Rockchip
瑞芯微电子

RK3368 Android7.1-Box
软件开发指南

发布版本:V1.01

日期:2017.08

前言

概述

文档主要介绍 Rockchip RK3368Android7.1-Box 软件开发指南，旨在帮助软件开发工程师更快上手 RK3368 的开发及调试。

产品版本

芯片名称	内核版本	Android 版本
RK3368	Linux4.4	Android7.1.2

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
2017-07-19	V1.00	YHC	CW,ZXZ	创建初始发布版本
2017-08-24	V1.01	YHC	ZY,CW,ZXZ	1. 增加显示框架配置说明 2. 系统分区配置与 parameter 自动关联 3. drmservice 的 demo 拷贝功能 4. 相关配置更新说明

目录

插图目录

表格目录

1 支持列表

1.1 DDR 支持列表

RK3368 DDR 支持类型有如下表所示。

表 1-1 RK3368 DRAM Support Type

Chip	DRAM Support Type
RK3368	DDR3/LPDDR2/LPDDR3

RK3368 DDR 颗粒支持程度列表，详见 RKDocs\Platform support lists 目录下《RK DDR Support List Ver2.27》，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 RKDDR 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

RK3368 兼容 eMMC 5.0/5.1，SDIO3.0，可运行 HS200 模式，详见 RKDocs\Platform support lists 目录下《RKeMMCSupportList Ver1.33_20170215》，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 RK 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，重点关注下厂商 Datasheet 中 performance 一章节。

如有选型上的疑问，也可直接联系我们的 FAE 窗口。

1.3 WiFi/BT 支持列表

RK3368 内核运行 Linux4.4，WiFi/BT 支持列表，详见 RKDocs\Platform support lists 目录下《Rockchip_WiFi_Situation_20170214》，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。

1.4 SDK 软件包适用硬件列表

本 SDK 是基于谷歌 Android7.1.2 64bit 系统，适配瑞芯微 RK3368 芯片的软件包，适用于 RK3368 Box 开发板及基于其上所有的开发产品。

SDK 附带了 RK3368 Box 硬件设计指南，见 RKDocs\Develop reference documents\RK3368 Hardware Design Guide for BOX_V10_201510.pdf。

1.5 多媒体编解码支持列表

RK3368 多媒体规格如下表所示:

表 1-4 RK3368 多媒体编解码支持列表

多媒体支持	H.264 4K@25fps,1080P@60fps
	H.265 4K@60fps
	1080P 多格式视频解码 (VC-1, MPEG-1/2/4, VP8)
	1080P H.264 VP8 格式视频编码
	支持 JPEG 编解码

具体的编解码支持列表，详见 RKDocs\Platform support lists 目录下《RK3368 Multimedia Codec Benchmark v1.3.pdf》。

1.6 Kodi 支持程度列表

在 Android7.1 上的 OMX 框架支持最新的官方原生 Kodi 发布版本（16.1Jarvis 及 17.0Krypton），RK 平台上的 OMX 框架支持最高 4K 60fps 的播放性能，并且最新的 AudioTrack 框架支持 17.0 Krypton 版本音频透传。

想获得更多的支持特性和更好的性能体验，我们推荐使用基于 Rockchip 芯片优化的 RKMC 版本。RKMC 是基于 Kodi 官方稳定发布版本 16.1Jarvis 进行深度优化和定制的开源发布版本，该优化版本将默认集成至 SDK 中。该版本较官方原生版本有如下优势和特性支持：

- 原生硬件加速，支持 H265/H264/VP9
- ISO/MKV Framepacking MVC3D 支持
- Exodus、Phoenix 等 TOP10 插件支持
- AC3、EAC3、TrueHD、DTS、DTSHD 最高到 7.1 声道透传支持
- 硬件去噪、硬件去交错支持

在开发支持上，拥有开源 github 的讨论和支持，开源地址为 https://github.com/JamesLinEngineer/RKMC，后期维护和修改同步实时更新。并且 17.0 的硬件加速版本已在开发。

具体的 Kodi 支持程度列表，详见 RKDocs\Platform support lists 目录下《Rockchip Kodi 支持程度列表_V1.0_20170217》。

2 文档/工具索引

2.1 文档索引

随 RK3368 Box SDK 发布的文档旨在帮助开发者快速上手开发及调试，文档中涉及的内容并不能涵盖所有的开发知识和问题。文档列表也正在不断更新，如有文档上的疑问及需求，请联系我们的 FAE 窗口。

RK3368 SDK 中在 RKDocs 目录下附带了 Develop reference documents（开发指导文档）、Platform support lists（平台支持列表）、RKTools manuals（工具使用文档）。

```
RKDocs/
├─── Develop reference documents
│    ├─── Camera_for_RockChipSDK 参考说明_v4.1.pdf
│    ├─── RealTek wifi 驱动移植说明_V1.1.pdf
│    ├─── RK3368 Hardware Design Guide for BOX_V10_201510.pdf
│    ├─── RK USB Compliance Test Note V1.2.pdf
│    ├─── Rockchip_android7.1_wifi_配置明 V1.4.pdf
│    ├─── Rockchip Audio 开发指南 V1.0-20160606.pdf
│    ├─── Rockchip Box 媒体中心使用说明-v1.0.1-20170216.pdf
│    ├─── Rockchip CPU-Freq 开发指南 V1.0.1-20170213.pdf
│    ├─── Rockchip DEVFreq 开发指南 V1.0-20160701.pdf
│    ├─── Rockchip DRM Panel Porting Guide.pdf
│    ├─── Rockchip I2C 开发指南 V1.0-20160629.pdf
│    ├─── Rockchip IO-Domain 开发指南 V1.0-20160630.pdf
│    ├─── RockChip_LCD 开发文档 v1.6.pdf
│    ├─── Rockchip Pin-Ctrl 开发指南 V1.0-20160725.pdf
│    ├─── Rockchip Recovery 用户操作指南 V1.00.pdf
│    ├─── Rockchip RK818 电量计 开发指南 V1.0-20160725.pdf
│    ├─── Rockchip SDMMC SDIO eMMC 开发指南 V1.0-20160630.pdf
│    ├─── Rockchip Secure Boot Application Note_v1.7_20170519.pdf
│    ├─── Rockchip SPI 开发指南 V1.0-20160629.pdf
│    ├─── Rockchip_TEE 安全 SDK 开发手册_V1.1_20170516.pdf
│    ├─── Rockchip Thermal 开发指南 V1.0.1-20170428.pdf
│    ├─── Rockchip UART 开发指南 V1.0-20160629.pdf
│    ├─── Rockchip U-Boot 开发指南 V3.7-20160708.pdf
│    ├─── Rockchip USB 开发指南 V1.0-20160704.pdf
│    ├─── Rockchip Vendor Storage Application Note.pdf
│    ├─── Rockchip RK3368 Datasheet V1.5 20170612.pdf
│    ├─── Rockchip 以太网 开发指南 V2.3.1-20160708.pdf
│    ├─── Rockchip 休眠唤醒 开发指南 V0.1-20160729.pdf
│    ├─── Rockchip 时钟子模块 开发指南 V1.0-20160630.pdf
│    ├─── Rockchip 电源 独立 DCDC 开发指南 V1.0-20170519.pdf
│    ├─── Rockchip 背光控制 开发指南 V0.1-20160729.pdf
│    ├─── Rockchip 量产烧录 指南 V1.0-20160718.pdf
│    └─── 压力测试 Stresstest 文档 forVR_ver3.0.pdf
```

```
├── Platform support lists
│   ├── RK3368 Multimedia Codec Benchmark v1.3.pdf
│   ├── RK DDR Support List Ver2.27.pdf
│   ├── RKeMMCSupportList Ver1.33_20170215.pdf
│   ├── RKISPV1_Camera_Module_AVL_v1.3.pdf
│   ├── RKISPV1_Camera_User_Manual_v2.0.pdf
│   ├── Rockchip Kodi 支持程度列表_V1.0_20170217.pdf
│   └── Rockchip_WiFi_Situation_20170214.pdf
└── RKTools manuals
    ├── Android 增加一个分区配置指南 V1.00.pdf
    ├── Android 开发工具手册.pdf
    ├── Recovery 升级相关文档.rar
    ├── REPO 镜像服务器搭建和管理_V2.2_20131231.pdf
    ├── RK SDK OTA 包生成方法.pdf
    ├── RK 固件升级失败原因分析_V1.1 20121122.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 厂测工具操作说明 V2.0.pdf
    ├── Rockchip Parameter File Format Ver1.3.pdf
    ├── Rockchip PCBA 模块 开发指南--20170210.pdf
    ├── Rockchip Recovery 用户操作指南 V1.01.pdf
    ├── WNpctool 简要使用说明_V1.1.0_0920.pdf
    ├── 压力测试 Stresstest 文档_ver1.1.pdf
    └── 量产工具升级及相关问题处理.pdf
```

