Security Class: Top-Secret( ) Secret( ) Internal( ) Public(√)

RK3588 Android 12.0 SDK Developer Guide

Status:
[ ] Draft
[√] Released
[ ] Modifying

File No.:	RK-KF-YF-302
Current Version:	V0.1.0
Author:	Wu Liangqing
Finish Date:	2022-01-21
Auditor:	Chen Haiyan
Finish Date:	2022-01-21

Version no.	Author	Revision Date	Revision Description	Remark
V0.0.1	Wu Liangqing/Bian Jinchen	2022-01-10	release rkr2 SDK support rk3588/rk3588S	Only for hardware debugging
V0.1.0	Wu Liangqing/Bian Jinchen	2022-01-21	release rkr4 SDK support rk3588/rk3588S	Alpha version

If there is any question about the document, please email to: wlq@rock-chips.com

DISCLAIMER

Trademark Statement

All rights reserved. ©2022. Rockchip Electronics Co., Ltd.

Rockchips Electronics Co., Ltd.
No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian, PRC
Website: www.rock-chips.com
Customer service Tel: +86-4007-700-590
Customer service Fax: +86-591-83951833
Customer service e-Mail: fae@rock-chips.com

RK3588 Android 12.0 SDK Chipset support

Chipset platform	Support or not	SDK version
RK3588	Support	RKR2
RK3588S	Support	RKR2

RK3588 Android 12.0 SDK code download and compile

Code download

Download address

```
repo init --repo-url=ssh://git@www.rockchip.com.cn:2222/repo-release/tools/repo.git -u ssh://git@www.rockchip.com.cn:2222/Android_S/rk3588-manifests.git -m Android12.xml
```

Download server mirroring

```
repo init --repo-url=ssh://git@www.rockchip.com.cn:2222/repo-release/tools/repo.git  -u ssh://git@www.rockchip.com.cn:2222/Android_S/rk3588-manifests.git -m Android12.xml --mirror
```

Note: repo is a script invoking git developed by Google using Python script, and mainly used to download, manage Android project software lib. The download address is as follows:

```
git clone ssh://git@www.rockchip.com.cn:2222/repo-release/tools/repo
```

Generally, Rockchip FAE contact will provide the initial compressed package of the corresponding version SDK in order to help customers acquire SDK source code quickly. Take `RK3588_ANDROID12.0_SDK_RELEASE.tar.gz.*` as an example, you can sync the source code through the following command after getting the initial package:

```
mkdir RK3588_ANDROID12.0_SDK_RELEASE
cat RK3588_ANDROID12.0_SDK_RELEASE.tar.gz* | tar -zx -C
RK3588_ANDROID12.0_SDK_RELEASE
cd RK3588_ANDROID12.0_SDK_RELEASE
.repo/repo/repo sync -l
.repo/repo/repo sync -c
```

Set up your own repo code server

Environment

You can install openssh-server for remote login, git for project management, and keychain for public key and private key management tools.

```
sudo apt-get install openssh-server git keychain
```

Set up gitolite

Server-side operation

Take server address: 10.10.10.206 as an example for description.

1. create git account:

```
sudo adduser --system --shell /bin/bash --group git
sudo passwd git
```

2. Log in to the server as a 'git' account;
3. Make sure that '~/.ssh/authorized_keys' is empty or non-existent;
4. Copy the server administrator's public key to '~/ yourname.pub';
5. Download gitolite source code;

```
git clone https://github.com/sitaramc/gitolite.git
```

6. Create bin directory in git user directory;

```
mkdir -p ~/bin
```

7. Please execute following commands to install gitolite, and the installation method is different for different versions. Please refer to the documentation in source code:

```
```

8. Set the administrator.

```
```

Client-side operation

1. Clone gitolite management warehouse of the server;

2. Add user's public key to the gitolite directory;

3. Add an administrator user.

Set up repo mirror

Server-side operation

1. Log in to the server as a 'git' account;
2. Download the repo tool in the root directory;

3. Create a new RK_Android12_mirror directory;

4. Enter the RK_Android12_mirror directory;

5. Download RK Android12 SDK mirror;

6. Create warehouse group permissions.

Client-side operation

1. Copy android_r.conf on the server-side to ·gitolite-admin/conf/· on the client-side;
2. Add group permissions.

3. Create your own new manifests warehouse.

Client-side operation

1. Download manifests_xxx warehouse on the client-side;
   Download manifests_xxx.git warehouse on other client-side

2. Download original manifests warehouse on the client-side;

3. Submit manifest.xml file to manifest_xxx warehouse created newly;
   Copy the files below original manifests to the manifests_xxx

check copy files

Local commits

Push to the remote branch

4. Create your own code download link.
   Download the repo tool in the root directory

After following the above steps, your own code download link is as follows

Thereinto:
`//10.10.10.206` which is your server address
You can complete your own repo server set-up with above steps, and you can share your code server links with colleagues to work together.

Code management

After setting up the code server with above steps, most of the code warehouses use the default branches of RK. If some warehouses need to modify their own codes, you can refer to the following steps for operation.

Switch your own code branches

1. Enter the code warehouse that needs to be modified, and we take the kernel directory as an example to illustrate;

2. Switch a local branch;

3. Push xxx_branch to remote server;

Thereinto, `rk29` is remote, which can be completed automatically by tab key directly

4. Enter.repo/manifests directory and modify the branch which is appointed by manifest;
   Enter.repo/manifests directory, and you can find the manifest location corresponding to the kernel warehouse by grep kernel

5. Submit the modified manifest to the remote branch.

After submitting manifests warehouse, other colleagues can synchronize the kernel codes of your own branches.

Code modification submittal

After switching branches according to the steps above, you can commit your modification on your branches and push them directly to the xxx_branch.

Synchronize RK codes

1. It's required to synchronize RK codes on the server-side;

2. The manifests that RK modifies are combined by client-side;

- Download the original manifests warehouse of RK;

The manifests (RK original) and the manifests_xxx (yourselves) are compared with the contrast tools to combine the different parts that RK modifies to your own warehouses (mainly modify the tag, adding and removing the warehouse, etc)

- After comparing and confirming, the modification will be pushed to the Manifests XXX.

You can also confirm which warehouses are modified in this step, and in the next step you will combine the modified warehouses.

