Rockchip
瑞芯微电子

Rockchip

RK3229 软件开发指南

发布版本:2.0

日期:2019.07

前言

概述

文档作为 Rockchip RK3229 软件开发指南，旨在帮助软件开发工程师和技术支持工程师更快上手 RK3229 的开发及调试。

产品版本

芯片名称	内核版本	**Android** 版本
RK3229	Linux4.4	Android9.0.0

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

修订记录

日期	版本	作者	审核	修改说明
2019-01-25	V1.00	XYP	CW，HuangJC	创建初始发布版本
2019-07-19	V2.00	XYP	CW，HuangJC	更新文档、工具索引

目录

插图目录

表格目录

1 支持列表

1.1 DDR 支持列表

RK3229 DDR 支持 DDR3、DDR3L、LPDDR2、LPDDR3。

表 1-1 RK3229 DRAM Support Type

Chip	DRAM Support Type
RK3229	DDR3/DDR3L/LPDDR2/LPDDR3

RK3229 DDR 颗粒支持程度列表，详见 RKDocs\common\Platform support lists 目录下《RK DDR Support List Ver2.38.pdf》，下表中所标示的 DDR 支持程度表，只建议选用 √、T/A 标示的颗粒。

表 1-2 RK3229 DDR Support Symbol

Symbol	Description
√	Fully Tested and Mass production
T/A	Fully Tested and Applicable
N/A	Not Applicable

1.2 NAND 支持列表

RK3229 Nand Flash 支持程度列表，详见 RKDocs\common\Platform support lists 目录下《RKNandFlashSupportList Ver2.73_20180615.pdf》，下表中所标示的 NAND 支持程度表，建议选用 √、 T/A 标示的颗粒。

如有选型上的疑问，也可直接联系 Rockchip FAE 窗口。

表 1-3 RK3229 NAND Support Symbol

Symbol	Description
√	Fully Tested , Applicable and Mass Production
T/A	Fully Tested , Applicable and Ready for Mass Production
D/A	Datasheet Applicable,Need Sample to Test
N/A	Not Applicable

1.3 EMMC 支持列表

RK3229 支持 eMMC 4.51，SDIO3.0，支持 HS200 模式，详见 RKDocs\common\Platform support lists 目录下《RKeMMCSupportList Ver1.43_2019_03_15.pdf》，下表中所标示的 EMMC 支持程度表，只建议选用 √、T/A 标示的颗粒。

表 1-4 RK3229 EMMC Support Symbol

Symbol	Description
√	Fully Tested , Applicable and Mass Production
T/A	Fully Tested , Applicable and Ready for Mass Production
D/A	Datasheet Applicable,Need Sample to Test
N/A	Not Applicable

1.3.1 高性能 EMMC 颗粒的选取

为了提高系统性能，需要选取高性能的 EMMC 颗粒。请在挑选 EMMC 颗粒前，参照 Rockchip 提供支持列表中的型号，重点关注下厂商 Datasheet 中 performance 一章节。

参照厂商大小以及 EMMC 颗粒读写的速率进行筛选。建议选取顺序读速率>200MB/s、顺序写速率>40MB/s。

如有选型上的疑问，也可直接联系 Rockchip Fae 窗口。

6.1.5 Performance

[Table 23] Performance

Density	Partition Type	Performance	
		Read(MB/s)	Write (MB/s)
16GB	General	285	40
32GB		310	70
64GB		310	140
128GB		310	140
16GB	Enhanced	295	80
32GB		320	150
64GB		320	245
128GB		320	245

图 1-1 EMMC Performance 示例

1.4 WiFi/BT 支持列表

RK3229 Android9.0 的 Kernel 版本为 Linux4.4，WiFi/BT 支持列表，详见 RKDocs\common\Platform support lists 目录下《Rockchip_Introduction_WiFi_Situation_CN.pdf》 ，文档列表中为目前 RK3229 上大量测试过的 Wifi/Bt 芯片列表，建议按照列表上的型号进行选型。如果有其他 WiFi/BT 芯片调试，需要 WiFI/BT 芯片原厂提供 Linux4.4 版本的内核驱动程序。

如果疑问和建议可以与 Rockchip Fae 窗口联系。

1.5 SDK 软件包适用硬件列表

本 SDK 是基于谷歌 Android9.0 32bit 系统，适配瑞芯微 RK3229 芯片的软件包，适用于 RK3229 Box Evb 开发板及基于其上所有的开发产品。

若是基于 RK3229 Box Evb 开发板开发，内核配置可参考 rk3229-evb-android-avb.dts 进行改动。

另 SDK 中附带了 RK3229 Box Evb 开发板的硬件使用说明。

表 1-3 RK3229 硬件说明列表

硬件板	对应文档说明
Evb 开发板	RKDocs\rk322x\ Rockchip_RK322X_Hardware_Design_Guide_V1.0_CN.pdf

1.6 多媒体编解码支持列表

RK3229 多媒体规格，详见表 1-5:

表 1-4 RK3229 多媒体编解码支持列表

多媒体支持	支持 4K VP9 and 4K 8bits H264/H265 视频解码，高达 60fps。
	1080P 多格式视频解码 (VC-1, MPEG-1/2/4, VP8)。支持 JPEG 解码。
	1080P H.264 格式视频编码。支持 JPEG 编码。

RK3229 具体的编解码支持列表，详见 RKDocs\rk322x 目录下
《Rockchip_RK3229_Multimedia_Codec_Benchmark_EN.pdf》

2 文档/工具索引

2.1 文档索引

随 RK3229 Box SDK 发布的文档旨在帮助开发者快速上手开发及调试，文档中涉及的内容并不能涵盖所有的开发知识和问题。文档列表也正在不断更新，如有文档上的疑问及需求，请联系我们的 Fae 窗口。