2.2 工具索引

随 RK3368 Box SDK 发布的工具，用于开发调试阶段及量产阶段。工具版本会随 SDK 更新不断更新，如有工具上的疑问及需求，请联系我们的 FAE 窗口。

RK3368 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_SecureBoot
│   ├── Linux_TA_Sign_Tool.rar
│   └── Linux_Upgrade_Tool
└── windows
    ├── AndroidTool
    │   ├── AndroidTool_Release_v2.42
    │   └── rockdev
    ├── Demo 镜像烧写工具包.zip
    ├── DriverAssitant_v4.5.zip
    ├── Efuse_Tool_V1.36
    ├── FactoryTool-v1.45e.rar
    ├── FWFactoryTool-5.3.rar
    └── OemTool_v1.2.rar
```

```
├── rk312x-pcba-tools.rar
├── RKImageMaker_v1.62.zip
├── Rockchip Box 厂测工具 V2.0-M-20170327.zip
├── Rockchip 平台 DDR 测试工具_V1.35 发布通知.7z
├── SD_Firmware_Tool._v1.46.zip
├── SecureBootTool_v1.83_foruser.rar
├── SpiImageTool_v1.33.zip
├── UpgradeDllTool_v1.35.zip
├── Windows_TA_Sign_Tool.rar
└── WNpctool_Setup_V1.1.2_1226.rar
```

3 SDK 编译/烧写

本章节参考源码工程 RKDocs 目录下的 SDK 发布说明文档中有关编译烧写的章节。

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\Develop reference documents 目录下《Rockchip U-Boot 开发指南 V3.7-20160708.pdf》。

4.1 Rockchip U-Boot 简介

Rockchip U-Boot 是基于开源的 U-Boot 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。

4.3 U-Boot 编译

本章节参考源码工程 RKDocs 目录下的 SDK 发布说明文档中有关 U-Boot 编译的章节。

5 内核开发常见配置

本节简单介绍内核一些常见配置的修改，主要是 DTS 的配置，帮助客户更快更方便的进行一些简单的修改。

5.1DTS 配置

5.1.1 WiFi&BT 配置

本平台上 WiFi、BT 有提供自动兼容方案，即一套固件可以支持多个 WiFi 模块。但当前发布的 SDK 所采用的还是原来将 WiFi 驱动编译进内核的方式，如果客户需要使用 WiFi 自动兼容的方案，只需按照 RKDocs\Develop reference documents 目录下《Rockchip Android7.1_WiFi_配置说明 V1.4》及《RealTek wifi 驱动移植说明_V1.1.pdf》提到的注意事项进行修改即可。

5.1.2 GPIO 对应关系注意

关于原理图上的 gpio 跟 dts 里面的 gpio 的对应关系，例如 GPIO4c0，那么对应的 dts 里面应该是"gpio4 16"。因为 GPIOA 有 8 个 pin，GPIOB 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。

5.2ARM、GPU 频率修改

参考 RKDocs\Develop reference documents 目录下《Rockchip DEVFreq 开发指南 V1.0-20160701.pdf》

5.3U-Boot logo 相关的配置

5.3.1 U-Boot logo 开关配置

在对应的 dts 内，对如下代码中的 status 进行更改，可以关闭或打开 uboot logo。

```
&route_hdmi {
    status = "okay";
};
```

5.3.2 U-Boot logo 图片更换

替换 kernel/logo.bmp 文件即可替换 uboot logo。

6 Android 开发常见配置

本节简单介绍 Android 开发中一些常见配置的修改，后续会不断推送版本更新至开发者手中。希望开发者把开发过程中遇到的问题及需要改进的功能点反馈给我们，一同完善 SDK。

6.1 Android 编译配置

6.1.1 lunch 选项说明

rkxxxx_box-userdebug：RKxxxx 平台 box 产品 userdebug（64 位）

rkxxxx_box-user：RKxxxx 平台 box 产品 user（64 位）

user 版本开启 ODEX 预编译，编译出来的固件偏大，但会提高开机速度。但开发过程中涉及到 APK 及 jar 的更新调试相对麻烦很多。

建议开发调试阶段默认选择 userdebug 编译。

6.1.2 添加一个新的产品

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

当需要添加一个新的产品时，可以基于已有的产品来建立，如下以建立一个新的产品为例进行说明，具体步骤为：

1）产品命令规则：

Box 产品名中需带有“box”字样；

请务必遵守以上规则，否则系统会异常。

2）新增文件夹 device/rockchip/rk3368/rk3368_box_000，基于 rk3368_box.mk 创建 rk3368_box_000.mk，将 rk3368_box 目录下的所有文件拷贝至 rk3368_box_000 目录下。

```
cd device/rockchip/rk3368
mkdir rk3368_box_000
cp rk3368_box.mk ./rk3368_box_000.mk
cp rk3368_box/* rk3368_box_000/
```

3）在 device/rockchip/rk3368/AndroidProducts.mk 中添加：

```
PRODUCT_MAKEFILES := \
    $(LOCAL_DIR)/rk3368.mk \
    $(LOCAL_DIR)/rk3368_box.mk \
$(LOCAL_DIR)/rk3368_box_000.mk \
```

4）在 vendorsetup.sh 中添加产品对应的 lunch 选项：

