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

Rockchip Android 11.0 SDK Developer Guide

Status: [ ] Draft [√] Released [ ] Modifying	File No.:	RK-KF-YF-282
	Current Version:	V1.1.6
	Author:	Wu Liangqing
	Finish Date:	2021-08-31
	Auditor:	Chen Haiyan
	Finish Date:	2021-08-31

Version no.	Author	Revision Date	Revision Description	Remark
V0.0.2	Wu Liangqing/Bian Jinchen	2021-01-06	Initial version release	
V1.0.0	Wu Liangqing/Bian Jinchen	2021-01-29	add rk3566/rk3568 evb board compile way	
V1.1.0	Wu Liangqing/Bian Jinchen	2021-02-23	RK3399 alpha release	
V1.1.2	Wu Liangqing/Bian Jinchen	2021-05-13	support rk3288w	
V1.1.3	Wu Liangqing	2021-05-23	Add Frequently Asked Questions (FAQ)	
V1.1.4	Wu Liangqing	2021-07-12	support RK3566 BOX products, support RK3328 BOX products, add repo server erection, Add Frequently Asked Questions (FAQ)	
V1.1.5	Wu Liangqing	2021-08-31	Add Frequently Asked Questions (FAQ)	
V1.1.6	Wu Liangqing	2022-01-18	Added data area performance optimization description	

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

DISCLAIMER

Trademark Statement

All rights reserved. ©2020. Rockchip Electronics Co., Ltd.

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 11.0 SDK Chipset support

Chipset platform	Support or not	SDK version
RK3566	Support	RKR1
RK3568	Support	RKR1
PX30/RK3326	Support	RKR1
RK3399	Support	RKR5
RK3288W	Support	RKR7
RK366-BOX	Support	RKR9
RK3328	Support	RKR9

Rockchip Android 11.0 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_R/manifests.git -m Android11.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_R/manifests.git -m Android11.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_Android11.0_SDK_Release.tar.gz.*` as an example, you can sync the source code through the following command after getting the initial package:

```
mkdir Rockcip_Android11.0_SDK
cat Rockchip_Android11.0_SDK_Release.tar.gz* | tar -zx -C
Rockcip_Android11.0_SDK
cd Rockcip_Android11.0_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 '~/ 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:

```
gitolite/install -to ~/bin
```

8. Set the administrator.

```
~/bin/gitolite setup -pk YourName.pub
```

Client-side operation

1. Clone gitolite management warehouse of the server;

```
git clone ssh://git@10.10.10.206/gitolite-admin.git
```

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

```
cp username.pub keydir/username.pub
```

3. Add an administrator user.

```
vi conf/gitolite.conf
@admin = admin1 admin2 admin3
repo gitolite-admin
RW+     =   @admin
```

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;

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

3. Create a new RK_Android11_mirror directory;

```
mkdir RK_Android11_mirror
```

4. Enter the RK_Android11_mirror directory;

```
cd RK_Android11_mirror
```

5. Download RK Android11 SDK mirror;

```
~/repo/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_R/manifests.git -m Android11.xml --mirror
```

6. Create warehouse group permissions.

```
.repo/repo/repo list -n > android_r.conf
sed -i 's/^/@android_r = RK_Android11_mirror\/&/g' android_r.conf
```

Client-side operation

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

```
vi conf/android_r.conf
@usergroup = user1 user2 user3
repo @android_r
R   = @usergroup
RW+ = @admin
```

```
vi conf/gitolite.conf
include "android_r.conf"
```

3. Create your own new manifests warehouse.

```
vi conf/android_r.conf
@android_r = Android_R/manifests_xxx
```

Client-side operation

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

```
git clone ssh://git@10.10.10.206/Android_R/manifests_xxx.git
```

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

```
git clone ssh://git@10.10.10.206/Android_R/manifests.git
```

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

```
cd manifests_xxx
cp -rf manifests/*.xml manifests_xxx/
```

check copy files

```
git status

 Android11.xml
 Android11_Express.xml
```

```
default.xml
include/partner_gms_express.xml
include/partner_modules_express.xml
include/rk3288_repository.xml
include/rk3326_repository.xml
include/rk3399_repository.xml
include/rk356x_repository.xml
include/rk_checkout_from_aosp.xml
include/rk_modules_repository.xml
remote.xml
remove_r.xml
```

Local commits

```
git add -A
git commit -m "init xxx"
```

Push to the remote branch

```
git push origin master:master
```

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

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

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

```
mkdir Android11
cd Android11
~/repo/repo init -u ssh://git@10.10.10.206/Android_R/manifests_xxx.git -m
Android11.xml
```

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;

```
cd kernel
```

2. Switch a local branch;

```
git checkout remotes/m/master -b xxx_branch
```

3. Push xxx_branch to remote server;

```
git push rk29 xxx_branch:xxx_branch
```

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

```
cd .repo/manifests
```

```
--- a/include/rk_modules_repository.xml
+++ b/include/rk_modules_repository.xml
@@ -10,7 +10,7 @@
   <project path="hardware/rockchip/libgraphicpolicy"
name="rk/hardware/rk29/libgraphicpolicy" remote="rk"
revision="refs/tags/android-11.0-mid-rkr8" />
   <project path="hardware/rockchip/libhwjpeg" name="rk/hardware/rk29/libhwjpeg"
remote="rk" revision="refs/tags/android-11.0-mid-rkr8"/>
   <project path="u-boot" name="rk/u-boot" remote="rk"
revision="refs/tags/android-11.0-mid-rkr8"/>
-  <project path="kernel" name="rk/kernel" remote="rk29"
revision="refs/tags/android-11.0-mid-rkr8"/>
+  <project path="kernel" name="rk/kernel" remote="rk29" revision="xxx_branch"/>

   <project path="bootable/recovery/rkupdate"
name="platform/bootable/recovery/rk_update" remote="rk"
revision="refs/tags/android-11.0-mid-rkr8"/>
   <project path="bootable/recovery/rkutility"
name="platform/bootable/recovery/rk_utility" remote="rk"
revision="refs/tags/android-11.0-mid-rkr8"/>
```

5. Submit the modified manifest to the remote branch.

```
git add include/rk_modules_repository.xml
git commit -m "change kernel branch on xxx_branch"
git push origin default:master
```

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;

```
cd RK_Android11_mirror
```

```
.repo/repo/repo sync -c
```

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

- Download the original manifests warehouse of RK;

```
git clone //10.10.10.206/wlq/test/manifests.git
```

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

```
wlq@wlq:~/home1/test2/kernel$ git branch -av
* android-11.0-mid-rkr7   0bde59fad73a ARM: configs: rockchip_defconfig enable
ION_CMA_HEAP
  xxx_branch              0bde59fad73a ARM: configs: rockchip_defconfig enable
ION_CMA_HEAP
  remotes/m/master        -> rk29/xxx_branch
  remotes/rk29/xxx_branch 0bde59fad73a ARM: configs: rockchip_defconfig enable
ION_CMA_HEAP
```

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)

```
git checkout remotes/m/xxx_branch -b local_xxx_branch
```

- Check latest TAG published currently by RK

```
wlq@wlq:~/home1/test2/kernel$ git tag | grep rkr
android-10.0-mid-rkr1
android-10.0-mid-rkr10
android-10.0-mid-rkr11
android-10.0-mid-rkr13
android-10.0-mid-rkr2
android-10.0-mid-rkr3
android-10.0-mid-rkr4
android-10.0-mid-rkr5
android-10.0-mid-rkr6
android-10.0-mid-rkr7
android-10.0-mid-rkr8
android-10.0-mid-rkr9
android-11.0-ebook-rkr1
android-11.0-ebook-rkr2
```

```
android-11.0-ebook-rkr3
android-11.0-ebook-rkr4
android-11.0-ebook-rkr5
android-11.0-ebook-rkr6
android-11.0-mid-rkr1
android-11.0-mid-rkr2
android-11.0-mid-rkr3
android-11.0-mid-rkr4
android-11.0-mid-rkr4.1
android-11.0-mid-rkr5
android-11.0-mid-rkr6
android-11.0-mid-rkr7
android-11.0-mid-rkr7-prev
android-11.0-mid-rkr8
```

You can find the latest tag of Android11 currently is `android-11.0-mid-rkr8`

- combine `android-11.0-mid-rkr8` to the local branch

```
git merge android-11.0-mid-rkr8
```

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

```
git push rk29 local_xxx_branch:xxx_branch
```

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

Code compiling

One key compiling command

```
./build.sh -UKAup
（ WHERE: -U = build uboot
       -C = build kernel with Clang
       -K = build kernel
       -A = build android
       -p = will build packaging in IMAGE
       -o = build OTA package
       -u = build update.img
       -v = build android with 'user' or 'userdebug'
       -d = huild kernel dts name
       -V = build version
       -J = build jobs
   ------------you can use according to the requirement, no need to record
uboot/kernel compiling commands------------------
 ）

============================================================
Please remember to set the environment variable before using the one key
compiling command, and select the platform to be compiled, for example:
source build/envsetup.sh
lunch rk3566_rgo-userdebug
```

