

Rockchip Android 12.0 SDK Developer Guide

<div>Status: <input type="checkbox"/> Draft <input checked="" type="checkbox"/> Released <input type="checkbox"/> Modifying</div>	File No.:	RK-KF-YF-300
	Current Version:	V1.0.0
	Author:	Wu Liangqing
	Finish Date:	2021-10-28
	Auditor:	Chen Haiyan
	Finish Date:	2021-10-31

Version no.	Author	Revision Date	Revision Description	Remark
V0.0.2	Wu Liangqing/Bian Jinchun	2021-10-28	release rkr1 support rk3568/rk3566/rk3326/px30	

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

DISCLAIMER

THIS DOCUMENT IS PROVIDED "AS IS". ROCKCHIP ELECTRONICS CO., LTD.("ROCKCHIP")DOES NOT PROVIDE ANY WARRANTY OF ANY KIND, EXPRESSED, IMPLIED OR OTHERWISE, WITH RESPECT TO THE ACCURACY, RELIABILITY, COMPLETENESS, MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE OR NON-INFRINGEMENT OF ANY REPRESENTATION, INFORMATION AND CONTENT IN THIS DOCUMENT. THIS DOCUMENT IS FOR REFERENCE ONLY. THIS DOCUMENT MAY BE UPDATED OR CHANGED WITHOUT ANY NOTICE AT ANY TIME DUE TO THE UPGRADES OF THE PRODUCT OR ANY OTHER REASONS.

Trademark Statement

"Rockchip", "瑞芯微", "瑞芯" shall be Rockchip's registered trademarks and owned by Rockchip. All the other trademarks or registered trademarks mentioned in this document shall be owned by their respective owners.

All rights reserved. ©2021. Rockchip Electronics Co., Ltd.

Beyond the scope of fair use, neither any entity nor individual shall extract, copy, or distribute this document in any form in whole or in part without the written approval of Rockchip.

Rockchip 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

Rockchip Android 12.0 SDK Chipset support

Chipset platform	Support or not	SDK version
RK3566	Support	RKR1
RK3568	Support	RKR1
PX30/RK3326	Support	RKR1

Rockchip 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/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/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

`Rockchip_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 Rockchip_Android12.0_SDK
cat Rockchip_Android12.0_SDK_Release.tar.gz* | tar -zx -C
Rockchip_Android12.0_SDK
cd Rockchip_Android12.0_SDK
.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 '~/.ssh/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 switched can be combined and submitted in this way

Code compiling

One key compiling command

Compiling command summary

Soc	type	model	Android	one key compiling	kernel compiling	uboot compiling
RK3566	tablet	device	build/envsetup.sh;lunch rk3566_sgo-userdebug	./build.sh - AUCKu	make ARCH=arm64 rockchip_defconfig android-11.config;make ARCH=arm64 rk3566-rk817-tablet.img -j24	./make.sh rk3566
RK3568	EVB	EVB1-DDR4-V10	build/envsetup.sh;lunch rk3568_s-userdebug	./build.sh - AUCKu	make ARCH=arm64 rockchip_defconfig rk356x_evb.config android-11.config;make ARCH=arm64 rk3568-evb1-ddr4-v10.img -j24	./make.sh rk3568
RK3568	EVB	EVB2-LPDDR4X-V10	build/envsetup.sh;lunch rk3568_s-userdebug	./build.sh - AUCKu	make ARCH=arm64 rockchip_defconfig rk356x_evb.config android-11.config;make ARCH=arm64 rk3568-evb2-lp4x-v10.img -j24	./make.sh rk3568
RK3566	EVB	EVB1-DDR4-V10	build/envsetup.sh;lunch rk3566_s-userdebug	./build.sh - AUCKu -d rk3566-evb2-lp4x-v10	make ARCH=arm64 rockchip_defconfig rk356x_evb.config android-11.config;make ARCH=arm64 rk3566-evb1-ddr4-v10.img -j24	./make.sh rk3566
RK3566	EVB	EVB2-LP4X-V10	build/envsetup.sh;lunch rk3566_s-userdebug	./build.sh - AUCKu -d rk3566-evb2-lp4x-v10	make ARCH=arm64 rockchip_defconfig rk356x_evb.config android-11.config;make ARCH=arm64 rk3566-evb2-lp4x-v10.img -j24	./make.sh rk3566
RK3326	tablet	863 device	build/envsetup.sh;lunch rk3326_sgo-userdebug	./build.sh - AUCKu	make ARCH=arm64 rockchip_defconfig android-11-go.config;make ARCH=arm64 rk3326-863-lp3-v10-rksip1.img -j24	./make.sh rk3326
PX30	EVB	px30-evb-ddr3-v10-avb	build/envsetup.sh;lunch PX30_Android12-userdebug	./build.sh - AUCKu	make ARCH=arm64 rockchip_defconfig android-11.config;make ARCH=arm64 px30-evb-ddr3-v10-avb.img -j24	./make.sh px30

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

The compiling principle: in kernel directory, replace the old `boot.img` with the newly compiled `kernel.img` and `resource.img`.