```
add_lunch_combo rk3368_box-eng
add_lunch_combo rk3368_box-userdebug
add_lunch_combo rk3368_box-user
add_lunch_combo rk3368_box_000-userdebug
add_lunch_combo rk3368_box_000-user
```

5）修改 rk3368_ box_000.mk 及 rk3368_box_000 目录下的新产品所需要修改的配置。

6）修改编译脚本或编译命令，重新 lunch 产品名称进行新产品编译。

6.2 常用配置说明

表 6-1 常用配置说明

宏配置	功能说明

BUILD_BOX_WITH_GOOGLE_MARKET	若为 true 则集成 GMS 包，false 不集成
BUILD_WITH_WIDEVINE	集成 Widevine level3 插件库
TARGET_ROCKCHIP_PCBATEST	使能 PCBA 测试
BOOT_SHUTDOWN_ANIMATION_RINGING	使能开关机动画+铃声
BOARD_USB_ALLOW_DEFAULT_MTP	默认使能 MTP
BOARD_SYSTEMIMAGE_PARTITION_SIZE	System 分区容量，默认自动配置，亦可手动配置

6.3 显示框架配置

当前 SDK 的显示框架增加了一些系统属性用于帮助客户能够根据需求配置显示。

6.3.1 主副显示器配置

表 6-2 主副显示器配置

属性	功能说明
sys.hwc.device.primary	设置显示接口做为主显
sys.hwc.device.extend	设置显示接口做为副显

以上两个属性的配置可加在产品配置目录下的 system.prop 里（如 device/rockchip/rk3368/rk3368_box/system.prop）。

默认情况下(即以上属性未配置时)，不支持热拔插设备（如 cvbs/mipi/edp 等）会作为主显，支持热插拔设备（如 hdmi/dp 等）会作为次显。以下是当前 RK3368 BOX SDK 默认采用的配置：

```
sys.hwc.device.primary=HDMI-A,TV
```

主屏配置里面同时存在两个或以上的设备时，优先使用已接入带热拔插的设备，故以上配置当 hdmi 插入时，主显使用 hdmi 做为显示，hdmi 拔出时，主显使用 cvbs 做为显示。

另外，由于主显的 framebuffer 分辨率无法动态更改，所以有两个或以上设备做为主显时，最好设定一个主显的 framebuffer 分辨率。

关于接口名称可以参见 hardware/rockchip/hwcomposer/drmresources.cpp 里的定义：

```
struct type_name connector_type_names[] = {
    { DRM_MODE_CONNECTOR_Unknown, "unknown" },
    { DRM_MODE_CONNECTOR_VGA, "VGA" },
    { DRM_MODE_CONNECTOR_DVII, "DVI-I" },
    { DRM_MODE_CONNECTOR_DVID, "DVI-D" },
    { DRM_MODE_CONNECTOR_DVIA, "DVI-A" },
    { DRM_MODE_CONNECTOR_Composite, "composite" },
    { DRM_MODE_CONNECTOR_SVIDEO, "s-video" },
    { DRM_MODE_CONNECTOR_LVDS, "LVDS" },
    { DRM_MODE_CONNECTOR_Component, "component" },
    { DRM_MODE_CONNECTOR_9PinDIN, "9-pin DIN" },
    { DRM_MODE_CONNECTOR_DisplayPort, "DP" },
    { DRM_MODE_CONNECTOR_HDMIA, "HDMI-A" },
    { DRM_MODE_CONNECTOR_HDMIB, "HDMI-B" },
    { DRM_MODE_CONNECTOR_TV, "TV" },
    { DRM_MODE_CONNECTOR_eDP, "eDP" },
    { DRM_MODE_CONNECTOR_VIRTUAL, "Virtual" },
    { DRM_MODE_CONNECTOR_DSI, "DSI" },
};
```

6.3.2 主副显示器接口查询

可以通过以下两个只读属性来分别查询主副显示器的输出接口的名称。

表 6-3 主副显示器查询

属性	功能说明
sys.hwc.device.main	查询当前主显的输出接口
sys.hwc.device.aux	查询当前副显的输出接口

6.3.3 FrameBuffer 分辨率配置

可以通过配置以下属性来设置 FrameBuffer 的分辨率

```
persist.sys.framebuffer.main=1920x1080
```

6.3.4 分辨率过滤配置

因为初始获取到的全部分辨率过多，有些分辨率对用户来说并不需要，因此在 SDK 的 HWC 模块中对分辨率进行了过滤。

位于 device/rockchip/common/resolution_white.xml 路径的配置文件定义了能够通过过滤的白名单，HWC 中会根据该配置文件对初始的分辨率进行过滤筛选后再传递给上层，该 XML 文件的每一个<resolution>块定义了一个能够通过过滤的分辨率，其中详细项的定义如下：

表 6-4 分辨率过滤项定义说明

项定义	说明
clock	时钟
hdisplay	见图 6-1 的标示
hsync_start	见图 6-1 的标示
hsync_end	见图 6-1 的标示
htotal	见图 6-1 的标示
hskew	见图 6-1 的标示
vdisplay	见图 6-1 的标示
vsync_start	见图 6-1 的标示
vsync_end	见图 6-1 的标示
vtotal	见图 6-1 的标示
vscan	见图 6-1 的标示
vrefresh	显示设备帧率
flags	flags 的定义如下： DRM_MODE_FLAG_PHSYNC (1<<0) DRM_MODE_FLAG_NHSYNC (1<<1) DRM_MODE_FLAG_PVSYNC (1<<2) DRM_MODE_FLAG_NVSYNC (1<<3) DRM_MODE_FLAG_INTERLACE (1<<4)
vic	HDMI 标准对应定义的 VIC 值，如 HDMI 标准中未定义置 0

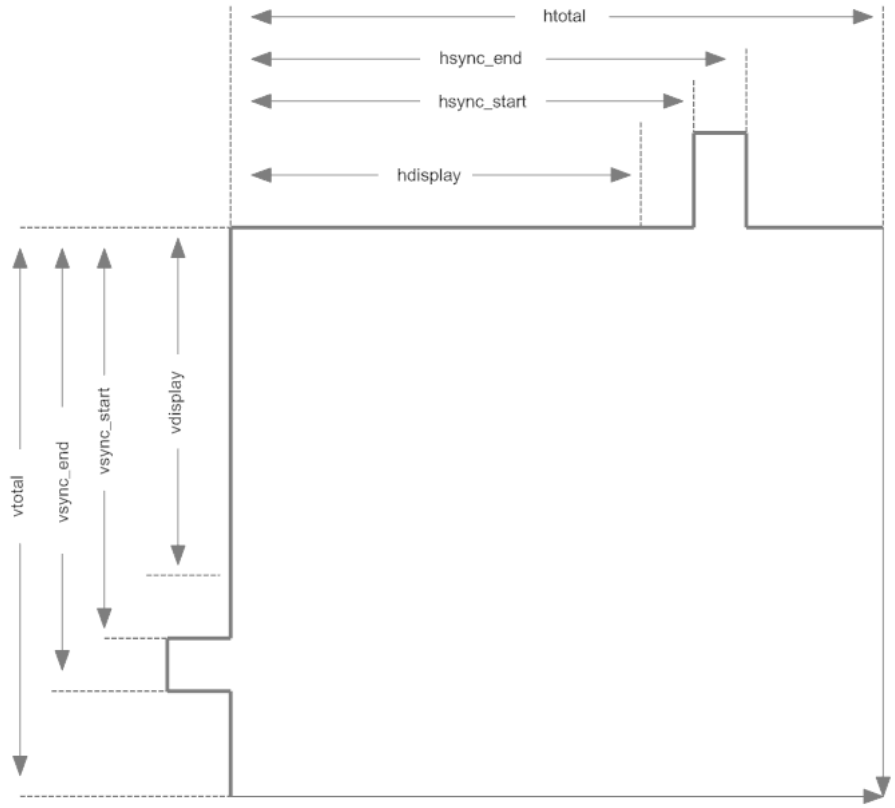

图 6-1 分辨率项定义示意图

6.4 预置 APK

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

预安装的 APK 应用需要得到对应厂商授权，若因为开发者及客户厂商私自预安装未授权应用进而需要承担法律责任的，RK 概不负责。

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

1）若是希望可卸载预安装，新增文件夹 device/rockchip/rk3368/rk3368_box/preinstall_del；若是不可卸载原装，新增文件夹 device/rockchip/rk3368/rk3368_box/preinstall。

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

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

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

6.5 开/关机动画及铃声

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

将开机铃声 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)

6.6 Parameter 说明

请参考 device/rockchip/rk3368/目录下 parameter.txt 文件来相应修改配置，关于 parameter 中各个参数、分区情况细节，请参考 RKDocs\RKTools manuals 目录下的《Rockchip Parameter File Format Ver1.3.pdf》文档。

由于 parameter 中定义了各个分区的大小，为保证 parameter 中的分区配置与系统分区配置关联匹配，请至少更新到 device/rockchip/common：c050ba78690062474bf15f3398d983322369e638。编译系统会从 TARGET_DEVICE_DIR（不同产品配置不同，可以通过 source build/envsetup.sh;get_build_var TARGET_DEVICE_DIR 查看。例如：device/rockchip/rk3368）读取 parameter 中的分区大小信息，并设置给 BOARD_SYSTEMIMAGE_PARTITION_SIZE。若客户需要自行配置，可在具体产品目录的 BoardConfig.mk 中覆盖自动配置的值。反之，若希望使用系统自动生成的分区配置，删除具体产品中的 BOARD_SYSTEMIMAGE_PARTITION_SIZE 配置。

6.7 新增分区配置

请参考 RKDocs\RKTools manuals 目录下的《Android 增加一个分区配置指南 V1.00.pdf》文档进行操作。

6.8 OTA 升级

6.8.1 OTA 介绍

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

详细的 OTA 升级及 Recovery 模块功能及配置，请参考 RKDocs\Develop reference documents 目录下《Rockchip Recovery 用户操作指南 V1.00》.

6.8.2 生成完整包

完整包所包含内容：system.img、recovery.img、boot.img

发布一个固件正确的顺序：

1、make -j4

2、make otapackage -j4

3、./mkimage.sh ota

发布固件必须使用./mkimage.sh ota,将 boot 与 kernel 打包，不需要单独烧 kernel，如果量产固件是分开的，将会影响后面差异包升级，除非你不需要用差异升级。

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

6.8.3 生成差异包

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

1、首先发布 v1 版本的固件，生成 v1 版本的完整包

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

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

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

5、生成 v1-v2 的差异升级包：

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

说明:生成差异包命令格式:

ota_from_target_files

-v -i 用于比较的前一个 target file

-p host 主机编译环境

-k 打包密钥

用于比较的后一个 target file

最后生成的 OTA 差异包

6.9 预制 Demo

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

6.10 系统灾难恢复

在系统使用过程中，为了避免由于升级异常、数据异常丢失等特殊原因出现的系统无法启动，RK3368 平台上加入了系统灾难恢复的机制，以在灾难出现的情况下，恢复系统启动。

如果一个 service 是 critical 的，而它又在短时间内反复挂掉，restart 后又总是运行失败，系统已经无法正常运行，于是系统需要启动灾难恢复操作。这样的 service 包括：