RK3229 SDK 的 RKDocs 目录结构如下所示。

```
RKDocs/
├── android
│   ├── bt
│   │   ├── Rockchip_Introduction_Android8.1_BT_Configuration_CN.pdf
│   │   └── Rockchip_Introduction_Android9.0_BT_Configuration_CN.pdf
│   ├── project.config
│   ├── Rockchip_Developer_Guide_Android_New_Partition_Configuration_CN.pdf
│   ├── Rockchip_Developer_Guide_PCBA_Test_Tool_CN&EN.pdf
│   ├── Rockchip_Developer_Guide_PCBA_Test_Tool_CN.pdf
│   ├── Rockchip_Introduction_Android8.0_Factory_Reset_Protection_CN.pdf
│   ├── Rockchip_Introduction_Android8.0_Power_On_Off_Animation_and_Tone_Customization_CN.pdf
│   ├── Rockchip_Introduction_Android8.1_BOX_Display_Framework_Configuration_CN.pdf
│   ├── Rockchip_Introduction_Android9.0_AVB_Howto_CN.pdf
│   ├── Rockchip_Introduction_Android9.0_Safety_Boot_Solution_CN.pdf
│   ├── Rockchip_Introduction_Android9.0_System_New_Feature_CN&EN.pdf
│   ├── Rockchip_Introduction_Android_AB_System_Upgrading_CN.pdf
│   ├── Rockchip_Introduction_Android_AB_System_Upgrading_EN.pdf
│   ├── Rockchip_Introduction_Android_Application_Preinstallation_CN.pdf
│   ├── Rockchip_Introduction_Android_Performance_Mode_CN.pdf
│   ├── Rockchip_Introduction_Android_Verify_Boot_CN.pdf
│   ├── Rockchip_Introduction_Android_Widevine_Project_Start_Preparation_CN.pdf
│   ├── Rockchip_Introduction_Box_Media_Application_CN.pdf
│   ├── Rockchip_Introduction_PCBA_Camera_Porting_CN.pdf
│   ├── Rockchip_User_Guide_Magisk_Installation_EN.pdf
│   ├── Rockchip_User_Guide_Recovery_CN&EN.pdf
│   └── wifi
│       ├── Rockchip_Introduction_Android9.0_WIFI_Configuration_CN.pdf
│       └── Rockchip_Introduction_RealTek_WIFI_Driver_Porting_CN.pdf
├── 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
    |   ├── camera
    |   |   ├── HAL1
    |   |   |   ├── Camera_Document_Directory.txt
    |   |   |   ├── CIF_ISP10_Driver_User_Manual_V1.0_20171124.pdf
    |   |   |   ├── CIF_ISP11_Driver_User_Manual_V1.0.pdf
    |   |   |   ├── readme_En.txt
    |   |   |   ├── RK312x_Camera_User_Manual_v1.4(3288&3368).pdf
    |   |   |   ├── RK_ISP10_Camera_User_Manual_v2.2.pdf
    |   |   |   ├── RKISPV1_Camera_Module_AVL_v1.7.pdf
    |   |   |   ├── Rockchip_Camera_AVL_v2.0_Package_20180515.7z
    |   |   |   ├──
Rockchip_Introduction_RKISPV1_Camera_Driver_Debugging_Method_CN.pdf
    |   |   |   ├── Rockchip_Introduction_RKISPV1_Camera_FAQ_CN.pdf
    |   |   |   └── Rockchip SOFIA 3G-
R_PMB8018(x3_C3230RK)_Camera_Module_AVL_v1.6_20160226.pdf
    |   |   ├── HAL3
    |   |   |   ├── camera_engine_rkisp_user_manual_v2.0.pdf
    |   |   |   ├── camera_hal3_user_manual_v2.1.pdf
    |   |   |   ├── RKCIF_Driver_User_Manual_v1.0.pdf
    |   |   |   └── RKISP_Driver_User_Manual_v1.2.pdf
    |   |   └── README.txt
    |   ├── CRU
    |   |   └── Rockchip-Clock-Developer-Guide-RTOS-CN.pdf
    |   ├── 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
    |   ├── debug
    |   |   ├── RK3399-LOG-EXPLANATION.pdf
    |   |   ├── Rockchip_Quick_Start_Linux_Perf.pdf
    |   |   ├── Rockchip_Quick_Start_Linux_Streamline.pdf
    |   |   └── Rockchip_Quick_Start_Linux_Systrace.pdf
    |   ├── display
    |   |   ├── Rockchip_Developer_Guide_DRM_Panel_Porting_CN.pdf
    |   |   ├──
Rockchip_Developer_Guide_Dual_Display_Rotation_Direction_Debugging_CN.pdf
    |   |   ├──
Rockchip_Developer_Guide_HDMI_Based_on_DRM_Framework_CN.pdf
    |   |   ├── Rockchip_Introduction_Baseparameter_Storage_Format_CN.pdf
    |   |   ├── Rockchip_Introduction_DRM_Integration_Helper_CN.pdf
    |   |   └── Rockchip_User_Guide_Android_Display_Based_on_DRM_CN.pdf
    |   ├── DVFS
    |   |   ├── Rockchip-Developer-Guide-Linux4.4-CPUFreq-CN.pdf
```

```
    |   |   └── Rockchip-Developer-Guide-Linux4.4-Devfreq.pdf
    |   ├── GMAC
    |   |   └── Rockchip_Developer_Guide_Ethernet_CN.pdf
    |   ├── hdmi-in
    |   |   └── Rockchip_Developer_Guide_HDMI_IN_CN.pdf
    |   ├── I2C
    |   |   └── Rockchip-Developer-Guide-Linux-I2C.pdf
    |   ├── IO-Domain
    |   |   └── Rockchip-Developer-Guide-Linux-IO-DOMAIN-CN.pdf
    |   ├── Leds
    |   |   └── Rockchip_Introduction_Leds_GPIO_Configuration_for_Linux4.4_CN.pdf
    |   ├── MCU
    |   |   ├── Rockchip-Developer-Guide-linux4.4-MCU.pdf
    |   |   └── Rockchip-Developer-Guide-MCU-EN.pdf
    |   ├── MMC
    |   |   └── Rockchip-Developer-Guide-linux4.4-SDMMC-SDIO-eMMC.pdf
    |   ├── mobile-net
    |   |   ├── Rockchip_Introduction_3G_Data_Card_USB_File_Conversion_CN.pdf
    |   |   └── Rockchip_Introduction_3G_Dongle_Configuration_CN.pdf
    |   ├── other
    |   |   ├── RK3399-CPUINFO.pdf
    |   |   ├── RK3399-LOG-EXPLANATION.pdf
    |   |   └── Rockchip_Introduction_Browser_FAQ_CN.pdf
    |   ├── PCie
    |   |   └── Rockchip-Developer-Guide-linux4.4-PCIe.pdf
    |   ├── PIN-Ctrl
    |   |   └── Rockchip-Developer-Guide-Linux-Pin-Ctrl-CN.pdf
    |   ├── Platform support lists
    |   |   ├── RK3128 BOX   Hardware Design Guide V10-201410.pdf
    |   |   ├── RK DDR Support List Ver2.38.pdf
    |   |   ├── RKeMMCSupportList Ver1.43_2019_03_15.pdf
    |   |   ├── RKNandFlashSupportList Ver2.73_20180615.pdf
    |   |   ├── Rockchip_Camera_AVL_v2.0_Package.7z
    |   |   ├── Rockchip_Introduction_WiFi_Situation_CN.pdf
    |   |   └── Rockchip_Kodi_Support_List_CN.pdf
    |   ├── PMIC
    |   |   ├── Archive.zip
    |   |   ├── Rockchip-Developer-Guide-Power-Discrete-DCDC-Linux4.4.pdf
    |   |   ├── Rockchip-Developer-Guide-RK805.pdf
    |   |   ├── Rockchip_Developer_Guide_RK817_RK809_Fuel_Gauge_CN.pdf
    |   |   ├── Rockchip-Developer-Guide-RK818_6-Fuel-Gauge.pdf
    |   |   └── Rockchip-RK818-RK816-FG-Log-Description-linux4.4.pdf
    |   ├── power
    |   |   └── Rockchip_Developer_Guide_Sleep_and_Resume_CN.pdf
```

```
    |   ├── PWM
    |   |   ├── Rockchip-Developer-Guide-Linux-PWM-CN.pdf
    |   |   └── Rockchip_Developer_Guide_PWM_IR_CN.pdf
    |   ├── RKTools manuals
    |   |   ├── RKDevInfoWriteTool_User_Guide_V1.0.3.pdf
    |   |   ├── RKIQTool_User_Manual_v1.5-CH.pdf
    |   |   ├── RKIQTool_User_Manual_v1.5-EN.pdf
    |   |   ├── RK_Platform_apache_tomcat_ota_Server_Setup_Introduction.rar
    |   |   ├── Rockchip_Box_Factory_Test_Tool_V2.0.rar
    |   |   ├── Rockchip_Introduction_Image_Upgrading_Failure_Analysis_CN.pdf
    |   |   ├── Rockchip_Introduction_MP_Tool_Upgrading_and_Related_Issues_Debugging_CN.pdf
    |   |   ├── Rockchip_Introduction_Parameter_File_Format_CN.pdf
    |   |   ├── Rockchip_Introduction_REPO_Mirror_Server_Build_and_Management_CN.pdf
    |   |   ├── Rockchip_Introduction_Stresstest_for_VR_CN.pdf
    |   |   ├── Rockchip_Introduction_WNpctool_Write_Tool_CN.pdf
    |   |   ├── Rockchip_User_Guide_Box_Factory_Test_Tool_CN.pdf
    |   |   ├── Rockchip_User_Guide_Keybox_Burning_EN.pdf
    |   |   ├── Rockchip_User_Guide_KeyWrite_CN.pdf
    |   |   ├── Rockchip_User_Guide_MP_Flashing_CN.pdf
    |   |   ├── Rockchip_User_Guide_RKDevInfoWriteTool_CN.pdf
    |   |   ├── Rockchip_User_Guide_RKDevInfoWriteTool_EN.pdf
    |   |   ├── Rockchip_User_Guide_RK_Platform_MP_Upgrading_CN.pdf
    |   |   ├── Rockchip_User_Manual_Android_Development_Tool_CN.pdf
    |   |   ├── Rockchip_User_Manual_RKIQTool_CN.pdf
    |   |   ├── Rockchip_User_Manual_RKIQTool_EN.pdf
    |   |   └── Rockchip_User_Manual_RKUpgrade_Dll_CN.pdf
    |   ├── security
    |   |   ├── Efuse process explain .pdf
    |   |   ├── RK3399_Efuse_Operation_Instructions_V1.00_20190214_EN.pdf
    |   |   ├── Rockchip_Developer_Guide_TEE_Secure_SDK_CN.pdf
    |   |   ├── Rockchip_RK3399_Introduction_Efuse_Operation_EN.pdf
    |   |   ├── Rockchip-Secure-Boot-Application-Note-V1.9.pdf
    |   |   └── Rockchip Vendor Storage Application Note.pdf
    |   ├── Sensors
    |   |   └── Rockchip_Developer_Guide_Sensors_CN.pdf
    |   ├── SPI
    |   |   └── Rockchip-Developer-Guide-linux4.4-SPI.pdf
    |   ├── Thermal
    |   |   ├── Rockchip-Developer-Guide-Linux4.4-Thermal-CN.pdf
    |   |   └── Rockchip-Developer-Guide-Linux4.4-Thermal-EN.pdf
    |   ├── TRUST
    |   |   ├── Rockchip-Developer-Guide-RK3308-System-Suspend.pdf
    |   |   └── Rockchip-Developer-Guide-Trust.pdf
```

```
|   ├── UART
|   |   ├── Rockchip-Developer-Guide-linux4.4-UART.pdf
|   |   └── Rockchip-Developer-Guide-RT-Thread-UART.pdf
|   ├── u-boot
|   |   ├── Rockchip-Developer-Guide-Linux-AB-System.pdf
|   |   ├── Rockchip-Developer-Guide-Trust.pdf
|   |   ├── Rockchip-Developer-Guide-Uboot-mmc-device-driver-analysis.pdf
|   |   └── Rockchip-Developer-Guide-UBoot-nextdev-CN.pdf
|   └── 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
└── rk322x
    ├──Rockchip_RK3229_Android9.0_Box_SDK_Release_V1.0.0_20190125
_CN.pdf
    ├── Rockchip_RK3229_Developer_Guide_Android9.0_Box_CN.pdf
    ├── Rockchip_RK3229_Multimedia_Codec_Benchmark_EN.pdf
    └── Rockchip_RK322X_Hardware_Design_Guide_V1.0_CN.pdf
```

2.2 工具索引

随 RK3229 Box SDK 发布的工具，用于开发调试阶段及量产阶段。工具版本会随 SDK 更新不断更新，如有工具上的疑问及需求，请联系我们的 Fae 窗口。

RK3229 SDK 中在 RKTools 目录下附带了 linux（Linux 操作系统环境下使用工具）、windows（Windows 操作系统环境下使用工具）。

表 2-1 工具索引表格

工具名称	工具说明	工具路径
AndroidTool	分立升级固件及整个 update 升级固件工具	RKTools\windows\AndroidTool_Release_v2.65
FactoryTool	量产升级工具	RKTools\windows\FactoryTool_v1.64
SecureBootTool	固件签名工具	RKTools\windows\SecureBootTool_v1.88
efuseTool	efuse 烧写工具	RKTools\windows\efuse_v1.37
WNpctool	写号工具	RKTools\windows\WNpctool_Setup_V1.3.0_180813
SD_Firmware_Tool	SD 卡镜像制作	RKTools\windows\SD_Firmware_Tool._v1.46

SpiImageTools	烧录器升级工具	RKTools\windows\SpiImageTools_v1.36
DriverAssitant	驱动安装工具	RKTools\windows\DriverAssitant_v4.5
OemTool	新增分区镜像制作工具	RKTools\windows\OemTool_v1.3
Rockchip 平台 DDR 测试工具	DDR 测试工具	RKTools\windows\Rockchip 平台 DDR 测试工具_V1.35
Rockchip Box 厂测工具	厂测工具	RKTools\windows\Rockchip Box 厂测工具 V3.0-P-20181113
KeyWrite_v1.61	Widevine KEY 烧写工具	RKTools\windows\KeyWrite_v1.64.zip
Linux_Pack_Firmware	Linux 打包工具	RKTools\linux\Linux_Pack_Firmware
Linux_SecureBoot	Linux 签名工具	RKTools\linux\Linux_SecureBoot
Linux_Upgrade_Tool	Linux 烧写工具	RKTools\linux\Linux_Upgrade_Tool

3 SDK 编译/烧写

3.1 SDK 获取

SDK 通过瑞芯微代码服务器对外发布。客户向瑞芯微技术窗口申请 SDK，需同步提供 SSH 公钥进行服务器认证授权，获得授权后即可同步代码。关于瑞芯微代码服务器 SSH 公钥授权，请参考《Rockchip_RK3229_Android9.0_Box_SDK_Release_V1.0.0_20190125_CN.pdf》，该文档与 SDK 一同发布。

3.2 SDK 编译配置

3.2.1 分区大小配置

Sdk 默认分区表一般定义在对应产品目录下的 parameter.txt 文件中（例如：rk3229_box 对应 device/rockchip/rk322x/rk322x_box/parameter.txt）.