Take RK3566 prototype as an example, replace the corresponding boot.img and DTS at compile time:

The `BOOT_IMG = ../rockdev/image-rk3566_s/boot.img` is the path of the old boot.img. The command is as follows:

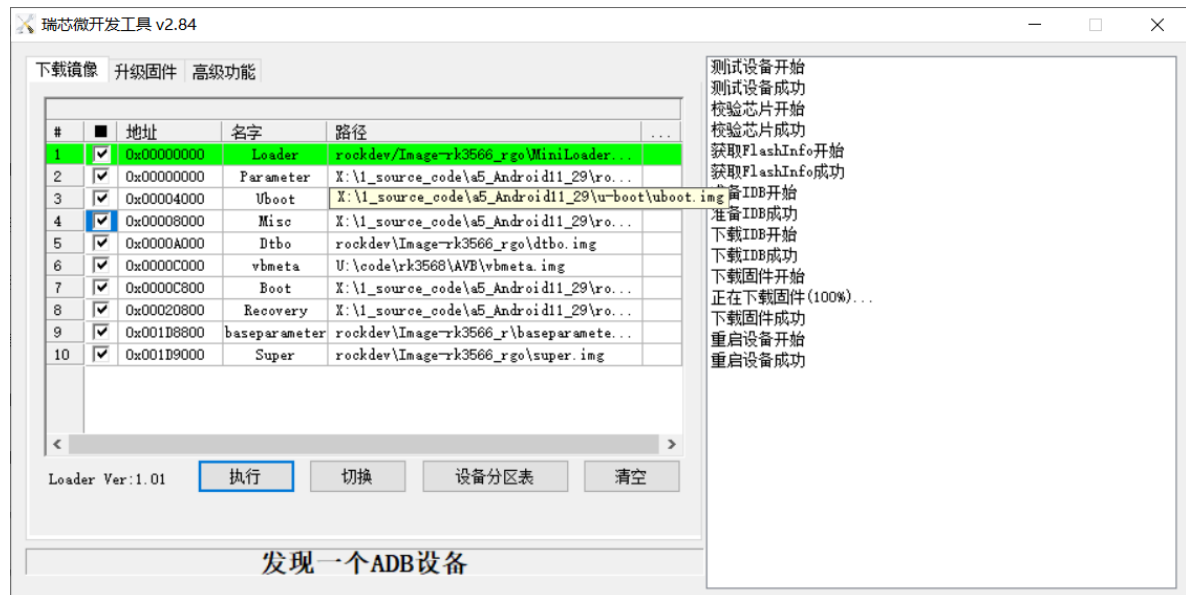
After compiling, you can directly flash the boot.img under kernel directory (**Note: it is zboot.img for 32bit platform, such as 3126c**) to the boot location of the device. **Please firstly load the partition table (parameter.txt)** before flashing, to prevent from flashing to the wrong place.

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: (take RK3566 as example, here lunch rk3566_rgo-userdebug)

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

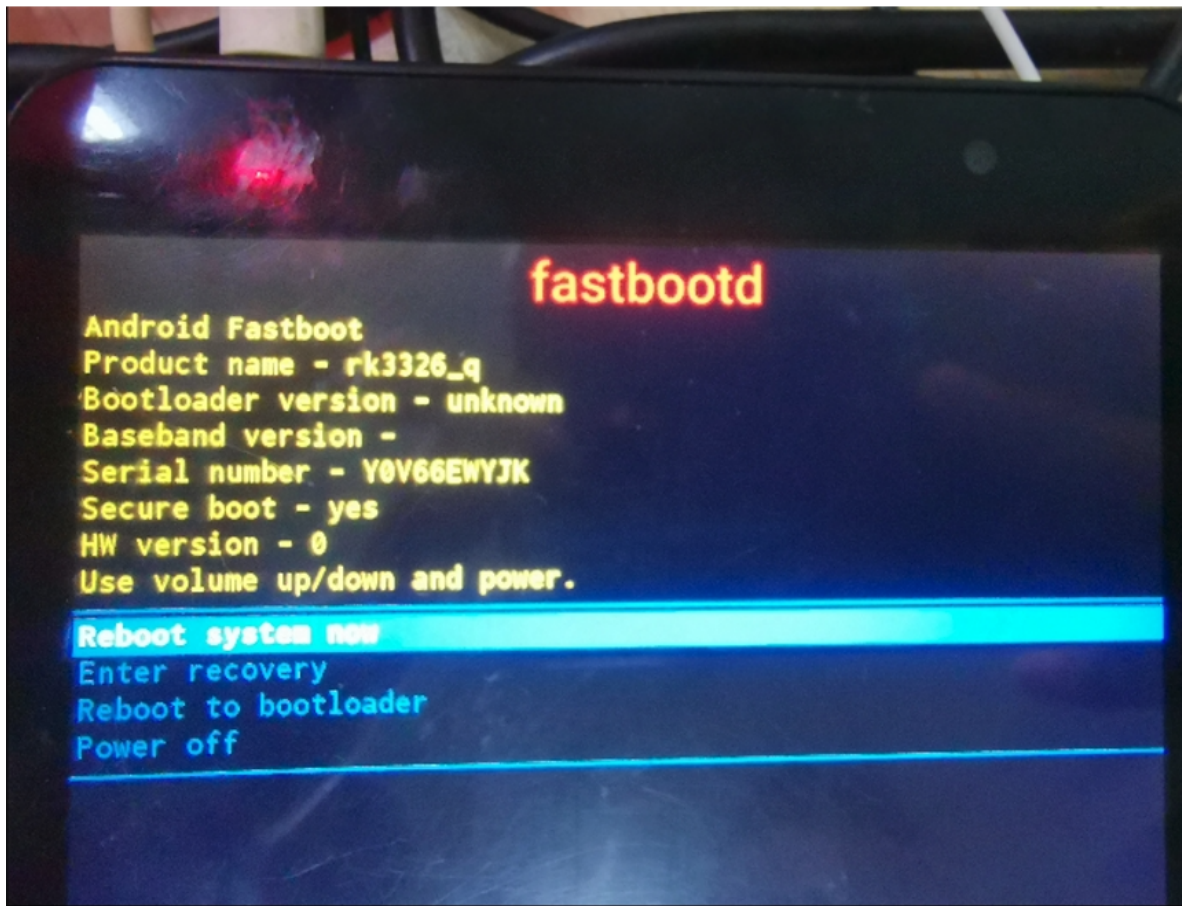
Image	Instruction
boot.img	including ramdis、 kernel、 dtb
boot-debug.img	the difference between boot.img and boot-debug.img is that user image can flash this boot.img to do root operation
dtbo.img	Device Tree Overlays refer to dtbo chapter instruction later
config.cfg	the configuration file of the flash 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
vbmata.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 rk3568_s product. The steps are as below:

