Rockchip RK3588 Linux SDK 快速入门

文档标识：RK-JC-YF-915

发布版本：V0.0.1

日期：2022-01-15

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RK3588 Linux SDK的基本使用方法，旨在帮助开发者快速了解并使用RK3588 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	**Debian**版本	**Buildroot**版本	**Yocto**版本
RK3588、RK3588S	2017.9	5.10	11	2018.02	3.2

修订记录

日期	版本	作者	修改说明
2022-01-15	V0.0.1	Caesar Wang	初始版本

目录

1. 开发环境搭建

我们推荐使用 Ubuntu 21.04 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 21.04 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 repo 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 \
```

建议使用 Ubuntu21.04 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 芯片资料

为帮助开发工程师更快上手熟悉 RK3588、RK3588S 的开发调试工作，随 SDK 发布
《Rockchip_RK3588_Datasheet_V1.0-20211221.pdf》和
《Rockchip_RK3588S_Datasheet_V1.0-20211221.pdf》芯片手册。

2.3 NPU 开发工具

本 SDK NPU 开发工具如下：

RKNN-TOOLKIT2 ：
开发工具在 `external/rknn-toolkit2` 目录下，主要用来实现模型转换，模型推理，模型性能评估功能等，具体使用说明请参考当前 doc/ 的目录文档：

```
├── RKNNToolKit2_OP_Support-1.2.0.md
├── Rockchip_Quick_Start_RKNN_Toolkit2_CN-1.2.0.pdf
├── Rockchip_Quick_Start_RKNN_Toolkit2_EN-1.2.0.pdf
├── Rockchip_User_Guide_RKNN_Toolkit2_CN-1.2.0.pdf
├── Rockchip_User_Guide_RKNN_Toolkit2_EN-1.2.0.pdf
├── changelog-1.2.0.txt
└── requirements-1.2.0.txt
```

RKNN API：
RKNN API的开发使用在工程目录 `external/rknpu2` 下，用于推理RKNN-Toolkit2生成的rknn模型。
具体使用说明请参考当前 doc/ 的目录文档：

```
└── Rockchip_RKNPU_User_Guide_RKNN_API_V1.2.0_CN.pdf
```

2.4 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 rk3588_linux_release.xml
# 例如:打印的版本号为v0.0.1，更新时间为20220115
# <SDK>/.repo/manifests/rk3588_linux_alpha_v0.0.1_20220115.xml
```

软件发布版本升级更新内容通过工程文本可以查看，具体方法如下：

```
.repo/manifests$ cat RK3588_Linux_SDK_Note.md
```

或者参考工程目录：

```
<SDK>/docs/RK3588/RK3588_Linux_SDK_Note.md
```

3. 硬件开发指南

硬件相关开发可以参考用户使用指南，在工程目录：

RK3588 硬件设计指南：

```
<SDK>/docs/RK3588/Rockchip_RK3588_Hardware_Design_Guide_V1.0.pdf
```

RK3588 EVB 硬件开发指南：

```
<SDK>/docs/RK3588/Rockchip_RK3588_EVB_User_Guide_V1.0_CN.pdf
```

4. SDK 配置框架说明

4.1 SDK 工程目录介绍

SDK目录包含有 buildroot、debian、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（2018.02-rc3）开发的根文件系统。
- debian：基于 Debian 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。
- yocto：存放基于 Yocto 3.2 开发的根文件系统。

4.2 SDK板级配置

进入工程`<SDK>/device/rockchip/rk3588`目录：

板级配置	说明
BoardConfig-rk3588-evb1-lp4-v10.mk	适用于 RK3588 EVB1 搭配 LPDDR4 开发板
BoardConfig-rk3588-evb3-lp5-v10.mk	适用于 RK3588 EVB3 搭配 LPDDR5 开发板
BoardConfig-rk3588s-evb1-lp4x-v10.mk	适用于 RK3588S EVB1 搭配 LPDDR4 开发板
BoardConfig-nvr.mk	适用于 RK3588 NVR 开发板
BoardConfig.mk	默认配置