```
service ueventd /sbin/ueventd
    class core
critical
    seclabel u:r:ueventd:s0

service healthd /sbin/healthd
class core
 critical
    seclabel u:r:healthd:s0
    group root system wakelock
```

6.10.1 系统默认灾难恢复机制

目前 SDK 中，若 zygote 反复挂掉，我们即认为系统已经面临巨大灾难（无法正常开机），需要马上进行恢复，恢复手段是重启进入 recovery，并做 data 分区数据清除动作。客户可根据项目本身，及系统的运行的具体情况，指定自己的灾难恢复策略，提高系统软件的自愈能力。

6.10.2 灾难恢复定制开发

如果有一定相关开发经验的开发者，可以根据测试情况结合项目潜在可能遇到的灾难性问题，做出合理的灾难恢复处理，使得系统灾难，得以自愈及恢复，完善用户的体验并避免不必要的返厂维修。

critical 进程添加：

若需要让系统记录并捕获进程短时间内异常退出次数，需在 init.rc 中定义进程 critical 标记，init.rc 并不固定文件，开发者需自行判定并添加修改对应 init.rc。zygote 进程修改示例如下：

```
    service zygote /system/bin/app_process64 -Xzygote /system/bin --zygote
--start-system-server --socket-name=zygote
    # class main
        class core
     critical
        socket zygote stream 660 root system
        onrestart write /sys/android_power/request_state wake
        onrestart write /sys/power/state on
        onrestart restart audioserver
        onrestart restart cameraserver
        onrestart restart media
        onrestart restart netd
        writepid /dev/cpuset/foreground/tasks

    service zygote_secondary /system/bin/app_process32 -Xzygote /system/bin
--zygote --socket-name=zygote_secondary
        # class main
        class core
     critical
        socket zygote_secondary stream 660 root system
        onrestart restart zygote
        writepid /dev/cpuset/foreground/tasks
```

critical 进程异常捕获后操作定制：

当 critical 进程在短时间内重复挂掉，进程异常退出的次数将会通过 signal 通知 init 进程，init 进程会对异常退出次数进行记录。

system/core/init/service.cpp 文件中对灾难恢复机制触发的次数进行了定义：

```
    #define CRITICAL_CRASH_THRESHOLD      4
```

灾难恢复功能定制也在 system/core/init/service.cpp 文件中实现，目前实现如下，若达到指定挂掉次数，会进入 recovery，并格式化 data：

```
        if ((flags_ & SVC_CRITICAL) && !(flags_ & SVC_RESTART)) {
            if (time_crashed_ + CRITICAL_CRASH_WINDOW >= now) {
                if (++nr_crashed_ > CRITICAL_CRASH_THRESHOLD) {
                    ERROR("critical process '%s' exited %d times in %d minutes; "
    "rebooting into recovery mode\n", name_.c_str(),
                          CRITICAL_CRASH_THRESHOLD,
CRITICAL_CRASH_WINDOW / 60);
    #if defined(TARGET_BOARD_PLATFORM_PRODUCT_BOX)
     FILE *fd = fopen("/cache/recovery/command", "wb+");
                    if(NULL == fd){
                        NOTICE("*******/cache/recovery/command can't
open*******\n");
                    }
                    char buffer[100] = "--wipe_data\n";
```

