Rockchip
瑞芯微电子

Rockchip Linux
软件开发指南

发布版本:1.06

日期:2019.02

免责声明

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

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

商标声明

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

本文档可能提及的其他所有注册商标或商标，由其各自拥有者所有。

版权所有 © 2019 福州瑞芯微电子股份有限公司

福州瑞芯微电子股份有限公司
Fuzhou Rockchip Electronics Co., Ltd.
地址： 福建省福州市铜盘路软件园 A 区 18 号
网址： www.rock-chips.com
客户服务电话： +86-591-83991906
客户服务传真： +86-591-83951833
客户服务邮箱： fae@rock-chips.com

前言

概述

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

产品版本

芯片名称	内核版本	Buildroot 版本
PX30	Linux4.4	2018.02_rc3
RK3308	Linux4.4	2018.02_rc3
RK3326	Linux4.4	2018.02_rc3
RK3399	Linux4.4	2018.02_rc3
RK1808	Linux4.4	2018.02_rc3
RK3399PRO	Linux4.4	2018.02_rc3

读者对象

本文档（本指南）主要适用于以下工程师：
技术支持工程师
软件开发工程师

修订记录

日期	版本	作者	审核	修改说明
2018-05-10	V1.00	yhx	cch	正式发布
2018-05-28	V1.01	yhx	cch	1. 更新 DDR 支持列表，详见 1.1 章节。 2. 更新 EMMC 支持列表，详见 1.2 章节。 3. 更新 SPI Nor 及 SLC Nand 支持列表，详见 1.3 章节。 4. 更新 SDK 文档/工具索引，详见 2 章节。 5. 更新分区表 data 分区改为 oem 分区，cfg 分区改为 userdata 分区，详见 6.3 章节。 6. 更新 upgrade_tool 烧写命令，详见 7.4 章节。 7. Flash 类型默认配置为 Nand，详见 10.1.7.1 章节。 8. 新增 Rootfs 切换为 ext2 文件系统说明，详见 10.1.7.2 章节。 9. 修改预置系统应用或数据说明，详见 10.3 章节。 10. 更新 WiFi/BT 开发指南，详见 10.8 章节。 11. 新增 DLNA 使用及开发说明，详见 10.11.1 章节。 12. 新增 RK3308 DuerOS 使用说明，详见 10.13.1 章

				节。 13.更新 11.1.4 章节 ADB 窗口乱码显示。 14.新增 PCBA 测试工具说明，详见 12.2 章节。
2018-07-10	v1.02	wxt		1.增加 rk3399/rk3326/px30 的芯片使用说明
2018-07-16	v1.03	yhx		1.更新全自动编译脚本存放位置，详见 6.4 章节。 2.新增模块部分编译说明，详见 6.5 章节。 3.新增 misc 和 recovery 分区烧写描述，详见 7.4 章节。 4.新增 OTA 升级说明，详见 10.5 章节。 5.新增 RK3308 EVB V12 配置说明，与 V11 兼容，详见 6.2 章节。 6. 新增 RK3308 语音模块板 32bit 编译说明，详见 6.2 章节。
2018-11-05	V1.04	cww		1. 修正 5.5 节目录名错误 2. 修正 6.2 节编译命令错误 3. 修正 7.6 节目录名错误 4. 修正 9.1.1 描述问题 5. 修正 11.1.3 ADB 功能描述问题 6. 修正 11.5 last_log 描述问题 7. 修正 12.3 固件统一打包的描述问题
2018-11-25	V1.05	hl		1.增加 rk1808 使用说明
2019-02-17	V1.06	wxt		1. 增加 rk3399pro 使用说明 2. 修正 2.1 工具索引中 dvfs/rk3399pro 3. 修正 4.2.3 依赖包的添加 4. Copyright2018 改为 Copyright2019
2019-06-20	V1.07	wxt		1. 新增 9.3~9.7 章节，主要新增 GPIO/DVFS/温控 DDR4/SDCARD 配置的说明 2. 更新客户服务邮箱

目录

插图目录

表格目录

1 支持列表

1.1 DDR 支持列表

Rockchip 平台 DDR 颗粒支持列表，详见 docs\Platform support lists 目录下《RK DDR Support List V2.32 20180525.pdf》，下表中所标示的 DDR 支持程度表，只建议选用 √、T/A 标示的颗粒。

表 1-1 Rockchip 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

Rockchip 平台 eMMC 颗粒支持列表，详见 docs\Platform support lists 目录下《RKeMMCSupportList Ver1.39_20180515.pdf》，下表中所标示的 EMMC 支持程度表，只建议选用 √、T/A 标示的颗粒。

表 1-2 Rockchip EMMC Support Symbol

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

1.2.1 高性能 EMMC 颗粒的选取

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

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

如有选型上的疑问，也可直接联系 Rockchip Fae 窗口<fae@rock-chips.com>。

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 SPI Nor 及 SLC Nand 支持列表

Rockchip 平台 SPI Nor 及 SLC Nand 支持列表，详见 docs\Platform support lists 目录下《RK SpiNor and SLC Nand SupportList Ver1.07_20180515.pdf》，文档中也有标注 SPI Nand 的型号，可供选型。下表中所标示的 EMMC 支持程度表，只建议选用 √、T/A 标示的颗粒。

表 1-3 Rockchip SPI Nor and SLC Nand Support Symbol

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

1.4 WiFi/BT 支持列表

Rockchip 平台 WiFi/BT 支持列表，详见 docs\Platform support lists 目录下《Rockchip_Linux_SDK_WiFi_Situation_20180507.pdf》，文档列表中为目前 Rockchip 平台上大量测试过的 Wifi/Bt 芯片列表，建议按照列表上的型号进行选型。如果有其他 WiFi/BT 芯片调试，需要 WiFI/BT 芯片原厂提供对应内核驱动程序。

成熟度详见文档中说明：

Perfect > Very Good > Good > Preliminary > X

如有选型上的疑问，建议可以与 Rockchip Fae 窗口<fae@rock-chips.com>联系。

1.5 SDK 软件包适用硬件列表

随 SDK 软件包发布，Rockchip 平台会有对应的参考硬件，硬件用户使用指南主要介绍参考硬件板基本功能特点、硬件接口和使用方法。旨在帮助相关开发人员更快、更准确地使用该 EVB，进行相关产品的应用开发。

表 1-4 Rockchip 平台硬件使用指南列表

平台	硬件板型	对应文档说明
PX30	Evb 开发板	docs\ SoC platform related\PX30\PX30 MINI EVB 硬件操作指南_20180710.pdf
RK3308	Evb 开发板	docs\ SoC platform related\RK3308\RK3308_EVB 用户使用指南 V001_201805.pdf
RK3399	Evb 开发板	docs\ SoC platform related\RK3399\Rockchip RK3399 挖掘机用户使用指_V2.0_20180424.pdf
RK1808	Evb 开发板	docs\ SoC platform related\RK1808\RK1808 EVB 用户指南_V10_20181226.pdf
RK3399PRO	Evb 开发板	docs\ SoC platform related\RK3399PRO\RK3399PRO 硬件设计指南及原理图 PCB 评审注意事项_V10_20190111

2 文档/工具索引

2.1 文档索引

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

Rockchip Linux SDK 中在 docs 目录下附带了 Develop reference documents（开发指导文档）、Platform support lists（支持列表）、RKTools manuals（工具使用文档）、SoC platform related（芯片平台相关文档）、Linux reference documents（Linux 系统开发指南）。

```
docs
├── Develop reference documents
│   ├── Audio
│   │   └── Rockchip Audio 开发指南 V1.1-20170215-linux4.4.pdf
│   ├── CAMERA
│   │   ├── CIF_ISP10_Driver_User_Manual_v1.0.pdf
│   │   ├── RK_ISP10_Camera_User_Manual_v2.2.pdf
│   │   └── Rockchip_Camera_AVL_v2.0_Package_20180515.7z
│   ├── CRU
│   │   └── Rockchip 时钟子模块 开发指南 V1.1-20170210.pdf
│   ├── DDR
│   │   ├── DDR 开发指南.pdf
│   │   ├── DDR 问题排查手册 1.0.pdf
│   │   ├── DDR 颗粒验证流程说明.pdf
│   │   └── RK DDR Application Note 5 --客户自行验证 DRAM 新料流程.pdf
│   ├── DEBUG
│   │   ├── perf 使用说明.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
│   ├── DVFS
│   │   ├── Linux 4.4 内核 DVFS 介绍.pptx
│   │   ├── Rockchip-Developer-Guide-Linux4.4-CPUFreq-CN.pdf
│   │   └── Rockchip-Developer-Guide-Linux4.4-Devfreq.pdf
│   ├── GMAC
│   │   └── Rockchip 以太网 开发指南 V2.3.1-20160708.pdf
│   ├── I2C
│   │   └── Rockchip-Developer-Guide-Linux-I2C.pdf
│   ├── IO-DOMAIN
│   │   └── Rockchip IO-Domain 开发指南 V1.0-20160630.pdf
│   ├── MCU
│   │   └── Rockchip-Developer-Guide-linux4.4-MCU.pdf
```

```
|   ├── MMC
|   |   └── Rockchip-Developer-Guide-linux4.4-SDMMC-SDIO-eMMC.pdf
|   ├── MPP
|   |   └── MPP 开发参考_v0.3.pdf
|   ├── NPU
|   |   ├── RK1808_RKNN_SDK 开发指南_V0.9.6.pdf
|   |   └── RKNN-Toolkit 使用指南_V0.9.6.pdf
|   ├── NVM
|   |   ├── appnote rk flash storage 2017.09.06.pdf
|   |   ├── appnote rk vendor storage 2017.01.23.pdf
|   |   └── Rockchip-Secure-Boot-Application-Note.pdf
|   ├── PCIe
|   |   ├── PCIe Protocol And Linux Driver Introduction.pdf
|   |   └── Rockchip-Developer-Guide-linux4.4-PCIe.pdf
|   ├── PERF
|   |   ├── perf 使用说明.pdf
|   |   ├── streamline 使用说明.pdf
|   |   └── systrace 使用说明.pdf
|   ├── Pin-Ctrl
|   |   ├── Rockchip GPIO 常见问题.pdf
|   |   └── Rockchip Pin-Ctrl 开发指南 V1.0-20160725.pdf
|   ├── PMIC
|   |   ├── Rockchip-Developer-Guide-Linux4.4-DC-DC.pdf
|   |   ├── Rockchip-Developer-Guide-RK805.pdf
|   |   ├── Rockchip-Developer-Guide-RK818_6-Fuel-Gauge.pdf
|   |   ├── Rockchip RK805 开发指南 V1.0-20170220.pdf
|   |   ├── Rockchip RK818_6 电量计 开发指南 V2.0-20170525.pdf
|   |   ├── Rockchip-RK818-RK816-FG-Log-Description-linux4.4.pdf
|   |   └── 小系统时钟方案概述.pptx
|   ├── PWM
|   |   └── Rockchip 背光控制 开发指南 V0.1-20160729.pdf
|   ├── S2R
|   |   ├── Rockchip 休眠唤醒 开发指南 V0.1-20160729.pdf
|   |   ├── 休眠唤醒流程.pdf
|   |   └── 休眠唤醒相关问题定位.pdf
|   ├── SECURITY
|   |   ├── Rockchip-Secure-Boot-Application-Note-V1.9.pdf
|   |   └── Rockchip TEE 安全 SDK 开发指南 V1.0.pdf
|   ├── SPI
|   |   └── Rockchip-Developer-Guide-linux4.4-SPI.pdf
|   ├── THERMAL
|   |   └── Rockchip Thermal 开发指南-4.4 V1.0.1-20170428.pdf
|   ├── TOOL
|   |   ├── Production-Guide-For-Firmware-Download.pdf
|   |   ├── RK3288 JTAG 配置指南 V1.0-20170316.pdf
|   |   ├── RK3399 JTAG 配置指南 V1.1-20170327.pdf
```

```
|   |   ├── RK3399-LOG-EXPLANATION.pdf
|   |   ├── rk_fw_upgrade_issue.pdf
|   |   ├── RKUpgrade_Dll_UserManual.pdf
|   |   └── Rockchip-Parameter-File-Format-Version1.4.pdf
|   ├── TRUST
|   |   ├── armv8 trusted firmware-说明 V1.0-20170525.pdf
|   |   └── Rockchip-Developer-Guide-Trust.pdf
|   ├── UART
|   |   └── Rockchip-Developer-Guide-linux4.4-UART.pdf
|   ├── U-Boot
|   |   ├── Rockchip-Developer-Guide-UBoot-nextdev.pdf
|   |   └── Rockchip-Developer-Guide-UBoot-rkdevelop-vs-nextdev.pdf
|   └── USB
|       ├── RK3399-USB-DTS.pdf
|       ├── Rockchip-Developer-Guide-linux4.4-USB.pdf
|       ├── Rockchip-USB-FFS-Test-Demo.pdf
|       ├── Rockchip-USB-Performance-Analysis-Guide.pdf
|       ├── Rockchip-USB-SQ-Test-Guide.pdf
|       └── USB-Initialization-Log-Analysis.pdf
├── Linux reference documents
|   ├── camera_engine_rkisp_user_manual_v1.0.pdf
|   ├── Linux QT 应用开发介绍.pdf
|   ├── Linux rga 说明文档.pdf
|   ├── Linux SDK 压力测试方法.pdf
|   ├── mipi-dsi.ppt
|   ├── RK3308_RTL8723DS_WIFI_BT_说明文档_V1.20.pdf
|   ├── RK_linux_camera_gstreamer_应用开发_v1.0_20171212.pdf
|   ├── RK_Linux_SDK 编译开发环境搭建.pdf
|   ├── Rockchip Debian 双屏异显音频使用说明.pdf
|   ├── Rockchip DLNA 开发指南_V1.00.pdf
|   ├── Rockchip_Gstreamer 使用说明.pdf
|   ├── Rockchip Linux DS5 性能调试工具 Streamline 使用说明.pdf
|   ├── Rockchip Linux WIFI BT 开发指南 V1.1 20180622.pdf
|   ├── Rockchip Linux 升级固件打包指南.pdf
|   ├── Rockchip recovery 开发指南 V1.0.3.pdf
|   ├── Rockchip ROS 使用指南.pdf
|   ├── Rockchip secureboot 使用指南.pdf
|   ├── Rockchip Linux Audio 开发介绍.pdf
|   ├── Rockchip Linux Camera 开发指南.pdf
|   ├── Rockchip Linux SDK uboot logo 显示开发指南.pdf
|   ├── Rockchip PCBA 测试开发指南_1.05.pdf
|   ├── Rockchip 分片升级开发指南-V1.0.0.pdf
|   └── The Buildroot User Manual.pdf
├── Platform support lists
|   ├── RK DDR Support List Ver2.34.pdf
|   ├── RKeMMCSupportList Ver1.41_20181030.pdf
```

```
|   ├── RKeMMCSupportList Ver1.42_2018_11_30.pdf
|   ├── RKNandFlashSupportList Ver2.73_20180615.pdf
|   ├── RK SpiNor and  SLC Nand SupportList Ver1.10_2018_1101.pdf
|   ├── Rockchip_Camera_Module_AVL_v2.0.pdf
|   └── Rockchip_Linux_SDK_WiFi_Situation_20180507.pdf
├── RKTools manuals
|   ├── Android 增加一个分区配置指南 V1.00.pdf
|   ├── Linux 开发工具使用手册_v1.32.pdf
|   ├── REPO 镜像服务器搭建和管理_V2.2_20131231.pdf
|   ├── RK3308_EQ_DRC_TOOL_v1.22.pdf
|   ├── Rockchip-Parameter-File-Format-Version1.4.pdf
|   ├── Rockchip Bug 系统使用指南 V1.0-201712.pdf
|   ├── Rockchip 量产烧录 指南 V1.1-20170214.pdf
|   ├── SecureBoot 开发工具指南.pdf
|   ├── WNpctool 简要使用说明_V1.1.0_0920.pdf
|   └── 瑞芯微开发工具手册_v1.1.pdf
├── Rockchip Linux 软件开发指南.pdf
└── SoC platform related
    ├── PX30
    |   ├── io_list_v0.1_for evb&ref 20180112.xlsx
    |   ├── PX30_LINUX_BETA_V0.2_20180803 发布说明.pdf
    |   ├── PX30 Linux SDK Release Note.pdf
    |   ├── PX30 MINI EVB 硬件操作指南_20180710.pdf
    |   ├── rk_evb_px30_ddr3p416dd6_v10.pdf
    |   └── Rockchip PX30 Datasheet V1.1-20180918.pdf
    ├── PX3SE
    |   ├── PX3SE_Linux_Beta_V0.3_20180817 发布说明.pdf
    |   ├── PX3SE Linux SDK Release Note.pdf
    |   ├── RK312x Multimedia Codec Benchmark v1.1.pdf
    |   └── Rockchip PX3 SE Datasheet V1.2-20171226.pdf
    ├── RK1808
    |   ├── RK1808 EVB 用户使用指南_V10_20181226.pdf
    |   ├── RK1808_RKNN_SDK 开发指南_V0.9.6.pdf
    |   ├── RKNN-Toolkit 使用指南_V0.9.6.pdf
    |   └── Rockchip RKNN_DEMO 模块开发指南 V0.1.pdf
    ├── RK3288
    |   ├── RK3288 Linux SDK Release Note.pdf
    |   ├── Rockchip RK3288 Datasheet V2.4-20180530.pdf
    |   ├── Rockchip_RK3288 Linux_SDK_V2.0_发布说明_20180620.pdf
    |   └── Rockchip_RK3288 Linux_SDK_V2.1_发布说明_20180926.pdf
    ├── RK3308
    |   ├── RK3308 Asound 配置说明_V1.20.pdf
    |   ├── RK3308 Audio Codec Benchmark v1.0.pdf
    |   ├── RK3308_Audio_Codec_Introduction_v0.3.0_CN.pdf
    |   ├── RK3308 DuerOS 使用说明 V1.00_20180929.pdf
    |   ├── RK3308_EVB 用户使用指南 V001_201805.pdf
```

```
|   ├── RK3308 input-event-daemon 说明文档.pdf
|   ├── RK3308_Linux_SDK_For_Robot_V1.00_20181208 发布说明.pdf
|   ├── RK3308 SD 卡&U 盘 开发指南_V1.0_20180907.pdf
|   ├── RK3308_VAD_寄存器配置_v1.1.pdf
|   ├── RK3308 VAD+讯飞识别 demo 说明文档_v1.1.pdf
|   ├── RK3308 LED 控制 Demo 说明文档.pdf
|   ├── RK3308 本地播放器 Demo 说明文档.pdf
|   ├── RK3308 思必驰语音 SDK 使用说明 V0.01_20180607.pdf
|   ├── RK3308 语音模块 32 位系统编译指南_v1.40.pdf
|   ├── RK3308_音频增益调节_说明文档 V1.1.pdf
|   ├── Rockchip RK3308 Datasheet V1.4-20180812.pdf
|   └── Rockchip_硬件 VAD 模块_使用文档_V1.0.pdf
├── RK3326
|   ├── RK3326 EVB 开发板用户指南 v10_20180705.docx
|   ├── RK3326 EVB 开发板用户指南 v10_20180705.pdf
|   ├── RK3326_LINUX_SDK_BETA_20180803 发布说明.pdf
|   ├── RK3326 Linux SDK Release Note.pdf
|   ├── RK_Linux_SDK 编译开发环境搭建.pdf
|   └── Rockchip RK3326 Datasheet V1.1 20180411.pdf
├── RK3328
|   ├── RK3328 BOX EVB 硬件使用指南 V1.0_20170223.pdf
|   ├── RK3328_Linux_Release_Note.pdf
|   ├── RK3328_Linux_Release_V1.0_20181102 发布说明.pdf
|   ├── RK_Linux_SDK 编译开发环境搭建.pdf
|   └── Rockchip RK3328 Datasheet V1.2-20180205.pdf
├── RK3399
|   ├── RK3399_Linux_NN_SDK_V1.0_20180713 发布说明.pdf
|   ├── RK3399 Linux SDK Release Note.pdf
|   ├── RK3399_LINUX_SDK_V2.09_20181102 发布说明.pdf
|   ├── RK3399 Multimedia Codec Benchmark v1.0.pdf
|   ├── RK3399_SSD_Android&Linux_V1.0_20180522.pdf
|   ├── Rockchip-Developer-Guide-linux4.4-MCU.pdf
|   ├── Rockchip RK3399 Datasheet V1.8-20180529.pdf
|   └── Rockchip RK3399 挖掘机用户使用指南_V2.0_20180424.pdf
└── RK3399PRO
    ├── RK3399Pro_EVB 板简介_20181212.pdf
    ├── RK3399PRO_IO_LIST_V11_for EVB 20181203
    ├── RK3399Pro 硬件设计指南及原理图 PCB 评审注意事项_V10_2019*/
    ├── RK3399PRO_LINUX_SDK_V0.01_20190217 发布说明.pdf
    ├── RK3399PRO_NPU 上电启动.pdf
    └── RK3399PRO-RKNN_DEMO 模块开发指南.pdf
```