BOARD_SYSTEMIMAGE_PARTITION_SIZE 等系统分区宏配置会自动读取分区表中分区大小赋值。

若需要调整分区大小，修改对应分区表定义文件 parameter.txt 即可。具体参见 6.4 章节 parameter 说明。

注意：**Android Pie 要求新增部分分区及说明如下：**

dtb 分区：用来存放 device tree blob 的镜像，预留，默认不烧；

dtbo 分区：用来存放 device tree blob overlay 的镜像，默认烧对应产品目录下的预置镜像 dtbo.img；

vbmeta 分区：用来存放 Android 验证启动 (AVB)模式下编译自动生成的各分区校验数据，由于目前 sdk 已升级到支持 vboot2.0，默认烧写时必须烧 vbmeta.img。

3.2.2 Android Pie 新增特性配置说明

默认 Android9.0 要求的新增特性配置修改在 sdk 源码 device/rockchip/common/BoardConfig.mk 中：

1.要求默认使用 64bit binder 驱动

```
#binder protocol(8)
TARGET_USES_64_BIT_BINDER := true
```

2.默认 cmdline 参数配置在产品配置中修改，不在 parameter 中了，如常见的 selinux 权限配置等。

```
ifneq ($(filter true, $(BOARD_AVB_ENABLE)), )
BOARD_KERNEL_CMDLINE := console=ttyFIQ0 androidboot.baseband=N/A
androidboot.selinux=permissive androidboot.wificountrycode=US androidb
oot.hardware=rk30board androidboot.console=ttyFIQ0
firmware_class.path=/vendor/etc/firmware init=/init skip_initramfs rootwait ro init=
/init
else
#Config the cmdline for boot or recpvery
BOARD_KERNEL_CMDLINE := console=ttyFIQ0 androidboot.baseband=N/A
androidboot.selinux=permissive androidboot.wificountrycode=US androidb
oot.veritymode=enforcing androidboot.hardware=rk30board androidboot.console=ttyFIQ0
firmware_class.path=/vendor/etc/firmware init=/init
 skip_initramfs rootwait ro init=/init root=PARTUUID=af01642c-9b84-11e8-9b2a-
234eb5e198a0
```

```
endif
```

3.要求默认开启 system as root 功能，主要变动为将原 ramdisk.img 整合到 system.img 中，启动时 system 作为 rootfs 挂载，如下图为分区镜像差异。

```
BOARD_BUILD_SYSTEM_ROOT_IMAGE := true
```

<table>
<tr><th>Component</th><th>Image</th><th>ramdisk (before P)</th><th>system-as-root (after P)</th></tr>
<tr><td rowspan="3">Image Content</td><td>boot.img</td><td>Contains a kernel and a ramdisk.img:
<pre>ramdisk.img
-/
- init.rc
- init
- etc -> /system/etc
- system/ (mount point)
- vendor/ (mount point)
- odm/ (mount point)
...</pre></td><td>Contains a normal boot kernel only.</td></tr>
<tr><td>recovery.img</td><td colspan="2">Contains a recovery kernel and a recovery-ramdisk.img.</td></tr>
<tr><td>system.img</td><td>Contains the following:
<pre>system.img
-/
- bin/
- etc
- vendor -> /vendor
- ...</pre></td><td>Contains the merged content of original system.img and ramdisk.img:
<pre>system.img
-/
- init.rc
- init
- etc -> /system/etc
- system/
- bin/
- etc/
- vendor -> /vendor
- ...
- vendor/ (mount point)
- odm/ (mount point)
...</pre></td></tr>
<tr><td>Partition Layout</td><td>N/A</td><td>1. /boot
2. /system
3. /recovery
4. /vendor, ... etc</td><td>1. /boot
2. /system
3. /recovery
4. /vendor, ... etc</td></tr>
</table>

详细介绍参见谷歌开发者网站 system as root 说明。

4.要求默认开启 VNDK 检测及属性兼容性配置，具体参见谷歌开发者网站中 VNDK 说明。

```
# Enable VNDK Check for Android P (MUST in P)
BOARD_VNDK_VERSION := current
PRODUCT_COMPATIBLE_PROPERTY_OVERRIDE := true
```

5.是否配置 AVB(android verified boot)功能，具体参见谷歌开发者网站中 AVB 功能说明。目前 sdk 中 ATV 产品默认要求开启，BOX 产品默认关闭。

```
# Enable android verified boot 2.0
BOARD_AVB_ENABLE ?= false
```

6.要求 userdebug 模式下也必须开启 DEXPREOPT 优化，提升开机速度和应用第一次启动速度。

注意：开启 DEXPREOPT 优化对开发阶段 debug 时 push apk 和 jar 文件操作有影响，直接 push 无效。

若需要临时版本可以关闭此开关，参考下面修改配置：

副作用：由于 art 中会校验镜像，关闭后每次开机时间会很长。

```
WITH_DEXPREOPT ?= true
diff --git a/BoardConfig.mk b/BoardConfig.mk
index 8543697..cb81367 100644
--- a/BoardConfig.mk
+++ b/BoardConfig.mk
@@ -167,18 +167,6 @@ TARGET_PROVIDES_INIT_RC ?= false
 //MAX-SIZE=512M, for generate out/.../system.img
 BOARD_FLASH_BLOCK_SIZE := 131072

+WITH_DEXPREOPT_BOOT_IMG_AND_SYSTEM_SERVER_ONLY ?= true
+
+# Enable dex-preoptimization to speed up first boot sequence
+ifeq ($(HOST_OS),linux)
+  ifeq ($(TARGET_BUILD_VARIANT),user)
+    ifeq ($(WITH_DEXPREOPT),)
+      WITH_DEXPREOPT ?= true
+    endif
+  else
+    WITH_DEXPREOPT ?= false
+  endif
+endif

 ART_USE_HSPACE_COMPACT ?= true
```

3.2.3 jack-server 配置

Android9.0 系统使用 jack-server 作为 java 代码编译器，在编译过程中可能会遇到以下类似的错误：

```
Jack server already installed in "/home/yhx/.jack-server"
Communication error with Jack server (1), try 'jack-diagnose' or see Jack server log
Communication error with Jack server 1. Try 'jack-diagnose'
Communication error with Jack server 1. Try 'jack-diagnose'
```

这种情况主要是由于 jack-server 本身编译器限制，同一个网络端口号不能多个用户同时使用。

也就是在服务器上协同开发过程中，多用户同时编译 Android9.0 时，需要配置各自使用不同的网络端口号。

jack-server 的两个配置文件，决定了它所使用的端口号：

```
~/.jack-server/config.properties
~/.jack-settings
```

这两个配置文件需要配置两个端口号，分别为服务端端口号，及客户端端口号，两个配置文件中的端口号要匹配。

```
jack.server.service.port=8074
jack.server.admin.port=8075
```

及

```
SERVER_PORT_SERVICE=8074
SERVER_PORT_ADMIN=8075
```

配置步骤如下：

1) 确保两个配置文件存在，并且权限设置为 0600：

```
chmod 0600 ~/.jack-server/config.properties
chmod 0600 ~/.jack-settings
```

2) 若两个配置文件不存在，请参照以下文本新建这两个配置文件。

config.properties 文件示例如下（端口号需按实际修改）：

```
jack.server.max-jars-size=104857600
jack.server.max-service=4
jack.server.service.port=8074
jack.server.max-service.by-mem=1\=2147483648\:2\=3221225472\:3\=4294967296
jack.server.admin.port=8075
jack.server.config.version=2
jack.server.time-out=7200
```

.jack-settings 文件示例如下（端口号需按实际修改）：

```
# Server settings
SERVER_HOST=127.0.0.1
SERVER_PORT_SERVICE=8074
SERVER_PORT_ADMIN=8075

# Internal, do not touch
SETTING_VERSION=4
```

3) 修改端口号，请更改 service port 及 admin port 为其他端口号，两个配置文件里的端口号需要匹配。示例如下：

```
jack.server.service.port=8023
jack.server.admin.port=8024

SERVER_PORT_SERVICE=8023
SERVER_PORT_ADMIN=8024
```

4) 重新编译 Android，看是否会报错，若依然报错，请尝试更改其他端口号，直至编译通过。

5) 若更改 5 次编译依然无法通过，可以执行 jack-admin dump-report 命令，解压命令生成的压缩包，分析 log 日志，若出现以下 log，可以重新安装下 libcurl：

```
$ JACK_EXTRA_CURL_OPTIONS=-v jack-admin list server
* Protocol https not supported or disabled in libcurl
* Closing connection -1
Communication error with Jack server 1. Try 'jack-diagnose'
```

3.2.4 全自动编译脚本

为了提高编译的效率，降低人工编译可能出现的误操作，该 SDK 中集成了全自动化编译脚本，方便固件编译、备份。

1）该全自动化编译脚本原始文件存放于：

```
device/rockchip/rk322x/build_box.sh
```

2）在 repo sync 的时候，通过 manifest 中的 copy 选项拷贝至工程根目录下：

```
<project path="device/rockchip/rk322x" name="rk/device/rockchip/rk322x"
```

```
remote="rk" revision="rk33/mid/8.0/develop">
        <copyfile src="buildspec_box.mk" dest="buildspec.mk"/>
        <copyfile src="build_box.sh" dest="build_box.sh"/>
    </project>
```

3）修改 build.sh 脚本中的特定变量以编出对应产品固件。

```
KERNEL_DTS=rk322x-evb-android-avb
```

变量请按实际项目情况，对应修改：

Android 默认编译为 rk322x_box-userdebug 模式，也可在脚本中对应修改，可改为 rk322x_box-user 及其它配置：

```
lunch rk322x_box-user
```

4）指定 update.img 打包用的 loader：

如 RKTools\linux\Linux_Pack_Firmware\rockdev\mkupdate.sh 脚本所示：

```
fi
./afptool -pack ./ Image/update.img || pause
./rkImageMaker -RK322A Image/MiniLoaderAll.bin Image/up
echo "Making update.img OK."
#echo "Press any key to quit:"
```

Windows 打包脚本（RKTools\windows\AndroidTool\rockdev\mkupdate.bat）也是类似，如下所示：

```
RKImageMaker.exe -RK322A Image\MiniLoaderAll.bin  Imag

rem update.img is new format, Image\update.img is old
```

update.img 打包用的 loader 被命名位 MiniLoaderAll.bin，由于 SDK 更新兼容 Loader，所以在此通过 u-boot 目录编译生成 rk322x_loader_v1.07.254.bin（拷贝时会重命名为 MiniLoaderAll.bin），需要指定脚本中 loader 文件名。

5）执行自动编译脚本：

```
source build_box.sh
```

该脚本会自动配置 JDK 环境变量，编译 u-boot，编译 kernel，编译 Android，继而生成固件，并打包成 update.img。

6）脚本生成内容：

脚本会将编译生成的固件拷贝至：

IMAGE/RK3229-EVB-ANDROID-AVB_9_*****_RELEASE_TEST/IMAGES 目录下，具体路径以实际生成为准。每次编译都会新建目录保存，自动备份调试开发过程的固件版本，并存放固件版本的各类信息。

该目录下的 update.img 可直接用于 Android 开发工具及工厂烧写工具下载更新。

3.3 量产烧写

量产上考虑到生产效率及工厂工位安排，量产烧写说明详见 RKDocs\common\RKTools manuals 目录下《Rockchip_User_Guide_MP_Flashing_CN.pdf》。

在量产过程中如涉及到工具上的问题，可以联系我们的 Fae 窗口。

4 U-Boot 开发

本节简单介绍 U-Boot 基本概念和编译的注意事项，帮助客户了解 RK 平台 U-Boot 框架，具体 U-Boot 开发细节可参考 RKDocs\common\u-boot 目录下《Rockchip-Developer-Guide-UBoot-nextdev-CN.pdf》。