1.Modify device/rockchip/rk356x/AndroidProducts.mk to add rk3568_s lunch

2.Create rk3568_s directory under device/rockchip/rk356x

Create referring to the existing rk3566_s product directory in device/rockchip/rk356x. You can directly copy rk3566_s to rk3568_s, and then replace all the `rk3566_s` under rk3568_r directory with `rk3568_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
RK3566	RK817	LPDDR4	tablet	device	rk3566-rk817-tablet
RK3566	RK809	LPDDR4	development board	RK3566 EVB2	rk3566-evb2-lp4x-v10
RK3568	RK809	DDR4	development board	RK3568 EVB1	rk3568-evb1-ddr4-v10
RK3566	discrete device	DDR4	development board	RK3566 BOX DEMO	rk3566-box-demo-v10
RK3326	RK817	DDR3	development board	evb	rk3326-evb-lp3-v10-avb
PX30	RK809	DDR3	development board	evb	px30-evb-ddr3-v10-avb
RK3326	RK817	DDR3	tablet	device	rk3326-863-lp3-v10-rkisp1
RK3326	RK809	DDR3	AI	EVB	rk3326-evb-ai-va-v12

Document instruction

Peripheral support list

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

[Link](#)

Android document

[Link](#)

Android_SELinux(Sepolicy) developer guide

[Link](#)

Android 12 System Optimization developer guide

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

[Link](#)

Wi-Fi document

[Link](#)

3G/4G module instruction document

[Link](#)

Kernel document

[Link](#)

DDR related document

[Link](#)

Audio module document

[Link](#)

CRU module document

[Link](#)

GMAC module document

[Link](#)

