

密级状态：绝密( )      秘密( )      内部( )      公开(√ )

# RK3399\_Android7.1\_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：	V2.08
	作 者：	郝小伟、刘益星、张文平
	完成日期：	2019-3-25
	审 核：	陈海燕、黄祖芳、邓训金
	完成日期：	2019-3-25

福州瑞芯微电子股份有限公司

Fuzhou Rockchips Semiconductor 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	郝小伟	2017.01.16	正式发布	
V2.00	刘益星	2018.03.12	修改适配最新代码 增加双屏异显、异触说明 增加 HDMI IN 功能说明 增加系统调试章节 常用工具详细说明	
V2.01	张文平	2018-4-18	增加 lpddr4 配置、深度学习、多路 camera、音视频多路编解码、audio 3A 算法等章节	
V2.02	张文平	2018-5-14	根据代码的修改，更新 lpddr4 的配置说明	
V2.03	张文平	2018-7-5	添加新的功能支持，包括手写优化、log 系统、梯形校正、SSD、widevine 等	
V2.04	张文平	2018-8-20	1. 新增 next-dev 分支 u-boot 支持, 详情请参考 4.7 节 2. 新增双屏异触功能说明	
V2.05	张文平	2018-9-27	新增显示参数的调整和保存功能，详情请参考 6.20 节。	
V2.06	张文平	2018-11-7	1. 新增多屏拼接功能说明，详见 6.7。 2. 更新显示参数的调整和保存说明文档，详见 6.21。	

			<ul style="list-style-type: none"> <li>3. 新增自动化测试和 debug 脚本说明, 详见 6.22。</li> <li>4. 新增文件系统切换为 EXT4 功能说明, 详见 6.23。</li> <li>5. 新增 DATA 分区加密功能使能和禁用说明, 详见 6.24。</li> <li>6. 新增 rk3399 性能说明文档, 详见 6.25。</li> </ul>	
V2.07	张文平	2019-1-29	<ul style="list-style-type: none"> <li>1. 更正行业 sdk 下载说明。</li> <li>2. 更新 8.4 节 DDR 测试工具说明。</li> <li>3. 更新性能开发文档名称</li> <li>4. 增加 rk3399K 芯片支持, 请参考 6.26 节。</li> </ul>	
V2.08	张文平/魏建兴	2019-3-29	<ul style="list-style-type: none"> <li>1. 根据最新代码, 更新主副屏旋转功能文档。</li> <li>2. 新增常见问题章节, 描述客户常见软硬件问题, 请参考第 9 章。</li> <li>3. 新增禁用串口打印功能说明, 详见 6.27 节。</li> <li>4. 更新 6.23 节, 加入 data 分区切换为 EXT4 的一些问题说明。</li> </ul>	

			5. 新增 dp 音频配置说明，请参考 6.28 节进行配置。	
--	--	--	---------------------------------	--

## 目 录

<b>RK3399_Android7.1_软件开发指南</b> .....	1
前 言 .....	1
1 支持列表.....	2
1.1 DDR支持列表.....	2
1.2 EMMC支持列表 .....	2
1.2.1 高性能EMMC颗粒的选取 .....	2
1.3 WiFi/BT支持列表 .....	3
1.4 SDK软件包适用硬件列表 .....	3
1.5 多媒体编解码支持列表 .....	4
2 文档/工具索引 .....	5
2.1 文档索引.....	5
2.2 工具索引.....	8
3 SDK编译/烧写.....	10
3.1 SDK获取 .....	10
3.1.1 SDK下载链接 .....	10
3.1.2 repo .....	10
3.1.3 SDK代码压缩包.....	10
3.2 SDK编译 .....	11
3.2.1 JDK安装 .....	11
3.2.2 编译模式 .....	11
3.2.3 挖掘机编译 .....	11
3.2.4 固件生成步骤 .....	12
3.2.5 jack-server配置 .....	12
3.2.6 全自动编译脚本 .....	14
3.3 固件烧写.....	15
3.4 量产烧写.....	16
4 U-Boot开发.....	16
4.1 Rockchip U-Boot简介.....	16

4.2	平台配置 .....	16
4.3	固件生成 .....	17
4.3.1	一级Loader模式 .....	17
4.3.2	二级Loader模式 .....	17
4.4	U-Boot编译 .....	18
4.5	U-Boot充电相关配置 .....	18
4.5.1	低电预充 .....	18
4.5.2	u-boot充电图标显示 .....	19
4.6	U-Boot logo相关的配置 .....	19
4.6.1	U-Boot logo开关配置 .....	19
4.6.2	U-Boot logo图片更换 .....	19
4.7	Next-dev分支U-Boot .....	19
4.7.1	Next-dev分支u-boot简介 .....	19
4.7.2	Next-dev分支u-boot开发 .....	20
5	Kernel开发 .....	21
5.1	DTS介绍 .....	21
5.1.1	DTS说明 .....	21
5.1.2	新增一个产品DTS .....	21
5.2	USB配置 .....	22
5.3	WiFi配置 .....	23
5.4	BT配置 .....	23
5.5	GPIO .....	24
5.6	ARM、GPU、DDR频率修改 .....	25
5.7	温控配置 .....	26
5.8	LPDDR4 配置 .....	27
5.8.1	需要lpddr4 的变频 .....	29
5.8.2	不需要lpddr4 变频 .....	29
6	Android常见配置 .....	30
6.1	Android产品配置 .....	30
6.1.1	lunch选项说明 .....	30

6.1.2	添加一个新的产品 .....	30
6.2	常用功能配置说明 .....	31
6.2.1	常用配置宏说明 .....	31
6.2.2	预装APK .....	32
6.2.3	开/关机动画及铃声 .....	32
6.3	Parameter说明 .....	33
6.4	新增分区配置 .....	33
6.5	OTA升级.....	33
6.6	双屏异显/异触功能.....	33
6.6.1	功能描述 .....	33
6.6.2	RK DUALSCREEN配置方法.....	34
6.6.3	相关代码位置.....	34
6.6.4	异触功能 .....	34
6.7	多屏拼接功能 .....	35
6.7.1	功能描述 .....	35
6.7.2	相关代码位置.....	35
6.7.3	固定拼接配置方法 .....	35
6.7.4	APK动态调整拼接配置方法 .....	36
6.8	HDMI IN功能 .....	38
6.8.1	功能描述 .....	38
6.8.2	功能要求 .....	38
6.8.3	相关代码位置.....	38
6.9	主副屏旋转功能 .....	38
6.10	深度学习.....	38
6.11	多路camera支持.....	39
6.12	音视频多路编解码.....	39
6.13	Audio 3A算法.....	39
6.14	双wifi (station+ap) .....	39
6.15	双以太网.....	39
6.16	手写优化.....	40

6.17	梯形校正功能 .....	40
6.18	基于深度学习的目标检测（SSD）方案 .....	40
6.19	Widevine补丁 .....	40
6.20	高可靠OTA .....	40
6.21	显示参数的调整和保存 .....	41
6.22	自动化测试和debug脚本 .....	41
6.23	DATA分区文件系统切换为EXT4 .....	43
6.23.1	注意事项 .....	43
6.23.2	Recovery擦除慢问题 .....	44
6.23.3	F2FS切换为EXT4 方法 .....	45
6.24	Data分区使能和禁用加密功能 .....	47
6.25	RK3399 性能优化方法 .....	49
6.26	RK3399K芯片支持 .....	49
6.27	禁用串口打印功能 .....	49
6.28	DP音频功能 .....	51
7	系统调试 .....	53
7.1	ADB工具 .....	53
7.1.1	概述 .....	53
7.1.2	USB ADB使用说明 .....	53
7.1.3	网络ADB使用要求 .....	54
7.1.4	SDK网络ADB端口配置 .....	54
7.1.5	网络ADB使用 .....	54
7.1.6	手动修改网络ADB端口号 .....	55
7.1.7	ADB常用命令详解 .....	55
7.2	Logcat工具 .....	57
7.2.1	Logcat命令使用 .....	57
7.2.2	常用的日志过滤方式 .....	57
7.2.3	查看上次log .....	58
7.3	Procrank工具 .....	58
7.3.1	使用procrank .....	58



7.3.2	检索指定内容信息 .....	59
7.3.3	跟踪进程内存状态 .....	59
7.4	Dumpsys工具 .....	60
7.4.1	使用Dumpsys .....	60
7.5	串口调试 .....	61
7.5.1	串口配置 .....	61
7.5.2	FIQ模式 .....	61
7.6	音频codec问题调试工具及文档 .....	61
7.7	Last log开启 .....	61
7.8	Log自动保存系统 .....	62
7.8.1	使用方法: .....	62
7.8.2	对log的说明 .....	64
8	常用工具说明 .....	64
8.1	StressTest .....	65
8.2	PCBA测试工具 .....	65
8.3	DeviceTest .....	66
8.4	DDR测试工具 .....	67
8.5	Android开发工具 .....	67
8.5.1	下载镜像 .....	67
8.5.2	升级固件 .....	68
8.5.3	高级功能 .....	69
8.6	update.img打包 .....	69
8.7	固件签名工具 .....	70
8.8	序列号/Mac/厂商信息烧写-WNpctool工具 .....	70
8.8.1	使用WNpctool写入 .....	70
8.8.2	使用WNpctool读取 .....	71
8.9	OemTool打包工具 .....	71
8.9.1	Oem打包工具步骤 .....	71
8.10	量产工具使用 .....	73
8.10.1	工具下载步骤 .....	73

9	常见问题分析 .....	73
9.1	硬件设计对应软件修改 .....	74
9.1.1	如何选择dts配置.....	74
9.1.2	IO Domain配置 .....	74
9.1.3	Vdd_log电压差异 .....	76
9.2	常见稳定性问题分析 .....	80
9.2.1	软件BUG导致死机或者卡死问题 .....	80
9.2.2	硬件相关导致的软件死机问题.....	81
9.3	Secure Boot常见问题 .....	82
9.4	Data分区切换为EXT4 后出现的问题 .....	82
9.5	更新代码后必现卡在android动画问题.....	82
9.6	USB3.0 外设待机唤醒后重新枚举问题 .....	83
9.7	LPDDR4 开启负载变频.....	84

## 前 言

### 概述

本文档主要介绍 Rockchip RK3399 Android7.1 软件开发指南，旨在帮助软件开发工程师更快上手 RK3399 的开发及调试。

### 产品版本

芯片名称	内核版本	Android 版本
RK3399	Linux4.4	Android7.1.1

### 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

RK3399 DDR 目前选型列表支持双通道 DDR3、DDR3L、LPDDR3、LPDDR4。

表 1-1 RK3399 DRAM Support Type

Chip	DRAM Support Type
RK3399	DDR3/DDR3L/LPDDR3/LPDDR4

RK3399 DDR 颗粒支持程度列表，详见 RKDocs\common\Platform support lists 目录下《RK DDR Support List Ver2.31》，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 RK3399 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

RK3399 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 Ver1.38\_2018\_01\_22》，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 RK3399 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

RK3399 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\_20171215.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。

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 Android7.1 64bit 系统，适配瑞芯微 RK3399 芯片的软件包，适用于 laptop 产品形态、Tablet 产品形态、sapphire（蓝宝石）& excavator（挖掘机）开发板、及其他基于 RK3399 平台开发的产品。

使用的是 sapphire（蓝宝石）& excavator（挖掘机）开发板的，kernel 配置可直接使用

rk3399-sapphire-excavator-edp.dts 进行配置。

参考《RK3399\_VR&Tablet\_V10\_20160620》硬件设计（可以通过 FAE 窗口获取）的 TABLET 样机，kernel 配置可以参考：

TABLET:rk3399-mid-818-android.dts

另外随 SDK 发布，附带了 Box 样机板，sapphire（蓝宝石） & excavator（挖掘机）开发板的硬件使用说明。

## 1.5 多媒体编解码支持列表

RK3399 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\rk3399 目录下《RK3399 Multimedia Codec Benchmark v1.0》。

## 2 文档/工具索引

### 2.1 文档索引

RK3399 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。

RK3399 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），rk3399(3399 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
  - | |—— Android 增加一个分区配置指南 V1.00.pdf
  - | |—— project.config
  - | |—— Rockchip\_android7.1\_wifi\_配置明 V1.5.pdf
  - | |—— ROCKCHIP\_PCBA 测试工具开发指南\_V1.1\_20171222.pdf
  - | |—— Rockchip Recovery 用户操作指南 V1.03.pdf
- |—— common
  - | |—— camera
    - | | |—— Camera\_for\_RockChipSDK 参考说明\_v4.1.pdf
    - | | |—— CIF\_ISP10\_Driver\_User\_Manual\_V1.0.pdf
    - | | |—— CIF\_ISP11\_Driver\_User\_Manual\_v1.0.pdf
    - | | |—— RK\_CIF10\_User\_Manual\_V2.0.pdf
    - | | |—— RK\_CIF11\_User\_Manual\_V2.0.pdf
    - | | |—— RK\_ISP10\_Camera\_User\_Manual\_v2.1.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
  - | |—— 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\_V1.3\_20171209.pdf
- | | | — Rockchip 基于 DRM 框架的 HDMI 开发指南 v1.0-20171214.pdf
- | | — driver
- | | | — 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 DEVFreq 开发指南 V1.0-20160701.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 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 Thermal 开发指南 V1.0.1-20170428.pdf
- | | | — Rockchip Vendor Storage Application Note.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
- | |—— Platform support lists
- | | |—— RK3036 Multimedia Codec Benchmark v1.2.pdf
- | | |—— RK3126B Multimedia Codec Benchmark v1.1.pdf
- | | |—— RK3126C Multimedia Codec Benchmark v1.0.pdf
- | | |—— RK3128 BOX Hardware Design Guide V10-201410.pdf
- | | |—— RK312x Multimedia Codec Benchmark v1.1.pdf
- | | |—— RK3288 Multimedia Codec Benchmark v1.8.pdf
- | | |—— RK3368 Multimedia Codec Benchmark v1.3.pdf
- | | |—— RK DDR Support List Ver2.29.pdf
- | | |—— RKeMMCSupportList Ver1.38\_2018\_01\_22.pdf
- | | |—— RKISPV11\_Camera\_Module\_AVL\_v1.5.pdf
- | | |—— RKISPV1\_Camera\_Module\_AVL\_v1.7.pdf
- | | |—— RKNandFlashSupportList Ver2.72\_2016\_08\_30.pdf
- | | |—— Rockchip Kodi 支持程度列表\_V2.0\_20170715.pdf
- | | |—— Rockchip\_WiFi\_Situation\_20171215.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 Box 厂测工具操作说明 V2.0.pdf
- | | |—— Rockchip Parameter File Format Ver1.3.pdf
- | | |—— Rockchip 量产烧录 指南 V1.1-20170214.pdf
- | | |—— WNPctool 简要使用说明\_V1.1.0\_0920.pdf

- | | |—— 压力测试 Stresstest 文档 forVR\_ver3.0.pdf
- | | |—— 瑞芯微 attestation\_keybox 烧录指南\_v1.0\_20171212.pdf
- | | |—— 量产工具升级及相关问题处理.pdf
- | |—— security
- | | |—— Rockchip\_Secure\_Boot\_Application\_Note\_V1.2.1\_20171128.pdf
- | | |—— Rockchip\_TEE 安全 SDK 开发手册\_V1.1\_20170516.pdf
- | |—— u-boot
- | | |—— Rockchip-Developer-Guide-Trust.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
- | |—— wifi
- | |—— RealTek wifi 驱动移植说明\_V1.1.pdf
- | |—— ROCKCHIP\_ANDROID\_8.1\_WIFI 配置说明\_V1.2.pdf
- |—— rk3399
  - |—— RK3399\_Android7.1\_CaffeOnACL\_开发说明文档\_V1.0\_20180225.pdf
  - |—— RK3399 Android 7.1 TABLET 软件开发指南 V1.00-2017-01-16.pdf
  - |—— RK3399\_BOX 双屏异显音频说明\_V1.1\_20171220.pdf
  - |—— RK3399 Multimedia Codec Benchmark v1.0.pdf
  - |—— RK3399\_SDK 多媒体性能指标说明文档\_V1.0\_20180109.pdf

## 2.2 工具索引

RK3399 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。

RK3399 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）。

### RKTools

- |—— linux

```

|   |—— Linux_Pack_Firmware (Linux 固件打包工具)
|   |—— Linux_SecureBoot (Linux 固件签名工具)
|   |—— Linux_Upgrade_Tool (Linux 开发工具)
|   |—— windows
|       |—— AndroidTool (开发工具)
|       |   |—— AndroidTool_Release_v2.38
|       |   |—— rockdev (固件打包工具)
|       |—— DriverAssitant_v4.5 (驱动安装助手)
|       |—— Efuse_Tool_V1.36 (Efuse 烧写工具)
|       |—— FactoryTool-v1.42e.rar (工厂量产工具)
|       |—— FWFactoryTool-5.3.rar (固件工厂工具)
|       |—— OemTool_v1.2.rar (Demo 镜像制作工具)
|       |—— SD_Firmware_Tool._v1.46.zip (SD 卡升级固件制作工具)
|       |—— SecureBootTool_v1.83_foruser.rar (固件签名工具)
|       |—— SpiImageTools_v1.36.zip
|       |—— UpgradeDIITool_v1.35.zip (厂商信息烧写工具—待更新版本)