```
============================================================
```

Compiling command summary

Soc	type	model	Android	one key compiling	kernel compiling	uboot compiling
RK3566	tablet	device	build/envsetup.sh;lunch rk3566_rgo-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_r-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_r-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
RK3568	EVB	EVB4-LPDDR3-V10	build/envsetup.sh;lunch rk3568_r-userdebug	./build.sh -AUCKu	make ARCH=arm64 rockchip_defconfig rk356x_evb.config android-11.config;make ARCH=arm64 rk3568-evb4-lp3-v10.img -j24	./make.sh rk3568
RK3568	EVB	EVB5-DDR4-V10	build/envsetup.sh;lunch rk3568_r-userdebug	./build.sh -AUCKu	make ARCH=arm64 rockchip_defconfig rk356x_evb.config android-11.config;make ARCH=arm64 rk3568-evb5-ddr4-v10.img -j24	./make.sh rk3568
RK3568	EVB	EVB6-DDR3-V10	build/envsetup.sh;lunch rk3568_r-userdebug	./build.sh -AUCKu	make ARCH=arm64 rockchip_defconfig rk356x_evb.config android-11.config;make ARCH=arm64 rk3568-evb6-ddr3-v10.img -j24	./make.sh rk3568
RK3568	EVB	EVB7-DDR4-V10	build/envsetup.sh;lunch rk3568_r-userdebug	./build.sh -AUCKu	make ARCH=arm64 rockchip_defconfig rk356x_evb.config android-11.config;make ARCH=arm64 rk3568-evb7-ddr4-v10.img -j24	./make.sh rk3568
RK3566	EVB	EVB1-DDR4-V10	build/envsetup.sh;lunch rk3566_r-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_r-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
RK3566	EVB	EVB3-DDR3-V10	build/envsetup.sh;lunch rk3566_r-userdebug	./build.sh -AUCKu -d rk3566-evb2-lp4x-v10	make ARCH=arm64 rockchip_defconfig rk356x_evb.config android-11.config;make ARCH=arm64 rk3566-evb3-ddr3-v10.img -j24	./make.sh rk3566

Soc	type	model	Android	one key compiling	kernel compiling	uboot compiling
RK3566	EVB	EVB5-LPDDR4X-V10	build/envsetup.sh;lunch rk3566_r-userdebug	./build.sh -AUCKu -d rk3566-evb2-lp4x-v10	make ARCH=arm64 rockchip_defconfig rk356x_evb.config android-11.config;make ARCH=arm64 rk3566-evb5-lp4x-v10.img -j24	./make.sh rk3566
RK3566	BOX-DEMO board	rk3566-box-demo-v10	build/envsetup.sh;lunch rk356x_box-userdebug	./build.sh -AUCKu -d rk3566-box-demo-v10	make ARCH=arm64 rockchip_defconfig rk356x_evb.config android-11.config;make ARCH=arm64 rk3566-box-demo-v10.img -j24	./make.sh rk3566
RK3326	tablet	863 device	build/envsetup.sh;lunch rk3326_rgo-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_Android11-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
RK3399	EVB	rk3399-sapphire-excavator-edp-avb	build/envsetup.sh;lunch rk3399_Android11-userdebug	./build.sh -AUCKu -d rk3399-sapphire-excavator-edp-avb	make ARCH=arm64 rockchip_defconfig android-11.config;make ARCH=arm64 rk3399-sapphire-excavator-edp-avb.img -j24	./make.sh rk3399
RK3399	IND EVB	rk3399-evb-ind-lpddr4-android-avb	build/envsetup.sh;lunch rk3399_Android11-userdebug	./build.sh -AUCKu	make ARCH=arm64 rockchip_defconfig android-11.config;make ARCH=arm64 rk3399-evb-ind-lpddr4-android-avb.img -j24	./make.sh rk3399
RK3288	EVB	rk3288-evb-android-rk808-edp-avb	build/envsetup.sh;lunch rk3288_Android11-userdebug	./build.sh -AUCKu	make ARCH=arm rockchip_defconfig android-11.config;make ARCH=arm rk3288-evb-android-rk808-edp-avb.img -j24	./make.sh rk3288
RK3328	BOX EVB	rk3328-box-liantong-avb	build/envsetup.sh;lunch rk3328_box-userdebug	./build.sh -AUCKu	make ARCH=arm rockchip_defconfig android-11.config;make ARCH=arm rk3328-box-liantong-avb.img -j24	./make.sh rk3328

Other compiling instruction

Android11.0 cannot directly flash kernel.img and resource.img

Android11.0 kernel.img and resource.img are included in boot.img. After updating and compiling kernel, need to execute ./mkimage.sh in android root directory to re-package boot.img, and then flash boot.img under rockdev. You can use the following method to compile kernel only.

Only compile kernel to generate boot.img

以`RK3566`样机为例，编译时替换对应的boot.img及dts：
其中`BOOT_IMG=../rockdev/Image-rk3566_r/boot.img`这里指定的是旧的boot.img的路径，命令如下：

```
cd kernel
make ARCH=arm64 rockchip_defconfig android-11.config
make ARCH=arm64 BOOT_IMG=../rockdev/Image-rk3566_r/boot.img rk3566-rk817-
tablet.img -j24
```

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_r /boot.img` is the path of the old boot.img. The command is as follows:

```
cd kernel
make ARCH=arm64 rockchip_defconfig android-11.config
make ARCH=arm64 BOOT_IMG=boot_sample.img rk3566-rk817-tablet.img -j24
```

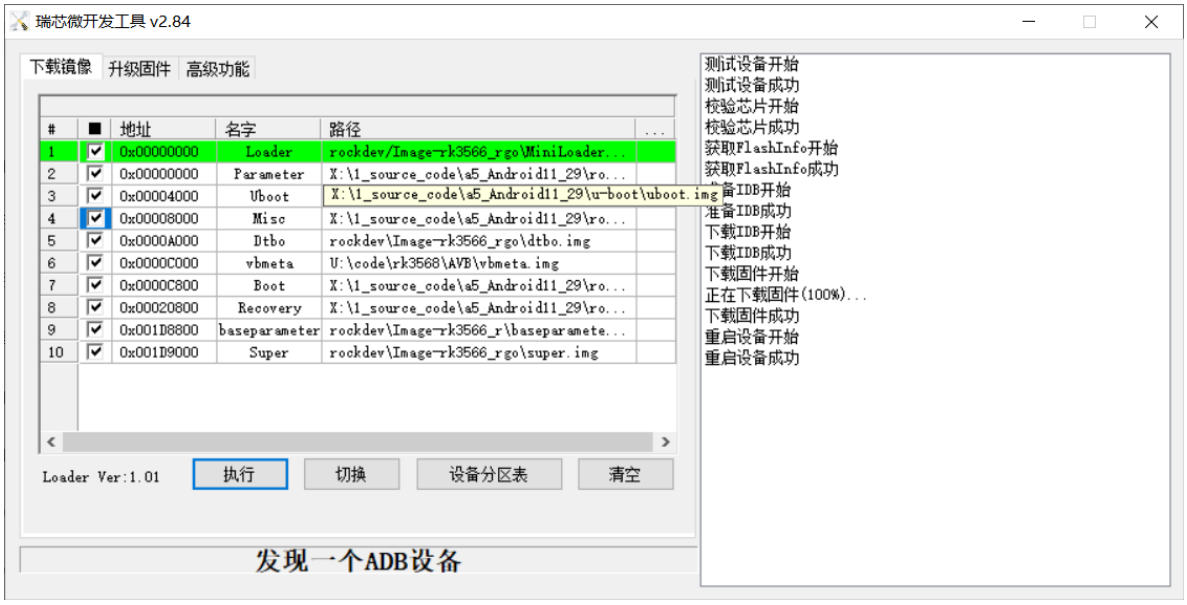
After compiling, you can directly flash the boot.img under kernel directory（**Note: it is zboot.img for 32bit platform, such as 3126c**） to the boot location of the device. **Please firstly load the partition table（parameter.txt）** before flashing, to prevent from flashing to the wrong place.

Image flashing

Image flashing tool

Android11 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:

```
RKTools/windows/AndroidTool/AndroidTool_Release_v2.84
```




```
RKTools/linux/Linux_Upgrade_Tool/Linux_Upgrade_Tool_v1.56
```

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)

```
rockdev/Image-rk3566_rgo/
├── boot-debug.img
├── boot.img
├── config.cfg
├── dtbo.img
├── MiniLoaderAll.bin
├── misc.img
├── parameter.txt
├── pcba_small_misc.img
├── pcba_whole_misc.img
├── recovery.img
├── resource.img
├── super.img
├── uboot.img
├── update.img
└── vbmeta.img
```

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

```
rockdev/Image-rk3566_rgo/
├── boot.img
├── dtbo.img
├── MiniLoaderAll.bin
├── misc.img
├── parameter.txt
├── recovery.img
├── super.img
├── uboot.img
└── vbmeta.img
```

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

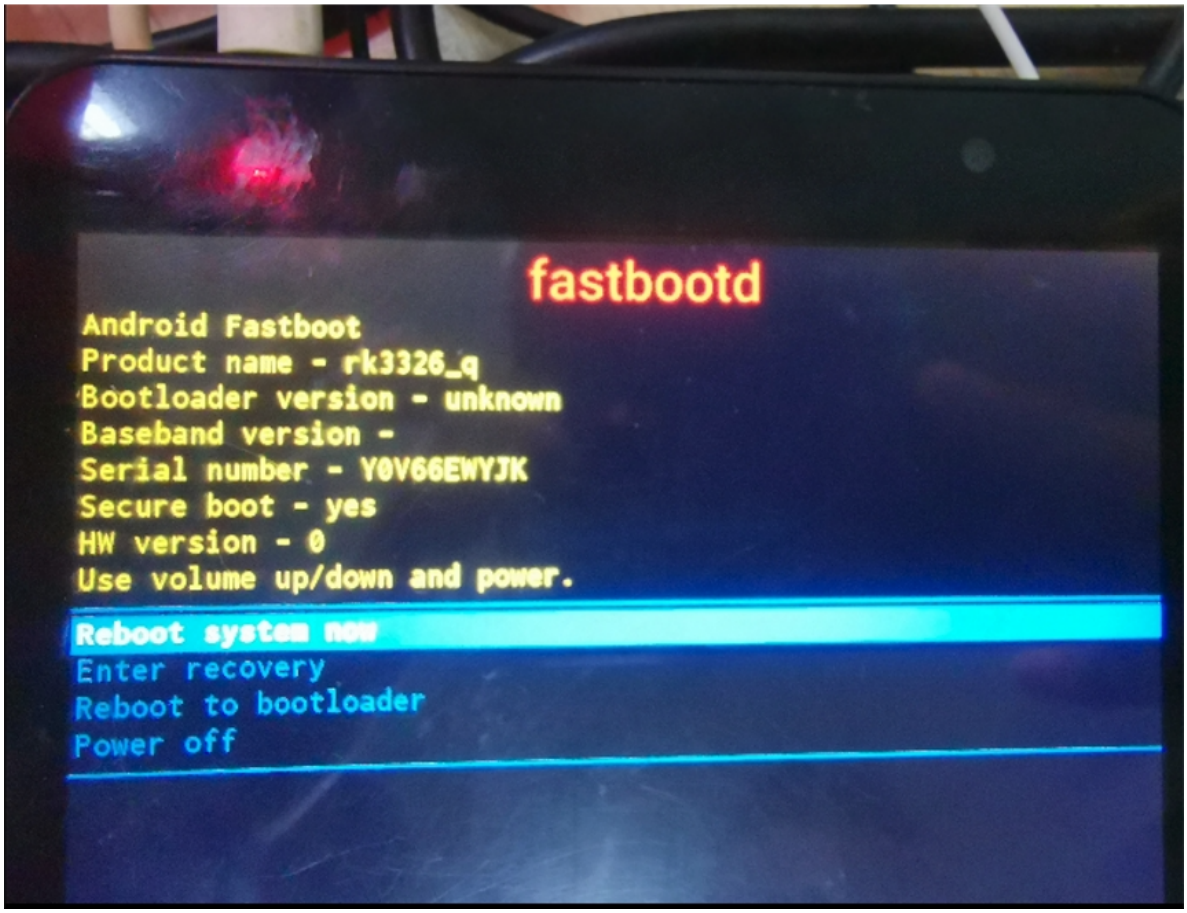
The new device with R supports dynamic partition, and already removes system/vendor/odm partitions. Please flash super.img. Use `fastbootd` can flash system/vendor/odm alone. The version of adb and fastboot should be the latest. SDK provides the compiled tool package:

```
RKTools/linux/Linux_adb_fastboot (Linux_x86 version)
RKTools/windows/adb_fastboot (Windows_x86 version)
```

- Use the command to flash dynamic partition:

```
adb reboot fastboot
fastboot flash vendor vendor.img
fastboot flash system system.img
fastboot flash odm odm.img
```

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:

```
adb reboot bootloader
fastboot oem at-unlock-vboot ## for the customers already flashing avb public
key, please refer to the corresponding document to unlock.
```

2. Reset to factory setting, reboot to fastbootd:

```
fastboot flash misc misc.img
fastboot reboot fastboot  ## now it will enter fastbootd
```

3. Start to flash GSI

```
fastboot delete-logical-partition product  ## (optional) for the device with
small partition space, you can execute this command to delete product partition
first and then flash GSI
fastboot flash system system.img
fastboot reboot  ## after flashing successfully, reboot the device
```

- 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:

```
get_build_var PRODUCT_DTBO_TEMPLATE
```

For example:

```
PRODUCT_DTBO_TEMPLATE := $(LOCAL_PATH)/dt-
overlay.in(device/rockchip/rk356x/rk3566_r/dt-overlay.in)
```

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

```
/dts-v1/;
/plugin/;

&chosen {
    bootargs_ext = "androidboot.boot_devices=${_boot_device}";
};

&firmware_android {
    vbmeta {
        status = "disabled";
    };
    fstab {
        status = "disabled";
    };
};

&reboot_mode {
    mode-bootloader = <0x5242C309>;
    mode-charge = <0x5242C30B>;
    mode-fastboot = <0x5242C303>;
    mode-loader = <0x5242C301>;
    mode-normal = <0x5242C300>;
    mode-recovery = <0x5242C303>;
};
```

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:

```
get_build_var PRODUCT_FSTAB_TEMPLATE
```

For example:

```
PRODUCT_FSTAB_TEMPLATE := device/rockchip/rk356x/rk3566_eink/fstab_eink.in
```

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

Modify parameter.txt

Android 11 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:

```
get_build_var PRODUCT_PARAMETER_TEMPLATE
```

For example:

```
PRODUCT_PARAMETER_TEMPLATE :=
device/rockchip/common/scripts/parameter_tools/parameter.in
```

2. Partition size configuration(example as below):

```
BOARD_SUPER_PARTITION_SIZE := 2688548864
BOARD_DTBOIMG_PARTITION_SIZE := xxxx
BOARD_BOOTIMAGE_PARTITION_SIZE := xxxxx
BOARD_CACHEIMAGE_PARTITION_SIZE := xxxx
```

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

```
device/rockchip/rk3326/rk3326_q/parameter.txt
```

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

```
parameter_tools --input
device/rockchip/common/scripts/parameter_tools/parameter.in --firmware-version
11.0 --machine-model rk3326 --manufacturer rockchip --machine rk3326_r --
partition-list
uboot_a:4096K,trust_a:4M,misc:4M,dtbo_a:4M,vbmeta_a:4M,boot_a:33554432,backup:30
0M,security:4M,cache:300M,metadata:4096,frp:512K,super:2G --output
parameter_new.txt
```

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_r product. The steps are as below:
1.Modify device/rockchip/rk356x/AndroidProducts.mk to add rk3568_r lunch

```
--- a/AndroidProducts.mk
+++ b/AndroidProducts.mk
@@ -17,10 +17,14 @@
 PRODUCT_MAKEFILES := \
         $(LOCAL_DIR)/rk3566_rgo/rk3566_rgo.mk \
         $(LOCAL_DIR)/rk3566_r/rk3566_r.mk \
+        $(LOCAL_DIR)/rk3568_r/rk3568_r.mk \

 COMMON_LUNCH_CHOICES := \
     rk3566_rgo-userdebug \
     rk3566_rgo-user \
     rk3566_r-userdebug \
     rk3566_r-user \
+    rk3568_r-userdebug \
+    rk3568_r-user \
```

2.Create rk3568_r directory under device/rockchip/rk356x
Create referring to the existing rk3566_r product directory in device/rockchip/rk356x. You can directly copy rk3566_r to rk3568_r, and then replace all the `rk3566_r` under rk3568_r directory with `rk3568_r`

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
RK3399	RK808	LPDDR3	development board	sapphire excavator	rk3399-sapphire-excavator-edp-avb
RK3399	RK809	LPDDR4	development board	IND EVB	rk3399-evb-ind-lpddr4-android-avb
RK3399	RK808	LPDDR4	tablet	BQ25703 dual battery	rk3399-tve1030g-avb
RK3399	RK818	LPDDR3	tablet	edp panel	rk3399-mid-818-android
RK3288W	RK808	LPDDR3	EVB	edp panel	rk3288-evb-android-rk808-edp-avb
RK3328	discrete device	DDR3	development board	RK3328 development board	rk3328-box-liantong-avb

Document instruction

Peripheral support list

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

```
https://redmine.rockchip.com.cn/projects/fae/documents
```

Android document

```
RKDocs\android
```

Android_SELinux(Sepolicy) developer guide

```
RKDocs/android/Rockchip_Developer_Guide_Android_SELinux(Sepolicy)_CN.pdf
```

Android 11 System Optimization developer guide

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

```
RKDocs\android\Rockchip_Developer_Guide_Android11_Optimization_CN.pdf
```

Wi-Fi document

```
RKDocs/android/wifi/
├── Rockchip_Introduction_Android10.0_WIFI_Configuration_CN&EN.pdf
└── Rockchip_Introduction_REALTEK_WIFI_Driver_Porting_CN&EN.pdf
```

3G/4G module instruction document

```
RKDocs/common/mobile-net/
├── Rockchip_Introduction_3G_Data_Card_USB_File_Conversion_CN.pdf
├── Rockchip_Introduction_3G_Dongle_Configuration_CN.pdf
└── Rockchip_Introduction_4G_Module_Configuration_CN&EN.pdf
```

Kernel document

```
RKDocs\common
```

DDR related document

```
RKDocs/common/DDR/
├── Rockchip-Developer-Guide-DDR-CN.pdf
├── Rockchip-Developer-Guide-DDR-EN.pdf
├── Rockchip-Developer-Guide-DDR-Problem-Solution-CN.pdf
├── Rockchip-Developer-Guide-DDR-Problem-Solution-EN.pdf
└── Rockchip-Developer-Guide-DDR-Verification-Process-CN.pdf
```

Audio module document

```
RKDocs/common/Audio/
├──
Rockchip_Developer_Guide_Audio_Call_3A_Algorithm_Integration_and_Parameter_Debug
ging_CN.pdf
├── Rockchip_Developer_Guide_Linux4.4_Audio_CN.pdf
└── Rockchip_Developer_Guide_RK817_RK809_Codec_CN.pdf
```

CRU module document

```
RKDocs/common/CRU/
├── Rockchip_Developer_Guide_Linux3.10_Clock_CN.pdf
└── Rockchip_RK3399_Developer_Guide_Linux4.4_Clock_CN.pdf
```

GMAC module document

```
RKDocs/common/GMAC/
└── Rockchip_Developer_Guide_Ethernet_CN.pdf
```

PCie module document

```
RKDocs/common/PCie/
└── Rockchip-Developer-Guide-linux4.4-PCIe.pdf
```

I2C module document

```
RKDocs/common/I2C/
└── Rockchip_Developer_Guide_I2C_CN.pdf
```

PIN-Ctrl GPIO module document

```
RKDocs/common/PIN-Ctrl/
└── Rockchip-Developer-Guide-Linux-Pin-Ctrl-CN.pdf
```

SPI module document

```
RKDocs/common/SPI/
└── Rockchip-Developer-Guide-linux4.4-SPI.pdf
```

Sensor module document

```
RKDocs/common/Sensors/
└── Rockchip_Developer_Guide_Sensors_CN.pdf
```

IO-Domain module document

```
RKDocs/common/IO-Domain/
└── Rockchip_Developer_Guide_Linux_IO_DOMAIN_CN.pdf
```

Leds module document

```
RKDocs/common/Leds/
└── Rockchip_Introduction_Leds_GPIO_Configuration_for_Linux4.4_CN.pdf
```

Thermal control module document

```
RKDocs/common/Thermal/
├── Rockchip-Developer-Guide-Linux4.4-Thermal-CN.pdf
└── Rockchip-Developer-Guide-Linux4.4-Thermal-EN.pdf
```

PMIC power management module document

```
RKDocs/common/PMIC/
├── Archive.zip
├── Rockchip_Developer_Guide_Power_Discrete_DCDC_EN.pdf
├── Rockchip-Developer-Guide-Power-Discrete-DCDC-Linux4.4.pdf
├── Rockchip-Developer-Guide-RK805.pdf
├── Rockchip_Developer_Guide_RK817_RK809_Fuel_Gauge_CN.pdf
├── Rockchip_RK805_Developer_Guide_CN.pdf
└── Rockchip_RK818_RK816_Introduction_Fuel_Gauge_Log_CN.pdf
```

MCU module document

```
RKDocs/common/MCU/
└── Rockchip_Developer_Guide_MCU_EN.pdf
```

Power consumption and sleep module document

```
RKDocs/common/power/
├── Rockchip_Developer_Guide_Power_Analysis_EN.pdf
└── Rockchip_Developer_Guide_Sleep_and_Resume_CN.pdf
```

UART module document

```
RKDocs/common/UART/
├── Rockchip-Developer-Guide-linux4.4-UART.pdf
└── Rockchip-Developer-Guide-RT-Thread-UART.pdf
```

DVFS CPU/GPU/DDR frequency scaling related document

```
RKDocs/common/DVFS/
├── Rockchip_Developer_Guide_CPUFreq_CN.pdf
├── Rockchip_Developer_Guide_CPUFreq_EN.pdf
├── Rockchip_Developer_Guide_Devfreq_CN.pdf
├── Rockchip_Developer_Guide_Linux4.4_CPUFreq_CN.pdf
└── Rockchip_Developer_Guide_Linux4.4_Devfreq_CN.pdf
```

EMMC/SDMMC/SDIO module document

```
RKDocs/common/MMC
└── Rockchip-Developer-Guide-linux4.4-SDMMC-SDIO-eMMC.pdf
```

PWM module document

```
RKDocs/common/PWM/
├── Rockchip-Developer-Guide-Linux-PWM-CN.pdf
└── Rockchip_Developer_Guide_PWM_IR_CN.pdf
```

USB module document