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

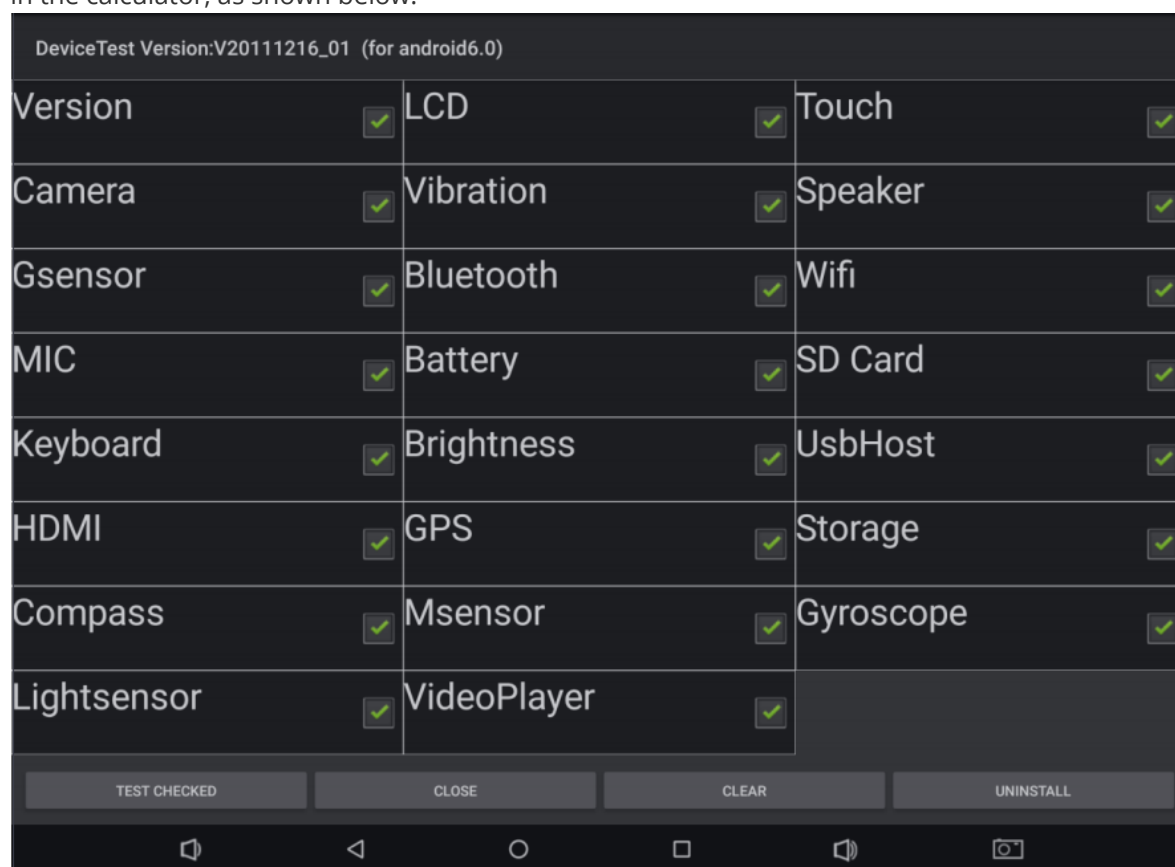
PCBA test tool

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:



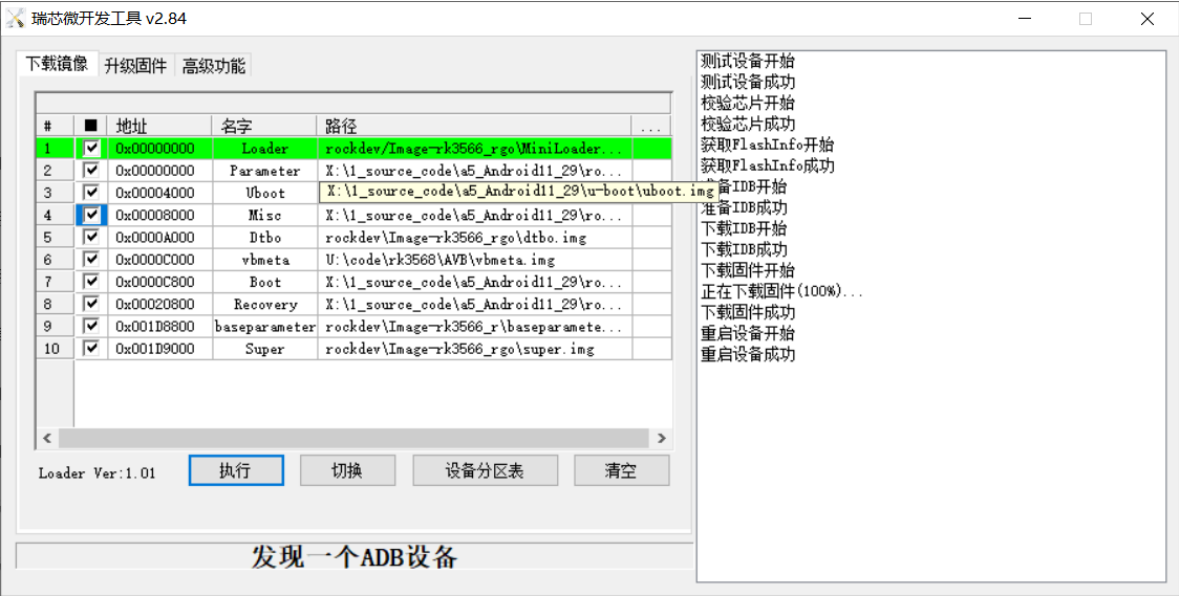
In factory, you can test the corresponding peripheral according to this interface. Click “TEST CHECKED” to test the items one by one. If succeed, click pass, if fail, click failed, the final result will display on the screen, as shown below. Red means failed item, others are pass, and the factory can 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