```

## 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《RK3399\_ANDROID7.1-TABLET-SDK\_V1.00 发布说明.pdf》。

#### 3.1.1 SDK下载链接

RK3399\_ANDROID7.1-Industry-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/rk3399-n-all/manifests.git -m rk3399_all_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 Rk3399\_Android7.1-Tablet-SDK.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 rk3399
tar zxvf rk3399_android7.1_Industry_v1.0.tar.gz -C rk3399
cd rk3399
.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

Android7.1 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 挖掘机编译

uboot 编译：

```
cd u-boot
make rk3399_defconfig
make ARCH=aarch64
```

kernel 编译：

```
cd kernel
make ARCH=arm64 rockchip_defconfig -j8
make ARCH=arm64 rk3399-sapphire-excavator-edp.img -j12
```

android 编译：

```
source build/envsetup.sh
lunch rk3399_all-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  
├── misc.img  
├── parameter.txt  
├── recovery.img  
├── resource.img  
├── RK3399MiniLoaderAll.bin  
├── system.img  
├── trust.img  
└── uboot.img
```

### 3.2.5 jack-server 配置

Android7.1 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/RK3399/build-rk3399-all.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/rk3399" name="rk/device/rockchip/rk3399"
remote="rk" revision="rk32/mid/7.0/develop">
```

```
<copyfile src="buildspec.mk" dest="buildspec.mk"/>
```

```
<copyfile src="build-rk3399-all.sh" dest="build-rk3399-all.sh"/>
```

```
</project>
```

- 3) 修改 `build-rk3399-all.sh` 脚本中的特定变量以编出对应产品固件。

```
KERNEL_DTS=rk3399-sapphire-excavator-edp
```

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 使用 build-rk3399-all.sh 编译时，请先进行下面两步操作：

```
source build/envsetup.sh  
lunch rk3399_all-userdebug
```

也可在脚本中对应修改，可改为 rk3399\_all-user 及其它配置：

```
lunch rk3399_all-userdebug
```

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，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/RK3399 \*\*\*\*\*\_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 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。

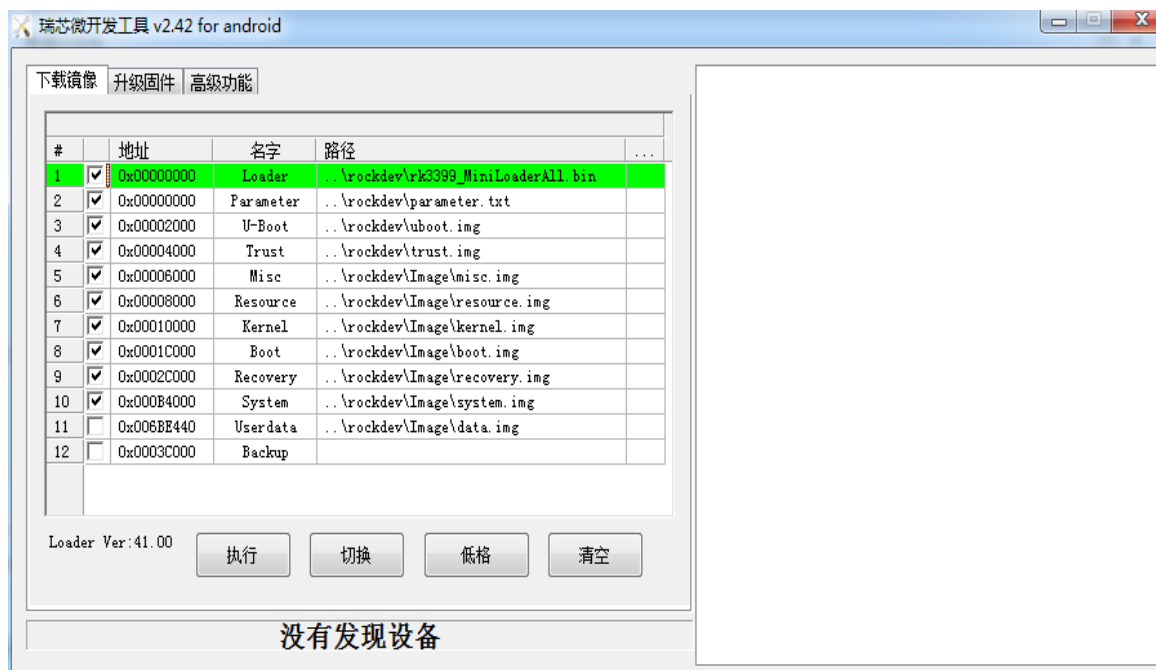


图 3-1 Android 开发工具烧写界面

注：烧写前，需安装最新的 USB 驱动，驱动详见：

```
RKTools/windows/  
|—— DriverAssitant_v4.5
```

### 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 U-Boot 开发指南 V3.8-20170214.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、rk322x、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
```

```
rk3399_defconfig
```

```
rk3288_box_defconfig
```

```
rk3128_box_defconfig
```

```
rk3036_box_defconfig
```

```
rk3368_box_defconfig
```

```
rk322x_box_defconfig
```

```
rk3399_box_defconfig
```

RK3399 Laptop/Tablet 选用的是 rk3399\_defconfig 配置。

## 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：

```
rk3399_loader_v1.09.110.bin
```

其中 V1.09.110 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 rk3399 编译生成的镜像为例：

```
rk3399_loader_v1.09.110.bin
```

```
uboot.img
```

```
trust.img
```

其中 V1.09.110 是发布的版本号，rockchip 定义 U-Boot loader 的版本，其中 1.09.110 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 等采用二级 loader 模式。

## 4.4 U-Boot编译

RK3399 SDK 编译使用的是如下配置：

```
make rk3399_defconfig  
make ARCHV=aarch64
```

编译完，会生成 trust.img、rk3399\_loader\_v1.09.110.bin、uboot.img 三个文件。

目前编译出来的 rk3399\_loader\_v1.09.110.bin DDR 为定频 800Mhz 版本。

## 4.5 U-Boot充电相关配置

### 4.5.1 低电预充

u-boot 支持低电预充，需要在 uboot/include/configs/rk33plat.h 中打开如下开关，默认该功能是关闭的。

```
#define CONFIG_UBOOT_CHARGE  
  
#define CONFIG_SCREEN_ON_VOL_THRESD 3400//3.4v  
  
#define CONFIG_SYSTEM_ON_VOL_THRESD 3500//3.5v
```

其中 CONFIG\_SCREEN\_ON\_VOL\_THRESD 是系统点亮屏幕的电压门限，低于这个电压，禁止系统亮屏。CONFIG\_SYSTEM\_ON\_VOL\_THRESD 是系统正常启动的电压门限，低于这个电压，禁止 uboot 启动内核。这两个电压可以根据具体的产品设计灵活调整。

产品板级 dts 中如下节点进行充电模式开关配置，可以灵活配置使用 uboot 还是 Android 的关机充电模式：

```
uboot-charge {  
    compatible = "rockchip,uboot-charge";  
    rockchip,uboot-charge-on = <0>;  
    rockchip,android-charge-on = <1>;  
};
```

rockchip,uboot-charge-on 开关 uboot 阶段的充电动画，rockchip,android-charge-on 开关 android 充电动画。

### 4.5.2 u-boot充电图标显示

如果选择使用 u-boot 阶段的充电动画，即 `rockchip,uboot-charge-on = <1>` 时，还需要将动画图片资源文件打包在 `resource.img`，充电图标打包可以参考文档 `RKDocs\common\u-boot\Rockchip U-Boot 开发指南 V3.8-20170214.pdf` 8.1.1 章节，具体打包方法如下：

- 进到 u-boot 目录下；

拷贝充电图片到 `tools\resource_tool\resources\images`，这里面默认有充电图片，替换可以参考这里面的图片格式进行替换；

- 执行如下打包命令：

```
sudo ./tools/resource_tool/pack_resource.sh tools/resource_tool/resources/ ../kernel/resource.img resource.img tools/resource_tool/resource_tool
```

执行后会将 `tools/resource_tool/resources/` 目录下的动画图片资源打包在 `../kernel/resource.img`，生成新的 `resource.img` 在 u-boot 根目录。

## 4.6 U-Boot logo相关的配置

### 4.6.1 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：

```
rockchip,uboot-logo-on = <1>;
```

如果需要关闭这个功能，请在 kernel 的 dts 中设置 `rockchip,uboot-logo-on = <0>`；即可。

### 4.6.2 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 `resource.img` 即可。

附：不一定要两张图片，可以只要一张，如果只有一张就保留 `logo.bmp` 这一张即可。

## 4.7 Next-dev分支U-Boot

### 4.7.1 Next-dev分支u-boot简介

前面几节介绍基于 `rkdevelop` 分支的 U-BOOT，但是有些客户对于 u-boot 有一些特殊的需求，例如 u 盘读写、有线网卡等功能，这些功能在 `rkdevelop` 上不支持，因此我们在 u-boot 代码上新开发了一个分支，称为 `next-dev`，下面简单介绍下 `next-dev` 分支的 u-boot 相关功能。

`next-dev` 是从 U-Boot 官方的 `v2017.09` 正式版本中切出来进行开发的版本。随着 upstream 的

U-Boot功能越来越完善, 以及我们实际产品上对U-Boot的需求更加多样, 因此我们进行了nextdev开发。next-dev作为rkdevelop的升级版本, 可以支持更多的功能, 并且同U-Boot官方版本会每隔一段时间进行同步。目前在平台上已经支持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。另外, 我们提供了一份说明文档描述next-dev在开发中容易碰到的问题以及配置, 请参考[RKDocs/common/u-boot/ Rockchip-Developer-Guide-UBoot-nextdev.pdf](#)。

#### 4.7.2 Next-dev分支u-boot开发

“RK3399\_ANDROID7.1-Industry-SDK\_V1.0”该 SDK 支持 next-dev 分支的 u-boot, 代码路径和原来一样, 但是客户默认同步下来是 rkdevelop 分支的 u-boot, 如果需要切到 next-dev 分支, 请确保 sdk 已经更新到 v1.4 版本以上, 可按如下方法确认当前 sdk 版本:

```
ls .repo/manifests/ -l
```

```
total 4
drwxrwxr-x 2 zwp zwp 4096 Aug 20 17:35 rk3399_all_release
lrwxrwxrwx 1 zwp zwp 47 Aug 20 17:36 rk3399_all_release.xml ->
rk3399_all_release/rk3399_all_v1.4_20180820.xml
```

并且按照如下方法切到 next-dev 分支：

```
cd u-boot
git fetch rk
git checkout -f rk/next-dev-release -b next-dev-release
```

编译方法为：

```
./make.sh rk3399
```

最终固件的生成还是用 android 目录下的 mkimage.sh 来生成，另外，假如客户使用 build-rk3399-all.sh 脚本来编译，请记得修改脚本里面 u-boot 对应的编译命令。

## 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。RK3399 kernel 版本是 4.4，config 配置文件统一为 arch/arm64/configs/rockchip\_defconfig，RK3399 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

RK3399 的 dts 文件在 kernel/arch/arm64/boot/dts/rockchip/下，其中 rk3399.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 Tablet 产品的 dts 文件 rk3399-mid-818-android.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。

挖掘机采用 rk3399-sapphire-excavator-edp.dts 这个文件。

#### 5.1.2 新增一个产品DTS

Rk3399 的产品 dts 文件需放在 kernel/arch/arm64/boot/dts/rockchip/下。

1、以 rk3399-sapphire-excavator-edp.dts 或 rk3399-mid-818-android.dts 为参照，

拷贝一份 dts 文件命名为 rk3399-product.dts。

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

```
+rk3399-product.dtb
```

3、修改编译脚本或编译命令。

4、重新编译内核。

## 5.2 USB配置

RK3399 Type-c 模块需要外挂一个 fusb302 逻辑检测芯片来识别接入设备类型及 USB 的正反插。Fusb302 的软件驱动在 dts 里面的配置如下：

```
fusb0: fusb30x@22 {  
    compatible = "fairchild,fusb302";  
    reg = <0x22>;  
    pinctrl-names = "default";  
    pinctrl-0 = <&fusb0_int>;  
    int-n-gpios = <&gpio1 1 GPIO_ACTIVE_HIGH>;  
    status = "okay";  
};  
  
中断脚配置：  
&pinctrl {  
    fusb30x {  
        fusb0_int: fusb0-int {  
            rockchip,pins = <1 2 RK_FUNC_GPIO &pcfg_pull_up>;  
        };  
    };  
}
```

在 fusb302 及 usb phy 检测区分接入 type-c 口的是哪一类设备（充电器、USB、OTG、DP 等）之后，通知系统，所以相关联的模块代码需要注册 fusb302 的 extcon notifier 来接收，需要在模块 dts 配置加入 extcon = <&fusb0>。

如 rk818 dts 节点中加入 extcon = <&fusb0>，通过 fusb302 及 usb phy 检测区分充电器、USB、OTG 的拔插后，rk818 模块决定相关的充电电流配置及 OTG 的开关。

目前 sdk 参考 dts 中默认 enable 了 fusb302 的配置，如果产品未使用 type-c 接口、未使用



fusb302, 产品 dts 中请 disabled 节点 tcphy0 及 fusb0, 并将 USB 相关联的模块 dts 中 extcon = <&fusb0>改为 extcon = <&u2phy0>。

### 5.3 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。

### 5.4 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";  
};
```

## 5.5 GPIO

RK3399 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 **dtb** 里面的 **gpio** 的对应关系，例如 GPIO4c0，那么对应的 dtb 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》。

## 5.6 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
```

## 5.7 温控配置

RK3399 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》。

## 5.8 LPDDR4 配置

rk3399 使用 lpddr4 的 dts 配置请参考文件：  
arch/arm64/boot/dts/rockchip/rk3399-evb-rev3-android-lp4.dts，将该文件中的下述 3 个节点拷贝到对应的产品 dts 中（从行业 SDK v2.1 版本开始，默认开启了 lpddr4 的负载变频）：

```
&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
    system-status-freq = <
        /*system status      freq(KHz)*/
        SYS_STATUS_NORMAL    800000
        SYS_STATUS_REBOOT    400000
        SYS_STATUS_SUSPEND    400000
        SYS_STATUS_VIDEO_1080P 400000
        SYS_STATUS_VIDEO_4K    800000
        SYS_STATUS_VIDEO_4K_10B 800000
        SYS_STATUS_PERFORMANCE 800000
        SYS_STATUS_BOOST       400000
        SYS_STATUS_DUALVIEW    800000
        SYS_STATUS_ISP         800000
    >;
    auto-min-freq = <400000>;
    auto-freq-en = <1>;//从 V2.1 版本开始默认开启了 LPDDR4 的负载变频
};

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

    opp-2000000000 {
```

```

        opp-hz = /bits/ 64 <200000000>;
        opp-microvolt = <825000>;
        status = "disabled";
    };
    opp-300000000 {
        opp-hz = /bits/ 64 <300000000>;
        opp-microvolt = <850000>;
        status = "disabled";
    };
    opp-400000000 {
        opp-hz = /bits/ 64 <400000000>;
        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>;
    };
    opp-928000000 {
        opp-hz = /bits/ 64 <928000000>;
        opp-microvolt = <900000>;
        status = "disabled";
    };
    opp-1056000000 {
        opp-hz = /bits/ 64 <1056000000>;
        opp-microvolt = <900000>;
        status = "disabled";
    };
};

```

另外，请确认下面这个配置项已经在内核使能（SDK v1.1 版本已经使能）：

**CONFIG\_SND\_SOC\_ROCKCHIP\_FORCE\_SRAM=y**

这里需要注意的是：

1) lpddr4 我们只支持 400M 和 800M 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 400M 和 800M 的频率，这个请务必注意；

2) 以上配置默认开启 ddr 变频，lpddr4 的变频功能对声卡数量有所限制，说明如下：

如果要支持 Lpddr4 变频，则需要将音频 buffer 移到 sram 中，RK3399 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：

### 5.8.1 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;
};
```

### 5.8.2 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;  
};
```

## 6 Android常见配置

### 6.1 Android产品配置

#### 6.1.1 lunch选项说明

rk3399\_all-userdebug: //rk3399 平台平板产品 userdebug（64 位）

rk3399\_all-user: //rk3399 平台平板产品 user（64 位）

#### 6.1.2 添加一个新的产品

rk3399 平台支持平板、Laptop、Box 等产品形态，当需要添加一个新的产品时，可以基于已有的 rk3399\_all 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：

