Rockchip RK3308 Linux5.10 SDK 快速入门

文档标识：RK-JC-YF-963

发布版本：V1.0.0

日期：2022-09-20

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

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： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

本文主要描述了RK3308 Linux5.10 SDK的基本使用方法，旨在帮助开发者快速了解并使用RK3308 Linux5.10 SDK开发包。

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

各芯片系统支持状态

芯片名称	Uboot版本	Kernel版本	Buildroot版本
RK3308B/RK3308H/RK3308B-S/RK3308H-S	2017.9	5.10	2021.11

修订记录

日期	版本	作者	修改说明
2022-09-20	V1.0.0	LinJianHua	初始版本

目录

Rockchip RK3308 Linux5.10 SDK 快速入门

1. 开发环境搭建
2. 软件开发指南
   - 2.1 开发向导
   - 2.2 软件更新记录
3. SDK 配置框架说明
   - 3.1 SDK 工程目录介绍
   - 3.2 SDK板级配置
   - 3.3 查看编译命令
   - 3.4 自动编译
   - 3.5 各模块编译及打包
     - 3.5.1 U-Boot编译
     - 3.5.2 Kernel编译
     - 3.5.3 Recovery编译
     - 3.5.4 Buildroot 编译
     - 3.5.5 交叉编译
       - 3.5.5.1 SDK目录内置交叉编译
       - 3.5.5.2 Buildroot内置交叉编译
     - 3.5.6 固件的打包
4. 刷机说明
   - 4.1 Windows 刷机说明
   - 4.2 Linux 刷机说明
   - 4.3 系统分区说明

1. 开发环境搭建

我们推荐使用 Ubuntu 20.04 的系统进行编译。其他的 Linux 版本可能需要对软件包做相应调整。除了系统要求外，还有其他软硬件方面的要求。
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。
软件要求：Ubuntu 20.04 系统：
编译 SDK 环境搭建所依赖的软件包安装命令如下：

```
sudo apt-get install git ssh make gcc libssl-dev liblz4-tool expect \
     g++ patchelf chrpath gawk texinfo chrpath diffstat binfmt-support \
     qemu-user-static live-build bison flex fakeroot cmake gcc-multilib \
     g++-multilib unzip device-tree-compiler ncurses-dev libgucharmap-2-90-dev \
     bzip2 expat gpgv2 cpp-aarch64-linux-gnu g++-aarch64-linux-gnu
```

建议使用 Ubuntu20.04 系统或更高版本开发，若编译遇到报错，可以视报错信息，安装对应的软件包。

2. 软件开发指南

2.1 开发向导

为帮助开发工程师更快上手熟悉 SDK 的开发调试工作，随 SDK 发布《Rockchip_Developer_Guide_Linux_Software_CN.pdf》，可在docs下获取，并会不断完善更新。

2.2 软件更新记录

软件发布版本升级通过工程 xml 进行查看，具体方法如下：

```
.repo/manifests$ realpath rk3308_linux5.10_release.xml
# 例如:打印的版本号为v1.0.0，更新时间为20220920
#<SDK>/.repo/manifests/rk3308_linux/rk3308_linux5.10_release_v1.0.0_20220920.xml
```

3. SDK 配置框架说明

3.1 SDK 工程目录介绍

SDK目录包含有 buildroot、recovery、app、kernel、u-boot、device、docs、external 等目录。每个目录或其子目录会对应一个 git 工程，提交需要在各自的目录下进行。

- app：存放上层应用 APP，主要是 qcamera/qfm/qplayer/qseting 等一些应用程序。
- buildroot：基于 Buildroot（2021.11）开发的根文件系统。

- device/rockchip：存放各芯片板级配置以及一些编译和打包固件的脚本和预备文件。
- docs：存放开发指导文件、平台支持列表、工具使用文档、Linux 开发指南等。
- IMAGE：存放每次生成编译时间、XML、补丁和固件目录。
- external：存放第三方相关仓库，包括音频、视频、网络、recovery 等。
- kernel：存放 Kernel 5.10 开发的代码。
- prebuilts：存放交叉编译工具链。
- rkbin：存放 Rockchip 相关 Binary 和工具。
- rockdev：存放编译输出固件。
- tools：存放 Linux 和 Window 操作系统下常用工具。
- u-boot：存放基于 v2017.09 版本进行开发的 U-Boot 代码。