3. The directories switched branches by yourselves need to push the merge that RK modifies to your own branches manually.
   Take kernel as an example：

- Check the pointed remote branches at present

You can find that the branch pointed at present is: `remotes/m/master -> rk29/xxx_branch`

- Create a local branch (switch from your own remote branch)

- Check latest TAG published currently by RK

You can find the latest tag of Android12 currently is `android-12.0-mid-rkr1`

- combine `android-12.0-mid-rkr1` to the local branch

Check if there is a conflict. If there is a conflict, resolve the conflict firstly. You can execute the next step when there is no conflict.

- push the codes which have been combined to the remote branch

```
```

- The other directories swiched can be combined and submitted in this way

kernel Code path description

RK3588 Android12 only supports version 5.10 of the kernel, kernel source code in the project kernel-5.10 directory,

Code compiling

One key compiling command

```
```

Compiling command summary

Soc	type	model	Android	one key compiling	kernel compiling	uboot compiling
RK3588S	tablet	prototype	build/envsetup.sh;lunch rk3588s_s-userdebug	./build.sh -AUCKu -d rk3588s-tablet-v10	make ARCH=arm64 rockchip_defconfig android-11.config pcie_wifi.config;make ARCH=arm64 rk3588s-tablet-v10.img -j24	./make.sh rk3588
RK3588S	development board	RK3588S-EVB1	build/envsetup.sh;lunch rk3588_s-userdebug	./build.sh -AUCKu -d rk3588s-evb1-lp4x-v10	make ARCH=arm64 rockchip_defconfig android-11.config pcie_wifi.config;make ARCH=arm64 rk3588s-evb1-lp4x-v10.img -j24	./make.sh rk3588
RK3588	development board	RK3588 EVB1	build/envsetup.sh;lunch rk3588_s-userdebug	./build.sh -AUCKu	make ARCH=arm64 rockchip_defconfig android-11.config pcie_wifi.config;make ARCH=arm64 rk3588-evb1-lp4-v10.img -j24	./make.sh rk3588

Other compiling instruction

Android12.0 cannot directly flash kernel.img and resource.img

Android12.0 kernel.img and resource.img are included in boot.img. which need to use 'build.sh -K' command for compiling kernel. Please flash boot.img under rockdev after compiling. You can also compile kernel separately by following method.

Only compile kernel to generate boot.img

Compile principle: Old `boot.img` is replaced by `kernel.img` and `resource.img` compiled and generated in the catalogue of kernel-5.10.
Take `RK3588` prototype for example， replace corresponding boot.img and dts when compiling：
Among the rest, `BOOT_IMG=../rockdev/Image-rk3588_s/boot.img` assigns old boot.img path，the commands are following:

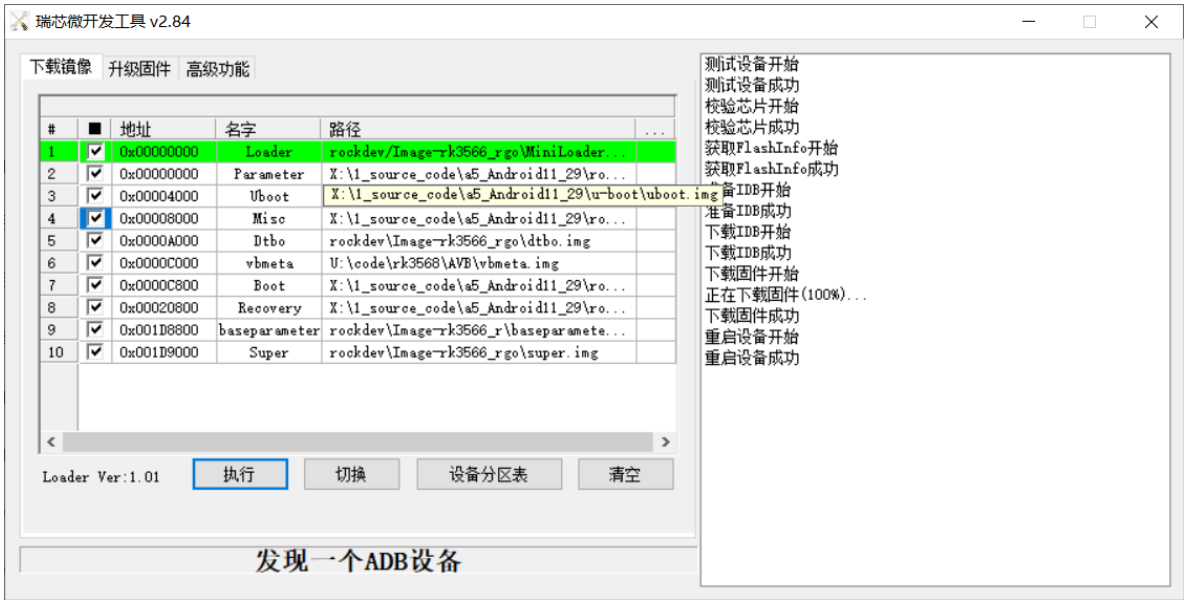
Export clang to the environment

You can flash boot.img under the catalogue of kernel-5.10 directly to boot position of machine after compiling, and please load the partition table（parameter.txt） when flashing, for fear of flashing to the wrong place.

Image flashing

Image flashing tool

Android12 requires to update the USB driver DriverAssitant to V5.1.1 version. You can refer to the tool instruction chapter to do the upgrade.
Windows flashing tool:



There are more details in the tool instruction chapter.

Image instruction

After complete compiling, it will generate the following files:

Just use the tool to flash the following files:(no need to flash trust.img for RK3566/RK3568)

or you can directly flash `update.img`

Image instruction

Image	Instruction
boot.img	including ramdis、kernel、dtb
boot-debug.img	the difference between boot.img and boot-debug.img is that user image can flash this boot.img to do root operation
dtbo.img	Device Tree Overlays refer to dtbo chapter instruction later
config.cfg	the configuration file of the flash tool, you can directly load the options required to be flashed for the flash tool
MiniLoaderAll.bin	including first level loader
misc.img	including recovery-wipe boot symbol information, after flashing it will enter recovery
parameter.txt	including partition information
pcba_small_misc.img	including pcba boot symbol information, after flashing it will enter the simple pcba mode
pcba_whole_misc.img	including pcba boot symbol information, after flashing it will enter the complete pcba mode
recovery.img	including recovery-ramdis、kernel、dtb
super.img	including the contents of odm、vendor、system partitions
trust.img	including BL31、BL32 which are not generated for RK3566/RK3568, no need to flash
uboot.img	including uboot image
vbmeta.img	including avb verification information, used for AVB verification
update.img	including the above img files to be flashed, can be used for the tool to directly flash the whole image package

