

密级状态：绝密( ) 秘密( ) 内部( ) 公开(√)

# Rockchip Android 11.0 SDK开发指南

文件状态：  
☐ 草稿  
☒ 正式发布  
☐ 正在修改

文件标识:	RK-KF-YF-279
当前版本:	V0.0.2
作者:	吴良清
完成日期:	2021-01-06
审核:	陈海燕
审核日期:	2021-01-06

版本号	作者	修改日期	修改说明	备注
V0.0.1	吴良清	2020-12-25	发布RK3566/RK3568 Alpha版本	
V0.0.2	卞金晨	2021-01-06	发布PX30/RK3326 Beta版本	

文档问题反馈：[wlq@rock-chips.com](mailto:wlq@rock-chips.com)

## 免责声明

本文档按“现状”提供，瑞芯微电子股份有限公司（“本公司”，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

## 商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自所有者所有。

## 版权所有 © 2020 瑞芯微电子股份有限公司

超越合理使用范畴，非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址：福建省福州市铜盘路软件园A区18号

网址：[www.rock-chips.com](http://www.rock-chips.com)

客户服务电话：+86-4007-700-590

客户服务传真: +86-591-83951833  
客户服务邮箱: [fae@rock-chips.com](mailto:fae@rock-chips.com)

# Rockchip Android 11.0 SDK支持芯片

芯片平台	是否支持	SDK版本
RK3566	支持	RKR1
RK3568	支持	RKR1
PX30/RK3326	支持	RKR1

# Rockchip Android 11.0 SDK代码下载编译

## 代码下载

### 下载地址

```
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
```

### 服务器镜像下载

```
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
```

注, repo是google用Python脚本写的调用git的一个脚本, 主要是用来下载、管理Android项目的软件仓库, 其下载地址如下:

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

为方便客户快速获取SDK源码, 瑞芯微技术窗口通常会提供对应版本的SDK初始压缩包。以 `ROCKCHIP_ANDROID11.0_SDK_RELEASE.tar.gz.*` 为例, 拷贝到该初始化包后, 通过如下命令可检出源码:

```
mkdir ROCKCHIP_ANDROID11.0_SDK_RELEASE  
cat ROCKCHIP_ANDROID11.0_SDK_RELEASE.tar.gz* | tar -zx -C  
ROCKCHIP_ANDROID11.0_SDK_RELEASE  
cd ROCKCHIP_ANDROID11.0_SDK_RELEASE  
.repo/repo/repo sync -l  
.repo/repo/repo sync -c
```

## 代码编译

### 一键编译命令

```
./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 = build kernel dts name
    -V = build version
    -J = build jobs
    -----大家可以按需使用，不用记录uboot/kernel编译命令了-----
)

=====
请注意使用一键编译命令之前需要设置环境变量，选择好自己需要编译的平台，举例：
source build/envsetup.sh
lunch rk3566_rgo-userdebug
=====
```

## 各个平台编译命令汇总

---

Soc	类型	机型	Android	一键编译	kernel编译	uboot编译
RK3566	平板	样机	build/envsetup.sh;lunch rk3566_rgo-userdebug	./build.sh -AUCKu	make ARCH=arm64 rockchip_defconfig android-11.config rk356x.config;make ARCH=arm64 rk3566-rk817-tablet.img -j24	./make.sh rk3566
RK3568	开发板	evb1-ddr4-v10	build/envsetup.sh;lunch rk3568_r-userdebug	./build.sh -AUCKu	make ARCH=arm64 rockchip_defconfig rk356x_evb.config android-11.config rk356x.config;make ARCH=arm64 rk3568-evb1-ddr4-v10.img -j24	./make.sh rk3568
RK3566	开发板	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 rk356x.config;make ARCH=arm64 rk3566-evb2-lp4x-v10.img -j24	./make.sh rk3566
RK3326	平板	863 样机	build/envsetup.sh;lunch rk3326_rgo-userdebug	./build.sh -AUCKu	make ARCH=arm64 rockchip_defconfig android-11-go.config rk3326.config;make ARCH=arm64 rk3326-863-lp3-v10-rksip1.img -j24	./make.sh rk3326
PX30	开发板	px30-evb-ddr3-v10-avb	build/envsetup.sh;lunch PX30_Android11-userdebug	./build.sh -AUCKu	make ARCH=arm64 rockchip_defconfig android-11-go.config rk3326.config;make ARCH=arm64 px30-evb-ddr3-v10-avb.img -j24	./make.sh px30

## 其他编译说明

### Android11.0不能直接烧写kernel.img和resource.img

Android11.0的kernel.img和resource.img包含在boot.img中，更新编译kernel后需要在android根目录下执行./mkimage.sh重新打包boot.img。打包后烧写rockdev下面的boot.img，可以使用如下方法单独编译kernel。

### 单独编译kernel生成boot.img

编译的原理：在kernel目录下将编译生成的 `kernel1.img` 和 `resource.img` 替换到旧的 `boot.img` 中，所以编译的时候需要用 `BOOT_IMG=xxx` 参数指定boot\_sample.img的路径，命令如下：  
以 RK3566 样机为例，编译时替换对应的boot.img及dts：

```
cd kernel
make ARCH=arm64 rockchip_defconfig android-11.config rk356x.config
make ARCH=arm64 BOOT_IMG=boot_sample.img rk3566-rk817-tablet.img -j24
```

编译后可以直接烧写kernel目录下的boot.img（**注意：32bit的平台是zboot.img，如3126c**(/developer\_guide/android\_10\_development\_guide/android\_tool\_v2.79.png)）到机器的boot位置，烧写时**请先加载分区表（parameter.txt）**，以免烧写位置错误。

## 固件烧写

### 固件烧写工具

Android11的USB驱动DriverAssitant需要更新到V5.1.1版本，可以参考下面的工具章节进行更新。  
Windows烧写工具：

RKTools/windows/AndroidTool/AndroidTool\_Release\_v2.79

#	地址	名字	路径
1	0x00000000	Loader	Z:\2_rk3568_android11\uboot\rk356x...
2	0x00000000	Parameter	Z:\2_rk3568_android11\rockdev\Image...
3	0x00004000	Uboot	Z:\2_rk3568_android11\uboot\uboot.img
4	0x00008000	Misc	Z:\2_rk3568_android11\rockdev\Image...
5	0x0000A000	Dtbo	Z:\2_rk3568_android11\rockdev\Image...
6	0x0000C000	vbmeta	Z:\2_rk3568_android11\rockdev\Image...
7	0x0000C800	Boot	Z:\2_rk3568_android11\kernel\boot.img
8	0x0001E800	Recovery	Z:\2_rk3568_android11\rockdev\Image...
9	0x001D6800	Super	Z:\2_rk3568_android11\rockdev\Image...

