

密级状态：绝密() 秘密() 内部() 公开(√)

RK3399Pro_Android8.1_软件开发指南

(技术部，第二系统产品部)

文件状态： [] 正在修改 [√] 正式发布	当前版本：	V1.03
	作 者：	周为新
	完成日期：	2018-12-12
	审 核：	吴良清
	完成日期：	2018-12-12

福州瑞芯微电子股份有限公司

Fuzhou Rockchips Electronics Co. , Ltd

(版本所有, 翻版必究)

版本历史

版本号	作者	修改日期	修改说明	备注
V1.00	周为新	2018.11.30	初版发布	
V1.01	周为新	2018.12.12	增加 evb v11 硬件版本的 dts 编译	
V1.02	吴良清	2019.04.17	修改 uboot 部分文档	
V1.03	周为新	2019.05.20	添加 sdcard 配置说明、更新 npu 开发文档路径、更新写号工具说明	

目 录

RK3399Pro_Android8.1_软件开发指南.....	1
前 言.....	1
1 支持列表.....	1
1.1 DDR 支持列表.....	1
1.2 EMMC 支持列表.....	1
1.2.1 高性能 EMMC 颗粒的选取.....	1
1.3 WiFi/BT 支持列表.....	2
1.4 SDK 软件包适用硬件列表.....	2
1.5 多媒体编解码支持列表.....	3
1.6 NPU 开发文档.....	3
2 文档/工具索引.....	4
2.1 文档索引.....	4
3 SDK 编译/烧写.....	12
3.1 SDK 获取.....	12
3.1.1 SDK 下载链接.....	12
3.1.2 repo.....	12
3.1.3 SDK 代码压缩包.....	12
3.2 SDK 编译.....	13
3.2.1 JDK 安装.....	13
3.2.2 编译模式.....	13
3.2.3 RK3399Pro Evb 编译.....	13
3.2.4 固件生成步骤.....	14
3.2.5 jack-server 配置.....	14
3.2.6 全自动编译脚本.....	16
3.3 固件烧写.....	17
3.4 量产烧写.....	18
4 U-Boot 开发.....	18
4.1 Rockchip U-Boot 简介.....	19

4.2 平台配置.....	19
4.3 固件生成.....	19
4.3.1 一级 Loader 模式.....	20
4.3.2 二级 Loader 模式.....	20
4.4 U-Boot 编译.....	20
4.5 U-Boot 充电相关配置.....	20
4.5.1 充电图片打包.....	20
4.5.2 DTS 使能充电.....	21
4.5.3 低功耗休眠.....	22
4.5.4 更换充电图片.....	22
5 Kernel 开发.....	23
5.1 DTS 介绍.....	23
5.1.1 DTS 说明.....	23
5.1.2 新增一个产品 DTS.....	23
5.2 WiFi 配置.....	23
5.3 BT 配置.....	24
5.4 GPIO.....	25
5.5 ARM、GPU、DDR 频率修改.....	25
5.6 温控配置.....	27
5.7 LPDDR4 配置.....	27
5.7.1 需要 lpddr4 的变频.....	31
5.7.2 不需要 lpddr4 变频.....	31
5.8 SD 卡配置.....	32
6 Android 常见配置.....	33
6.1 Android 产品配置.....	33
6.1.1 lunch 选项说明.....	33
6.2 常用功能配置说明.....	33
6.2.1 常用配置宏说明.....	33
6.2.2 预装 APK.....	34
6.2.3 开/关机动画及铃声.....	34

6.3 Parameter 说明.....	34
6.4 新增分区配置.....	34
6.5 OTA 升级.....	34
7 系统调试.....	34
7.1 ADB 工具.....	34
7.1.1 概述.....	34
7.1.2 USB ADB 使用说明.....	35
7.1.3 网络 ADB 使用要求.....	35
7.1.4 SDK 网络 ADB 端口配置.....	36
7.1.5 网络 ADB 使用.....	36
7.1.6 手动修改网络 ADB 端口号.....	36
7.1.7 ADB 常用命令详解.....	36
7.2 Logcat 工具.....	38
7.2.1 Logcat 命令使用.....	38
7.2.2 常用的日志过滤方式.....	38
7.2.3 查看上次 log.....	39
7.3 Procrank 工具.....	39
7.3.1 使用 procrank.....	40
7.3.2 检索指定内容信息.....	41
7.3.3 跟踪进程内存状态.....	41
7.4 Dumpsys 工具.....	41
7.4.1 使用 Dumpsys.....	42
7.5 串口调试.....	42
7.5.1 串口配置.....	42
7.5.2 FIQ 模式.....	42
7.6 音频 codec 问题调试工具及文档.....	43
7.7 Last log 开启.....	43
8 常用工具说明.....	43
8.1 StressTest.....	44
8.2 PCBA 测试工具.....	44

8.3 DDR 测试工具.....	45
8.4 Android 开发工具.....	45
8.4.1 下载镜像.....	45
8.4.2 升级固件.....	46
8.4.3 高级功能.....	46
8.5 update.img 打包.....	47
8.6 固件签名工具.....	47
8.7 序列号/Mac/厂商信息烧写-RKDevInfoWriteTool 工具.....	47
8.7.1 使用 RKDevInfoWriteTool 写入.....	48
8.7.2 使用 RKDevInfoWriteTool 读取.....	49
8.8 量产工具使用.....	50
8.8.1 工具下载步骤.....	50

前 言

概述

本文档主要介绍 Rockchip RK3399Pro Android8.1 软件开发指南，旨在帮助软件开发工程师更快上手 RK3399Pro 的开发及调试。

产品版本

芯片名称	内核版本	Android 版本
RK3399Pro	Linux4.4	Android8.1.0

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

1 支持列表

1.1 DDR 支持列表

RK3399Pro DDR 目前选型列表支持双通道 DDR3、DDR3L、LPDDR3、LPDDR4。

表 1-1 RK3399Pro DRAM Support Type

Chip	DRAM Support Type
RK3399Pro	DDR3/DDR3L/LPDDR3/LPDDR4

RK3399Pro DDR 颗粒支持程度列表，详见 RKDocs\common\Platform support lists 目录下《RK DDR Support List Ver2.34》，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 RK3399Pro DDR Support Symbol

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

1.2 EMMC 支持列表

RK3399Pro 支持 eMMC 5.1，SDIO3.0，可运行 HS200,HS400 模式，详见 RKDocs\common\Platform support lists 目录下《RKeMMCSupportList Ver1.41》，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 1-3 RK3399Pro EMMC Support Symbol

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

1.2.1 高性能 EMMC 颗粒的选取

为了提高系统性能，选取高性能的 EMMC 颗粒也是需要的。请在挑选 EMMC 颗粒前，参照我们的支持列表的型号，对应的研究下厂商提供的 Datasheet，重点关注下厂商标注的 performance 一章节。

参照厂商大小、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，也可直接联系我们的 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

1.3 WiFi/BT 支持列表

RK3399Pro 和 RK3399 WiFi/BT 支持列表可以共用，RK3399Pro 内核运行 Linux4.4，WiFi/BT 支持列表，详见 RKDocs\common\Platform support lists 目录下《Rockchip_WiFi_Situation_20180403.pdf》，下表中所标示为目前 RK3399 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，可先与 WiFi/BT 芯片原厂沟通，是否有可以稳定在 Linux4.4 运行的驱动程序，并能提供调试帮助。

另外后续我们会不断更新支持列表，如果疑问和建议可以与我们的 Fae 窗口联系(WiFi/BT avl 可以和 RK3399 共用)。

RK3399 Wi-Fi Situation													
WiFi Chip	IFACE	IEEE 802.11 Standard	2.4GHz Band	5.0GHz Band	BT	GPS	NFC	11AC	SDIO3.0	MIMO	BT4.0	BT4.2	Android7.1
AP6330	SDIO	IEEE 802.11A/B/G/N	✓	✓	✓	×	×	×	×	×	✓	×	✓
AP6255	SDIO	IEEE 802.11A/B/G/N/AC	✓	✓	✓	×	×	✓	✓	×	✓	✓	✓
AP6354	SDIO	IEEE 802.11A/B/G/N/AC	✓	✓	✓	×	×	✓	✓	✓	✓	×	✓
1. ✓：支持 ×：不支持 注：空的表示没调过													
2. 该列表仅适用kernel4.4													

图 1-2 RK3399 目前大量测试的 Wifi/Bt 支持列表

1.4 SDK 软件包适用硬件列表

本 SDK 是基于谷歌 Android8.1 64bit 系统，适配瑞芯微 RK3399Pro 芯片的软件包。

如果使用瑞芯微提供的开发板，具体参考《3399Pro_Evb 板说明》，kernel 配置可直接使用 rk3399pro-evb-v10.dts 进行配置。

1.5 多媒体编解码支持列表

RK3399Pro 多媒体方面支持强大，支持 4K VP9 and 4K 10bits H265/H264 视频解码，高达 60fps，1080P 多格式视频解码 (WMV, MPEG-1/2/4, VP8)，1080P 视频编码，支持 H.264，VP8 格式，视频后期处理器：反交错、去噪、边缘/细节/色彩优化。

具体的编解码支持列表，详见 RKDocs\rk3399Pro 目录下《RK3399 Multimedia Codec Benchmark v1.0》。

1.6 NPU 开发文档

NPU 的开发资料参考以下目录：

/rknn-toolkit

/RKNPUTools

RKDocs/rk3399pro/RK3399Pro_npu 上电及启动说明_V1.0_20190510.pdf

2 文档/工具索引

2.1 文档索引

RK3399Pro SDK 发布文档旨在帮助开发者快速上手开发及调试，文档中涉及的并不能涵盖所有的知识和问题。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。

