

# Rockchip RK356X Linux SDK 快速入门

---

文档标识: RK-FB-YF-942

发布版本: V1.0.2

日期: 2022-11-20

文件密级: ☐绝密 ☐秘密 ☐内部资料 ☒公开

## 免责声明

本文档按“现状”提供, 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© 2022 瑞芯微电子股份有限公司

超越合理使用范畴, 非经本公司书面许可, 任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部, 并不得以任何形式传播。

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](http://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](mailto:fae@rock-chips.com)

## 前言

## 概述

本文主要描述了RK356X Linux SDK的基本使用方法，旨在帮助开发者快速了解并使用RK356X 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   | Buildroot | Debian | Yocto |
|--------|-----------|--------|-------|
| RK3566 | Y         | Y      | Y     |
| RK3568 | Y         | Y      | Y     |

## 修订记录

| 日期         | 版本     | 作者          | 修改说明           |
|------------|--------|-------------|----------------|
| 2022-06-20 | V1.0.0 | Caesar Wang | 初始版本。          |
| 2022-09-20 | V1.0.1 | Caesar Wang | 增加Linux5.10支持。 |
| 2022-11-20 | V1.0.2 | Caesar Wang | 更新刷机说明。        |

# 目录

## Rockchip RK356X Linux 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 芯片资料
  - 2.3 Debian开发指南
  - 2.4 第三方OS移植
  - 2.5 NPU 开发工具
  - 2.6 软件更新记录
3. 硬件开发指南
4. IO电源设计注意事项
5. SDK 工程目录介绍
6. SDK 编译说明
  - 6.1 SDK板级配置
  - 6.2 查看编译命令
  - 6.3 自动编译
  - 6.4 各模块编译及打包
    - 6.4.1 U-Boot编译
    - 6.4.2 Kernel编译
    - 6.4.3 Recovery编译
    - 6.4.4 Buildroot 编译
    - 6.4.5 Buildroot 的交叉编译
      - 6.4.5.1 Buildroot 中模块编译
    - 6.4.6 Debian 编译
    - 6.4.7 Yocto 编译
    - 6.4.8 固件的打包
7. 刷机说明
  - 7.1 Windows 刷机说明
  - 7.2 Linux 刷机说明
  - 7.3 系统分区说明
8. RK356X SDK 固件

# 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。

考虑客户开发环境搭建时间成本，我们也提供了交叉编译器docker镜像方式供客户验证，缩短编译环境搭建耗时。

参考文档 [Docker/Rockchip\\_Developer\\_Guide\\_Linux\\_Docker\\_Deploy\\_CN.pdf](#)。

**Docker**编译镜像系统兼容性测试结果参考如下：

| 发行版本         | Docker 版本 | 镜像加载 | 固件编译         |
|--------------|-----------|------|--------------|
| ubuntu 21.10 | 20.10.12  | pass | pass         |
| ubuntu 21.04 | 20.10.7   | pass | pass         |
| ubuntu 18.04 | 20.10.7   | pass | pass         |
| fedora35     | 20.10.12  | pass | NR (not run) |

# 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 RK3566、RK3568 的开发调试工作，随 SDK 发布

《Rockchip\_RK3566\_Datasheet\_V1.1\_20210305.pdf》和

《Rockchip\_RK3568\_Datasheet\_V1.1\_20210305.pdf》芯片手册。

## 2.3 Debian开发指南

为帮助开发工程师更快上手熟悉 RK356X Debian的开发调试，随 SDK 发布

《Rockchip\_Developer\_Guide\_Debian\_CN.pdf》开发指南，可在docs/ApplicationNote下获取，并会不断完善更新。

## 2.4 第三方OS移植

为帮助开发工程师更快上手熟悉 RK356X 第三方OS的移植适配，随 SDK 发布

《Rockchip\_Developer\_Guide\_Third\_Party\_System\_Adaptation\_CN.pdf》开发向导，可在 docs/ApplicationNote下获取，并会不断完善更新。