RKTools/linux/Linux\_Upgrade\_Tool/Linux\_Upgrade\_Tool\_v1.56

在下文工具说明章节有详细说明

### 固件说明

完整编译后会生成如下文件：（以RK3566为例，这里lunch的是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
```

工具烧写如下文件即可：(RK3566/RK3568不需要烧写trust.img)

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

也可以直接烧写 update.img

## 固件说明

固件	说明
boot.img	包含ramdis、kernel、dtb
boot-debug.img	与boot.img的差别是user固件可以烧写这个boot.img进行root权限操作
dtbo.img	Device Tree Overlays 参考下面的dtbo章节说明
config.cfg	烧写工具的配置文件，可以直接导入烧写工具显示需要烧写的选项
MiniLoaderAll.bin	包含一级loader
misc.img	包含recovery-wipe开机标识信息，烧写后会进行recovery
parameter.txt	包含分区信息
pcba_small_misc.img	包含pcba开机标识信息，烧写后会进入简易版pcba模式
pcba_whole_misc.img	包含pcba开机标识信息，烧写后会进入完整版pcba模式
recovery.img	包含recovery-ramdis、kernel、dtb
super.img	包含odm、product、vendor、system、system_ext分区内容
trust.img	包含BL31、BL32 RK3566/RK3568没有生成这个固件，不需要烧写
uboot.img	包含uboot固件
vbmeta.img	包含avb校验信息，用于AVB校验
update.img	包含以上需要烧写的img文件，可以用于工具直接烧写整个固件包

## fastboot烧写动态分区

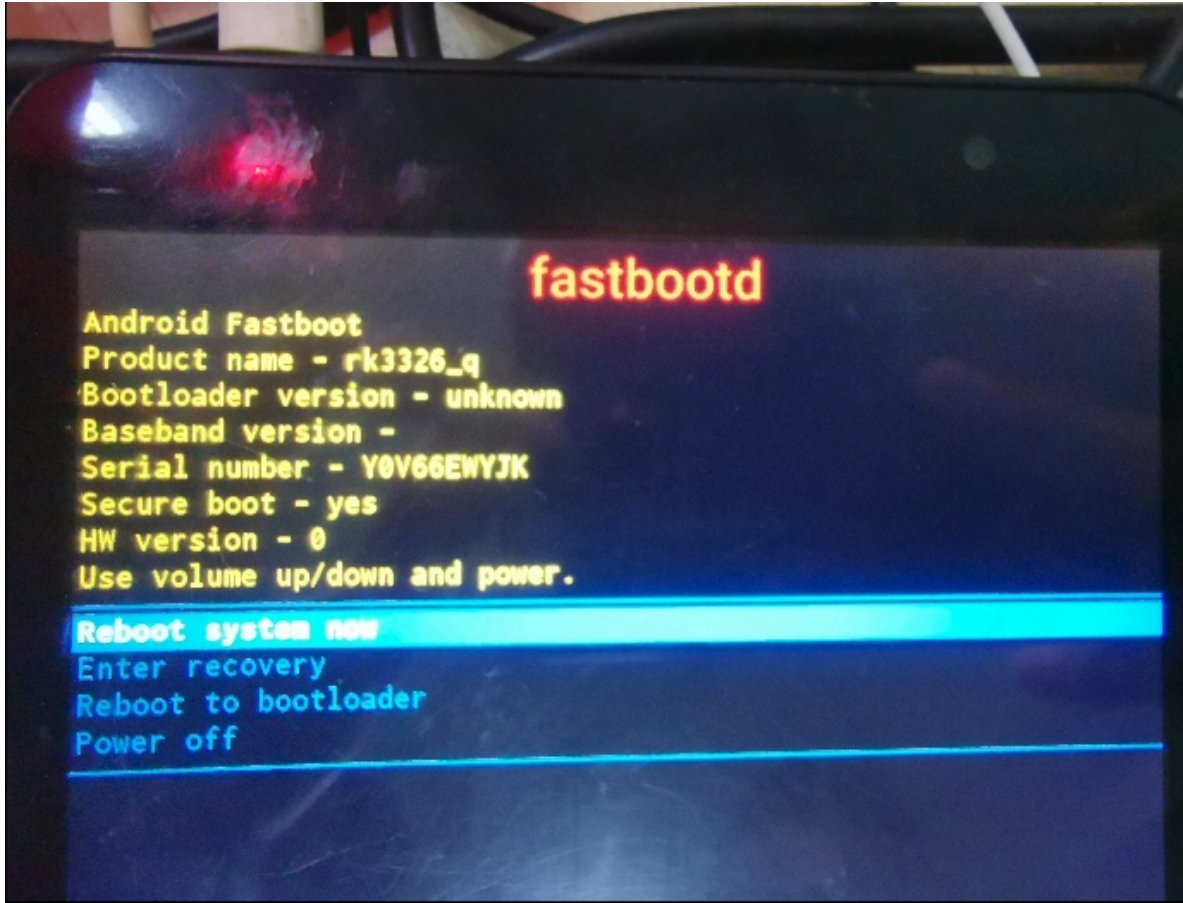
R的新设备支持动态分区，已经移除了system/vendor/odm/product/system\_ext分区，请烧写super.img，单独烧写system/vendor/odm等（可以在out下面找到对应img文件）可以用fastbootd，要求adb和fastboot版本均为最新，SDK提供了编译好的工具包：

```
RKTools/linux/Linux_adb_fastboot (Linux_x86版本)
RKTools/windows/adb_fastboot (windows_x86版本)
```

- 使用命令烧写动态分区:

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

注: 进入fastbootd模式后, 屏幕上会显示相关设备信息, 如图所示:



注: 非动态分区使用fastboot, 请进入bootloader:

```
adb reboot bootloader
```

烧写GSI的方法:

- 确认机器解锁后, 进入fastbootd, 只需要烧写GSI中的system.img及固件中的misc.img, 烧写后会进入recovery进行恢复出厂设置。下面附上整个烧写流程:

1. 重启至bootloader, 未解锁->机器解锁:

```
adb reboot bootloader
fastboot oem at-unlock-vboot ## 对于烧写过avb公钥的客户, 请参考对应的文档解锁。
```

2. 恢复出厂设置, 重启至fastbootd:

```
fastboot flash misc misc.img
fastboot reboot fastboot ## 此时将进入fastbootd
```

### 3. 开始烧写GSI

```
fastboot delete-logical-partition product ## (可选)对于分区空间紧张的设备,可以先执行
本条命令删除product分区后再烧写GSI
fastboot flash system system.img
fastboot reboot ## 烧写成功后,重启
```

- 注: 也可以使用DSU(Dynamic System Updates)烧写GSI, 目前Rockchip平台已经默认支持DSU。由于该功能需要消耗大量内存, 不建议1G DDR及以下的设备使用, 有关DSU的说明和使用, 请参考Android官网:  
<https://source.android.com/devices/tech/ota/dynamic-system-updates>
- 注1: VTS测试时, 需要同时烧写编译出的boot-debug.img到boot分区;
- 注2: CTS-ON-GSI测试时则不需要烧boot-debug.img;
- 注3: 测试时请使用Google官方发布的, 带有-signed结尾的GSI镜像;

## 使用DTBO功能

Android 10.0及以上支持Device Tree Overlays功能, 开发过程体现在需要烧写dtbo.img, 用于多个产品间的兼容等。

修改方法:

1. 找到(或指定)模板文件:

```
get_build_var PRODUCT_DTBO_TEMPLATE
```

例如:

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

2. 添加或修改需要的节点:

例如:

```
/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>;
};
```

**注: 使用dtbo时一定要确保dts中存在alias, 否则无法成功overlay**

## 修改fstab文件

1. 找到(或指定)模板文件:

```
get_build_var PRODUCT_FSTAB_TEMPLATE
```

例如:

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

2. 修改: 添加分区挂载、修改swap\_zram参数, 修改data分区格式等

## 修改parameter.txt

Android 11添加了生成parameter.txt的工具, 支持根据配置参数编译出parameter.txt。如果没有配置模板文件, 则会寻找添加修改好的parameter.txt文件。

1. 找到(或指定)模板文件:

```
get_build_var PRODUCT_PARAMETER_TEMPLATE
```

例如:

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

2. 修改配置分区大小(例如):

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

3. 不使用parameter生成工具:

添加一个parameter.txt文件到你的device目录下即可:

例如: device/rockchip/rk3326/rk3326\_q/parameter.txt

4. 仅使用工具生成parameter.txt(例如):

```
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
```

注: 如果需要大版本OTA升级, 请直接使用之前版本的parameter.txt

# Android常用配置

## 新建产品lunch

以RK356x平台新建rk3568\_r产品为例, 分以下步骤:

1) 修改device/rockchip/rk356x/AndroidProducts.mk增加rk3568\_r的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) 在device/rockchip/rk356x目录下新建rk3568\_r目录

参考device/rockchip/rk356x下已有的rk3566\_r产品目录新建, 可以先直接拷贝rk3566\_r为rk3568\_r, 然后将rk3568\_r目录下的所有 rk3566\_r 字符改为 rk3568\_r

# Kernel dts说明

## 新建产品dts

产品新建dts可以根据下表的配置选择对应的dts作为参考。

Soc	PMIC	DDR	开发板类型	机型	DTS
RK3566	RK817	LPDDR4	平板	样机	rk3566-rk817-tablet
RK3566	RK809	LPDDR4	开发板	RK3566 EVB2	rk3566-evb2-lp4x-v10
RK3568	RK809	DDR4	开发板	RK3568 EVB1	rk3566-evb1-ddr4-v10

# 文档说明

## 外设支持列表

DDR/EMMC/NAND FLASH/WIFI/3G/CAMERA的支持列表实时更新在redmine上, 链接如下:

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

## Android文档

---

RKDocs\android

## Android\_SELinux(Sepolicy)开发指南

---

RKDocs/android/Rockchip\_Developer\_Guide\_Android\_SELinux(Sepolicy)\_CN.pdf

## Wi-Fi文档

---

RKDocs/android/wifi/

- |— Rockchip\_Introduction\_Android10.0\_WIFI\_Configuration\_CN&EN.pdf
- |— Rockchip\_Introduction\_REALTEK\_WIFI\_Driver\_Porting\_CN&EN.pdf

## 3G/4G模块说明文档

---

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文档

---

RKDocs\common

## DDR相关文档

---

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模块文档

---

RKDocs/common/Audio/

- |— Rockchip\_Developer\_Guide\_Audio\_Call\_3A\_Algorithm\_Integration\_and\_Parameter\_Debugging\_CN.pdf
- |— Rockchip\_Developer\_Guide\_Linux4.4\_Audio\_CN.pdf
- |— Rockchip\_Developer\_Guide\_RK817\_RK809\_Codec\_CN.pdf

## CRU模块文档

---

RKDocs/common/CRU/

- └─ Rockchip\_Developer\_Guide\_Linux3.10\_Clock\_CN.pdf
- └─ Rockchip\_RK3399\_Developer\_Guide\_Linux4.4\_Clock\_CN.pdf

## GMAC模块文档

---

RKDocs/common/GMAC/

- └─ Rockchip\_Developer\_Guide\_Ethernet\_CN.pdf

## PCie模块文档

---

RKDocs/common/PCie/

- └─ Rockchip-Developer-Guide-linux4.4-PCie.pdf

## I2C模块文档

---

RKDocs/common/I2C/

- └─ Rockchip\_Developer\_Guide\_I2C\_CN.pdf

## PIN-Ctrl GPIO模块文档

---

RKDocs/common/PIN-Ctrl/

- └─ Rockchip-Developer-Guide-Linux-Pin-Ctrl-CN.pdf

## SPI模块文档

---

RKDocs/common/SPI/

- └─ Rockchip-Developer-Guide-linux4.4-SPI.pdf

## Sensor模块文档

---

RKDocs/common/Sensors/

- └─ Rockchip\_Developer\_Guide\_Sensors\_CN.pdf

## IO-Domain模块文档

---

RKDocs/common/IO-Domain/

- └─ Rockchip\_Developer\_Guide\_Linux\_IO\_DOMAIN\_CN.pdf

## Leds模块文档

---

RKDocs/common/Leds/

- └─ Rockchip\_Introduction\_Leds\_GPIO\_Configuration\_for\_Linux4.4\_CN.pdf

## Thermal温控模块文档

---

RKDocs/common/Thermal/

