: Explore Other Modes, Proceed to OS Installation

- sw2402 mode
- Proceed to OS "Installation" through the HiKey970 documentation landing page

4.6 Troubleshooting

- If recovery script "./recovery-flash.sh /dev/ttyUSBx" fail to run to completion and you see "< waiting for any device >" in a loop, then try uninstalling modem manager from your host machine. The script will work after that. Don't forget to install modem manager back after recovery.

Switch	Normal Mode	Fastboot Mode	Recovery Mode
Switch 1	ON	ON	ON
Switch 2	OFF	OFF	ON
Switch 3	OFF	ON	ON

- If you run into trouble, see the README-technical.mdfile in this directory.

5 Another way to upgrade

Another way to upgrade using Hisi-idt tool, the advantage of this tool is cross-platform,at Ubuntu or Windows OS.

Hisi-idt for downloading binaries to soc RAM and DDR through serial port,and then upgrade the other images through fastboot command.

5.1 Command

- Linux

```
sudo python hisi-idt.py -d /dev/ttyUSBx --img1 ./ sec_usb_xloader.img
--img2 ./sec_usb_xloader2.img --img3 ./l-loader.bin
```

- Windows

```
python hisi-idt.py -d commxx --img1 sec_usb_xloader.img
--img2 sec_usb_xloader2.img --img3 l-loader.bin
```

5.2 Download Steps

Step 1 Insert USB cable and connect with PC;

Step 2 Enter force download mode:

For hikey970 board: sw2402

switch 1 mode: ON

switch 2 mode: ON

switch 3 mode: ON

release “Reset” key then will enter into “force download” mode;

Step 3 Check if there have the device node “/dev/ttyUSBx”, if there have device node that means the PC has detected the target board; d. Use command "sudo python hisi-idt.py" to run the script; after IDT download binaries successfully, it will print out below log:

```
+----------------------+
 Serial:  /dev/ttyUSB1
 Image1:  fastboot1.img
 Image2:  fastboot2.img
+----------------------+

Sending fastboot1.img ...
Done

Sending fastboot2.img ...
Done
```

5.3 Burn Images

After download sec_usb_xloader.img   sec_usb_xloader2.img and l-loader.bin on the board, then can use fastboot command to burn images:

```
sudo fastboot flash fip  fip.bin
sudo fastboot flash fastboot l-loader.bin
```

……

5.4 Troubleshooting

Step 1 After enter the force download mode, if Ubuntu PC cannot recognize the device ttyUSBx; this issue can be fixed by input below commands: sudo echo 12D1 3609 > /sys/bus/usb-serial/drivers/option1/new_id sudo makenod /dev/ttyUSB0 c 188 0

Step 2 Need supervisor permission for hisi-idt.py: "sudo python hisi-idt.py"

Step 3 Need supervisor permission for fastboot: "sudo fastboot"

Step 4 If download binaries failed with below message:

```
Sending fastboot1.img ...
failed
failed
```

Usually this means you are using the wrong ttyUSBx device; the reason is when connect board with the UART cable and USB cable, then PC will create two device nodes /dev/ttyUSB0 and /dev/ttyUSB1;But the nodes which are randomly binding to UART and USB, so sometimes /dev/ttyUSB0 is created for the UART and /dev/ttyUSB1 is for the USB port, in this case should use /dev/ttyUSB1 for the IDT; if PC exchanges the nodes then should use /dev/ttyUSB0.