```
RKDocs/common/usb/
├── putty20190213_162833_1.log
├── Rockchip-Developer-Guide-Linux4.4-RK3399-USB-DTS-CN.pdf
├── Rockchip-Developer-Guide-Linux4.4-USB-CN.pdf
├── Rockchip-Developer-Guide-Linux4.4-USB-FFS-Test-Demo-CN.pdf
├── Rockchip-Developer-Guide-Linux4.4-USB-Gadget-UAC-CN.pdf
├── Rockchip-Developer-Guide-USB-Initialization-Log-Analysis-CN.pdf
├── Rockchip-Developer-Guide-USB-Performance-Analysis-CN.pdf
├── Rockchip-Developer-Guide-USB-PHY-CN.pdf
└── Rockchip-Developer-Guide-USB-SQ-Test-CN.pdf
```

HDMI-IN function document

```
RKDocs/common/hdmi-in/
└── Rockchip_Developer_Guide_HDMI_IN_CN.pdf
```

Security module document

```
RKDocs/common/security/
├── Efuse process explain .pdf
├── RK3399_Efuse_Operation_Instructions_V1.00_20190214_EN.pdf
├── Rockchip_Developer_Guide_Secure_Boot_V1.1_20190603_CN.pdf
├── Rockchip_Developer_Guide_TEE_Secure_SDK_CN.pdf
├── Rockchip_RK3399_Introduction_Efuse_Operation_EN.pdf
├── Rockchip-Secure-Boot2.0.pdf
├── Rockchip-Secure-Boot-Application-Note-V1.9.pdf
└── Rockchip Vendor Storage Application Note.pdf
```

uboot introduction document

```
RKDocs\common\u-boot\Rockchip-Developer-Guide-UBoot-nextdev-CN.pdf
```

Trust introduction document

```
RKDocs/common/TRUST/
├── Rockchip_Developer_Guide_Trust_CN.pdf
└── Rockchip_Developer_Guide_Trust_EN.pdf
```

Camera document

```
RKDocs\common\camera\HAL3\
```

Camera IQ Tool document

```
external/camera_engine_rkaiq/rkisp2x_tuner/doc/
├── Rockchip_Color_Optimization_Guide_ISP2x_CN_v2.0.0.pdf
├── Rockchip_IQ_Tools_Guide_ISP2x_CN_v2.0.0.pdf
└── Rockchip_Tuning_Guide_ISP21_CN_v2.0.0.pdf
```

Tool document

```
RKDocs\common\RKTools manuals
```

PCBA development and usage document

```
RKDocs\android\Rockchip_Developer_Guide_PCBA_Test_Tool_CN&EN.pdf
```

Panel driver debugging guide

```
RKDocs\common\display\Rockchip_Developer_Guide_DRM_Panel_Porting_CN.pdf
```

HDMI debugging guide

```
RKDocs\common\display\Rockchip_Developer_Guide_HDMI_Based_on_DRM_Framework_CN.pd
f
```

Graphic display DRM Hardware Composer（HWC) issue analyzing

```
RKDocs\common\display\Rockchip FAQ DRM Hardware Composer V1.00-20181213.pdf
```

DRM display developer guide

```
RKDocs\common\display\Rockchip DRM Display Driver Development Guide V1.0.pdf
```

RGA related issues analyzing

```
RKDocs\common\display\Rockchip_RGA_FAQ.pdf
```

Graphic display framework common issue analysis

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

```
RKDocs\common\display\Rockchip_Trouble_Shooting_Graphics
```

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:

```
RKDocs\android\Rockchip_Developer_Guide_PCBA_Test_Tool_CN&EN.pdf_V1.1_20171222.pdf。
```

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

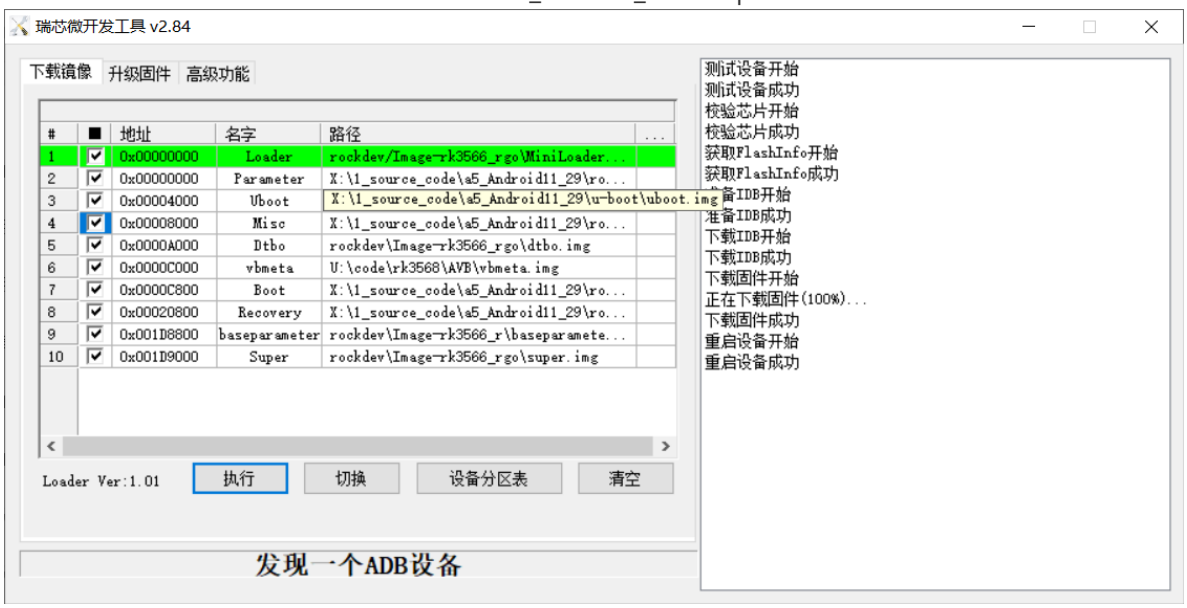
Rockchip USB driver install package includes ADB and image flashing driver

```
RKTools\windows\DriverAssitant_v5.1.1.zip
```

Development flashing tool

Windows version

RKTools/windows/AndroidTool/AndroidTool_Release_v2.84.zip



Linux version

RKTools/linux/Linux_Upgrade_Tool/Linux_Upgrade_Tool_v1.56.zip

```
Linux_Upgrade_Tool_v1.56$ sudo ./upgrade_tool -h
Program Data in /home/wlq/.config/upgrade_tool

----------------------Tool Usage ----------------------
Help:              H
Quit:              Q
Version:           V
Clear Screen:      CS
------------------Upgrade Command ------------------
ChooseDevice:            CD
ListDevice:                  LD
SwitchDevice:            SD
UpgradeFirmware:         UF <Firmware> [-noreset]
UpgradeLoader:           UL <Loader> [-noreset]
DownloadImage:           DI <-p|-b|-k|-s|-r|-m|-u|-t|-re image>
DownloadBoot:            DB <Loader>
EraseFlash:              EF <Loader|firmware> [DirectLBA]
PartitionList:           PL
WriteSN:                 SN <serial number>
ReadSN:              RSN
----------------Professional Command -----------------
TestDevice:              TD
ResetDevice:             RD [subcode]
ResetPipe:               RP [pipe]
ReadCapability:          RCB
ReadFlashID:             RID
ReadFlashInfo:           RFI
ReadChipInfo:            RCI
ReadSector:              RS  <BeginSec> <SectorLen> [-decode] [File]
WriteSector:             WS  <BeginSec> <File>
ReadLBA:                 RL  <BeginSec> <SectorLen> [File]
WriteLBA:                WL  <BeginSec> <File>
EraseLBA:                EL  <BeginSec> <EraseCount>
EraseBlock:              EB <CS> <BeginBlock> <BlokcLen> [--Force]
-------------------------------------------------------
```

Tool to implement SD upgrading and boot

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

```
RKTools\windows\SDDiskTool_v1.59.zip
```

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:

```
RKDocs\common\RKTools manuals\RKDevInfoWriteTool_User_Guide_V1.0.3.pdf
```

DDR welding test tool

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

```
RKTools\windows\Rockchip_Platform_DDR_Test_Tool_V1.38_Release_Annoucement_CN.7z
RKTools\windows\Rockchip_Platform_DDR_Test_Tool_V1.38_Release_Annoucement_EN.7z
```

efuse flashing tool

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

```
RKTools\windows\efuse_v1.37.rar
```

efuse/otp sign tool

It is used to sign efuse/otp of image.

```
RKTools\windows\SecureBootTool_v1.94.zip
```

Factory production image flashing tool

It is uesed for batch image flashing in factory.

```
RKTools\windows\FactoryTool_1.66.zip
```

Image update tool

It is used to modify update.img.

```
RKTools\windows\FWFactoryTool_V5.52.rar
```

userdata partition data prebuilt tool

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

```
RKTools\windows\OemTool_v1.3.rar
```

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:

```
adb shell
```

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.

Network ADB usage requirement

ADB early versions only support device debugging through USB, and the function of debugging Android devices through tcp/ip is added from adb v1.0.25.
If you need to use network ADB to debug the device, must meet below conditions:
1、The device must have network port, or connect the network through Wi-Fi.
2、The device and PC are already in the local network and the device has IP address.
3、Need to confirm the device and PC can ping each other.
4、PC already installs abd.
5、Confirm Android device adbd process (adb background process) is already run. adbd process will monitor port 5555 to do adb connection debugging.

SDK network ADB port configuration

SDK doesn’t enable the network ADB by default. Need to manually enable in the developer option.
Setting-System-Advanced-Developer options-Open net adb

Network ADB usage

This chapter assumes the device IP is 192.168.1.5. This IP will be used for adb connection and device debugging in the following context.
1、Firstly the Android device should boot up, if possible, confirm adbd is started (use ps command to check).
2、In PC cmd, input:

```
adb connect 192.168.1.5:5555
```

If successful, it will prompt relative hints, if fail, you can execute kill-server command and then retry connection.

```
adb kill-server
```

3、After connected, you can input ADB related commands to debug in PC, such as adb shell, it will connect the device through TCP/IP which is the same as USB debugging.
4、After debugging, input the following command to disconnect the connection in PC:

```
adb disconnect 192.168.1.5:5555
```

Manually change the network ADB port number

If SDK doesn't add adb port number configuration, or want to change adb port number, you can change through below method:
1、Firstly also connect the device normally through USB, execute adb shell in windows cmd to enter.
2、Set ADB monitor port:

```
#setprop service.adb.tcp.port 5555
```

3、Look up adbd pid using ps command.
4、Reset adbd.
#kill -9, This pid is just the one found in last step.
After killing adbd, adbd will automatically restart after Android init process. After adbd restart, if service.adb.tcp.port is set, it will automatically change to monitor network request.

ADB commonly used command elaboration

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

```
adb devices
```

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:

```
adb install <apk file path>
```

For example:

```
adb install “F:\WishTV\WishTV.apk”
```

Re-install application:

```
adb install –r “F:\WishTV\WishTV.apk”
```

（3）Uninstall APK
Complete uninstall:

```
adb uninstall <package>
```

For example:

```
adb uninstall com.wishtv
```

（4）Use rm to remove apk file:

```
adb shell rm <filepath>
```

For example:

```
adb shell rm “system/app/WishTV.apk”
```

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:

```
adb shell
```

（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:

```
adb push “F:\WishTV\WishTV.apk” “system/app”
```

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.