## 2.5 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_API_Difference_With_Toolkit1-1.4.0.md
|— RKNNToolKit2_OP_Support-1.4.0.md
|— Rockchip_Quick_Start_RKNN_Toolkit2_CN-1.4.0.pdf
|— Rockchip_Quick_Start_RKNN_Toolkit2_EN-1.4.0.pdf
|— Rockchip_User_Guide_RKNN_Toolkit2_CN-1.4.0.pdf
|— Rockchip_User_Guide_RKNN_Toolkit2_EN-1.4.0.pdf
|— changelog-1.4.0.txt
|— requirements_cp36-1.4.0.txt
|— requirements_cp38-1.4.0.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：

```
|— RKNN_Compiler_Support_Operator_List_v1.4.0.pdf
|— Rockchip_Quick_Start_RKNN_SDK_V1.4.0_CN.pdf
|— Rockchip_RKNPU_User_Guide_RKNN_API_V1.4.0_CN.pdf
|— Rockchip_RKNPU_User_Guide_RKNN_API_V1.4.0_EN.pdf
```

## 2.6 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 rk356x_linux_release.xml
# 例如:打印的版本号为v1.3.0，更新时间为20220620
# <SDK>/.repo/manifests/rk356x_linux_release_v1.3.0_20220620.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/rk356x_linux$ cat RK356X_Linux5.10_SDK_Note.md
```

或者参考工程目录：

```
<SDK>/docs/RK356X/RK356X_Linux5.10_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：

RK3566 EVB 硬件开发指南：

```
<SDK>/docs/RK356X/Hardware/Rockchip_RK3566_EVB2_User_Guide_V1.1_CN.pdf
```

RK3568 EVB硬件开发指南：

```
<SDK>/docs/RK356X/Hardware/Rockchip_RK3568_EVB_User_Guide_V1.0_CN.pdf
```

RK3568 NVR硬件开发指南：

```
<SDK>/docs/RK356X/Hardware/Rockchip_RK3568_NVR_DEMO_User_Guide_V1.2_CN.pdf
```

### 4. IO电源设计注意事项

主控电源域的IO电平要与对接外设芯片的IO电平保持一致，还要注意软件的电压配置要跟硬件的电压一致，否则可能会导致GPIO的损坏。



#### 注意

关于GPIO电源域IO电平匹配问题：

GPIO的电源域PMUIO0\_VDD, PMUIO1\_VDD, VCCIO1\_VDD, VCCIO2\_VDD, VCCIO3\_VDD, VCCIO4\_VDD, VCCIO5\_VDD, VCCIO6\_VDD, VCCIO7\_VDD, 这些电源的电压要跟所接的外设的IO电平的电压保持一致，否则可能会导致GPIO的损坏。

还要注意软件的电压配置要跟硬件的电压一致：比如硬件IO电平接1.8V，软件的电压配置也要相应的配成1.8V；硬件IO电平接3.3V，软件的电压配置也要用3.3V，否则也可能会导致GPIO的损坏。

更多信息参考：

```
<SDK>/docs/RK356X/Rockchip_RK356X_Introduction_IO_Power_Domains_Configuration.pdf  
<SDK>/docs/Common/IO-DOMAIN/Rockchip_Developer_Guide_Linux_IO_DOMAIN_CN.pdf
```

## 5. 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，主要是一些应用Demo。
- buildroot：基于 Buildroot开发的根文件系统。
- debian：基于 Debian 10或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 4.19、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 4.1 开发的根文件系统。

## 6. SDK 编译说明

### 6.1 SDK板级配置

进入工程 <SDK>/device/rockchip/rk356x 目录：

| 板级配置  | 说明                                      |
|---|---|
| BoardConfig-rk3566-evb2-lp4x-v10-32bit.mk       | 适用于 RK3566 EVB 搭配 LPDDR4 开发板，运行32位根文件系统 |
| BoardConfig-rk3566-evb2-lp4x-v10.mk             | 适用于 RK3566 EVB 搭配 LPDDR4 开发板            |
| BoardConfig-rk3568-evb1-ddr4-v10-32bit.mk       | 适用于 RK3568 EVB 搭配 LPDDR4 开发板，运行32位根文件系统 |
| BoardConfig-rk3568-evb1-ddr4-v10-spi-nor-64M.mk | 适用于 RK3568 EVB 搭配 DDR4/SPI NOR 开发板      |
| BoardConfig-rk3568-evb1-ddr4-v10.mk             | 适用于 RK3568 EVB 搭配 DDR4 开发板              |
| BoardConfig-rk3568-uvb-evb1-ddr4-v10.mk         | 适用于 RK3568 UVC 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**RK3566 EVB 搭配 LPDDR4 开发板，运行32位根文件系统**的板级配置：

```
./build.sh device/rockchip/rk356x/BoardConfig-rk3566-evb2-lp4x-v10-32bit.mk
```