RK3399Pro SDK 中在 RKDocs 目录下附带了三大块的文档，分别为：android（android 相关开发文档），rk3399Pro(3399Pro 相关发布文档)，common（公共开发文档）；common 目录细分为内核驱动开发文档、uboot 开发文档、模块开发文档、Platform support lists（支持列表）、RKTools manuals（工具使用文档）等。

- |—— android
 - | |—— Android8.0_OEM 内容预置功能说明_V1.0_20171122.pdf
 - | |—— Android8.0_定制开关机动画（铃声）说明_V1.0_20170923.pdf
 - | |—— Android8.0_性能模式使用说明_V1.0_20170923.pdf
 - | |—— Android8.0_恢复出厂设置保护功能说明_V1.0_20170923.pdf
 - | |—— Android8.0_预安装应用功能说明文档_V1.0_20171109.pdf
 - | |—— Android8.0_验证启动功能说明_V1.0_20171109.pdf
 - | |—— Android 增加一个分区配置指南 V1.00.pdf
 - | |—— bt
 - | |—— ROCKCHIP_ANDROID_8.1_BT 配置说明_V1.0_20180103.pdf
 - | |—— project.config
 - | |—— RK_PCBA_Camera_移植说明_v1.0.pdf
 - | |—— Rockchip Android 8.1 BOX 显示框架配置说明文档 V1.0-20180210.pdf
 - | |—— Rockchip Box 媒体中心使用说明-v1.0.1-20170216.pdf
 - | |—— ROCKCHIP_PCBA 测试工具开发指南_V1.2_20180509.pdf
 - | |—— Rockchip Recovery 用户操作指南 V1.03.pdf
 - | |—— wifi
 - | |—— RealTek wifi 驱动移植说明_V1.1.pdf
 - | |—— ROCKCHIP_ANDROID_8.1_WIFI 配置说明_V1.2.pdf
- |—— common
 - | |—— camera

		——	Camera 目录文档说明.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(适用 3288&3368).pdf
		——	RK_ISP10_Camera_User_Manual_v2.2.pdf
		——	RKISPV1_Camera_Module_AVL_v1.7.pdf
		——	RKISPV1_Camera_常见问题解决方法 V1.0.pdf
		——	RKISPV1_Camera_驱动调试方法 V1.0.pdf
		——	Rockchip_Camera_AVL_v2.0_Package_20180515.7z
		——	Rockchip SOFIA 3G-R_PMB8018(x3_C3230RK)_Camera_Module_
			AVL_v1.6_20160226.pdf
		——	DDR
		——	DDR 开发指南.pdf
		——	DDR 问题排查手册.pdf
		——	debug
		——	perf 使用说明.pdf
		——	RK3399-LOG-EXPLANATION.pdf
		——	streamline 使用说明.pdf
		——	systrace 使用说明.pdf
		——	display
		——	rockchip_drm_integration_helper-zh.pdf
		——	Rockchip_DRM_Panel_Porting_Guide_V1.5_20180830.pdf
		——	Rockchip 基于 DRM 框架的 HDMI 开发指南 v1.1-20180322.pdf
		——	基于 DRM 的 Android 显示使用指南_V1.0_20180129.pdf
		——	driver
		——	RK817_RK809_Codec 开发指南_V1.0_20180228.pdf
		——	RK 语音通话 3A 算法集成说明及参数调试说明文档_V3.0.pdf
		——	Rockchip Audio 开发指南 V1.1-20170215-linux4.4.pdf
		——	Rockchip CPU-Freq 开发指南 V1.0.1-20170213.pdf

- | | |—— Rockchip-Developer-Guide-linux4.4-PCIe.pdf
- | | |—— Rockchip-Developer-Guide-linux4.4-SDMMC-SDIO-eMMC.pdf
- | | |—— Rockchip-Developer-Guide-linux4.4-USB.pdf
- | | |—— Rockchip-Developer-Guide-MCU.pdf
- | | |—— Rockchip-Developer-Guide-SPI.pdf
- | | |—— Rockchip-Developer-Guide-UART.pdf
- | | |—— Rockchip DEVFreq 开发指南 V1.0-20160701.pdf
- | | |—— Rockchip gmac 模块 开发指南 V1.0-20170221.pdf
- | | |—— Rockchip I2C 开发指南 V1.0-20160629.pdf
- | | |—— Rockchip IO-Domain 开发指南 V1.0-20160630.pdf
- | | |—— Rockchip Pin-Ctrl 开发指南 V1.0-20160725.pdf
- | | |—— Rockchip pwm ir 开发指南 V1.00.pdf
- | | |—— Rockchip pwm 背光 开发指南-20170220.pdf
- | | |—— Rockchip RK805 开发指南 V1.0-20170217.pdf
- | | |—— Rockchip RK816 开发指南 V1.pdf
- | | |—— Rockchip RK818_6 电量计 开发指南 V2.0-20170525mo.pdf
- | | |—— Rockchip RK818 电量计 开发指南 V1.0-20160725.pdf
- | | |—— Rockchip_Sensors_开发指南_V1.0_20180605.pdf
- | | |—— Rockchip Thermal 开发指南 V1.0.1-20170428.pdf
- | | |—— Rockchip Vendor Storage Application Note.pdf
- | | |—— Rockchip 以太网 开发指南 V2.3.1-20160708.pdf
- | | |—— Rockchip 休眠唤醒 开发指南 V0.1-20160729.pdf
- | | |—— Rockchip 时钟子模块 开发指南 V1.1-20170210.pdf
- | | |—— Rockchip 电源 独立 DCDC 开发指南 V1.0-20170519.pdf
- | |—— hdmi-in
- | | |—— HDMI_IN_开发指南_V1.0_20180726.pdf
- | |—— mobile-net
- | | |—— 3G 数据卡 USB 切换文件制作说明_v1.2.pdf
- | | |—— ROCKCHIP_3G_DONGLE_配置说明_V1.0.pdf
- | |—— Platform support lists

- | | |—— RK3128 BOX Hardware Design Guide V10-201410.pdf
- | | |—— RKeMMCSupportList Ver1.41_20181030.pdf
- | | |—— RKNandFlashSupportList Ver2.72_2016_08_30.pdf
- | | |—— RK DDR Support List Ver2.34.pdf
- | | |—— Rockchip_Camera_AVL_v2.0_Package.7z
- | | |—— Rockchip Kodi 支持程度列表_V2.0_20170715.pdf
- | | |—— Rockchip_WiFi_Situation_20180611.pdf
- | |—— RKTools manuals
- | | |—— Android 开发工具手册.pdf
- | | |—— REPO 镜像服务器搭建和管理_V2.2_20131231.pdf
- | | |—— RKUpgrade_DII_UserManual.pdf
- | | |—— RK 平台 apache_tomcat_ota 服务器搭建说明.rar
- | | |—— rk 平台量产升级指导文档 V1.1.pdf
- | | |—— RockChip Box 厂测工具 V2.0.rar
- | | |—— Rockchip Box 厂测工具操作说明 V2.0.pdf
- | | |—— Rockchip Keybox Burning Guide V1.2-20180315.pdf
- | | |—— Rockchip Parameter File Format Ver1.3.pdf
- | | |—— Rockchip 量产烧录 指南 V1.1-20170214.pdf
- | | |—— WNPctool 写号工具简要使用说明_V1.1.2.pdf
- | | |—— 压力测试 Stresstest 文档 forVR_ver3.0.pdf
- | | |—— 瑞芯微 KeyWrite 使用指南_V1.3_20180508.pdf
- | | |—— 量产工具升级及相关问题处理.pdf
- | |—— security
- | | |—— Rockchip-Secure-Boot-Application-Note-V1.9.pdf
- | | |—— Rockchip_TEE 安全 SDK 开发手册_V1.1_20170516.pdf
- | |—— u-boot
- | | |—— Rockchip-Developer-Guide-Trust.pdf
- | | |—— Rockchip-Developer-Guide-UBoot-nextdev.pdf
- | | |—— Rockchip U-Boot 开发指南 V3.8-20170214.pdf
- | |—— usb

- | |—— RK USB Compliance Test Note V1.2.1.pdf
- | |—— Rockchip-Developer-Guide-linux4.4-USB.pdf
- | |—— Rockchip-USB-Performance-Analysis-Guide.pdf
- | |—— Rockchip-USB-SQ-Test-Guide.pdf
- | └—— rk3399pro
 - | |—— RK3399 Multimedia Codec Benchmark v1.0.pdf
 - | |—— RK3399Pro_EVB 板简介_20181121.pdf
 - | |—— RK3399_SDK 多媒体性能指标说明文档_V1.0_20180109.pdf
 - | └—— RK3399 USB DTS Configuration Instruction.pdf 工具索引

RK3399Pro SDK 发布的工具，用于开发调试阶段及量产阶段使用。工具可能随 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。