- 新增文件夹 device/rockchip/rk3399/rk3399\_xxx，基于 rk3399\_all.mk 创建 rk3399\_xxx.mk，将 rk3399\_all 目录下的所有文件拷贝至 rk3399\_xxx 目录下。



```
cd device/rockchip/rk3399
mkdir rk3399_xxx
cp rk3399_all.mk ./rk3399_xxx.mk
cp rk3399_all/* rk3399_xxx/
```

- 在 device/rockchip/rk3399/AndroidProducts.mk 中添加:

```
PRODUCT_MAKEFILES := \
    $(LOCAL_DIR)/rk3399.mk \
    $(LOCAL_DIR)/rk3399_all.mk \
    $(LOCAL_DIR)/rk3399_xxx.mk
```

- 在 vendorsetup.sh 中添加产品对应的 lunch 选项:

```
add_lunch_combo rk3399_all-userdebug
add_lunch_combo rk3399_all-user
add_lunch_combo rk3399_xxx-userdebug
add_lunch_combo rk3399_xxx-user
```

- 修改 rk3399\_ xxx.mk 及 rk3399\_xxx 目录下的新产品所需要修改的配置。

## 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
BOARD_USE_AUDIO_3A	是否使能 AUDIO 3A 算法

### 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：

- 新增文件夹 device/rockchip/rk3399/rk3399\_all/preinstall\_del，要确认是在 TARGET\_DEVICE\_DIR 定义的目录，get\_build\_var TARGET\_DEVICE\_DIR 可以看到。
- 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。

编译结束后会在 out/target/product/rk3399\_all/system/目录，生成 preinstall\_del 文件夹，文件夹内包含了预制的第三方应用。烧录后，系统会自动安装这些应用到 data/app 目录。因此他们是可卸载的。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。

### 6.2.3 开/关机动画及铃声

需要在产品的 makefile 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。

将开机铃声 复制到 device/rockchip/common/startup.wav (源码路径)

将关机铃声 复制到 device/rockchip/common/startup.wav (源码路径)

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)

### 6.3 Parameter说明

rk3399 Android 7.1 平台有平板、Box、Laptop 等产品形态，不同的产品形态可能需要不同的 parameter 参数，请参考 device/rockchip/rk3399/下子目录 rk3399\_all 中的来相应修改配置，关于 parameter 中各个参数、分区情况细节，请参考\RKDocs\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》。

### 6.6 双屏异显/异触功能

#### 6.6.1 功能描述

RockChip（以下简称 RK） SDK 平台上支持接双显示设备异显功能，包含 Android Presentation 和 RK Dualscreen 两种方案。

**Android Presentation** 是 Google 提供的双屏方案，实现了 View 级别的 VOP 派发，逻辑均在同一个 APP 上进行控制，请参考 google 官方文档进行开发。

**RK dualscreen** 则是实现了 APP 级别的 VOP 派发，异显的两部分分别是不同的 APP。

Presentation 适用于对自身需求进行深入定制的方案，目前的双屏异触功能是基于此方案适配的；RK dualscreen 在满足深入定制方案下，支持快速集成多方 APP，进行功能整合，目前双屏异显双声功能是基于此方案适配，请参考文档：《RKDocs/rk3399/ RK3399\_BOX 双屏异显音频说明\_V1.1\_20171220.pdf》。客户请根据产品需求选择对应方案，下表为两个方案的简单说明。

方案	双屏异触	双屏异声	控制逻辑
<b>RK dualscreen</b>	不支持	支持	两个独立的 app 显示

			在不同屏上
<b>Android Presentation</b>	支持	不支持	一个 app 控制不同 View 显示在不同屏上

### 6.6.2 RK DUALSCREEN配置方法

代码默认已集成此功能，在系统 设置-显示-HDMI-双屏异显 中打开：



默认同时按音量“+”和音量“-”两个按键触发异显/同显模式切换；

组合按键可通过设置属性 `sys.dual_screen.keycodes` 修改配置，如：

音量加减：`sys.dual_screen.keycodes=24,25` 其中 24,25 分别为按键的 android 键值。

### 6.6.3 相关代码位置

```
frameworks/base
frameworks/native
packages/apps/Settings
```

### 6.6.4 异触功能

双屏异触功能，请参考如下路径的补丁以及说明文档：

[RKDocs/rk3399/patches/双屏异触 Patch\\_V1.0-20170724.zip](#)。

## 6.7 多屏拼接功能

### 6.7.1 功能描述

本 SDK 提供横向和纵向多屏拼接方案，在不超过 vop 尺寸的前提下，支持 vopb 和 vopl 之间任意比例拼接，方便客户根据产品需求进行选择。多屏拼接功能可以用于体感游戏显示设备、健身房、广告机等多显示场景。

### 6.7.2 相关代码位置

```
device/rockchip/rk3399
hardware/rockchip/hwcomposer
hardware/rockchip/libgralloc
```

请确认代码已经更新到 v1.7 版本，可在 .repo/repo/repo sync 后，按照如下方法确认当前版本：

```
$ ls .repo/manifests/rk3399_all_release.xml -l
lrwxrwxrwx 1 zwq zwq 47 Oct 22 11:52 .repo/manifests/rk3399_all_release.xml -> rk3399_all_release/rk3399_all_v1.7_20181016.xml
```

### 6.7.3 固定拼接配置方法

使用过程中客户不需要动态调整拼接比例和拼接方式，可以把拼接比例和拼接方式固定到系统固件里面。固定拼接配置：

device/rockchip/rk3399/rk3399\_all.mk 文件中打开拼接功能，make installclean 后重新编译固件。

1. BOARD\_MULTISCREEN\_SPLICING := true      多屏拼接功能开关
2. BOARD\_MULTISCREEN\_SPLICING\_MODE := 0    0 表示左右拼接，1 表示上下拼接
3. 多屏拼接相关属性配置

```
#three 1080p screen splicing(1:1:1)
PRODUCT_PROPERTY_OVERRIDES += \
persist.sys.framebuffer.main=5760x1080@60 \
persist.sys.framebuffer.aux=5760x1080@60 \
```

```
persist.sys.dualModeRatioPri=2 \
```

```
persist.sys.dualModeRatioAux=1
```

- 1) `persist.sys.framebuffer.main` 是主屏 UI 布局的分辨率，`persist.sys.framebuffer.aux` 是副屏 UI 布局的分辨率。多屏拼接功能里这两个属性值必须一样，否则无法实现多屏拼接功能。例如 3 个屏 1920x1080 分辨率（左右拼接），为了每个屏显示效果好，推荐设为 5760x1080@60。
- 2) `persist.sys.dualModeRatioPri` 是主屏占的拼接比例，`persist.sys.dualModeRatioAux` 是副屏占的拼接比例。通常三个 1080p 的物理屏拼接，两个 mipi 屏在 vopb 上，hdmi 显示设备在 vopl 上。设置 mipi 为主屏，hdmi 设置为副屏，当配置 `persist.sys.dualModeRatioPri=2` 和 `persist.sys.dualModeRatioAux=1`（主屏：副屏=2:1）时，由于两个 mipi 屏是等分输出，所以三个屏的拼接比例为 1:1:1。

**注意：**受 vop 的尺寸限制，主副屏 UI 布局尺寸不能超过 vop 的尺寸。

	VOPB	VOPL
Max in/out	4096x2304/4096x2160	2048x1536/2048x1536

例如：主副屏的 UI 布局分辨率设置为 5760x1080@60，且主副屏比例设为 3:1，那么主屏的宽为  $5760 \times 3/4 = 4320$ ，超过了 vopb 的最大宽 4096，因此主副屏不能设置为 3:1。

下面举个固定三等分屏左右拼接的例子，对应配置如下：

```
# multi-screen splicing  
BOARD_MULTISCREEN_SPLICING := false  
BOARD_MULTISCREEN_SPLICING_MODE := 0  
#three 1080p screen splicing(1:1:1)  
PRODUCT_PROPERTY_OVERRIDES += \  
persist.sys.framebuffer.main=5760x1080@60 \  
persist.sys.framebuffer.aux=5760x1080@60 \  
persist.sys.dualModeRatioPri=2 \  
persist.sys.dualModeRatioAux=1
```

#### 6.7.4 APK动态调整拼接配置方法

在 `device/rockchip/rk3399/rk3399_all.mk` 文件中打开拼接功能，然后 `make installclean` 后重新编译固件。

1. BOARD\_MULTISCREEN\_SPLICING : = true          拼接功能开关
2. 动态设置拼接屏需要改动的 property 属性介绍:
  - 1) persist.sys.dualModeEnable 1          拼接使能开关
  - 2) sys.hwc.compose\_policy 0          关闭 hwc, 用 gpu 合成
  - 3) persist.sys.dualModeTB 1          拼接模式选择 0 是左右拼接, 1 是上下拼接
  - 4) persist.sys.framebuffer.main          主屏 UI 布局的分辨率
  - 5) persist.sys.framebuffer.aux          副屏 UI 布局的分辨率
  - 6) persist.sys.dualModeRatioPri 2          主屏占总的比例
  - 7) persist.sys.dualModeRatioAux 1          副屏占总的比例

动态配置属性时, 需要注意以下:

1. 拼接屏要求主副屏向上申请的 framebuffer size 一致, 所以需要在切换分辨率的同时, 设置 framebuffer size, 具体命令如下:

```
setprop persist.sys.framebuffer.main 5760x1080@60
```

```
setprop persist.sys.framebuffer.aux 5760x1080@60
```

例如:

3 个 1080p 左右拼接, 设置为:

```
setprop persist.sys.framebuffer.main 5760x1080@60
```

```
setprop persist.sys.framebuffer.aux 5760x1080@60
```

3 个 1080p 上下拼接, 设置为:

```
setprop persist.sys.framebuffer.main 1920x3240@60
```

```
setprop persist.sys.framebuffer.aux 1920x3240@60
```

## 2. 拼接比例

```
persist.sys.dualModeRatioPri=2
```

```
persist.sys.dualModeRatioAux=1
```

主屏: 副屏 = 2:1

注意: 受 vop 的尺寸限制, 显示尺寸不能超过 vop 的尺寸。

	VOPB	VOPL
Max in/out	4096x2304/4096x2160	2048x1536/2048x1536

例如:

三个 1080p 的物理屏, 两个 mipi 屏在 VOPB 上, hdmi 显示设备在 VOPL 上, mipi 设置为

主屏，hdmi 设置为副屏，假如主副屏比例为 3:1，那么主屏的宽为  $5760 \times 3/4 = 4320$ ，超过了 4096。因此主副屏不能设置为 3:1。

属性设置完成后，需要重启 android，通过 stop;start 重启即可。

## 6.8 HDMI IN功能

### 6.8.1 功能描述

通过 HDMI 接口，将接入的设备画面显示到本机的显示画面中的功能方案，RK3399 平台底层已支持。

### 6.8.2 功能要求

1. 硬件上需要支持 HDMI IN 设计；
2. 需要安装 HDMI IN 应用；
3. 需要根据实际的硬件连接，修改内核 dts 中 tc358749x 节点的 gpio 配置；
4. 需要根据实际的硬件连接，修改

hardware/rockchip/camera/Config/cam\_board\_rk3399.xml 的配置。

### 6.8.3 相关代码位置

```
Kernel/  
hardware/rockchip/camera
```

## 6.9 主副屏旋转功能

在双屏同显或者异显的场景下，由于屏幕的差别，例如主屏为物理横屏，副屏为物理竖屏，需要将主屏或者副屏进行旋转，为了更方便的进行配置，行业 sdk 在 v2.1 版本以后整理了相关的代码，请更新代码到 v2.1 版本，并且参考文档“RKDocs\common\display\Rockchip\_双屏显示旋转方向调试文档\_V1.01\_20190226.pdf”进行调试。

## 6.10 深度学习

本 SDK 集成了深度学习框架 Caffe-on-ACL，源码路径为 external/caffe-on-acl/，具体使用方法请参考：[RKDocs\rk3399\RK3399\\_Android7.1\\_CaffeOnACL\\_开发说明文档\\_V1.0\\_20180225.pdf](#)。



## 6.11 多路camera支持

SDK 支持同时接多路 camera, rk3399 最多支持两路 mipi 加多路 usb camera (产品设计请注意 usb 带宽限制), 默认代码已经支持, app 样例源码请参考 [RKDocs\rk3399\patches\DoubleCamera-Example.rar](#)。

## 6.12 音视频多路编解码

SDK 提供了整套接口以及 demo 源码, 请参考文档: [RKDocs\rk3399\RK3399\\_Android7.1\\_MPI\\_Demo\\_说明文档\\_V1.0\\_20180409.pdf](#) 进行开发。

## 6.13 Audio 3A算法

本 SDK 源码已经集成了 AUDIO 3A 算法, 默认该功能未启用, 如果需要启用, 请修改 device/rockchip/rk3399/BoardConfig.mk, 将下面这个宏改为 true:

```
diff --git a/BoardConfig.mk b/BoardConfig.mk
index 8f68933..8c420fc 100755
--- a/BoardConfig.mk
+++ b/BoardConfig.mk
@@ -42,7 +42,7 @@ TARGET_BOARD_PLATFORM_GPU := mali-t860
BOARD_USE_DRM := true

# for audio 3A algorithm
-BOARD_USE_AUDIO_3A := false
+BOARD_USE_AUDIO_3A := true
```

AUDIO 3A对应的调试文档请参考[RKDocs/common/driver/RK语音通话 3A算法集成说明及参数调试说明文档 V3.0.pdf](#)。

## 6.14 双wifi (station+ap)

SDK 提供范例实现供客户参考, 补丁目录为: [RKDocs\rk3399\patches\dual\\_wifi\\_patch\\_for\\_android7.1\\_and\\_android8.1.rar](#)。

## 6.15 双以太网

SDK 支持双以太网卡, 通过配置 device/rockchip/rk3399 目录下的 device.mk 文件, 可以配置以太网的模式, 有 3 种选择: nomal/multi/bridge, normal 表示单以太网, multi 表示普通的双以太网, bridge 表示双以太网的桥接方式。如下补丁, 表示将以太网改为双网卡模式。

```
diff --git a/device.mk b/device.mk
index e37c3cf..f367a29 100755
--- a/device.mk
+++ b/device.mk
@@ -77,7 +77,7 @@ endif

# normal for single ethernet, multi for two, bridge for lan bridge
# You can change this value in system
# It would work after reboot
-ADDITIONAL_DEFAULT_PROPERTIES += persist.net.ethernet.mode=normal
+ADDITIONAL_DEFAULT_PROPERTIES += persist.net.ethernet.mode=multi
```

## 6.16 手写优化

手写优化相关的补丁和说明，请参考 [RKDocs/rk3399/patches/手写优化 v1.0.rar](#)。

## 6.17 梯形校正功能

SDK 支持通过 gpu 对输出进行梯形变换来达到投影设备梯形校正的效果，例如车载投影设备投影到前窗玻璃时，由于玻璃平面和投影设备间的角度以及投影面的不规则，会导致输出图像存在梯形畸变，该补丁可以让客户进行畸变的调校，具体的补丁和说明请参考：

[RKDocs/rk3399/patches/keystone\\_patch\\_for\\_rk3399\\_Industry\\_android7.1\\_v0.4\\_20171225.tar.gz](#)。

## 6.18 基于深度学习的目标检测（SSD）方案

SDK 提供基于深度学习的目标检测（SSD）优化方案，可为 AI 人工智能行业提供准 Turnkey 解决方案，具体的使用方法请参考文档：[RKDocs/rk3399/ RK3399\\_SSD\\_Android\&Linux\\_V1.0\\_20180522.pdf](#)。另外，demo 应用源码请参考 [RKDocs/rk3399/patches/RK3399\\_SSD\\_Android\\_V1.0\\_20180613.rar](#)，该源码需要使用 Android Studio 进行编译。

## 6.19 Widevine补丁

SDK 支持 Widevine 功能，如果需要支持 widevine 功能，请先参考文档：[RKDocs/android/Rockchip Android Widevine 项目启动准备指南 V2.0-20180622.pdf](#) 进行相关准备工作，另外相关 patch 请参考 [RKDocs/rk3399/patches/widevine 补丁.rar](#)。

## 6.20 高可靠OTA

SDK 支持高可靠 OTA 升级方法，可以防止机器在升级过程中掉电等异常操作导致的机器再也无法启动问题，SDK 默认为不使能，如果要开启该功能请参考文档：[RKDocs/android/可靠 OTA](#)