2.2 工具索引

随 Rockchip Linux SDK 发布的工具，用于开发调试阶段及量产阶段。工具版本会随 SDK 更

新不断更新，如有工具上的疑问及需求，请联系我们的 Fae 窗口<fae@rock-chips.com>。

Rockchip Linux SDK 中在 tools 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）。

```
tools/
├── linux
│   ├── Linux_Pack_Firmware（Linux 固件打包工具）
│   ├── Linux_SecureBoot（Linux 固件签名工具）
│   ├── Linux_TA_Sign_Tool.rar
│   ├── Linux_Upgrade_Tool（Linux 开发工具）
└── windows
    ├── AndroidTool（开发工具）
    │   ├── AndroidTool_Release_v2.52
    │   └── rockdev（固件打包工具）
    ├── DriverAssitant_v4.7.zip（驱动安装助手）
    ├── PCBATool_Setup_V1.0.6_0516_3308.exe（PCBA 测试 PC 端工具）
    ├── efuse_v1.37.zip（Efuse 烧写工具）
    ├── FactoryTool_v1.61.zip（工厂量产工具）
    ├── SecureBootTool_v1.89.zip（固件签名工具）
    ├── SpiImageTools_v1.36.zip
    ├── EQ_DRC_TOOL_v1.1.zip（语音音效实时调参工具）
    └── WNpctool_Setup_V1.1.9_180118.zip（厂商信息烧写工具）
```

3 SDK 软件架构

3.1 SDK 概述

Rockchip Buildroot Linux SDK 是基于 Buildroot-2018.02 的版本的软件开发包，其包含了基于 Linux 系统开发用到的各种系统源码，驱动，工具，应用软件包。

Buildroot 是 Linux 平台上一个开源的嵌入式 Linux 系统自动构建框架。整个 Buildroot 是由 Makefile 脚本和 Kconfig 配置文件构成的。你可以通过 Buildroot 配置，编译出一个完整的可以直接烧写到机器上运行的 Linux 系统软件。

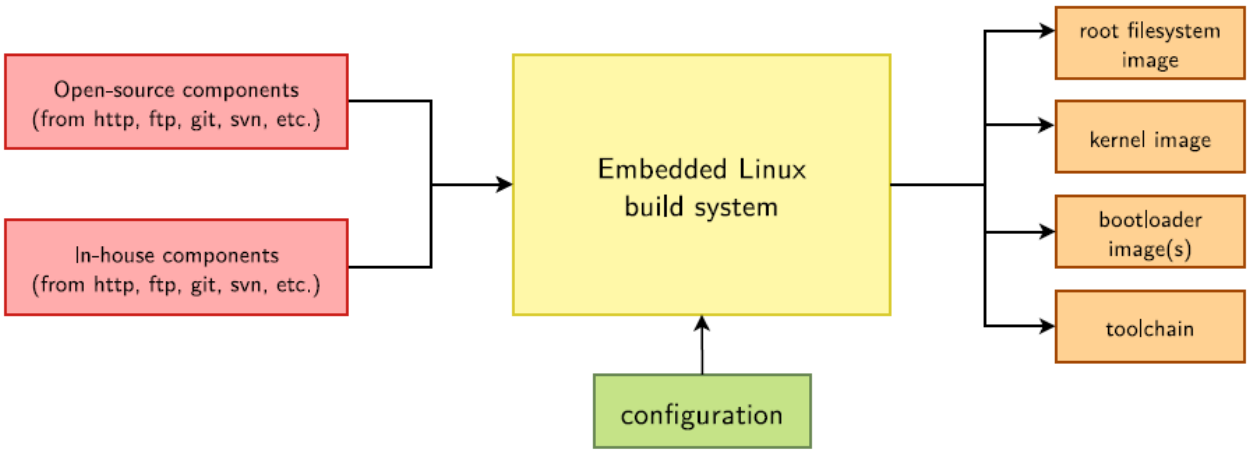


图 3-1 Buildroot 编译框图

Buildroot 有以下几点优势：

1. 通过源码构建，有很大的灵活性；
2. 方便的交叉编译环境，可以进行快速构建；
3. 各系统组件配置方便，方便定制开发。

3.2 SDK 软件框图

SDK 软件框图 3-2 所示， 从下至上分为 Bootloader、Linux Kernel、Libraries、Applications 四个层次。

各层次内容如下：

- Bootloader 层主要提供底层系统支持包，如 Bootloader、U-Boot、ATF 相关支持
- Kernel 层主要提供 Linux Kernel 的标准实现， Linux 也是一个开放的操作系统。Rockchip 平台的 Linux 核心为标准的 Linux4.4 内核，提供安全性，内存管理，进程管理，网络协议栈等基础支持；主要是通过 Linux 内核管理设备硬件资源，如 CPU 调度、缓存、内存、I/O 等。
- Libraries 层对应一般嵌入式系统，相当于中间件层次。包含了各种系统基础库，及第三方开源程序库支持，对应用层提供 API 接口，系统定制者和应用开发者可以基于 Libraries 层的 API 开发新的应用。
- Applications 层主要是实现具体的产品功能及交互逻辑，需要一些系统基础库及第三方程序库支持，开发者可以开发实现自己的应用程序，提供系统各种能力给到最终用户。

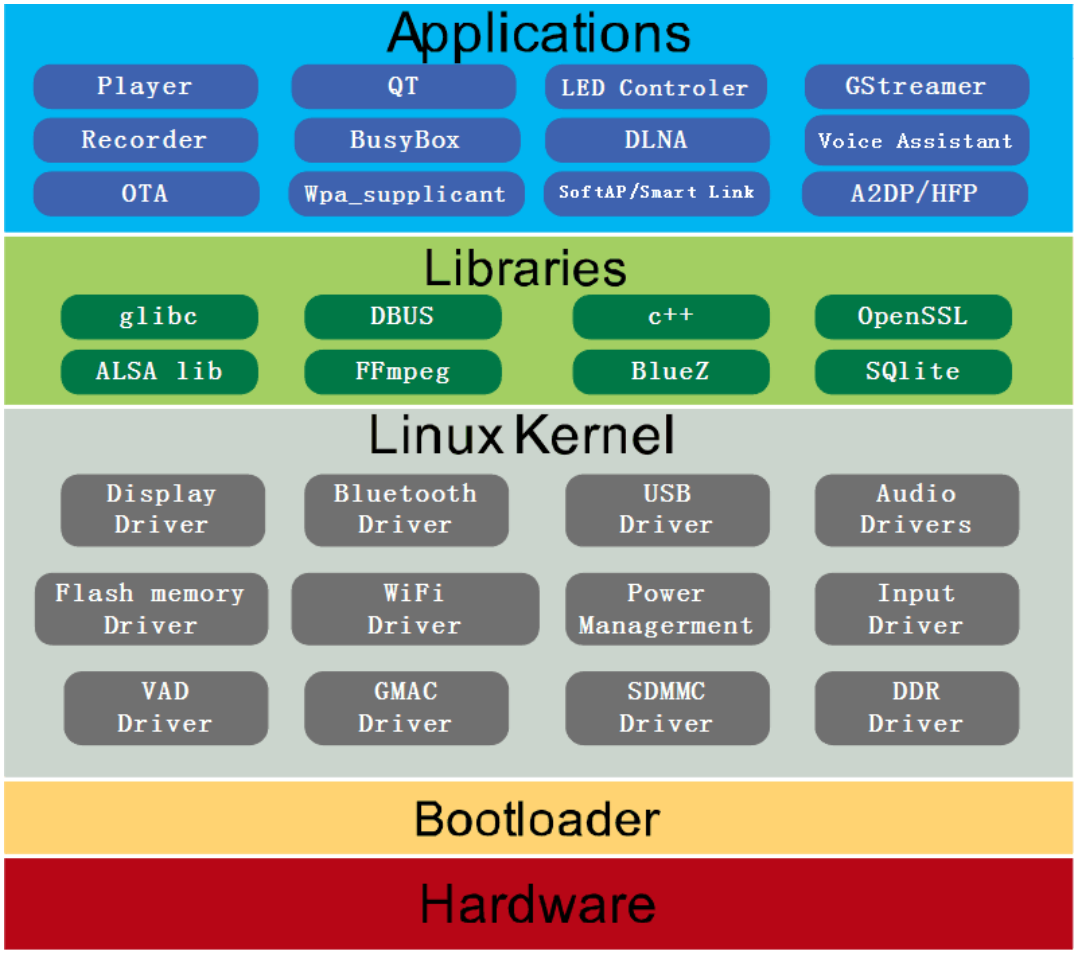


图 3-2 Buildroot 系统软件层次

SDK 系统启动流程如图 3-3 所示。

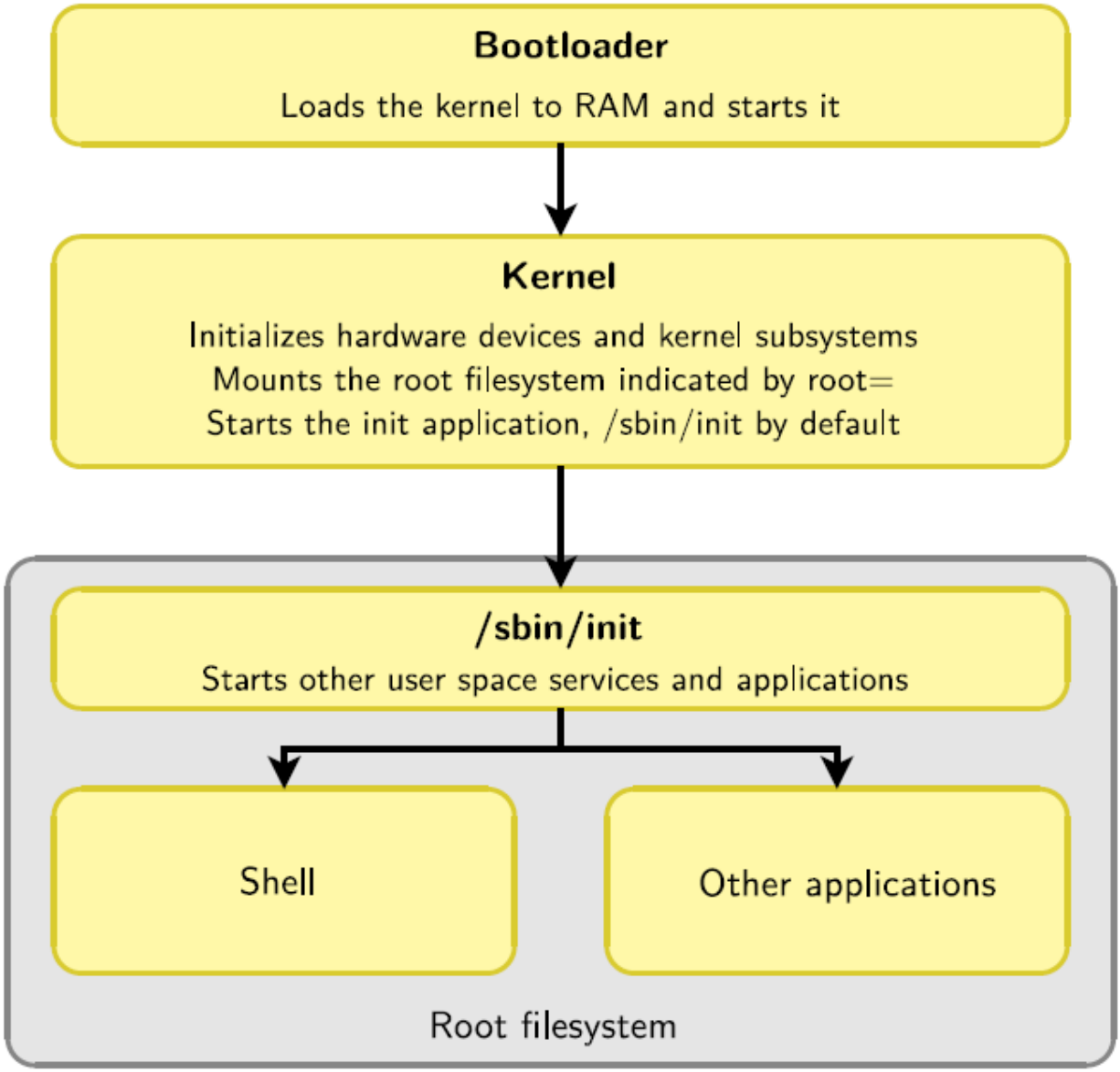


图 3-3 Linux 系统启动流程图

3.3 SDK 开发流程

Rockchip Buildroot Linux 系统是基于 Linux Kernel，针对多种不同产品形态开发的 SDK。可以基于本 SDK，有效地实现系统定制和应用移植开发。

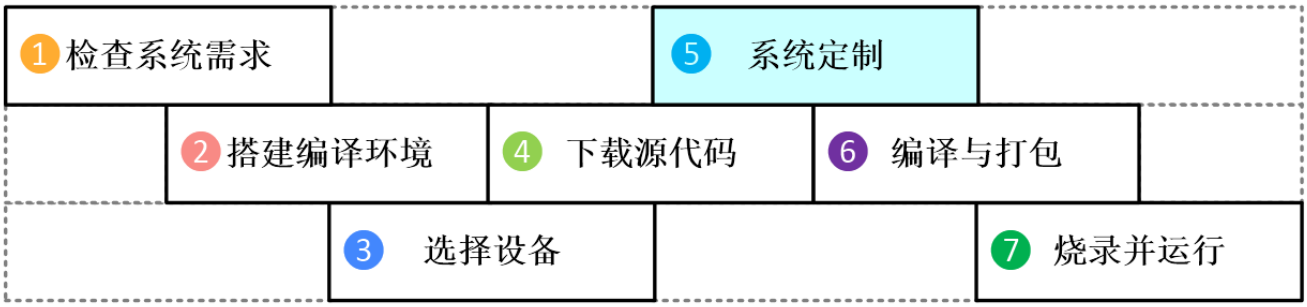


图 3-4 开发流程

如图 3-4 所示，开发者可以遵循上述开发流程，在本地快速构建 Rockchip Buildroot Linux

系统的开发环境和编译代码。下面将简单介绍下该流程：

检查系统需求：在下载代码和编译前，需确保本地的开发设备能够满足需求，包括机器的硬件能力，软件系统，工具链等。目前 SDK 支持 Linux 操作系统环境下编译，并仅提供 Linux 环境下的工具链支持，其他如 MacOS，Windows 等系统暂不支持。

搭建编译环境：介绍开发机器需要安装的各种软件包和工具，详见 4 章 开发环境搭建，获知 Rockchip Buildroot Linux 已经验证过的 Linux 操作系统版本，编译时依赖的库文件等。

选择设备：在开发过程中，需要开发者根据自己的需求，选择对应的硬件板型，详见 1.6 节 SDK 软件包使用硬件列表。

下载源代码：选定设备类型后，需要安装 repo 工具用于批量下载源代码，详见 5.1 节 SDK 获取。

系统定制：开发者可以根据使用的硬件板子、产品定义，定制 U-Boot（详见 8 章 U-Boot 开发）、Kernel（详见 9 章 Kernel 开发）及 Buildroot（详见 10 章 Buildroot 开发），请参考章节中相关开发指南和配置的描述。

编译与打包：介绍具备源代码后，选择产品及初始化相关的编译环境，而后执行编译命令，包括整体或模块编译以及编译清理等工作，进一步内容详见 6.1 节 SDK 编译。

烧录并运行：继生成镜像文件后，将介绍如何烧录镜像并运行在硬件设备，进一步内容详见 7 章 SDK 镜像烧写。

4 开发环境搭建

4.1 概述

本节主要介绍了如何在本地搭建编译环境来编译 Rockchip Buildroot Linux SDK 源代码。当前 SDK 只支持在 Linux 环境下编译，并提供 Linux 下的交叉编译工具链。