```
                    fwrite(buffer, 1, strlen("--wipe_data"), fd);
                    fclose(fd);
                    fd = NULL;
                    sync()
#endif
android_reboot(ANDROID_RB_RESTART2, 0, "recovery");
                    return false;
                }
            } else {
                time_crashed_ = now;
                nr_crashed_ = 1;
            }
        }
```

灾难恢复推荐操作如下：

1、系统重启

2、重启进入 recovery，进入菜单模式

3、重启进入 recovery，格式化指定分区

4、删除可能被破坏的数据，重启

6.11 drmservice 的 demo 拷贝功能

- 功能描述及典型应用场景

将 ro.boot.copy_source 中配置的路径文件拷贝到 ro.boot.copy_dest 目录下

可将媒体资源文件，集成在固件中，drmservice 将这些资源文件拷贝到用户可见的 /data/media/0 目录下，达到预制媒体资源的目的。

- 使用方法

工程中准备好预制素材文件，并配置

ro.boot.copy_source

ro.boot.copy_dest

- 相关代码位置

system/core/drmservice/drmservice.c

7 系统调试

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

7.1 ADB 工具

7.1.1 概述

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

- 运行设备的 shell（命令行）
- 管理模拟器或设备的端口映射
- 计算机和设备之间上传/下载文件
- 将本地 apk 软件安装至模拟器或 Android 设备

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

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

7.1.2 USB ADB 使用说明

USB ADB 使用有以下限制：

- 只支持 USB OTG 口
- 不支持多个客户端同时使用（如 cmd 窗口，eclipse 等）
- 只支持主机连接一个设备，不支持连接多个设备

连接步骤如下：

1、Box 机器已经运行 Android 系统， 设置->开发者选项->已连接到计算机 打开，usb 调试开关打开。

2、PC 主机只通过 USB 线连接到机器 USB OTG 口，然后电脑通过如下命令与 Box 机器相连。

```
adb shell
```

3、测试是否连接成功，运行”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/rk3328/rk3328_box/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 7-1 跟踪进程内存状态

7.4 Dumpsys 工具

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

dumpsys 输出打印的条件：

1、只能打印已经加载到 ServiceManager 中的服务；

2、如果服务端代码中的 dump 函数没有被实现，则没有信息输出。

7.4.1 使用 Dumpsys

- 查看 Dumpsys 帮助

作用：输出 dumpsys 帮助信息。

```
dumpsys  -help
```

- 查看 Dumpsys 包含服务列表

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

```
dumpsys  -l
```

- 输出指定服务的信息

作用：输出指定的服务的 dump 信息。

格式：dumpsys [servicename]

示例：输出服务 SurfaceFlinger 的信息，可执行命令：