使用说明文档.pdf。

## 6.21 显示参数的调整和保存

RK 平台提供一个分区来保存显示参数，系统可以从该分区读取显示参数并且应用。该分区我们称为 baseparameter 分区，有以下两种功能：

- 保存和调整 LCD 色温、对比度、保护度、色度等信息；
- 保存和调整 hdmi 或者 dp 等显示设备支持的分辨率、时序等信息；

如果客户需要使用上面的功能，请按照下面的方法使能 BaseParameter 分区功能：

```
device/rockchip/common$ git diff
diff --git a/BoardConfig.mk b/BoardConfig.mk
index afeb107..b371b97 100755
--- a/BoardConfig.mk
+++ b/BoardConfig.mk
@@ -56,7 +56,7 @@ TARGET_CPU_ABI2 ?=
TARGET_CPU_SMP ?= true
endif

-BOARD_BASEPARAMETER_SUPPORT ?= false
+BOARD_BASEPARAMETER_SUPPORT ?= true

ifeq ($(strip $(BOARD_BASEPARAMETER_SUPPORT)), true)
TARGET_RECOVERY_OVERSCAN_PERCENT := 2
```

系统会编译生成对应的 baseparameter.img 固件，但是默认的烧写工具中不包含这个分区，请根据文档：《RKDocs/android/ Android 增加一个分区配置指南 V1.00.pdf》增加一个分区。

baseparameter 分区功能更为详细的介绍请参考文档：《RKDocs/android/ 显示参数存储和配置说明文档\_V1.1\_20181024.pdf》。另外，有些客户有在运行时动态修改这些显示参数的需求，因此我们在框架层增加了一些接口以方便客户进行调用，具体也可以参考上述文档进行设计。

## 6.22 自动化测试和debug脚本

针对自动化测试，我们提供了StressTest apk给客户，详情请见[8.1 节StressTest](#)，为了覆盖更多的测试场景，我们新增了自动化测试和debug脚本供客户使用，测试脚本位于如下代码目录中：

```
device/rockchip/rk3399/rockchip_test
```

目录结构如下:

```

├── ddr
│   ├── ddr_test.sh
│   ├── libstlport.so
│   ├── memtester_32bit
│   ├── memtester_64bit
│   ├── memtester_test.sh
│   ├── stressapptest
│   └── stressapptest_test.sh
├── dvfs
│   ├── auto_cpu_freq_test.sh
│   ├── auto_ddr_freq_test.sh
│   ├── auto_gpu_freq_test.sh
│   └── dvfs_test.sh
├── rockchip_test.sh
└── system_monitor
    ├── hardware_monitor.sh
    └── memory_monitor.sh
  
```

用户可以使用如下方式调用这些脚本:

```

sh /system/bin/rockchip_test.sh

*****

***                               ***

***          *****          ***

***      *ROCKCHIPS TEST TOOLS*      ***

***          *              *          ***

***          *****          ***

***                               ***

*****

*****

*****

ddr test :          1 (memtester & stressapptest)
  
```

```
dvfs_test:          2 (dvfs stresstest, including cpu/gpu/ddr)
memory_monitor:     3 (tools used to detect memory leak)
hardware_monitor:   4 (tools used to monitor cpu/gpu/ddr freq and temp
erature)

*****

please input your test moudle:
```

输入对应的选择，即会跑对应的脚本，详细解释如下：

- Ddr test: ddr 压力测试工具 memtester 和 stressapptest，可用于测试 ddr 稳定性。
- Dvfs test: 测试 cpu、gpu、ddr 变频测试，测试变频的稳定性。
- Memory\_monitor: 该脚本用于检测内存泄露问题，会将内存使用情况保存在机器的 /data/logs/rockchip\_test/memory\_monitor/ 目录下。
- hardware\_monitor: 该脚本用于实时打印当前的 cpu、gpu、ddr 频率、负载以及温度情况。

## 6.23 DATA分区文件系统切换为EXT4

从行业 SDK 的 V2.1 版本开始，data 分区默认从 F2FS 格式切换为 EXT4 格式文件系统，F2FS 相对于以前使用的 EXT4 而言，随机读写的效率会优于 EXT4。但是另外一方面，EXT4 使用的范围更广，经历的时间考验也更久，稳定性这块我们认为可能会优于 F2FS，所以对于行业客户而言，如果更注重文件系统的稳定性，建议使用 EXT4。

### 6.23.1 注意事项

**【注意】**从 F2FS 文件系统切到 EXT4 需要格式化 data 分区，会导致 data 分区数据全部清空，因此如果机器已经量产，不建议切换文件系统。已量产的客户如果 sdk 更新到了 V2.1 版本，并且原先的文件系统为 F2FS，请回退 device/rockchip/rk3399 目录下的如下提交：

```
commit 025a783298e3b74fa51c252a07678b867b4ba431
Author: Wenping Zhang <wenping.zhang@rock-chips.com>
Date:   Fri Mar 8 15:42:14 2019 +0800

fstab: change the userdata partition to ext4 format.

Change-Id: I15f6c33cc4e297cb48e86434ef1e31ff9ef52d5f
Signed-off-by: Wenping Zhang wenping.zhang@rock-chips.com
```

如果客户能够接受 **data** 分区擦除，需使用完整固件烧写，不能够使用 **ota** 升级（因为 **ota** 升级默认不会重新格式化 **data** 分区）。或者可以考虑使用 **sd** 升级卡，将 **sd** 卡做成升级卡进行机器的升级，如果使用 **sd** 升级卡，需要作如下配置确保会重新格式化 **data** 分区：

```
device/rockchip/common$ git diff
diff --git a/BoardConfig.mk b/BoardConfig.mk
index c7ac4d3..3db7bdc 100755
--- a/BoardConfig.mk
+++ b/BoardConfig.mk
@@ -366,7 +366,7 @@ BOARD_USB_ALLOW_DEFAULT_MTP ?= false
BOARD_USE_FIX_WALLPAPER ?= false

# SDBoot: Format data.
-RECOVERY_SDBOOT_FORMATATE_DATA ?= false
+RECOVERY_SDBOOT_FORMATATE_DATA ?= true

HIGH_RELIABLE_RECOVERY_OTA := false
BOARD_USES_FULL_RECOVERY_IMAGE := false
```

### 6.23.2 Recovery擦除慢问题

从 F2FS 切到 EXT4 以后，我们发现 **recovery** 擦除 **data** 分区会很慢，这个是因为 **emmc** 擦除的方式分为两种：

- Secure Discard

这个擦除方式为物理擦除，会完全擦除 **data** 分区的数据，主要是为了确保数据的安全性，保证 **data** 分区不会被恶意导出。但是这种方式的缺点为耗时长，尤其是数据写满的情况下（比如从 F2FS 文件系统切换为 EXT4 文件系统时）。

注：有些小厂的 **emmc**，**Secure Discard** 方式并未实现完全擦除，而是和 **Discard** 方式一样只是擦除索引文件。

- Discard

这种擦除方式为逻辑擦除，类似于擦除索引文件，但不会擦除存储上的每个物理位。这种方式的优点是擦除速度快，缺点是在擦除后，由于只擦除了索引文件，**data** 分区数据理论上有被恶意导

出的风险,建议使能 data 分区的加密功能来降低被恶意导出的风险(SDK 加密功能默认为开启的)。

系统默认使用 Secure Discard 方式,考虑了上面的风险以后,如果客户仍然需要选择格式化更快速的 Discard 方式,请打下面的补丁:

```
system/extras$ git diff
diff --git a/ext4_utils/wipe.c b/ext4_utils/wipe.c
index 5766632..0700797 100644
--- a/ext4_utils/wipe.c
+++ b/ext4_utils/wipe.c
@@ -44,7 +44,7 @@ int wipe_block_device(int fd, s64 len)

     range[0] = 0;
     range[1] = len;
-    ret = ioctl(fd, BLKSECDISCARD, &range);
+    ret = ioctl(fd, BLKDISCARD, &range);
     if (ret < 0) {
         range[0] = 0;
         range[1] = len;
```

### 6.23.3 F2FS切换为EXT4 方法

如果是 V2.1 以前的版本, data 分区默认还是 F2FS 文件系统, 请更新到 V2.1 以后的版本, 如果不想所有更新, 请按照下述方法切换为 EXT4:

1. 首先确认是否有使用强制加密功能, 即 BUILD\_WITH\_FORCEENCRYPT 这个宏是否有是能, SDK 默认为使能状态, 这个宏会决定使用哪个 fstab 文件, 详见下述代码:

```
device/rockchip/rk3399$ vim device.mk
ifeq ($(BUILD_WITH_FORCEENCRYPT),true)
PRODUCT_COPY_FILES += \
    $(LOCAL_PATH)/fstab.rk30board.bootmode.forceencrypt.unknown:root/fstab.
rk30board.bootmode.unknown \
    $(LOCAL_PATH)/fstab.rk30board.bootmode.forceencrypt.emmc:root/fstab.rk
30board.bootmode.emmc \
    $(LOCAL_PATH)/fstab.rk30board.bootmode.forceencrypt.nvme:root/fstab.rk3
0board.bootmode.nvme
```

```

else
PRODUCT_COPY_FILES += \
    $(LOCAL_PATH)/fstab.rk30board.bootmode.unknown:root/fstab.rk30board.b
ootmode.unknown \
    $(LOCAL_PATH)/fstab.rk30board.bootmode.emmc:root/fstab.rk30board.boot
mode.emmc \
    $(LOCAL_PATH)/fstab.rk30board.bootmode.nvme:root/fstab.rk30board.boot
mode.nvme
endif

```

2. 需要根据存储的介质找到对应的 fstab，然后修改对应的 data 分区加载选项，下面以 emmc 为例：

```

device/rockchip/rk3399$ git diff
diff --git a/fstab.rk30board.bootmode.forceencrypt.emmc b/fstab.rk30board.bo
otmode.forceencrypt.emmc
index 9adbfb9..833c85e 100755
--- a/fstab.rk30board.bootmode.forceencrypt.emmc
+++ b/fstab.rk30board.bootmode.forceencrypt.emmc
@@ -8,11 +8,11 @@
    #/dev/block/platform/fe330000.sdhci/by-name/system          /system
    ext4      ro,noatime,nodiratime,noauto_da_alloc
    wait,check,verify
    /dev/block/platform/fe330000.sdhci/by-name/cache            /cache
    ext4      noatime,nodiratime,nosuid,nodev,noauto_da_alloc,discard
    wait,check
    /dev/block/platform/fe330000.sdhci/by-name/metadata         /metadata
    ext4      noatime,nodiratime,nosuid,nodev,noauto_da_alloc,discard
    wait,check
    -/dev/block/platform/fe330000.sdhci/by-name/userdata         /data
    f2fs      noatime,nodiratime,nosuid,nodev,discard,inline_xattr
    wait,check,notrim,forceencrypt=/metadata/key_file
    +#/dev/block/platform/fe330000.sdhci/by-name/userdata       /data
    f2fs      noatime,nodiratime,nosuid,nodev,discard,inline_xattr
    wait,check,notrim,forceencrypt=/metadata/key_file
    #data for f2fs nobarrier
    #/dev/block/platform/fe330000.sdhci/by-name/userdata        /data
    f2fs      noatime,nodiratime,nosuid,nodev,discard,inline_xattr,nobarrier
    wait,check,notrim,forceencrypt=/metadata/key_file
    #data for ext4
    -#/dev/block/platform/fe330000.sdhci/by-name/userdata       /data
    ext4      noatime,nodiratime,nosuid,nodev,noauto_da_alloc,discard,errors=panic
    wait,check,forceencrypt=/metadata/key_file

```



```
+ /dev/block/platform/fe330000.sdhci/by-name/userdata      /data
ext4      noatime,nodiratime,nosuid,nodev,noauto_da_alloc,discard,errors=panic
wait,check,forceencrypt=/metadata/key_file
  /dev/block/platform/fe330000.sdhci/by-name/misc          /misc
emmc      defaults      defaults
# sdcard
  /devices/platform/fe320000.dwmmc/mmc_host*              auto  auto
defaults      voldmanaged=sdcard1:auto,encryptable=userdata
```

### 3. 修改 recovery 模式时 data 分区加载选项:

```
device/rockchip/rk3399$ git diff
diff --git a/recovery.emmc.fstab b/recovery.emmc.fstab
index d232dbd..ac58660 100755
--- a/recovery.emmc.fstab
+++ b/recovery.emmc.fstab
@@ -5,7 +5,7 @@
  /dev/block/platform/fe330000.sdhci/by-name/system      /system
      ext4      defaults      defaults
  /dev/block/platform/fe330000.sdhci/by-name/cache      /cache
      ext4      defaults      defaults
  /dev/block/platform/fe330000.sdhci/by-name/metadata   /metadata
      ext4      defaults      defaults
- /dev/block/platform/fe330000.sdhci/by-name/userdata   /data
      f2fs      defaults      defaults
+ /dev/block/platform/fe330000.sdhci/by-name/userdata   /data
      ext4      defaults      defaults
  /dev/block/platform/fe330000.sdhci/by-name/cust       /cust
      ext4      defaults      defaults
  /dev/block/platform/fe330000.sdhci/by-name/custom     /custom
      ext4      defaults      defaults
  /dev/block/platform/fe330000.sdhci/by-name/misc
```

## 6.24 Data分区使能和禁用加密功能

SDK 默认使能了 data 的加密功能，但是我们发现有些客户会有一些特殊的需求，比如在

android 很早的初始化阶段去访问 data 分区，但是此时 data 分区正处于加密流程，读取会出现失败问题。举例说明：

假如客户需要配置一个 persist 属性，需要重启后生效，并且这个属性在 surfaceflinger 服务启动的时候要去判断，此时会发现 surfaceflinger 服务读取该属性失败。

为了解决上述问题，并且不需要 data 分区的加密功能，可以将加密功能去掉，如下面补丁所示：

```
device/rockchip/rk3399$ git diff
diff --git a/fstab.rk30board.bootmode.forceencrypt.emmc b/fstab.rk30board.bo
tmode.forceencrypt.emmc
index 9adbfb9..c0afafc 100755
--- a/fstab.rk30board.bootmode.forceencrypt.emmc
+++ b/fstab.rk30board.bootmode.forceencrypt.emmc
@@ -8,7 +8,7 @@
    #/dev/block/platform/fe330000.sdhci/by-name/system          /system
    ext4      ro,noatime,nodiratime,noauto_da_alloc
    wait,check,verify
    /dev/block/platform/fe330000.sdhci/by-name/cache            /cache
    ext4      noatime,nodiratime,nosuid,nodev,noauto_da_alloc,discard
    wait,check
    /dev/block/platform/fe330000.sdhci/by-name/metadata          /metadata
    ext4      noatime,nodiratime,nosuid,nodev,noauto_da_alloc,discard
    wait,check
    -/dev/block/platform/fe330000.sdhci/by-name/userdata          /data
    f2fs      noatime,nodiratime,nosuid,nodev,discard,inline_xattr
    wait,check,notrim,forceencrypt=/metadata/key_file
    +/dev/block/platform/fe330000.sdhci/by-name/userdata          /data
    f2fs      noatime,nodiratime,nosuid,nodev,discard,inline_xattr
    wait,check,notrim,encryptable=/metadata/key_file
    #data for f2fs nobarrier
```

```
#/dev/block/platform/fe330000.sdhci/by-name/userdata      /data
f2fs      noatime,nodiratime,nosuid,nodev,discard,inline_xattr,nobarrier    wait,c
heck,notrim,forceencrypt=/metadata/key_file
#data for ext4
```

需要将 device/rockchip/rk3399 下面对应 fstab 文件中，将 data 分区的挂载参数从 forceencrypt 改为 encryptable。

## 6.25 RK3399 性能优化方法

我们提供了一些方法对 RK3399 的性能进行优化，但是请注意，性能上的优化会引起功耗的增以及发热量的增大，这块请客户自行考量项目的需求。性能优化的方法详见文档：

RKDocs/rk3399/RK3399 性能优化方法\_V1.3\_20190116.pdf

## 6.26 RK3399K 芯片支持

从 sdk 版本 v1.9（通过 ls .repo/manifests/ -l 确认版本）开始支持 RK3399K 芯片，但需要在最终使用的 dts 中手动加入下面的补丁，以 RK3399 SDK 为例：

```
git diff
diff --git a/arch/arm64/boot/dts/rockchip/rk3399-sapphire-excavator-edp.dts b/arch/arm64/boot/dts/rockchip/rk3399-sapphire-excavator-edp.dts
index 728b430..387b132 100644
--- a/arch/arm64/boot/dts/rockchip/rk3399-sapphire-excavator-edp.dts
+++ b/arch/arm64/boot/dts/rockchip/rk3399-sapphire-excavator-edp.dts
@@ -45,6 +45,7 @@
#include "rk3399-excavator-sapphire.dtsi"
#include "rk3399-android.dtsi"
#include "rk3399-vop-clk-set.dtsi"
+#include "rk3399k-opp.dtsi"

/ {
    model = "Rockchip RK3399 Excavator Board edp (Android)";
```

## 6.27 禁用串口打印功能

有些客户需要关闭串口打印功能，另外有些客户可能因为串口不够用需要将打印串口用作普通的串口使用，但是发现在调试过程中，经常出现由于配置不到位导致的系统无法开机问题。从 SDK