```
adb pull <remote path> <local path>
```

For example:

```
adb pull system/app/Contacts.apk F:\
```

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.

```
adb shell getprop
```

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:

```
[adb] logcat [<option>] [<filter-spec>]
```

For example:

```
adb shell
logcat
```

The commonly used logcat filter method

Several ways to control the logcat output:

- Control the logcat output priority
  For example:

```
adb shell
logcat *:W
```

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

- Control the logcat label and output priority
  For example:

```
adb shell
logcat ActivityManager:I MyApp:D *:S
```

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:

```
adb shell
logcat WishTV:* *:S
```

or

```
adb shell
logcat –s WishTV
```

Note: only output the logcat with label of WishTV.

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

```
adb shell
logcat WishTV:I *:S
```

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:

```
procrank [ -w ] [ -v | -r | -p | -u | -h ]
```

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:

```
procrank
```

Output the memory snapshot in VSS decreasing order:

```
procrank –v
```

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:

```
procrank | grep [cmdline | PID]
```

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

```
procrank | grep “com.android.systemui”
```

or:

```
procrank  | grep 3396
```

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

```
#!/bin/bash
while true;do
adb shell procrank | grep “com.android.systemui”
sleep 1
done
```

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.

```
  dumpsys  -help
```

- 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.

```
  dumpsys  -l
```

- 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:

```
dumpsys 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:

```
dumpsys meminfo 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

```
ramoops_mem: ramoops_mem {
    reg = <0x0 0x110000 0x0 0xf0000>;
    reg-names = "ramoops_mem";
};

ramoops {
    compatible = "ramoops";
    record-size = <0x0 0x20000>;
    console-size = <0x0 0x80000>;
    ftrace-size = <0x0 0x00000>;
    pmsg-size = <0x0 0x50000>;
    memory-region = <&ramoops_mem>;
};
```

```
- Check last log in the device
130|root@rk3399:/sys/fs/pstore # ls
dmesg-ramoops-0   Log saved after last kernel panic
pmsg-ramoops-0    Log of last user space, android log
ftrace-ramoops-0    Print function trace during some period
console-ramoops-0  kernel log when last_log was enabled last time, but only save
the log with higher priority than default log level
```

- Usage:

```
cat dmesg-ramoops-0
cat console-ramoops-0
logcat -L （pmsg-ramoops-0） pull out by logcat and parse
cat ftrace-ramoops-0
```

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:

```
127|console:/ $ fiq
debug>  help
FIQ Debugger commands:
 pc            PC status
 regs          Register dump
 allregs       Extended Register dump
 bt            Stack trace
 reboot [<c>]  Reboot with command <c>
 reset [<c>]   Hard reset with command <c>
 irqs          Interupt status
 kmsg          Kernel log
 version       Kernel version
 sleep         Allow sleep while in FIQ
 nosleep       Disable sleep while in FIQ
 console       Switch terminal to console
 cpu           Current CPU
 cpu <number>  Switch to CPU<number>
 ps            Process list
 sysrq         sysrq options
 sysrq <param> Execute sysrq with <param>
```

log auto collection

- Collection content

  ```
  android：android log
  kernel ：kernel log
  ```

- Enable method
- Open Developer options
- Setting-System-Advanced-Developer options-Android bug collector
- Save path of log

  ```
  data/vendor/logs/
  ```

Common issues

What is current kernel version and u-boot version?

```
The corresponding kernel version of Android11.0 is: develop-4.19, u-boot branch
is next-dev branch
```

How to acquire the corresponding RK release version for current SDK

Rockchip Android11.0 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

```
vim .repo/manifests/include/rk_checkout_from_aosp.xml
<project groups="pdk" name="platform/build" path="build/make" remote="rk"
revision="refs/tags/android-11.0-mid-rkr1">
```

Means RK version is android-11.0-mid-rkr1

- AOSP part

```
vim .repo/manifests/default.xml
<default revision="refs/tags/android-10.0.0_r14"
```

Means OASP version is android-10.0.0_r14
Just provide the above two version information when needed.
You can directly acquire tag information through the following command for single lib:

```
/kernel$ git tag
android-11.0-mid-rkr1
develop-4.4-20190201
```

RK version is incremental with the format of android-11.0-mid-rkrxx, so current latest tag is android-11.0-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

```
.repo/repo/repo manifest -r -o release_manifest_rkr6_local.xml
```

- Comparing .repo/manifests/commit/commit_release_rkr6.xml with release_manifest_rkr6_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 RKR6 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:

```
charge-animation {
                compatible = "rockchip,uboot-charge";
                rockchip,uboot-charge-on = <1>;
                rockchip,android-charge-on = <0>;
                rockchip,uboot-low-power-voltage = <3400>;
                rockchip,screen-on-voltage = <3500>;
                status = "okay";
        };
```

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.

```
u-boot/tools/images/
├── battery_0.bmp
├── battery_1.bmp
├── battery_2.bmp
├── battery_3.bmp
├── battery_4.bmp
├── battery_5.bmp
└── battery_fail.bmp
```

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:

```
cd u-boot
./scripts/pack_resource.sh ../kernel/resource.img
cp resource.img ../kernel/resource.img
```

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:

```
vim device/rockchip/common/BoardConfig.mk
#for rk 4g modem
-BOARD_HAS_RK_4G_MODEM ?= false
+BOARD_HAS_RK_4G_MODEM ?= true
```

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:

```
vim device/rockchip/common/BoardConfig.mk
#0:   ROTATION_NONE rotate 0 degree
#90:  ROTATION_RIGHT rotate 90 degrees
#180: ROTATION_DOWN rotate 180 degrees
#270: ROTATION_LEFT rotate 270 degrees
# For Recovery Rotation
TARGET_RECOVERY_DEFAULT_ROTATION ?= ROTATION_NONE
```

Android Surface rotation

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

```
# For Surface Flinger Rotation
SF_PRIMARY_DISPLAY_ORIENTATION ?= 0
```

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

```
< remote name="aosp" fetch="./" review="https://10.10.10.29" />
```

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

```
< remote name="aosp" fetch="https://aosp.tuna.tsinghua.edu.cn" />;
```

for foreign customers: (Google mirror source)

```
< remote name="aosp" fetch="https://android.googlesource.com" />
```

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:

```
cd device/rockchip/common

diff --git a/scripts/fstab_tools/fstab.in b/scripts/fstab_tools/fstab.in
index 2ec6c265..c890cc84 100755
--- a/scripts/fstab_tools/fstab.in
+++ b/scripts/fstab_tools/fstab.in
@@ -23,6 +23,6 @@ ${_block_prefix}odm     /odm      ext4 ro,barrier=1
${_flags},first_stage_mount
 # For sdmmc
 /devices/platform/${_sdmmc_device}/mmc_host*        auto  auto    defaults
   voldmanaged=sdcard1:auto
 #  Full disk encryption has less effect on rk3326, so default to enable this.
-/dev/block/by-name/userdata /data f2fs
noatime,nosuid,nodev,discard,reserve_root=32768,resgid=1065
latemount,wait,check,fileencryption=aes-256-xts:aes-256-
cts:v2+inlinecrypt_optimized,keydirectory=/metadata/vold/metadata_encryption,quo
ta,formattable,reservedsize=128M,checkpoint=fs
+/dev/block/by-name/userdata /data f2fs
noatime,nosuid,nodev,discard,reserve_root=32768,resgid=1065,fsync_mode=nobarrier
latemount,wait,check,fileencryption=aes-256-xts:aes-256-
cts:v2+inlinecrypt_optimized,keydirectory=/metadata/vold/metadata_encryption,quo
ta,formattable,reservedsize=128M,checkpoint=fs
 # for ext4
 #/dev/block/by-name/userdata    /data      ext4
discard,noatime,nosuid,nodev,noauto_da_alloc,data=ordered,user_xattr,barrier=1
latemount,wait,formattable,check,fileencryption=software,quota,reservedsize=128M
,checkpoint=block
diff --git a/scripts/fstab_tools/fstab_go.in b/scripts/fstab_tools/fstab_go.in
index 582557f2..05c7653c 100755
--- a/scripts/fstab_tools/fstab_go.in
+++ b/scripts/fstab_tools/fstab_go.in
@@ -17,6 +17,6 @@ ${_block_prefix}odm     /odm      ext4 ro,barrier=1
${_flags},first_stage_mount
 # For sdmmc
 /devices/platform/${_sdmmc_device}/mmc_host*        auto  auto    defaults
   voldmanaged=sdcard1:auto
 #  Full disk encryption has less effect on rk3326, so default to enable this.
```

```
-/dev/block/by-name/userdata /data f2fs
noatime,nosuid,nodev,discard,reserve_root=32768,resgid=1065
latemount,wait,check,fileencryption=aes-256-xts:aes-256-
cts:v2+inlinecrypt_optimized,keydirectory=/metadata/vold/metadata_encryption,quo
ta,formattable,reservedsize=128M,checkpoint=fs
+/dev/block/by-name/userdata /data f2fs
noatime,nosuid,nodev,discard,reserve_root=32768,resgid=1065i,fsync_mode=nobarrie
r latemount,wait,check,fileencryption=aes-256-xts:aes-256-
cts:v2+inlinecrypt_optimized,keydirectory=/metadata/vold/metadata_encryption,quo
ta,formattable,reservedsize=128M,checkpoint=fs
 # for ext4
 #/dev/block/by-name/userdata    /data      ext4
discard,noatime,nosuid,nodev,noauto_da_alloc,data=ordered,user_xattr,barrier=1
latemount,wait,formattable,check,fileencryption=software,quota,reservedsize=128M
,checkpoint=block
```

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:

```
device/rockchip/common$ git diff
diff --git a/scripts/fstab_tools/fstab.in b/scripts/fstab_tools/fstab.in
index 6e78b00..a658332 100755
--- a/scripts/fstab_tools/fstab.in
+++ b/scripts/fstab_tools/fstab.in
@@ -20,6 +20,6 @@ ${_block_prefix}system_ext /system_ext  ext4 ro,barrier=1
${_flags},first_stage_
 # For sdmmc
 /devices/platform/${_sdmmc_device}/mmc_host*        auto  auto    defaults
   voldmanaged=sdcard1:auto
 #  Full disk encryption has less effect on rk3326, so default to enable this.
-/dev/block/by-name/userdata /data f2fs
noatime,nosuid,nodev,discard,reserve_root=32768,resgid=1065
latemount,wait,check,fileencryption=aes-256-xts:aes-256-
cts:v2+inlinecrypt_optimized,quota,formattable,reservedsize=128M,checkpoint=fs
+#/dev/block/by-name/userdata /data f2fs
noatime,nosuid,nodev,discard,reserve_root=32768,resgid=1065
latemount,wait,check,fileencryption=aes-256-xts:aes-256-
cts:v2+inlinecrypt_optimized,quota,formattable,reservedsize=128M,checkpoint=fs
 # for ext4
-#/dev/block/by-name/userdata    /data      ext4
discard,noatime,nosuid,nodev,noauto_da_alloc,data=ordered,user_xattr,barrier=1
latemount,wait,formattable,check,fileencryption=software,quota,reservedsize=128M
,checkpoint=block
+/dev/block/by-name/userdata    /data      ext4
discard,noatime,nosuid,nodev,noauto_da_alloc,data=ordered,user_xattr,barrier=1
latemount,wait,formattable,check,fileencryption=software,quota,reservedsize=128M
,checkpoint=block
```

```
device/rockchip/rk356x$ git diff
diff --git a/rk3566_r/recovery.fstab b/rk3566_r/recovery.fstab
index 7532217..cf789ac 100755
```

```
--- a/rk3566_r/recovery.fstab
+++ b/rk3566_r/recovery.fstab
@@ -7,7 +7,7 @@
 /dev/block/by-name/odm                  /odm                ext4
defaults                  defaults
 /dev/block/by-name/cache                /cache              ext4
defaults                  defaults
 /dev/block/by-name/metadata             /metadata           ext4
defaults                  defaults
-/dev/block/by-name/userdata             /data               f2fs
defaults                  defaults
+/dev/block/by-name/userdata             /data               ext4
defaults                  defaults
 /dev/block/by-name/cust                 /cust               ext4
defaults                  defaults
 /dev/block/by-name/custom               /custom             ext4
defaults                  defaults
 /dev/block/by-name/radical_update       /radical_update     ext4
defaults                  defaults
```

Modify power on/off animation and tones

Reference document:

```
RKDocs\android\Rockchip_Introduction_Android_Power_On_Off_Animation_and_Tone_Customization_CN&EN.pdf
```

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)

```
< app package="package name" mode="whether to enable the acceleration, 1 for enable, 0 for disable"/>
```

Take antutu as example as below:

```
< app package="com.antutu.ABenchMark"mode="1"/>
< app package="com.antutu.benchmark.full"mode="1"\/>
< app package="com.antutu.benchmark.full"mode="1"\/>
```

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.

```
RKDocs\android\Rockchip_User_Guide_Dr.G_CN&EN.pdf
```

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:

```
RKDocs\common\security\Rockchip-Secure-Boot-Application-Note-V1.9.pdf
```

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

```
device/rockchip/common$
--- a/BoardConfig.mk
+++ b/BoardConfig.mk
@@ -67,7 +67,7 @@ endif

 # Enable android verified boot 2.0
 BOARD_AVB_ENABLE ?= false
-BOARD_SELINUX_ENFORCING ?= false
+BOARD_SELINUX_ENFORCING ?= true
```

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:

```

```

```
diff --git
a/services/core/java/com/android/server/wm/ActivityTaskManagerService.java
b/services/core/java/com/android/server/wm/ActivityTaskManagerService.java
index 595c340..d4e495a 100644
--- a/services/core/java/com/android/server/wm/ActivityTaskManagerService.java
+++ b/services/core/java/com/android/server/wm/ActivityTaskManagerService.java
@@ -6555,7 +6555,7 @@ public class ActivityTaskManagerService extends
IActivityTaskManager.Stub {
                 } catch (RemoteException e) {
                 }

-                if (!Build.isBuildConsistent()) {
+                if (0 && !Build.isBuildConsistent()) {
                     Slog.e(TAG, "Build fingerprint is not consistent, warning
user");
                     mUiHandler.post(() -> {
                         if (mShowDialogs) {
```

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:

```
  --- a/BoardConfig.mk
+++ b/BoardConfig.mk
@@ -146,3 +146,6 @@ endif
 ifeq ($(strip $(BOARD_USES_AB_IMAGE)), true)
 DEVICE_MANIFEST_FILE :=
device/rockchip/$(TARGET_BOARD_PLATFORM)/manifest_ab.xml
 endif

+# for ethernet
+BOARD_HS_ETHERNET := true
```

About AVB and security boot

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

```
RKDocs/common/security/RK356X_SecurityBoot_And_AVB_instructions_CN.pdf
```

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:

```
wlq@ubuntu:~/1_source_code/a3_rk3399_android10.0_29/kernel$ vim
kernel/configs/android-11.config
```

For GO produts, need to modify:

```
wlq@ubuntu:~/1_source_code/a3_rk3399_android10.0_29/kernel$ vim
kernel/configs/android-11-go.config
```

delete the following line:

```
# CONFIG_DEVMEM is not set
```

```
wlq@ubuntu:~/1_source_code/a3_rk3399_android10.0_29/kernel$ vim
configs/r/android-4.19/android-base.config
```

delete the following line:

```
# CONFIG_DEVMEM is not set
```

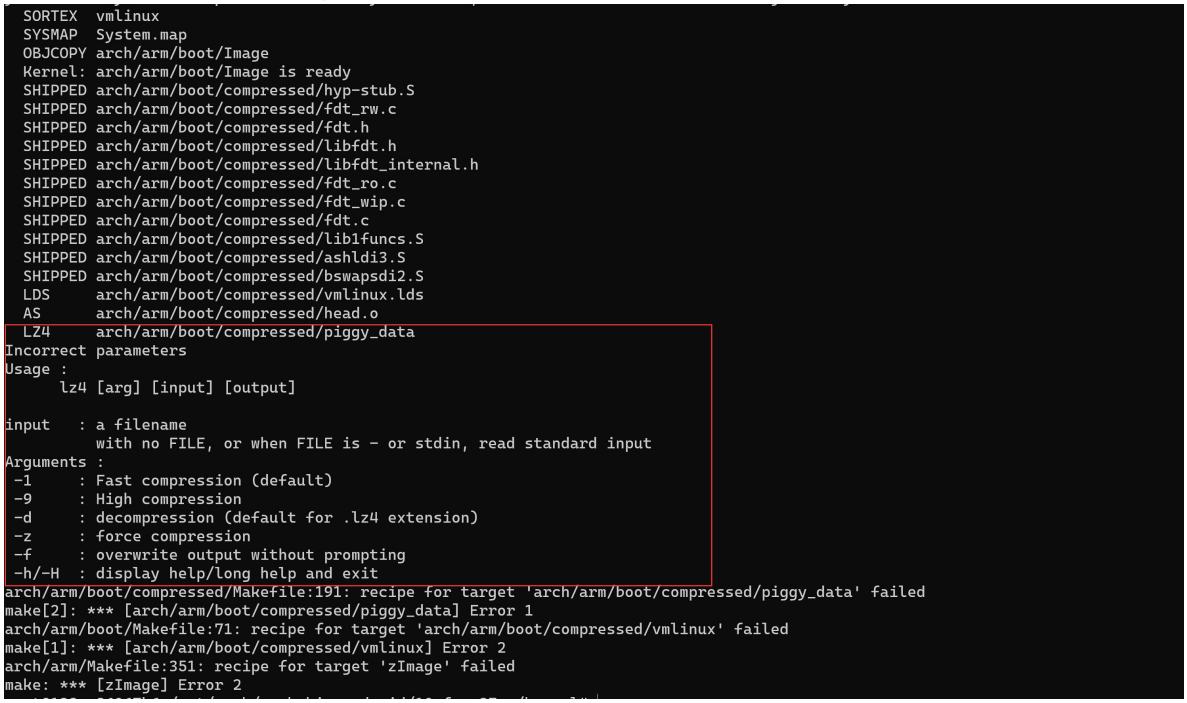
Both of them need to be deleted

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:



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

```
wlq@ubuntu:~$ lz4 -v
*** LZ4 command line interface 64-bits v1.8.3, by Yann Collet ***
refusing to read from a console
```

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

```
sudo cp out/host/linux-x86/bin/lz4 /usr/bin/lz4
```

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.

```
commit 65a3eeea668ca570009caa1423f9780d44efebbf
Author: Wu Liangqing <wlq@rock-chips.com>
Date:   Fri Apr 23 17:11:25 2021 +0800

    arm64: dts: rockchip: add not-save-power-en to rk356x board

    fix reboot block as follows log:
    [   15.874382] binder: release 247:268 transaction 4234 in, still active
    [   15.874418] binder: send failed reply for transaction 4234 to 395:455
    [   15.959849] binder: undelivered TRANSACTION_ERROR: 29189
    [   16.085993] binder: 147:147 transaction failed 29189/-22, size 100-0 line
3059
    [   16.128154] android_work: sent uevent USB_STATE=DISCONNECTED
    [   16.145570] logd.klogd: 24 output lines suppressed due to ratelimiting
    [   16.690141] cpu cpu0: min=816000, max=816000
    [   16.696558] rk808 0-0020: reboot: force RK817_RST_FUNC_REG ok!
    [   33.769778] vcc5v0_otg: disabling
    [   33.770099] vcc3v3_lcd0_n: disabling
    [   33.770424] vcc3v3_lcd1_n: disabling
    [   37.699768] rcu: INFO: rcu_preempt detected stalls on CPUs/tasks:
    [   37.700342] rcu:     3-...0: (0 ticks this GP)
idle=b52/1/0x4000000000000000 softirq=4194/4194 fqs=2012
    [   37.701150] rcu:     (detected by 0, t=6302 jiffies, g=3301, q=16)
    [   37.701684] Task dump for CPU 3:
    [   37.701981] init            R  running task        0     1      0
0x0400000a
    [   37.702609] Call trace:
    [   37.702851]  __switch_to+0xe4/0x138
    [   37.703166]  lock_timer_base+0x5c/0xa0
    [   37.703502]  try_to_del_timer_sync+0x30/0x98
    [   37.703883]  del_timer_sync+0x50/0x60
    [   37.704220]  schedule_timeout+0x19c/0x478
    [   37.704582]  clk_gate_endisable+0x2c/0xc8
    [  100.716421] rcu: INFO: rcu_preempt detected stalls on CPUs/tasks:
    [  100.716991] rcu:     3-...0: (0 ticks this GP)
idle=b52/1/0x4000000000000000 softirq=4194/4194 fqs=7921
    [  100.717799] rcu:     (detected by 0, t=25207 jiffies, g=3301, q=19)
    [  100.718334] Task dump for CPU 3:
    [  100.718632] init            R  running task        0     1      0
0x0400000a
    [  100.719260] Call trace:
    [  100.719500]  __switch_to+0xe4/0x138
    [  100.719816]  lock_timer_base+0x5c/0xa0
    [  100.720152]  try_to_del_timer_sync+0x30/0x98
    [  100.720533]  del_timer_sync+0x50/0x60
```

```
    [  100.720870]  schedule_timeout+0x19c/0x478
    [  100.721231]  clk_gate_endisable+0x2c/0xc8
    [  163.733075] rcu: INFO: rcu_preempt detected stalls on CPUs/tasks:
    [  163.733643] rcu:     3-...0: (0 ticks this GP)
idle=b52/1/0x4000000000000000 softirq=4194/4194 fqs=13833
    [  163.734462] rcu:     (detected by 0, t=44112 jiffies, g=3301, q=20)
    [  163.734997] Task dump for CPU 3:
    [  163.735294] init            R  running task        0     1      0
0x0400000a
    [  163.735921] Call trace:
    [  163.736160]  __switch_to+0xe4/0x138
    [  163.736475]  lock_timer_base+0x5c/0xa0
    [  163.736811]  try_to_del_timer_sync+0x30/0x98
    [  163.737192]  del_timer_sync+0x50/0x60
    [  163.737528]  schedule_timeout+0x19c/0x478
    [  163.737889]  clk_gate_endisable+0x2c/0xc8

    Change-Id: I663b6b3e0b081ad17bf2845629b34e2ec9d2d76d
    Signed-off-by: Wu Liangqing <wlq@rock-chips.com>
```

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.