```
dumpsys SurfaceFlinger
```

- 输出指定服务和应有进程的信息

作用：输出指定服务指定应用进程信息。

格式：dumpsys [servicename] [应用名]

示例：输出服务名为 meminfo，进程名为 com.android.systemui 的内存信息，执行命令：

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

注意：服务名称是大小写敏感的，并且必须输入完整服务名称。

7.5 串口调试

7.5.1 串口配置

调试过程中最方便的就是串口的输入输出，这里需要注意的 RK3368 波特率设置为 1.5M。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 音视频问题调试工具及文档

在 RKDocs\Develop reference documents 目录下《Rockchip Videodebug_V1.0_20170109》为音视频问题调试工具包，其中也包含了详细的调试说明文档《Rockchip Audio 开发指南 V1.0-20160606》可帮助开发者对音视频问题做出基本的排查和分析。

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\ RKTools manuals 目录下工具文档。

8.1 StressTest

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

Stresstest 测试工具测试的内容主要包括：

模块相关

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

非模块相关

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

8.2 PCBA 测试工具

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

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

具体 PCBA 功能配置及使用说明，请参考

\RKDocs\RKTools manuals\Rockchip PCBA 模块开发指南--20170210.pdf

8.3 DDR 测试工具

设备上使用 DDR 测试工具，对待测设备的 DDR 进行稳定性测试，确保 DDR 功能正常及稳定。DDR 测试工具见 RKTools\windows\Rockchip 平台 DDR 测试工具_V1.35 发布通知.7z

8.4 Android 开发工具

8.4.1 下载镜像

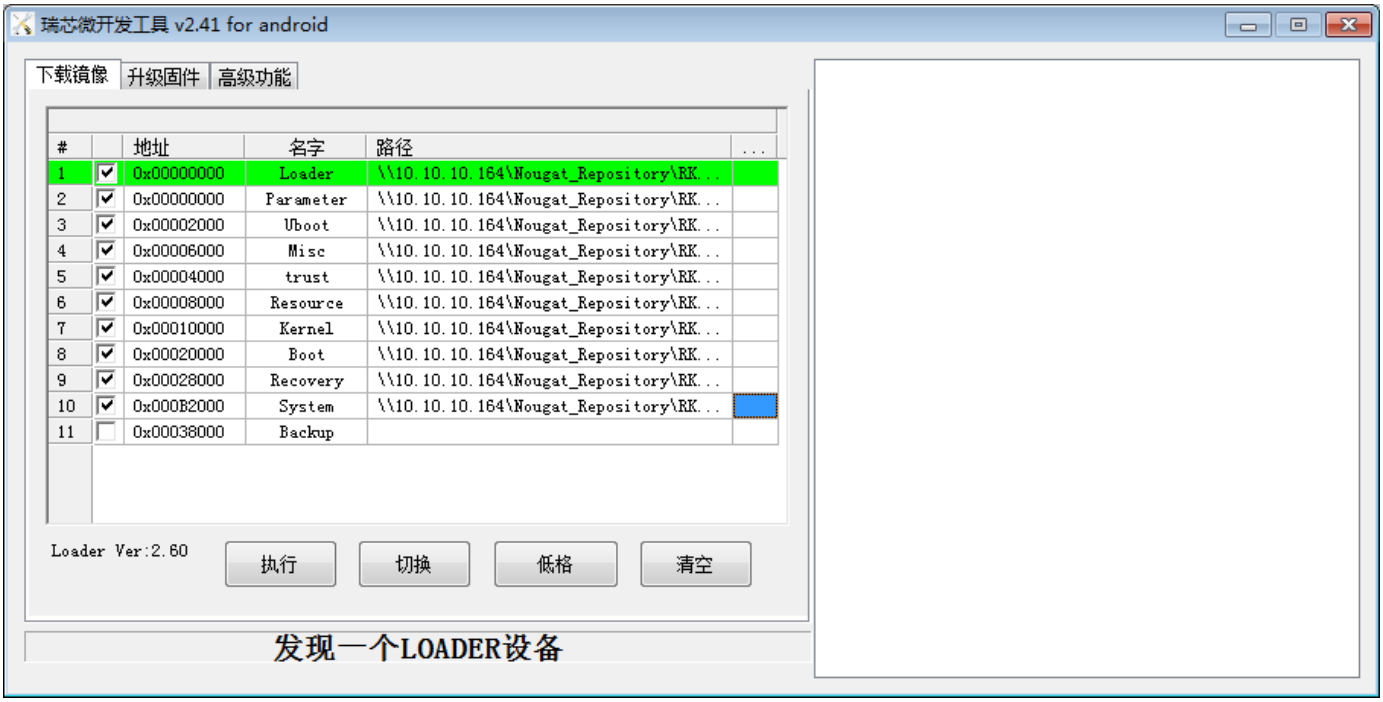

图 8-1 Android 开发工具下载镜像

1）连接开发板进入下载模式。

下载模式：先按住开发板 recovery 按键，再上电即可进入下载模式。

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

3）依次设置所有 img 文件的路径。

4）配置完成后，点击“执行”。右侧信息框将显示相关信息。

5）按钮说明

“低格”按钮：用于擦除设备

“清空”按钮：清空信息框

8.4.2 升级固件

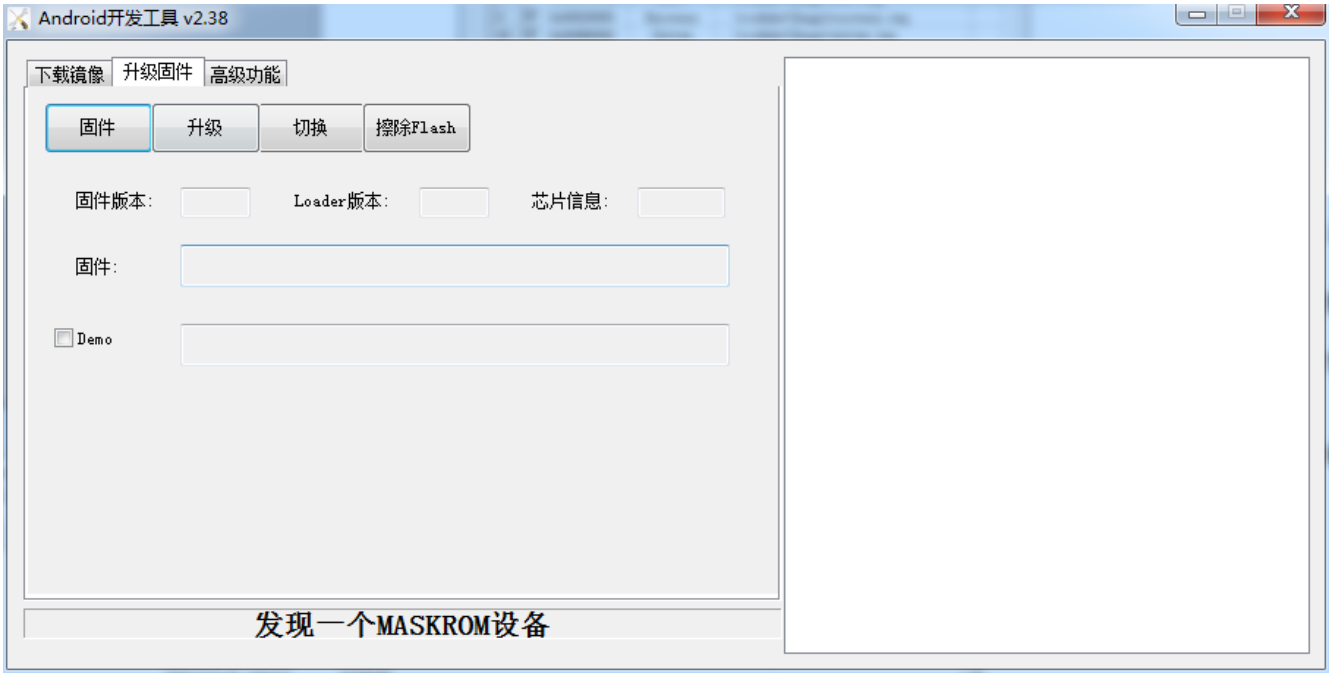

图 8-2 Android 开发工具升级固件

1）准备目标固件。（可参考 update.img 打包）

2）确认设备已经进入下载模式。

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

3）点击“固件”按钮，选择目标固件 update.img 文件。

4）点击“升级”按钮进行下载。右侧信息框将显示相关信息。

8.4.3 高级功能

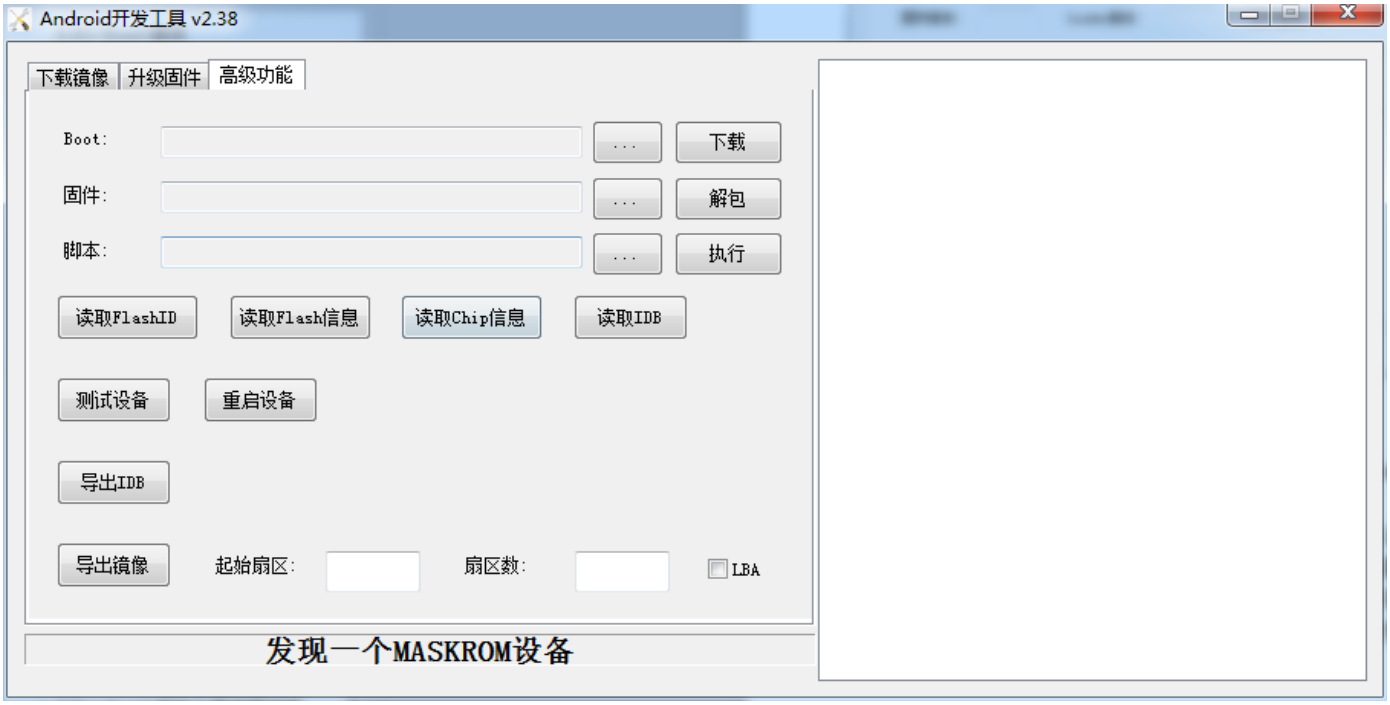

图 8-3 Android 开发工具高级功能

高级功能说明：

1）Boot 只能选择打包好的 update.img 文件或是 loader 文件。

2）固件必须现使用打包后的 update.img。

3）解包功能可将 update.img 拆解为各部分镜像文件。

8.5 update.img 打包

RK3368 平台支持将各零散镜像文件，打包成一个完成的 update.img 形式，方便量产烧写及升级。具体打包步骤如下：

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

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。