3.2 SDK板级配置

进入工程`<SDK>/device/rockchip/rk3308`目录：

板级配置	说明
BoardConfig_rk3308bs_64bit.mk	适用于 RK3308BS EVB V11\V20 开发板 运行64位系统
BoardConfig_rk3308bs_32bit.mk	适用于 RK3308BS EVB V11\V20 开发板 运行32位系统
BoardConfig_rk3308hs_32bit.mk	适用于 RK3308HS MODULE V10 开发板 运行32位系统
BoardConfig_rk3308b_64bit.mk	适用于 RK3308B EVB V10 开发板 运行64位系统
BoardConfig_rk3308b_32bit.mk	适用于 RK3308B EVB V10 开发板 运行32位系统
BoardConfig_rk3308h_32bit.mk	适用于 RK3308H MODULE V10 开发板 运行32位系统

方法1
`./build.sh` 后面加上板级配置文件, 例如：

选择**适用于RK3308BS EVB V11\V20 开发板** 运行64位系统的板级配置：

```
rk3308$./build.sh device/rockchip/rk3308/BoardConfig_rk3308bs_64bit.mk
```

方法2

```
rk3308$ ./build.sh lunch
processing option: lunch

You're building on Linux
Lunch menu...pick a combo:

0. default BoardConfig.mk
1. BoardConfig.mk
2. BoardConfig_32bit.mk
3. BoardConfig_rk3308b_32bit.mk
4. BoardConfig_rk3308b_64bit.mk
5. BoardConfig_rk3308bs_32bit.mk
6. BoardConfig_rk3308bs_64bit.mk
7. BoardConfig_rk3308h_32bit.mk
8. BoardConfig_rk3308hs_32bit.mk
9. BoardConfig_robot32.mk
```

```
10. BoardConfig_robot64.mk
11. BoardConfig_soundai_cmcc.mk
Which would you like? [0]: 6
```

3.3 查看编译命令

在根目录执行命令：./build.sh -h|help

```
rk3308$ ./build.sh -h
Usage: build.sh [OPTIONS]
Available options:
BoardConfig*.mk    -switch to specified board config
lunch              -list current SDK boards and switch to specified board config
wifibt             -build wifibt
uboot              -build uboot
uefi               -build uefi
spl                -build spl
loader             -build loader
kernel             -build kernel
modules            -build kernel modules
toolchain          -build toolchain
rootfs             -build default rootfs, currently build buildroot as default
buildroot          -build buildroot rootfs
ramboot            -build ramboot image
multi-npu_boot     -build boot image for multi-npu board
yocto              -build yocto 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
otapackage         -pack ab update otapackage image (update_ota.img)
sdpackage          -pack update sdcard package image (update_sdcard.img)
save               -save images, patches, commands used to debug
allsave            -build all & firmware & updateimg & save
check              -check the environment of building
info               -see the current board building information
app/<pkg>          -build packages in the dir of app/*
external/<pkg>     -build packages in the dir of external/*

createkeys         -create secureboot root keys
security_rootfs    -build rootfs and some relevant images with security paramter
(just for dm-v)
security_boot      -build boot with security paramter
security_uboot     -build uboot with security paramter
security_recovery  -build recovery with security paramter
security_check     -check security paramter if it's good

Default option is 'allsave'.
```

查看部分模块详细编译命令，例如：./build.sh -h kernel

```
rk3308$ ./build.sh -h kernel
###Current SDK Default [ kernel ] Build Command###
cd kernel
make ARCH=arm64 rk3308_linux_defconfig
make ARCH=arm64 rk3308bs-evb-amic-v11.img -j12
```