- └─ Rockchip-Developer-Guide-Linux4.4-Thermal-CN.pdf
- └─ Rockchip-Developer-Guide-Linux4.4-Thermal-EN.pdf

## PMIC电源管理模块文档

---

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模块文档

---

RKDocs/common/MCU/

- └─ Rockchip\_Developer\_Guide\_MCU\_EN.pdf

## 功耗与休眠模块文档

---

RKDocs/common/power/

- └─ Rockchip\_Developer\_Guide\_Power\_Analysis\_EN.pdf
- └─ Rockchip\_Developer\_Guide\_Sleep\_and\_Resume\_CN.pdf

## UART模块文档

---

RKDocs/common/UART/

- └─ Rockchip-Developer-Guide-linux4.4-UART.pdf
- └─ Rockchip-Developer-Guide-RT-Thread-UART.pdf

## DVFS CPU/GPU/DDR变频相关文档

---

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模块文档

---

RKDocs/common/MMC

- └─ Rockchip-Developer-Guide-linux4.4-SDMMC-SDIO-eMMC.pdf

## PWM模块文档

---

RKDocs/common/PWM/

- |— Rockchip-Developer-Guide-Linux-PWM-CN.pdf
- |— Rockchip\_Developer\_Guide\_PWM\_IR\_CN.pdf

## USB模块文档

---

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功能文档

---

RKDocs/common/hdmi-in/

- |— Rockchip\_Developer\_Guide\_HDMI\_IN\_CN.pdf

## 安全模块文档

---

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介绍文档

---

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

## Trust介绍文档

---

RKDocs/common/TRUST/

- |— Rockchip\_Developer\_Guide\_Trust\_CN.pdf
- |— Rockchip\_Developer\_Guide\_Trust\_EN.pdf

## Camera文档

---

RKDocs\common\camera\HAL3\

## 工具文档

---

RKDocs\common\RKTools manuals

## PCBA开发使用文档

---

RKDocs\android\Rockchip\_Developer\_Guide\_PCBA\_Test\_Tool\_CN&EN.pdf

## 显示屏驱动调试指南

---

RKDocs\common\display\Rockchip\_Developer\_Guide\_DRM\_Panel\_Porting\_CN.pdf

## HDMI调试指南

---

RKDocs\common\display\Rockchip\_Developer\_Guide\_HDMI\_Based\_on\_DRM\_Framework\_CN.pdf

## 图像显示DRM Hardware Composer (HWC) 问题分析排查

---

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

## DRM显示开发指南

---

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

## RGA相关问题分析排查

---

RKDocs\common\display\Rockchip\_RGA\_FAQ.pdf

## 图形显示框架常见问题分析

---

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

RKDocs\common\display\Rockchip\_Trouble\_Shooting\_Graphics

## 工具使用

---

## StressTest

---

设备上使用Stresstest 工具，对待测设备的各项功能进行压力测试，确保整个系统运行的稳定性。SDK 通过打开计算器应用，输入“83991906=” 暗码，可启动StressTest应用，进行各功能压力测试。

Stresstest 测试工具测试的内容主要包括：

## 模块相关

- Camera 压力测试：包括Camera 打开关闭，Camera 拍照以及Camera 切换。
- Bluetooth 压力测试：包括Bluetooth 打开关闭。
- Wi-Fi 压力测试： 包括Wi-Fi 打开关闭，（ ping 测试以及iperf 测试待加入）。

## 非模块相关

- 飞行模式开关测试
- 休眠唤醒拷机测试
- 视频拷机测试
- 重启拷机测试
- 恢复出厂设置拷机测试
- Arm 变频测试
- Gpu 变频测试
- DDR 变频测试

## PCBA测试工具

---

PCBA 测试工具用于帮助在量产的过程中快速地甄别产品功能的好坏，提高生产效率。目前包括屏幕（LCD）、无线（Wi-Fi）、蓝牙（bluetooth）、DDR/EMMC 存储、SD 卡（sdcard）、USB HOST、按键（KEY），喇叭耳机（Codec）测试项目。

这些测试项目包括自动测试项和手动测试项，无线网络、DDR/EMMC、以太网为自动测试项，按键、SD卡、USB HOST、Codec、为手动测试项目。

具体PCBA功能配置及使用说明，请参考：

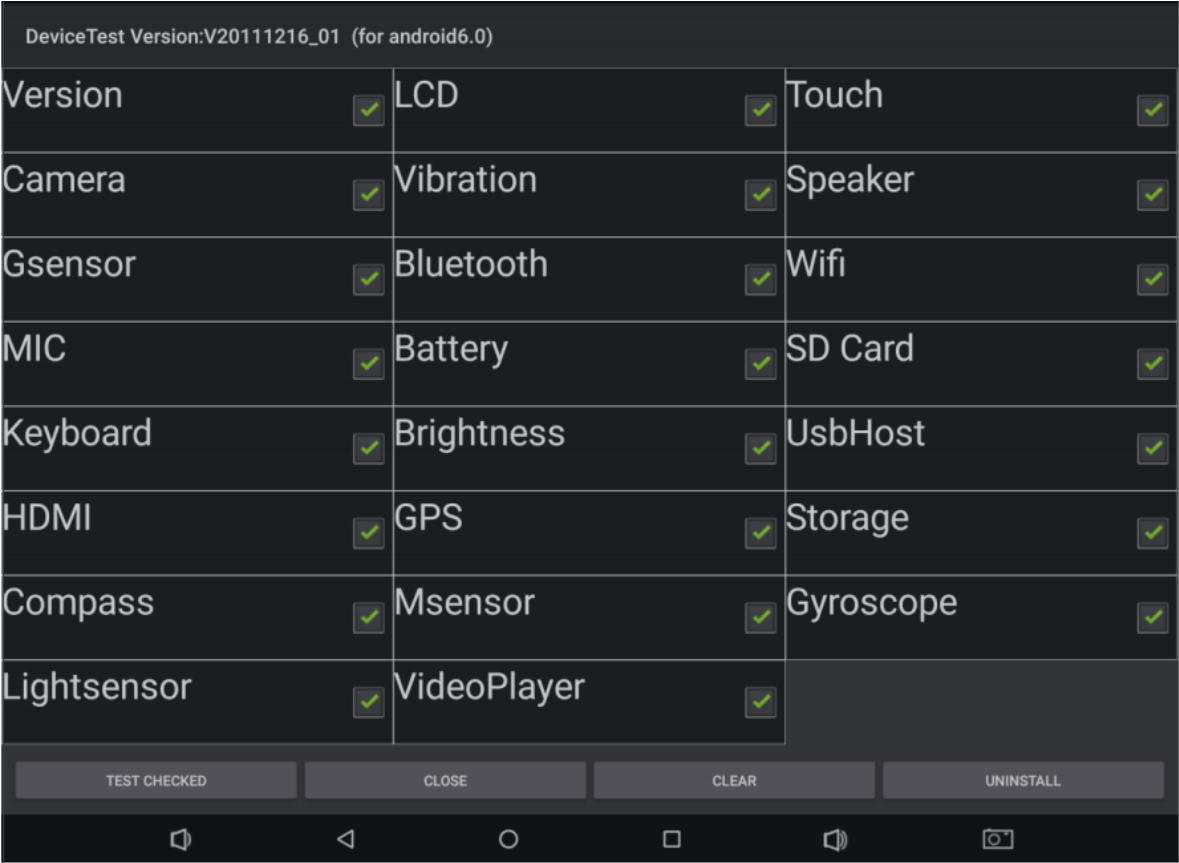
RKDocs\android\Rockchip\_Developer\_Guide\_PCBA\_Test\_Tool\_CN&EN.pdf\_v1.1\_20171222.pdf。

## DeviceTest

---



DeviceTest 用于工厂整机测试，主要测试装成整机以后外围器件是否正常。SDK 通过打开计算器，输入暗码“000.”进入 DeviceTest，如下所示：



在产线可以根据这个界面进行对应外设的测试，测试时点击“TEST CHECKED”对所测项目逐项进行测试，测试如果成功点击 pass，失败点击 failed，最终结果会显示在界面上，如下图所示，红色为 failed 项，其余为通过项，工厂可根据测试结果进行相应的维修。另外，如果客户需要对该工具进行定制，请联系 FAE 窗口申请对应的源码。

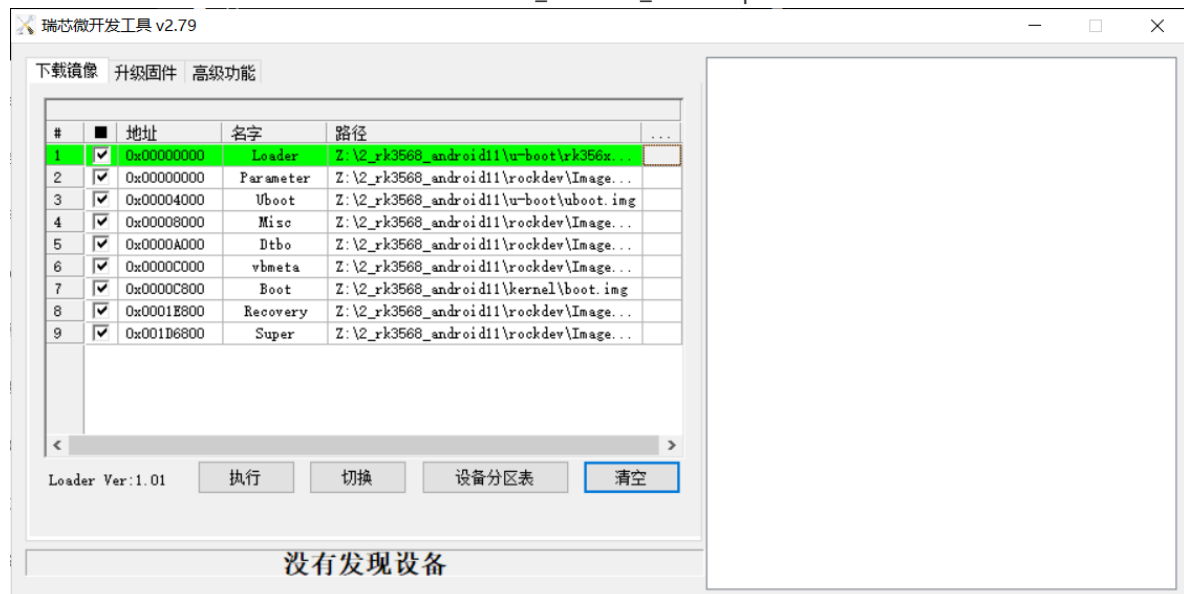
## USB驱动

Rockchip USB驱动安装包，包括ADB、固件烧写驱动

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

## 开发烧写工具

### Windows版本



## Linux版本