```
i2cset  -yf 0 0x20 0x99 0x0 b
i2cset  -yf 0 0x20 0xa4 0x0 b
```

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

GPIO voltage of RK3566/RK3568 IO-Domain is configurated 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:

```
arch/arm64/boot/dts/rockchip/rk3568-evb.dtsi
&pmu_io_domains {
         status = "okay";
         pmuio2-supply = <&vcc3v3_pmu>;
         vccio1-supply = <&vccio_acodec>;
         vccio3-supply = <&vccio_sd>;
         vccio4-supply = <&vcc_3v3>;
         vccio5-supply = <&vcc_3v3>;
         vccio6-supply = <&vcc_3v3>;
         vccio7-supply = <&vcc_3v3>;
    };
```

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

```
&pmu_io_domains {
         status = "okay";
         pmuio2-supply = <&vcc3v3_pmu>;
         vccio1-supply = <&vccio_acodec>;
         vccio3-supply = <&vccio_sd>;
         vccio4-supply = <&vcc_1v8>;
         vccio5-supply = <&vcc_3v3>;
         vccio6-supply = <&vcc_1v8>;
         vccio7-supply = <&vcc_3v3>;
    };
```

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.

```
&pmu_io_domains {
         status = "okay";
         pmuio2-supply = <&vcc3v3_pmu>;
         vccio1-supply = <&vccio_acodec>;
         vccio3-supply = <&vccio_sd>;
         vccio4-supply = <&vcc_1v8>;
         vccio5-supply = <&vcc_3v3>;
         vccio6-supply = <&vcc_1v8>;
         vccio7-supply = <&vcc_3v3>;
    };
```

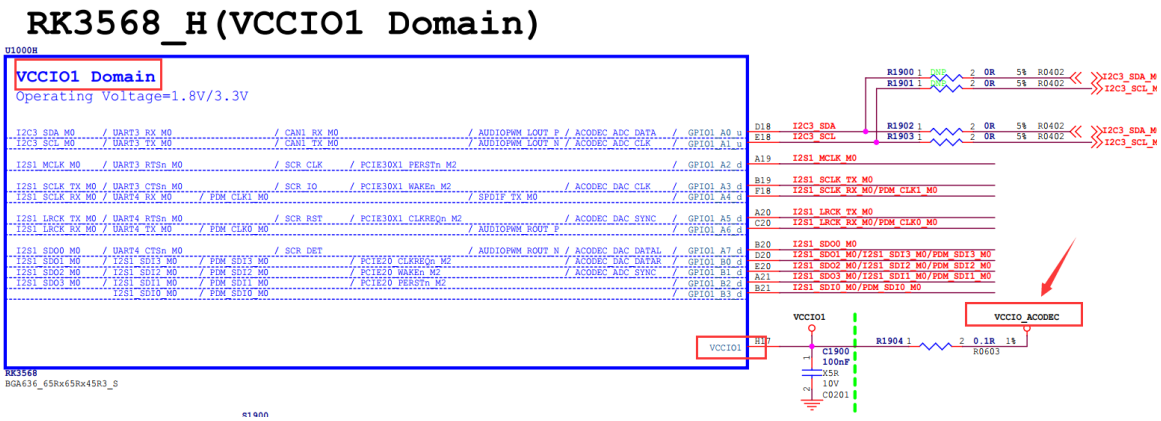
Take vccio1-supply as an example, firstly check the hardware schematic to confirm the configuration of vccio1 power domain (VCCIO1) as shown in the figure.

IO Power Domain Map

Updates must be Revision accordingly!

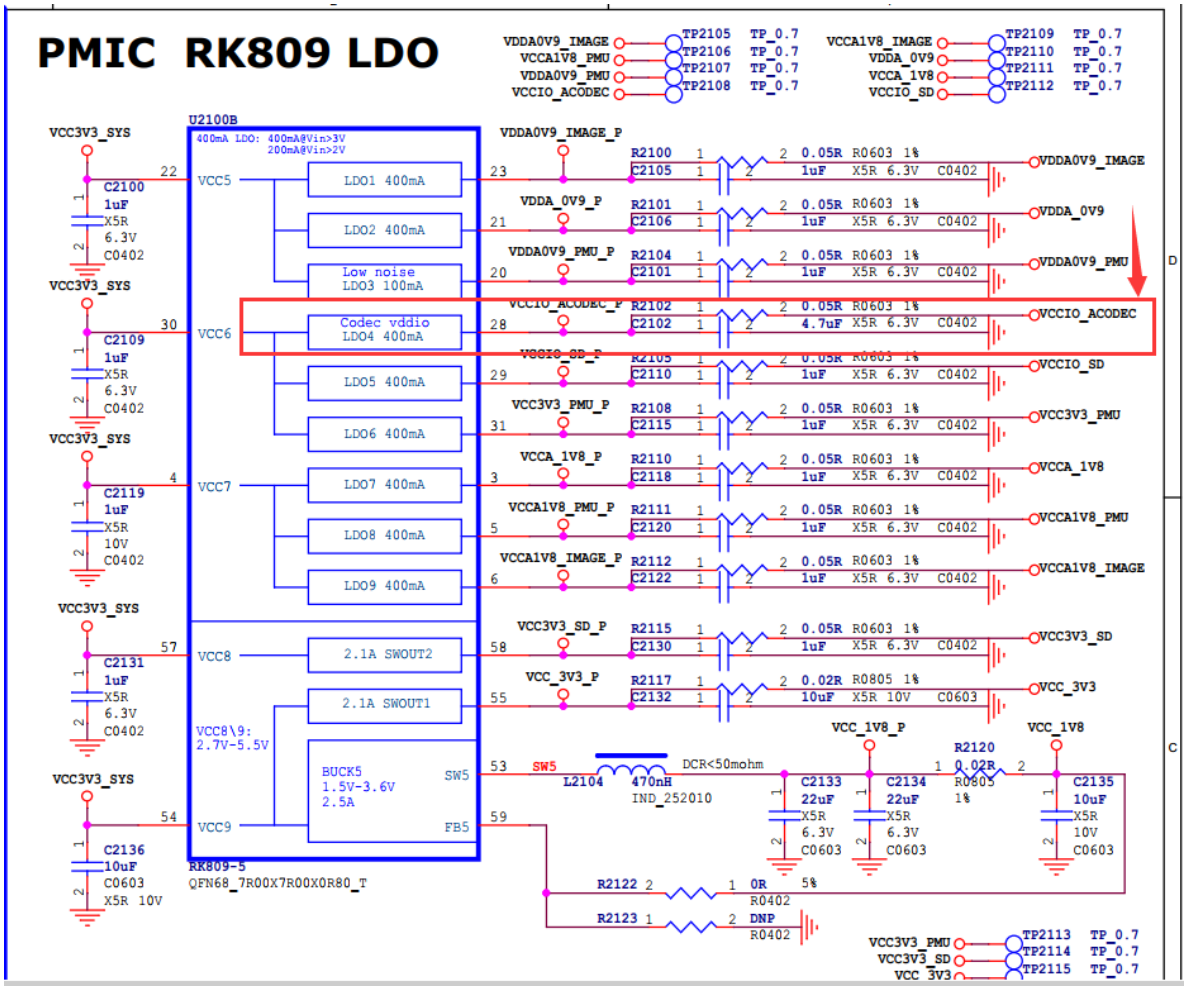
IO Domain	Pin Num	Support IO Voltage		Actual assigned IO Domain Voltage			Notes
		3.3V	1.8V	Supply Power Net Name	Power Source	Voltage	
PMUIO1	Pin Y20	✓	✗	VCC3V3_PMU	VCC3V3_PMU	3.3V	
PMUIO2	Pin W19	✓	✓	VCC3V3_PMU	VCC3V3_PMU	3.3V	
VCCIO1	Pin H17	✓	✓	VCCIO_ACODEC	VCCIO_ACODEC	3.3V	
VCCIO2	Pin H18	✓	✓	VCCIO_FLASH	VCC_1V8	1.8V	PIN "FLASH_VOL_SEL" must be logic High if VCCIO_FLASH=3.3V,FLASH_VOL_SEL must be logic lo
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:



We can find the power of `VCCIO1` 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.