Use fastboot to flash dynamic partition

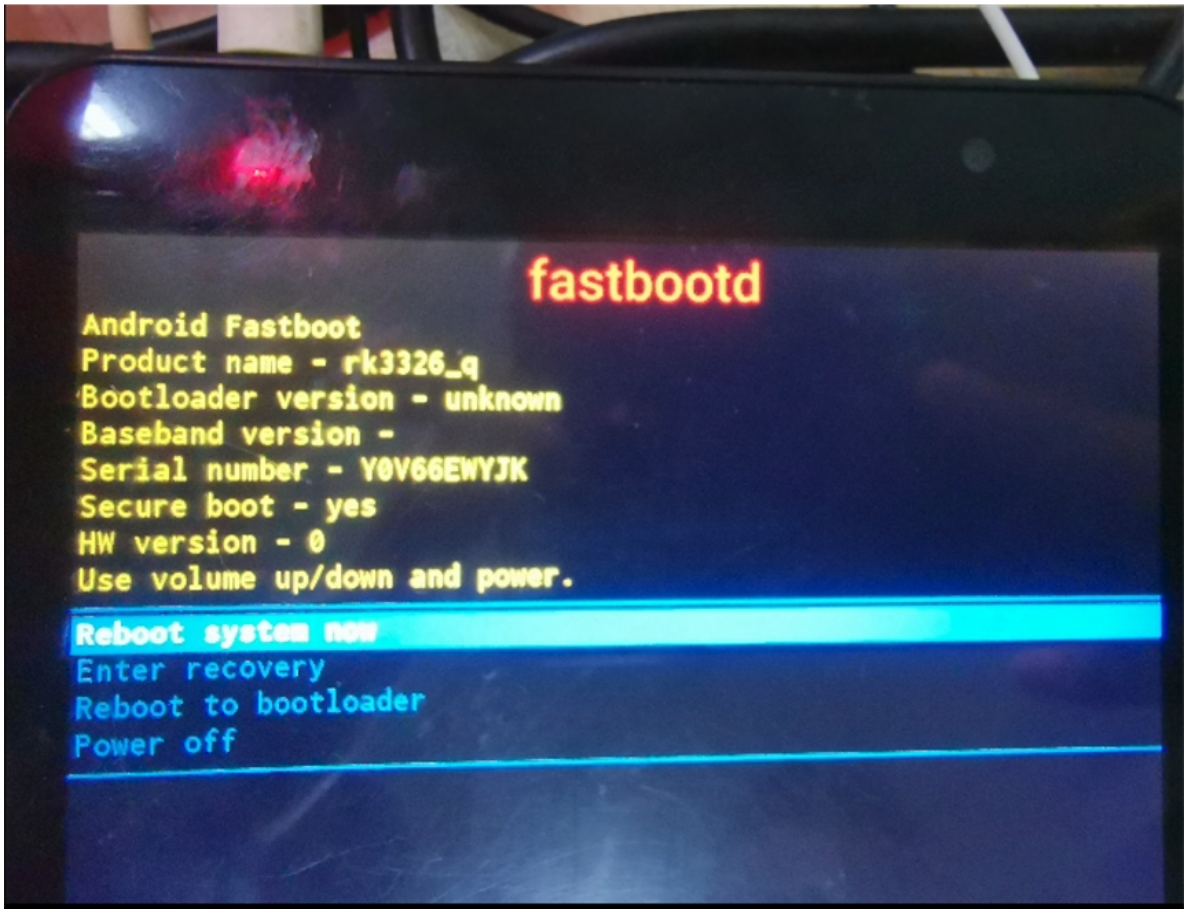
The new device with R supports dynamic partition, and already removes system/vendor/odm partitions. Please flash super.img. Use `fastbootd` can flash system/vendor/odm alone. The version of adb and fastboot should be the latest. SDK provides the compiled tool package:

```
```

- Use the command to flash dynamic partition:

```
```

Note: After entering fastbootd mode, relative information of the device will be displayed on the screen, as shown below:



The way to flash GSI:

- After the device is unlocked, enter fastbootd, only need to flash system.img of GSI and misc.img of the image, and after flashing it will enter recovery to do factory reset. Attach the complete flashing process as below:

1. Reboot to bootloader, lock->unlock the device:

```
```

2. Reset to factory setting, reboot to fastbootd:

```
```

3. Start to flash GSI

```
```

- Use DSU(Dynamic System Updates) to flash GSI, and current Rockchip platform already supports DSU by default. As this function requires large memory, it is not recommended to use on the device with 1G DDR or less. For the instruction and usage of DSU, please refer to Android official website: https://source.android.com/devices/tech/ota/dynamic-system-updates
- Note 1: when testing VTS, need to flash the compiled boot-debug.img to boot partition.
- Note 2: when testing CTS-ON-GSI, no need to flash boot-debug.img.
- Note 3: please use GSI image ended with -signed released by Google for testing.

DTBO function

Android 10.0 and above versions support Device Tree Overlays function, which requires to flash dtbo.img during development, and is compatible with multiple products.
The modification method:

1. Find (or specify) the template file:

For example:

2. Add or modify the required node:
   For example:

Note: There must be alias in the dts when using dtbo, otherwise it cannot overlay successfully

Modify fstab file

1. Fine (or specify) the template file:

For example:

2. Modify: add partition mounting, modify swap_zram parameter, modify data partition format and so on

Modify parameter.txt

Android 12 adds the tool that can generate parameter.txt, and support to compile parameter.txt according to the configuration parameters. If there is no configuration template file, it will find and add the modified parameter.txt file.

1. Find (or specify) the template file:

For example:

2. Partition size configuration(example as below):

3. Not to use parameter generation tool:
   Just add a parameter.txt file to your device directory:
   For example:

4. Only use the tool to generate parameter.txt(example as below):

Note: If need to do the big version upgrade through OTA, please directly use the previous version's parameter.txt

Android common configuration

Create product lunch

Take RK356x platform as example to create a new rk3588_s product. The steps are as below:
1.Modify device/rockchip/rk3588/AndroidProducts.mk to add rk3588_s lunch

2.Create rk3588s_s directory under device/rockchip/rk3588
Create referring to the existing rk3588s_s product directory in device/rockchip/rk356x. You can directly copy rk3588s_s to rk3588s_s, and then replace all the `rk3588_s` under rk3588s_s directory with `rk3588s_s`

Kernel dts instruction

Create new product dts

You can select the corresponding dts according to the configuration in the following table as reference to create new product dts.