```
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> <BlockLen> [--Force]
```

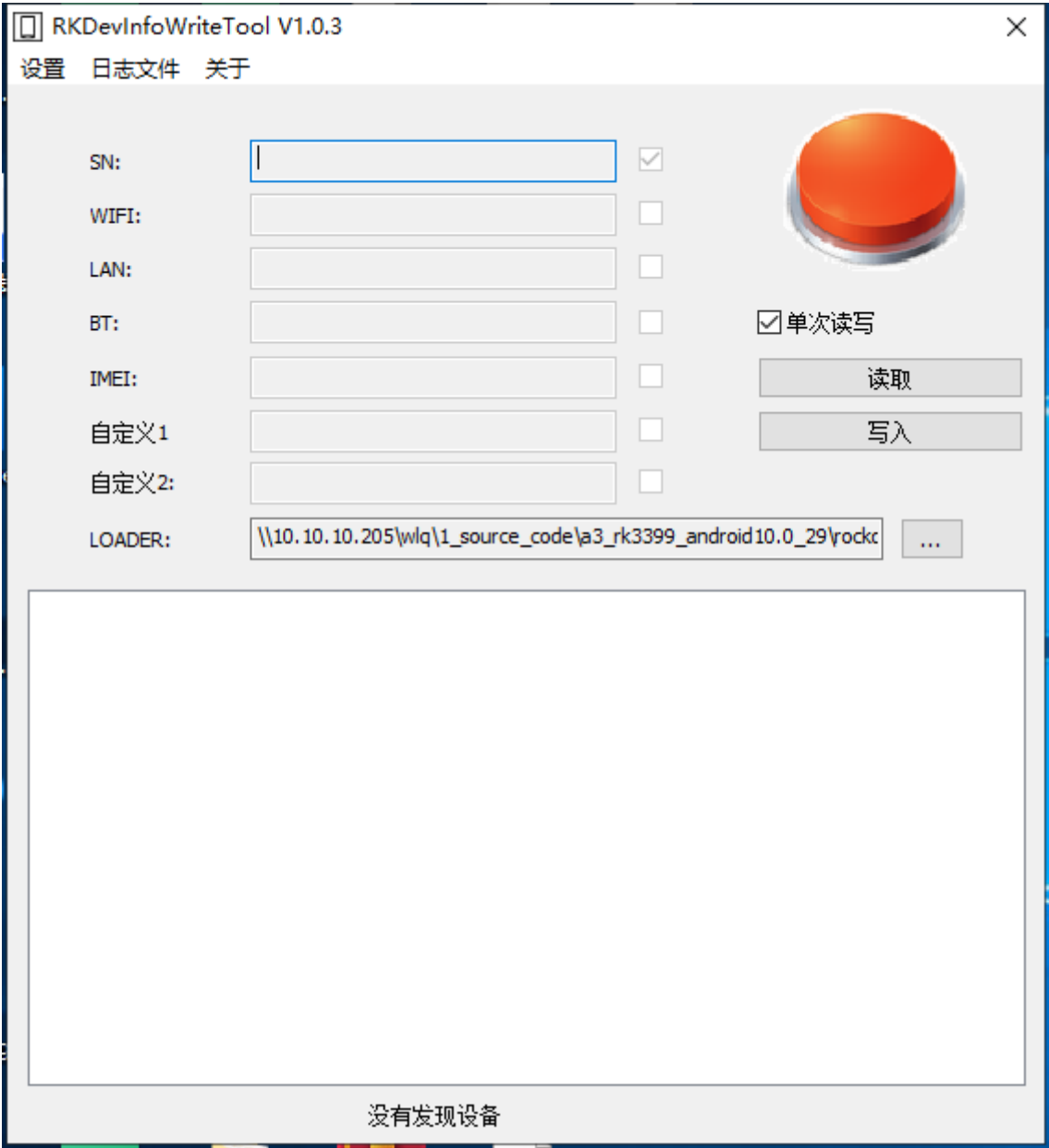
# SD升级启动制作工具

用于制作SD卡升级、SD卡启动、SD卡PCBA测试

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

# 写号工具

RKTools\windows\RKDevInfoWriteTool\_Setup\_V1.0.3.rar  
解压RKDevInfoWriteTool\_Setup\_V1.0.3.rar后安装  
以管理员权限打开软件



工具说明请参考：

```
RKDocs\common\RKTools manuals\RKDevInfowriteTool_User_Guide_V1.0.3.pdf
```

# DDR焊接测试工具

用于测试DDR的硬件连接，排查虚焊等硬件问题

```
RKTools\windows\Rockchip_Platform_DDR_Test_Tool_v1.38_Release_Announcement_CN.7z
RKTools\windows\Rockchip_Platform_DDR_Test_Tool_v1.38_Release_Announcement_EN.7z
```

## efuse烧写工具

用于efuse的烧写，适用于RK3288W/RK3368/RK3399平台

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

## efuse/otp签名工具

用于固件的efuse/otp签名

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

## 工厂生产固件烧写工具

用于工厂批量烧写固件

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

## 固件修改工具

用于修改update.img固件