V2.1 版本开始，加入了禁用串口打印功能的支持。需要注意如下几个配置：

- 版本升级到 V2.1 以后，如果出现串口打印到一半停掉不再打印（系统未死机）的问题，请确认 dts 中 uart2 是否有 disable，因为新代码中 fiq 和普通串口只能二选一。如下所示：

```
--- a/arch/arm64/boot/dts/rockchip/rk3399-firefly-android.dts
+++ b/arch/arm64/boot/dts/rockchip/rk3399-firefly-android.dts
@@ -1050,7 +1050,7 @@
};

&uart2 {
-    status = "okay";
+    status = "disabled";
};

&usbdrd3_0 {
```

- 如果想禁用 uart2 的 fiq 调试功能，把 uart2 当成普通串口用，请按照如下方式修改：

```
--- a/arch/arm64/boot/dts/rockchip/rk3399-firefly-android.dts
+++ b/arch/arm64/boot/dts/rockchip/rk3399-firefly-android.dts
@@ -1050,7 +1050,12 @@
};

&uart2 {
-    status = "disabled";
+    status = "okay";
+};
+
+&fiq_debugger {
+    rockchip,serial-id = <0xffffffff>;
+    status = "okay";
};
```

```
&usbd3_0 {
```

## 6.28 DP音频功能

从行业 SDK V2.1 版本开始，为了解决一些 bug 以及提升 DP 音频的兼容性，DP 音频从 I2S 切换到 SPDIF 通道，该修改涉及几个部分，请同步下面几个仓库的修改：

```
hardware/rockchip/audio
frameworks/base
kernel
```

经过修改以后 HDMI 音频走 I2S 通道，DP 音频走 spdif 通道。上述修改中内核部分请同步到最新代码，并按照下面的说明确认当前的 dts 配置是否正确（注意，由于客户参考的 dts 配置不同，请自行在对应的 dts 配置中添加对应的修改）：

- 删除下面的节点

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3399-android.dtsi b/arch/arm64/boot/dts/rockchip/rk3399-android.dtsi
index 6028ac8..62a427b 100644
--- a/arch/arm64/boot/dts/rockchip/rk3399-android.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3399-android.dtsi
@@ -255,13 +255,6 @@
        rockchip,android-charge-on = <0>;
    };

-    hdmi_dp_sound: hdmi-dp-sound {
-        status = "disabled";
-        compatible = "rockchip,rk3399-hdmi-dp";
-        rockchip,cpu = <&i2s2>;
-        rockchip,codec = <&hdmi>, <&cdn_dp>;
-    };
-
```

- 添加 dp\_sound 节点

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3399-sapphire.dtsi b/arch/arm64/b
oot/dts/rockchip/rk3399-sapphire.dtsi
index 8a3d7ea..a309c08 100644
--- a/arch/arm64/boot/dts/rockchip/rk3399-sapphire.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3399-sapphire.dtsi
@@ -100,6 +100,13 @@
        #sound-dai-cells = <0>;

    };

+    dp_sound: dp-sound {
+        status = "disabled";
+        compatible = "rockchip,cdndp-sound";
+        rockchip,cpu = <&spdif>;
+        rockchip,codec = <&cdn_dp 1>;
+    };
+
+

```

- 禁用以及使能一些节点

```
+&spdif {
+    status = "okay";
+};
+
+&dp_sound {
+    status = "okay";
+};
+
+/*
+ * if enable dp_sound, should disable spdif_sound and spdif_out
+ */
+&spdif_out {
+    status = "disabled";

```

```
+};  
  
+  
+&spdif_sound {  
+    status = "disabled";  
+};  
  
+  
+&hdmi_sound {  
+    status = "okay";  
+};  
  
+
```

## 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/system.prop 文件，添加如下配置：

```
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。

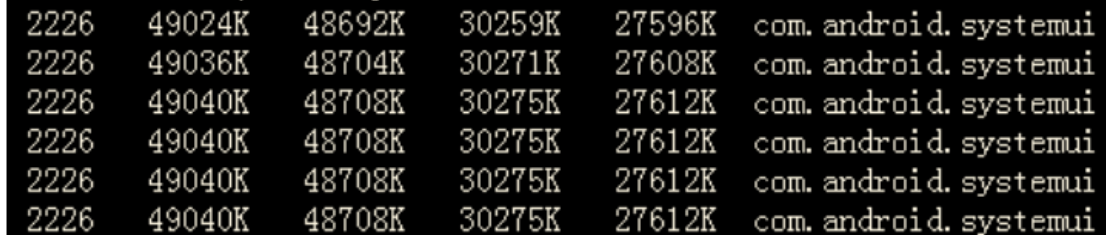
示例：输出进程名为 com.android.systemui 的应用内存占用状态，查看是否有泄露：

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 0-1 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

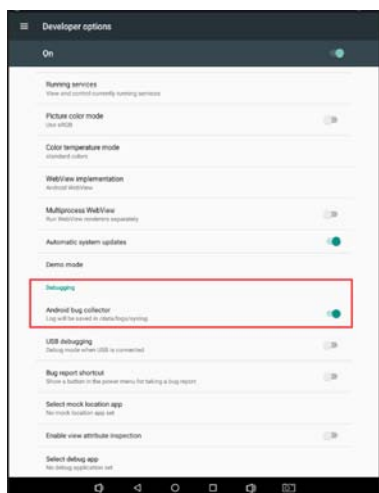
cat ftrace-ramoops-0

## 7.8 Log自动保存系统

为了简化客户的操作, 让没有 android 开发基础的人抓出问题时的 log, 我们新增一个 log 自动保存功能。

### 7.8.1 使用方法:

(1)在开发者选项中点击 Android bug collector 来开启此功能:

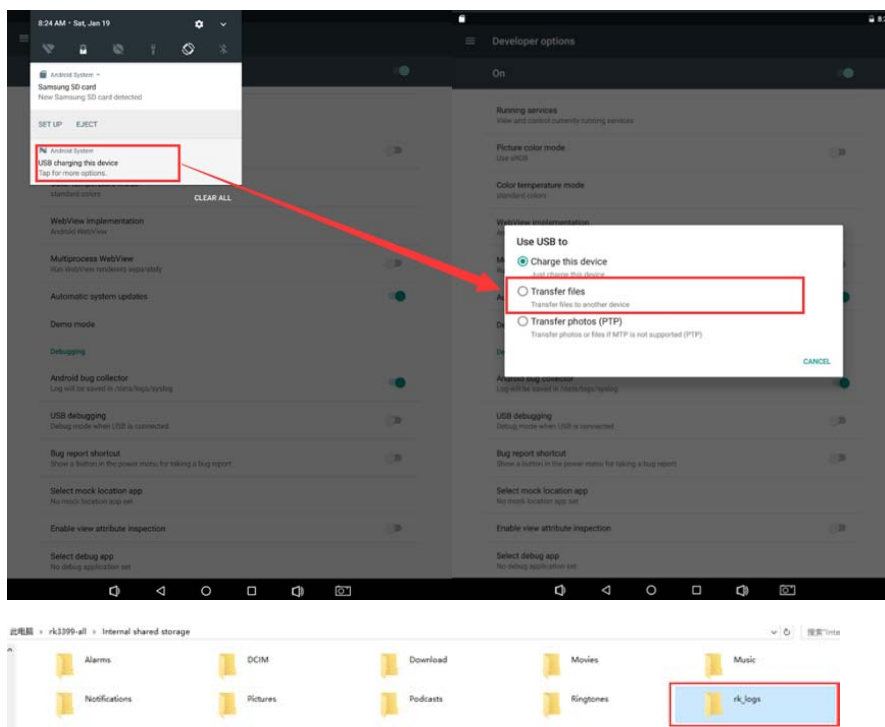


(2)导出 flash 中的日志到 PC 的使用方法

重新开机后连接电脑, 选择可以看到/data/media/0 目录下有 rk\_logs 目录有以下内容, 并且有 COPY-COMplete 表示拷贝完成。可以直接通过 adb pull /data/media/0/rk\_logs 导出日志, 或者打开 usb 传输模式, 直接拷贝到电脑上, 见下图。



```
rk3399_all:/data/media/0/rk_logs # ls -al
total 9562
drwxrwxrwx 6 root root 3488 2013-01-20 09:19 .
drwxrwx 12 media_rw media_rw 3488 2013-01-20 09:19 ..
-rw-rw-rw 1 root root 0 2013-01-20 09:19 COPY-COMplete
drwxrwxr-x 2 root root 3488 2013-01-20 09:19 anr
-rw-rw-rw 1 root root 4867234 2013-01-20 09:19 bugreport.log
drwxrwx--- 16 root root 3488 2013-01-20 09:19 logs
drwxr-xr-x 2 root root 3488 2013-01-20 09:19 pstore
drwxr-xr-x 2 root root 3488 2013-01-20 09:19 tombstones
```

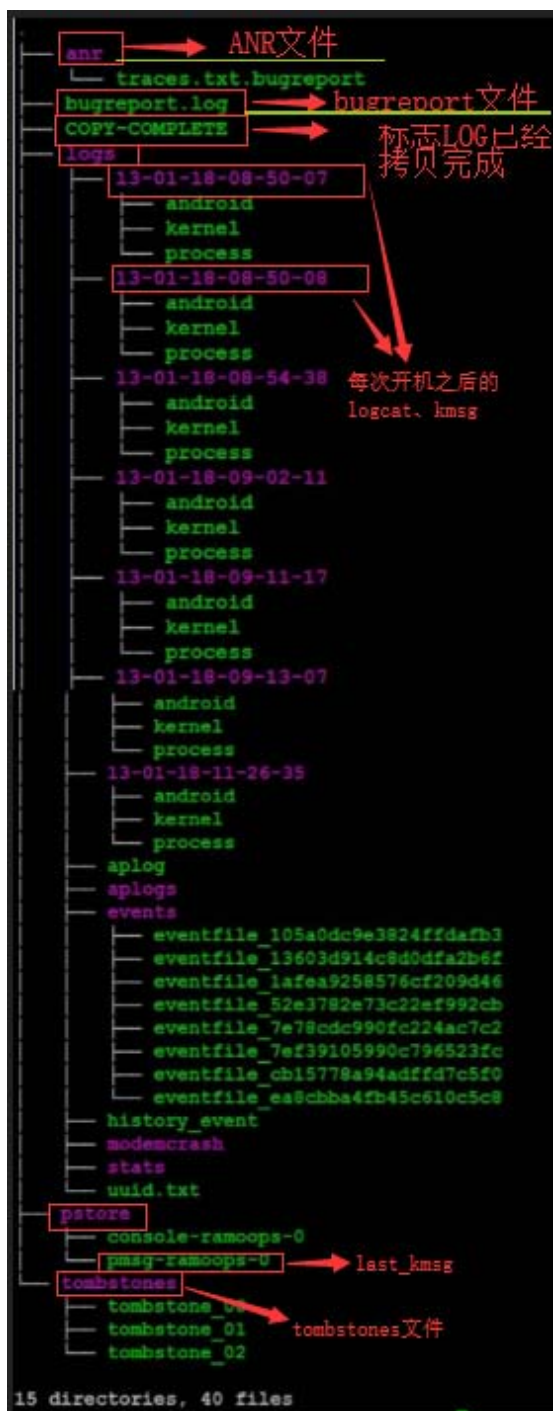


### (3)导出 Log 到 SD 卡的使用方法

sd 卡插入后,几秒后(生成 bugreport 需要一些时间),后可以看到/storage/8527-18E3/ 目录(该目录每个产品可能各有不同)下有 rk\_logs 目录有以下内容,并且有 COPY-COMplete 表示拷贝完成。