Soc	PMIC	DDR	Type	Model	DTS
RK3588	RK806 * 2	LPDDR4	development board	RK3588 EVB1	rk3588-evb1-lp4-v10
RK3588	RK806 * 2	LPDDR5	development board	RK3588 EVB3	rk3588-evb3-lp5-v10
RK3588	RK806+RK860	LPDDR4	development board	RK3588 EVB	rk3588-evb4-lp4-v10
RK3588S	RK806 * 2	LPDDR4X	development board	RK3588S EVB1	rk3588s-evb1-lp4x-v10
RK3588S	RK806 * 2	LPDDR4	tablet	prototype	rk3588s-tablet-v10

Document instruction

Peripheral support list

DDR/EMMC/NAND FLASH/WIFI/3G/CAMERA support lists keep updating in redmine, through the following link:

Android document

Android_SELinux(Sepolicy) developer guide

Android 12 System Optimization developer guide

Including bootup speed up, app startup speed up, performance, memory optimization and commonly used analysis tools

Wi-Fi document

3G/4G module instruction document

Kernel document

DDR related document

Audio module document

CRU module document

GMAC module document

PCie module document

I2C module document

PIN-Ctrl GPIO module document

SPI module document

Sensor module document

IO-Domain module document

Leds module document

Thermal control module document

PMIC power management module document

MCU module document

Power consumption and sleep module document

UART module document

DVFS CPU/GPU/DDR frequency scaling related document

EMMC/SDMMC/SDIO module document

PWM module document

USB module document

HDMI-IN function document

Security module document

uboot introduction document

Trust introduction document

Camera document

Camera IQ Tool document

Tool document

PCBA development and usage document

Panel driver debugging guide

HDMI debugging guide

Graphic display DRM Hardware Composer（HWC) issue analyzing

DRM display developer guide

RGA related issues analyzing

Graphic display framework common issue analysis

include frameworks、GPU.Gralloc、GUI、HWComposer、HWUI、RGA

Tool usage

StressTest

Use the Stresstest tool to do the stress test for the various functions on the target devices to make sure the whole system running stably. SDK can start StressTest application and perform stress test of various functions by entering `“83991906=”` code in the calculator.
The test items of Stresstest tool mainly include:

Module related

- Camera stress test: including Camera on/off, Camera taking photo and Camera switch.
- Bluetooth stress test: including Bluetooth on/off.
- Wi-Fi stress test: including Wi-Fi on/off, (plan to add ping test and iperf test).

Non module related

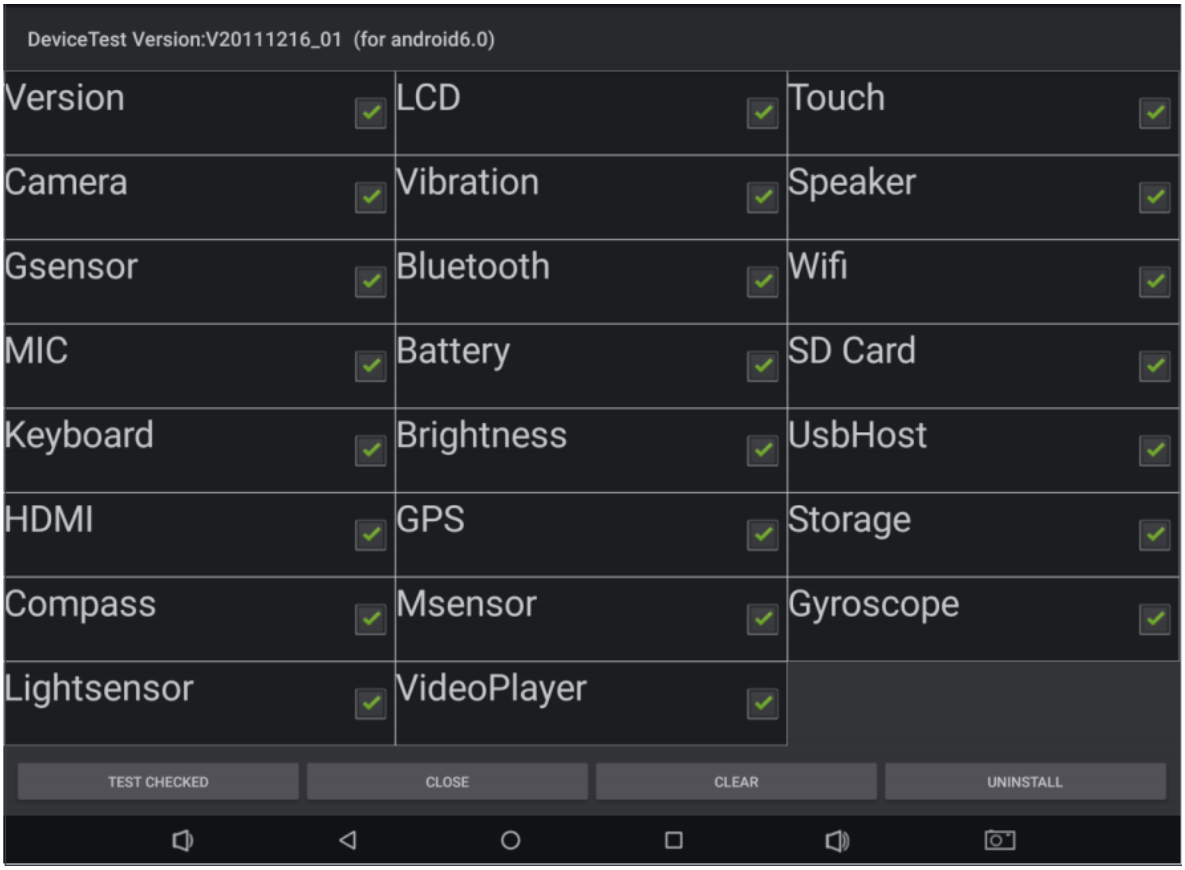
- Fly mode on/off test
- Suspend and resume stress test
- Video playing stress test
- Reboot stress test
- Recovery stress test
- ARM frequency scaling test
- GPU frequency scaling test
- DDR frequency scaling test

PCBA test tool

PCBA test tool is used to help quickly identify good and bad product features during production to improve the production efficiency. Current test items include panel (LCD), wireless (Wi-Fi), Bluetooth, DDR/EMMC memory, SD card, USB HOST, key, speaker earphone (Codec).
These test items include automatic test item and manual test item. Wireless network, DDR/EMMC, Ethernet are automatic test items, while key, SD card, USB Host, Codec are manual test items.
For the detailed PCBA function configuration and usage, please refer to:

DeviceTest

DeviceTest is used for the whole device test in factory, which mainly test whether the peripheral components work normally after assembling. SDK will enter DeviceTest by entering “000.=” code in the calculator, as shown below:



In factory, you can test the corresponding peripheral according to this interface. Click “TEST CHECKED” to test the items one by one. If succeed, click pass, if fail, click failed, the final result will display on the screen, as shown below. Red means failed item, others are pass, and the factory can repair accordingly based on the test result. Besides, if customers need to customize the tool, please contact FAE to apply for the corresponding source code.