一个典型的嵌入式开发环境通常包括 Linux 服务器、Windows PC 和目标硬件版，以 RK3308 为例，典型开发环境如图 4-1 所示。

- Linux 服务器上建立交叉编译环境，为软件开发提供代码更新下载，代码交叉编译服务。
- Windows PC 和 Linux 服务器共享程序，并安装 SecureCRT 或 puTTY，通过网络远程登陆到 Linux 服务器，进行交叉编译，及代码的开发调试。
- Windows PC 通过串口和 USB 与目标硬件板连接，可将编译后的镜像文件烧写到目标硬件板，并调试系统或应用程序。

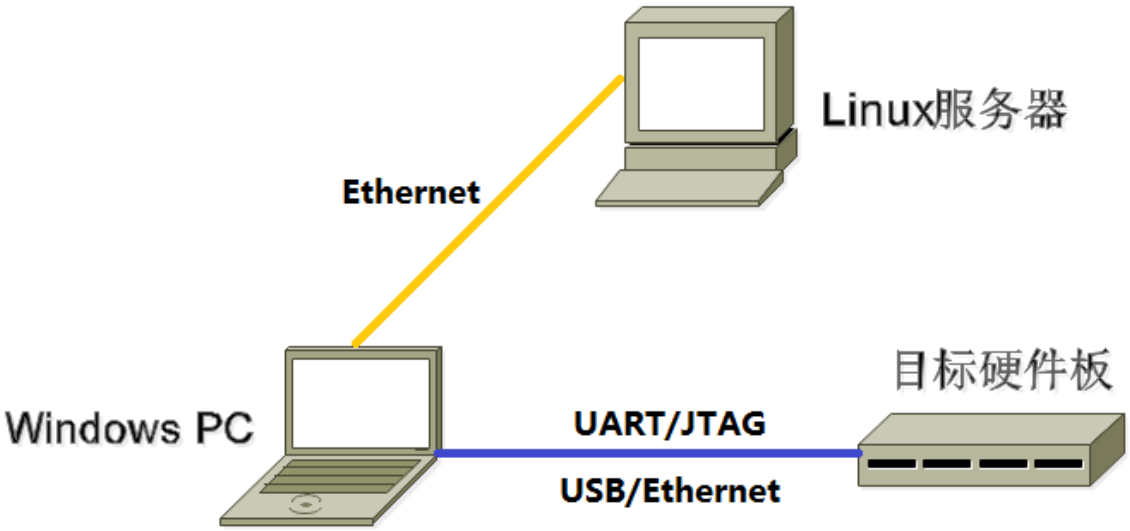


图 4-1 典型开发环境

注：开发环境中使用了 Windows PC，实际上很多工作也可以在 Linux PC 上完成，如使用 minicom 代替 SecureCRT 或 puTTY 等，用户可自行选择。

4.2 Linux 服务器开发环境搭建

Rockchip Buildroot Linux SDK 是在 Ubuntu 16.04 上开发测试的。因此，我们推荐使用 Ubuntu 16.04 的系统进行编译。其他版本没有具体测试，可能需要对软件包做相应调整。

除了系统要求外，还有其他软硬方面的要求。

- 硬件要求：64 位系统，硬盘空间大于 40G。如果您进行多个构建，将需要更大的硬盘空间。
- 软件包依赖：除了 python 2.7，make 3.8，git 1.7 之外，还需要安装一些额外的软件包，将在软件包安装章节中列出。

4.2.1 发布包使用 Linux 服务器系统版本

本 SDK 开发环境安装如下版本 Linux 系统，SDK 默认均以此 Linux 系统进行编译：

Ubuntu 16.04.2 LTS

Linux version 4.4.0-62-generic (buildd@lcy01-30) (gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.4)) #83-Ubuntu SMP Wed Jan 18 14:10:15 UTC 2017

4.2.2 网络环境搭建

请用户自行配置网络，并安装 nfs，samba，ssh 等网络组件。

4.2.3 软件包安装

操作系统安装好后，且用户已自行配置好网络环境，则可继续如下步骤完成相关软件包的安装。

1. apt-get update

```
$ sudo apt-get update
```

2. 安装 Kernel 及 U-Boot 编译需要依赖的软件包

```
sudo apt-get install git-core gnupg flex bison gperf build-essential zip curl zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386 lib32ncurses5-dev x11proto-core-dev libx11-dev lib32z1-dev ccache libgl1-mesa-dev libxml2-utils xsltproc unzip device-tree-compiler liblz4-tool
```

3. 安装 Buildroot 编译需要依赖的软件包

```
sudo apt-get install libfile-which-perl sed make binutils gcc g++ bash patch gzip bzip2 perl tar cpio python unzip rsync file bc libmpc3 git repo texinfo pkg-config cmake tree texinfo
```

若编译遇到报错，可以视报错信息，安装对应的软件包。

4.2.4 交叉编译工具链介绍

鉴于 Rockchip Buildroot SDK 目前只在 Linux 下编译，我们也仅提供了 Linux 下的交叉编译工具链。其中 U-Boot 及 Kernel 使用的编译工具链预置目录在 prebuilt/gcc 下，buildroot 使用该开源软件中编译出来的工具链。

U-Boot 及 Kernel 编译工具链

```
prebuilts/gcc/linux-x86/aarch64/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-
```

对应版本

```
gcc version 6.3.1 20170404 (Linaro GCC 6.3-2017.05)
```

Buildroot 编译工具链

```
buildroot/output/rockchip_rk3308_release/host/bin/aarch64-rockchip-linux-gnu-
```

对应版本

```
gcc version 6.4.0 (Buildroot 2018.02-rc3-00017-g9c68ede
```

如果需要其他平台或版本的工具链，需自行编译。

上述环境准备好后，Linux 服务器开发环境搭建已完成，可以下载编译源代码了。

4.2.5 Linux 烧写工具使用

Linux 系统上的的烧录工具发布在 tools\linux\Linux_Upgrade_Tool\Linux_Upgrade_Tool 目录下，可用于 Linux 环境下开发调试，固件的烧写。

具体的使用说明见 12.4 节 Upgrade_tool。

4.3 Windows PC 开发环境搭建

4.3.1 开发工具安装

请用户自行安装 SourceInsight，Notepad++等编辑软件。

4.3.2 Rockchip USB 驱动安装

开发调试阶段，需要将设备切换至 Loader 模式或是 Maskrom 模式，需要安装 Rockusb 驱动才能正常识别设备。

Rockchip USB 驱动安装助手存放在 tools\windows\DriverAssitant_v4.x.zip。支持 xp,win7_32,win7_64,win8_32,win8_64 操作系统。

安装步骤如下：

图 4-3 Rockchip USB 驱动安装 1

图 4-4 Rockchip USB 驱动安装 2

图 4-5 Rockchip USB 驱动安装 3

4.3.3 Windows 烧录工具使用

Windows 系统上的的烧录工具发布在 tools\windows\AndroidTool/AndroidTool_Release，可用于 Windows 环境下开发调试，固件的烧写。

具体的使用说明见 12.3 节 瑞芯微开发工具。

4.4 目标硬件板准备

请参考 1.6 节 SDK 软件包适用硬件列表，选择对应硬件板子，进行后续的开发调试。对应的硬件使用说明文档，会介绍硬件接口，使用说明，及烧录操作方法。

5 SDK 安装准备工作

5.1 简要介绍

Rockchip Buildroot Linux SDK 的代码和相关文档被划分为了若干 git 仓库分别进行版本管理，开发者可以使用 repo 对这些 git 仓库进行统一的下载，提交，切换分支等操作。

5.2 安装 repo

确保主目录下有一个 bin/ 目录，并且该目录包含在路径中：

```
mkdir ~/bin
export PATH=~/bin:$PATH
```

如果可以访问 google 的地址，下载 Repo 工具，并确保它可执行：

```
curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
chmod a+x ~/bin/repo
```

中国国内环境如果执行上述命令后发现~/bin/repo 为空，此时可以访问国内的站点来下载 repo 工具

```
curl https://mirrors.tuna.tsinghua.edu.cn/git/git-repo -o ~/bin/repo
chmod a+x ~/bin/repo
```

除以上两种方式外，也可以使用如下命令获取 repo

```
sudo apt-get install repo
```

5.3 Git 配置

在使用 repo 之前请配置一下自己的 git 信息，否则后面的操作可能会遇到 hook 检查的障碍

```
git config --global user.name "your name"
git config --global user.email "your mail"
```

5.4 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，请参考附录 13.1 SSH 公钥操作说明。

5.4.1 SDK 下载命令

Rockchip Buildroot Linux SDK 被设计为可以适配到不同的芯片平台上，如 RK3308，RK3288，RK3326，RK3399，RK3399PRO, RK1808, PX30，PX3-SE 等，对于不同芯片平台的源代码会有一定程度的不同。开发者下载源代码的时候声明自己想要的芯片平台，从而不必下载自己不需要的代码。

SDK 使用 -m <芯片平台_linux_release>.xml 来声明自己想要下载的对应芯片平台。

RK3308_LINUX_SDK 下载命令如下：

```
mkdir rk3308
cd rk3308
repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u
ssh://git@www.rockchip.com.cn/linux/rk/platform/manifests -b linux
-m rk3308_linux_release.xml
.repo/repo/repo sync -c
```

代码将开始自动下载，后面只需耐心等待。源代码文件将位于工作目录中对应的项目名称下。初始同步操作将需要 1 个小时或更长时间才能完成。

5.4.2 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 rk3308_linux_v1.00_20180510.tgz 为例，拷贝到该初始化包后，通过如下命令可检出源码：

```
mkdir rk3308
tar xvf rk3308_linux_v1.00_20180510.tgz -C rk3308
cd rk3308
.repo/repo/repo sync -l
.repo/repo/repo sync -c
```

表 5-1 Rockchip SDK 对应压缩包

芯片平台	压缩包	版本
PX30	px30_linux_beta_v1.0_20180710.tgz	V1.00
RK3308	rk3308_linux_v1.00_20180510.tgz	V1.00
RK3326	rk3326_linux_sdk_20180426.tgz	V1.00
RK3399	rk3399_linux_sdk_v2.0_20180517.tgz	V2.00
RK1808	Rk1808_linux_v1.00_20181225.tgz	V1.00
RK3399PRO	rk3399pro_linux_sdk_beta_v0.01_20190217.tgz	V0.01

5.4.3 ReleaseNote

为方便客户及开发者了解到 SDK 更新所覆盖的修改，解决的问题。SDK 上加入了 ReleaseNote.txt，该文件会记录每次更新解决的问题，及是否建议客户全部更新。请客户及开发者更新 ReleaseNote 后，查看更新内容，并决定是否更新 SDK 代码。

ReleaseNote 存放目录详见表 5-2：

表 5-2 Rockchip SDK 对应 ReleaseNote

芯片平台	**ReleaseNote**
PX30	.repo/manifests/px30_linux_v1.00/PX30_Release_Note.txt
RK3308	.repo/manifests/rk3308_linux_v1.00/RK3308_Release_Note.txt
RK3326	.repo/manifests/rk3326_linux_v1.00/RK3326_Release_Note.txt
RK3399	.repo/manifests/rk3399_linux_v2.00/RK3399_Release_Note.txt

RK1808	.repo/manifests/rk1808_linux_v1.00/RK1808_Release_Note.txt
RK3399PRO	.repo/manifests/rk3399pro_linux_v0.01/RK3399PRO_Release_Note.txt

单独更新 ReleaseNote 方法：

1.首先记录.repo/manifests/目录当前提交号 commit1

2.执行以下命令更新 ReleaseNote 文件

```
cd .repo/manifests/
git pull origin linux
```

查看 ReleaseNote 文件说明，若决定更新，则在根目录下执行以下命令：

```
.repo/repo/repo sync
```

若不想更新代码，则在.repo/manifests 目录执行以下命令，将提交回退到上一版本状态：

```
git reset –hard commit1
```

5.5 SDK 目录结构

SDK 下载完成后，在根目录下可以看到如下目录结构：

```
├── buildroot
├── build.sh
├── device
├── docs
├── external
├── IMAGE
├── kernel
├── mkfirmware.sh
├── prebuilts
├── rkbin
├── rockdev
├── tools
└── u-boot
```

- Buildroot 目录存放 buildroot 开源项目代码，可定制根文件系统
- build.sh 为系统编译脚本，执行可进行 SDK 的完整编译
- device 目录存放板级配置及一些预置文件，开机脚本等
- docs 目录存放 SDK 提供的相关文档
- external 目录存放 SDK 相关库及工具源码
- IMAGE 为 build.sh 执行后存放所有镜像文件备份，及版本编译信息
- kernel 为内核部分源码
- mkfirmware.sh 脚本可对镜像文件进行打包，并统一拷贝至 rockdev/目录
- prebuilts 目录存放 U-Boot、Kernel 编译使用的交叉编译工具链
- rkbin 目录存放 Rockchip 平台一些关键性二进制文件，包括 ddr.bin,miniloader.bin,bl31.bin，在 U-Boot 编译过程中会用到
- rockdev 执行 mkfirmware.sh 会把系统编译的生成的镜像，统一拷贝至 rockdev
- tools 目录存放着 Windows 及 Linux 环境下的开发工具、调试工具、量产工具
- u-boot 目录存放着 U-Boot 部分的源码

5.6 SDK 更新

后续开发者可根据 Fae 窗口定期发布的更新说明，通过命令同步更新。

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

5.7 SDK 问题反馈

Rockchip bug 系统（Redmine）为了更好的帮助客户开发，记录了用户问题处理过程及状态，方便双方同时跟踪，使问题处理更及时更高效。后续 SDK 问题、具体技术问题、技术咨询等都可以提交到此 Bug 系统上，Rockchip 技术服务会及时将问题进行分发、处理和跟踪。

详见 docs\RKTools manuals\目录下《Rockchip Bug 系统使用指南 V1.0-201712.pdf》。

6 SDK 编译

6.1 Uboot 编译

- **PX30 平台：**

```
cd u-boot
./make.sh evb-px30
```

编译完，会生成 trust.img、px30_loader_v1.07.107.bin、uboot.img 三个镜像文件。

- **RK3308 平台：**

```
cd u-boot
./make.sh evb-rk3308
```

编译完，会生成 trust.img、rk3308_loader_v1.17.101.bin、uboot.img 三个镜像文件。

RK3308 32bit 编译

```
cd u-boot
./make.sh evb-aarch32-rk3308
```

- **RK3326 平台：**

```
cd u-boot
./make.sh evb-rk3326
```

编译完，会生成 trust.img、rk3326_loader_v1.07.110.bin、uboot.img 三个镜像文件。

- **RK3399 平台：**

```
cd u-boot
./make.sh evb-rk3399
```

编译完，会生成 trust.img、rk3399_loader_v1.12.112.bin、uboot.img 三个镜像文件。

- **RK1808 平台：**

```
cd u-boot
./make.sh rk1808
```

编译完，会生成 trust.img、rk1808_loader_v1.01.101.bin、uboot.img 三个镜像文件。

- **RK3399PRO 平台：**

```
cd u-boot
./make.sh rk3399pro
```

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

6.2 Kernel 编译

- **PX30 平台：**

PX30 EVB V10 开发板

硬件板本	板上丝印	参考设计
V10	PX30_Mini_EVB_V10_20180526	rk_evb_px30_ddr3p416dd6_v10.pdf

PX30	对应板级配置文件	编译命令
EVB	px30-evb-ddr3-v10-linux-v10.	cd kernel

	dts	make px30_linux_defconfig make px30-evb-ddr3-v10-linux.img

- **RK3308 平台：**

1. RK3308 EVB V10 开发板

硬件板本	板上丝印	参考设计
V10	RK_EVB_RK3308_DDR3P116SD4_V10_20180301	RK3308_AI-VA_BETA_V01_20180307

RK3308 EVB V10 开发板搭配不同的麦克风阵列小板，需要选用不同的板级配置文件，区分如下：

麦克风陈列小板	对应板级配置文件	编译命令
I2S 数字麦克风	rk3308-evb-dmic-i2s-v10.dts	cd kernel make rk3308_linux_defconfig make rk3308-evb-dmic-i2s-v10.img
模拟麦克风	rk3308-evb-amic-v10.dts	cd kernel make rk3308_linux_defconfig make rk3308-evb-amic-v10.img
PDM 数字麦克风	rk3308-evb-dmic-pdm-v10.dts	cd kernel make rk3308_linux_defconfig make rk3308-evb-dmic-pdm-v10.img

2. RK3308 EVB V11、V12 开发板：

硬件板本	板上丝印	参考设计
V11	RK_EVB_RK3308_DDR3P116SD4_V11_20180420 RK_EVB_RK3308_DDR3P116SD4_V12	RK3308_AI-VA_REF_V10

RK3308 EVB V11、V12 开发板搭配不同的麦克风阵列小板，需要选用不同的板级配置文件，区分如下：

麦克风陈列小板	对应板级配置文件	编译命令
I2S 数字麦克风	rk3308-evb-dmic-i2s-v11.dts	cd kernel make rk3308_linux_defconfig make rk3308-evb-dmic-i2s-v11.img
模拟麦克风	rk3308-evb-amic-v11.dts	cd kernel make rk3308_linux_defconfig make rk3308-evb-amic-v11.img
PDM 数字麦克风	rk3308-evb-dmic-pdm-v11.dts	cd kernel make rk3308_linux_defconfig make rk3308-evb-dmic-pdm-v11.img

编译完成后，kernel 根目录，生成 boot.img 镜像文件。

3. RK3308 语音模块板：

详见 docs\ SoC platform related\RK3308\目录下《RK3308 语音模块 32 位系统编译指南_v1.00.pdf》。

- **RK3326 平台：**

RK3326 EVB V11 开发板

硬件板本	板上丝印	参考设计
V11	RK_EVB_RK3326_LP3178P132SD4_V11_20180301_FZB	RK_EVB_RK3326_LP3S178P132SD4_V11_20180301