方法1
`./build.sh` 后面加上板级配置文件, 例如：

选择**RRK3588 EVB1** 搭配 **LPDDR4** 开发板的板级配置：

```
./build.sh device/rockchip/rk3588/BoardConfig-rk3588-evb1-lp4-v10.mk
```

选择适用于 **RK3588 EVB3** 搭配 **LPDDR5** 开发板的板级配置：

```
./build.sh device/rockchip/rk3588/BoardConfig-rk3588-evb3-lp5-v10.mk
```

选择适用于 **RK3588S EVB1** 搭配 **LPDDR4** 开发板的板级配置：

```
./build.sh device/rockchip/rk3588/BoardConfig-rk3588s-evb1-lp4x-v10.mk
```

选择** 适用于 RK3588 NVR 开发板**的板级配置：

```
./build.sh device/rockchip/rk3588/BoardConfig-nvr.mk
```

方法2

```
rk3588$ ./build.sh lunch
processing option: lunch

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

0. default BoardConfig.mk
1. BoardConfig-nvr.mk
2. BoardConfig-rk3588-evb1-lp4-v10.mk
3. BoardConfig-rk3588-evb3-lp5-v10.mk
4. BoardConfig-rk3588s-evb1-lp4x-v10.mk
5. BoardConfig.mk
Which would you like? [0]:
```

4.3 查看编译命令

在根目录执行命令：./build.sh -h|help

```
rk3588$  ./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
uboot              -build uboot
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
distro             -build distro 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          -build packages in the dir of app/*
external      -build packages in the dir of external/*

Default option is 'allsave'.
```

查看部分模块详细编译命令，例如：./build.sh -h kernel

```
rk3588$ ./build.sh -h kernel
###Current SDK Default [ kernel ] Build Command###
cd kernel
make ARCH=arm64 rockchip_linux_defconfig
make ARCH=arm64 rk3588-evb1-lp4-v10-linux.img -j12
```

4.4 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
```

默认是 Buildroot，可以通过设置坏境变量 RK_ROOTFS_SYSTEM 指定 rootfs。RK_ROOTFS_SYSTEM目前可设定三个类型：buildroot、debian、 yocto 。

如需要 debain 可以通过以下命令进行生成：

```
$export RK_ROOTFS_SYSTEM=debian
$./build.sh
```

4.5 各模块编译及打包

4.5.1 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
```

4.5.2 Kernel编译

```
### Kernel编译命令
./build.sh kernel

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

4.5.3 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

4.5.4 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_rk3588/images下生成 rootfs.ext4。

4.5.5 Debian 编译

```
./build.sh debian
```

或进入 debian/ 目录：

```
cd debian/
```

后续的编译和 Debian 固件生成请参考当前目录 readme.md。

(1) Building base Debian system

```
sudo apt-get install binfmt-support qemu-user-static live-build
sudo dpkg -i ubuntu-build-service/packages/*
sudo apt-get install -f
```

编译 64 位的 Debian:

```
RELEASE=bullseye TARGET=desktop ARCH=arm64 ./mk-base-debian.sh
```

编译完成会在 debian/ 目录下生成：linaro-bullseye-alip-xxxxx-1.tar.gz（xxxxx 表示生成时间戳)。

FAQ:

- 上述编译如果遇到如下问题情况：

```
noexec or nodev issue /usr/share/debootstrap/functions: line 1450:
..../rootfs/ubuntu-build-service/bullseye-desktop-arm64/chroot/test-dev-null:
Permission denied E: Cannot install into target '/rootfs/ubuntu-build-
service/bullseye-desktop-arm64/chroot' mounted with noexec or nodev
```

解决方法：

```
mount -o remount,exec,dev xxx (xxx 是工程目录), 然后重新编译
```

另外如果还有遇到其他编译异常，先排除使用的编译系统是 ext2/ext4 的系统类型。

- 由于编译 Base Debian 需要访问国外网站，而国内网络访问国外网站时，经常出现下载失败的情况:

Debian 使用 live build,镜像源改为国内可以这样配置:

```
+++ b/ubuntu-build-service/buster-desktop-arm64/configure
@@ -11,6 +11,11 @@ set -e
 echo "I: create configuration"
 export LB_BOOTSTRAP_INCLUDE="apt-transport-https gnupg"
 lb config \
+ --mirror-bootstrap "https://mirrors.tuna.tsinghua.edu.cn/debian" \
+ --mirror-chroot "https://mirrors.tuna.tsinghua.edu.cn/debian" \
+ --mirror-chroot-security "https://mirrors.tuna.tsinghua.edu.cn/debian-security"
\
+ --mirror-binary "https://mirrors.tuna.tsinghua.edu.cn/debian" \
+ --mirror-binary-security "https://mirrors.tuna.tsinghua.edu.cn/debian-security"
  --apt-indices false \
  --apt-recommends false \
  --apt-secure false \
```

如果其他网络原因不能下载包，有预编生成的包分享在百度云网盘，放在当前目录直接执行下一步操作。

(2) Building rk-debian rootfs

编译 64位的 Debian：

```
VERSION=debug ARCH=arm64 ./mk-rootfs-bullseye.sh
```

(3) Creating the ext4 image(linaro-rootfs.img)

```
./mk-image.sh
```

此时会生成 linaro-rootfs.img。

4.5.6 Yocto 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：
RK3588/RK3588S EVB 开发板：

```
./build.sh yocto
```

编译后在 yocto 目录 build/lastest 下生成 rootfs.img。

FAQ：
上面编译如果遇到如下问题情况：

```
Please use a locale setting which supports UTF-8 (such as LANG=en_US.UTF-8).
Python can't change the filesystem locale after loading so we need a UTF-8
when Python starts or things won't work.
```

解决方法:

```
locale-gen en_US.UTF-8
export LANG=en_US.UTF-8 LANGUAGE=en_US.en LC_ALL=en_US.UTF-8
```

或者参考 setup-locale-python3 编译后生成的 image 在 yocto/build/lastest/rootfs.img， 默认用户名登录是 root。
Yocto 更多信息请参考 Rockchip Wiki。

4.5.7 交叉编译

4.5.7.1 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

4.5.7.2 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RK3588,其交叉编译工具位于`buildroot/output/rockchip_rk3588/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_rk3588/host/usr/bin
./aarch64-linux-gcc --version
```

此时会打印如下信息：

```
aarch64-linux-gcc.br_real (Buildroot 2018.02-rc3-gc6d04ba007) 10.3.0
```

比如 qplayer 模块，常用相关编译命令如下：

- 编译 qplayer

```
SDK$make qplayer
```

- 重编 qplayer

```
SDK$make qplayer-rebuild
```

- 删除 qplayer

```
SDK$make qplayer-dirclean
或者
SDK$rm -rf /buildroot/output/rockchip_rk3588/build/qlayer-1.0
```