RK3399Pro 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）。

- | |—— linux
 - | |—— Linux_AttestationKeyboxPack_Tool.rar
 - | |—— Linux_Pack_Firmware
 - | |—— Linux_rockdev.zip
 - | └—— rockdev
 - | |—— afptool
 - | |—— Image
 - | |—— boot.img
 - | |—— kernel.img
 - | |—— MiniLoaderAll.bin
 - | |—— misc.img
 - | |—— parameter.txt
 - | |—— pcba_small_misc.img
 - | |—— pcba_whole_misc.img
 - | |—— recovery.img
 - | |—— resource.img
 - | |—— system.img


```

|   |   |   |—— Log2018-05-10.txt
|   |   |   |—— ScanLog2018-05-10.txt
|   |   |—— Readme.txt
|   |—— AndroidTool_Release_v2.54.zip
|   |—— rockdev
|       |—— AFPTool.exe
|       |—— backupimage
|       |   |—— backup.img
|       |   |—— package-file
|       |—— baseparamer.img
|       |—— mkupdate.bat
|       |—— package-file
|       |—— recover-script
|       |—— RKImageMaker.exe
|       |—— update-script
|—— Demo 镜像烧写工具包.zip
|—— DriverAssitant_v4.5.zip
|—— efuse_v1.37.rar
|—— FactoryTool_v1.63.zip
|—— FWFactoryTool-5.4.rar
|—— KeyBoxWrite_v1.53.rar
|—— OemTool_v1.3.rar
|—— parameter_adjustment_tool.xlsx
|—— rk312x-pcba-tools.rar
|—— RKImageMaker_v1.62.zip
|—— Rockchip Box 厂测工具 V2.0-M-20170327.zip
|—— Rockchip 平台 DDR 测试工具_V1.35 发布通知.7z
|—— SDDiskTool_v1.56.zip
|—— SecureBootTool_v1.85_foruser.zip
|—— SpiImageTools_v1.41.zip

```

- |—— UpgradeDIIITool_v1.35.zip
- |—— Windows_TA_Sign_Tool.rar
- |—— WNpctool_Setup_V1.2.0.0522.rar
- └—— 电池曲线检测工具
 - |—— ADC 电池测试工具_V2.3.pdf
 - └—— BatteryArray_V2.4.apk

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。

3.1.1 SDK 下载链接

RK3399Pro_ANDROID8.1_SDK 下载地址如下：

```
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_oreo_stable/platform/rk3399pro/manifests.git -m Rk3399pro_Android_Oreo_release.xml
```

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/repo/rk/tools/repo
```

3.1.2 repo

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/repo/rk/tools/repo
```

3.1.3 SDK 代码压缩包

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，开发者可以通过这种方式，获得 SDK 代码的初始压缩包，该压缩包解压得到的源码，与通过 repo 下载的源码是一致的。以 Rk3399Pro_Android8.1_SDK_Beta_V0.1_20181130.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 rk3399Pro
tar zxvf Rk3399Pro_Android8.1_SDK_Beta_V0.1_20181130.tar.gz -C rk3399Pro
cd rk3399Pro
.repo/repo/repo sync -l
.repo/repo/repo sync
```

后续开发者可根据 Fae 窗口定期发布的更新说明，通过“.repo/repo/repo sync”命令同步更新。

3.2 SDK 编译

3.2.1 JDK 安装

Android8.1 系统编译依赖于 JAVA 8。编译之前需安装 OpenJDK。

安装命令如下：

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

配置 JAVA 环境变量，例如，安装路径为/usr/lib/jvm/java-8-openjdk-amd64，可在终端执行如下命令配置环境变量：

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PATH=$JAVA_HOME/bin:$PATH
export CLASSPATH=.:$JAVA_HOME/lib:$JAVA_HOME/lib/tools.jar
```

SDK 带有 Open JDK8 的配置脚本，在工程根目录下，命名为 javaenv.sh。

可直接执行以下命令，配置 JDK：

```
source javaenv.sh
```

3.2.2 编译模式

SDK 默认以 userdebug 模式编译。

使用 adb 时，需要先执行 adb root，adb disable-verity 关闭 system 分区的 verity 特性，重启后再执行 adb root, adb remount，进而进行 push 操作来 debug。

3.2.3 RK3399Pro Evb 编译

uboot 编译：

```
cd u-boot
make rk3399pro_defconfig
./mkv8.sh
```

如果更新了 next_dev 分支的 uboot 分支使用 ./make.sh rk3399pro

kernel 编译：

如果客户有拿到我们的 evb_v10 的硬件板子（绿色）kernel 编译方法如下：

```
cd kernel
make ARCH=arm64 rockchip_defconfig -j8
make ARCH=arm64 rk3399pro-evb-v10.img -j12
```

如果客户有拿到我们的 evb_v11 的硬件板子（黑色）kernel 编译方法如下：

```
cd kernel  
  
make ARCH=arm64 rockchip_defconfig -j8  
  
make ARCH=arm64 rk3399pro-evb-v11.img -j12
```

android 编译:

```
source build/envsetup.sh  
  
lunch rk3399pro-userdebug  
  
make -j12  
  
./mkimage.sh
```

3.2.4 固件生成步骤

执行./mkimage.sh 后, 在 rockdev/Image-xxx/ 目录生成完整的固件包(xxx 是具体 lunch 的产品名)

```
rockdev/Image-xxx/  
├── boot.img  
├── kernel.img  
├── MiniLoaderAll.bin  
├── misc.img  
├── oem.img  
├── parameter.txt  
├── pcba_small_misc.img  
├── pcba_whole_misc.img  
├── recovery.img  
├── resource.img  
├── system.img  
└── trust.img
```

3.2.5 jack-server 配置

Android8.1 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 **Android7.1** 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(yhx 为对应用户的用户名), 决定了它所使用的端口号:

```
/home/yhx/.jack-server/config.properties
```

```
/home/yhx/.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 /home/yhx/.jack-server/config.properties
```

```
chmod 0600 /home/yhx/.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\=42
94967296
```

```
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'
```

3.2.6 全自动编译脚本

如前几节所述，编译可大致分为 u-boot、kernel、android 三大部分进行编译，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/RK3399Pro/build.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/rk3399Pro"
name="rk/device/rockchip/rk3399Pro" remote="rk"
revision="rk33/mid/8.1/develop">
    <copyfile src="buildspec.mk" dest="buildspec.mk"/>
    <copyfile src="build.sh" dest="build.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=rk3399pro-evb-v10
```

变量请按实际项目情况，对应修改：

KERNEL_DTS 变量指定编译 kernel 的产品板极配置；

Android 编译需要指定对应的 lunch 选项，请在执行 build.sh 之前执行 lunch 操作，确保使用了正确的 lunch 选项，例如：

```
lunch rk3399Pro-user
```

4) 执行自动编译脚本：

```
source build.sh
```

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

5) 脚本生成内容：

脚本会将编译生成的固件拷贝至：

IMAGE/RK3399Pro *****_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。建议在每次大版本编译的时候，使用这个编译脚本生成固件，里面包含了很多的版本信息，便于追查问题的时候定位代码的状态。

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 固件烧写

刷机说明详见 RKDocs\common\RKTools manuals 目录下《Android 开发工具手册.pdf》。SDK 提供烧写工具，如下图所示。编译生成相应的固件后，进入烧写模式，即可进行刷机。对于已烧过其它固件的机器，可以选择重新烧录固件，或是选择低格设备，擦除 idb，然后进行刷机。

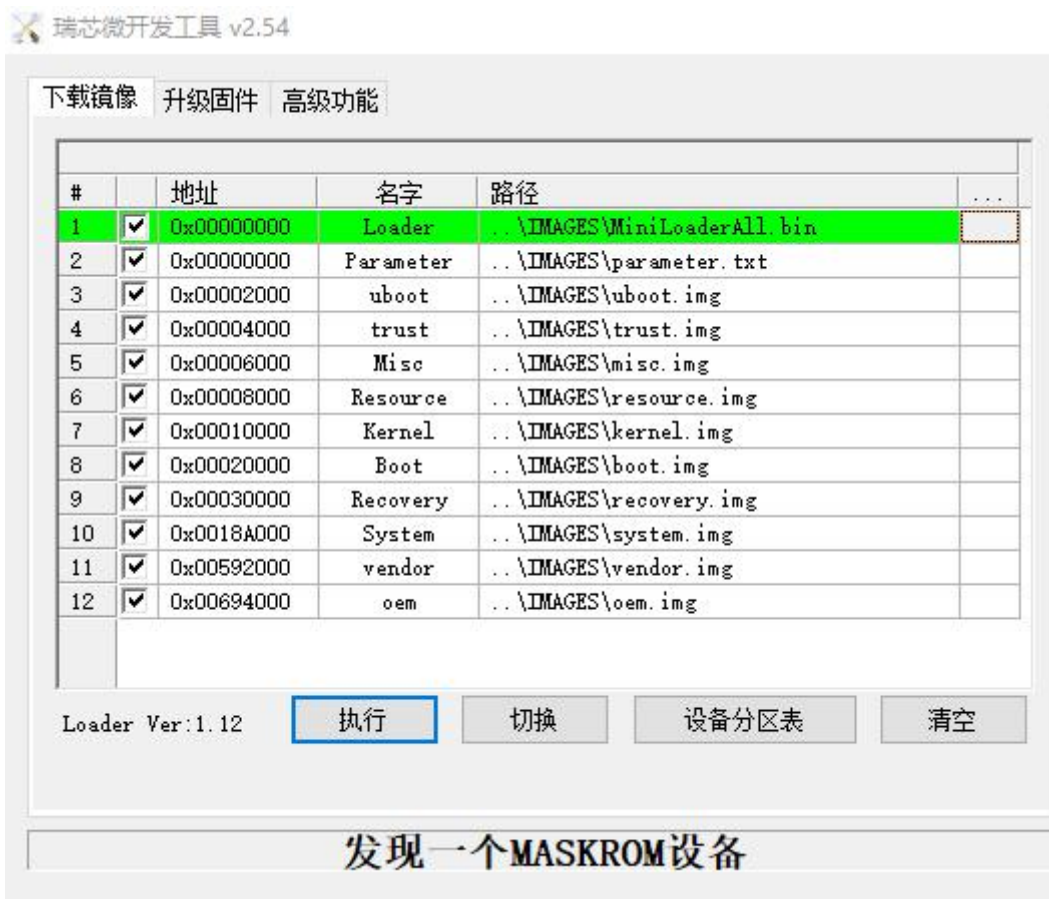


图 3-1 Android 开发工具烧写界面

注:

- 1) 烧写前, 需安装最新的 USB 驱动, 驱动详见:

RKTools/windows/

└── DriverAssitant_v4.5.zip

- 2) Android 8.1 多了 vendor.img 和 oem.img, 固件烧写的时候必须烧写这两个 img, 否则系统无法开机。

3.4 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 量产烧录指南 V1.1-20170214.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.pdf》。

4.1 Rockchip U-Boot 简介

Rockchip U-Boot 是基于开源的 UBoot 2014.10 正式版进行开发的，主要支持：

- 支持芯片：rk3288、rk3036、rk312x、rk3368、rk312x、rk3366、rk3399 等；
- 支持 Android 平台的固件启动；
- 支持 ROCKUSB 和 Google Fastboot 两种方式烧写；
- 支持 secure boot 固件签名加密保护机制；
- 支持 LVDS、EDP、MIPI、HDMI、CVBS 等显示设备；
- 支持 SDCard、Emmc、Nand Flash、U 盘等存储设备；
- 支持开机 logo 显示、充电动画显示，低电管理、电源管理；
- 支持 I2C、SPI、PMIC、CHARGE、GUAGE、USB、GPIO、PWM、DMA、GMAC、EMMC、NAND 中断等驱动；

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，并根据产品形态分为 MID 和 BOX 两种配置：