RK3326	对应板级配置文件	编译命令
EVB	rk3326-evb-linux-lp3-v10.dts	cd kernel make rk3326_linux_defconfig make rk3326-evb-linux-lp3-v10.img

- **RK3399 平台：**

RK3399 EVB V13 开发板

硬件板本	板上丝印	参考设计
V13	RK_EXCAVATOR_MAIN_V13_20170911_FZB	RK_SAPPHIRE_SOCBOARD_RK3399_LPDDR3D178P232SD8_V12_20161109 和 RK_EXCAVATOR_MAIN_V13_2017091

RK3399	对应板级配置文件	编译命令
EVB	rk3399-sapphire-excavator-linux.dts	cd kernel make rockchip_linux_defconfig make rk3399-sapphire-excavator-linux.img

- **RK1808 平台：**

RK1808 EVB V11 开发板

硬件板本	板上丝印	参考设计
V11	RK_EVB_RK1808_LP3D1789132SD6_V11_20181107_FINAL_LINT	RK_EVB_RK1808_LP3D1789132SD6_V11_20181107

RK1808	对应板级配置文件	编译命令
EVB	rk1808-evb-v10.dts	cd kernel make rk1808_linux_defconfig make rk1808-evb-v10.img

- **RK3399PRO 平台：**

RK3399PRO EVB V10 V11 开发板

硬件板本	板上丝印	参考设计
V10	RK_EVB_RK3399PRO_LP35178P332SD8_V10	RK_EVB_RK3399PRO_LP3S178P332SD8_V10 20180921_RZF 和 RK_EVB_RK3399PRO_LP3S178P332SD8_V10 20180925_final_lint.brd
V11	RK_EVB_RK3399PRO_LP35178P332SD8_V11	RK_EVB_RK3399PRO_LP3S178P332SD8_V11 20181113_RZF 和 rk_evb_rk3399pro_lp3s178p332sd8_v11-20181115.brd

RK3399PRO	对应板级配置文件	编译命令
EVB V10	rk3399pro-evb-v10-linux.dts	cd kernel make rockchip_linux_defconfig make rk3399pro-evb-v10-linux.img
EVB V11	rk3399pro-evb-v11-linux.dts	cd kernel make rockchip_linux_defconfig make rk3399pro-evb-v11-linux.img

6.3 Buildroot 编译

客户按实际编译环境配置好编译依赖后，按照以下步骤配置完后，执行 make 即可。

```
$ source buildroot/build/envsetup.sh

You're building on Linux
Lunch menu...pick a combo:
1. rockchip_rk3308_release
2. rockchip_rk3308_debug
3. rockchip_rk3308_robot_release
4. rockchip_rk3308_robot_debug
5. rockchip_rk3308_mini_release

Which would you like? [1]
```

如选择 rockchip_rk3308_release，输入对应序号 1。

```
$ make
```

完成编译后执行 SDK 根目录下的 mkfirmware.sh 脚本生成固件

```
$ ./mkfirmware.sh
```

所有烧写所需的镜像将都会拷贝于 rockdev 目录。

```
rockdev
├── boot.img
├── misc.img
├── parameter.txt
├── recovery.img
├── MiniLoaderAll.bin（即 rk3308_loader_v1.17.101.bin）
├── oem.img
├── userdata.img
```

```
├── rootfs.img
├── trust.img
└── uboot.img
```

得到了所有镜像文件后，为了方便烧写及量产，通常可手动将这些单独的镜像通过脚本打包成为一个 update.img，打包方法详见 12.3 节统一固件打包工具。若使用全自动编译脚本会自动打包 update.img 出来。

6.4 全自动编译脚本

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

1）该全自动化编译脚本原始文件存放于：

```
device/rockchip/common/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：

3）修改 device/rockchip/rkxx(芯片平台)/BoardConfig.mk 脚本中的特定变量以编出对应产品固件。

如 RK3308 平台，可修改 device/rockchip/rk3308/BoardConfig.mk 文件：

```
#buildroot defconfig
LUNCH=rockchip_rk3308_release
#uboot defconfig
UBOOT_DEFCONFIG=evb-rk3308
#kernel defconfig
KERNEL_DEFCONFIG=rk3308_linux_defconfig
#kernel dts
KERNEL_DTS= rk3308-evb-dmic-pdm-v11
```

以下变量请按实际项目情况，对应修改：

```
LUNCH 变量指定 Buildroot 编译 defconfig。
KERNEL_DTS 变量指定编译 kernel 的产品板极配置。
```

4）执行自动编译脚本：

```
./build.sh
```

该脚本会自动配置环境变量，编译 U-Boot，编译 Kernel，编译 Buildroot，编译 Recovery 继而生成固件。

5）脚本生成内容:

脚本会将编译生成的固件拷贝至：

IMAGE/RK3308-EVB-DMIC-PDM-V11_****_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。

6.5 模块部分编译

为了方便开发调试，全自动化编译脚本也支持单独模块进行编译，方便模块调试，可指定并编译部分模块。

模块部分编译可参见使用说明：

```
./build.sh -h
====USAGE: build.sh modules====
uboot                   -build uboot
```

kernel	-build kernel
rootfs	-build default rootfs, currently build buildroot as default
buildroot	-build buildroot rootfs
yocto	-build yocto rootfs, currently build ros as default
ros	-build ros rootfs
debian	-build debian rootfs
pcba	-build pcba
recovery	-build recovery
all	-build uboot, kernel, rootfs, recovery image
cleanall	-clean uboot, kernel, rootfs, recovery
firmware	-pack all the image we need to boot up system
updateimg	-pack update image
save	-save images, patches, commands used to debug
default	-build all modules

如单独编译 kernel，只需要执行以下命令：

```
./build.sh kernel
```

7 SDK 镜像烧写

7.1 概述

本章节主要介绍如何将构建完成的镜像文件（image）烧写并运行在硬件设备上的流程。

Rockchip 平台提供的几种镜像烧写工具介绍如表 7-1 所示，可以选择合适的烧写方式进行烧写。烧写前，需安装最新的的 USB 驱动，详见 4.3.3 节 Rockchip USB 驱动安装。

表 7-1 Rockchip 平台烧写工具

工具	运行系统	描述
瑞芯微开发工具	Windows	分立升级固件及整个 update 升级固件工具
FactoryTool 工具	Windows	量产升级工具，支持 USB 一拖多烧录
upgrade_tool 工具	Linux	Linux 下开发工具

7.2 烧写模式介绍

Rockchip 平台硬件运行的几种模式如表 7-2 所示，只有当设备处于 Maskrom，及 Loader 模式下，才能够烧写固件，或对板上固件进行更新操作。

表 7-2 Rockchip 平台设备模式介绍

模式	工具烧录	介绍
Maskrom	支持	Flash 在未烧录固件时，芯片会引导进入 Maskrom 模式，可以进行初次固件的烧写；开发调试过程中若遇到 Loader 无法正常启动的情况，也可进入 Maskrom 模式烧写固件。
Loader	支持	Loader 模式下，可以进行固件的烧写、升级。可以通过工具单独烧写某一个分区镜像文件，方便调试。
Recovery	不支持	系统引导 recovery 启动，主要作用是升级、恢复出厂设置类操作。
Normal Boot	不支持	系统引导 rootfs 启动，加载 rootfs，大多数的开发都是在这个模式在调试的。

进入烧写模式方式以下几种方法：

1. 未烧录过固件，上电，进入 Maskrom 模式。
2. 烧录过固件，按住 recovery 按键上电或复位，系统将进入 Loader 固件烧写模式。
3. 烧录过固件，按住 Maskrom 按键上电或复位，系统将进入 MaskRom 固件烧写模式。
4. 烧录过固件，上电或复位后开发板正常进入系统后，瑞芯微开发工具上显示"发现一个 ADB 设备"或"发现一个 MSC 设备"，然后点击工具上的按钮"切换"，进入 Loader 模式。
5. 烧录过固件，可在串口或 adb 命令行模式下，输入 reboot loader 命令，进入 Loader 模式。

以 RK3308 EVB 板为例，Maskrom 键如图 7-1 所示，Recovery 键如图 7-2 所示。

图 7-1 RK3308 EVB 板 Maskrom 键

图 7-2 RK3308 EVB 板 Recovery 键

7.3 瑞芯微开发工具烧写

烧录说明详见 docs\ RKTools manuals 目录下《瑞芯微开发工具手册.pdf》。

SDK 提供烧写工具，如图 7-3 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。

7.3.1 下载镜像

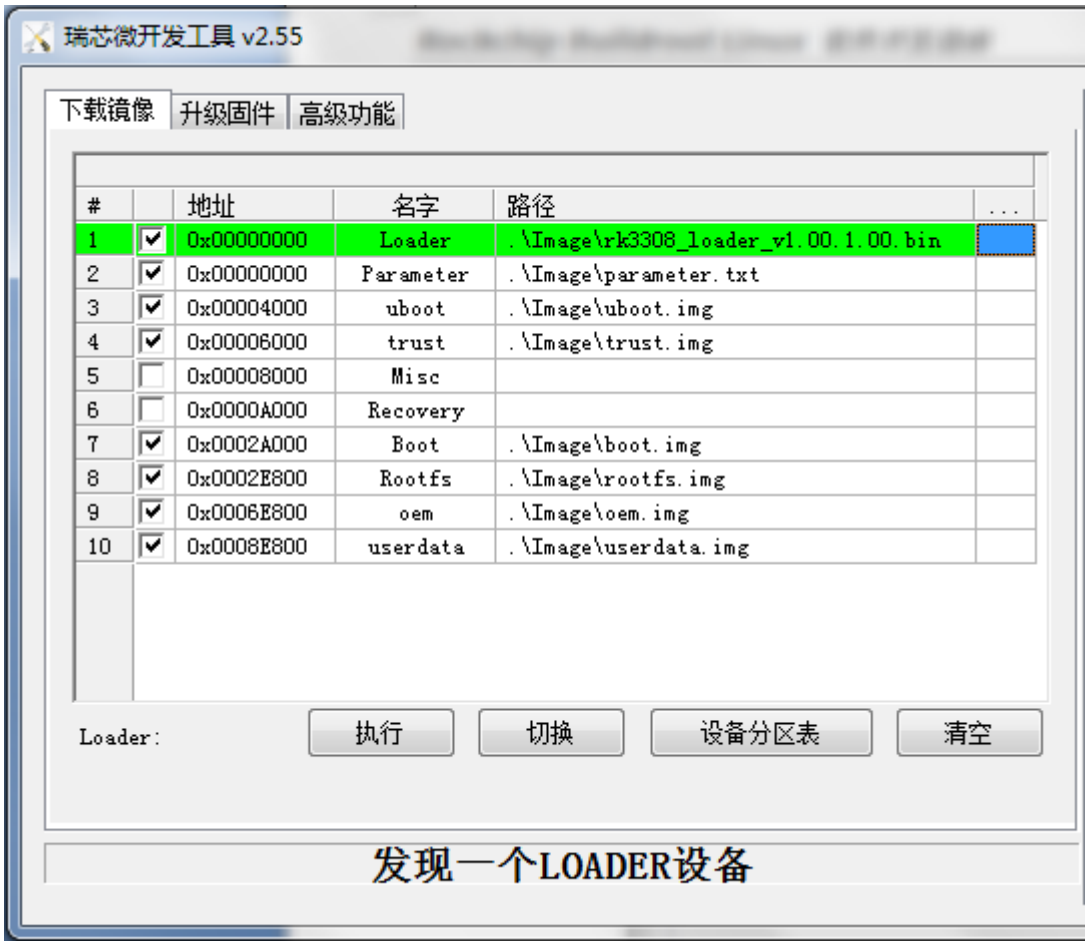


图 7-3 瑞芯微开发工具烧写界面

1）USB 连接硬件板进入烧写模式。

2）打开工具点击下载镜像菜单，点击...列会跳出一个文件选择框，可以选择对应分区的 img 本地地址，其他几项依次配置。

3）配置完成后，点击执行就可以看到右边空白框进入下载提示。

其中 “低格”按钮是用来擦除设备的，“清空”按钮是清空编辑框文本。

7.3.2 升级固件

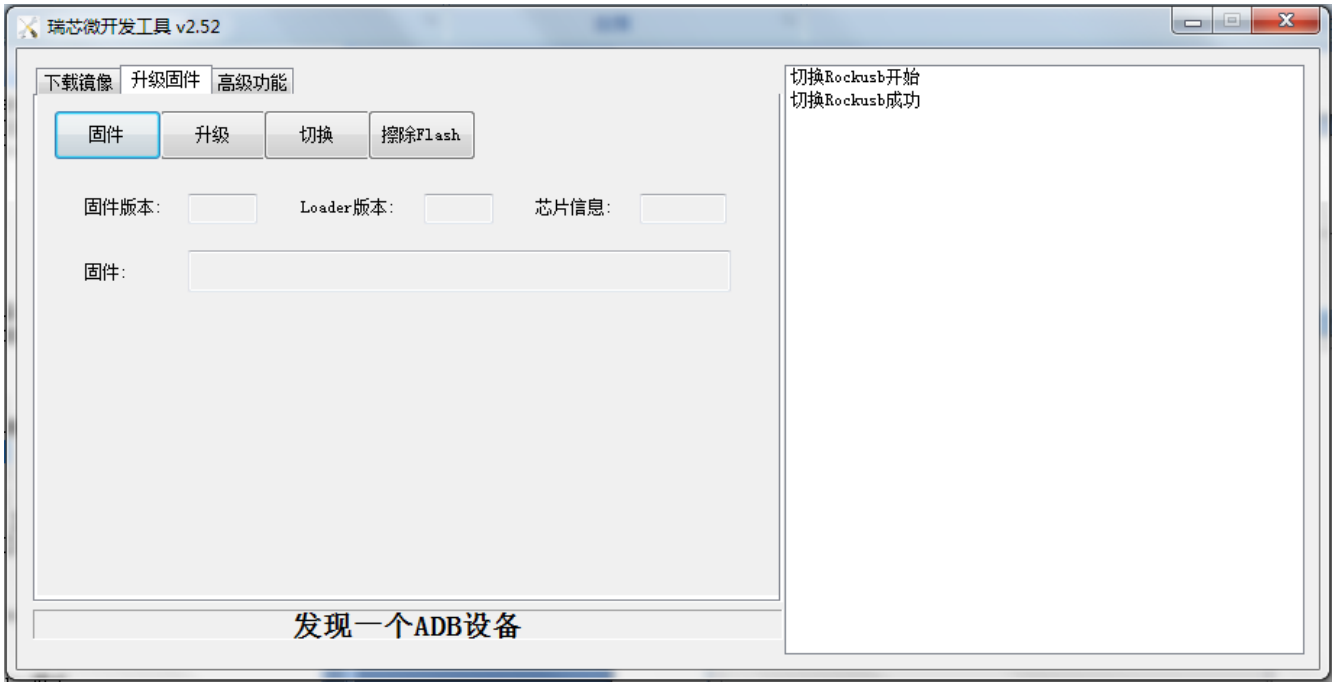


图 7-4 Android 开发工具升级固件

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

7.3.3 高级功能

图 7-5 Android 开发工具高级功能

Boot 只能选择打包好的 update.img 文件或是 loader 的文件；

固件必须现使用打包后的 update.img;

解包功能可将 update.img 拆解为各部分镜像文件。

7.4 Upgrade_tool 工具烧写

烧录说明详见 docs\ RKTools manuals 目录下《Linux 开发工具使用手册_v1.32.pdf》。

使用 USB 线连接至 Linux 主机，请确认你的板子进入烧写模式，升级命令如下：