```
RKTools\windows\FWFactoryTool_v5.52.rar
```

## userdata分区数据预置工具

用于制作userdata分区预置数据包的工具

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

# 系统调试

## ADB工具

### 概述

ADB (Android Debug Bridge) 是 Android SDK里的一个工具，用这个工具可以操作管理 Android 模拟器或真实的 Android 设备。主要功能有：

- 运行设备的 shell (命令行)
- 管理模拟器或设备的端口映射
- 计算机和设备之间上传/下载文件

- 将本地 apk 软件安装至模拟器或 Android 设备  
ADB 是一个“客户端 - 服务器端”程序，其中客户端主要是指 PC，服务器端是 Android 设备的实体机器或者虚拟机。根据 PC 连接 Android 设备的方式不同，ADB 可以分为两类：
- 网络 ADB：主机通过有线/无线网络（同一局域网）连接到 STB 设备
- USB ADB：主机通过 USB 线连接到 STB 设备

## USB adb使用说明

USB adb 使用有以下限制：

- 只支持 USB OTG 口
  - 不支持多个客户端同时使用（如 cmd 窗口，eclipse 等）
  - 只支持主机连接一个设备，不支持连接多个设备
- 连接步骤如下：
- 1、Android 设备已经运行 Android 系统，设置->开发者选项->已连接到计算机 打开，usb 调试开关打开。
  - 2、PC 主机只通过 USB 线连接到机器 USB otg 口，然后电脑通过如下命令与 Android 设备相连。

```
adb shell
```

3、测试是否连接成功，运行“adb devices”命令，如果显示机器的序列号，表示连接成功。

## 网络adb使用要求

adb 早期版本只能通过 USB 来对设备调试，从 adb v1.0.25 开始，增加了对通过 tcp/ip 调试 Android 设备的功能。

如果你需要使用网络 adb 来调试设备，必须要满足如下条件：

- 1、设备上面首先要有网口，或者通过 WiFi 连接网络。
- 2、设备和研发机（PC 机）已经接入局域网，并且设备设有局域网的 IP 地址。
- 3、要确保研发机和设备能够相互 ping 得通。
- 4、研发机已经安装了 adb。
- 5、确保 Android 设备中 adbd 进程（adb 的后台进程）已经运行。adbd 进程将会监听端口 5555 来进行 adb 连接调试。

## SDK网络adb端口配置

SDK 默认未开启网络 adb，需要手动在开发者选项中打开。

Setting-System-Advanced-Developer options-Open net adb

## 网络adb使用

本节假设设备的 ip 为 192.168.1.5，下文将会用这个 ip 建立 adb 连接，并调试设备。

- 1、首先 Android 设备需要先启动，如果可以的话，可以确认 adbd 是否启动（ps 命令查看）。
- 2、在 PC 机的 cmd 中，输入：

```
adb connect 192.168.1.5:5555
```

如果连接成功会进行相关的提示，如果失败的话，可以先 kill-server 命令，然后重试连接。

```
adb kill-server
```

3、如果连接已经建立，在研发机中，可以输入adb相关的命令进行调试了。比如adb shell，将会通过tcp/ip连接设备上面。和USB调试是一样的。

4、调试完成之后，在研发机上面输入如下的命令断开连接：

```
adb disconnect 192.168.1.5:5555
```

## 手动修改网络adb端口号

若SDK未加入adb端口号配置，或是想修改adb端口号，可通过如下方式修改：

1、首先还是正常地通过USB连接目标机，在windows cmd下执行adb shell进入。

2、设置adb监听端口：

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

3、通过ps命令查找adbd的pid

4、重启adbd

#kill -9，这个pid就是上一步找到那个pid

杀死adbd之后，android的init进程后自动重启adbd。adbd重启后，发现设置了service.adb.tcp.port，就会自动改为监听网络请求。

## ADB常用命令详解

(1) 查看设备情况

查看连接到计算机的 Android 设备或者模拟器：

```
adb devices
```

返回的结果为连接至开发机的 Android 设备的序列号或是IP和端口号（Port）、状态。

(2) 安装apk

将指定的 apk 文件安装到设备上：

```
adb install <apk文件路径>
```

示例如下：

```
adb install "F:\wishTV\wishTV.apk"
```

重新安装应用：

```
adb install -r "F:\wishTV\wishTV.apk"
```

(3) 卸载apk

完全卸载：

```
adb uninstall <package>
```

示例如下：

```
adb uninstall com.wishtv
```

(4) 使用 rm 移除 apk 文件：

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

示例如下：

```
adb shell rm "system/app/wishTV.apk"
```

示例说明：移除“system/app”目录下的“WishTV.apk”文件。

(5) 进入设备和模拟器的shell

进入设备或模拟器的 shell 环境：

```
adb shell
```

(6) 从电脑上传文件到设备

用 push 命令可以把本机电脑上的任意文件或者文件夹上传到设备。本地路径一般指本机电脑；远程路径一般指 adb 连接的单板设备。

adb push <本地路径> <远程路径>

示例如下：

```
adb push "F:\wishTV\wishTV.apk" "system/app"
```

示例说明：将本地“WishTV.apk”文件上传到 Android 系统的“system/app”目录下。

(7) 从设备下载文件到电脑

pull 命令可以把设备上的文件或者文件夹下载到本机电脑中。

```
adb pull <远程路径> <本地路径>
```

示例如下：

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

示例说明：将 Android 系统“system/app”目录下的文件或文件夹下载到本地“F:\”目录下。

(8) 查看 bug 报告

需要查看系统生成的所有错误消息报告，可以运行 adb bugreport 指令来实现，该指令会将 Android 系统的 dumpsys、dumpstate 与 logcat 信息都显示出来。

(9) 查看设备的系统信息

在 adb shell 下查看设备系统信息的具体命令。

```
adb shell getprop
```

## Logcat 工具

Android 日志系统提供了记录和查看系统调试信息的功能。日志都是从各种软件和一些系统的缓冲区中记录下来的，缓冲区可以通过 Logcat 来查看和使用。Logcat 是调试程序用的最多的功能。该功能主要是通过打印日志来显示程序的运行情况。由于要打印的日志量非常大，需要对其进行过滤等操作。

## Logcat 命令使用

用 logcat 命令来查看系统日志缓冲区的内容：

基本格式：

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

示例如下：

```
adb shell
logcat
```

## 常用的日志过滤方式

控制日志输出的几种方式：

- 控制日志输出优先级

示例如下：

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

示例说明：显示优先级为 warning 或更高的日志信息。

- 控制日志标签和输出优先级

示例如下：

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

示例说明：支持所有的日志信息，除了那些标签为“ActivityManager”和优先级为“Info”以上的、标签为“MyApp”和优先级为“Debug”以上的。

- 只输出特定标签的日志

示例如下：

```
adb shell
logcat wishTV:* *:S
```

或者

```
adb shell
logcat -s wishTV
```

示例说明：只输出标签为 WishTV 的日志。

- 只输出指定优先级和标签的日志

示例如下：

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

示例说明：只输出优先级为 I，标签为 WishTV 的日志。

## Procrank工具

---



Procrank 是 Android 自带的一款调试工具，运行在设备侧的 shell 环境下，用来输出进程的内存快照，便于有效的观察进程的内存占用情况。

包括如下内存信息：

- VSS: Virtual Set Size 虚拟耗用内存大小（包含共享库占用的内存）
- RSS: Resident Set Size 实际使用物理内存大小（包含共享库占用的内存）
- PSS: Proportional Set Size 实际使用的物理内存大小（比例分配共享库占用的内存）
- USS: Unique Set Size 进程独自占用的物理内存大小（不包含共享库占用的内存）

注意：

- USS 大小代表只属于本进程正在使用的内存大小，进程被杀死后会被完整回收；
- VSS/RSS 包含了共享库使用的内存，对查看单一进程内存状态没有参考价值；
- PSS 是按照比例将共享内存分割后，某单一进程对共享内存区的占用情况。

## 使用procrank

执行procrank前需要先让终端获取到root权限

su

命令格式：

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

常用指令说明：

- v: 按照 VSS 排序
- r: 按照 RSS 排序
- p: 按照 PSS 排序
- u: 按照 USS 排序
- R: 转换为递增[递减]方式排序
- w: 只显示 working set 的统计计数
- W: 重置 working set 的统计计数
- h: 帮助

示例：

输出内存快照：