4.1 Rockchip U-Boot nextdev 简介

- next-dev 是 Rockchip 从 U-Boot 官方的 v2017.09 正式版本中切出来进行开发的版本。目前在该平台上已经支持 RK 所有主流在售芯片。
- 目前支持的功能主要有：
- 支持 RK Android 平台的固件启动；
- 支持最新 Android AOSP(如 GVA)固件启动；
- 支持 Linux Distro 固件启动；
- 支持 Rockchip miniloader 和 SPL/TPL 两种 pre-loader 引导；
- 支持 LVDS、EDP、MIPI、HDMI 等显示设备；
- 支持 Emmc、Nand Flash、SPI Nand flash、SPI NOR flash、SD 卡、 U 盘等存储设备启动；
- 支持 FAT、EXT2、EXT4 文件系统；
- 支持 GPT、RK parameter 分区格式；
- 支持开机 logo 显示、充电动画显示，低电管理、电源管理；
- 支持 I2C、PMIC、CHARGE、GUAGE、USB、GPIO、PWM、GMAC、EMMC、NAND、中断等驱动；
- 支持 RockUSB 和 Google Fastboot 两种 USB gadget 烧写 EMMC；
- 支持 Mass storage, ethernet, HID 等 USB 设备；
- 支持使用 kernel 的 dtb；
- 支持 dtbo 功能；

U-Boot 的 doc 目录下提供了很丰富的 README 文档，它们向开发者介绍了 U-Boot 里各个功能模块的概念、设计理念、实现方法等，建议读者好好利用这些文档提高开发效率。

4.2 平台配置

平台配置文件位于 U-Boot 根目录下的 configs 文件夹下，其中 Rockchip 相关的以 RK 开头：

```
rk3288_defconfig
rk3126_defconfig
rk3128x_defconfig
rk322x_defconfig
rk3288_defconfig
rk3326_defconfig
rk322x_defconfig
rk3399_defconfig
```

RK3229 Box 开发调试选用的是 rk322x_defconfig 配置。

4.3 固件生成

Rockchip 平台支持 MiniLoader，固件支持所有的存储设备，根据不同的平台配置生成相应的 Loader 固件。同时引入 Arm Trusted Firmware 后会生成 trust image。

以 RK322x 编译生成的镜像为例：

```
rk322x_loader_v1.07.254.bin
uboot.img
trust.img
```

其中 254 是发布的版本号，rockchip 定义 U-Boot loader 的版本，其中 254 是根据存储版本定义的，客户务必不要修改这个版本。

uboot.img 是 U-Boot 作为二级 loader 的打包。

trust.img 是 trust fireware 的打包镜像。

4.4 U-Boot 编译

RK322x Box SDK 编译使用的是如下配置：

```
./make.sh rk322x
```

编译完，会生成 trust.img、rk322x_loader_v1.07.254.bin、uboot.img 三个文件。

4.5 U-Boot logo 相关的配置

4.5.1 U-Boot logo 开关配置

Sdk 默认开启 U-Boot logo 功能，以达到更快显示开机 logo 的目的，见 kernel/ arch/arm/boot/dts/rk3229-evb-android.dtsi 中如下配置：

```
&display_subsystem {
        logo-memory-region = <&drm_logo>;
        secure-memory-region = <&secure_memory>;
        status = "okay";
        route {
                route_hdmi: route-hdmi {
                        status = "okay";
                        logo,uboot = "logo.bmp";
                        logo,kernel = "logo_kernel.bmp";
                        logo,mode = "center";
                        charge_logo,mode = "center";
                        connect = <&vop_out_hdmi>;
                };

                route_tve: route-tve {
                        status = "okay";
                        logo,uboot = "logo.bmp";
                        logo,kernel = "logo_kernel.bmp";
                        logo,mode = "center";
                        charge_logo,mode = "center";
                        connect = <&vop_out_tve>;
                };
        };
}; };
```

如果需要关闭该功能，请将上述的 dts 文件中改为 status = “disabled”。

4.5.2 U-Boot Logo 图片更换

U-boot logo 显示的两张图片是 kernel 根目录下的 logo.bmp 和 logo_kernel.bmp，如果需要更换，用同名的 bmp 替换掉，重新编译内核即可。

附：logo 替换不一定要两张图片，可以只要一张，如果只有开发者手上只有一张 logo 图片，就保留 logo.bmp 这一张即可。

附：开机 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。

4.6 U-Boot OPTEE RPMB 配置

RPMB（Replay Protected Memory Block）Partition 是 eMMC 中的一个具有安全特性的分区。

当机器硬件 flash 采用 EMMC 的情况，目前 U-boot 安全部分默认配置 optee 使用 rpmb，其主要存储安全相关的 key，和 AVB(android verified boot)相关的值。

注意：若客户机器硬件 EMMC 物料本身 rpmb 区域已被编程过（如写过非 rk 密钥）等非新料的情况，会导致开机正常安全代码引导流程异常，导致无法开机，则需要切换为非 rpmb 方式处理。

可通过下面修改切换成非 rpmb 配置：

```
diff --git a/configs/rk322x_defconfig b/configs/rk322x_defconfig
index 6c769f9af3..81301f49cf 100644
--- a/configs/rk322x_defconfig
+++ b/configs/rk322x_defconfig
@@ -121,3 +121,4 @@ CONFIG_RK_AVB_LIBAVB_USER=y
 CONFIG_OPTEE_CLIENT=y
 CONFIG_OPTEE_V1=y
 CONFIG_TEST_ROCKCHIP=y
+CONFIG_OPTEE_ALWAYS_USE_SECURITY_PARTITION=y
```

5 内核开发常见配置

本节简单介绍内核一些常见配置的修改，主要是 dts 的配置，帮助客户更快更方便的进行一些简单的修改。RK3229 kernel 版本是 4.4，config 配置文件统一为 arch/arm/configs/ rockchip_defconfig 。RK3229 的串口波特率为 1500000，调试时请保证设置准确。

5.1 DTS 介绍

5.1.1 DTS 说明

RK3229 的 dts 文件在 kernel/arch/arm/boot/dts/下，如 RK3229 evb 评估板的 dts 文件为 rk3229-evb-android-avb.dts。产品的 dts 里需根据具体的产品需求配置 CPU、GPU、DDR 的频率和电压表；配置 io、wifi、bt、温控、电配置等等。

请各位开发者尽量以 SDK 发布的示例产品 dts 文件做参考，进行后期的开发。

5.1.2 新增一个产品 DTS

RK3229 的产品 dts 文件需放在 kernel/arch/arm/boot/dts/下，

1、以 rk3229-evb-android-avb.dts 为参照，拷贝一份 dts 文件命名为 rk3229-product.dts。

2、修改 arch/arm/boot/dts/Makefile 文件，添加对应 dtb 申明

```
+rk3229-product.dtb
```

3、修改编译脚本或编译命令。

4、重新编译内核。

5.2 WiFi&BT 的配置

RK3229 Android 9.0 平台上 WiFi、BT 可做到自动兼容，按照 RK 提供的编译 Android9.0 编译步骤，生成固件后，默认就可以支持相应的 WiFi 模块,并且一套固件可以支持多个 WiFi 模块。 目前 rk3229 android 9.0 平台 wifi、bt 模块 android 和 kernel 无需做任何配置。

5.3 GPIO 对应关系注意

关于原理图上的 gpio 跟 dts 里面的 gpio 的对应关系，这边有个需要注意的地方：例如 GPIO4_C0，那么对应的 dts 里面应该是"gpio4 16"。GPIO 分为 4 个端口 PORTA（0-7）、PORTB（8-15）、PORTC（16-23）、PORTD（24-31），每个 PORT 有 8 个 PIN，以此计算可得 C0 是 16，C1 口是 17，以次类推。

GPIO 的使用请参考 RKDocs\common\PIN-Ctrl 目录下《Rockchip-Developer-Guide-Linux-Pin-Ctrl-CN.pdf》

5.4 ARM、GPU、DDR 频率修改

DVFS（Dynamic Voltage and Frequency Scaling）动态电压频率调节，是一种实时的电压和频率调节技术。目前 4.4 内核中支持 DVFS 的模块有 CPU、GPU、DDR。CPU 使用 cpufreq 框架，GPU 和 DDR 使用 devfreq 框架。

CPUFreq 是内核开发者定义的一套支持动态调整 CPU 频率和电压的的框架模型。它能有效的降低 CPU 的功耗，同时兼顾 CPU 的性能。

CPUFreq 通过不同的变频策略，选择一个合适的频率供 CPU 使用，目前的内核版本提供了以下几种策略：

- interactive：根据 CPU 负载动态调频调压；
- conservative：保守策略，逐级调整频率和电压；
- ondemand：根据 CPU 负载动态调频调压，比 interactive 策略反应慢；

- userspace：用户自己设置电压和频率，系统不会自动调整；
- powersave：功耗优先，始终将频率设置在最低值；
- performance：性能优先，始终将频率设置为最高值。

详细的模块功能及配置，请参考 RKDocs\common\DVFS 目录下《Rockchip-Developer-Guide-Linux4.4-CPUFreq-CN.pdf》

DEVFreq 是内核开发者定义的一套支持动态调整设备频率和电压的的框架模型。它能有效的降低该设备的功耗，同时兼顾其性能。目前我们的平台，有 GPU 和 DDR 在使用 DEVFreq。DEVFreq 通过不同的变频策略，选择一个合适的频率供设备使用，目前的内核版本提供了以下几种策略：

- Simple Ondemand ：根据负载动态调频调压；
- Userspace：用户自己设置电压和频率，系统不会自动调整；
- Powersave：功耗优先，始终将频率设置在最低值；
- Performance：性能优先，始终将频率设置为最高值；
- Dmc Ondemand：我司实现的 ddr 变频策略，支持负载和场景变频；

GPU 默认使用的是 Simple Ondemand 负载变频，DDR 默认使用 DMC Ondemand 的变频策略是 RK 自己实现的。

5.5 温控配置

在 Linux 内核中，定义一套温控框架 linux Generic Thermal Sysfs Drivers，它可以通过不同的策略控制系统的温度，目前常用的有以下几种策略：

- power_allocator：引入 PID（比例-积分-微分）控制，根据当前温度，动态给各模块分配 power，并将 power 转换为频率，从而达到根据温度限制频率的效果。
- step_wise ：根据当前温度，逐级限制频率。
- userspace：不限制频率

详细的模块功能及配置，请参考 RKDocs\common\Thermal 目录下《Rockchip-Developer-Guide-Linux4.4-Thermal-CN.pdf》

5.6 PWM IR 配置

红外遥控的发射电路是采用红外发光二极管来发出经过调制的红外光波；红外接收电路由红外接收二极管、三极管或硅光电池组成，它们将红外发射器发射的红外光转换为相应的电信号，再送后置放大器。鉴于家用电器的品种多样化和用户的使用特点，生产厂家对进行了严格的规范编码，这些编码各不相同，从而形成不同的编码方式，统一称为红外遥控器编码传输协议。目前 RK 平台只支持 NEC 编码的红外协议。

RK3229 平台详细的遥控器适配，键值添加，红外按键定义，及遥控器功能相关调试内容请参考 RKDocs\common\PWM 目录下《Rockchip_Developer_Guide_PWM_IR_CN.pdf》。