4.5.8 固件的打包

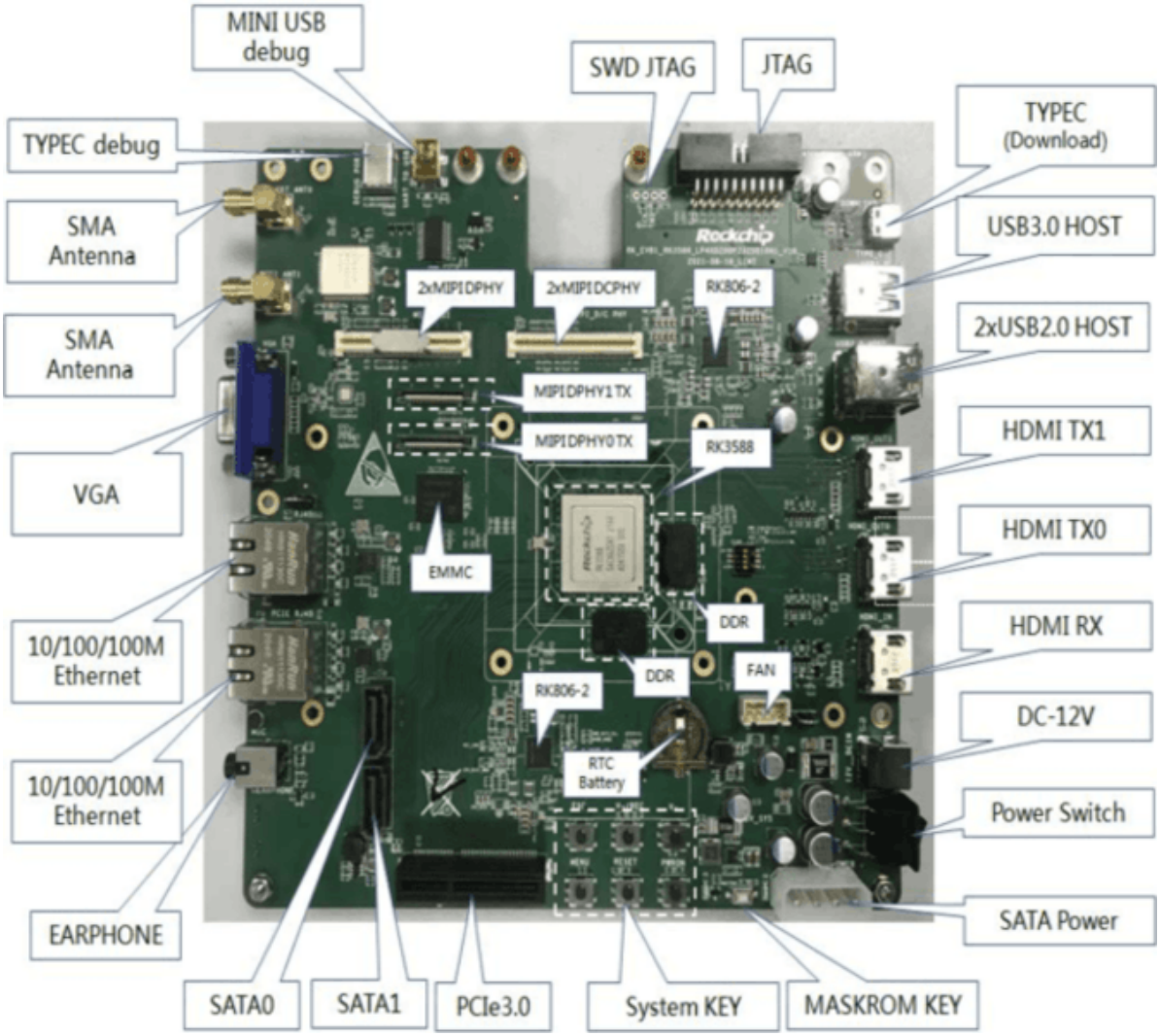
上面 Kernel/U-Boot/Recovery/Rootfs 各个部分的编译后，进入工程目录根目录执行以下命令自动完成所有固件打包到 rockdev 目录下：

固件生成:

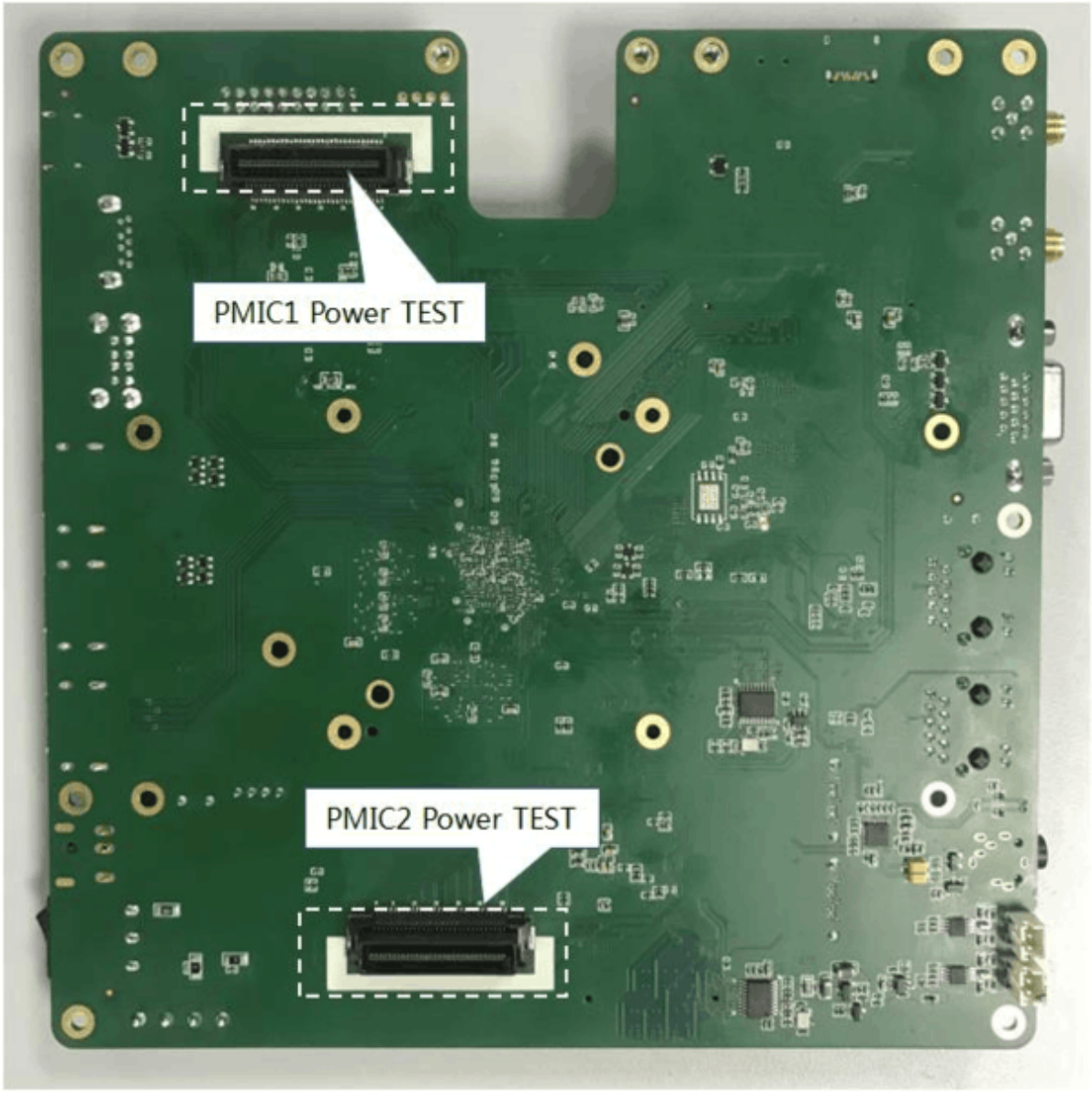
```
./mkfirmware.sh
```

5. 刷机说明

RK3588 EVB 开发板正面接口分布图如下：



RK3588 EVB1 开发板背面接口分布图如下：

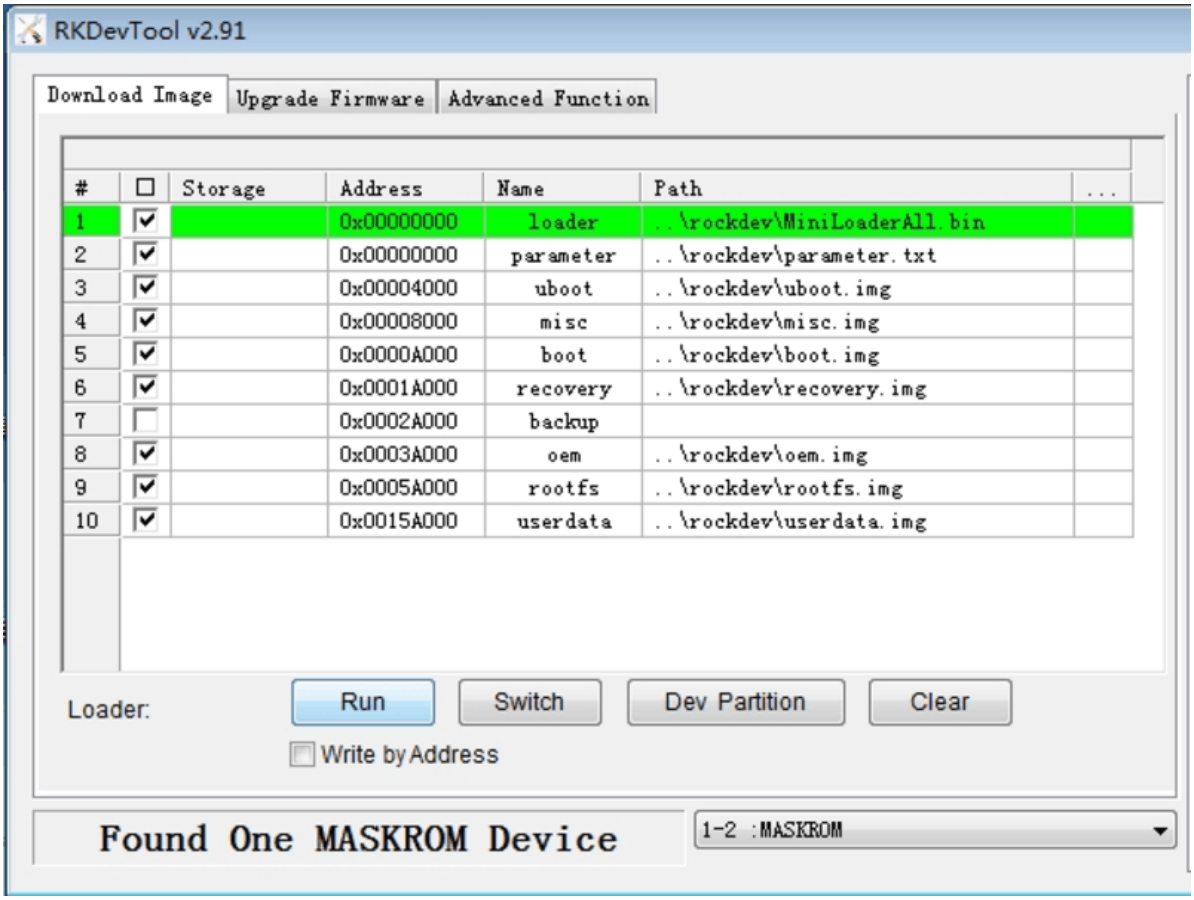


5.1 Windows 刷机说明

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
```

5.2 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 -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
```

5.3 系统分区说明

默认分区说明 (下面是 RK3588 EVB 分区参考）

Number	Start (sector)	End (sector)	Size	Name
1	8389kB	12.6MB	4194kB	uboot
2	12.6MB	16.8MB	4194kB	misc
3	16.8MB	83.9MB	67.1MB	boot
4	83.9MB	218MB	134MB	recovery
5	218MB	252MB	33.6MB	bakcup
6	252MB	15.3GB	15.0GB	rootfs
7	15.3GB	15.4GB	134MB	oem
8	15.6GB	31.3GB	15.6GB	userdata

- uboot 分区：供 uboot 编译出来的 uboot.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、debian 或 yocto 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。

6. RK3588 SDK 固件

- 百度云网盘

Buildroot

Debian rootfs

Yocto rootfs

- 微软 OneDriver

Buildroot

Debian rootfs

Yocto rootfs