```
sudo ./upgrade_tool ul rk3308_loader_v1.17.101.bin
sudo ./upgrade_tool di -p parameter.txt
sudo ./upgrade_tool di -u uboot.img
sudo ./upgrade_tool di -t trust.img
sudo ./upgrade_tool di -misc misc.img
sudo ./upgrade_tool di -r recovery.img
sudo ./upgrade_tool di -b boot.img
sudo ./upgrade_tool di -rootfs rootfs.img
sudo ./upgrade_tool di -oem oem.img
sudo ./upgrade_tool di -userdata userdata.img
sudo ./upgrade_tool rd
```

7.5 FactoryTool 工具烧写

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。

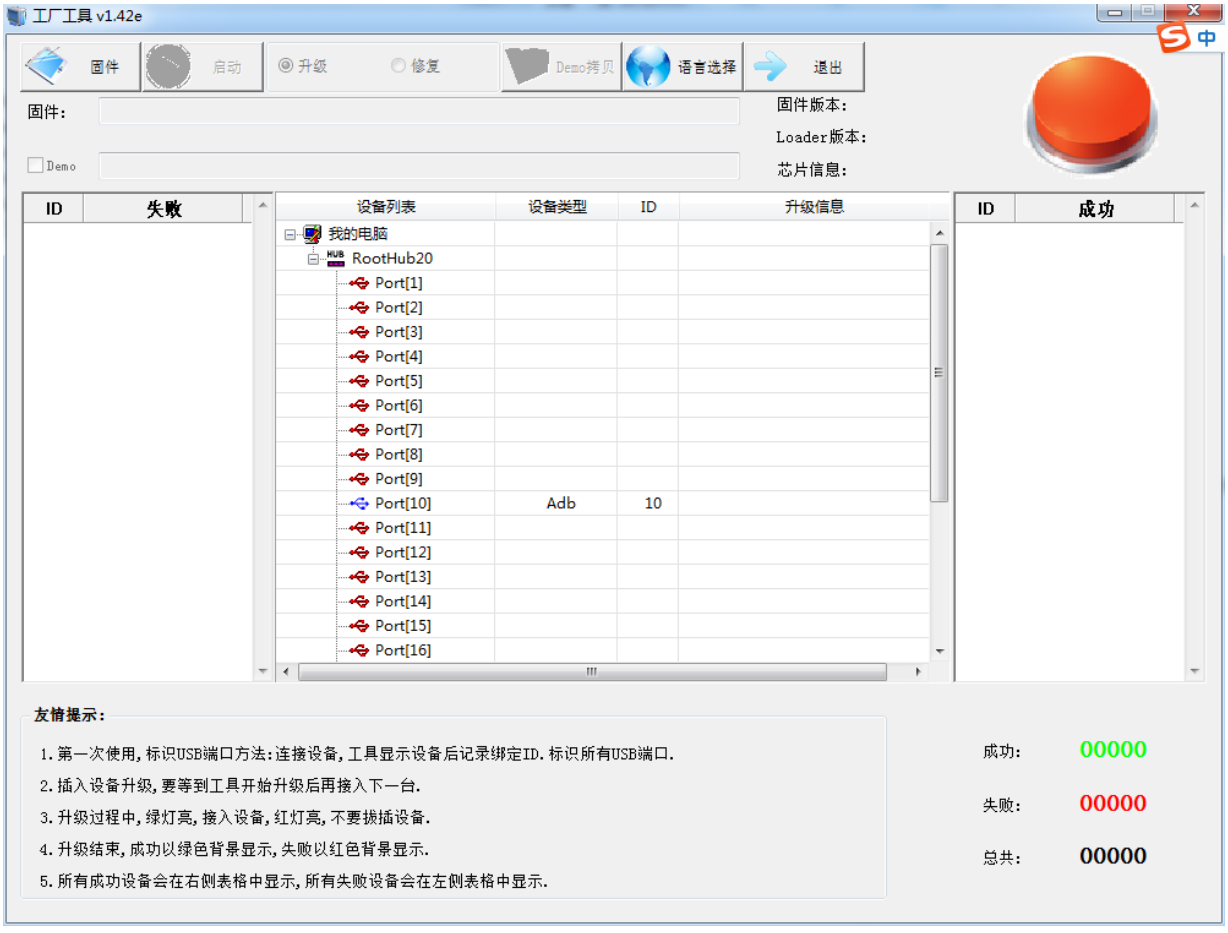


图 7-6 量产工具

7.6 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 docs\RKTools manuals 目录下《Rockchip 量产烧录 指南 V1.1-20170214.pdf》。

目前 Rockchip Linux SDK 支持 USB 升级方案，及烧录器升级方案。

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<fae@rock-chips.com>。

8 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 docs\Develop reference documents 目录下《Rockchip-Developer-Guide-UBoot-nextdev.pdf》。

8.1 Rockchip U-Boot 简介

Rockchip U-Boot next-dev 分支是 Rockchip 从 U-Boot 官方的 v2017.09 正式版本中切出来进行开发的版本。目前在该平台上已经支持 RK 所有主流在售芯片。

- 支持芯片：rk3288、rk3036、rk312x、rk3288、rk3308、rk3326、rk3399、px30 等；
- 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 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；

8.2 平台编译

8.2.1 rkbin

rkbin 工程主要存放了 Rockchip 不开源的 bin 文件（trust、loader 等）、脚本、打包工具等，所以 rkbin 只是一个"工具包"工程 。

rkbin 工程需要和 U-Boot 工程保持同级目录关系，否则编译时会报找不到 rkbin 仓库。当在 U-Boot 工程执行编译的时候，编译脚本会从 rkbin 仓库里索引相关的 bin 文件和打包工具，最后在 U-Boot 根目录下生成 trust.img、uboot.img、loader 等相关固件。

8.2.2 gcc 工具链

默认使用的编译器是 gcc-linaro-6.3.1 版本：

```
32 位编译器：gcc - linaro - 6.3.1 - 2017.05 - x86_64_arm - linux - gnueabihf
64 位编译器：gcc - linaro - 6.3.1 - 2017.05 - x86_64_aarch64 - linux - gnu
```

默认使用 Rockchip 提供的工具包：prebuilts，请保证它和 U-Boot 工程保持同级目录关系。

如果需要更改编译器路径，可以修改编译脚本./make.sh 里的内容：

```
GCC_ARM32=arm - linux - gnueabihf -
GCC_ARM64=aarch64 - linux - gnu -
TOOLCHAIN_ARM32=../prebuilts/gcc/linux - x86/arm/gcc - linaro - 6.3.1 -
2017.05 - x86_64_arm - linuxgnueabihf/
bin
TOOLCHAIN_ARM64=../prebuilts/gcc/linux - x86/aarch64/gcc - linaro - 6.3.1 -
2017.05 - x86_64_aarch64 -
linux - gnu/bin
```

8.2.3 编译

编译命令：

```
./make.sh [board] - - - -  [board]的名字来源是：configs/[board]_defconfig 文件。
```

命令范例：

```
./make.sh evb- rk3308 - - - -  build for evb- rk3308_defconfig
./make.sh firefly- rk3288 - - - -  build for firefly- rk3288_defconfig
```

9 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。Kernel 版本以 4.4 作为基准，做相应的介绍。

9.1 DTS 介绍

9.1.1 概述

早期版本的 Linux Kernel 是直接在板级配置文件配置板子相关的信息，如 IOMUX，默认拉高/低的 GPIO，每个 I2C/SPI 总线下的 client 设备信息。为了摒弃这种‘hard code'的方式，Linux 引入设备树（Device Tree）的概念来描述不同的硬件结构。

Device Tree 数据可读性较高，遵循 DTS 规范，通常被描述在.dtsi 和.dts 源文件。在内核编译的过程中，被编译为.dtb 的二进制文件。在开机启动阶段，dtb 会被 bootloader（如 U-Boot）加载到 RAM 的某个地址空间，并且将该地址作为参数传递给 Kernel space。内核解析整个 dtb 文件，提炼每个设备信息以初始化。

本文旨在介绍如何新增一个的板子 dts 配置以及一些常见的 dts 语法说明，关于更详细 dts 的语法介绍不在本文范围内，如有兴趣，请参考：

1. https://www.devicetree.org/specifications/
2. Documentation/devicetree/bindings

9.1.2 新增一个产品 DTS

9.1.2.1 创建 dts 文件

Linux Kernel 目前支持多平台使用 dts，RK 平台的 dts 文件存放于：

```
ARM:arch/arm/boot/dts/
ARM64 :arch/arm64/boot/dts/rockchip
```

一般 dts 文件的命名规则为”soc-board_name.dts”，如 rk3308-evb-dmic-i2s-v10.dts。soc 指的是芯片名称，board_name 一般是根据板子丝印来命名。

如果你的板子是一体板，则只需要一个 dts 文件来描述即可。

```
rk3308-ai-va-v10.dts
  └── rk3308.dtsi
```

如果硬件设计上是核心板和底板的结构，或者产品有多个产品形态，可以把公用的硬件描述放在 dtsi 文件，而 dts 文件则描述不同的硬件模块，并且通过 include "xxx.dtsi"将公用的硬件描述包含进来。

```
├── rk3308-evb-amic-v10.dts
│   ├── rk3308-evb-ext-v10.dtsi
│   └── rk3308-evb-v10.dtsi
│       └── rk3308.dtsi
└── rk3308-evb-dmic-i2s-v10.dts
    ├── rk3308-evb-ext-v10.dtsi
    └── rk3308-evb-v10.dtsi
```

```
└── rk3308.dtsi
```

9.1.2.2 修改 dts 所在目录的 Makefile

```
diff --git a/arch/arm64/boot/dts/rockchip/Makefile
b/arch/arm64/boot/dts/rockchip/Makefile
index 073281d..7c329d4 100644
--- a/arch/arm64/boot/dts/rockchip/Makefile
+++ b/arch/arm64/boot/dts/rockchip/Makefile
@@ -1,6 +1,9 @@
 dtb-$(CONFIG_ARCH_ROCKCHIP) += px30-evb-ddr3-v10.dtb
 dtb-$(CONFIG_ARCH_ROCKCHIP) += px30-evb-ddr3-lvds-v10.dtb
 dtb-$(CONFIG_ARCH_ROCKCHIP) += px30-evb-ddr4-v10.dtb
+dtb-$(CONFIG_ARCH_ROCKCHIP) += rk3308-evb-amic-v10.dtb
```

编译 kenrel 的时候可以直接 make dts-name.img（如 rk3308-evb-amic-v10.img），即可生成对应的 resource.img（包含 dtb 数据）。

9.1.2.3 dts 语法的几个说明

dts 语法可以像 c/c++一样，通过#include xxx.dtsi 来包含其他公用的 dts 数据。dts 文件将继承包含的 dtsi 文件的所有设备节点的属性和值。

如 property 在多个 dts/dtsi 文件被定义，它的值最终为 dts 的定义。所有和芯片相关的控制器节点都会被定义在 soc.dtsi，如需使能该设备功能，需要在 dts 文件中设置其 status 为”okay"。

```
/dts-v1/;
#include "rk3308-evb-v10.dtsi"

/ {
    /* rk3308-evb-v10.dtsi 也有定义该属性，最终值为该 dts 的定义值。 */
    model = "Rockchip RK3308 evb analog mic board";
    compatible = "rockchip,rk3308-evb-amic-v10", "rockchip,rk3308";

    /* 根节点下的节点为新增设备节点 */
    sound {
        compatible = "simple-audio-card";
        ...
    };
};
/* 非根节点下的节点为引用所包含 dtsi 描述的设备节点， 并可以重新设定值*/
&i2s_8ch_2 {
    status = "okay";

    #sound-dai-cells = <0>;
};
```

9.2 内核模块开发文档

docs\ Develop reference documents\目录下分功能模块发布了对应的开发文档，本节主要对这些开发文档进行一个归纳索引，大家结合实际开发遇到的问题，参照以下表格阅读学习对应的开发指南。

模块功能	子目录	对应文档
PWM、背光	PWM	Rockchip 背光控制 开发指南 V0.1-20160729.pdf
安全、加密、OPTEE	SECURITY	Rockchip TEE 安全 SDK 开发指南 V1.0.pdf Rockchip-Secure-Boot-Application-Note.pdf
USB	USB	USB-Initialization-Log-Analysis.pdf Rockchip-USB-SQ-Test-Guide.pdf Rockchip-USB-Performance-Analysis-Guide.pdf Rockchip-Developer-Guide-linux4.4-USB.pdf RK3399-USB-DTS.pdf
UART 开发配置	UART	Rockchip-Developer-Guide-linux4.4-UART.pdf
Trust、ATF	TRUST	Rockchip-Developer-Guide-Trust.pdf
Tsadc、温控、THERMAL	THERMAL	Rockchip Thermal 开发指南-4.4 V1.0.1-20170428.pdf
SPI	SPI	Rockchip-Developer-Guide-linux4.4-SPI.pdf
PMIC、电量计、DC-DC	PMIC	Rockchip-Developer-Guide-Linux4.4-DC-DC.pdf Rockchip RK818_6 电量计 开发指南 V2.0-20170525.pdf Rockchip RK805 开发指南 V1.0-20170220.pdf
GPIO、IOMUX、Pin-Ctrl	Pin-Ctrl	Rockchip Pin-Ctrl 开发指南 V1.0-20160725.pdf
PCIe	PCIe	Rockchip-Developer-Guide-linux4.4-PCIe.pdf
eMMC、SDIO、SDMMC	MMC	Rockchip-Developer-Guide-linux4.4-SDMMC-SDIO-eMMC.pdf
IO-DOMAIN、电源域配置	IO-DOMAIN	Rockchip IO-Domain 开发指南 V1.0-20160630.pdf
I2C	I2C	Rockchip I2C 开发指南 V1.0-20160629.pdf
GMAC、以太网	GMAC	Rockchip 以太网 开发指南 V2.3.1-20160708.pdf
DVFS、CPU Freq	DVFS	Rockchip CPU-Freq 开发指南 V1.0.1-20170213.pdf
CLK、时钟配置	CRU	Rockchip 时钟子模块 开发指南 V1.1-20170210.pdf
Audio	Audio	Rockchip Audio 开发指南 V1.1-20170215-linux4.4.pdf RK3308_Audio_Introduction_v0.1.0_CN.pdf
DDR、DDR 稳定性	DDR	RK DDR Application Note 5 --客户自行验证 DRAM 新料流程.pdf DDR 问题排查手册 1.0.pdf DDR 开发指南.pdf
Display、DRM	DISPLAY	Rockchip 基于 DRM 框架的 HDMI 开发指南 v1.1-20180322.pdf rockchip_drm_integration_helper-zh.pdf Rockchip_DRM_Panel_Porting_Guide_V1.3_20171209

9.3 GPIO

比如 RK3399/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 docs\ Develop reference documents\Pin-Ctrl\目 录下 《 Rockchip Pin-Ctrl 开 发指南 V1.0-20160725.pdf》。

9.4 ARM、GPU、DDR 频率修改

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

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 docs\ Develop reference documents\DVFS\目 录下文档。

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 RK3399/RK3399pro 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
cat /sys/class/devfreq/ff9a0000.gpu/cur_freq
```

9.5 温控配置

RK3399/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 = ; /* millicelsius */
};
```

设置温控上限温度：

```
&target {
    temperature = ; /* millicelsius */
};
```

设置软件关机温度：

```
&soc_crit {
    temperature = ; /* millicelsius */
};
```

配置硬件关机温度：

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

温控的具体说明可以参考 docs\ Develop reference documents\THERMAL\目录下相关文档。

9.6 LPDDR4 配置

rk3399Pro 使用 lpddr4 的 dts 配置请参考文件：arch/arm64/boot/dts/rockchip/rk3399pro-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 = ;
    downdifferential = ;
    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 = ;
    };
    &dmc_opp_table {
        compatible = "operating-points-v2";
    opp-200000000 {
        opp-hz = /bits/ 64 ;
        opp-microvolt = <900000> ;
        status = "disabled";
    };
    opp-300000000 {
        opp-hz = /bits/ 64 ;
        opp-microvolt = <900000> ;
        status = "disabled";
    };
    opp-400000000 {
        opp-hz = /bits/ 64 ;
        opp-microvolt = <900000> ;
        status = "disabled";
    };
    opp-416000000 {
        opp-hz = /bits/ 64 ;
        opp-microvolt = <900000> ;
    };
    opp-528000000 {
        opp-hz = /bits/ 64 ;
        opp-microvolt = <900000> ;
        status = "disabled";
    };
    opp-600000000 {
        opp-hz = /bits/ 64 ;
```

```
            opp-microvolt = <900000>;
            status = "disabled";
        };
        opp-800000000 {
            opp-hz = /bits/ 64 ;
            opp-microvolt = <900000>;
            status = "disabled";
        };
        opp-856000000 {
            opp-hz = /bits/ 64 ;
            opp-microvolt = <900000>;
        };
        opp-928000000 {
            opp-hz = /bits/ 64 ;
            opp-microvolt = <900000>;
            status = "disabled";
        };
        opp-1056000000 {
            opp-hz = /bits/ 64 ;
            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：

9.6.1.1 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 = ; /* 128k */
```

```
};
```

2. 相对应的产品 dts 中引用 iram 节点。

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

9.6.1.2 不需要 LPDDR4 的变频

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 9.6.1.1 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 = ; /* 128k */
};
```

2. 删除 dts 中的如下配置：

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

9.7 SDCard configuration

有些芯片比如 RK3326/RK3399pPRO 的 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-sdr104;
    + status = "okay";
};
```

10 Buildroot 开发

本节简单介绍 Buildroot 开发中一些常见配置的修改，Rockchip Buildroot Linux SDK 使用的是 Buildroot-2018.02 版本。

10.1 Buildroot 开发基础

Buildroot 基础开发指南，见 docs\ Linux reference documents\目录下《The Buildroot User Manual.pdf》。

也可直接访问官网链接浏览：http://buildroot.org/downloads/manual/manual.html

10.1.1 默认配置选择及编译

客户按实际编译环境配置好编译依赖后，按照以下步骤配置完后，执行 make 即可。

```
$ source buildroot/build/envsetup.sh

You're building on Linux
Lunch menu...pick a combo:
1. rockchip_rk3308_release
2. rockchip_rk3308_debug
3. rockchip_rk3308_robot_release
4. rockchip_rk3308_robot_debug
5. rockchip_rk3308_mini_release

Which would you like? [1]
```

如选择 rockchip_rk3308_release，输入对应序号 1。

```
$ make
```

完成编译后执行 SDK 根目录下的 mkfirmware.sh 脚本生成固件

make 执行以下过程

- 下载源码；
- 配置、编译、安装交叉工具链；
- 配置、编译、安装选择的包；
- 安装选择的格式生成根文件系统；

Buildroot 输出结果保存在 output 目录，具体目录由配置文件决定，上面的例子，保存在 buildroot/output/rockchip_rk3308_release 目录，后续编译可以在 buildroot/output/rockchip_rk3308_release 目录或是工程根目录下执行（make menuconfig 也可以在工程根目录下执行），这个目录底下包括几个子目录：

- image：包含压缩好的根文件系统镜像文件。
- build/：包含所有的源文件，包括 Buildroot 所需主机工具和选择的包，这个目录包含所有模块源码。
- staging/:这个目录类似根文件系统的目录结构，包含编译生成的所有头文件和库，以及其他开发文件，不过他们没有裁剪，比较庞大，不适用于目标文件系统。
- target/:包含完整的根文件系统，对比 staging，它没有开发文件，不包含头文件，二进制文件也经过 strip 处理。

- host/:主机端编译需要的工具包括交叉编译工具。

10.1.2 模块配置编译

以上步骤选择了一种默认配置，但不一定满足我们需求，我们可能需要增加一些第三方包，或者修改包的配置选项，Buildroot 支持图形化方式去做选择配置。

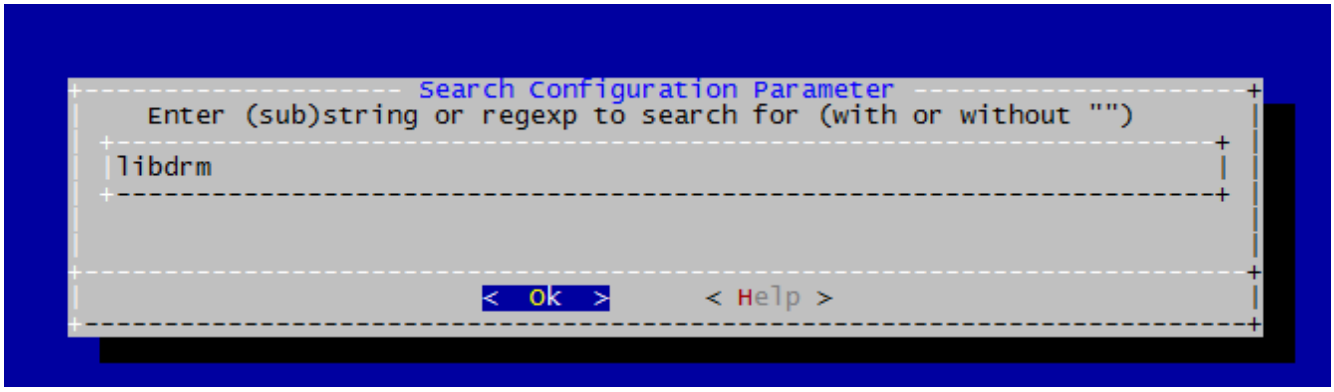
Buildroot 支持 make menuconfig|nconfig|gconfig|xconfig