Linux version

RKTools/linux/Linux_Upgrade_Tool/Linux_Upgrade_Tool_v1.56.zip

Tool to implement SD upgrading and boot

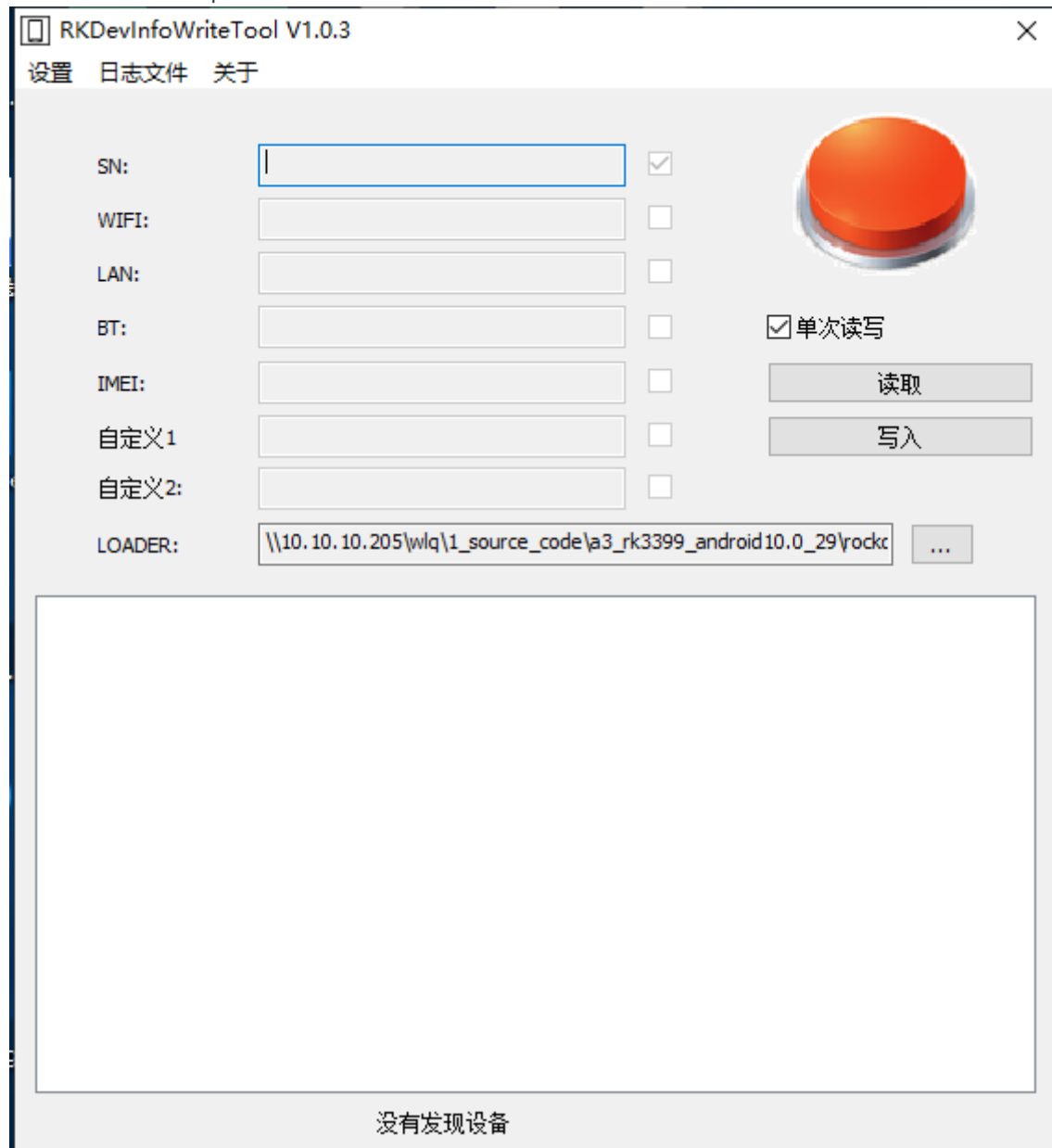
It is used to implement SD card upgrading, SD card boot, SD card PCBA test.

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 connection, 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 used 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

Enter 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 dumpsys 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 submitted 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 displayed, 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/.

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)

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

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 transmit), 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 products, need to modify:

delete the following line:

delete the following line:

Both of them need to be deleted

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:

```
SORTEX vmlinux
SYSMAP System.map
OBJCOPY arch/arm/boot/Image
Kernel: arch/arm/boot/Image is ready
SHIPPED arch/arm/boot/compressed/hyp-stub.S
SHIPPED arch/arm/boot/compressed/fdt_rw.c
SHIPPED arch/arm/boot/compressed/fdt.h
SHIPPED arch/arm/boot/compressed/libfdt.h
SHIPPED arch/arm/boot/compressed/libfdt_internal.h
SHIPPED arch/arm/boot/compressed/fdt_ro.c
SHIPPED arch/arm/boot/compressed/fdt_wip.c
SHIPPED arch/arm/boot/compressed/fdt.c
SHIPPED arch/arm/boot/compressed/lib1funcs.S
SHIPPED arch/arm/boot/compressed/ashldi3.S
SHIPPED arch/arm/boot/compressed/bswapdi2.S
LDS arch/arm/boot/compressed/vmlinux.lds
AS arch/arm/boot/compressed/head.o
LZ4 arch/arm/boot/compressed/piggy_data
Incorrect parameters
Usage :
    lz4 [arg] [input] [output]

input : a filename
        with no FILE, or when FILE is - or stdin, read standard input
Arguments :
    -1 : Fast compression (default)
    -9 : High compression
    -d : decompression (default for .lz4 extension)
    -z : force compression
    -f : overwrite output without prompting
    -h/-H : display help/long help and exit
arch/arm/boot/compressed/Makefile:191: recipe for target 'arch/arm/boot/compressed/piggy_data' failed
make[2]: *** [arch/arm/boot/compressed/piggy_data] Error 1
arch/arm/boot/Makefile:71: recipe for target 'arch/arm/boot/compressed/vmlinux' failed
make[1]: *** [arch/arm/boot/compressed/vmlinux] Error 2
arch/arm/Makefile:351: recipe for target 'zImage' failed
make: *** [zImage] Error 2
```

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

Fail to boot up or unable to use the commands of reboot loader on the device with battery after upgrading to RKR7 version or above

Issue analysis:

There is the following commit in kernel with RKR7 version or above, which is used to fix the issue of unable to power off with probability.