5.7 DDR 频率修改说明

请参考 RKDocs\common\DDR 目录下《Rockchip-Developer-Guide-DDR-CN.pdf》如何修改 ddr 频率章节。

6 Android 开发常见配置

本节简单介绍 Android 开发中一些常见配置的修改，RK3229 平台搭载的是最新的 Android9.0 系统。

6.1 Android 编译配置

6.1.1 lunch 选项说明

rk322x_box-userdebug：rk3229 平台 box 产品 userdebug（32 位）

rk322x_box-user：rk3229 平台 box 产品 user（32 位）

User 版本开启 selinux 权限校验，。开发过程中涉及到 apk 及 jar 的更新，log 打印调试相对麻烦很多。

建议开发调试阶段默认选择 userdebug 编译。

6.1.2 添加一个新的产品

各开发厂商可能有同款芯片不同产品开发的需求，一套 SDK 需同时编译生成多款产品固件。

RK3229 平台支持 Box 类型各种产品形态，当需要添加一个新的产品时，可以基于已有的 rk3229_box 来建立，如下以建立一个新的平板产品为例进行说明，具体步骤为：

1）产品命令规则：

Box 产品名中需带有“box”字样；

请务必遵守以上规则，否则系统会异常。

2）新增文件夹 device/rockchip/rk322x/rk322x_box _000，基于 rk322x_box.mk 创建 rk322x_box_000.mk，将 rk322x_box 目录下的所有文件拷贝至 rk322x_ box _000 目录下。

```
cd device/rockchip/rk322x
mkdir rk322x_box _000
cp rk322x_box.mk ./ rk322x_box _000.mk
cp rk322x_box/* rk322x_ box _000/
```

3）在 device/rockchip/rk322x/ AndroidProducts.mk 中添加：

```
PRODUCT_MAKEFILES := \
    $(LOCAL_DIR)/rk322x.mk \
    $(LOCAL_DIR)/rk322x_box.mk \
    $(LOCAL_DIR)/rk322x_box_000.mk \
```

4）在 vendorsetup.sh 中添加产品对应的 lunch 选项：

```
add_lunch_combo rk322x_box-eng
add_lunch_combo rk322x_box-userdebug
add_lunch_combo rk322x_box-user
add_lunch_combo rk322x_box_000-userdebug
add_lunch_combo rk322x_box_000-user
```

5）修改 rk322x_ box _000.mk 及 rk322x_box_000 目录下的新产品所需要修改的配置。

6）修改编译脚本或编译命令，重新 lunch 产品名称进行新产品编译。

6.2 预置 APK

Android 上的应用预安装功能，主要是指配置产品时，根据厂商要求，将事先准备好的第三方应用预置进 Android 系统。

预安装的 APK 应用需要得到对应厂商授权，若因为开发者及客户厂商私自预安装未授权应用进而需要

承担法律责任的，RK 概不负责。

预安装分为可卸载预安装和不可卸载预安装，本文主要阐述的是可卸载预安装的功能。配置步骤如下：

1）若是希望可卸载预安装，新增文件夹 device/rockchip/rk322x/rk322x_box/preinstall_del；若是不可卸载原装，新增文件夹 device/rockchip/rk322x/rk322x_box/preinstall。

2）拷贝需要预制的第三方应用到上述文件夹，注意 apk 文件名尽量使用英文，避免空格。

3）编译结束后会将预制的文件拷贝至 system 固件中。烧录后，系统会自动安装这些应用到 data/app 目录。

4）需要注意的是，在 preinstall 目录中的应用，即使用户在使用过程中将其卸载，但在恢复出厂设置后，应用又会自动安装。如果希望恢复出厂设置后不再恢复预安装应用，可以将上述文件夹名字改为 preinstall_del_forever 即可实现。

6.3 开/关机动画

需要在产品的 device/rockchip /common/BoardConfig.mk 中配置 BOOT_SHUTDOWN_ANIMATION_RINGING := true，并且准备如下相应资源文件，编译结束后对应的资源文件会拷贝到相应的 out 目录下。

将开机动画 复制到 device/rockchip/common/bootanimation.zip (源码路径)

将关机动画 复制到 device/rockchip/common/shutdownanimation.zip (源码路径)

6.4 Parameter 说明

请参考 device/rockchip/rk322x/rk322x_box 目录下 parameter.txt 文件来相应修改配置，关于 parameter 中各个参数、分区情况细节，请参考\RKDocs\common\RKTools manuals 目录下的《Rockchip_Introduction_Parameter_File_Format_CN.pdf》文档。

6.5 新增分区配置

请参考 RKDocs\android\Rockchip_Developer_Guide_Android_New_Partition_Configuration_CN.pdf

6.6 显示框架配置

请参考 RKDocs\android\Rockchip_Introduction_Android8.1_BOX_Display_Framework_Configuration_CN.pdf

6.7 OTA 升级

6.7.1 OTA 介绍

OTA （over the air）升级是 Android 系统提供的标准软件升级方式。它功能强大，提供了完全升级（完整包）、增量升级模式（差异包），可以通过本地升级，也可以通过网络升级。

详细的 OTA 升级及 Recovery 模块功能及配置，请参考 RKDocs\android 目录下《Rockchip_User_Guide_Recovery_CN&EN.pdf》。

6.7.2 生成完整包

完整包所包含内容：boot.img uboot.img vbmeta.img 及 system、vendor、oem 的升级 patch。

发布一个固件正确的顺序：

1、make -j4

2、make otapackage -j4

3、./mkimage.sh

在 out/target/product/rkxxxx/目录下会生成 ota 完整包 rkxxxx-ota-eng.root.zip，改成 update.zip 即可拷贝到 T 卡或者内置的 flash 进行升级。

6.7.3 生成差异包

OTA 差异包只有差异内容，包大小比较小，主要用于 OTA 在线升级，也可 T 卡本地升级。OTA 差异包制作需要特殊的编译进行手动制作。

1、首先发布 v1 版本的固件，生成 v1 版本的完整包

2、保存

out/target/product/rkxxxx/obj/PACKAGING/target_files_intermediates/rk3188-target_files-eng.root.zip 为 rkxxxx-target_files-v1.zip，作为 v1 版本的基础素材包。

3、修改 kernel 代码或者 android 代码，发布 v2 版本固件，生成 v2 版本完整包

4、保存

out/target/product/rkxxxx/obj/PACKAGING/target_files_intermediates/rk3188-target_files-eng.root.zip 为 rkxxxx-target_files-v2.zip，作为 v2 版本的基础素材包。

5、生成 v1-v2 的差异升级包：

```
./build/tools/releasetools/ota_from_target_files  --block -v  -i  rkxxxx-target_files-v1.zip  -p  out/host/linux-x86  -k  build/target/product/security/testkey  rkxxxx-target_files-v2.zip  out/target/product/rk322x/rkxxxx-v1-v2.zip
```

说明： 生成差异包命令格式:

ota_from_target_files

--block

-v -i 用于比较的前一个 target file

-p host 主机编译环境

-k 打包密钥

用于比较的后一个 target file

最后生成的 ota 差异包

6.8 预制 Demo

在开发及样机准备中，多数开发者及厂商有需要集成测试音视频资源、图片资源等，本 SDK 也附带了预置 Demo 资源的功能，详情见 8.8 节 OemTool 打包工具使用。

6.9 开机视频

需要在产品的 device/rockchip/common/BoardConfig.mk 中配置 BOOT_VIDEO_ENABLE ?= true，并且准备如下相应开机视频文件 bootanimation.ts（默认代码中识别此视频后缀，其它格式可直接修改其文件名及后缀为 bootanimation.ts 即可，不需要转格式），编译结束后对应的资源文件会拷贝到相应的 out 目录下。

将开机视频复制到 device/rockchip/common/bootvideo/bootanimation.ts(源码路径)

注意：开启开机视频功能后，默认已配置为将视频播完，通过属性 persist.sys.bootvideo.showtime 可控制播放时间：

-1：代表没设置时长，按照开机自然阶段时间展示；

-2：代表要将视频播完才能进入 launcher；

配置其它大于 0 的数字表示具体要播放的，超过 120 秒按 120 秒播放。

6.10 低内存机器内存优化配置

需要在产品的 device/rockchip/rk322x/rk322x_box/BoardConfig.mk 中添加配置

BUILD_WITH_GO_OPT := true

重新编译系统生效。

主要优化内容：

1.系统开启 Android Go 内存优化成果；

2.视频库播放内存占用及缓存大小优化；

3.lowmemorykiller 水线及策略调整；

目前 SDK 默认未开启。

6.11 DRM Widevine Level 1 配置

按如下修改打开配置编译，注意需要烧写 L1 key 才能生效。

```
device/rockchip/rk322x$ git diff .
diff --git a/rk322x_box/BoardConfig.mk b/rk322x_box/BoardConfig.mk
index 1425e95..3978741 100755
--- a/rk322x_box/BoardConfig.mk
+++ b/rk322x_box/BoardConfig.mk
@@ -95,7 +95,7 @@ BUILD_WITH_GTVS := false
 BUILD_WITH_GOOGLE_FRP := false

 # for widevine drm
-BOARD_WIDEVINE_OEMCRYPTO_LEVEL := 3
+BOARD_WIDEVINE_OEMCRYPTO_LEVEL := 1

 #for microsoft drm
 BUILD_WITH_MICROSOFT_PLAYREADY :=true
```

注意：开启 L1 功能时，播放需要支持 SVP 的视频，目前需要在内核中将安全内存单独划分出来，如 322x 样机板开启 svp 内存 256M 配置方法：

```
kernel$ git diff
diff --git a/arch/arm/boot/dts/rk3229-evb-android.dtsi b/arch/arm/boot/dts/rk3229-evb-android.dtsi
index 1f8d999cfb71..267f13b7e4fe 100644
--- a/arch/arm/boot/dts/rk3229-evb-android.dtsi
+++ b/arch/arm/boot/dts/rk3229-evb-android.dtsi
@@ -265,7 +265,7 @@
 * enable like this:
 * reg = <0x80000000 0x10000000>;
 */
-        reg = <0x80000000 0x0>;
+        reg = <0x80000000 0x10000000>;
 };
```

6.12 媒体中心

1.新增 4K 图片展示功能，可在对应产品目录下 device.mk 配置（例如：rk3229_box 对应

device/rockchip/rk322x/device.mk）.

persist.media.4k 属性控制默认状态。当前默认配置为 false。

persist.media.4k： true 媒体中心 4K 图片，默认显示原图大小

persist.media.4k： false 媒体中心 4K 图片，进行大小自适应调整显示

2.新增 heic 动态图片支持，对 heic 动态图片可长按“center”键，显示动态效果。

6.13 TWRP recovery

Team Win Recovery Project (TWRP)是一个开放源码软件的定制恢复模式映像，供基于安卓的设备使用。它提供了一个支持鼠标操作的界面，允许用户向第三方安装固件和备份当前的系统。

可根据 https://github.com/rockchip-software/TWRP 编译对应的 recovery.img,并通过 RKTools\windows\AndroidTool_Release_v2.65 工具烧写。