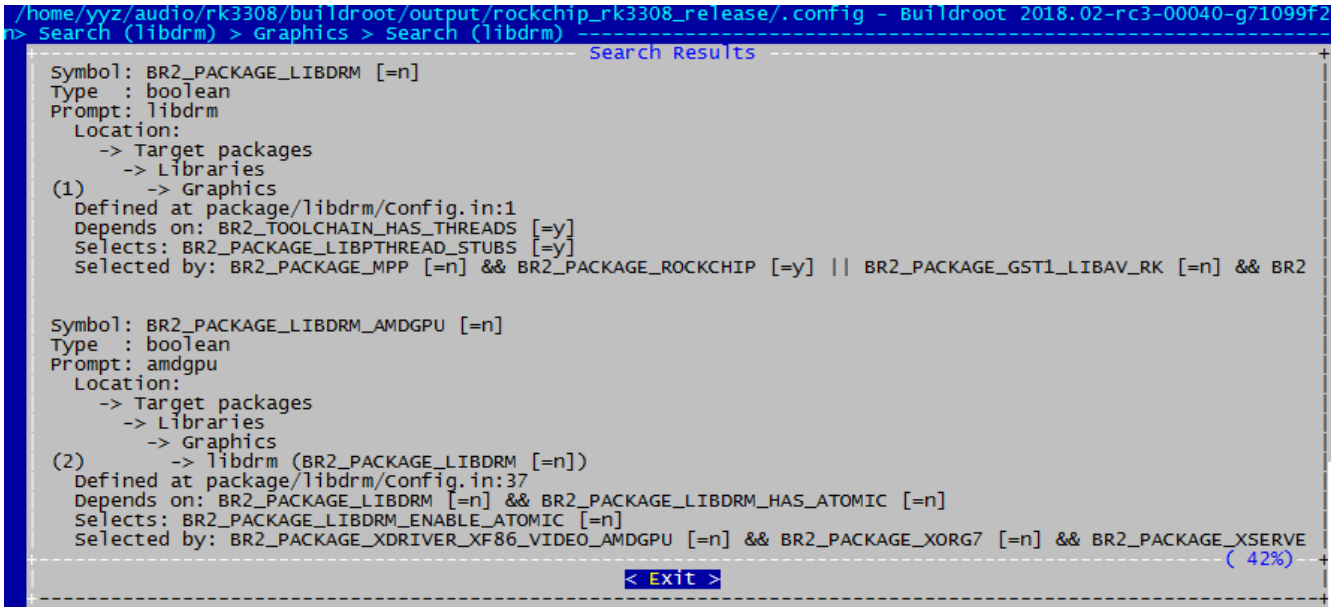
示例，增加 libdrm 支持

执行 source 命令，配好环境变量后，可以在根目录下：

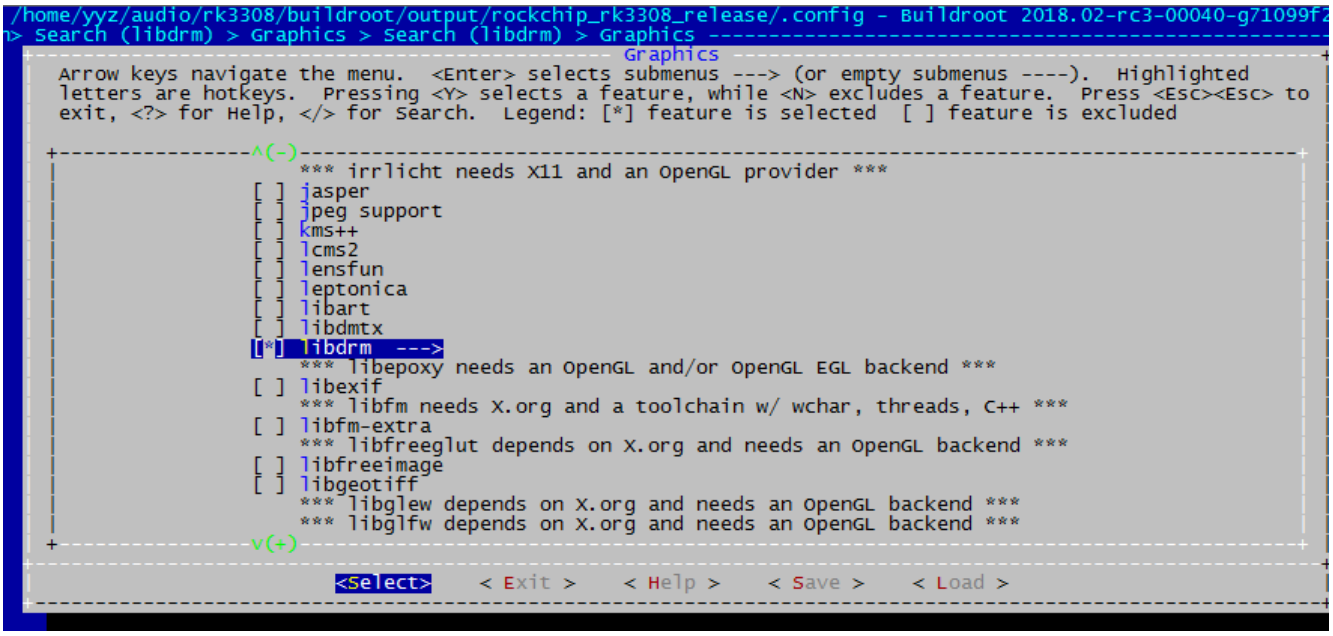
```
make menuconfig
```

输入/，跳出搜索界面如下，输入 libdrm，按回车键：





选择 1，然后按空格选择上。



然后保存退出，保存配置命令，将会修改 buildroot/configs/ 目录下的配置文件；

```
make savedefconfig
```

编译 libdrm，生成 libdrm

```
make libdrm
```

10.1.3 Busybox 配置修改

配置命令：

```
make busybox-menuconfig
```

修改完成后，通过命令保存配置：

```
make busybox-update-config
```

10.1.4 交叉编译工具

Buildroot 编译完成后，会在指定的输出目录 host 目录下生成交叉编译工具，我们可以用来编译目标程序。默认配置生成的交叉编译工具目录为：

```
buildroot/output/rockchip_rk3308_release/host/usr/bin/output/host/usr/bin/
```

我们可以直接用交叉编译工具编译程序

```
./buildroot/output/rockchip_rk3308_release/host/usr/bin/output/host/usr/bin/aarch64-rockchip-linux-gnu-gcc main.c -o test
```

浮点支持（以下配置打开 neon 支持），RK3308 支持 crc/crypto/fp/simd 这几个 feature，配置如下：

CFLAGS += -mcpu=cortex-a35+crc+crypto

10.1.5 新增本地源码包

以上介绍都是在 Buildroot 已有源码包的情况下，我们去选择打开编译即可，如果 Buildroot 没有或者我们自己写的应用该如何集成到 Buildroot 呢？

Buildroot 支持多种模块编译方式，包括 generic-package、cmake-package、autotools-package 等，我们以 generic-package 举例说明；

例子：package/rockchip/alsa_capture

1. 创建目录 package/rockchip/alsa_capture
2. 创建 Config.in

```
config BR2_PACKAGE_ALSA_CAPTURE
    bool "Simple ALSA Capture Demo"
```

3. 创建 alsa_capture.mk，其中源码目录指向 external/alsa_capture/src

```
################################################################################
#
# alsa_capture
#
################################################################################
ifeq ($(BR2_PACKAGE_ALSA_CAPTURE), y)
ALSA_CAPTURE_VERSION:=1.0.0
ALSA_CAPTURE_SITE=$(TOPDIR)/../external/alsa_capture/src
ALSA_CAPTURE_SITE_METHOD=local

define ALSA_CAPTURE_BUILD_CMDS
    $(TARGET_MAKE_ENV) $(MAKE) CC=$(TARGET_CC) CXX=$(TARGET_CXX) -C $(@D)
endef

define ALSA_CAPTURE_CLEAN_CMDS
    $(TARGET_MAKE_ENV) $(MAKE) -C $(@D) clean
endef

define ALSA_CAPTURE_INSTALL_TARGET_CMDS
    $(TARGET_MAKE_ENV) $(MAKE) -C $(@D) install
endef

define ALSA_CAPTURE_UNINSTALL_TARGET_CMDS
    $(TARGET_MAKE_ENV) $(MAKE) -C $(@D) uninstall
endef

$(eval $(generic-package))
endif
```

4. 创建目录 external/alsa_capture/src，编写 alsa_capture.c

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
printf("hello world\n");
return 0;
}
```

5. 编写 Makefile 文件

```
DEPS =
OBJ = alsa_capture.o
CFLAGS = -std=c++11 -lasound
%.o: %.cpp $(DEPS)
    $(CC) -c -o $@ $< $(CFLAGS)

alsa_capture: $(OBJ)
    $(CXX) -o $@ $^ $(CFLAGS)

.PHONY: clean

clean:
    rm -f *.o *~ alsa_capture

.PHONY: install

install:
    cp -f alsa_capture $(TARGET_DIR)/usr/bin/

.PHONY: uninstall

uninstall:
    rm -f $(TARGET_DIR)/usr/bin/alsa_capture
```

6. 在将新建包加入到 Buildroot 编译系统内；
7. 修改 package/rockchip/Config.in 加入下面这行

```
source "package/rockchip/alsa_capture/Config.in"
```

8. 配置选择包

```
make menuconfig 然后选上 alsa_capture 包；
```

9. 编译

```
make alsa_capture
```

10. 打包进文件系统

```
make
```

11. 修改源码后重新编译包

```
make alsa_capture-rebuild
```

10.1.6 fs-overlay

默认编译出来根文件系统，有些配置文件可能不能满足客制化需求，这时候 fs-overlay 就能排上用场，fs-overlay 目录会在编译的最后阶段替换到文件系统目录，打包进根文件系统。fs-overlay 路径由默认配置文件指定：

```
BR2_ROOTFS_OVERLAY="board/rockchip/rk3308/fs-overlay"
```

例如，我们会修改 fstab 文件，增加 debugfs 和 pstore：

```
vi board/rockchip/rk3308/fs-overlay/etc/fstab
# <file system> <mount pt>      <type>  <options>       <dump>  <pass>
/dev/root       /               ext2    rw,noauto       0       1
proc            /proc           proc    defaults        0       0
```

```
devpts          /dev/pts        devpts  defaults,gid=5,mode=620 0       0
tmpfs           /dev/shm        tmpfs   mode=0777       0       0
tmpfs           /tmp            tmpfs   mode=1777       0       0
tmpfs           /run            tmpfs   mode=0755,nosuid,nodev  0       0
sysfs           /sys            sysfs   defaults        0       0
debug           /sys/kernel/debug       debugfs defaults        0       0
pstore          /sys/fs/pstore  pstore  defaults        0       0
```

10.1.7 SDK 常用配置修改

10.1.7.1 Flash 类型修改

配置文件：device/rockchip/rk3308/BoardConfig.mk，默认配置为 nand 设备。

```
# Set flash type.
# support <emmc, nand, spi_nand, spi_nor>
FLASH_TYPE := emmc
```

配置说明：

- EMMC 设备

```
FLASH_TYPE=emmc
```

- NAND 设备

```
FLASH_TYPE=nand
```

10.1.7.2 Rootfs 切换为 ext2

Rootfs 可配置为可读写 ext2 文件系统，方便系统调试使用。

1. 修改 Kernel 中 bootargs 配置：

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3308-evb-v10.dtsi
b/arch/arm64/boot/dts/rockchip/rk3308-evb-v10.dtsi
index d39faf8..549af2e 100644
--- a/arch/arm64/boot/dts/rockchip/rk3308-evb-v10.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3308-evb-v10.dtsi
@ -12,7 +12,7 @
compatible = "rockchip,rk3308-evb", "rockchip,rk3308";
chosen {
-               bootargs = "earlycon=uart8250,mmio32,0xff0c0000 swiotlb=1
console=ttyFIQ0 root=PARTUUID=614e0000-0000 rootfstype=squashfs rootwait";
+               bootargs = "earlycon=uart8250,mmio32,0xff0c0000 swiotlb=1
console=ttyFIQ0 root=PARTUUID=614e0000-0000 rootfstype=ext2 rootwait";
        };
```

2. 修改 device\rockchip\rk3308\rockimg\对应的 parameter 文件，确保 rootfs 分区大小足够存放分区镜像。

3. 修改 device\rockchip\rk3308\BoardConfig.mk 中 rootfs 文件系统类型：

```
diff --git a/BoardConfig.mk b/BoardConfig.mk
index 8adbcdd..3a2be4c 100755
--- a/BoardConfig.mk
+++ b/BoardConfig.mk
@@ -2,8 +2,8 @@
 SDK_ROOT := $(PWD)/../../..

 # Set rootfs type, see buildroot.
-# ext4 squashfs
-ROOTFS_TYPE := squashfs
+# ext4 ext2 squashfs
+ROOTFS_TYPE := ext2
```

4. rootfs 分区 ext2 文件系统镜像会自动打包生成，也可以直接在以下路径获取：

```
buildroot/output/rockchip_rk3308_release/images/rootfs.ext2
```

10.2 Buildroot RK 软件包介绍

Rockchip 平台在 Buildroot 上添加了许多软件包支持，统一放在 buildroot/package/rockchip 目录下，以下简单介绍 SDK 上带的软件包：

```
buildroot/package/rockchip
├── adbd----------------------adb daemon
├── alexaClientSDK------------ alexa 语音 SDK
├── alsa_capture---------------alsa 设备录音 demo 程序
├── cypress_bsa--------------- cypress 蓝牙协议栈及 demo
├── DuerClientSDK-------------百度语音 DuerOS SDK
├── gst1-libav-rk--------------gstreamer libav 插件
├── gstreamer1-iep------------gstreamer iep 插件
├── gstreamer1-rockchip------gstreamer rk 插件
├── libcutils----------------------从 Android 移植过来的 cutils 库
├── libiep-------------------------iep 库
├── libion-------------------------ion 库
├── liblog-------------------------从 Android 移植过来的 log 库
├── LocalPlayer-----------------本地无屏音乐播放器
├── mdev_mount---------------mdev 自动挂载脚本
├── mpp----------------------------mpp 库
├── pcba---------------------------pcba 测试程序
├── pm-suspend-api-------------休眠唤醒多进程 api 以及 demoo 程序
├── rkwifibt-----------------------wifi、bt firmware 及芯片配置
├── rockchip_modules-----------内核 modules 驱动
├── rockchip_test----------------测试脚本
├── rockchip_utils---------------一些通用程序，比如 io 命令
├── rv1108-firmware------------rv1108 固件
├── softap-------------------------wifi 启动 ap 模式
├── softapServer-----------------wifi 网络配置服务
```

```
├── wakeWordAgent------------alexa 唤醒词代理程序
├── wifiAutoSetup----------------wifi smart config 配网程序
└── wpa_supplicant_realtek-----wpa_supplicant realtek 版本
```

10.3 预置系统应用或数据

10.3.1 oem.img 说明

oem.img：oem 分区镜像文件，默认为 ext2 文件系统格式，用来预置客户的应用程序；直接执行打包命令即可打包 oem 分区镜像。

```
./mkfirmware.sh
```

生成 rockdev/oem.img;

10.3.2 oem 镜像打包目录修改

oem 分区默认打包 dueros 目录 device/rockchip/rk3308/oem，可以通过修改以下配置文件修改打包目录和 oem 分区文件系统。

oem 路径目前有三种配置：

- oem：不打包语音算法程序，只有基础功能
- dueros：百度语音 SDK，默认配置
- aispeech：思必驰语音 SDK
- iflytekSDK：科大讯飞语音 SDK

配置文件：device/rockchip/rk3308/BoardConfig.mk

```
# Set oem partition type.
# ext2 squashfs
OEM_PARTITION_TYPE=ext2

#OEM config: oem/dueros/aispeech/iflytekSDK
OEM_PATH=oem
```

10.3.3 userdata.img 说明

userdata.img：userdata 分区镜像文件，默认为 ext2 文件系统格式，用来存放运行时数据，一般是可读写文件；直接执行打包命令即可打包 userdata 分区镜像。

```
mkfirmware.sh
```

生成 rockdev/userdata.img;

10.4 新增分区配置

请参考\RKDocs\RKTools manuals\Android 增加一个分区配置指南 V1.00.pdf

10.5 OTA 升级

10.5.1 编译依赖

Rootfs 系统依赖配置：

```
BR2_PACKAGE_RECOVERYSYSTEM=y
```

Recovery 编译命令：

```
./build.sh recovery
./mkfirmware.sh
```

编译后会在 rockdev 目录下生成 misc.img，及 recovery.img

10.5.2 升级固件制作

参考 12.3 节 统一固件打包，升级固件可支持完整分区分区升级，也可指定分区升级。可修改 package-file 文件，将不要升级的分区去掉，这样可以减少升级包（update.img ）的大小。

注：recovery.img 暂时不支持升级，不可打包进去。

10.5.3 升级

在 rootfs 下，命令行运行：

```
recoverySystem ota /xxx/update.img
```

机器会重启进入 recovery，并进行升级，如升级包放入 U 盘中，可执行以下命令：

```
recoverySystem ota /mnt/usb_storage/update.img
```

可使用的路径

```
U 盘的挂载路径：/mnt/usb_storage/
sdcard 的挂载路径：/mnt/external_sd/
flash 的挂载路径：/userdata/
```

10.6 恢复出厂设置

我们把可以读写的配置文件保存在 userdata 分区，出厂固件会默认一些配置参数，用户使用一段时间后会生成或修改配置文件，有时用户需要清除这些数据，我们就需要恢复到出厂配置。

- 方案：打包固件时，生成 userdata.img，用户请求恢复出厂配置时，将 userdata 分区格式化。
- SDK 实现：功能键 RECOVERY + VOLUMEUP 触发恢复出厂配置，代码请参考：
  buildroot/board/rockchip/rk3308/fs-overlay/etc/input-event-daemon.conf
  board/rockchip/rk3308/fs-overlay/usr/sbin/factory_reset_cfg