详细的编译命令以实际对应的SDK版本为准，主要是配置可能会有差异。build.sh编译命令是固定的。

3.4 自动编译

进入工程根目录执行以下命令自动完成所有的编译：

```
./build.sh all # 只编译模块代码（U-Boot, Kernel, Rootfs, Recovery）
               # 需要再执行./mkfirmware.sh 进行固件打包

./build.sh     # 在./build.sh all基础上
               # 1. 增加固件打包 ./mkfirmware.sh
               # 2. update.img打包
               # 3. 复制rockdev目录下的固件到IMAGE/***_RELEASE_TEST/IMAGES目录
               # 4. 保存各个模块的补丁到IMAGE/***_RELEASE_TEST/PATCHES目录
               # 注: ./build.sh 和 ./build.sh allsave 命令一样
```

3.5 各模块编译及打包

3.5.1 U-Boot编译

```
### U-Boot编译命令
./build.sh uboot

### 查看U-Boot详细编译命令
./build.sh -h uboot
```

3.5.2 Kernel编译

```
### Kernel编译命令
./build.sh kernel

### 查看Kernel详细编译命令
./build.sh -h kernel
```

3.5.3 Recovery编译

```
### Recovery编译命令
./build.sh recovery

### 查看Recovery详细编译命令
./build.sh -h recovery
```

注：Recovery是非必需的功能，有些板级配置不会设置

3.5.4 Buildroot 编译

进入工程目录根目录执行以下命令自动完成 Rootfs 的编译及打包：

```
./build.sh rootfs
```

编译后在 Buildroot 目录 output/rockchip_rk3308_bs_release/images下生成rootfs.squashfs。

3.5.5 交叉编译

3.5.5.1 SDK目录内置交叉编译

SDK prebuilts目录预置交叉编译，如下：

目录	说明
prebuilts/gcc/linux-x86/aarch64/gcc-arm-10.3-2021.07-x86_64-aarch64-none-linux-gnu	gcc arm 10.3.1 64位工具链
prebuilts/gcc/linux-x86/arm/gcc-arm-10.3-2021.07-x86_64-arm-none-linux-gnueabihf	gcc arm 10.3.1 32位工具链

3.5.5.2 Buildroot内置交叉编译

若需要编译单个模块或者第三方应用，需对交叉编译环境进行配置。比如RK3308,其交叉编译工具位于`buildroot/output/rockchip_rk3308_bs_release/host/usr`目录下，需要将工具的bin/目录和`aarch64-buildroot-linux-gnu/bin/`目录设为环境变量，在顶层目录执行自动配置环境变量的脚本：

```
source envsetup.sh
```

输入命令查看：

```
cd buildroot/output/rockchip_rk3308_bs_release/host/usr/bin
./aarch64-linux-gcc --version
```

此时会打印如下信息：

```
aarch64-linux-gcc.br_real (Buildroot -g900f5662) 11.3.0
```

比如 rkscript 模块，常用相关编译命令如下：

- 编译 rkscript

```
SDK$make rkscript
```

- 重编 rkscript

```
SDK$make rkscript-rebuild
```

- 删除 rkscript

```
SDK$make rkscript-dirclean
或者
SDK$rm -rf buildroot/output/rockchip_rk3308_bs_release/build/rkscript
```