6.14 Support Magisk

Magisk 是一套用于定制 Android 的开源工具，支持高于 Android 5.0 (API 21)的设备。它涵盖了 Android 定制的基本部分: root, boot scripts, SELinux patches, AVB2.0 / dm-verity / forceencrypt removals 等。

可根据 RKDocs/android/Rockchip_User_Guide_Magisk_Installation_EN.pdf 在 Rockchip 平台安装 Magisk。

6.15 安全启动方案

Rockchip 安全启动方案基于 RK 芯片提供的硬件保护机制， 对机顶盒的引导程序 loader, uboot,trust 镜像以及 Android 系统（boot(含 kernel), recovery、 system、 vendor、 oem 等镜像） 提供可靠的安全保护。 对于机顶盒产品可用于保护机顶盒系统安全， 防止机顶盒被刷机或业务相关应用被篡改等。

可参考文档 RKDocs/common/security/Rockchip-Secure-Boot-Application-Note-V1.9.pdf

6.16 Microsoft PlayReady

PlayReady 为微软公司的新的 DRM 系统，它是 WMDRM(Windows Media DRM)的升级产品，可以为数字媒体提供内容保护支持。微软 PlayReady 官方网站为 http://www.microsoft.com/playready/，可以从该网站获取 PlayReady 产品、技术、文档、License 及支持等信息；PlayReady 官方测试网站为 http://test.playready.microsoft.com，可以使用该网站的测试用例和码流对用户的 PlayReady 产品进行基本功能测试。

Rockchip PlayReady 方案分为 SW 版本和 HW 版本，根据微软证书定义的 Security Level，我们通常也把 SW 版本称为 SL2000 版本，HW 版本称为 SL3000 版本。SW 版本能为内容和证书提供基本和必要的保护，对芯片和方案没有特别要求，适用于内容提供商没有特殊要求的场景。HW 版本使用 Rockchip 芯片的 Trustzone 硬件保护机制，对证书、加解密密钥以及解密后的码流提供更高级别的保护，HW 版本适用于内容提供商明确要求 Trustzone 特性或 Sevure Video Path 的场景。

在 rockkchip 平台使用 playready 功能，需先向微软申请 license，然后向 rockchip 申请相关补丁。

6.17 动态加载 UiMode

在盒子里，有部分 App 无法正常显示。因为全局的 UIMode 是 Configuration.UI_MODE_TYPE_ TELEVISION，这部分 App 不支持 Tv 端的显示。现在我们提供了一种方式，通过白名单配置来动态的为 App 加载 UIMode（源码路径：device/rockchip/common/uimode/uimode_app.xml,设备路径：vendor/etc/uimode_app.xml）。

目前此方式仅支持 box 设备,使用方法请参考 device/rockchip/common/uimode/ReadME.md

7 系统调试

本节重点介绍 SDK 开发过程中的一些调试工具和调试方法，并会不断补充完善，帮助开发者快速上手基础系统调试，并做出正确的分析。

7.1 ADB 工具

7.1.1 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 Box 机器的方式不同，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

7.1.2 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、Box 机器已经运行 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 Box 机器相连。

```
adb shell
```

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

7.1.3 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。

7.1.4 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。

7.1.5 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，如果可以话，可以确保一下 adbd 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
```

7.1.6 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<pid>，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。

7.1.7 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 <apk 文件路径>
```

示例如下：

```
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
```

7.2 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。

7.2.1 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
```

7.2.2 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。

7.3 Procrank 工具

Procrank 是 Android 自带一款调试工具，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）

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。

- 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）

7.3.1 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。

7.3.2 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
```

7.3.3 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。如图所示。

```
2226   49024K   48692K   30259K   27596K   com.android.systemui
2226   49036K   48704K   30271K   27608K   com.android.systemui
2226   49040K   48708K   30275K   27612K   com.android.systemui
2226   49040K   48708K   30275K   27612K   com.android.systemui
2226   49040K   48708K   30275K   27612K   com.android.systemui
2226   49040K   48708K   30275K   27612K   com.android.systemui
```

图 7-1 跟踪进程内存状态

7.4 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。

7.4.1 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。

8 常用工具说明

本节简单介绍 SDK 附带的一些开发及量产工具的使用说明，方便开发者了解熟悉 RK 平台工具的使用。详细的工具使用说明请见 RKTools 目录下各工具附带文档，及 RKDocs\common\ RKTools manuals 目录下工具文档。

8.1 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，确保各项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。
- Wifi 压力测试： 包括 Wifi 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

8.2 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（wifi）、蓝牙（bluetooth）、DDR/EMMC 存储、SD 卡（sdcard）、UST 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.pdf。

8.3 DDR 测试工具

设备上使用 DDR 测试工具，对待测设备的 DDR 进行稳定性测试，确保 DDR 功能正常及稳定。RK322x DDR 测试工具还未发布，后续会随 SDK 更新。

8.4 Android 开发工具

8.4.1 下载镜像

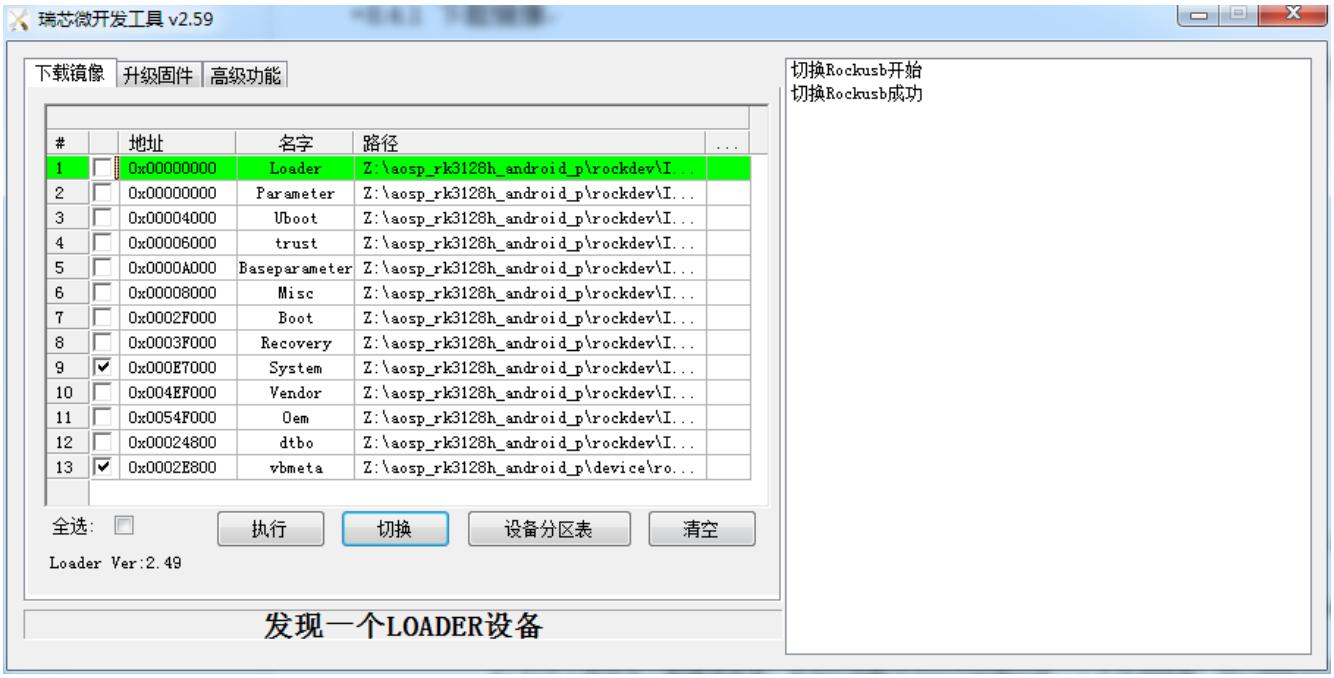

图 8-1 Android 开发工具下载镜像

1) 连接开发板进入下载模式（下载模式先按住开发板 reset 按键，再长按 recover 按键约 3-4s 时间进入 loader 模式）。
2) 打开工具点击下载镜像菜单，点击红色箭头对应列会跳出来一个文件选择框，可以选择对应分区的 img 本地地址，其他几项依次配置。
3) 配置完成后，点击执行就可以看到右边空白框进入下载提示。