10.7 OPTEE 开发

该部分介绍 Rockchip TEE 安全相关固件说明、TEE 环境搭建、CA/TA 开发测试、TA 调试方法、TA 签名方法以及注意事项。

详细开发指南见 docs\ Develop reference documents\SECURITY 目录下《Rockchip TEE 安全 SDK 开发指南 V1.0.pdf》。

10.8 WiFi/WiFi 配网/BT

WiFi 配置，WiFi 配网，BT 相关功能，详细开发指南见 docs\Linux reference documents 目录下《Rockchip Linux WIFI BT 开发指南_V1.00.pdf》。

10.9 音频播放器

10.9.1 音乐播放器 Demo

SDK 提供本地音乐播放器 Demo，该程序包含了初始化、播放、暂停、暂停恢复以及停止播放的 API 接口，可用来参考做音乐播放功能实现。

详细 Demo 源码位置及使用说明见 docs\SoC platform related\RK3308\目录下《RK3308 本地播放器 Demo 说明文档.pdf》。

10.10 LED 控制

SDK 提供 LED 控制 Demo 程序，该程序包含了对 LED 设备初始化、各个模式切换以及反初始化的说明。

本 LED 控制 DemoAPI 支持以下几个模式：

- 模式 1 全亮
- 模式 2 全灭
- 模式 3 旋转
- 模式 4 呼吸灯
- 模式 5 点亮固定个数灯

详细 Demo 源码位置及使用说明见 docs\SoC platform related\RK3308\目录下《RK3308 LED 控制 Demo 说明文档.pdf》。

10.11 互联功能

10.11.1 DLNA

DLNA 的全称是 DIGITAL LIVING NETWORK ALLIANCE(数字生活网络联盟)。旨在解决个人 PC ，消费电器，移动设备在内的无线网络和有线网络的互联互通，使得数字媒体和内容服务的无限制的共享和增长成为可能。

DLNA 的配置及使用、开发说明，详见 docs\Linux reference documents\目录下《Rockchip DLNA 开发指南_V1.00.pdf》。

10.12 休眠唤醒

10.12.1 概述

● 休眠（suspend）是指，系统冻结进程，然后依次挂起设备的电源停止工作，进入低功耗模式；休眠命令：

```
echo mem > /sys/power/state
```

● 唤醒（resume）是指，从休眠模式恢复到正常工作模式。EVB 上 power 键具有唤醒功能，按下 power 键，系统就将唤醒。

有些应用需要在休眠前做一些处理，关闭资源，保存状态，唤醒后再恢复，这就需要系统提供一套休眠唤醒接口，便于应用开发。

10.12.2 单应用模式

单应用是指，系统只有一个应用需要关注休眠唤醒流程，这种情况应用可以直接接管休眠唤醒，进入休眠应用可以直接调用休眠接口进入休眠，事件触发唤醒后，应用继续往下执行代码。类似以下逻辑：

```
int main (int argc, char *argv[])
{
    printf("running...\n");
    printf("enter sleep mode...\n");
    system("echo mem > /sys/power/state");
    printf("app resume running...");
    return 0;
}
```

10.12.3 多应用模式

多应用是指，系统有多个应用需要监听休眠唤醒动作；休眠前，这些应用都需要做休眠处理，唤醒后，也需要做唤醒动作。我们引入 pm-utils 去管理休眠唤醒动作，pm-utils 是一个轻量级的脚本集合，相当于休眠唤醒 HAL。

pm-utils 提供休眠命令 pm-suspend，pm-suspend 执行后，在休眠前会依次执行 /usr/lib/pm-utils/sleep.d/ 目录下的脚本，我们增加了一个脚本，以 dbus-send 的方式同步调用应用的 suspend 接口，调用完成后，系统进入休眠；同样，系统唤醒后，也会依次执行此目录下的脚本，以 dbus-send 的方式调用应用的 resume 接口，并附带唤醒 irq 作为参数。

为了方便开发，我们封装了一个基于 dbus 接口的休眠唤醒 api。

代码目录：package/rockchip/pm-suspend-api

配置选项：BR2_PACKAGE_PM_SUSPEND_API=y

API 接口：

```
SUSPEND_API int request_system_suspend(void);

SUSPEND_API int register_suspend_listener(char *bus_name, CALLBACK
suspend_callback, CALLBACK resume_callback);
```

Demo 应用 1：每运行 10 秒，申请系统进入休眠。

```
#include <stdio.h>
#include "dbus_suspend_api.h"

int main()
{
    printf("hello sleep_test\n");
    while (1) {
        sleep(10);
        printf("time elapse 10 seconds,request sleep\n");
        request_system_suspend();
    }
    return 0;
```

```
}
```

Demo 应用 2：注册休眠唤醒接口，休眠前打印状态，唤醒后打印状态

```
#include <stdio.h>
#include "dbus_suspend_api.h"

int suspend(void *data)
{
    printf("app_test1 suspending\n");
    usleep(100*1000);
    //to do...
    printf("app_test1 suspend succeed\n");
    return 0;
}

int resume(void *data)
{
    printf("app_test1 resumeing \n");
    usleep(100*1000);
    //to do...
    printf("app_test1 resume succeed\n");
    return 0;
}

int main()
{
    printf("hello world\n");
    register_suspend_listener("com.rockchip.suspend.app_test1", suspend,
resume);
    while(1)
        sleep(2);
    return 0;
}
```

另外需要安装 dbus 配置文件到 dbus 目录/etc/dbus-1/system.d/

```
<!DOCTYPE busconfig PUBLIC
 "-//freedesktop//DTD D-BUS Bus Configuration 1.0//EN"
 "http://www.freedesktop.org/standards/dbus/1.0/busconfig.dtd">
<busconfig>
    <policy context="default">
        <allow own="com.rockchip.suspend.app_test1"/>
        <allow send_destination="com.rockchip.suspend.app_test1"/>
        <allow own="com.rockchip.suspend.app_test2"/>
        <allow send_destination="com.rockchip.suspend.app_test2"/>
    </policy>
</busconfig>
```

具体 demo 请参考代码: package/rockchip/pm-suspend-api/src/。

10.13 语音识别 SDK

目前 Buildroot sdk 已经支持 Dueros、思必驰、科大讯飞三种语音套件。

10.13.1 Dueros

DuerOS 配置：
Buildroot：BR2_PACKAGE_DUERCLIENTSDK=y
device/rockchip/rk3308/BoardConfig.mk 配置 OEM_PATH=dueros
Dueros 配置、使用及开发说明，详见 docs\SoC platform related\RK3308\目录下《RK3308 DuerOS 使用说明.pdf》。

10.13.2 思必驰

思必驰配置：
device/rockchip/rk3308/BoardConfig.mk 配置 OEM_PATH=aispeech

10.13.3 科大讯飞

科大讯飞配置：
Buildroot：BR2_PACKAGE_IFLYTEKSDK=y
device/rockchip/rk3308/BoardConfig.mk 配置 OEM_PATH=iflytekSDK

10.13.3.1 VAD+讯飞识别 Demo

RK3308 VAD+讯飞识别 Demo 程序集成了语音检测（VAD，Voice Activity Detection）和科大讯飞前端处理识别模块，可以实现 VAD 唤醒并送至科大讯飞识别唤醒。
详细 Demo 源码位置及使用说明见 docs\ SoC platform related\RK3308\目录下《RK3308 VAD+讯飞识别 Demo 说明文档.pdf》。

11 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。

11.1 ADB 工具

11.1.1 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硬件设备

ADB 是一个“客户端－服务器端”程序，其中客户端主要是指 PC，服务器端是 Android 设备的实体机器或者虚拟机。根据 PC 连接 Box 机器的方式不同，ADB 可以分为两类：

- 网络 ADB：主机通过有线/无线网络（同一局域网）连接到硬件设备
- USB ADB：主机通过 USB 线连接到硬件设备

11.1.2 Buildroot 配置

```
Symbol: BR2_PACKAGE_ADBD [=y]
Type  : boolean
Prompt: adbd porting for Linux
  Location:
    -> Target packages
(1)   -> rockchip BSP packages (BR2_PACKAGE_ROCKCHIP [=y])
  Defined at package/rockchip/adbd/Config.in:1
  Depends on: BR2_PACKAGE_ROCKCHIP [=y] && BR2_PACKAGE_LIBCUTILS [=y]
```

图 11-1 adb buildroot 对应配置

11.1.3 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、机器已经运行 Android 系统， 设置->开发者选项->已连接到计算机 打开，usb 调试开关打开。（如果是 Linux 系统，配置已默认打开）

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机器相连。

```
adb shell
```

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

11.1.4 ADB 窗口乱码显示

如有遇到下图所示的乱码情况，主要是由于 Linux 系统输出颜色字符，该工具并无法正常解析导致的。

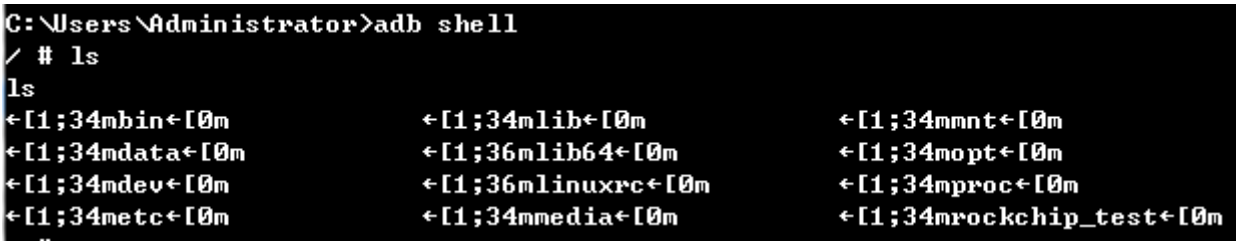


图 11-2 adb buildroot 对应配置

有两种解决方案：

- 使用 puTTY 工具进行 adb 调试。
- 命令行下输入以下命令，去除颜色显示。

```
alias ls='ls --color=never'
```

11.1.5 ADB 常用命令详解

（1）查看设备情况

查看连接到计算机的设备或者模拟器：

```
adb devices
```

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

（2）进入设备和模拟器的 shell

进入设备或模拟器的 shell 环境：

```
adb shell
```

（3）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”文件上传到设备系统的“/system/app”目录下。

（4）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：将设备系统里“/system/app”目录下的文件或文件夹下载到本地“F:\”目录下。

11.2 Lrzsz 工具

lrzsz 是一款 Linux 下面的文件传输工具。实现原理是通过 Xmodem / Ymodem / Zmodem 协议传输文件。lrzsz 可以在支持这三种协议的 Shell 界面的工具下工作，比如 SecureCRT，puTTY 等。

11.2.1 Buildroot 配置

```
Symbol: BR2_PACKAGE_LRZSZ [=y]
Type  : boolean
Prompt: lrzsz
  Location:
    -> Target packages
(1)   -> Networking applications
  Defined at package/lrzsz/Config.in:1
  Depends on: !BR2_STATIC_LIBS [=n]
```

图 11-3 lrzsz buildroot 对应配置

11.2.2 使用说明

sz 命令发送文件到本地：

```
sz filename
```

rz 命令本地上传文件到服务器：

```
rz
```

执行该命令后，在弹出框中选择要上传的文件即可。

说明：打开 SecureCRT 软件 -> Options -> session options -> X/Y/Zmodem 下可以设置上传和下载的目录。

11.3 Procrank 工具

Procrank 是一款调试工具，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。

11.3.1 Buildroot 配置

```
Symbol: BR2_PACKAGE_PROCRANK_LINUX [=n]
Type  : boolean
Prompt: procrank_linux
  Location:
    -> Target packages
(1)   -> System tools
  Defined at package/procrank_linux/Config.in:1
```

图 11-4 procrank buildroot 对应配置

11.3.2 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。

11.3.3 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
```

11.3.4 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。

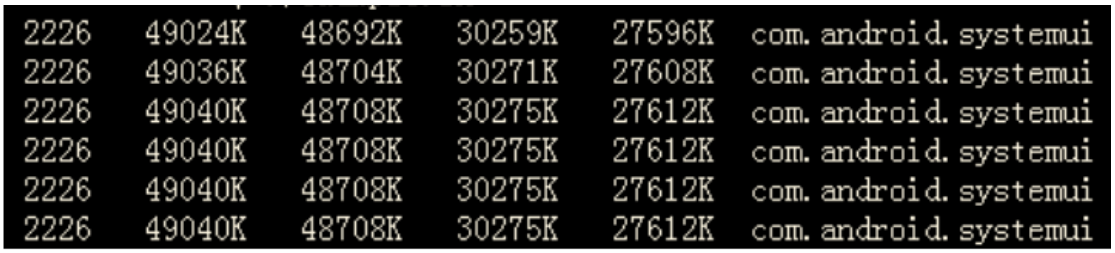


图 11-5 跟踪进程内存状态

11.4 FIQ

FIQ debugger 是集成到内核中的一种系统调试手段。

一般情况下串口是普通的 console 模式，SecureCRT 或 puTTY 下输入切换命令"f+i+q"，串口会切换到 FIQ debugger 模式。

因为 FIQ 是不可屏蔽中断，所以这种调试手段适合调试 cpu 被 hang 住的情况，可以在 hang 住的时候用 FIQ debugger 打印出 cpu 的故障现场，常用命令是 sysrq。

Fiq debugger 相关使用命令：

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

11.5 Last_log

```
cat /sys/fs/pstore/console-ramoops-0
```

打印出上次系统复位前的设备信息。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系统运行状态的日志。

11.6 i2c-tools

11.6.1 Buildroot 配置

```
Symbol: BR2_PACKAGE_I2C_TOOLS [=y]
Type  : boolean
Prompt: i2c-tools
  Location:
    -> Target packages
(1)   -> Hardware handling
  Defined at package/i2c-tools/Config.in:1
  Depends on: BR2_PACKAGE_BUSYBOX_SHOW_OTHERS [=y]
  Selected by: BR2_PACKAGE_EEPROG [=n] && !BR2_SKIP_LEGACY [=n]
```

图 11-6 i2c-tool buildroot 对应配置

11.6.2 使用说明

Buildroot 配置 i2c-tools 后，rootfs 会集成以下四个工具：

- i2cdetect
- i2cdump
- i2cget
- i2cset

i2cdetect 列举 I2C bus：

```
# i2cdetect -l
i2c-0   i2c             imx-i2c                 I2C adapter
i2c-1   i2c             imx-i2c                 I2C adapter
i2c-2   i2c             imx-i2c                 I2C adapter
```

列举 I2C bus i2c-1 上面连接的所有设备

```
# i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- UU -- -- -- -- -- -- --
10: -- UU -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- 3a -- -- -- -- --
40: -- -- -- -- -- -- -- -- UU -- -- -- -- -- -- --
50: UU -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

发现 I2C 设备的位置显示为 UU 或者表示设备地址的数值，UU 表示该设备在 driver 中被使用。

i2cdumpdump I2C 设备大批量 register 的值：

```
# i2cdump -y -f 1 0x3a
No size specified (using byte-data access)
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: eb 00 7f 05 3d 00 00 00 08 06 00 00 00 00 00 00    ?.??=...??......
10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
20: 01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10    ?#Eg????????vT2?
30: f0 e1 d2 c3 00 00 00 00 00 00 00 00 00 00 00 00    ????............
40: 80 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00    ?.?.............
50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
```

```
f0: 00 00 00 00 00 00 00 00 00 00 30 00 00 00 00    ..........0....
```

i2cget 读取 I2C 设备某个 register 的值：

```
# i2cget -y -f 1 0x3a 0x02
0x7f
```

i2cset 设置 I2C 设备某个 register 的值：

```
[cpp] view plain copy
# i2cset -y -f 1 0x3a 0x02 0x05
```

11.7 io

11.7.1 Buildroot 配置

```
Symbol: BR2_PACKAGE_ROCKCHIP_UTILS [=y]
Type  : boolean
Prompt: rockchip utils
  Location:
    -> Target packages
      -> rockchip BSP packages (BR2_PACKAGE_ROCKCHIP [=y])
  Defined at package/rockchip/rockchip_utils/Config.in:1
  Depends on: BR2_PACKAGE_ROCKCHIP [=y]
```

图 11-7 io buildroot 对应配置

11.7.2 使用说明

Buildroot 配置 io 后，rootfs 会集成 io 工具。io 命令可以动态的读取或是配置寄存器的值，使用说明如下：

```
io 0x1000                Reads one byte from 0x1000
io 0x1000 0x12           Writes 0x12 to location 0x1000
io -2 -l 8 0x1000        Reads 8 words from 0x1000
io -r -f dmp -l 100 200  Reads 100 bytes from addr 200 to file
io -w -f img 0x10000      Writes the whole of file to memory
```

12 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 tools 目录下各工具附带文档，及 docs\ RKTools manuals 目录下工具文档。

12.1 压力测试工具 –rockchip_test

SDK 中集成了 Linux 平台使用的压力测试工具，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。

12.1.1 Buildroot 配置

```
Symbol: BR2_PACKAGE_ROCKCHIP_TEST [=y]
Type  : boolean
Prompt: stress test tools
  Location:
    -> Target packages
(1)   -> rockchip BSP packages (BR2_PACKAGE_ROCKCHIP [=y])
  Defined at package/rockchip/rockchip_test/Config.in:1
  Depends on: BR2_PACKAGE_ROCKCHIP [=y]
```

图 12-1 rockchip_test 对应配置

12.1.2 使用说明

Buildroot 配置 rockchip_test 后，rootfs 会集成 rockchip_test 压力测试用的所有脚本及依赖。

执行 rockchip_test/rockchip_test.sh 脚本进行测试。

模块相关

- 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

- 休眠唤醒拷机测试。
- 重启拷机测试
- 恢复出厂设置拷机测试。
- Arm 变频测试
- DDR 压力测试
- Flash 读写压力测试

12.2 PCBA 测试工具

PCBA 测试用于帮助在量产的过程中快速地甄别产品功能的好坏，即重点 FCT（Functional Test）测试，进而提高生产效率。PCBA 测试工具在 Window 系统下开发，仅支持在 Window 系统下运行，配合设备端测试程序，即可以验证所需要重点关注的功能或器件的完整与好坏。