```
rk3288_defconfig
rk3126_defconfig
rk3128_defconfig
rk3368_defconfig
rk3399Pro_defconfig

rk3288_box_defconfig
rk3128_box_defconfig
rk3036_box_defconfig
rk3368_box_defconfig
rk322x_box_defconfig
rk3399_box_defconfig
```

4.3 固件生成

Rockchip 平台 Loader 分为一级模式和二级模式，根据不同的平台配置生成相应的 Loader 固件。通过宏 CONFIG_SECOND_LEVEL_BOOTLOADER 定义二级 Loader 模式。

4.3.1 一级 Loader 模式

U-BOOT 作为一级 Loader 模式，那么仅支持 EMMC 存储设备，编译完成后生成的镜像：

```
rk3399pro_loader_v1.15.115.bin
```

其中 V1.15.115 是发布的版本号。

4.3.2 二级 Loader 模式

U-Boot 作为二级 Loader 模式，那么固件支持所有的存储设备，该模式下，需要 MiniLoader 支持，通过宏 CONFIG_MERGER_MINILOADER 进行配置生成。同时引入 Arm Trusted Firmware 后会生成 trust image，这个通过宏 CONFIG_MERGER_TRUSTIMAGE 进行配置生成。

以 rk3399pro 编译生成的镜像为例：

```
rk3399pro_loader_v1.15.115.bin
uboot.img
trust.img
```

其中 v1.15.115 是发布的版本号，rockchip 定义 U-Boot loader 的版本，其中 1.15.115 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 U-Boot 作为二级 loader 的打包。

RK3036、RK3126、RK3128、RK322x、RK3368、RK3366、RK3399、RK3399Pro 等采用二级 loader 模式。

4.4 U-Boot 编译

RK3399Pro SDK 编译使用的是如下配置：

```
./make.sh rk3399pro
```

编译完，会生成 trust.img、rk3399pro_loader_v1.15.115.bin、uboot.img 三个文件。

目前编译出来的 rk3399pro_loader_v1.15.115.bin DDR 为定频 800MHz 版本。

4.5 U-Boot 充电相关配置

4.5.1 充电图片打包

充电图片需要打包进 resource.img 才能被充电驱动读取并且显示。编译内核时默认不会打包充电图片，所以需要另

外单独把这些图片打包进 resource.img。

打包命令:

```
./pack_resource.sh <input resource.img>
```

这个命令默认会把./tools/images/目录里的图片作为充电图片打包进 resource.img，新的 resource.img 会生成在 UBoot 根目录下，烧写的时候请烧写这个新的 resource.img。

如下是打包时的提示信息:

```
./pack_resource.sh /home/guest/3399/kernel/resource.img
Pack ./tools/images/ & /home/guest/3399/kernel/resource.img to resource.img
...
Unpacking old image(/home/guest/3399/kernel/resource.img):
rk-kernel.dtb logo.bmp logo_kernel.bmp
Pack to resource.img succeeded!
Packed resources:
rk-kernel.dtb battery_1.bmp battery_2.bmp battery_3.bmp battery_4.bmp battery_5.bmp
battery_fail.bmp logo.bmp logo_kernel.bmp battery_0.bmp
resource.img is packed ready
```

4.5.2 DTS 使能充电

默认代码已经使能了该驱动，通过在 dts 里增加并且使能 charge-animation 节点即可使能充电动画的功能。

```
charge-animation {
compatible = "rockchip,uboot-charge";
status = "okay";
rockchip,uboot-charge-on = <0>; // 是否在 U-Boot 进行充电
rockchip,android-charge-on = <1>; // 是否在 Android 进行充电
rockchip,uboot-exit-charge-level = <5>; // U-Boot 充电时，允许开机的最低电量
rockchip,uboot-exit-charge-voltage = <3650>; // U-Boot 充电时，允许开机的最低电压
rockchip,screen-on-voltage = <3400>; // U-Boot 充电时，允许点亮屏幕的最低电压
```

rockchip,uboot-low-power-voltage = <3350>; // U-Boot无条件强制进入充电模式的最低电压

rockchip,system-suspend = <1>; // 灭屏时进入trust 进行低功耗待机

rockchip,auto-off-screen-interval = <20>; // 亮屏超时后自动灭屏，单位秒。(如果没有这个属性，则默认 15s)

rockchip,auto-wakeup-interval = <10>; // 休眠自动唤醒时间，单位秒。(如果值为 0 或没有这个属性，则禁止休眠自动唤醒)

rockchip,auto-wakeup-screen-invert = <1>; // 休眠自动唤醒的时候，是否让屏幕产生亮/灭效果};

自动休眠唤醒功能的作用：

1. 考虑到有些电量计（比如 adc）需要定时更新软件算法，否则会造成电量统计不准，因此不能让 cpu一直处于休眠状态；
2. 方便进行休眠唤醒的压力测试；

4.5.3 低功耗休眠

进入充电流程后可通过短按 power 实现系统亮灭屏，灭屏时进入低功耗待机状态，再次按下按键可唤醒。非低电状态下，长按 power 可退出充电流程进行开机。

4.5.4 更换充电图片

1. 更换./tools/images/目录下的图片，图片采用8bit 或 24bit bmp 格式。使用命令“ls | sort”确认图片排列顺序是低电量到高电量，在使用pack_resource.sh 脚本打包时，所有图片会按照这个顺序被打包进 resource;

2. 修改./drivers/power/charge_animation.c里的图片和电量关系信息：

name: 图片的名字；

soc: 图片对应的电量；

period: 图片刷新时间（单位：ms）；

****注意： ****最后一张图片一定要是 failed 的图片，且“soc=-1”不可改变。

3. 执行pack_resource.sh 打包命令获取新的 resource.img 即可；

5 Kernel 开发

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。RK3399Pro kernel 版本是 4.4，config 配置文件统一为 arch/arm64/configs/rockchip_defconfig，RK3399Pro 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

RK3399Pro 的 dts 文件在 kernel/arch/arm64/boot/dts/rockchip/ 下，其中 rk3399pro.dtsi 是核心配置文件定义了平台相关的内容；RK3399-android.dtsi 是产品级配置文件定义了一些外围设备；具体的产品 dts 需要 include 这两个文件，如 RK3399Pro evb 的 dts 文件 rk3399pro-evb-v10.dts。产品的 dts 里面根据具体的产品需求配置 CPU、GPU、DDR 的频率和电压表；配置 io、屏、wifi、bt、sensor、温控、背光、电池、系统供电配置等等。

5.1.2 新增一个产品 DTS

Rk3399Pro 的产品 dts 文件需放在 kernel/arch/arm64/boot/dts/rockchip/ 下。

1、以 rk3399pro-evb-v10.dts 为参照，拷贝一份 dts 文件命名为 rk3399Pro-product.dts。

2、修改 arch/arm64/boot/dts/rockchip/Makefile 文件，添加对应 dtb 声明：

```
+rk3399Pro-product.dtb
```

3、修改编译脚本或编译命令。

4、重新编译内核。

5.2 WiFi 配置

```
wireless-wlan {  
    compatible = "wlan-platdata";  
    rockchip,grf = <&grf>;  
    wifi_chip_type = "ap6354";  
    sdio_vref = <1800>;  
    WIFI,host_wake_irq = <&gpio0 3 GPIO_ACTIVE_HIGH>; /* GPIO0_a3 */  
    status = "okay";  
}
```

```
};/
```

上面部分内容是 WiFi 的 dts 配置内容，主要包括电源控制、中断等功能脚的配置。下面将对各个配置项（一般客户只需要修改下面红色标出部分参数）的功能进行详细描述：