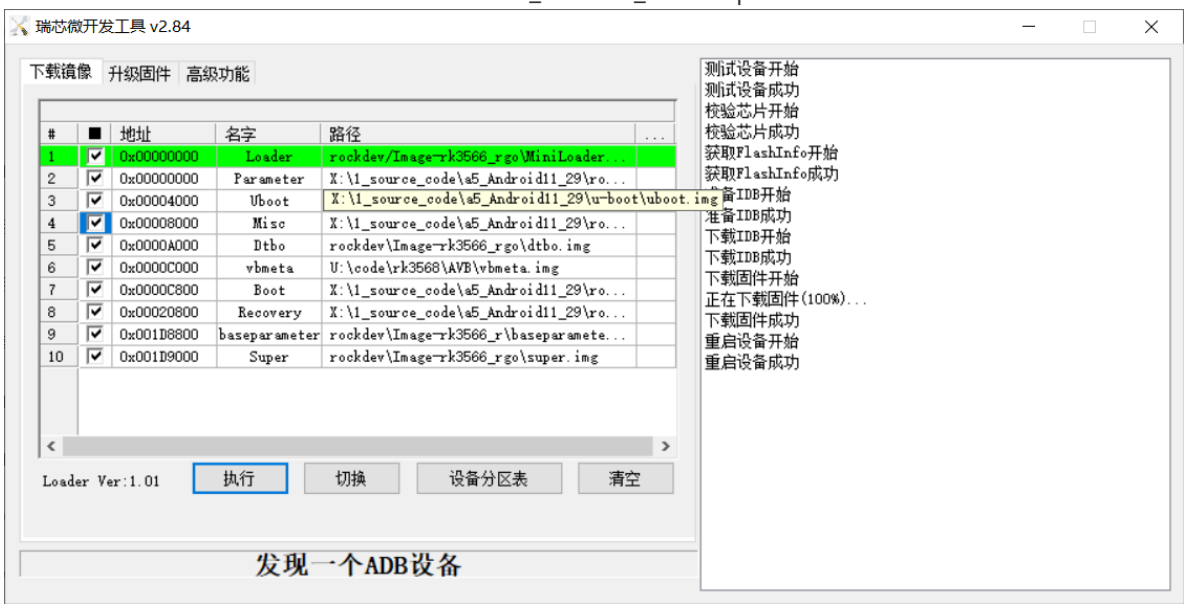
USB driver

Rockchip USB driver install package includes ADB and image flashing driver

Development flashing tool

Windows version

RKTools/windows/AndroidTool/AndroidTool_Release_v2.90.zip



Linux version

RKTools/linux/Linux_Upgrade_Tool/Linux_Upgrade_Tool_v1.56.zip

Tool to implement SD upgrading and boot

It is used to implement SD card upgrading, SD card boot, SD card PCBA test.

Write SN tool

RKTools\windows\RKDevInfoWriteTool_Setup_V1.0.3.rar
Install after unzip RKDevInfoWriteTool_Setup_V1.0.3.rar
Use admin ID to open the software

For the tool instruction, please refer to:

DDR welding test tool

It is used to test DDR hardware conncection, troubleshooting hardware issues such as virtual welding.

efuse flashing tool

It is used to flash efuse, suitable for RK3288W/RK3368/RK3399 platforms.

efuse/otp sign tool

It is used to sign efuse/otp of image.

Factory production image flashing tool

It is uesed for batch image flashing in factory.

Image update tool

It is used to modify update.img.

userdata partition data prebuilt tool

It is the tool used to make userdata partition pre-built data package.

System debugging

ADB tool

Overview

ADB (Android Debug Bridge) is a tool in Android SDK which can be used to operate and manage Android simulator or the real Android device. The functions mainly include:

- Run the device shell (command line)
- Manage the port mapping of the simulator or the device
- Upload/download files between the computer and the device
- Install the local apk to simulator or Android device
  ADB is a “client – server” program. Usually the client is PC and the server is the actual Android device or simulator. The ADB can be divided into two categories according to the way PC connects to the Android device:
- Network ADB: PC connects to STB device through cable/wireless network.
- USB ADB: PC connects to STB device through USB cable.

USB adb usage

USB adb usage has the following limitations:

- Only support USB OTG port
- Not support multiple clients at the same time (such as cmd window, eclipse etc.)
- Support host connects to only one device but not multiple devices
  The connection steps are as below:
  1、The device already running Android system, setting -> developer option -> connect to the computer, enable usb debugging switch.
  2、PC connects to the device USB otg port only through USB cable, and then the computer connects with Android device through below command:

3、Execute the command ”adb devices” to see if the connection is successful or not. If the device serial number shows up, the connection is successful.

ADB commonly used command elaboration

（1）Check the device situation
Check the Android device or simulator connected to computer:

The return result is the serial number or IP and port number, status of the Android device connected to PC.
（2）Install APK
Install the specific apk file to the device:

For example:

Re-install application:

（3）Uninstall APK
Complete uninstall:

For example:

（4）Use rm to remove apk file:

For example:

Note: remove “WishTV.apk” file in the directory of “system/app”.
（5）Enter shell of the device and simulator
Enthe the shell environment of the device or simulator:

（6）Upload the file to the device from PC
Use push command can upload any file or folder from PC to the device. Generally local path means the computer and remote path means the single board device connected with ADB.
adb push
For example:

Note: upload local “WishTV.apk” file to the “system/app” directory of the Android system.

（7）Download the file from the device to PC

Use pull command can download the file or folder from the device to local computer.

For example:

Note: download the file or folder from the “system/app” directory of Android system to local “F:\” directory.