Solution:

Power down once to reset pmic.

You can input the following command through serial port or ADB for the device with battery to clear PMIC.

Confirm GPIO voltage configuration of IO-Domain, it will damage GPIO of the chipset if GPIO voltage is not configured correctly

GPIO voltage of RK3566/RK3568 IO-Domain is configured as 3.3V by default, and it should be configured according to the actual hardware schematic. Take RK3566/RK3568 EVB board as an example:

Default configuration of IO-Domain:

According to the actual hardware schematic, it should be changed to:

IO-Domain configuration process is as follows:

Step1: Get the hardware schematic and confirm the power design solution.

Here we take RK_EVB1_RK3568_DDR4P216SD6_V10_20200911 EVB board as an example to introduce.

Hardware schematic:RK_EVB1_RK3568_DDR4P216SD6_V10_20200911.pdf

Power solution: As seen from the hardware schematic, EVB board

RK_EVB1_RK3568_DDR4P216SD6_V10_20200911 has PMU (RK809-5).

Step2: Find the corresponding kernel dts configuration file.

As we know from step1, the power design of this EVB board is with PMU solution, and the corresponding kernel dts configuration file is located in:

arch/arm64/boot/dts/rockchip/rk3568-evb.dtsi (the solution discussed in this document)

Step3: Modify the power domain configuration node pmu_io_domains in kernel dts.

Take vccio1-supply as an example, firstly check the hardware schematic to confirm the configuration of vccio1 power domain (VCCIO1) as shown in the figure.

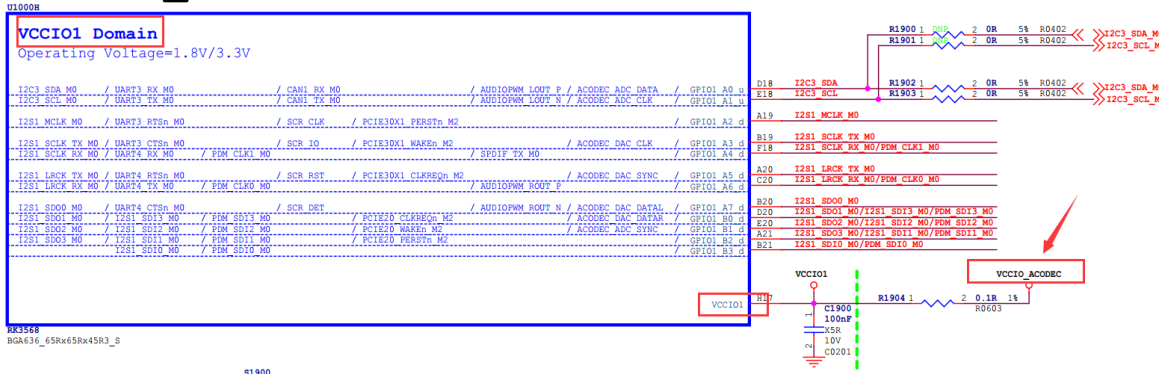
IO Power Domain Map

Updates must be Revision accordingly!

IO Domain	Pin Num	Support IO Voltage		Actual assigned IO Domain Voltage			Notes
		3.3V	1.8V	Supply Power Net Name	Power Source	Voltage	
PMUIO1	Pin Y20	✓	✗	VCC3V3_PMU	VCC3V3_PMU	3.3V	
PMUIO2	Pin W19	✓	✓	VCC3V3_PMU	VCC3V3_PMU	3.3V	
VCCIO1	Pin H17	✓	✓	VCCIO_ACODEC	VCCIO_ACODEC	3.3V	
VCCIO2	Pin H18	✓	✓	VCCIO_FLASH	VCC_1V8	1.8V	PIN "FLASH_VOL_SEL" must be logic High if VCCIO_FLASH=3.3V,FLASH_VOL_SEL must be logic low
VCCIO3	Pin L22	✓	✓	VCCIO_SD	VCCIO_SD	3.3V	
VCCIO4	Pin J21	✓	✓	VCCIO4	VCC_1V8	1.8V	
VCCIO5	Pin V10 Pin V11	✓	✓	VCCIO5	VCC_3V3	3.3V	
VCCIO6	Pin R9 Pin U9	✓	✓	VCCIO6	VCC_1V8	1.8V	
VCCIO7	Pin V12	✓	✓	VCCIO7	VCC_3V3	3.3V	