```
1  Afptool -pack .\backupimage backupimage\backup.img
2  Afptool -pack ./ Image\update.img
3
4
5  RKImageMaker.exe -RK322H RK3328MiniLoaderAll.bin  Image\update.img update.img -os_type:androidos
6
7  rem update.img is new format, Image\update.img is old format, so delete older format
8  del  Image\update.img
9
10 pause
```

图 8-4 update.img 打包脚本

4）修改 loader 名称为实际存放的 loader 名称。

5）点击 mkupdate.bat 运行，结束后会在该目录生成一个 update.img。

8.6 固件签名工具

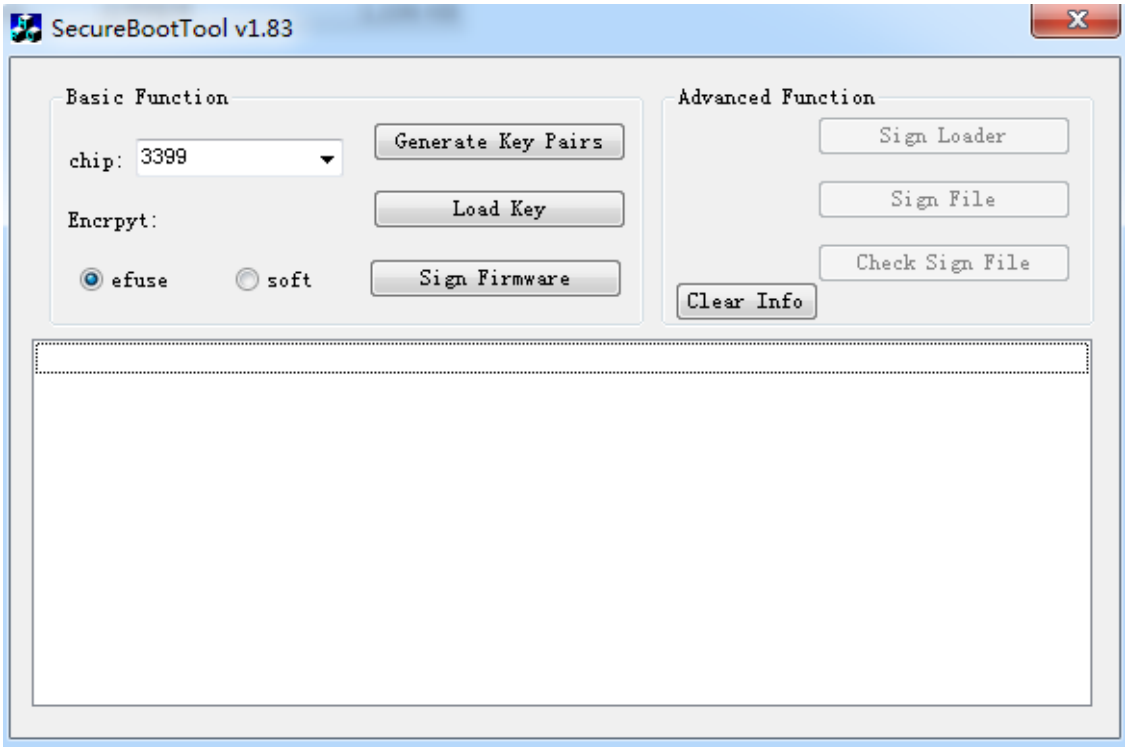

图 8-5 固件签名工具

1）选择 chip 类型和加密类型，如果是 RK3368 则选择 eFuse。

2）点击“Generate Key Pairs”按钮生成公私钥对，点击保存。

3）点击“Load Key”按钮加载密钥，会连续弹出两次“选择密钥文件”界面。

第一次为选择私钥文件，第二次为公钥选择文件。

4）点击”Sign Firmware”按钮，签名 update.img 文件。

5）隐藏功能开/关：键盘按下 Ctrl+Alt+R+K 键，可以启用/关闭“Advanced Function”中的功能

8.7 序列号/Mac/厂商信息烧写-WNpctool 工具

在 RK 平台上，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。

8.7.1 使用 WNpctool 写入

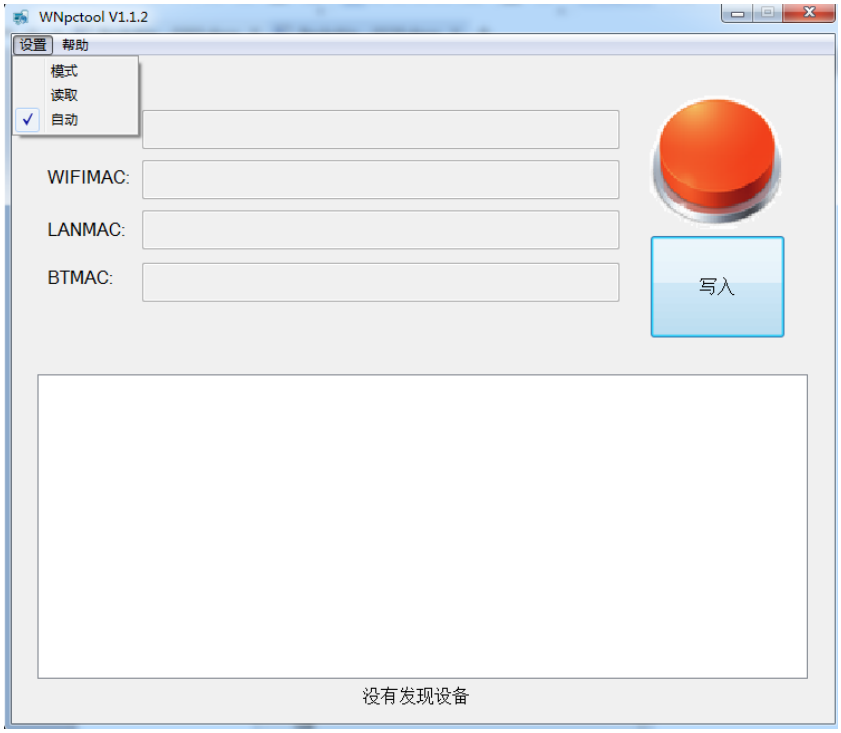