```
rk3399_all:/storage/8527-18E3/rk_logs # ls -al
total 14744
drwxrwx--x 6 root sdcard_rw 4096 2013-01-20 09:27 .
drwxrwx--x 14 root sdcard_rw 4096 2013-01-20 09:27 ..
-rwxrwx--x 1 root sdcard_rw 0 2013-01-20 09:27 COPY-COMplete
drwxrwx--x 2 root sdcard_rw 4096 2013-01-20 09:27 anr
-rwxrwx--x 1 root sdcard_rw 7522502 2013-01-20 09:27 bugreport.log
drwxrwx--x 16 root sdcard_rw 4096 2013-01-20 09:27 logs
drwxrwx--x 2 root sdcard_rw 4096 2013-01-20 09:27 pstore
drwxrwx--x 2 root sdcard_rw 4096 2013-01-20 09:27 tombstones
```

## 7.8.2 对log的说明



## 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。

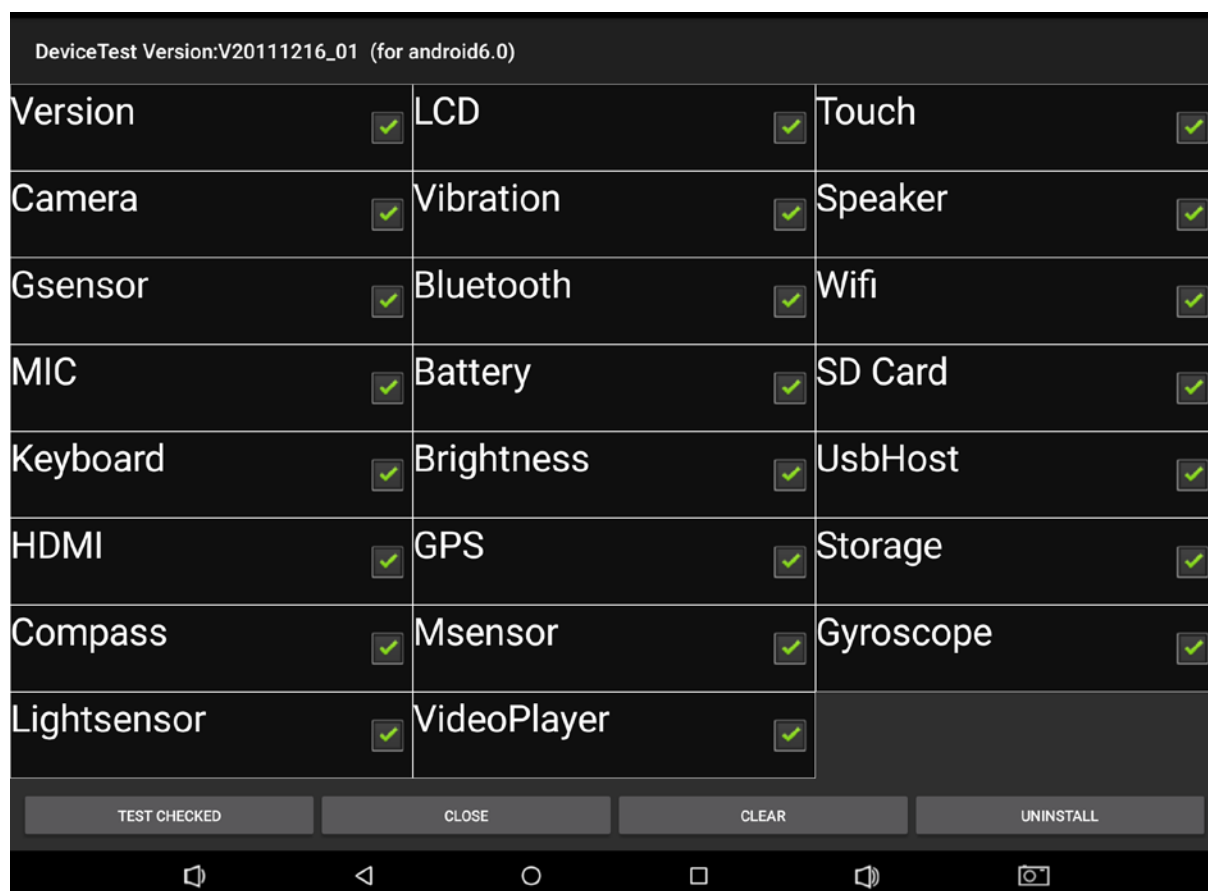
这些测试项目包括自动测试项和手动测试项。无线网络、DDR/eMMC、以太网为自动测试项，按键、SD 卡、USB Host、Codec、为手动测试项目。

具体 PCBA 功能配置及使用说明，请参考：

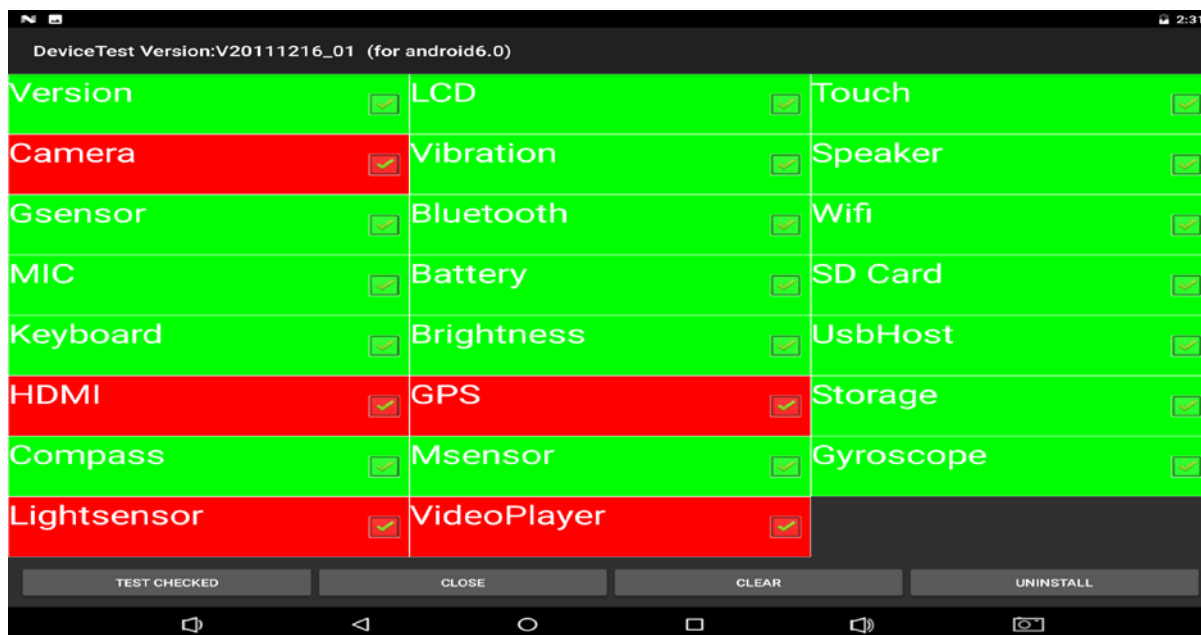
[\RKDocs\common\RKTools\\_manuals\Rockchip\\_PCBA模块开发指南--20170210.pdf](#)

### 8.3 DeviceTest

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



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



## 8.4 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。DDR 稳定性测试工具请参考 6.22 节，另外我们还提供产线测试工具，具体请联系 FAE 获取。

## 8.5 Android开发工具

### 8.5.1 下载镜像

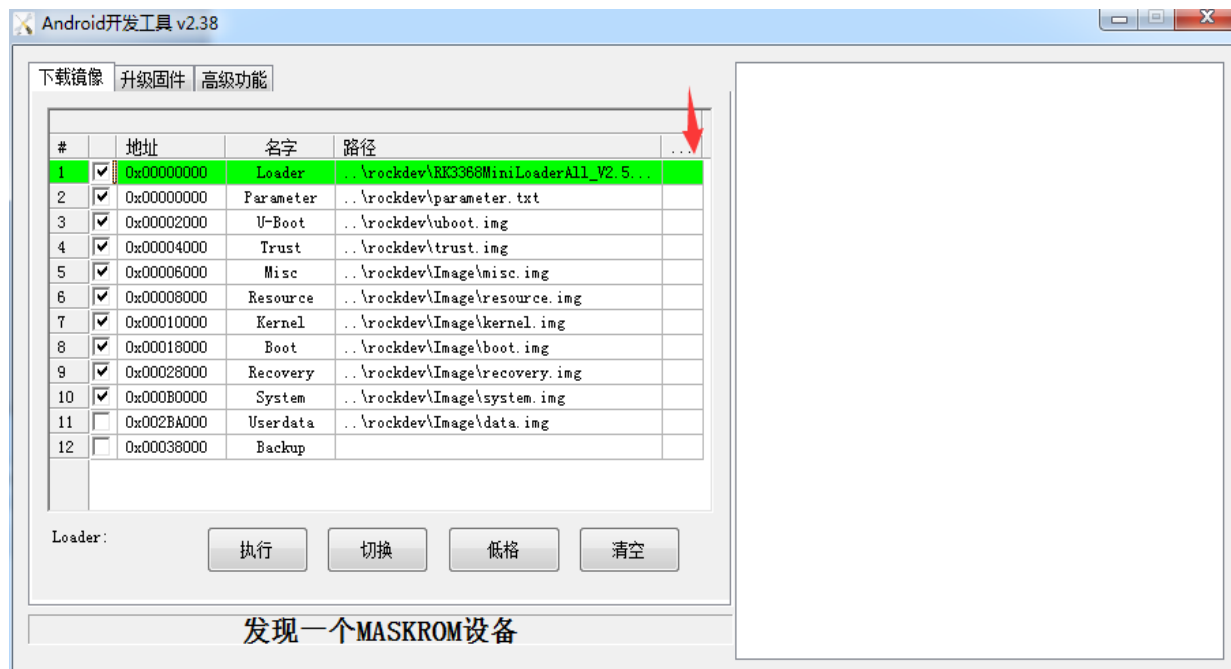


图 8-1 Android 开发工具下载镜像

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

### 8.5.2 升级固件

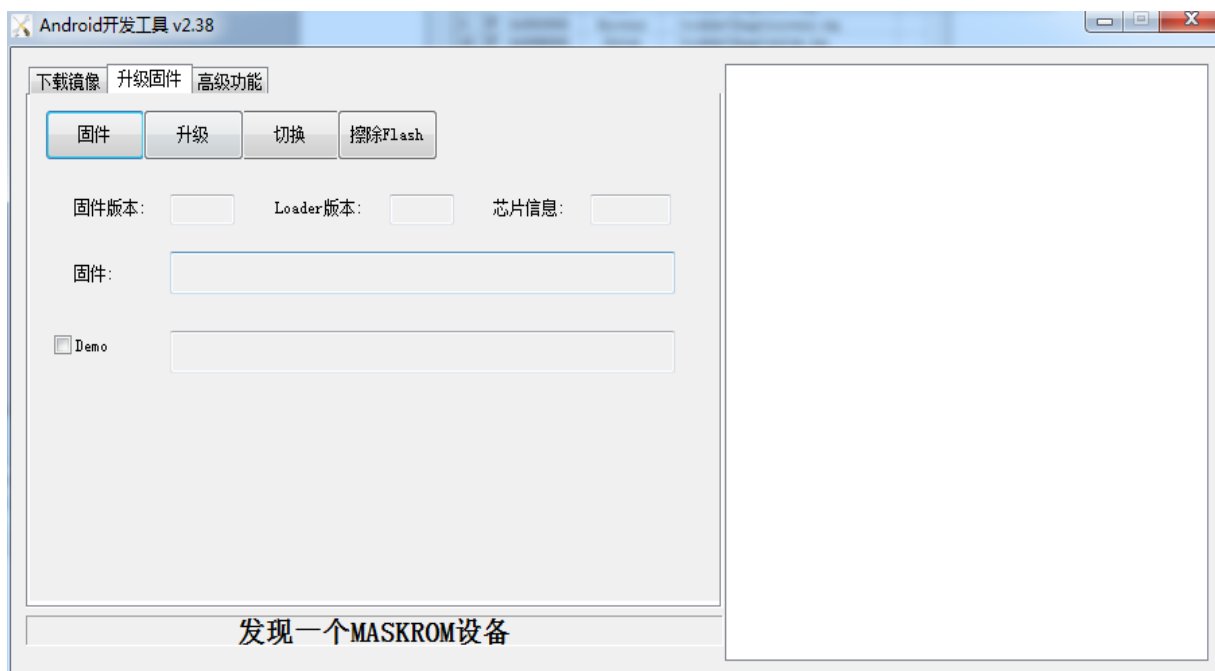


图 8-2 Android 开发工具升级固件

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。

### 8.5.3 高级功能

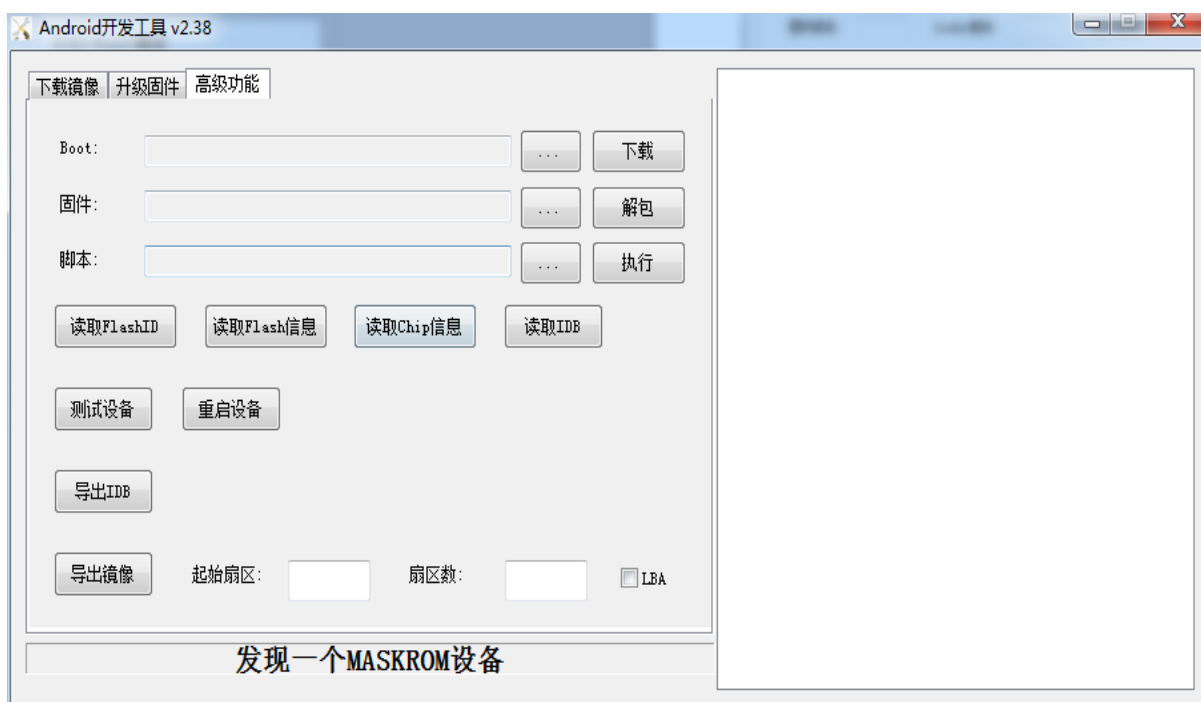


图 8-3 Android 开发工具高级功能

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。

## 8.6 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。

## 8.7 固件签名工具

参考 RKTools\windows\SecureBootTool\_v1.83\_foruser.rar 中的《Rockchip Secure Boot Application Note》

## 8.8 序列号/Mac/厂商信息烧写-WNpctool工具

本平台使用 WNpctool 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。

### 8.8.1 使用WNpctool写入

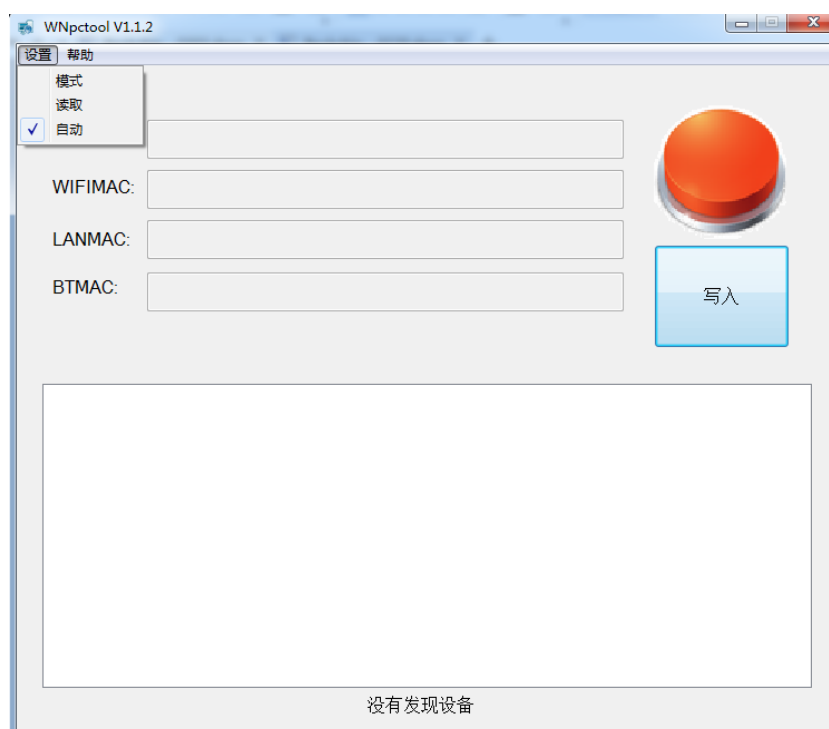


图 8-4WNpctool 工具

- 进入 loader 模式。
  - 点击“设置”菜单，下拉框中取消勾选“读取”。
- （勾选“读取”进行读取，未勾选“读取”则切换到写入功能）
- 点击“设置”菜单，点击“模式”，弹出“模式”窗口，用来设置 SN/WIFI/LAN/BT



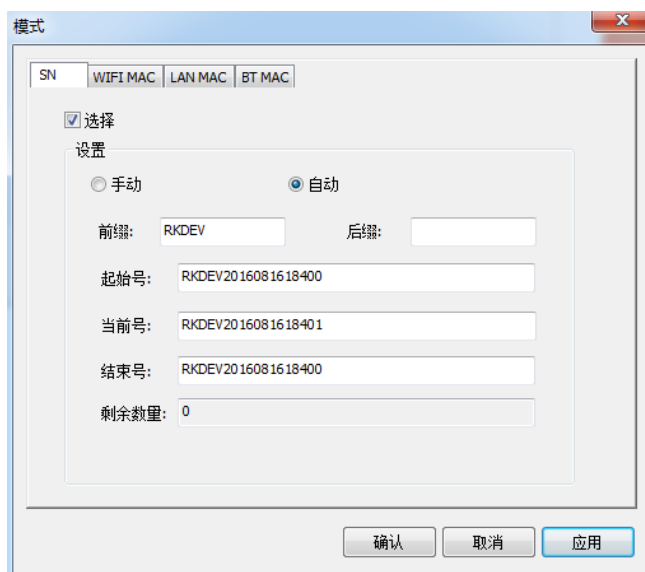


图 8-5WNpctool 工具模式设置

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

### 8.8.2 使用WNpctool读取

- 1) 进入 loader 模式。
- 2) 点击“设置”菜单，下拉框中勾选“读取”。  
(勾选“读取”进行读取，未勾选“读取”则切换到写入功能)
- 3) 点击“读取”按钮即可。

## 8.9 OemTool打包工具

### 8.9.1 Oem打包工具步骤

下载分区默认 UserData 分区，可直接不填写。

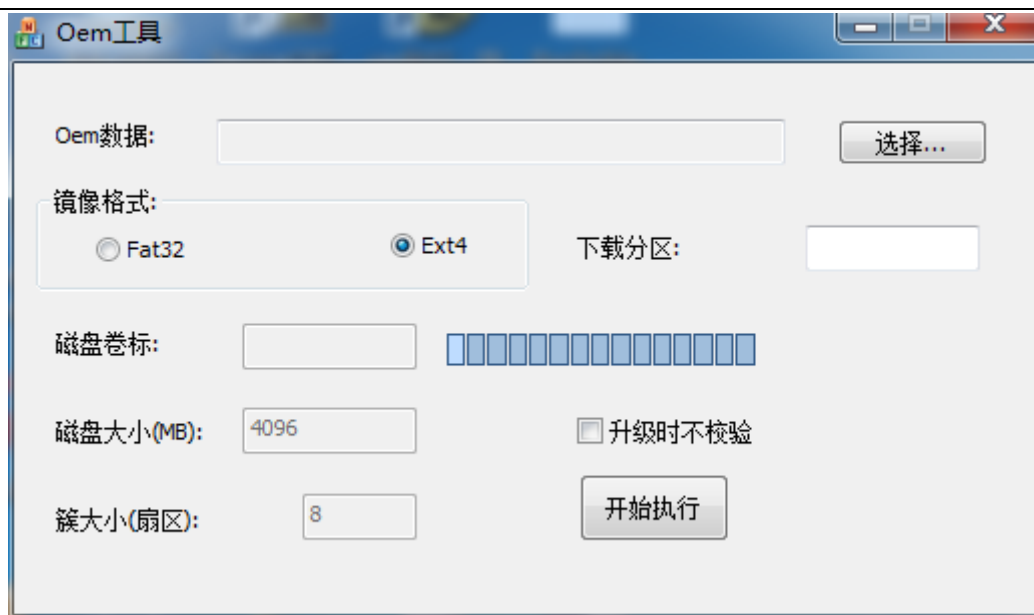


图 8-6 Oem 工具

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，假设目录为/data/media/0,且0有一个文件为sss.txt(如下图示)。则当升级完 demo 镜像的时候，会在设备的 data/media/0 目录下生成 sss.txt。

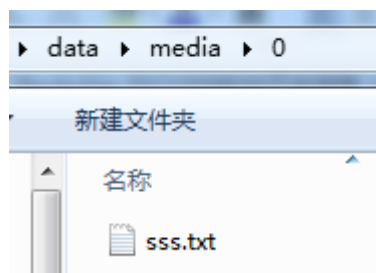


图 8-7 Oem 工具镜像制作文件夹路径要求

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。

## 8.10 量产工具使用

### 8.10.1 工具下载步骤

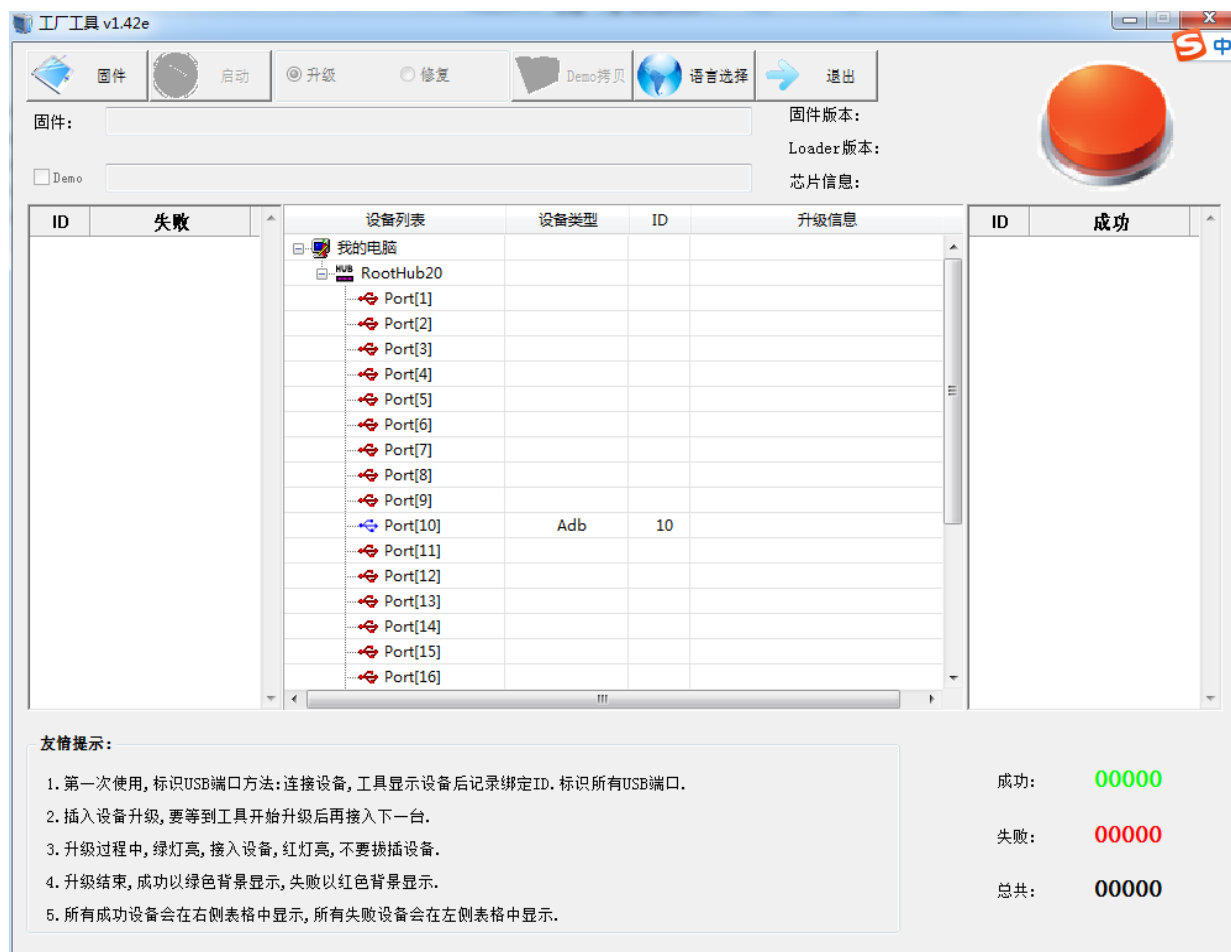


图 8-8 量产工具

- 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) 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。

## 9 常见问题分析

本章主要针对 RK3399 Android7.1 行业 SDK (RK3399\_ANDROID7.1-Industry-SDK\_V 1.0\_20180408) 常见问题进行分析和解答, 以方便客户更快的进行板级调试以及稳定性问题分析。

## 9.1 硬件设计对应软件修改

本节主要介绍不同硬件设计对应的软件差异部分。

### 9.1.1 如何选择dts配置

客户根据不同参考设计，部分核心电路可能会有差异，比如参考平板的原理图以及挖掘机 SDK 或者 firefly 板子进行设计的电路，在软件配置上有些许的不同。首先客户在新机器进行板级调试前，先选择对应的 dts 配置，参考以下原则：

- 参考平板以及挖掘机原理图设计的硬件请基于 rk3399-sapphire-excavator-edp.dts 进行配置，如果是 lpddr4 的机器，请在这个 dts 基础上加上 5.8 节描述的内容。
- 参考 firefly 板子原理图进行设计的硬件请基于 rk3399-firefly-android.dts 进行配置，如果是 lpddr4 的机器，请在这个 dts 基础上加上 5.8 节描述的内容。

### 9.1.2 IO Domain配置

硬件上各个 IO domain 如下图所示：

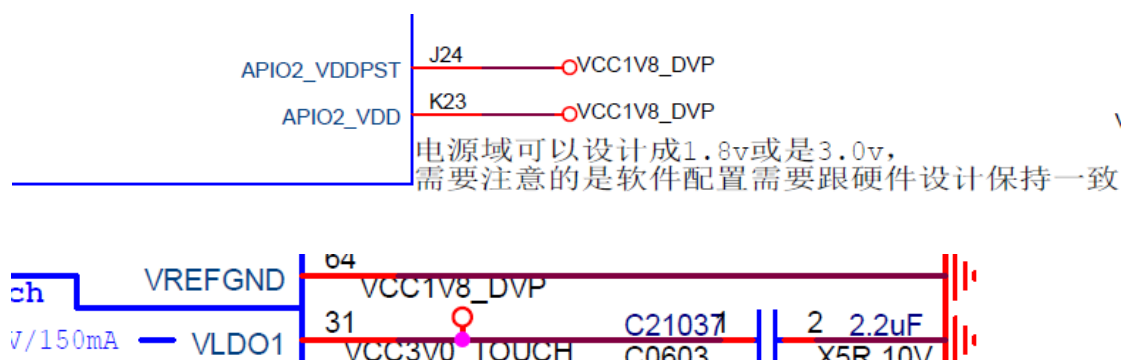
Part Port	Domain	Pin name in datasheet	I/O type
Part C	PMUI01	pmui01_gpio0ab	1.8V only
Part E	PMUI02	pmu1830_gpiolabed	1.8V(Default) 3.0V
Part I	API01	gmac_gpio3abc	3.3V only
Part L	API02	bt656_gpio2ab	1.8V(Default) 3.0V
Part G	API03	wifi/bt_gpio2cd	1.8V only
Part K	API04	gpiol830_gpio4cd	1.8V 3.0V(Default)
Part J	API05	audio_gpio3d_gpio4a	1.8V(Default) 3.0V
Part F	SDMMC0	sdmmc_gpio4b	1.8V 3.0V(Default)

IO Domain 如果配置错误，可能出现一些莫名其妙的问题，在新机器调试前，先确认

io-domain，请参考文档“RKDocs/common/IO-Domain/Rockchip IO-Domain 开发指南 V1.0-20160630.pdf”，并且结合板子电路图进行配置。下面简单描述下配置的方法：

- bt656-supply: The supply connected to APIO2\_VDD.
- audio-supply: The supply connected to APIO5\_VDD.
- sdmmc-supply: The supply connected to SDMMC0\_VDD.
- gpio1830 The supply connected to APIO4\_VDD.

上面列出了几个可配置的 io-domain 对应的芯片管脚名称，例如 bt656 对应的是 APIO2\_VDD，所以需要查询电路图，确认 APIO2\_VDD 连到了 PMU 的哪个 LDO 输出，举例说明：



上图中 APIO2\_VDD 连到了 RK808 的 LDO1 上，RK808 的 LDO1 的定义如下：