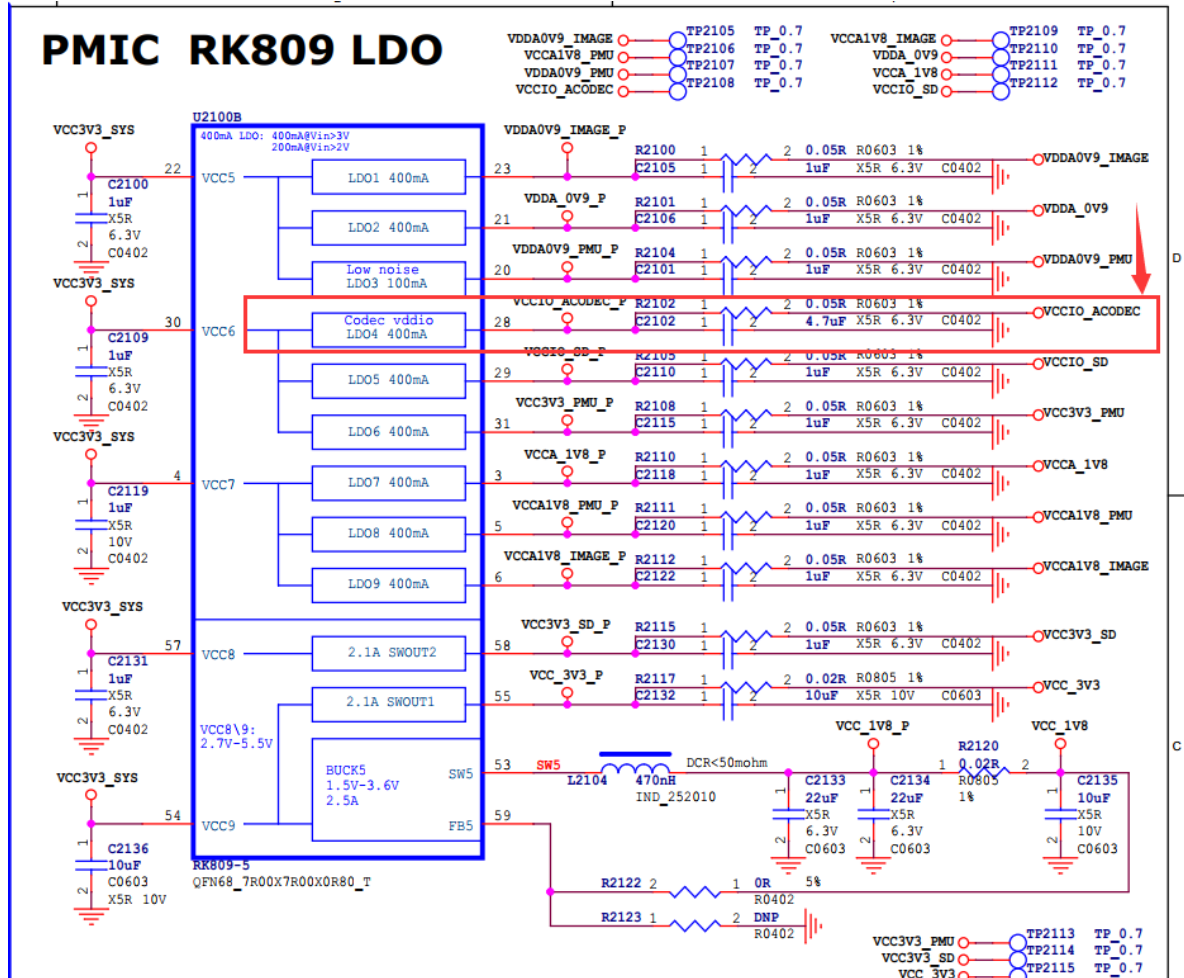
Search 'VCCIO1'in the hardware schematic as follows:

RK3568_H (VCCIO1 Domain)



We can find the power of `VCCI01` is `vccio_acodec` as in the schematic above .

Search `vccio_acodec` in the schematic, you can find the result as follows:



The power of `vccio_acodec` is supplied by LD04 of RK809.

Find the configuration information of LDO_REG4 (LDO4) in software DTS as follows.

Configure `vccio_acodec` above to 'vccio1-supply = <&vccio_acodec>' of `pmu_io_domains` node, then the voltage configuration of `vccio1` is completed.

Just configure for other power domains as above, among which the voltage of VCCIO2 is fixed by hardware and no need to configure.