图 8-6WNpctool 工具

1）进入 loader 模式。

2）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）

3）点击“设置”菜单，点击“模式”，弹出“模式“窗口，用来设置 SN/WIFI/LAN/BT

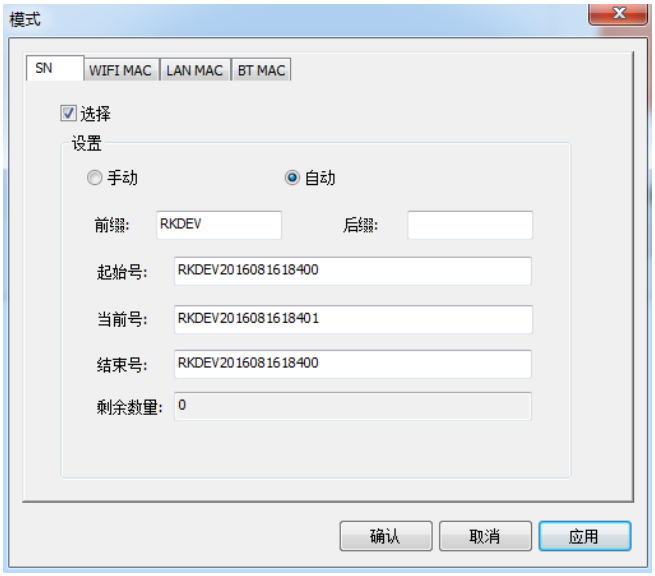

图 8-7WNpctool 工具模式设置

4）设置完成后，点击“应用”按钮，关闭模式设置窗口，返回主窗口。

5）点击“写入”按钮即可。

8.7.2 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。

8.8 OemTool 打包工具

8.8.1 Oem 打包工具步骤

RK3368 只支持 Ext4 镜像格式，故镜像格式选择 Ext4。

下载分区默认 UserData 分区，可直接不填写。

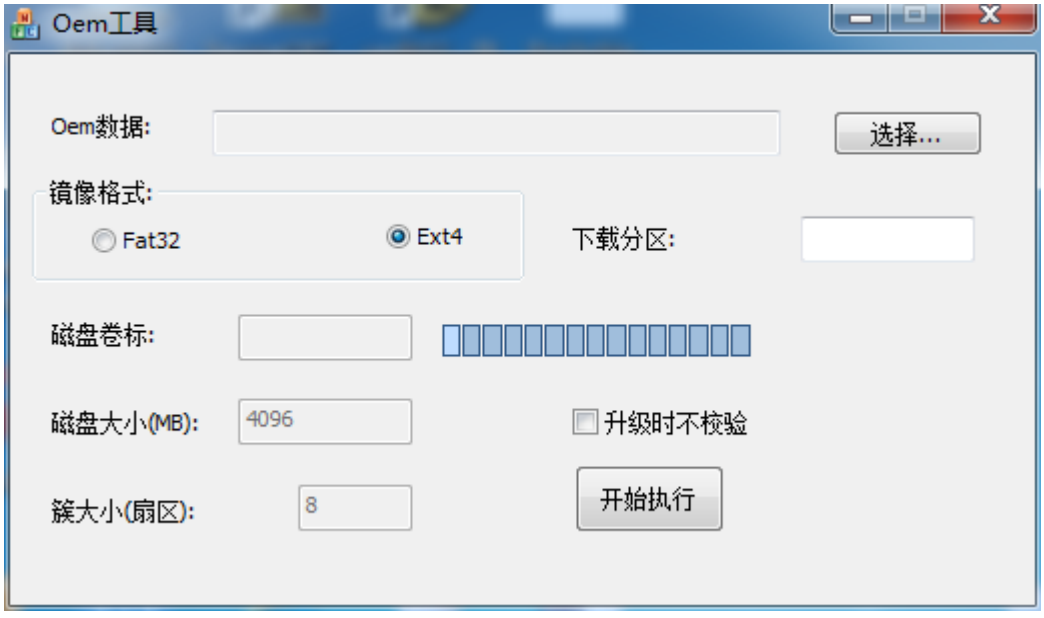

图 8-8 Oem 工具

点击选择按钮选择要打包的数据，数据必须是目录。目录最外围默认为 data 目录，假设你目录

为/data/media/0,且 0 有一个文件为 sss.txt(如下图示)。则当你升级完 demo 镜像的时候，会在系统的 data 目录下有目录 media/0,且在 0 目录下有文件 sss.txt 存在。

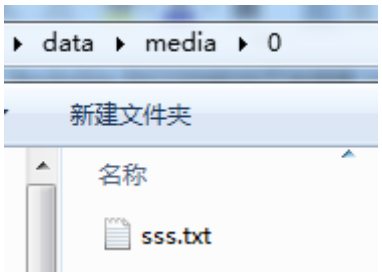

图 8-9 Oem 工具镜像制作文件夹路径要求

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

8.9 量产工具使用

8.9.1 工具下载步骤

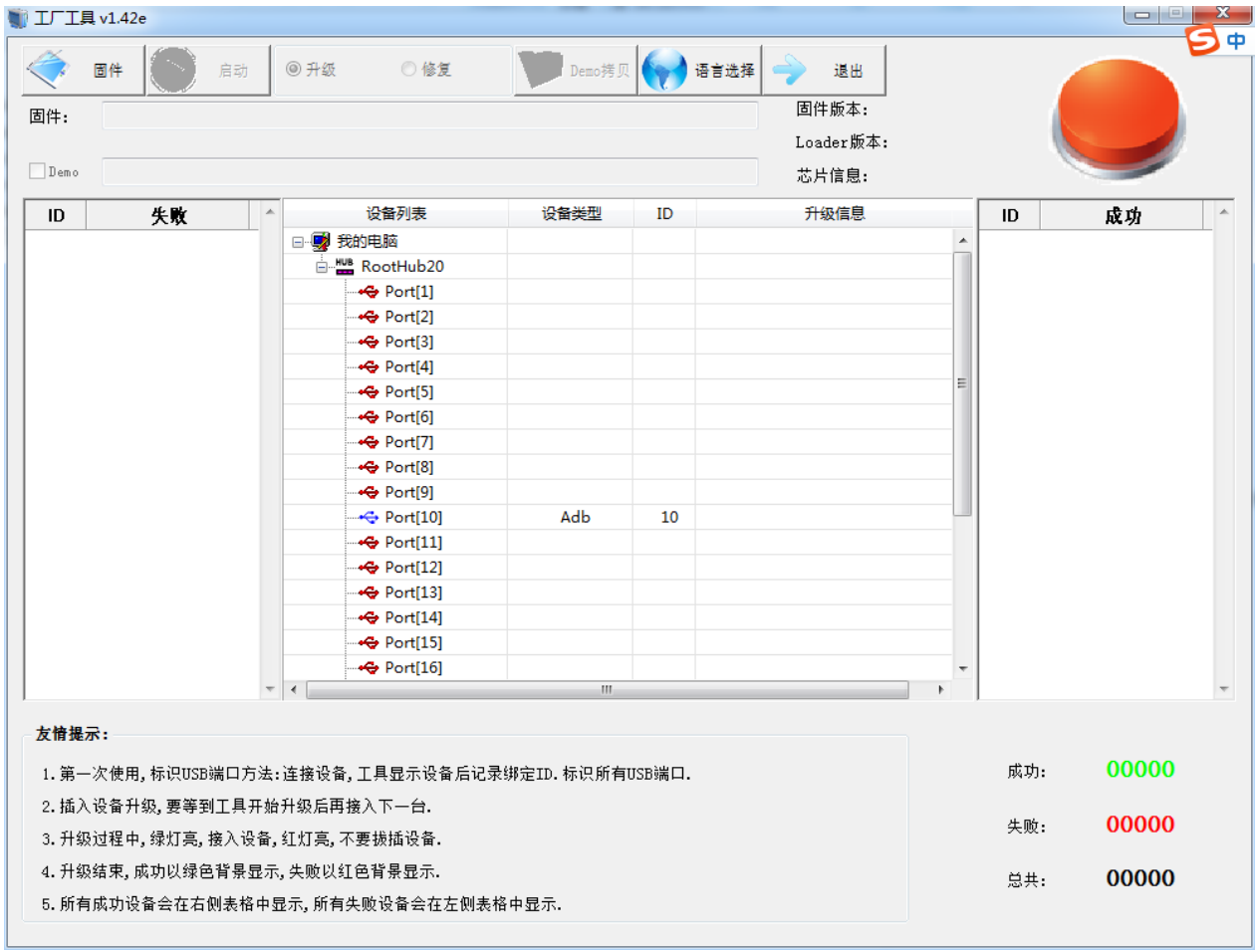

图 8-10 量产工具

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。