```
vcc1v8_dvp: LDO_REG1 {  
    regulator-always-on;  
    regulator-boot-on;  
    regulator-min-microvolt = <1800000>;  
    regulator-max-microvolt = <1800000>;  
    regulator-name = "vcc1v8_dvp";  
    regulator-state-mem {  
        regulator-off-in-suspend;  
    };  
};
```

所以 bt656-supply 要定义为 vcc1v8\_dvp，如下所示：

```
&io_domains {
    status = "okay";

    bt656-supply = <&vcc1v8_dvp>; /* bt656_gpio2ab_ms */
    audio-supply = <&vcca1v8_codec>; /* audio_gpio3d4a_ms */
    sdmmc-supply = <&vcc_sd>; /* sdmmc_gpio4b_ms */
    gpio1830-supply = <&vcc_3v0>; /* gpio1833_gpio4cd_ms */
};
```

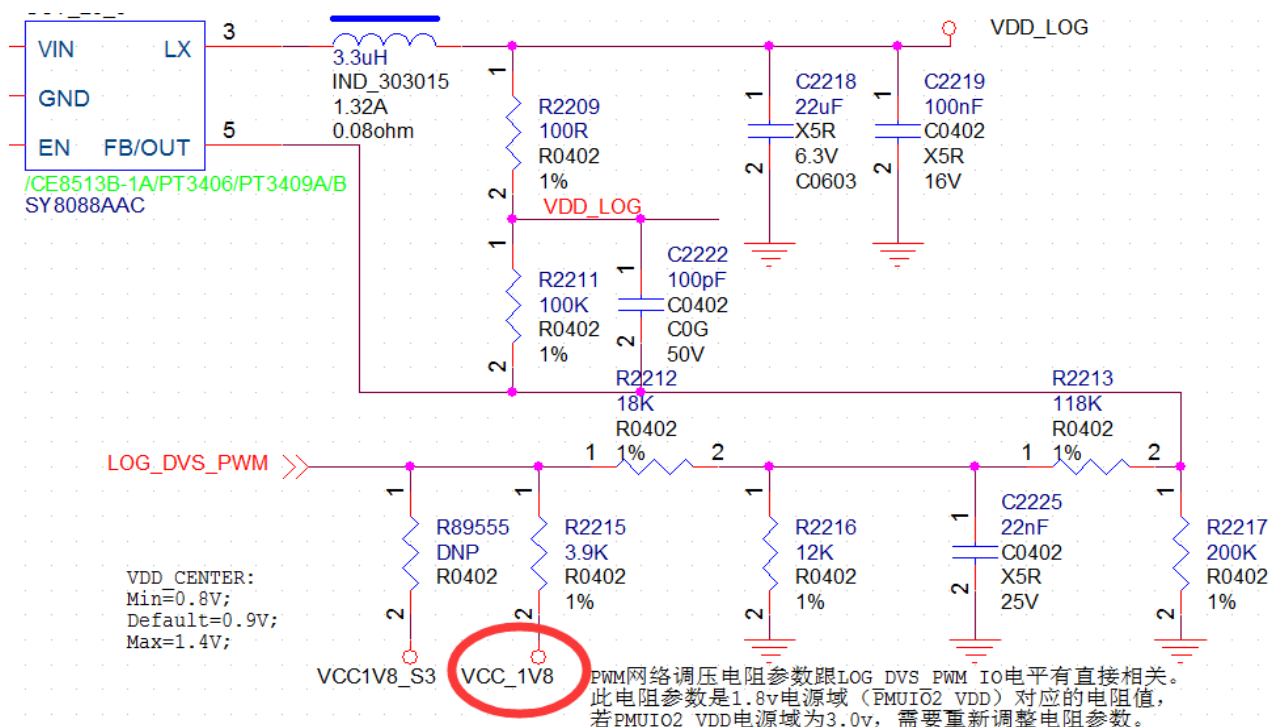
### 9.1.3 Vdd\_log电压差异

VDD\_LOG 电压采用的是 PMUIO2 电源域里面的 PWM (Pin M28) 进行调压。PMUIO2 可配置为 1.8V (参考图) 或 3.0V (挖掘机)，所以 PWM 出来的逻辑电平也分 1.8V 和 3.0V 两种，对应的上拉电阻电源和调压电阻的参数也不一样。

VDD\_LOG 电压我们要求在 0.9V~0.95V 以内，但是不建议太高，太高则功耗很高，由于电阻实际的阻值不是很精确，有可能电压并不是软件配置出的实际值，所以强烈建议在配置完以后量下实际的 vdd\_Log 的值，如果不对再微调下参数。另外上一段中描述的两种硬件设计中，RK808 对应的 io 电压也不一样，这块也需要软件进行修改，以下分别说明：

#### 1. PMUIO 为 1.8V

PMUIO2 配置为 1.8V 的上拉电源和电阻参数，如下图所示：



请参考如下配置：

```
vdd_log: vdd-log {
    compatible = "pwm-regulator";
```

```
pwms = <&pwm2 0 25000 1>;

regulator-name = "vdd_log";

regulator-min-microvolt = <800000>;

regulator-max-microvolt = <1400000>;

regulator-always-on;

regulator-boot-on;

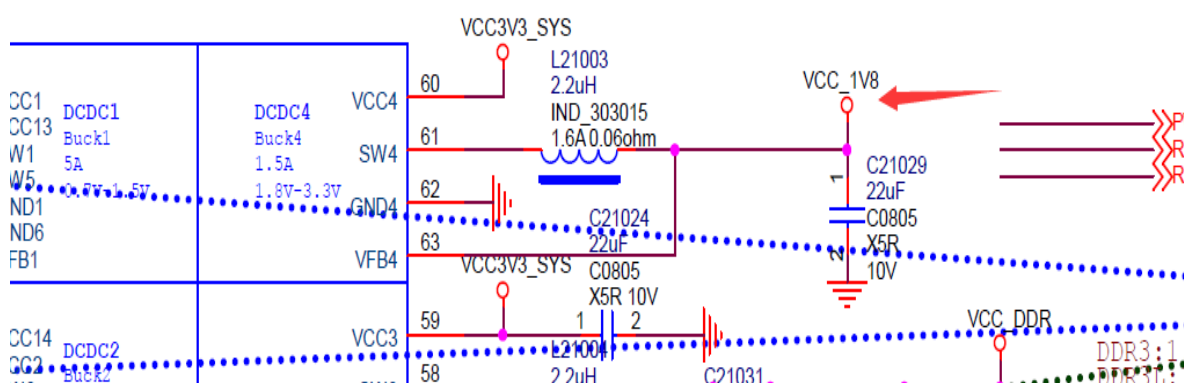
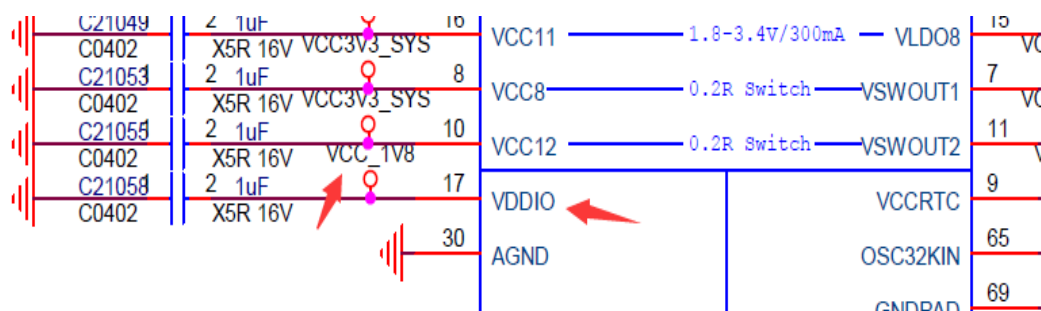
/* for rockchip boot on */

rockchip,pwm_id= <2>;

rockchip,pwm_voltage = <900000>;

};
```

RK808 的 VDDIO 要用 VCC\_1V8 供电，如下图所示：



这里 RK808 的 VDDIO 使用 VCC\_1V8 供电，而 VCC\_1V8 连到了 RK808 的 DCDC4，所以 RK808 的 dts 节点里面配置：

```
vddio-supply = <&vcc_1v8>;
```

vcc\_1v8 对应的电源为 DCDC4：

```
vcc_1v8: DCDC_REG4 {
```

```
regulator-always-on;

regulator-boot-on;

regulator-min-microvolt = <1800000>;

regulator-max-microvolt = <1800000>;

regulator-name = "vcc_1v8";

regulator-state-mem {

    regulator-on-in-suspend;