```
procrank
```

按照 VSS 降序排列输出内存快照：

```
procrank -v
```

默认procrank输出是通过PSS排序。

## 检索指定内容信息

查看指定进程的内存占用状态，命令格式如下：

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

其中 cmdline 表示需要查找的应用程序名，PID 表示需要查找的应用进程。

输出 systemUI 进程的内存占用状态：

```
procrank | grep "com.android.systemui"
```

或者：

```
procrank | grep 3396
```

## 跟踪进程内存状态

通过跟踪内存的占用状态，进而分析进程中是否存在内存泄露场景。使用编写脚本的方式，连续输出进程的内存快照，通过对比 USS 段，可以了解到此进程是否有内存泄露。

示例：输出进程名为 com.android.systemui 的应用内存占用状态，查看是否有泄露：

1、编写脚本 test.sh

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

2、通过 adb 工具连接到设备后，运行此脚本：./test.sh

## Dumpsys工具

Dumpsys 工具是 Android系统中自带的一款调试工具，运行在设备侧的 shell 环境下，提供系统中正在运行的服务状态信息功能。正在运行的服务是指 Android binder机制中的服务端进程。

dumpsys 输出打印的条件：

- 1、只能打印已经加载到 ServiceManager中的服务；
- 2、如果服务端代码中的 dump 函数没有被实现，则没有信息输出。

## 使用Dumpsys

- 查看Dumpsys帮助  
作用：输出dumpsys帮助信息。

```
dumpsys -help
```

- 查看Dumpsys包含服务列表  
作用：输出dumpsys所有可打印服务信息，开发者可以关注需要调试服务的名称。

```
dumpsys -l
```

- 输出指定服务的信息  
作用：输出指定的服务的 dump 信息。  
格式：dumpsys [servicename]  
示例：输出服务 SurfaceFlinger的信息，可执行命令：

```
dumpsys SurfaceFlinger
```

- 输出指定服务和应有进程的信息  
作用：输出指定服务指定应用进程信息。  
格式：dumpsys [servicename] [应用名]  
示例：输出服务名为 meminfo，进程名为 com.android.systemui 的内存信息，执行命令：

```
dumpsys meminfo com.android.systemui
```

注意：服务名称是大小写敏感的，并且必须输入完整服务名称。

## Last log 开启

- 在dts文件里面添加下面两个节点

```
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>;
};
```

- 在机器中查看last log

```
130|root@rk3399:/sys/fs/pstore # ls
```

dmesg-ramoops-0 上次内核panic后保存的log。

pmsg-ramoops-0 上次用户空间的log，android的log。

ftrace-ramoops-0 打印某个时间段内的function trace。

console-ramoops-0 last\_log 上次启动的kernel log，但只保存了优先级比默认log level 高的log。

- 使用方法：

```
cat dmesg-ramoops-0
cat console-ramoops-0
logcat -L (pmsg-ramoops-0) 通过logcat 取出来并解析pull out by logcat and parse
cat ftrace-ramoops-0
```

## FIQ模式

当设备死机或者卡住的时候可以在串口输入fiq命令查看系统的状态，具体命令如下：

```
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        Interrupt 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自动收集

- 收集的内容

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

- 打开方式
- 开启Developer options
- Setting-System-Advanced-Developer options-Android bug collector
- log保存路径

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

## 常见问题

### 当前kernel和u-boot版本?

Android11.0 对应的kernel版本为: develop-4.19, u-boot的分支为next-dev分支

### 如何获取当前SDK对应的RK release版本

Rockchip Android11.0 SDK包括AOSP原始代码和RK修改的代码两部分, 其中RK修改的仓库包含在 .repo/manifests/include 目录下面的xml中, AOSP默认的仓库在 .repo/manifests/default.xml。

版本确认:

- RK修改部分

```
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">
```

说明RK的版本是android-11.0-mid-rkr1

- AOSP部分

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

说明AOSP的版本是android-10.0.0\_r14

当需要提供版本信息的时候提供以上两个版本信息即可。

单个仓库可以直接通过如下命令获取tag信息

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

RK的版本是以android-11.0-mid-rkrxx的格式递增的，所以当前的最新tag是android-11.0-mid-rkr1

## 如何确认本地SDK已经完整更新RK发布的SDK状态

RK发布SDK版本时会在.repo/manifests/commit/目录下对应提交该版本的commit信息，客户可以通过对比这个commit信息来确认是否有完整更新SDK，具体操作如下：

- 按“如何获取当前SDK对应的RK release版本”的说明先确认SDK的RK版本，下面以RK版本是RKR6为例进行说明；
- 用如下命令保存本地的commit信息

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

- 通过比较.repo/manifests/commit/commit\_release\_rkr6.xml和release\_manifest\_rkr6\_local.xml，即可确认SDK代码是否更新完整，其中.repo/manifests/commit/commit\_release\_rkr6.xml为RK版本RKR6发布的commit信息。

## uboot和kernel阶段logo图片替换

uboot和kernel阶段的logo分别为开机显示的第一张和第二张logo图片，可以根据产品需求进行修改替换。

uboot logo源文件：kernel/logo.bmp

kernel logo源文件：kernel/logo\_kernel.bmp

如果需要更换某一张，只需用同名的bmp替换掉，重新编译内核即可，编译后的文件在boot.img中。

说明：Logo图片大小目前只支持到8M以内大小的bmp格式图片，支持8、16、24、32位的bmp。

## 关机充电和低电预充

关机充电和低电预充可以在dts中配置，具体如下：

```
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";
};
```

其中：

rockchip,uboot-charge-on：uboot关机充电，与android关机充电互斥

rockchip,android-charge-on：android关机充电，与uboot关机充电互斥

rockchip,uboot-low-power-voltage：配置低电预充到开机的电压，可以根据实际需求进行配置

rockchip,screen-on-voltage：配置低电预充到亮屏的电压，可以根据实际需求进行配置

## Uboot阶段充电图片打包和替换

充电图片路径，可以直接替换同名文件，格式要求与原文件一样。

```
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
```

如果打开uboot充电，但是没有显示充电图片，可能是图片没有打包到resource.img中，可以按如下命令打包

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

执行以上命令后uboot充电图片会打包到kernel目录的resource.img中，此时需要再将resource.img打包到boot.img中，可以在android根目录执行./mkimage.sh，然后烧写rockdev/下面的boot.img即可。