```
vccio_acodec: LDO_REG4 {

                regulator-always-on;
        regulator-boot-on;
        regulator-min-microvolt = <3300000>;
        regulator-max-microvolt = <3300000>;
        regulator-name = "vccio_acodec";
        regulator-state-mem {
                regulator-off-in-suspend;
        };
};
```

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

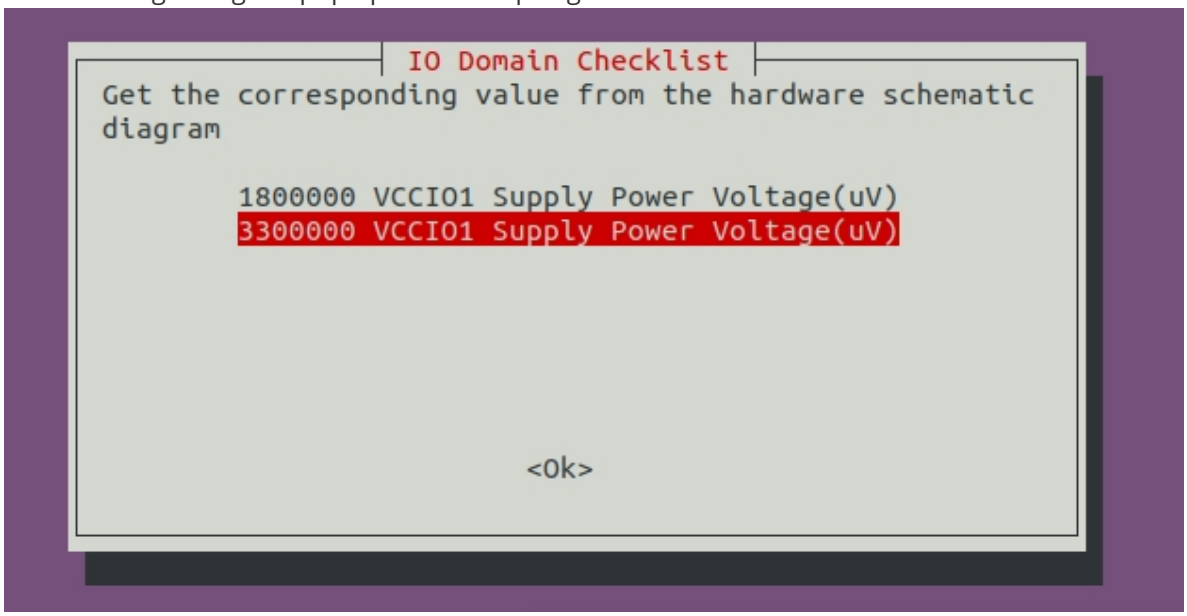
```
&pmu_io_domains {
        status = "okay";
        pmuio2-supply = <&vcc3v3_pmu>;
        vccio1-supply = <&vccio_acodec>;
        vccio3-supply = <&vccio_sd>;
        vccio4-supply = <&vcc_1v8>;
        vccio5-supply = <&vcc_3v3>;
        vccio6-supply = <&vcc_1v8>;
        vccio7-supply = <&vcc_3v3>;
    };
```

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-domian 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:

```
[   21.449927] rcu: INFO: rcu_preempt detected stalls on CPUs/tasks:
[   21.449935] rcu: INFO: rcu_sched detected stalls on CPUs/tasks:
```

```
[   21.449960] rcu:     1-...0: (1 GPs behind) idle=486/1/0x4000000000000000
softirq=29/29 fqs=2057
[   21.449989] rcu:     1-...0: (87 ticks this GP) idle=486/1/0x4000000000000000
softirq=7/29 fqs=2100
[   21.450003] rcu:     2-...0: (100 ticks this GP)
idle=412/1/0x4000000000000000 softirq=7/52 fqs=2100
[   21.450010] rcu:     (detected by 0, t=6302 jiffies, g=-1183, q=1)
[   21.450023] Task dump for CPU 1:
[   21.450031] rk-pcie         R  running task        0    53      2 0x0000002a
[   21.450048] Call trace:
[   21.450065] rcu:     2-...0: (15 ticks this GP) idle=412/1/0x4000000000000000
softirq=37/52 fqs=2057
[   21.450071] rcu:     (detected by 3, t=6302 jiffies, g=-1147, q=720)
[   21.450084] Task dump for CPU 1:
[   21.450092] rk-pcie         R  running task        0    53      2 0x0000002a
[   21.450125]  __switch_to+0xe4/0x138
[   21.450139]  kthread+0x12c/0x158
[   21.450150]  ret_from_fork+0x10/0x18
[   21.450156] Task dump for CPU 2:
[   21.450163] kworker/u8:1    R  running task        0    32      2 0x0000002a
[   21.450178] Call trace:
[   21.450199] Workqueue: events_unbound enable_ptr_key_workfn
[   21.450216]  __switch_to+0xe4/0x138
[   21.450224] Call trace:
[   21.450239]  kthread+0x12c/0x158
[   21.450249]  ret_from_fork+0x10/0x18
[   21.450256] Task dump for CPU 2:
[   21.450270]  __switch_to+0xe4/0x138
[   21.450283]  bp_hardening_data+0x0/0x10
[   21.450300] kworker/u8:1    R  running task        0    32      2 0x0000002a
[   21.450327] Workqueue: events_unbound enable_ptr_key_workfn
[   21.450340] Call trace:
[   21.450356]  __switch_to+0xe4/0x138
[   21.450372]  bp_hardening_data+0x0/0x10
[   48.026601] watchdog: BUG: soft lockup - CPU#3 stuck for 22s! [swapper/0:1]
[   48.026638] Modules linked in:
[   48.026664] CPU: 3 PID: 1 Comm: swapper/0 Not tainted 4.19.172 #10
[   48.026676] Hardware name: Rockchip RK3568 EVB1 DDR4 V10 Board (DT)
[   48.026692] pstate: 80c00009 (Nzcv daif +PAN +UAO)
[   48.026716] pc : smp_call_function_many+0x314/0x350
[   48.026753] lr : smp_call_function_many+0x2d4/0x350
  ... ...
[   48.032654] Call trace:
[   48.032678]  smp_call_function_many+0x314/0x350
[   48.032700]  smp_call_function+0x38/0x68
[   48.032729]  on_each_cpu+0x30/0x80
[   48.032759]  clock_was_set+0x1c/0x28
[   48.032787]  do_settimeofday64+0x130/0x180
[   48.032819]  rtc_hctosys+0x78/0x118
[   48.032846]  __rtc_register_device+0xd4/0x160
[   48.032874]  rk808_rtc_probe+0x174/0x2c0
[   48.032904]  platform_drv_probe+0x50/0xa8
[   48.032922]  really_probe+0x228/0x2a0
[   48.032940]  driver_probe_device+0x58/0x100
[   48.032969]  __device_attach_driver+0x90/0xc0
[   48.032995]  bus_for_each_drv+0x70/0xc8
[   48.033023]  __device_attach+0xec/0x148
[   48.033051]  device_initial_probe+0x10/0x18
```

```
[   48.033077]  bus_probe_device+0x94/0xa0
[   48.033103]  device_add+0x384/0x698
[   48.033130]  platform_device_add+0x108/0x248
[   48.033160]  mfd_add_device+0x2d8/0x358
[   48.033177]  mfd_add_devices+0xb0/0x170
[   48.033211]  devm_mfd_add_devices+0x78/0xe0
[   48.033238]  rk808_probe+0x724/0x780
[   48.033264]  i2c_device_probe+0x200/0x278
[   48.033296]  really_probe+0x228/0x2a0
[   48.033323]  driver_probe_device+0x58/0x100
[   48.033350]  __device_attach_driver+0x90/0xc0
[   48.033377]  bus_for_each_drv+0x70/0xc8
[   48.033404]  __device_attach+0xec/0x148
[   48.033422]  device_initial_probe+0x10/0x18
[   48.033438]  bus_probe_device+0x94/0xa0
[   48.033463]  device_add+0x384/0x698
[   48.033488]  device_register+0x1c/0x28
[   48.033516]  i2c_new_device+0x1c0/0x398
[   48.033546]  of_i2c_register_devices+0x134/0x160
[   48.033573]  i2c_register_adapter+0x150/0x408
[   48.033601]  __i2c_add_numbered_adapter+0x5c/0xa8
[   48.033628]  i2c_add_adapter+0xa4/0xd8
[   48.033655]  rk3x_i2c_probe+0x324/0x428
[   48.033674]  platform_drv_probe+0x50/0xa8
[   48.033708]  really_probe+0x228/0x2a0
[   48.033736]  driver_probe_device+0x58/0x100
[   48.033764]  device_driver_attach+0x6c/0x78
[   48.033791]  __driver_attach+0xb0/0xf0
[   48.033818]  bus_for_each_dev+0x68/0xc8
[   48.033844]  driver_attach+0x20/0x28
[   48.033870]  bus_add_driver+0xf8/0x1f0
[   48.033898]  driver_register+0x60/0x110
[   48.033917]  __platform_driver_register+0x40/0x48
[   48.033938]  rk3x_i2c_driver_init+0x18/0x20
[   48.033966]  do_one_initcall+0x48/0x240
[   48.033997]  kernel_init_freeable+0x210/0x37c
[   48.034024]  kernel_init+0x10/0x108
[   48.034052]  ret_from_fork+0x10/0x18
```

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:

```
&pcie3x1 {
   status = "disabled";
};

&pcie3x2 {
   status = "disabled";
};
```

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

```
RKDocs/android/Rockchip_Introduction_Android_Samba_CN.pdf
```

NFS boot

Refer to the documents and patches:

```
RKDocs/android/patches/customized_functions/nfs_boot_patch_v1.1.0.zip
```

Multi-screen display and touch

Referenced document

```
RKDocs\android\patches\customized_functions/Android11异显开发说明.zip
```

Different screen different sound

Referenced document

```
RKDocs/android/patches/customized_functions/Dual_Audio_v1.0.zip
```

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 update
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:

```
openjdk-8-jre-headless_8u45-b14-1_amd64.deb with SHA256
0f5aba8db39088283b51e00054813063173a4d8809f70033976f83e214ab56c0
openjdk-8-jre_8u45-b14-1_amd64.deb with SHA256
9ef76c4562d39432b69baf6c18f199707c5c56a5b4566847df908b7d74e15849
openjdk-8-jdk_8u45-b14-1_amd64.deb with SHA256
6e47215cf6205aa829e6a0a64985075bd29d1f428a4006a80c9db371c2fc3c4c
```

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 {downloaded.deb file}
```

3. Install the packages:

```
    sudo apt-get update
```

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

```
    sudo dpkg -i {downloaded.deb file}
```

To fix missing dependencies:

```
sudo apt-get -f install
```

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

```
sudo update-alternatives --config javasudo update-alternatives --config javac
```

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.

```
sudo apt-get install git-core gnupg flex bison gperf build-essential zip curl
zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386 lib32ncurses5-dev x11proto-
core-dev libx11-dev lib32z-dev ccache libgl1-mesa-dev libxml2-utils xsltproc
unzip python-pyelftools python3-pyelftools device-tree-compiler libfdt-dev
libfdt1
```

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.

```
# adb protocol on passion (Rockchip products)
SUBSYSTEM=="usb", ATTR{idVendor}=="2207", ATTR{idProduct}=="0010", MODE="0600",
OWNER="<username>"
```

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:

```
ssh-keygen -t rsa -C "user@host"
```

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:

```
$sudo aptitude install keychain
```

2. Configure to use the key:

```
$vim ~/.bashrc
```

Add the following command:

```
eval `keychain --eval ~/.ssh/id_rsa`
```

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:

```
Agent admitted failture to sign using the key
```

Input the following command at console can fix it.

```
ssh-add ~/.ssh/id_rsa
```

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

You can refer to ssh_config document to configure ssh.

```
~$ man ssh_config
```

Use the following commands to configure ssh for current user.

```
~$ cp /etc/ssh/ssh_config ~/.ssh/config
~$ vi .ssh/config
```

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.