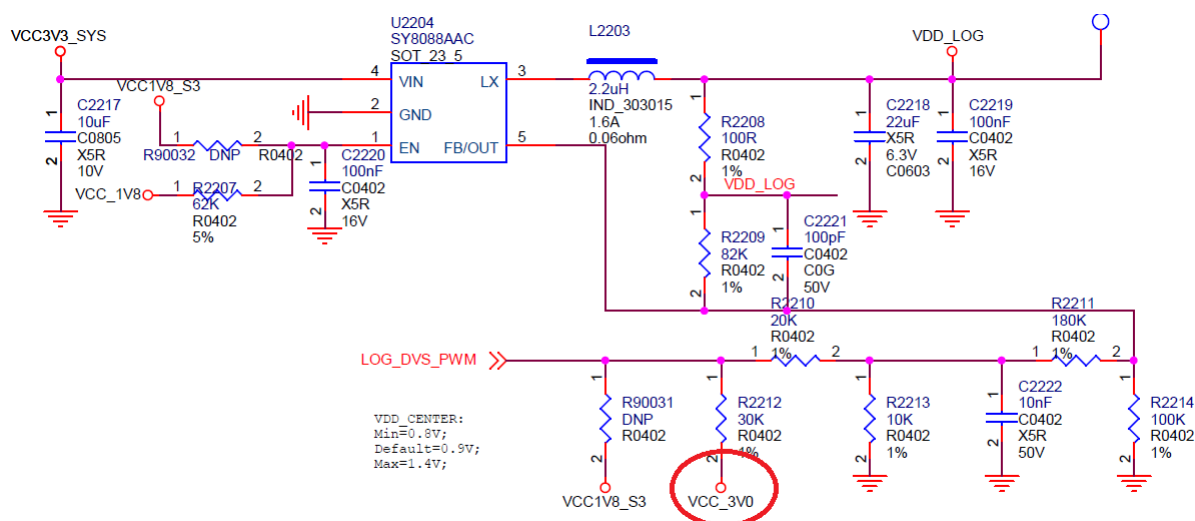
    regulator-suspend-microvolt = <1800000>;

};

};
```

## 2. PMUIO 为 3.3V

PMUIO2 配置为 3.0V 的上拉电源和电阻参数:



请参考如下配置:

```
vdd_log: vdd-log {

    compatible = "pwm-regulator";

    pwms = <&pwm2 0 25000 1>;

    regulator-name = "vdd_log";

    regulator-min-microvolt = <800000>;

    regulator-max-microvolt = <1400000>;
```



```

regulator-always-on;

regulator-boot-on;

/* for rockchip boot on */

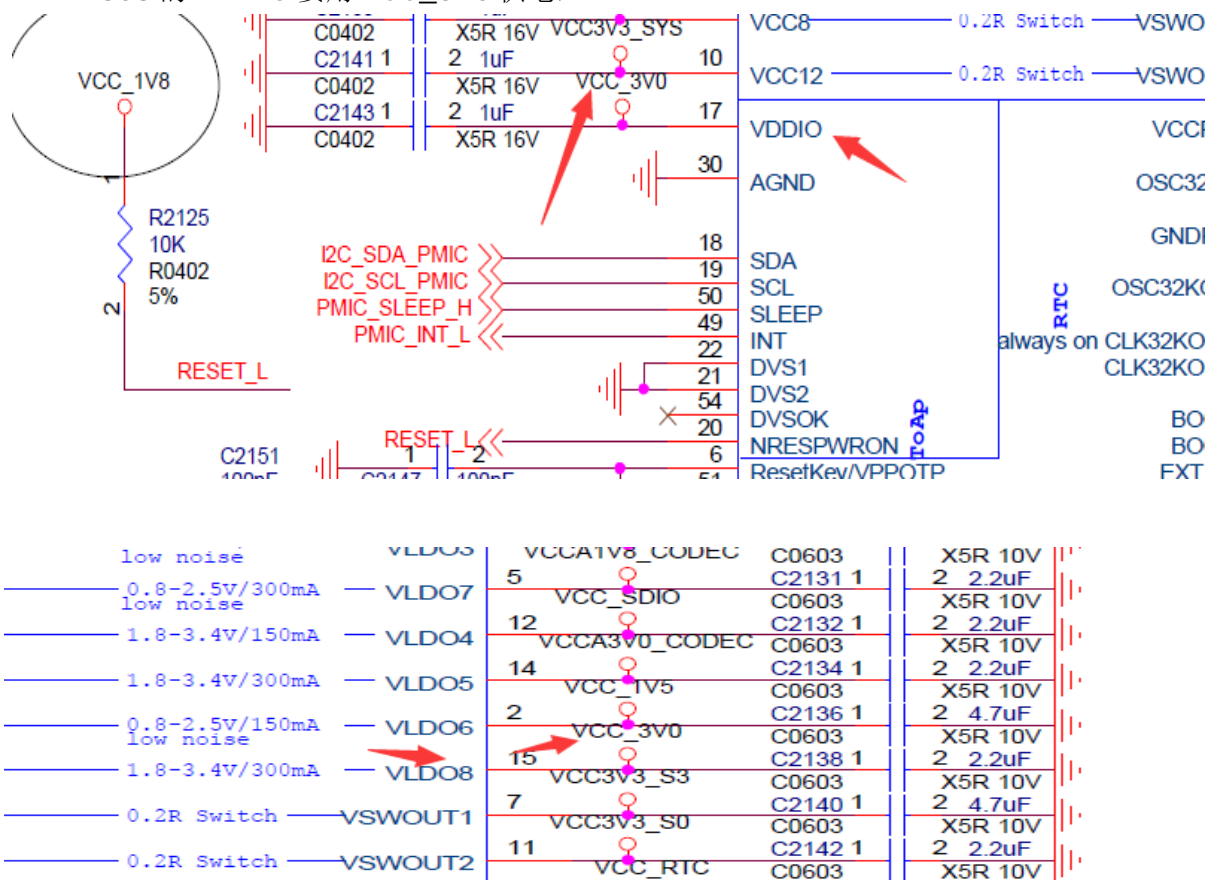
rockchip,pwm_id= <2>;

rockchip,pwm_voltage = <900000>;

};

```

RK808 的 VDDIO 要用 VCC\_3V0 供电:



这里 RK808 的 VDDIO 使用 VCC\_3V0 供电，而 VCC\_3V0 连到了 RK808 的 LDO8，所以 RK808 的 dts 节点里面配置：

```
vddio-supply = <&vcc_3v0>;
```

vcc\_3v0 对应的电源为 LDO8:

```

vcc_3v0: LDO_REG8 {
    regulator-always-on;
    regulator-boot-on;
    regulator-min-microvolt = <3000000>;
    regulator-max-microvolt = <3000000>;
};

```

```
regulator-name = "vcc_3v0";  
regulator-state-mem {  
    regulator-on-in-suspend;  
    regulator-suspend-microvolt = <3000000>;  
};  
};
```

## 9.2 常见稳定性问题分析

本章主要介绍 RK3399 Android7.1 行业 SDK 中碰到的一些常见的稳定性问题以及解决思路和方法。

稳定性问题分成两类，一类是代码中确实存在软件 bug；一类是随机死机问题，下面分开描述。

### 9.2.1 软件BUG导致死机或者卡死问题

在某些特定场景下出现死机或者卡住问题，并且这类死机有着固定的死机 log 或者没有任何异常 log，这类问题需要分析具体的 log 确认问题点，下面主要介绍可能存在的类别并且需要抓取的 log。

#### 1. 内核的死机

如果内核的死机每次都是相同的 log，需要抓取 kernel log（串口 log 或者 last log），并且保留死机版本对应内核编译出来的 vmlinux（在 kernel 根目录下），然后提供给 RK 进行分析。

#### 2. 内核驱动死锁

这种场景下内核的 log 有可能会有一些堆栈打印出来（接串口才能打印），另外还需要提供以下信息，在串口终端中输入“fiq”三个字符，会进入 fiq 模式，在该模式下连续输入几次“sysrq w”，从这些 log 可能可以看出一些异常，如果无法自行定位，请将打印得到的信息以及内核 log 一起发给 RK 进行确认。

#### 3. Android 上层的死锁或者某些库崩溃问题

该场景下需要一些 android 端的 log 来定位问题，出现此问题时，请抓取 aplog，aplog 默认开启，位于机器中的/data/logs/目录，请将该目录全部 pull 出来（同时抓取出问题时的 logcat）。另外，一些 android 死锁问题会产生 ANR 信息，请获取/data/anr/traces.txt 文件进行分析，如果出现此类问题，并且无法自行定位，请将上述所有信息发送给 RK 进行分析和定位。更详细的调试方法请参考文档：[RKDocs/android/死机问题快速分析指南\\_V1.0\\_20190312.pdf](#)

### 9.2.2 硬件相关导致的软件死机问题

如果在调试过程中碰到一些莫名其妙的死机问题，并且这些死机问题出现时，log 中的堆栈是随机的，或者死机的点在内核很核心的函数，或者 Android 某些核心库出现非法指针，则需要关注以下几个方面：

#### 1. DDR 相关

RK3399 支持 DDR3、DDR3L、LPDDR3、LPDDR4 多种颗粒，在 DDR 选型上，请关注 RK FAE 窗口定期更新的 DDR 支持列表，根据里面的型号进行选型。如果存在一些特殊情况，确实需要使用某个颗粒，并且不在支持列表中，请联系 FAE 窗口协调新颗粒的调试事宜。如果有在支持列表，请确认更新最新的 loader 和 trust 是否能够解决问题，如果 u-boot 使用默认分支，更新 u-boot 即可，如果有参考本文档 4.7 节切换 next-dev 分支的 u-boot，则更新 rkbin 即可更新 loader 和 trust。

另外，确保内核有按照本文档的 5.8 节进行配置，目前 LPDDR4 只支持 400M、800M 两个频点，因此需要注意 LPDDR4 与 DDR3/DDR3L/LPDDR3 的频点不同，LPDDR4 的频率表中不能包含 400M、800M 以外的频点，否则会导致系统异常。

如果软件上无法定位到原因，请将问题登记到 RK redmine 系统，并且发出 pcb 和原理图，申请 RK 硬件进行 review，可能是信号或者电源上的异常导致。

#### 2. IO domain 配置错误

请参考 9.1.2 节进行配置。

#### 3. 电源的异常

如果出现随机性死机问题，并且上述章节中的内容均已确认无误，则建议量下各路供电是否有异常，请重点排查以下几个部分：

- CPU 频率及电压

- 1) 请勿随意修改 RK 提供的频率电压表；
- 2) 早期的行业 SDK 版本在低温情况下会出现概率性死机问题，如果发现版本比较旧，请更新到 V1.4 以后的版本（强烈建议代码更新到最新）；
- 3) CPU 供电的纹波，如果纹波比较大，则需要将电压适当往上抬，但是需要注意，电压的增加是

以 12.5mV 为单位，不能随意增减电压值，因为 PMIC 无法输出非 12.5mV 倍数的电压；

- Logic 电压 (vdd\_log)

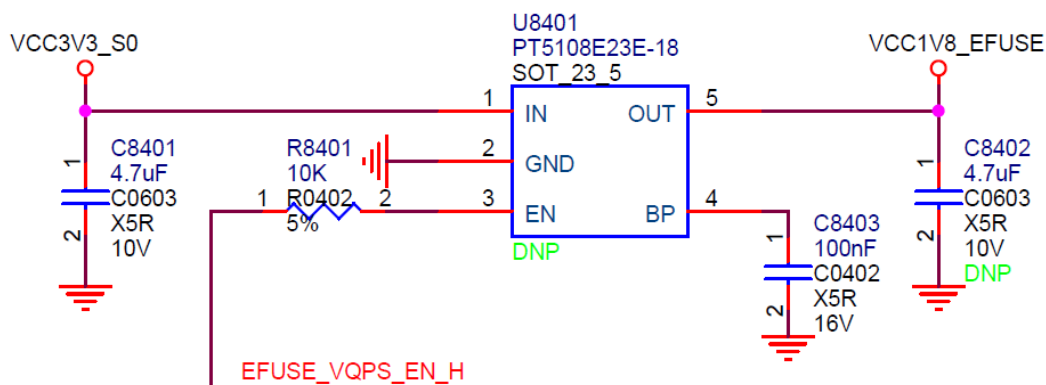
Logic 电压与 IO Domain 电压配置有关，强烈建议客户在板级调试的时候都量下 vdd\_Log 的实际电压，确保这个电压在 0.9V 以上，具体请参考 9.1.3 节进行配置和确认。

- DDR 电压

确保 DDR 颗粒供电符合颗粒要求。

### 9.3 Secure Boot常见问题

烧写 efuse 以及对固件进行签名和固件烧写的说明请参考“RKDocs/rk3399/RK3399\_Efuse\_Operation\_Instructions\_xxx.pdf”文档进行操作，需要注意的是，efuse 的烧写需要确保下图中的 VCC1V8\_EFUSE 供电正常，该 LDO 的控制脚软件上是固定的 (EFUSE\_VQPS\_EN\_H)，为 GPIO4\_D3，不能改变。



### 9.4 Data分区切换为EXT4后出现的问题

DATA 分区切换为 EXT4 格式存在两个问题需要考虑：

- 量产客户不建议将 data 分区切为 EXT4 格式，因为切换分区格式会导致 data 分区格式化，并且 ota 升级存在问题。
- Recovery 擦除分区慢问题。

以上具体说明请参考本文档 6.23 节内容进行修改。

### 9.5 更新代码后必现卡在android动画问题

这种情况比较大的可能有两个：

1. SDK Android 和 kernel 都有更新，但是客户只更新了 kernel 或者只更新了 android。

2. 客户的内核自己有新建一个 defconfig, 导致 SDK 对于内核 defconfig 的更新没有同步到客户自己新建的 defconfig 上, 需要客户把 arch/arm64/configs/rockchip\_defconfig 上更新的内容同步到自己的 defconfig 上。

上述的问题其实都是因为 GPU 库的上层和下层不匹配导致 android 无法启动, 这个是目前碰到比较多的情况, 如果不是上述所说问题, 请根据 log 具体分析。

## 9.6 USB3.0 外设待机唤醒后重新枚举问题

目前系统默认在待机唤醒后, usb3.0 的外设会重新枚举, 这个本身不会有问题。但是有些客户使用的外设, 如果重新枚举, 所有的流程要重新跑一遍, 导致用户体验不佳, 例如有些 4G 模组, 由于在待机唤醒后需要重新枚举, 所有的联网操作要重新跑一遍(重新拨号并联网的操作比较耗时), 可能最终需要几分钟才能连上网络。如果客户选择的外设存在这个问题, 请更新到 sdk V2.1 版本, 并且打上下面的补丁:

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3399.dtsi b/arch/arm64/boot/dts/rockchip/rk3399.dtsi
index bf5d7c3..d02b186 100644
--- a/arch/arm64/boot/dts/rockchip/rk3399.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3399.dtsi
@@ -407,7 +407,7 @@
        #address-cells = <2>;
        #size-cells = <2>;
        ranges;
-       needs-reset-on-resume;
+       //needs-reset-on-resume;
        status = "disabled";
        usbdrd_dwc3_0: dwc3@fe800000 {
                compatible = "snps,dwc3";
@@ -441,7 +441,7 @@
```

```
#address-cells = <2>;

#size-cells = <2>;

ranges;

-      needs-reset-on-resume;
+      //needs-reset-on-resume;

status = "disabled";

usbdrd_dwc3_1: dwc3@fe900000 {

    compatible = "snps,dwc3";
```

## 9.7 LPDDR4 开启负载变频

从 SDK V2.1 版本开始默认开启 lpddr4 负载变频，负载变频可以较为显著的降低 lpddr4 的功耗，建议更新到 V2.1 版本以后再开启负载变频，负载变频配置方法如下述红字部分：

```
&dmc {

    status = "okay";

    center-supply = <&vdd_center>;

    system-status-freq = <

        /*system status      freq(KHz)*/

        SYS_STATUS_NORMAL      800000

        SYS_STATUS_REBOOT      400000

        SYS_STATUS_SUSPEND      400000

        SYS_STATUS_VIDEO_1080P  800000

        SYS_STATUS_VIDEO_4K     800000

        SYS_STATUS_VIDEO_4K_10B 800000

        SYS_STATUS_PERFORMANCE  800000

        SYS_STATUS_BOOST        800000

        SYS_STATUS_DUALVIEW     800000

        SYS_STATUS_ISP          800000
```

```
>;  
  
vop-bw-dmc-freq = <  
  
/* min_bw(MB/s) max_bw(MB/s) freq(KHz) */  
  
0      577      400000  
  
578     99999     800000  
  
>;  
  
auto-min-freq = <400000>;  
  
auto-freq-en = <1>;  
  
};
```

LPDDR4 的负载变频在 sdk 端已经测试过绝大多数场景，但是由于行业客户的应用场景多样化，我们无法覆盖所有场景，请在开启负载变频的时候多测试下，确认是否有 ddr 带宽不足导致的闪屏问题。