```
wifi_chip_type = " ap6354";
```

用来确认 WiFi 芯片型号，实际使用什么型号的 WiFi 需要在这里指定：

```
sdio_vref = <1800>; //1800mv or 3300mv
```

这个配置项配置 WiFi 模组的 IO 参考电压值，根据实际硬件设计中提供给 WiFi 模组参考电压输入的电压值来进行设定，参考电压设置错误会导致 WiFi 通信异常，引起 WiFi 打不开或者工作不稳定。

```
WIFI,host_wake_irq = <&gpio0 3 GPIO_ACTIVE_HIGH>;
```

这个配置项是 WiFi 中断脚的配置，某些 WiFi 模组没有这个脚可以不用配置直接将此配置项注释掉。使用 Broadcom 的 WiFi，比如 AP6xxx 以及 RK90x 等模组都需要正确配置这 GPIO。

Broadcom wifi AP6xxx 系统会使用此中断脚作为 WiFi 数据中断脚，此中断脚有异常将会导致 WiFi 无法正常工作。其它 WiFi，例如 RTL8723BS，在机器进入休眠时，如果有 WiFi 数据到来时此中断用来唤醒机器。此中断脚有异常并不会造成 WiFi 无法正常工作。

5.3 BT 配置

```
wireless-bluetooth {  
    compatible = "bluetooth-platdata";  
    //wifi-bt-power-toggle;  
    uart_rts_gpios = <&gpio2 19 GPIO_ACTIVE_LOW>; /* GPIO2_C3 */  
    pinctrl-names = "default", "rts_gpio";  
    pinctrl-0 = <&uart0_rts>;  
    pinctrl-1 = <&uart0_gpios>;  
    //BT,power_gpio = <&gpio3 19 GPIO_ACTIVE_HIGH>; /* GPIOx_xx */  
    BT,reset_gpio = <&gpio0 9 GPIO_ACTIVE_HIGH>; /* GPIO0_B1 */  
    BT,wake_gpio = <&gpio2 26 GPIO_ACTIVE_HIGH>; /* GPIO2_D2 */  
    BT,wake_host_irq = <&gpio0 4 GPIO_ACTIVE_HIGH>; /* GPIO0_A4 */  
    status = "okay";  
};
```

以上是 BT 在 dts 里面的配置，下面对常见可能需要修改的部分进行简单的说明：

```
BT,reset_gpio = <&gpio0 9 GPIO_ACTIVE_HIGH>;
```

这个配置项是关于 BT 的 RESET 脚配置，这个脚不同的 BT 模组不一定都有，具体以实际原理图为准。

```
BT,power_gpio = <&gpio3 19 GPIO_ACTIVE_HIGH>
```

这个配置项是关于 BT 的电源控制 GPIO 配置，高电平有效，具体以实际原理图为准。

```
BT,wake_gpio = <&gpio2 26 GPIO_ACTIVE_HIGH>;
```

这个配置项是关于 BT 的 WAKE 脚配置，对应原理图中的 BT_WAKE 管脚，高电平有效。

```
BT,wake_host_irq = <&gpio0 4 GPIO_ACTIVE_HIGH>
```

这个配置项是关于 BT 的中断脚配置，对应原理图中的 BT_HOST_WAKE 管脚，高电平有效。

默认 BT 使用 uart0 接口连接，uart0 的配置如下：

```
&uart0 {  
    pinctrl-names = "default";  
    pinctrl-0 = <&uart0_xfer &uart0_cts>;  
    status = "okay";  
};
```

5.4 GPIO

RK3399Pro 提供 5 组 GPIO(GPIO0~GPIO4)共 122 个，所有的 GPIO 都可以用作中断，GPIO0/GPIO1 可以作为系统唤醒脚，所有 GPIO 都可以软件配置为上拉或者下拉，所有 GPIO 默认为输入，GPIO 的驱动能力软件可以配置。

关于原理图上的 **gpio** 跟 **dts** 里面的 **gpio** 的对应关系，例如 GPIO4c0，那么对应的 dts 里面应该是“gpio4 16”。因为 GPIO4A 有 8 个 pin，GPIO4B 也有 8 个 pin，以此计算可得 c0 口就是 16，c1 口就是 17，以此类推；

GPIO 的使用请参考 RKDocs\common\driver\目录下《Rockchip Pin-Ctrl 开发指南 V1.0-20160725.pdf》。

5.5 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。

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/driver/目录下《Rockchip CPU-Freq 开发指南 V1.0.1-20170213.pdf》和《Rockchip DEVFreq 开发指南 V1.0-20160701.pdf》文档。

A53/A72/GPU/DDR 分别有对应的调试接口，可以通过 ADB 命令进行操作，对应的接口目录如下：

A53: /sys/devices/system/cpu/cpu0/cpufreq/

A72: /sys/devices/system/cpu/cpu4/cpufreq/

GPU: /sys/class/devfreq/ff9a0000.gpu/

DDR: /sys/class/devfreq/dmc/

这些目录下有如下类似节点：

- **available_frequencies**: 显示支持的频率
- **available_governors**: 显示支持的变频策略
- **cur_freq**: 显示当前频率
- **Governor**: 显示当前的变频策略
- **max_freq**: 显示当前最高能跑的频率
- **min_freq**: 显示当前最低能跑的频率

以 GPU 为例进行定频操作，流程如下：

- 查看支持哪些频率

```
cat /sys/class/devfreq/ff9a0000.gpu/available_frequencies
```

- 切换变频策略

```
echo userspace > /sys/class/devfreq/ff9a0000.gpu/governor
```

- 定频

```
echo 400000000 > /sys/class/devfreq/ff9a0000.gpu/userspace/set_freq
```

- 设置完后，查看当前频率

```
cat /sys/class/devfreq/ff9a0000.gpu/cur_freq
```

5.6 温控配置

RK3399Pro 芯片的 ARM 核和 GPU 核分别带有温控传感器，可以实时监控 cpu 和 gpu 的温度，并通过算法来控制 cpu 和 gpu 的频率从而控制 cpu 和 gpu 的温度。每个产品的硬件设计和模具不同对应的散热情况也不同，可以通过 dts 中的如下配置进行适当的调整温控参数来适配产品：

设置温控开启的温度：

```
&threshold {  
    temperature = <85000>; /* millicelsius */  
};
```

设置温控上限温度：

```
&target {  
    temperature = <100000>; /* millicelsius */  
};
```

设置软件关机温度：

```
&soc_crit {  
    temperature = <105000>; /* millicelsius */  
};
```

配置硬件关机温度：

```
&tsadc {  
    rockchip,hw-tshut-mode = <1>; /* tshut mode 0:CRU 1:GPIO */  
    rockchip,hw-tshut-polarity = <1>; /* tshut polarity 0:LOW 1:HIGh */  
    rockchip,hw-tshut-temp = <110000>;  
    status = "okay";  
};
```

温控的具体说明可以参考 RKDocs\common\driver 目录下《Rockchip Thermal 开发指南 V1.0.1-20170428.pdf》。

5.7 LPDDR4 配置

rk3399Pro 使用 lpddr4 的 dts 配置请参考文件：arch/arm64/boot/dts/rockchip/rk3399-pro-evb-lp4-v11-avb.dts，将该文件中的下述 3 个节点拷贝到对应的产品 dts 中即可：

```
&dfi {
```

```

status = "okay";

};

&dmc {
    status = "okay";
    center-supply = <&vdd_center>;//这里需要客户根据实际硬件电路来配置
    upthreshold = <40>;
    downthreshold = <20>;
    system-status-freq = <
        /*system status      freq(KHz)*/
        SYS_STATUS_NORMAL      856000
        SYS_STATUS_REBOOT      416000
        SYS_STATUS_SUSPEND     416000
        SYS_STATUS_VIDEO_1080P 416000
        SYS_STATUS_VIDEO_4K     856000
        SYS_STATUS_VIDEO_4K_10B 856000
        SYS_STATUS_PERFORMANCE 856000
        SYS_STATUS_BOOST        856000
        SYS_STATUS_DUALVIEW     856000
        SYS_STATUS_ISP          856000
    >;
    vop-pn-msch-readlatency = <
        /* plane_number  readlatency */
        0  0
        4  0x20
    >;
    vop-bw-dmc-freq = <
        /* min_bw(MB/s) max_bw(MB/s) freq(KHz) */
        763    1893    416000
        3013    99999    856000
    >;

```

```

    auto-min-freq = <416000>;
};

&dmc_opp_table {
    compatible = "operating-points-v2";

    opp-200000000 {
        opp-hz = /bits/ 64 <200000000>;
        opp-microvolt = <900000>;
        status = "disabled";
    };

    opp-300000000 {
        opp-hz = /bits/ 64 <300000000>;
        opp-microvolt = <900000>;
        status = "disabled";
    };

    opp-400000000 {
        opp-hz = /bits/ 64 <400000000>;
        opp-microvolt = <900000>;
        status = "disabled";
    };

    opp-416000000 {
        opp-hz = /bits/ 64 <416000000>;
        opp-microvolt = <900000>;
    };

    opp-528000000 {
        opp-hz = /bits/ 64 <528000000>;
        opp-microvolt = <900000>;
        status = "disabled";
    };
};

```