（8）Check bug report

Run adb bugreport command can check all the error message report generated by system. The command will show all dumpsys, dumpstate and logcat information of the Android system.

（9）Check the device system information

The specific commands in adb shell to check the device system information.

Logcat tool

Android logcat system provides the function to record and check the system debugging information. The logcats are all recorded from various softwares and some system buffer. The buffer can be checked and used through Logcat. Logcat is the function most commonly used by debugging program. The function shows the program running status mainly by printing logcat. Because the amount of logcat is very large, need to do filtering and other operations.

Logcat command usage

Use logcat command to check the contents of the system logcat buffer:
The basic format:

For example:

The commonly used logcat filter method

Several ways to control the logcat output:

- Control the logcat output priority
  For example:

Note: show the logcat information with priority of warning or higher.

- Control the logcat label and output priority
  For example:

Note: support all the logcat information except those with label of “ActivityManager” and priority of “Info” above, label of “MyApp” and priority of “Debug” above.

- Only output the logcat with the specific label
  For example:

```
```

or

```
```

Note: only output the logcat with label of WishTV.

- Only output the logcat with the specific priority and label
  For example:

```
```

Note: only output the logcat with priority of I and label of WishTV.

Procrank tool

Procrank is a debugging tool with Android, running in the shell environment of the device, used to output the memory snapshot of the process in order to effectively observe the memory usage status of the process.
Include the following memory information:

- VSS：Virtual Set Size The memory size used by virtual (including the memory used by the shared lib)
- RSS：Resident Set Size The actually used physical memory size (including the memory used by the shared lib)
- PSS：Proportional Set Size The actually used physical memory size (allocate the memory used by the shared lib in proportion)
- USS：Unique Set Size The physical memory used exclusively by the process (not including the memory used by the shared lib)
  Note:
- USS size represents the memory size only used by the process, and it will be completely recovered after the process is killed.
- VSS/RSS includes the memory used by the shared lib, so it is not helpful to check the memory status of the single process.
- PSS is the shared memory status used by the specific single process after the shared memory is allocated in proportion.

Use procrank

Make sure the terminal has the root authority before executing procrank
su
The command format:

```
```

The commonly used command instructions:
-v：order by VSS
-r：order by RSS
-p：order by PSS
-u：order by USS
-R：convert to order by increasing[decreasing] method
-w：only display the statistical count of working set

-W：reset the statistical count of working set
-h：help
For example:
Output the memory snapshot:

```
```

Output the memory snapshot in VSS decreasing order:

```
```

Procrank is output in PSS order by default.

Search the specific content information

Use below command format to view the memory status of the specific process:

```
```

cmdline means the target application name, PID means the target application process.
Output the memory status used by systemUI process:

```
```

or:

```
```

Trace the process memory status

Analyze if there is memory leakage in the process by tracing the memory usage status. Use the script to continuously output the process memory snapshot, and compare with USS segment to see if there is memory leakage in this process.
For example: output the application memory usage of the process named com.android.systemui to see if there is leakage:
1、Write the script test.sh

```
```

2、After connect to the device by adb tool, run the script: ./test.sh

Dumpsys tool

Dumpsys tool is a debugging tool in Android system, running in the shell environment of the device, and provides the service status information running in the system. The running service means the service process in the Android binder mechanism.
The conditions for dumpsys to output the print:
1、Only print the services already loaded to ServiceManager.
2、If the dump function in the service code is not implemented, there will be no information output.

Use Dumpsys

- View Dumpsys help
  Function: output dympsys help information.

```
```

- View the service list of Dumpsys
  Function: output all the printable service information of dumpsys, developer can pay attention to the service names required for debugging.

- Output the specific service information
  Function: output the specific service dump information.
  Format: dumpsys [servicename]
  For example: execute below command can output the service information of SurfaceFlinger:

- Output the specific service and application process information
  Function: output the specific service and application process information.
  Format: dumpsys [servicename] [application name]
  For example: execute below command to output the memory information for the service named meminfo and process named com.android.systemui:

Note: the service name is case sensitive and must input the full service name.

Last log enable

- Add the following two nodes in dts file

- Usage:

FIQ mode

You can input fiq command through the serial port to check the system status when the device crashes or gets stuck. The specific command is as below:

Common issues

What is current kernel version and u-boot version?

How to acquire the corresponding RK release version for current SDK

Rockchip Android12.0 SDK includes AOSP source code and RK changed code. RK changed libs are involved in xml under the directory `.repo/manifests/include`, while AOSP default libs are in `.repo/manifests/default.xml`。
Version confirm:

- RK modification part

```
```

Means RK version is android-12.0-mid-rkr1

- AOSP part

```
```

Means OASP version is android-12.0.0_r2
Just provide the above two version information when needed.
You can directly acquire tag information through the following command for single lib:

```
```

RK version is incremental with the format of android-12.0-mid-rkrxx, so current latest tag is android-12.0-mid-rkr1

How to confirm if local SDK is already updated to the latest SDK version released by RK

When RK SDK is released, the commit information corresponding to the version will be submited under the .repo/manifests/commit/ directory. Customers can confirm whether SDK is updated completely or not by comparing with the commit information. The specific operations are as follows:

- First confirm RK version of SDK according to the instruction of "How to acquire the corresponding RK release version for current SDK". Below take RKR6 version as example to introduce.
- Use the following command to save local commit information

```
```

- Comparing .repo/manifests/commit/commit_release_rkr1.xml with release_manifest_rkr1_local.xml can confirm whether SDK code is completely updated or not, while .repo/manifests/commit/commit_release_rkr6.xml is the commit information released along with RKR1 version.

Replace uboot and kernel logo picture

uboot and kernel logo are the first and second logo picture displayed during bootup, and they can be changed according to the product requirement.
uboot logo source file: `kernel/logo.bmp`
kernel logo source file: `kernel/logo_kernel.bmp`
If need to change the picture, just use the bmp with the same name to replace, and re-compile kernel. The compiled file is in boot.img.
Note: Logo picture size currently only supports to 8M with 8, 16, 24, 32bit bmp format.

Power off charging and low battery precharging

Power off charging and low battery precharging can be configured in dts, as shown below:

Note:
rockchip,uboot-charge-on: uboot power off charging is mutually exclusive with android power off charging
rockchip,android-charge-on: android power off charging is mutually exclusive with uboot power off charging
rockchip,uboot-low-power-voltage: configure the voltage for low battery precharging to boot, it can be configured according to the actual requirement
rockchip,screen-on-voltage: configure the voltage for low battery precharging to light the panel, it can be configured according to the actual requirement

Uboot charging logo package and replace

Charging logo path, you can directly replace with the file with the same name, and the format should be the same as original file.