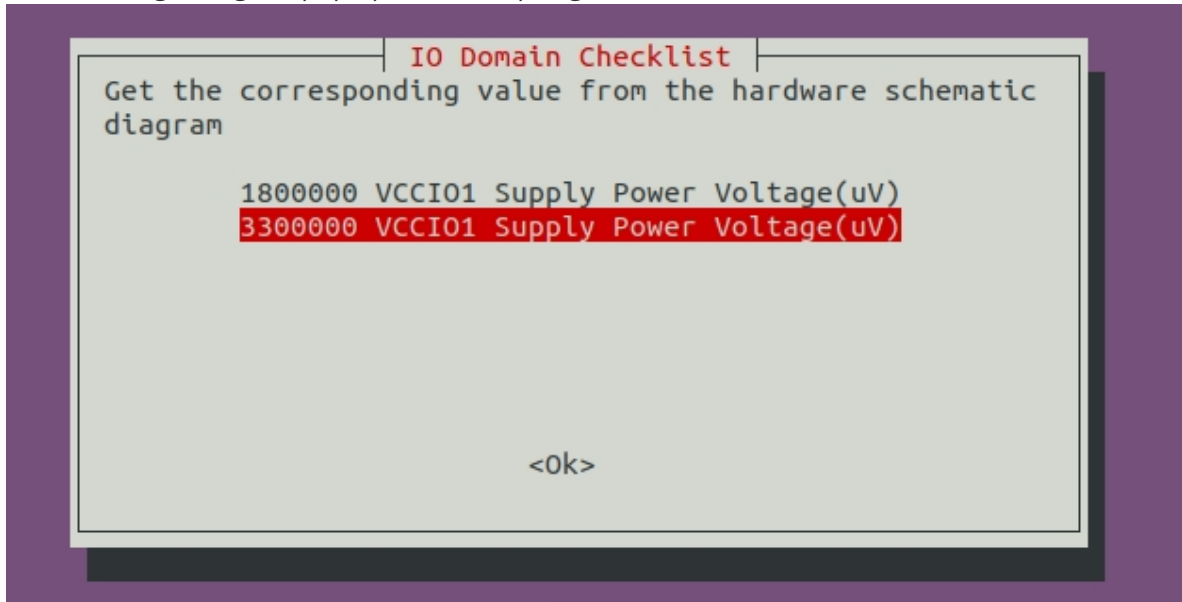
- pmuio2
- vccio1
- vccio3

- vccio5
- vccio6
- vccio7

Just configure the GPIO voltage as above.

IO-Domain confirmation dialog pop-up while compiling RK356X kernel

The following dialog will pop up while compiling kernel:



The purpose of the pop-up dialog is to check whether the GPIO voltage configured in software dts matches with the actual hardware schematic. You need to configure the GPIO voltage according to the actual hardware schematic. It's very important to check and confirm the configuration with your hardware engineers if you are a software engineer. If the GPIO voltage is not configured correctly, GPIO of the chipset will be damaged.

This dialog will not pop up after you confirm the GPIO voltage correctly, but it will pop up again for you to confirm if dts name or io-domain in the dts is changed.

Related questions of RK356x PCIE module

If the PCIE3.0 port does not provide an external clock to the chip but enables this interface in the dts, it will cause the boot jam problem.

Because PCIE3.0 requires external clock supply, if PCIE3.0 is enabled in the dts, but the clock is not provided to RK356x PCIE3.0 module in the exterior (for example, the external clock module is not attached, or the clock module works abnormally), the system will be stuck. The problem log is as follows:

When the log appears, if there is an external clock module attached, then check whether the module is working properly; If not, you should indicate that the PCIE3.0 port is not required, and the following items need to be disabled in the dts:

rk356x pcie2x1 controller and sata2 controller can not be turned on at the same time.

Since rk356x pcie2x1 controller and sata2 controller are multiplexed, they cannot be turned on at the same time, otherwise it will cause the interfere with each other and anomalies.

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:

```
sudo apt-get install openjdk-8-jdk
```

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:

```
wget http://old-releases.ubuntu.com/ubuntu/pool/universe/o/openjdk-8-jdk/openjdk-8-jdk_8u60-b1-1ubuntu1~14.04_amd64.deb
```

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

```
sha256sum openjdk-8-jdk_8u60-b1-1ubuntu1~14.04_amd64.deb
```

3. Install the packages:

```
sudo dpkg --get-selections | grep openjdk-8-jdk
```

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

```
sudo dpkg --get-selections | grep openjdk-8-jdk
```

To fix missing dependencies:

```
sudo dpkg --get-selections | grep openjdk-8-jdk
```

Update the default Java version - optional

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

```
sudo update-java-alternatives -s java-8-openjdk-amd64
```

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.

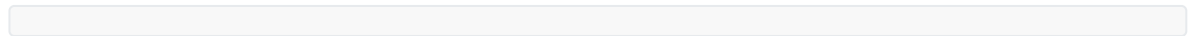
APPENDIX B-3 Multiple devices use the same ssh public key

In order to use on different devices, you can copy ssh private key file id_rsa to the target device "~/ssh/id_rsa".

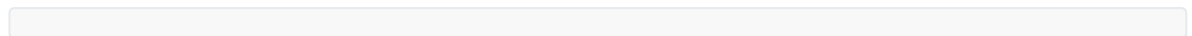
Below hint will show up if using the wrong private key. Please replace with the correct private key.

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:

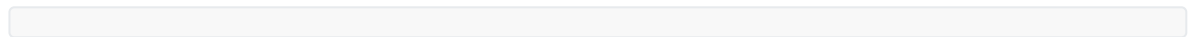


Input the following command at console can fix it.

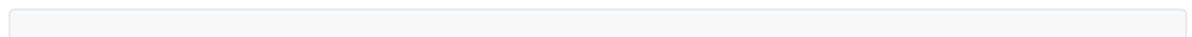


APPENDIX B-4 Switch different ssh public keys on one device

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.