## RM310 4G配置

4G功能SDK默认是关闭的，如需打开，按以下操作：

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

## Recovery旋转配置

支持Recovery旋转0/90/180/270度，默认不旋转（即旋转0度），旋转配置说明如下：

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

## Android Surface旋转

Android系统显示旋转，可以修改如下配置，配置参数为0/90/180/270

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

## 替换 AOSP 部分源代码的 remote

客户下载RK的release代码时速度较慢，可以将AOSP的remote修改为国内镜像源，国外的客户可以修改为Google的镜像源。这样可以提高下载速度。具体操作方法如下：

执行repo init（或者解压base包）后，修改.repo/manifests/remote.xml，把其中的 AOSP 这个remote的fetch从

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

改为

国内客户：（国内以清华大学镜像源为例，可以根据需要修改为其他国内镜像源）

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

国外的客户：（Google镜像源）

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

## userdata区文件系统换为EXT4

默认data分区的文件系统为f2fs，建议不带电池的产品可以将data区的文件系统改为ext4，可以减小异常掉电后数据丢失的概率。修改方法如下：

以rk3566\_r为例说明：

```
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

```

## 修改开关机动画和开关机铃声

参考文档：

RKDocs\android\Rockchip\_Introduction\_Android\_Power\_On\_Off\_Animation\_and\_Tone\_Customization\_CN&EN.pdf

## APP设置性能模式

device/rockchip/rk3xxx/下配置文件：package\_performance.xml，在其中的节点中加入需要使用性能模式的包名：（使用 `aapt dump badging (file_path.apk)` 获取包名）

```
< app package="包名" mode="是否启用加速，启用为 1，关闭为 0"/>
```

例如针对安兔兔的参考如下：

```

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

```

编译时会将文件打包进固件。

## GPU相关问题排查方法

参考下面文档，可以做初步的问题排查

RKDocs\android\Rockchip\_User\_Guide\_Dr.G\_CN&EN.pdf



# OTP和efuse说明

## OTP支持芯片

- RK3326
- PX30
- RK3566
- RK3568

## EFUSE支持芯片

- RK3288
- RK3368
- RK3399

固件签名和otp/efuse烧写参考文档

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

## 代码中如何判断设备的OTP/EFUSE是否已经烧写

OTP/EFUSE的状态会通过kernel的cmdline进行传递，cmdline中的fuse.programmed用来标识OTP/EFUSE状态，具体如下：

- "fuse.programmed=1"：软件固件包已经进行了secure-boot签名，硬件设备的efuse/otp已经被烧写。
- "fuse.programmed=0"：软件固件包已经进行了secure-boot签名，硬件设备的efuse/otp没有被烧写。
- cmdline中没有fuse.programmed：软件固件包没有进行secure-boot签名（Miniloader不传递），或者Miniloader太旧没有支持传递。

## 开关selinux

如下修改，false为关闭，true为打开

```
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
```

## 开机弹出“Android系统出现问题”警告

出现警告框的原因有两种：

1. 固件不匹配，system/boot/vendor三个fingerprint不一致，不是同一套固件。
2. 机器打开支持了IO调试功能的config，编译时，使用文档前面所说的内核编译命令即可关闭。
3. 对于需要使用IO调试功能的项目，可以直接不管上述两种原因，直接合入frameworks/base下的patch去掉弹窗：

```
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");

            mHandler.post(() -> {
                if (mShowDialogs) {
```

## 如何打开设置中以太网的设置项

系统设置中默认没有以太网设置的选项，如果项目中需要以太网可以按如下配置打开：

```
--- 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
```

## 关于AVB

AVB的相关说明和配置可以参考文档

[RKDocs/common/u-boot/Rockchip\\_Developer\\_Guide\\_UBoot\\_Nextdev\\_CN.pdf](#)

## IO命令无法使用

IO命令需要依赖DEVMEM，而DEVMEM默认是关闭的，所以导致IO默认无法使用，如果调试需要使用IO命令可以按如下修改：

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

如果是GO的产品则需要修改：

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

删除掉下面这行：

```
# CONFIG_DEVMEM is not set
```

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

删掉下面这行：

```
# CONFIG_DEVMEM is not set
```

以上两个需要同时删掉

## SN号的命令规则

SN号必须以字母开头，长度14个字节以内。

# 附录 A 编译开发环境搭建 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](http://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 java
sudo 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
```

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>

## 附录 B SSH公钥操作说明 SSH public key operation instruction

### 附录 B-1 SSH公钥生成 SSH public key generation

使用如下命令生成：

Use the following command to generate:

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

请将user@host替换成您的邮箱地址。

Please replace user@host with your email address.

命令运行完成会在你的目录下生成key文件。

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

请妥善保存生成的私钥文件id\_rsa和密码，并将id\_rsa.pub发邮件给SDK发布服务器的管理员。

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

### 附录 B-2 使用key-chain管理密钥 Use key-chain to manage the key

推荐您使用比较简易的工具keychain管理密钥。

Recommend you use the simple tool keychain to manage the key.

具体使用方法如下：

The detailed usage is as follows:

1. 安装keychain软件包：

Install keychain software package:

```
$sudo aptitude install keychain
```

2. 配置使用密钥：

Configure to use the key:

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

增加下面这行：

Add the following command:

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

其中，id\_rsa是私钥文件名称。

Among which, id\_rsa is the file name of the private key.

以上配置以后，重新登录控制台，会提示输入密码，只需输入生成密钥时使用的密码即可，若无密码可不输入。

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.

另外，请尽量不要使用sudo或root用户，除非您知道如何处理，否则将导致权限以及密钥管理混乱。

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.

## 附录 B-3 多台机器使用相同ssh公钥 Multiple devices use the same ssh public key

在不同机器使用，可以将你的ssh私钥文件id\_rsa拷贝到要使用的机器的“~/.ssh/id\_rsa”即可。

In order to use on different devices, you can copy ssh private key file id\_rsa to the target device “~/.ssh/id\_rsa”.

在使用错误的私钥会出现如下提示，请注意替换成正确的私钥。

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

添加正确的私钥后，就可以使用git 克隆代码，如下图。

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

添加ssh私钥可能出现如下提示错误。

Below error may occur when adding ssh private key:

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

在console输入如下命令即可解决。

Input the following command at console can fix it.

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

## 附录 B-4 一台机器切换不同ssh公钥 Switch different ssh public keys on one device

---

可以参考ssh\_config文档配置ssh。

You can refer to ssh\_config document to configure ssh.

```
~$ man ssh_config
```

通过如下命令，配置当前用户的ssh配置。

Use the following commands to configure ssh for current user.

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

如图，将ssh使用另一个目录的文件“~/.ssh1/id\_rsa”作为认证私钥。通过这种方法，可以切换不同的密钥。

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

## 附录 B-5 密钥权限管理 Key authority management

---

服务器可以实时监控某个key的下载次数、IP等信息，如果发现异常将禁用相应的key的下载权限。

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.

请妥善保管私钥文件。并不要二次授权与第三方使用。

Please keep carefully the private key file. DO NOT re-authorize it to the third party.

## 附录 B-6 Git权限申请说明 Git authority application instruction

---

参考上述章节，生成公钥文件，发邮件至 [fae@rock-chips.com](mailto:fae@rock-chips.com)，申请开通SDK代码下载权限。

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