If uboot charging is enabled, but there is no charging logo displaed, maybe it is because the picture is not packaged into resource.img. You can package per the following command:

After executing the above command, uboot charging logo will be packaged into resource.img in kernel directory. Now need to re-package resource.img into boot.img. You can execute ./mkiamge.sh in android root directory, and then flash boot.img under rockdev/.

HDMI IN configuration

hdmi in function is disabled in SDK by default. If need to enable, operate as below:

RM310 4G configuration

4G function is disabled in SDK by default. If need to enable, operate as below:

Recovery rotation configuration

Support Recovery rotation with 0/90/180/270 degree. Disabled by default（that is to rotate 0 degree）. The rotation configuration is described as below:

Android Surface rotation

For Android system display rotation, you can modify the following configuration with the parameters 0/90/180/270

Replace some remote of AOSP source code

The speed for customer to download RK release code is relatively slow. You can change the remote of AOSP to domestic mirror source, or Google mirror source for foreign customers, to improve the downloading speed. The detailed method is described as below:
After executing repo init (or unpacking base package), modify .repo/manifests/remote.xml.
Change the remote fetch of AOSP from

to
for domestic customers: (here we take Tsinghua university mirror source as example. You can change to other domestic mirror source)

for foreign customers: (Google mirror source)

Data area read and write performance optimization

For devices with batteries, advised to add 'fsync_mode=nobarrier' to the data partition mounting parameter of fstab to improve storage read/write rates and performance. This parameter may cause data damage on devices without batteries. Therefore, it is not recommended to add this parameter to devices without batteries. Modified patches as follows:

Change userdata partition file system to EXT4

The default file system of data partition is f2fs. Recommend to change the file system of data partition to ext4 for the product without battery, as it can reduce the risk of data loss after abnormal power down. The modification method is as below:
Take rk3566_r as example:

Modify power on/off animation and tones

Reference document:

APP performance mode setting

Configure the file: package_performance.xml in device/rockchip/rk3xxx/. Add the package names which need to use performance mode in the node: (use aapt dump badging (file_path.apk) to acquire the package name)

Take antutu as example as below:

It will package the file into the image when compiling.

Debugging method of GPU related issues

You can do the initial debugging for the issues referring to the following documents.

OTP and efuse instruction

OTP support chipset

- RK3326
- PX30
- RK3566
- RK3568
- RK3588
- RK3568S
  EFUSE support chipset
- RK3288
- RK3368
- RK3399
  Refer to the document for image signing and otp/efuse flashing:

How to judge from the code whether OTP/EFUSE of the device is already flashed or not

The status of OTP/EFUSE will be transmitted through kernel cmdline and fuse.programmed in cmdline is used to mark the status of OTP/EFUSE. The details are as follows:

- "fuse.programmed=1": the software image package is already signed by secure-boot and efuse/otp of the hardware device is already flashed.
- "fuse.programmed=0": the software image package is already signed by secure-boot but efuse/otp of the hardware device is not flashed.
- there is no fuse.programmed in cmdline: the software image package is not signed by secure-boot (Miniloader doesn't tramsmit), or Miniloader is too old to support transmission.

Enable/disable selinux

Refer to the following modification, false to disable, true to enable

Warning “There's an internal problem with your device.” pops up after boot up

There are two reasons to pop up the warning:

1. Image mismatch, the fingerprints of system/boot/vendor are not consistent.
2. The device is enabled with a configuration that supports IO debugging. This problem can be solved by using the previous compile kernel command in the documentation.

3. For projects with IO debugging, regardless of the above two reasons, please merge the following patches directly to eliminate the warning:

How to enable the setting options for Ethernet in Settings

There is no default option of Ethernet setting in the system Settings. If Ethernet is needed in the project, it can be turned on as follows:

About AVB and security boot

For AVB and security boot related instruction and configurations, you can refer to the document

Cannot use IO commands

IO commands rely on DEVMEM which is disabled by default, so it is not able to use IO by default. If need to use IO commands for debugging, you can modify as follows:

For GO produts, need to modify:

delete the following line:

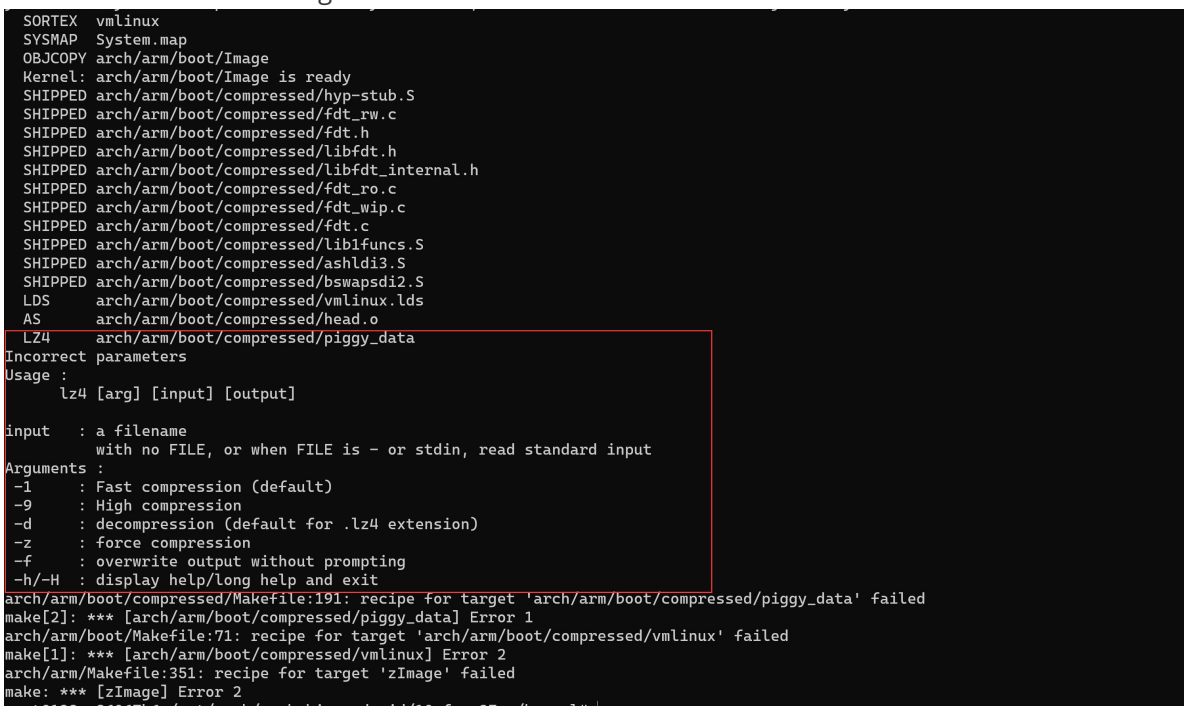
If you want to compile Android, you also need to modify the following code

The SN command rules

The SN must begin with a letter and be no more than 14 bytes.