其中 “低格”按钮是用来擦除设备的，“清空”按钮是清空编辑框文本。

8.4.2 升级固件

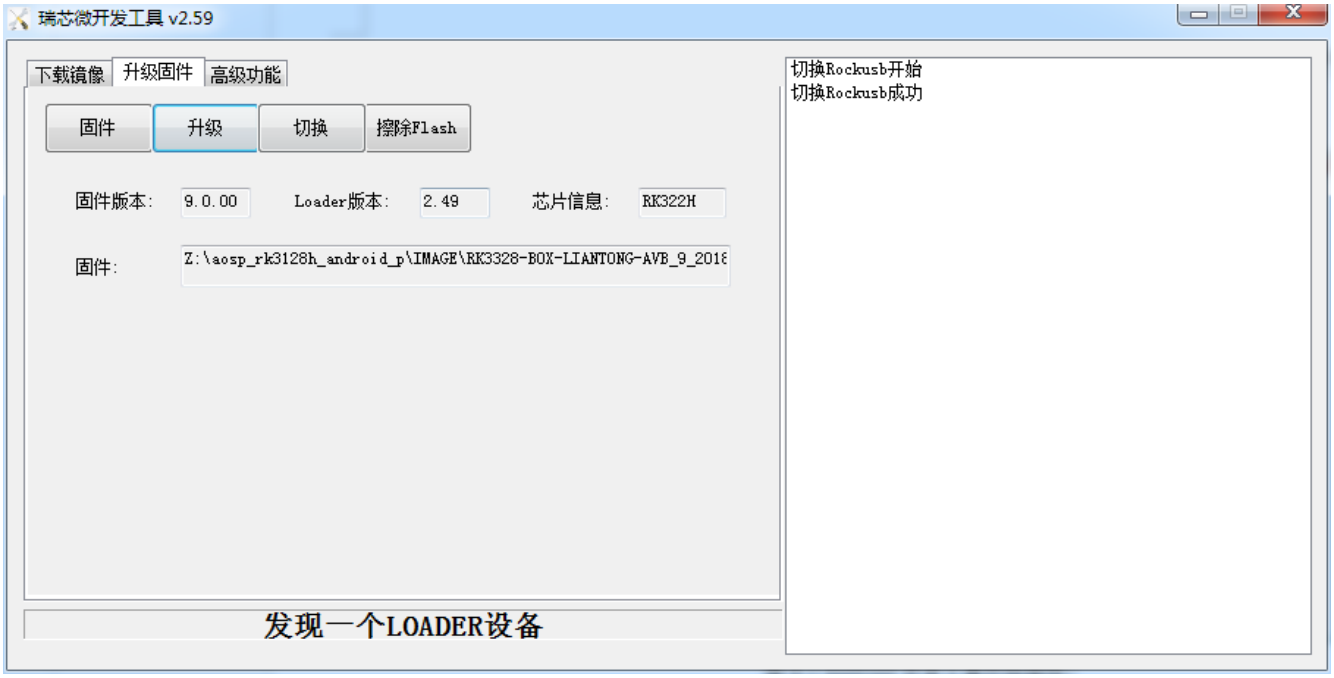

图 8-2 Android 开发工具升级固件

1) 进行打包固件。
2) 点击固件选择刚打包好的 update.img 文件，并点击升级按钮进行下载。（注意设备必须在下载模式下）。

8.4.3 高级功能

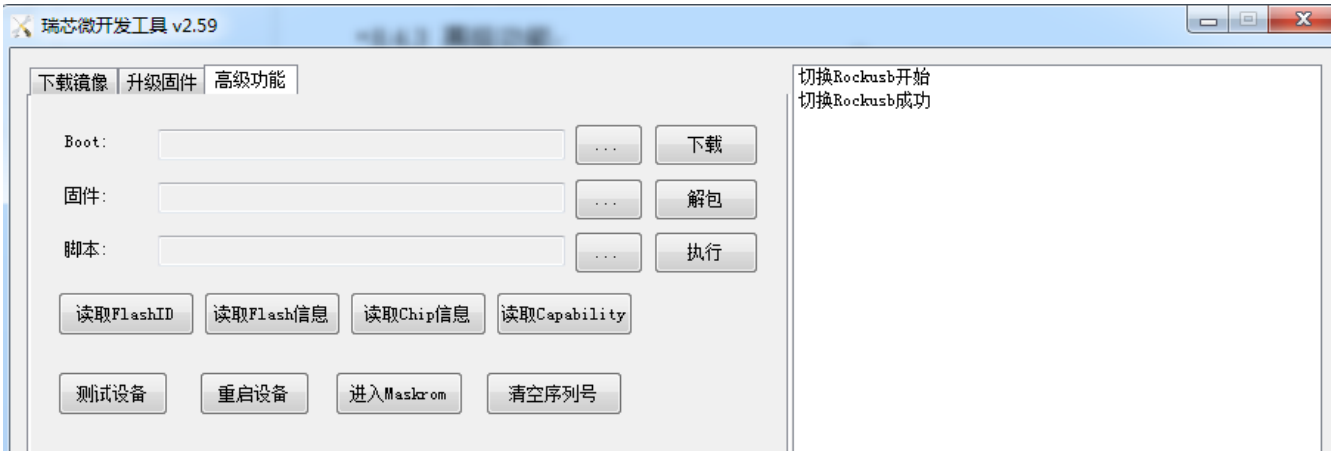

图 8-3 Android 开发工具高级功能

Boot 只能选择打包好的 update.img 文件或是 loader 的文件；
固件必须现使用打包后的 update.img;
解包功能可将 update.img 拆解为各部分镜像文件。

8.5 update.img 打包

RK322x 平台支持将各零散镜像文件，打包成一个完成的 update.img 形式，方便量产烧写及升级。具体打包步骤如下：

1) 打开 AndroidTool 工具目录底下的 rockdev 目录。编辑 package-file。

按照 package-file 进行配置，package-file 里面有一些 img 镜像放在 Image 目录底下的，如果没有

该目录存在，则自己手工新建该 Image 目录，并将需要放到 Image 目录的镜像放进去即可。且注意配置时，镜像名字的准确。其中注意 bootloader 选项，应该根据自己生成的 loader 名称进行修改。

2) 编辑 mkupdate.bat

```
Afptool -pack .\backupimage backupimage\backup.img
Afptool -pack ./ Image\update.img


RKImageMaker.exe -RK322A Image\MiniLoaderAll.bin  Image\update.img update.img -os_type:androidos

rem update.img is new format, Image\update.img is old format, so delete older format
del  Image\update.img

pause

```

图 8-4 update.img 打包脚本

需要修改 loader 名称为实际存放的 loader 名称即可。

3) 点击 mkupdate.bat 运行即可，运行完会在该目录生成一个 update.img。

8.6 固件签名工具

选择 chip 类型和加密类型，如果是 RK3229 则选择 efuse。

点击”Generate Key Pairs”按钮，则会生成公私钥对，点击保存。

点击加载密钥，会连续跳出来两次选择密钥文件的界面，第一次为选择私钥文件，第二次为公钥选择文件。

点击”Sign Firmware”按钮，签名 update.img 文件。

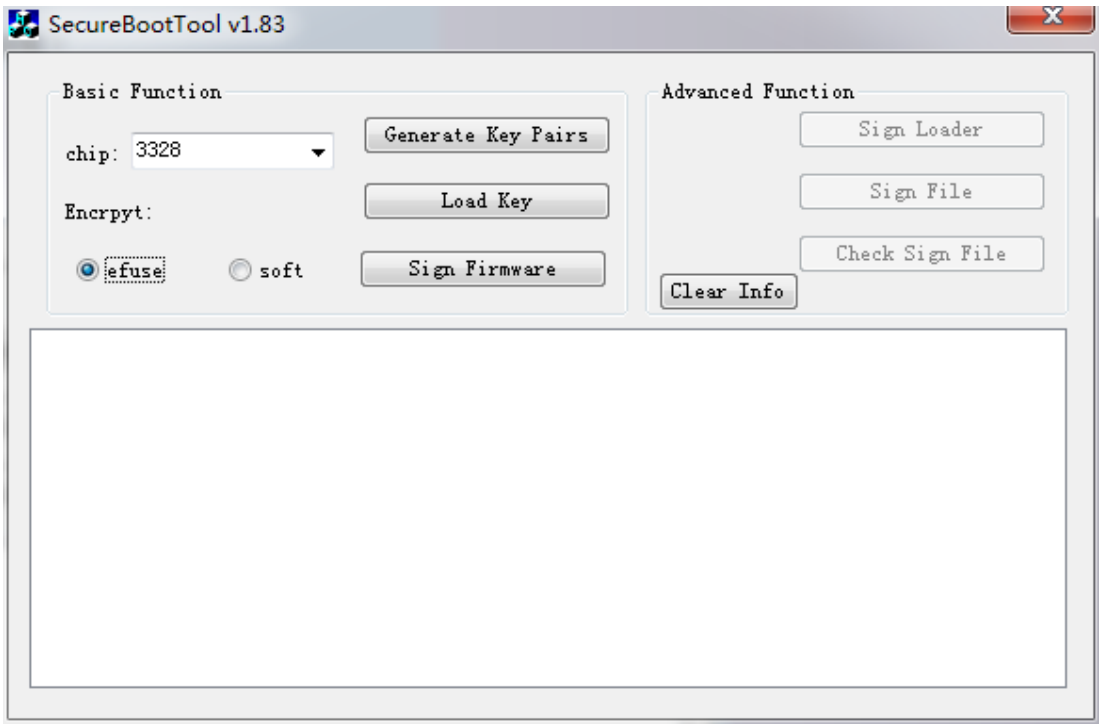

图 8-5 固件签名工具

附键盘输入 R+K+Ctrl+Alt 键可打开右侧隐藏功能。

8.7 序列号/Mac/厂商信息烧写-WNpctool 工具

在 RK3229 平台上，序列号/Mac/厂商信息烧写，都是使用 WNpctool 工具进行的。以下说明该工具基本的用法。

8.7.1 序列号获取

在 RK3229 平台上当未用工具烧写过序列号时，默认是读取 WiFi Mac 地址，并依此随机产生一个序列号的。若需要读取工具烧录的序列号值，需要手动修改对应的配置选项。

需修改/system/core/ drmservice/drmservice.c 文件中：

```
#define SERIALNO_FROM_IDB 1   //if 1 read sn from idb3;   if 0 generate sn auto
```

设为 1 后，默认会从 vendor storage 中读取工具写入的序列号。

8.7.2 WNpctool 写入步骤

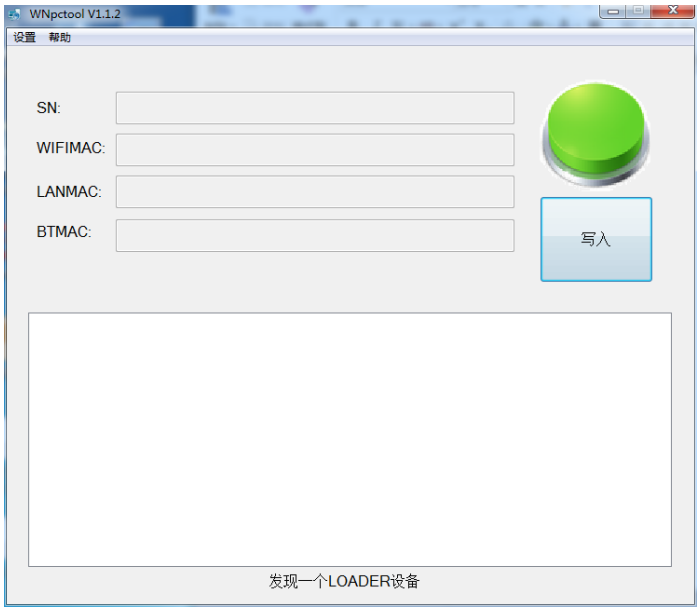

图 8-6 WNpctool 工具

1) 进入 loader 模式。
2) 点击设置按钮，会有一个下拉框按钮，点击“读取”按钮，用来切换是写入还是读取功能。切换到写入功能。
3) 点击模式，出现下列窗口，用来设置 SN/WIFI/LAN/BT

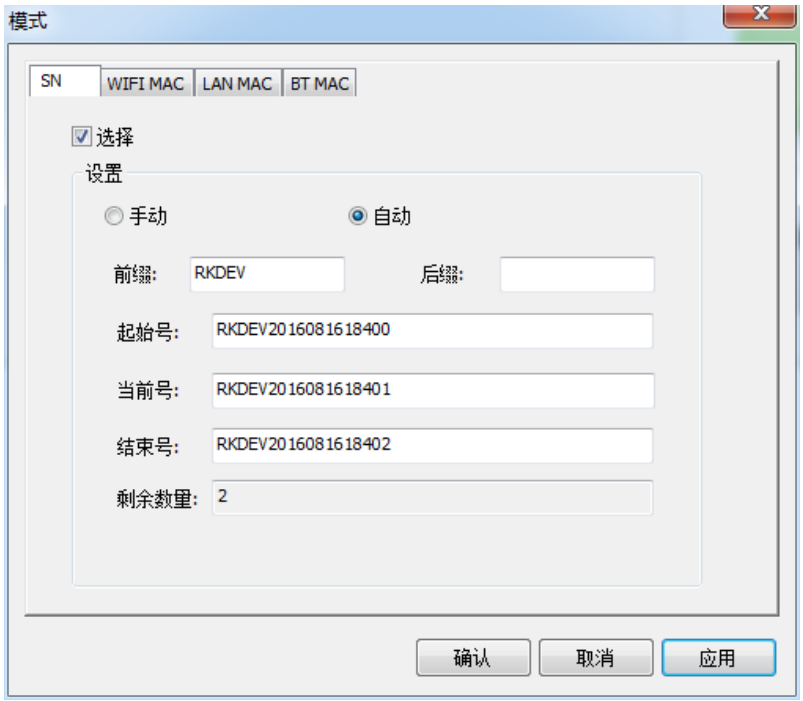

图 8-7 WNpctool 工具模式设置

4) 设置完成后，点击应用按钮，关闭窗口，返回主窗口，点击写入按钮即可。

8.7.3 WNpctool 读取步骤

1) 进入 loader 模式。
2) 点击设置按钮，会有一个下拉框按钮，点击“读取”按钮，用来切换是写入还是读取功能。切换到读取功能。
3) 点击“读取”即可。