3.5.6 固件的打包

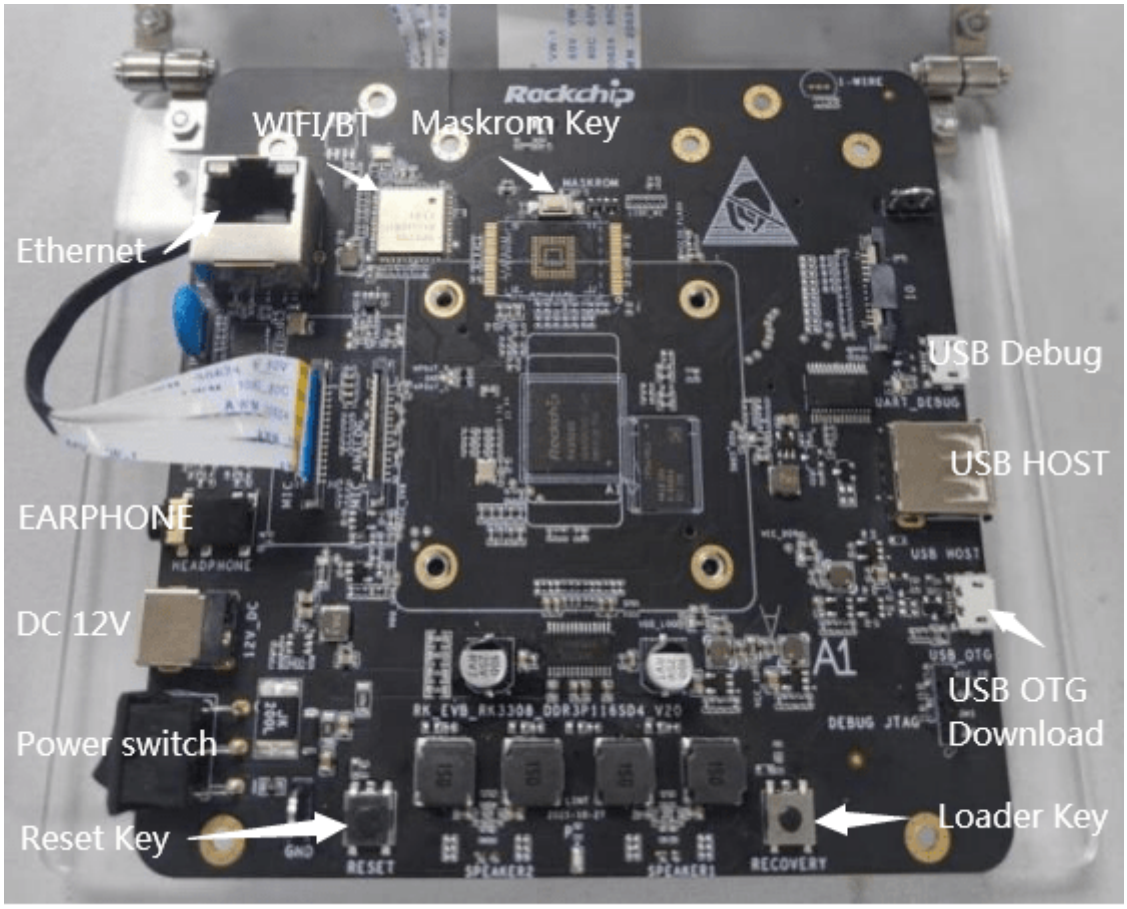
上面 Kernel/U-Boot/Recovery/Rootfs 各个部分的编译后，进入工程目录根目录执行以下命令自动完成所有固件打包到 rockdev 目录下:

固件生成:

```
./mkfirmware.sh
```

4. 刷机说明

RK3308B EVB V20 开发板正面接口分布图如下:

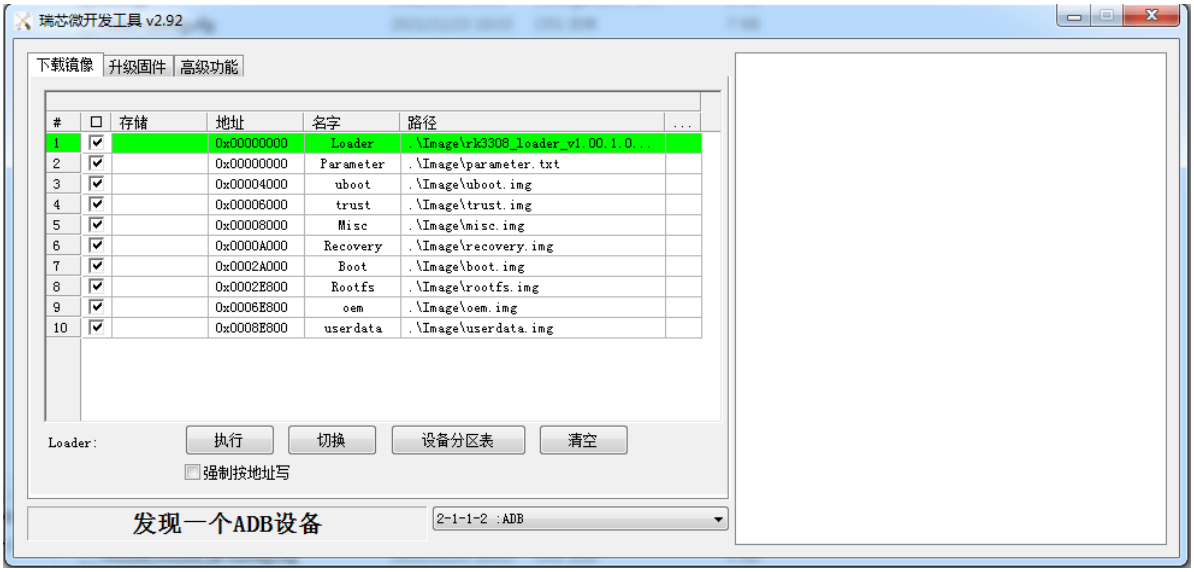


4.1 Windows 刷机说明

SDK 提供 Windows 烧写工具(工具版本需要 V2.91 或以上)，工具位于工程根目录：

```
tools/
├── windows/RKDevTool
```

如下图，编译生成相应的固件后，设备烧写需要进入 MASKROM 或 BootROM 烧写模式，连接好 USB 下载线后，按住按键“MASKROM”不放并按下复位键“RST”后松手，就能进入MASKROM 模式，加载编译生成固件的相应路径后，点击“执行”进行烧写，也可以按 “recovery"按键不放并按下复位键 “RST” 后松手进入 loader 模式进行烧写，下面是 MASKROM 模式的分区偏移及烧写文件。
(注意： Windows PC 需要在管理员权限运行工具才可执行)



注：烧写前，需安装最新 USB 驱动，驱动详见：

```
<SDK>/tools/windows/DriverAssitant_v5.11.zip
```

4.2 Linux 刷机说明

Linux 下的烧写工具位于 tools/linux 目录下(Linux_Upgrade_Tool 工具版本需要 V2.1 或以上)，请确认你的板子连接到 MASKROM/loader rockusb。比如编译生成的固件在 rockdev 目录下，升级命令如下：

```
sudo ./upgrade_tool ul -noreset rockdev/MiniLoaderAll.bin
sudo ./upgrade_tool di -p rockdev/parameter.txt
sudo ./upgrade_tool di -u rockdev/uboot.img
sudo ./upgrade_tool di -t rockdev/trust.img
sudo ./upgrade_tool di -misc rockdev/misc.img
sudo ./upgrade_tool di -b rockdev/boot.img
sudo ./upgrade_tool di -recovery rockdev/recovery.img
sudo ./upgrade_tool di -oem rockdev/oem.img
sudo ./upgrade_tool di -rootfs rocdev/rootfs.img
sudo ./upgrade_tool di -userdata rockdev/userdata.img
sudo ./upgrade_tool rd
```

或升级打包后的完整固件：

```
sudo ./upgrade_tool uf rockdev/update.img
```

或在根目录，机器在 MASKROM 状态运行如下升级：

```
./rkflash.sh
```

4.3 系统分区说明

默认分区说明 (下面是 RK3308 EVB 分区参考）

- uboot 分区：供 uboot 编译出来的 uboot.img。
- trust 分区：供 uboot 编译出来的 trust.img。
- misc 分区：供 misc.img，给 recovery 使用。
- boot 分区：供 kernel 编译出来的 boot.img。
- recovery 分区：供 recovery 编译出的 recovery.img。
- backup 分区：预留，暂时没有用，后续跟 Android 一样作为 recovery 的 backup 使用。
- rootfs 分区：供 buildroot、或 debian 编出来的 rootfs.img。
- oem 分区：给厂家使用，存放厂家的 APP 或数据。挂载在 /oem 目录。
- userdata 分区：供 APP 临时生成文件或给最终用户使用，挂载在 /userdata 目录下。