```

opp-600000000 {
    opp-hz = /bits/ 64 <600000000>;
    opp-microvolt = <900000>;
    status = "disabled";
};

opp-800000000 {
    opp-hz = /bits/ 64 <800000000>;
    opp-microvolt = <900000>;
    status = "disabled";
};

opp-856000000 {
    opp-hz = /bits/ 64 <856000000>;
    opp-microvolt = <900000>;
};

opp-928000000 {
    opp-hz = /bits/ 64 <928000000>;
    opp-microvolt = <900000>;
    status = "disabled";
};

opp-1056000000 {
    opp-hz = /bits/ 64 <1056000000>;
    opp-microvolt = <900000>;
    status = "disabled";
};
};

```

这里需要注意的是，1) **lpddr4** 我们只支持 **416M** 和 **856M** 两档频率，其他频率被 **disabled** 掉了，所以如果客户要使用同一个 **dts** 来支持 **lpddr4** 和其他类型的 **ddr**，则其他类型的 **ddr** 也将只有 **416M** 和 **856M** 的频率，这个请务必注意；2) 以上配置默认开启 **DDR** 变频功能。**lpddr4** 的变频功能对声卡的数量有所限制，说明如下：

如果 lpddr4 需要变频功能,则需要将音频 buffer 移到 sram 中,RK3399Pro 的 sram 空间有限,可用空间 128k,目前预分配给单个音频流的空间为 32k,所以系统支持的上限声卡数最多只能 2 个(32k * 2 * 2,每个声卡包含 playback 和 capture),更多的声卡无法创建成功,除非减小单个流的预分配大小,但这也相对的减小了底下支持的 buffer size max,如果用户层使用声卡想设置更大 buffer 时将受限。需注意,USB 声卡由于未使用 dma,所以不在限制范围内,也就是说,可以有 2 个声卡(包含 hdmi、spdif、i2s 等接口的声卡)加上多个 usb 声卡。因此,接下来分成两种情况描述:

5.7.1 需要 lpddr4 的变频

如果需要 lpddr4 变频,则需要将音频 buffer 移到 sram 中,此时系统最多只能支持 2 个声卡,请按照如下方法进行配置:

1. dts 中添加 sram 节点

```
/* first 64k(0xff8c0000~0xff8d0000) for ddr and suspend */
iram: sram@ff8d0000 {
    compatible = "mmio-sram";
    reg = <0x0 0xff8d0000 0x0 0x20000>; /* 128k */
};
```

2. 相对应的产品 dts 中引用 iram 节点。

```
&dmac_bus {
    iram = <&iram>;
    rockchip,force-iram;
};
```

5.7.2 不需要 lpddr4 变频

由于 lpddr4 变频有 2 个声卡的限制,因此如果需要 3 个以上声卡,需要关闭 lpddr4 的变频,即在对应产品的 dts 中将 dmc 节点 disable,如下所示:

```
&dmc {
    status = "disabled";
    ... ..
};
```

```
};
```

另外，需要确保在内核中删除掉 5.8.1 节中描述的 2 个配置：

1. 删除 dts 中的如下配置：

```
/* first 64k(0xff8c0000~0xff8d0000) for ddr and suspend */
iram: sram@ff8d0000 {
    compatible = "mmio-sram";
    reg = <0x0 0xff8d0000 0x0 0x20000>; /* 128k */
};
```

2. 删除 dts 中的如下配置：

```
&dmac_bus {
    iram = <&iram>;
    rockchip,force-iram;
};
```

5.8 SDCard 配置

Uart debug 与 sdcard 复用，默认配置是打开 debug，如果要使用 sdcard 需要如下配置：

```
&fiq_debugger {
+    status = "disabled";
    pinctrl-0 = <&uart2a_xfer>;
};
&sdmmc {
    sd-uhs-sdr12;
    sd-uhs-sdr25;
    sd-uhs-sdr50;
    sd-uhs-sdr104;
+    status = "okay";
};
```

6 Android 常见配置

6.1 Android 产品配置

6.1.1 lunch 选项说明

rk3399pro-userdebug: //rk3399Pro 平台产品 userdebug (64 位)

rk3399pro-user: //rk3399Pro 平台产品 user (64 位)

6.2 常用功能配置说明

6.2.1 常用配置宏说明

宏配置	功能说明
BUILD_WITH_GOOGLE_MARKET	若为 true 则集成 GMS 包, false 不集成
BUILD_WITH_GOOGLE_MARKET_ALL	若为 true 集成 full 的 GMS 包, false 集成 mini 的 GMS 包
BUILD_WITH_GOOGLE_FRP	使能恢复出厂设置保护 FRP 功能
BUILD_WITH_FORCEENCRYPT	使能默认全盘加密
PRODUCT_SYSTEM_VERITY	使能 Verified boot
BUILD_WITH_GMS_CER	GMS 认证配置选项
BUILD_WITH_WIDEVINE	集成 Widevine level3 插件库
BOARD_NFC_SUPPORT	使能 NFC 功能
BOARD_SENSOR_ST	选用 ST 的 sensor 框架
BOARD_SENSOR_MPU	选用 MPU 的 sensor 框架
BOARD_SENSOR_MPU_VR	选用 MPU_VR 的 sensor 框架
BOARD_GRAVITY_SENSOR_SUPPORT	使能 G-Sensor
BOARD_COMPASS_SENSOR_SUPPORT	使能 Compass
BOARD_GYROSCOPE_SENSOR_SUPPORT	使能陀螺仪 Gyroscope
BOARD_PROXIMITY_SENSOR_SUPPORT	使能距离感应器
BOARD_LIGHT_SENSOR_SUPPORT	使能光感应器
BOARD_PRESSURE_SENSOR_SUPPORT	使能压力感应器
BOARD_TEMPERATURE_SENSOR_SUPPORT	使能温度传感器
BOARD_ENABLE_3G_DONGLE	使能 3G Dongle 功能

TARGET_ROCKCHIP_PCBATEST	使能 PCBA 测试
BOOT_SHUTDOWN_ANIMATION_RINGING	使能开关机动画+铃声
BOARD_SYSTEMIMAGE_PARTITION_SIZE	System 分区最大容量

6.2.2 预装 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制进 Android 系统。预安装分为不可卸载安装、可永久卸载安装以及卸载后恢复出厂设置后自动恢复安装，详细配置和使用请参阅工程目录 RKDocs/android/下相关说明文档：

《Android8.0_预安装应用功能说明文档_V1.0_20171109.pdf》。

6.2.3 开/关机动画及铃声

定制 Android8.1 的开机铃声，关机铃声，开机动画，关机动画的详细方法请参阅工程目录 RKDocs/android/下的说明文档：《Android8.0_定制开关机动画（铃音）说明_V1.0_20170923.pdf》。

6.3 Parameter 说明

rk3399Pro Android 8.1 平台有不同产品形态，不同的产品形态可能需要不同的 parameter 参数，关于 parameter 中各个参数、分区情况细节，请参考 \RKDocs\common\RKTools manuals\ Rockchip Parameter File Format Ver1.3.pdf。

6.4 新增分区配置

请参考 \RKDocs\android\《Android 增加一个分区配置指南 V1.00.pdf》。

6.5 OTA 升级

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 Recovery 用户操作指南 V1.03》。

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

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、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设备相连。

```
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 默认未对网络 ADB 端口进行配置，需要手动修改打开配置。

修改 device/rockchip/rkxxxx/device.mk 文件，在 PRODUCT_PROPERTY_OVERRIDES 后面追加如下配置：

```
service.adb.tcp.port=5555
```

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，如果可以的话，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。

7.2.3 查看上次 log

可以加-L 参数来打印出上次系统复位前的 **logcat** 信息。若出现拷机异常或者异常掉电的情况，可通过该命令打印出上一次 **Android** 运行状态的日志。命令如下：

```
adb shell
logcat -L
```

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）
- 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。

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

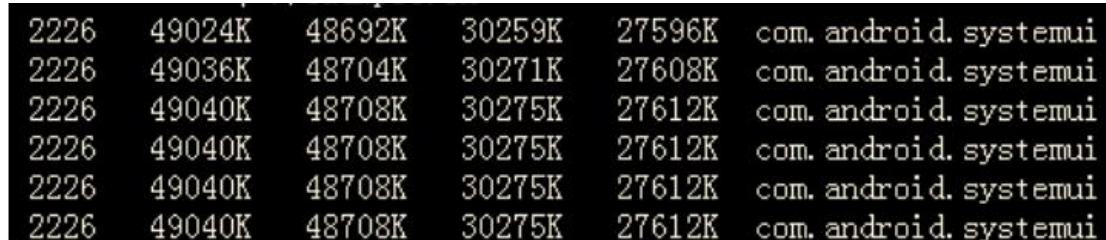
通过跟踪内存的占用状态，进而分析进程中是否存在内存泄露场景。使用编写脚本的方式，连续输出进程的内存快照，通过对比 USS 段，可以了解到此进程是否内存泄露。