8.8 OemTool 打包工具

8.8.1 Oem 打包工具步骤

RK3229 只支持 Ext4 镜像格式，故镜像格式选择 Ext4.下载分区默认 UserData 分区，可直接不填写。

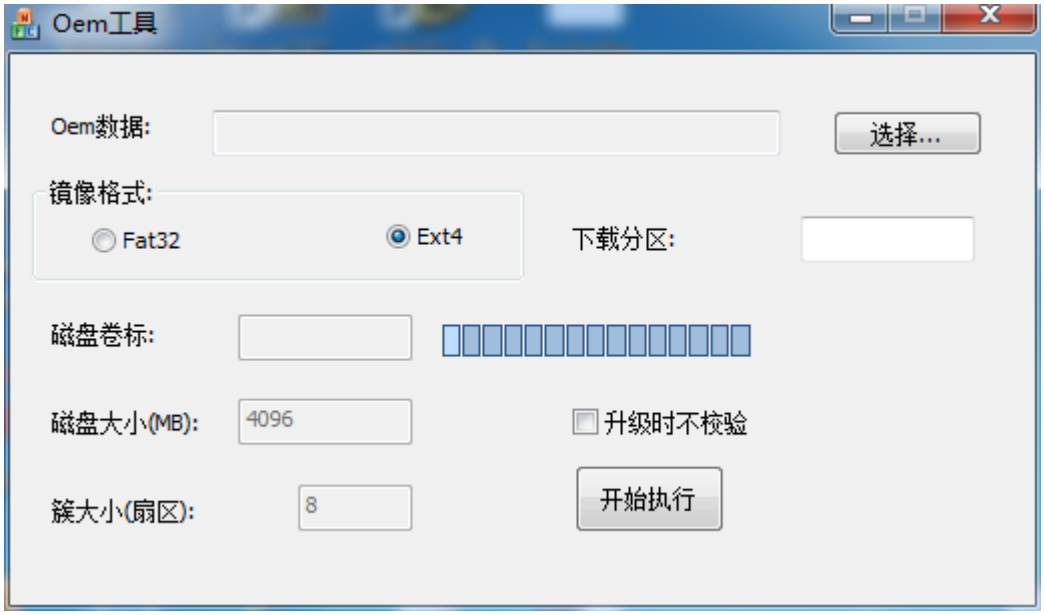

图 8-8 Oem 工具

点击选择按钮选择要打包的数据，数据必须是目录。目录最外围默认为 data 目录，假设你目录为 /data/media/0,且 0 有一个文件为 sss.txt(如下图示)。则当你升级完 demo 镜像的时候，会在 RK3229 系统上的 data 目录下有目录 media/0,且在 0 目录下有文件 sss.txt 存在。

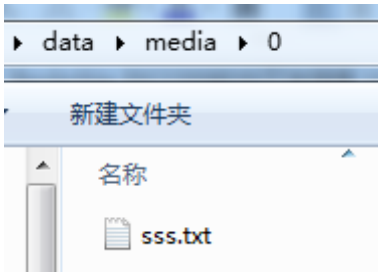

图 8-9 Oem 工具镜像制作文件夹路径要求

文件选择成功后，直接点击开始执行，会在 Oem 工具目录生成一个 OemImage.img 镜像。将镜像放在 FactoryTool 工具上下载即可。

8.9 量产工具使用

8.9.1 工具下载步骤

1) 点击固件按钮，选择打包工具打包后的 update.img，等待解包成功。
2) 如果需要 demo 镜像，则点击 Demo 拷贝按钮，添加由 Oem 工具打包的镜像，并单击 Demo 复选框。
3) 连接设备，并让设备进入 loader 或者 maskrom 模式，工具会自动进行下载。
4) 可同时连接多台设备，进行一拖多烧写，提高工厂烧写效率。

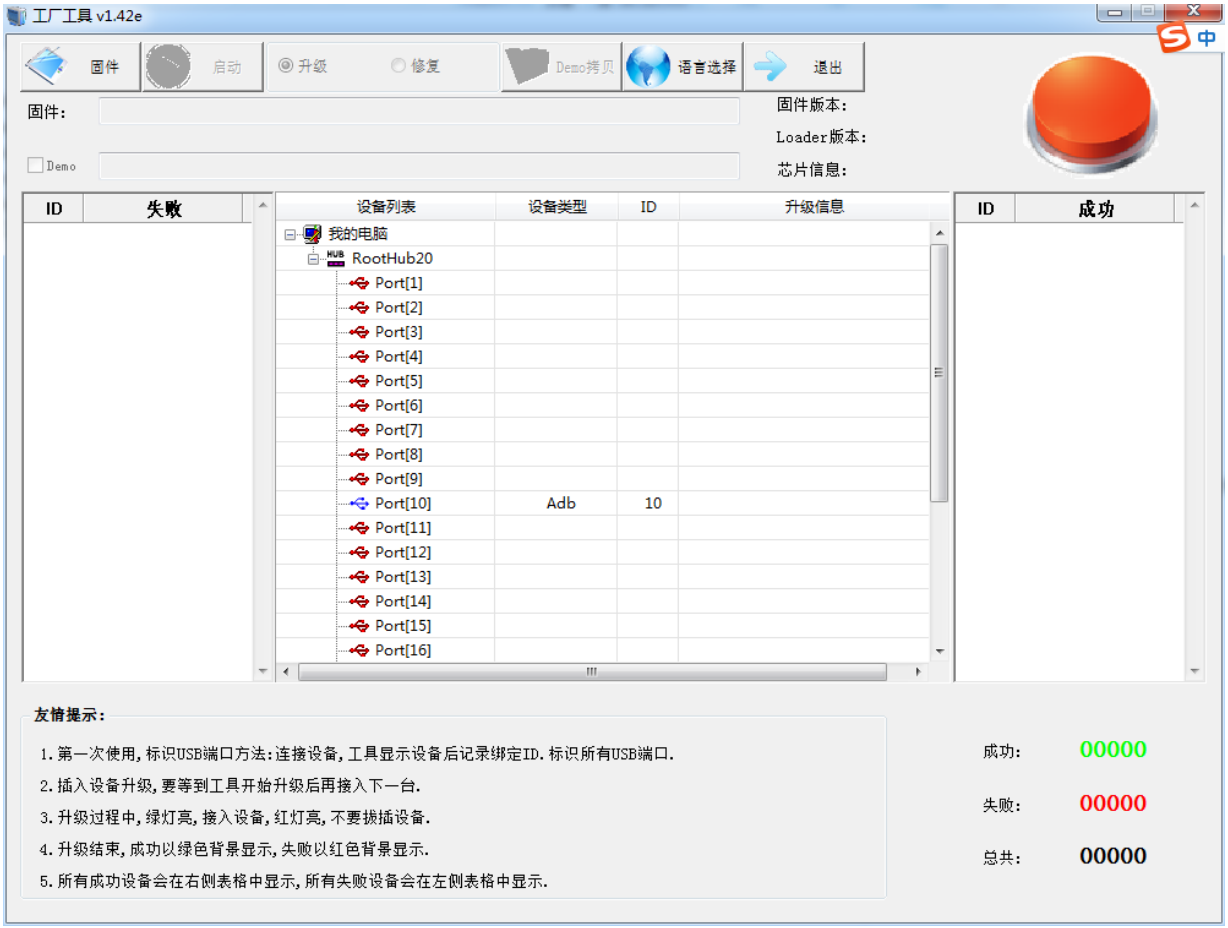

图 8-10 量产工具

8.10 Box 厂测工具

本测试工具用于帮助在量产过程中测试设备的好坏以及长时间运行的老化稳定性测试。 测试工具只需用 U 盘或 SDCard 引导启动，方便快捷，提高生产效率。

本测试工具适用于运行完整固件的 PCBA 或整机测试，包含功能测试和老化测试。功能测试主要包含 WiFi、BT、LAN、SD、USB、HDMI、左右声道、按键、LED、CVBS 等。老化测试包含 CPU、VPU、GPU、Memory 的测试。

SDK 默认编译已带有该测试工具，具体操作说明请参考 RKDocs\common\RKTools manuals\Rockchip_User_Guide_Box_Factory_Test_Tool_CN.pdf。

配置文件参考请见 RKTools\windows\Rockchip Box 厂测工具 V3.0.rar。

图 8-11 功能测试界面

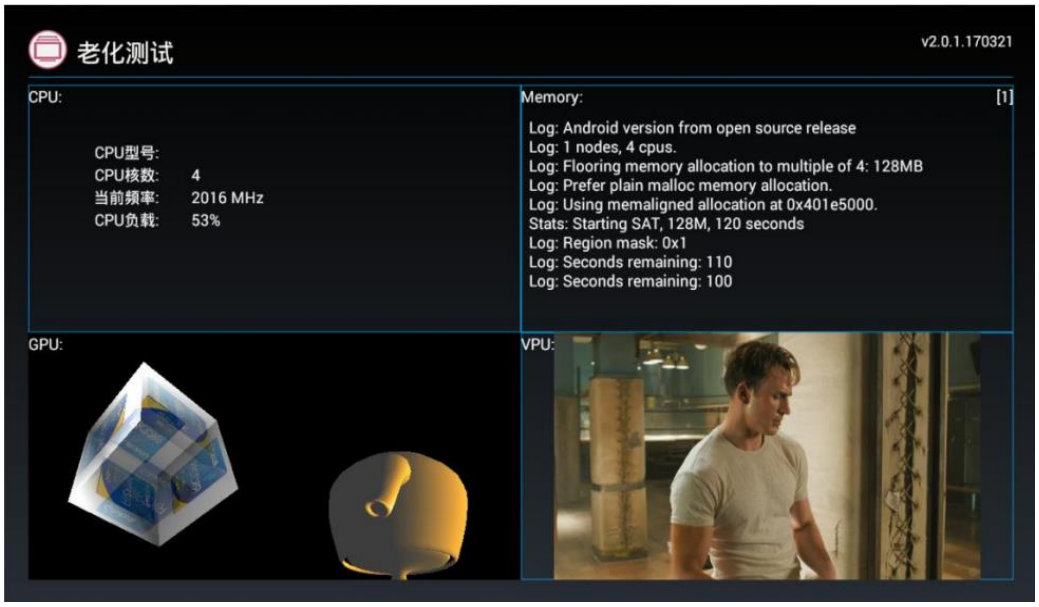

图 8-12 老化测试界面

注：由于默认该工具插外设启动，如果客户是烧机后手动安装的方式，需要重启后再测试，不然由于系统本身限制，apk 服务不会马上启动，引起无法识别测试文件问题。