RK3288 build failer about LZ4

RK3288 build kernel fail log as:



problem:

The LZ4 version of the system is too low, and the version 1.8.3 or above is required

solution:

copy the LZ4 compiled by Android to override the LZ4 of the system

Android Samba function

reference file

NFS boot

Refer to the documents and patches:

Multi-screen display and touch

Referenced document

Different screen different sound

Referenced document

APPENDIX A Compiling and development environment setup

Initializing a Build Environment

This section describes how to set up your local work environment to build the Android source files. You must use Linux or Mac OS; building under Windows is not currently supported.
For an overview of the entire code-review and code-update process, see Life of a Patch.
Note: All commands in this site are preceded by a dollar sign ($) to differentiate them from output or entries within files. You may use the Click to copy feature at the top right of each command box to copy all lines without the dollar signs or triple-click each line to copy it individually without the dollar sign.

Choosing a Branch

Some requirements for the build environment are determined by the version of the source code you plan to compile. For a full list of available branches, see Build Numbers. You can also choose to download and build the latest source code (called master), in which case you will simply omit the branch specification when you initialize the repository.
After you have selected a branch, follow the appropriate instructions below to set up your build environment.

Setting up a Linux build environment

These instructions apply to all branches, including master.
The Android build is routinely tested in house on recent versions of Ubuntu LTS (14.04) and Debian testing. Most other distributions should have the required build tools available.
For Gingerbread (2.3.x) and newer versions, including the master branch, a 64-bit environment is required. Older versions can be compiled on 32-bit systems.
Note: See Requirements for the complete list of hardware and software requirements, then follow the detailed instructions for Ubuntu and Mac OS below.

Installing the JDK

The master branch of Android in the Android Open Source Project (AOSP) comes with prebuilt versions of OpenJDK below prebuilts/jdk/ so no additional installation is required.
Older versions of Android require a separate installation of the JDK. On Ubuntu, use OpenJDK. See JDK Requirements for precise versions and the sections below for instructions.
For Ubuntu >= 15.04
Run the following:

```
```

For Ubuntu LTS 14.04
There are no available supported OpenJDK 8 packages for Ubuntu 14.04. The Ubuntu 15.04 OpenJDK 8 packages have been used successfully with Ubuntu 14.04. Newer package versions (e.g. those for 15.10, 16.04) were found not to work on 14.04 using the instructions below.

1. Download the .deb packages for 64-bit architecture from old-releases.ubuntu.com:

```
```

2. Optionally, confirm the checksums of the downloaded files against the SHA256 string listed with each package above. For example, with the sha256sum tool:

3. Install the packages:

Run dpkg for each of the .deb files you downloaded. It may produce errors due to missing dependencies:

To fix missing dependencies:

Update the default Java version - optional
Optionally, for the Ubuntu versions above update the default Java version by running:

Note: If, during a build, you encounter version errors for Java, see Wrong Java version for likely causes and solutions.
Installing required packages (Ubuntu 14.04)
You will need a 64-bit version of Ubuntu. Ubuntu 14.04 is recommended.

Note: To use SELinux tools for policy analysis, also install the python-networkx package. Note: If you are using LDAP and want to run ART host tests, also install the libnss-sss:i386 package.

Configuring USB Access

Under GNU/linux systems (and specifically under Ubuntu systems), regular users can't directly access USB devices by default. The system needs to be configured to allow such access.
The recommended approach is to create a file /etc/udev/rules.d/51-android.rules (as the root user) and to copy the following lines in it. must be replaced by the actual username of the user who is authorized to access the phones over USB.

Those new rules take effect the next time a device is plugged in. It might therefore be necessary to unplug the device and plug it back into the computer.
This is known to work on both Ubuntu Hardy Heron (8.04.x LTS) and Lucid Lynx (10.04.x LTS). Other versions of Ubuntu or other variants of GNU/linux might require different configurations.
References : http://source.android.com/source/initializing.html

APPENDIX B SSH public key operation instruction

APPENDIX B-1 SSH public key generation

Use the following command to generate:

Please replace user@host with your email address.

It will generate the key file in your directory after the command is executed successfully.

Please keep carefully the generated private key file id_rsa and password, and send id_rsa.pub to SDK release server admin through email.

APPENDIX B-2 Use key-chain to manage the key

Recommend you use the simple tool keychain to manage the key.
The detailed usage is as follows:

1. Install keychain software package:

2. Configure to use the key:

Add the following command:

Among which, id_rsa is the file name of the private key.
Log in the console again after configuring as above, and it will prompt to input the password.
Only need to input the password used for generating the key if there is one.
Besides, please avoid using sudo or root user unless you know clearly how to deal with, otherwise it will cause the authority and key management problems.

APPENDIX B-3 Multiple devices use the same ssh public key

In order to use on different devices, you can copy ssh private key file id_rsa to the target device "~/.ssh/id_rsa".
Below hint will show up if using the wrong private key. Please replace with the correct private key.

After adding the correct private key, you can use git to clone code, shown as below picture:

Below error may occur when adding ssh private key:

Input the following command at console can fix it.

APPENDIX B-4 Switch different ssh public keys on one device

You can refer to ssh_config document to configure ssh.

Use the following commands to configure ssh for current user.

```
```

As below picture, identify another directory ssh file ”~/.ssh1/id_rsa” as certificate private key. In this way, you can switch different keys.

APPENDIX B-5 Key authority management

The server can real-time monitor for the specific key the download times, IP and other information. If any abnormal case is found, it will prohibit the download authority of the corresponding key.
Please keep carefully the private key file. DO NOT re-authorize it to the third party.

APPENDIX B-6 Git authority application instruction

Refer to above chapters, generate the public key file, and send email to fae@rock-chips.com applying for SDK code download authority.