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。

7.5 串口调试

7.5.1 串口配置

调试过程中最方便的就是串口的输入输出, 这里需要注意的是 RK3399 波特率设置为 1500000。RTS/CTS 不要勾选, 否则串口无法输入。

7.5.2 FIQ 模式

快速中断请求 (Fast Interrupt Request, FIQ) 在 ARM 中, FIQ 模式是特权模式中的一种, 同时也属于异常模式一类。

RK 平台上, 在串口输入 “fiq”, 可以进入该模式。此时会有使用帮助跳出, 可根据情况进行

一些调试。经常在死机，或系统卡死的时候起作用。

7.6 音频 codec 问题调试工具及文档

请参考 RKDocs\common\driver\ Rockchip Audio 开发指南 V1.1-20170215-linux4.4.pdf。

7.7 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>;
};
```

- 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
cat ftrace-ramoops-0

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\ RK

Tools 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\common\RKTools manuals\Rockchip PCBA 模块开发指南--20170210.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。

本平台 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

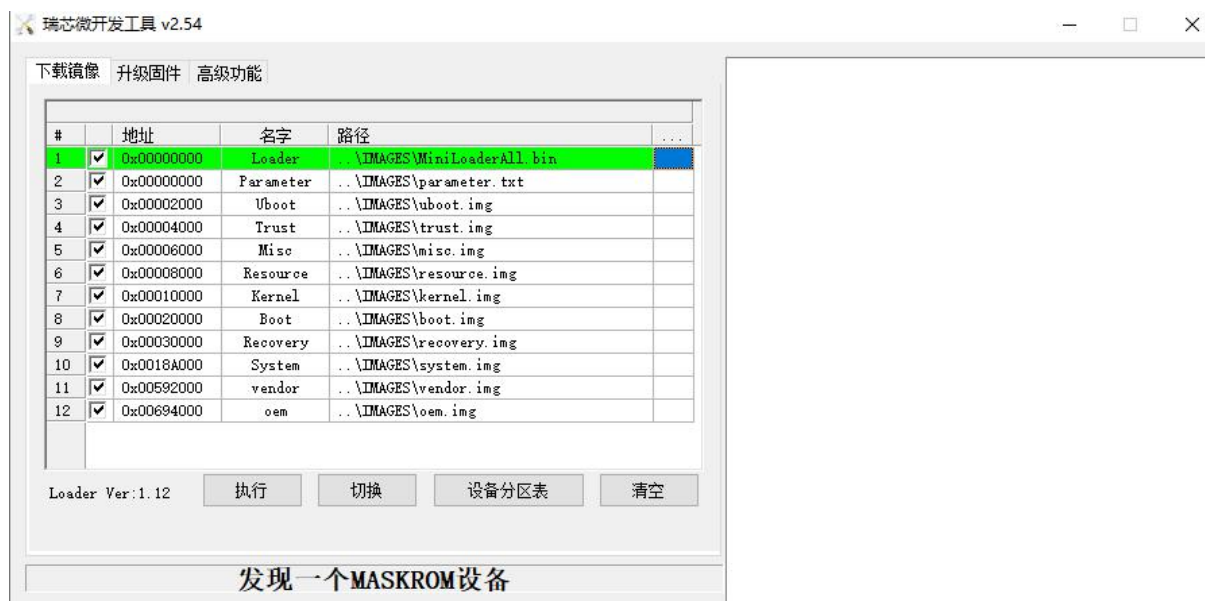


图 8-1 Android 开发工具下载镜像

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 recovery 按键约 3-4s 时间进入。

2) 打开工具，点击“下载镜像”菜单。单击每一行末尾红色箭头所指处，会弹出文件选择框。选择对应分区的 img 文件路径。

3) 依次设置所有 img 文件的路径。

4) 配置完成后，点击“执行”。右侧信息框将显示相关信息。

5) 按钮说明

“低格”按钮：用于擦除设备

“清空”按钮：清空信息框

8.4.2 升级固件



图 8-2 Android 开发工具升级固件

1) 准备目标固件。（可参考 [update.img 打包](#)）

2) 确认设备已经进入下载模式。

下载模式进入方法：先按住开发板 **reset** 按键，再长按 **recovery** 按键约 3-4s 时间进入。