目前测试项包括：SD 卡测试、wifi 测试、蓝牙测试、DDR 测试、环麦测试、USB host 测试、led 灯测试、放音测试、录音测试、按键测试、PDM Mic 测试、Audio Line in 测试、SPDIF IN/OUT 测试等，后续还可以添加扩展的测试项。

这些测试项目包括自动测试项和手动测试项，其中 SD 卡测试、wifi 测试、蓝牙测试、DDR 测

试、环麦测试、USB host 测试为自动测试项目；led 灯测试、放音测试、录音测试、按键测试、PDM Mic 测试、Audio Line in 测试、SPDIF IN/OUT 测试为手动测试项目。

自动测试项目无需人工干预测试结束后会直接上报测试结果并显示通过与否，人工测试项目需要人为判断测试项是否正确完成，并给出判断（通过或不通过）。

工具使用说明详见 docs\ Linux reference documents\目录下《Rockchip_PCBA 测试开发指南_1.01.pdf》

12.3 统一固件打包工具

固件打包工具可将各零散镜像文件，打包成一个完成的 update.img 形式，方便量产烧写及升级。

12.3.1 Windows 下打包

Windows 系统下，打包工具存放在 tools\windows\AndroidTool\rockdev，打包步骤如下：

1）打开 rockdev 目录，编辑 package-file。

按照 package-file 进行配置，package-file 里面配置“*.img”镜像放在 Image 目录底下的，将需要放到 Image 目录的镜像拷贝进去即可。且注意配置时，镜像名字的准确。其中注意 bootloader 选项，应该根据自己生成的 loader 名称进行修改。

2）编辑 mkupdate.bat

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

RKImageMaker.exe -RK3308 Image\MiniLoaderAll.bin  Image\update.img update.img -os_type:androidos

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

```

图 12-2 update.img 打包脚本

需要修改 loader 名称为实际存放的 loader 名称即可。

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

12.3.2 Linux 下打包

Linux 系统下，打包工具存放在 w:\RK3308\tools\linux\Linux_Pack_Firmware\rockdev\，打包步骤如下：

1）打开 rockdev 目录，编辑 package-file。

按照 package-file 进行配置，package-file 里面配置“*.img”镜像放在 Image 目录底下的，将需要放到 Image 目录的镜像拷贝进去即可。且注意配置时，镜像名字的准确。其中注意 bootloader 选项，应该根据自己生成的 loader 名称进行修改。

2）编辑 mkupdate.sh

```
./afptool -pack ./ Image/update.img || pause
./rkImageMaker -RK3308 Image/MiniLoaderAll.bin Image/update.img update.img -os_type:androidos || pause
echo "Making update.img OK."
#echo "Press any key to quit:"
#read -n1 -s key
exit $?
```

图 12-3 update.img 打包脚本

需要修改 loader 名称为实际存放的 loader 名称即可。

3）在 rockdev 目录下，执行以下命令，运行完会在当前目录生成一个 update.img。

```
./mkupdate.sh
```

12.4 固件签名工具

固件签名工具说明待更新。

12.5 序列号/Mac/厂商信息烧写-WNpctool 工具

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

12.5.1 WNpctool 写入步骤

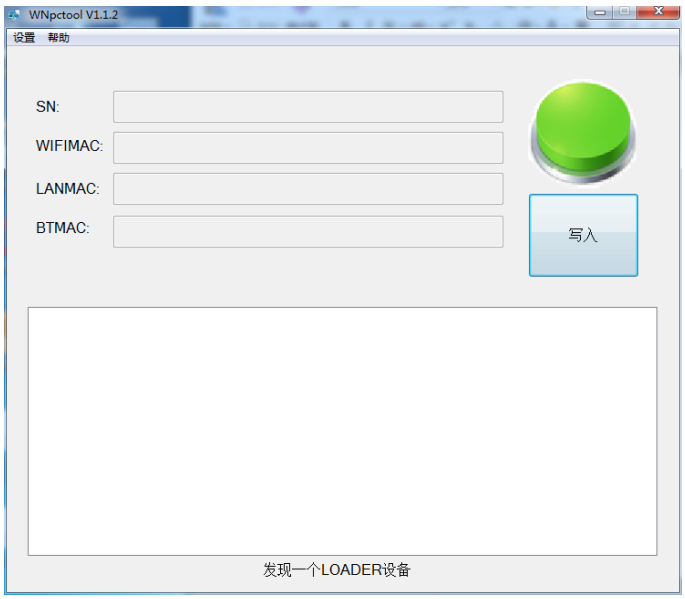


图 12-4 WNpctool 工具

1）进入 loader 模式。

2）点击设置按钮，会有一个下拉框按钮，点击“读取”按钮，用来切换是写入还是读取功能。切换到写入功能。

3）点击模式，出现下列窗口，用来设置 SN/WIFI/LAN/BT

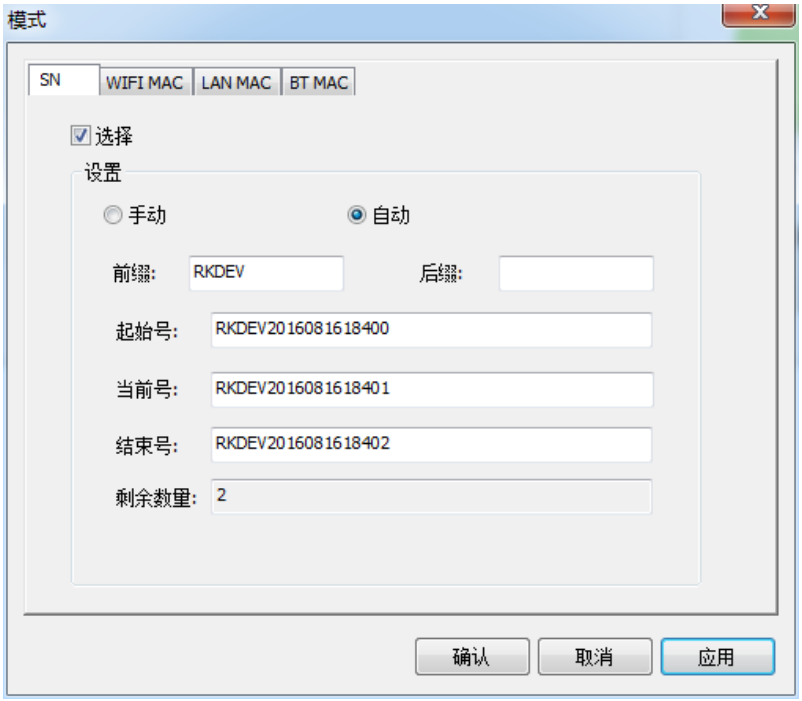


图 12-5 WNpctool 工具模式设置

4）设置完成后，点击应用按钮，关闭窗口，返回主窗口，点击写入按钮即可。

12.5.2 WNpctool 读取步骤

1）进入 loader 模式。

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

3）点击“读取”即可。

12.6 EQ_DRC 工具

EQ_DRC 工具为 RK3308 的语音音效实时调参工具 EQ_DRC 工具，使用该工具可以实时调整板端的各类音频参数。

工具使用说明详见 docs\ RKTools manuals\目录下

《RK3308_EQ_DRC_TOOL_V1.1_20180510.pdf》，通过实例介绍 EQ_DRC 调参工具的使用方法和注意事项。

13 附录

13.1 SSH 公钥操作说明

13.1.1 SSH 公钥生成

使用如下命令生成：

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

请将 user@host 替换成您的邮箱地址。

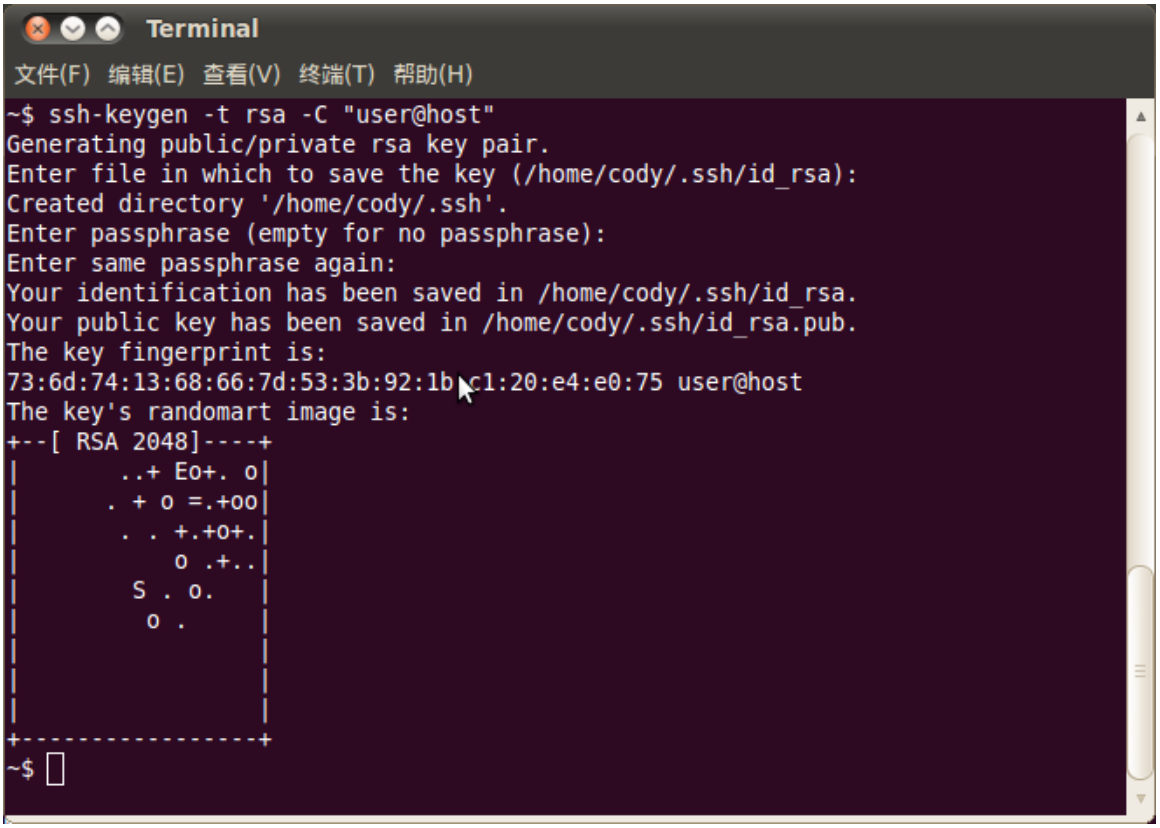


图 13-1 公钥生成

命令运行完成会在你的目录下生成 key 文件。

```
~$ ls -l .ssh/
总用量 8
-rw------- 1 cody cody 1675 2012-10-15 11:38 id_rsa
-rw-r--r-- 1 cody cody  391 2012-10-15 11:38 id_rsa.pub
```

图 13-2 .ssh 目录文件

请妥善保存生成的私钥文件id_rsa和密码，并将id_rsa.pub发邮件给fae@rock-chips.com，申请开通 SDK 代码下载权限。

13.1.2 SSH 公钥生成使用 key-chain 管理密钥

推荐您使用比较简易的工具 keychain 管理密钥。

具体使用方法如下：

1. 安装 keychain 软件包：

```
$sudo aptitude install keychain
```

2. 配置使用密钥：

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

增加下面这行：

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

其中，id_rsa 是私钥文件名称。

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

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

13.1.3 多台机器使用相同 ssh 公钥

在不同机器使用，可以将你的 ssh 私钥文件 id_rsa 拷贝到要使用的机器的“~/.ssh/id_rsa”即可。在使用错误的私钥会出现如下提示，请注意替换成正确的私钥。

```
~/tmp$ git clone git@172.16.10.211:rk292x/mid/4.1.1_r1
Initialized empty Git repository in /home/cody/tmp/4.1.1_r1/.git/
The authenticity of host '172.16.10.211 (172.16.10.211)' can't be established.
RSA key fingerprint is fe:36:dd:30:bb:83:73:e1:0b:df:90:e2:73:e4:61:46.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.10.211' (RSA) to the list of known hosts.
git@172.16.10.211's password:
```

图 13-3 错误私钥异常提示

添加正确的私钥后，就可以使用 git 克隆代码，如下图。

```
~$ cd tmp/
~/tmp$ git clone git@172.16.10.211:rk292x/mid/4.1.1_r1
Initialized empty Git repository in /home/cody/tmp/4.1.1_r1/.git/
The authenticity of host '172.16.10.211 (172.16.10.211)' can't be established.
RSA key fingerprint is fe:36:dd:30:bb:83:73:e1:0b:df:90:e2:73:e4:61:46.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.10.211' (RSA) to the list of known hosts.
remote: Counting objects: 237923, done.
remote: Compressing objects: 100% (168382/168382), done.
Receiving objects:   9% (21570/237923), 61.52 MiB | 11.14 MiB/s
```

图 13-4 正常 clone 代码

添加 ssh 私钥可能出现如下提示错误。

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

在 console 输入如下命令即可解决。

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

13.1.4 一台机器切换不同 ssh 公钥

可以参考 ssh_config 文档配置 ssh。

```
~$ man ssh_config
```

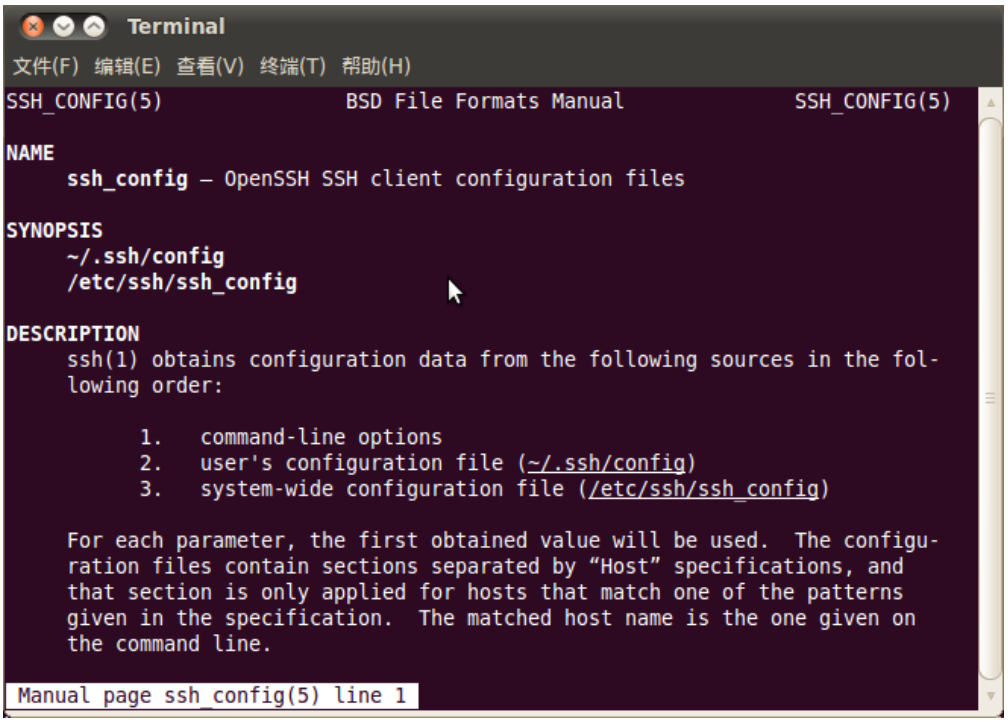


图 13-5 ssh_config 配置说明

通过如下命令，配置当前用户的 ssh 配置。

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

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

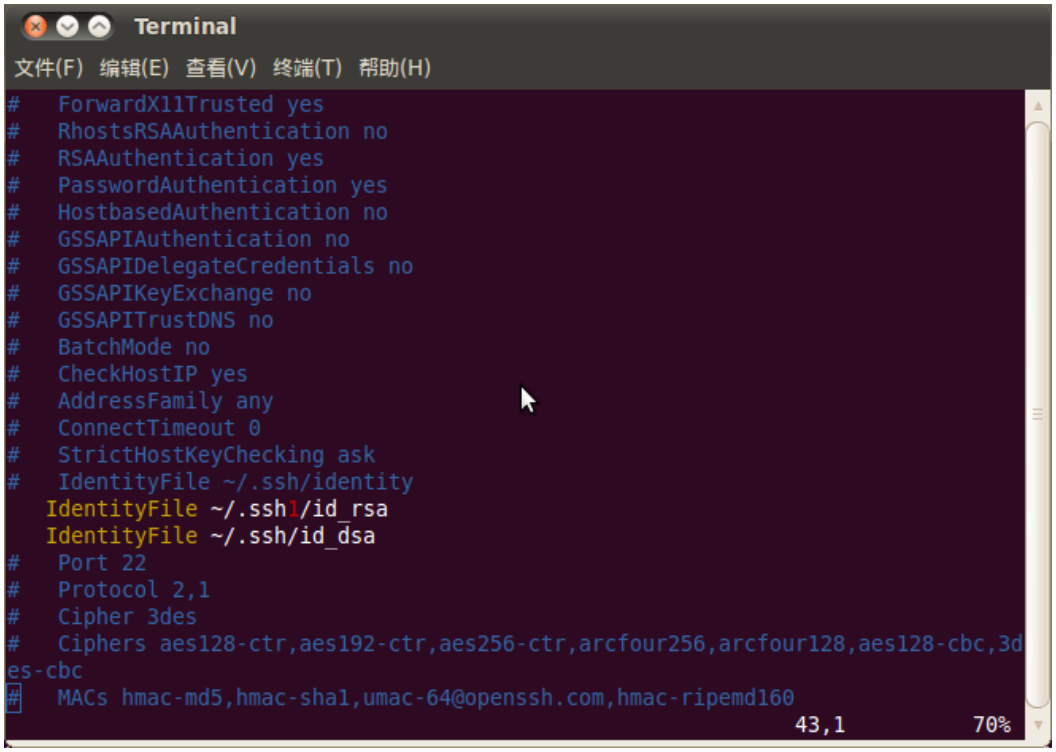


图 13-6 .ssh/config 配置修改

13.1.5 密钥权限管理

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

请妥善保管私钥文件。并不要二次授权与第三方使用。

13.1.6 Git 权限申请说明

参考上述章节，生成公钥文件，发邮件至<fae@rock-chips.com>，申请开通 SDK 代码下载权限。