3) 点击“固件”按钮，选择目标固件 **update.img** 文件。

4) 点击“升级”按钮进行下载。右侧信息框将显示相关信息。

8.4.3 高级功能

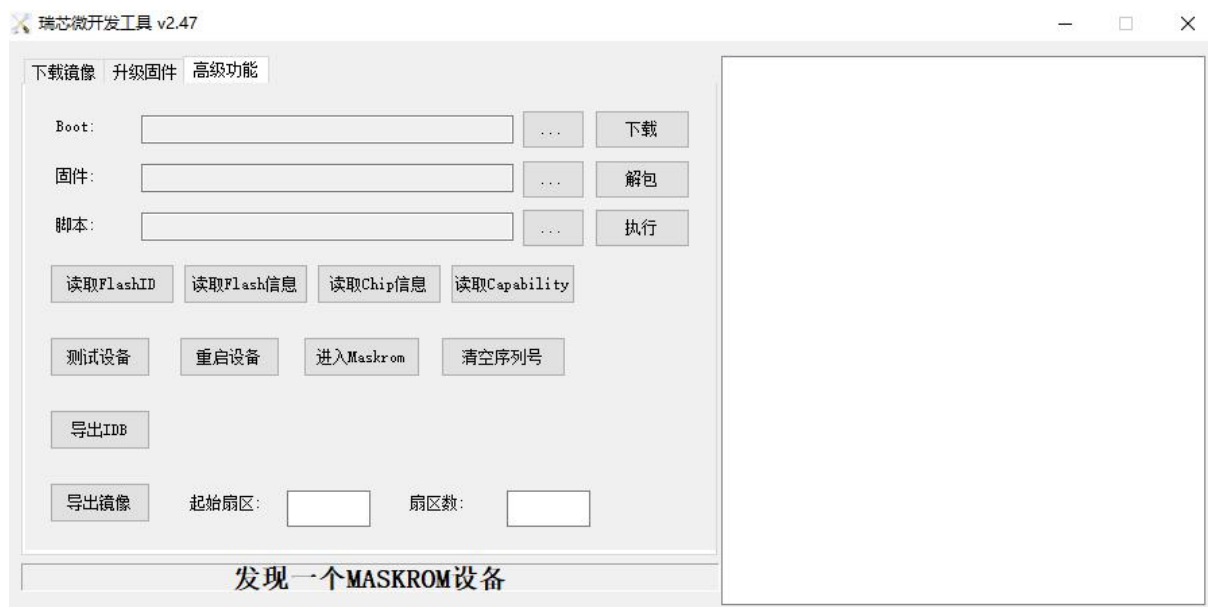


图 8-3 Android 开发工具高级功能

高级功能说明：

- 1) Boot 只能选择打包好的 update.img 文件或是 loader 文件。
- 2) 固件必须使用打包后的 update.img。
- 3) 解包功能可将 update.img 拆解为各部分镜像文件。

8.5 update.img 打包

本平台支持将各零散镜像文件，打包成一个完整的 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。
- 2) 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。
- 3) 编辑 mkupdate.bat。
- 4) 修改 loader 名称为实际存放的 loader 名称。
- 5) 点击 mkupdate.bat 运行，结束后会在该目录生成一个 update.img。

8.6 固件签名工具

参考 RKTools\windows\SecureBootTool_v1.83_foruser.rar 中的《Rockchip Secure Boot Application Note》

8.7 序列号/Mac/厂商信息烧写-RKDevInfoWriteTool 工具

本平台使用 RKDevInfoWriteTool 工具进行序列号/Mac/厂商信息的烧写。以下说明该工具的基本用法。

8.7.1 使用 RKDevInfoWriteTool 写入

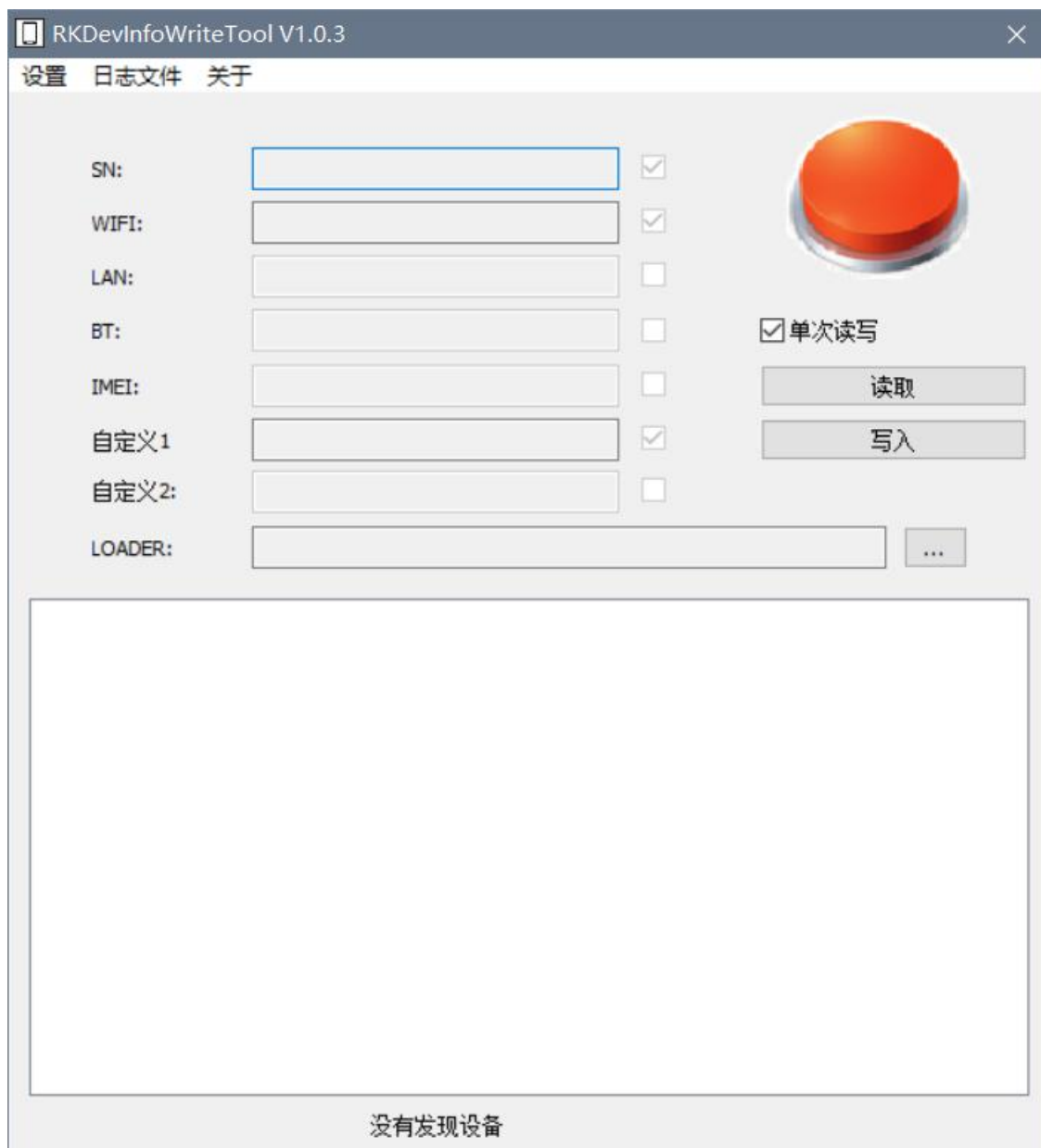


图 8-4WNpctool 工具

- 1) 进入 loader 模式。
- 2) 点击“设置”菜单，弹出设置窗口，用来设置 SN/WIFI/LAN/BT/IMEI

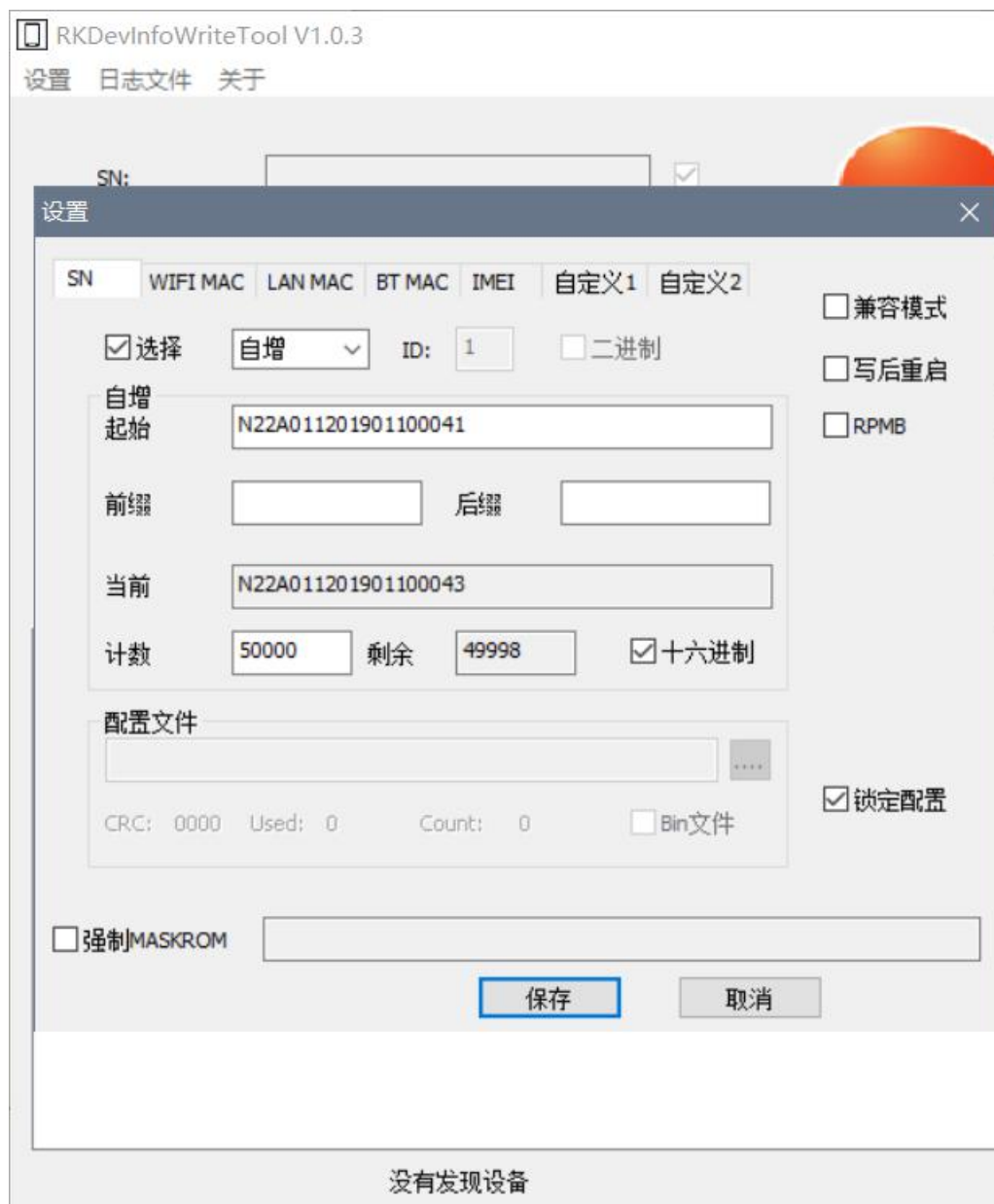


图 8-5 工具模式设置

- 3) 设置完成后，点击“保存”按钮，关闭模式设置窗口，返回主窗口。
- 4) 点击“写入”按钮即可。

8.7.2 使用 RKDevInfoWriteTool 读取

- 1) 进入 loader 模式。
- 2) 点击“读取”按钮即可。

8.8 量产工具使用

8.8.1 工具下载步骤

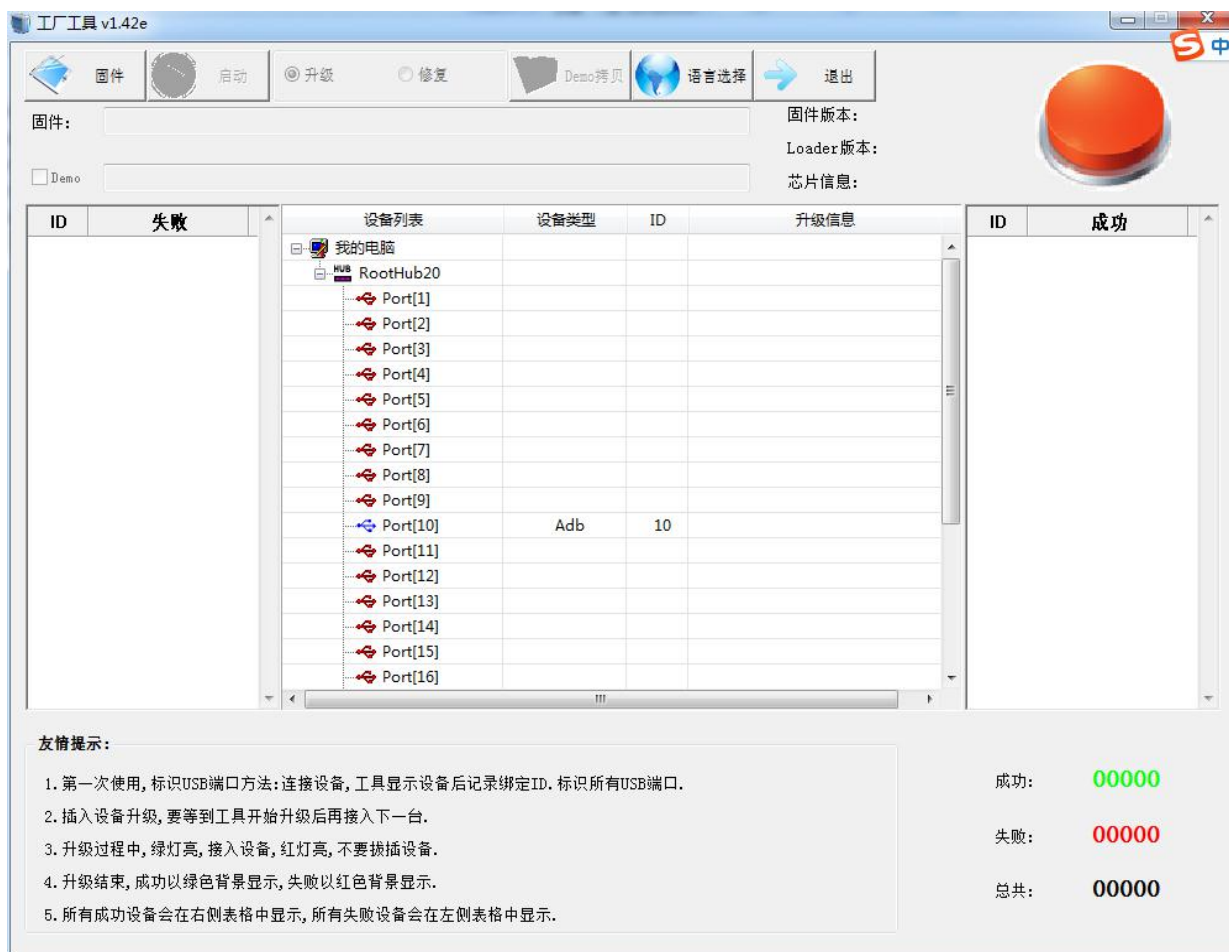


图 8-8 量产工具

- 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) 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。
- 3) 可同时连接多台设备, 进行一拖多烧写, 提高工厂烧写效率。