选择**RK3566 EVB 搭配 LPDDR4 开发板**的板级配置：

```
./build.sh device/rockchip/rk356x/BoardConfig-rk3566-evb2-lp4x-v10.mk
```

选择**RK3568 EVB 搭配 DDR4/SPI NOR 开发板**的板级配置：

```
./build.sh device/rockchip/rk356x/BoardConfig-rk3568-evb1-ddr4-v10-spi-nor-64M.mk
```

选择**RK3568 EVB 搭配 DDR4 开发板，运行32位根文件系统**的板级配置：

```
./build.sh device/rockchip/rk356x/BoardConfig-rk3568-evb1-ddr4-v10-32bit.mk
```

选择**RK3568 EVB 搭配 DDR4 开发板**的板级配置：

```
./build.sh device/rockchip/rk356x/BoardConfig-rk3568-evb1-ddr4-v10.mk
```

选择**RK3568 UVC 开发板**的板级配置：

```
./build.sh device/rockchip/rk356x/BoardConfig-rk3568-uvc-evb1-ddr4-v10.mk
```

方法2

```
rk356x$ ./build.sh lunch
processing option: lunch

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

1. BoardConfig-ab-base.mk
2. BoardConfig-rk3566-evb2-lp4x-v10-32bit.mk
3. BoardConfig-rk3566-evb2-lp4x-v10.mk
4. BoardConfig-rk3568-evb1-ddr4-v10-32bit.mk
5. BoardConfig-rk3568-evb1-ddr4-v10-spi-nor-64M.mk
6. BoardConfig-rk3568-evb1-ddr4-v10.mk
7. BoardConfig-rk3568-uvc-evb1-ddr4-v10.mk
8. BoardConfig-security-base.mk
9. BoardConfig.mk
Which would you like? [0]:
```

## 6.2 查看编译命令

在根目录执行命令：./build.sh -h|help

```
rk356x$ ./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

```

rk356x$ ./build.sh -h kernel
###Current SDK Default [ kernel ] Build Command###
cd kernel
make ARCH=arm64 rockchip_linux_defconfig
make ARCH=arm64 rk3568-evb1-ddr4-v10-linux.img -j12

```

## 6.3 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, 可以通过设置环境变量 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
```

## 6.4 各模块编译及打包

### 6.4.1 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
```

### 6.4.2 Kernel编译

```
### Kernel编译命令
./build.sh kernel

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

### 6.4.3 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

## 6.4.4 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\_芯片命名/images下生成 rootfs.ext4。

## 6.4.5 Buildroot 的交叉编译

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RK3568，其交叉编译工具位于 buildroot/output/rockchip\_rk3568/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（只对当前控制台有效）：

```
source envsetup.sh
```

输入命令查看：

```
cd buildroot/output/rockchip_rk3568/host/usr/bin  
./aarch64-linux-gcc --version
```

此时会打印如下信息：

```
aarch64-linux-gcc.br_real (Buildroot linux-5.10-gen-rkr2-297-ge5de4b00a7) 11.3.0
```

### 6.4.5.1 Buildroot 中模块编译

比如 rockchip-test 模块，常用相关编译命令如下：

- 编译 rockchip-test

```
SDK$make rockchip-test
```

- 重编 rockchip-test

```
SDK$make rockchip-test-rebuild
```

- 删除 rockchip\_test

```
SDK$make rockchip-test-dirclean
```

或者

```
SDK$rm -rf output/rockchip_rk3568/build/rockchip-test-master/
```

## 6.4.6 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=buster TARGET=desktop ARCH=arm64 ./mk-base-debian.sh
```

编译完成会在 debian/ 目录下生成: linaro-buster-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/buster-desktop-arm64/chroot/test-dev-null:  
Permission denied E: Cannot install into target '/rootfs/ubuntu-build-  
service/buster-desktop-arm64/chroot' mounted with noexec or nodev
```

解决方法:

```
mount -o remount,exec,dev xxx(工程目录) yyy(mount得位置), 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-buster.sh
```

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

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

此时会生成 linaro-rootfs.img。

## 6.4.7 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：

RK3566/RK3568 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/latest/rootfs.img，默认用户名登录是 root。

Yocto 更多信息请参考 [Rockchip Wiki](#)。

## 6.4.8 固件的打包

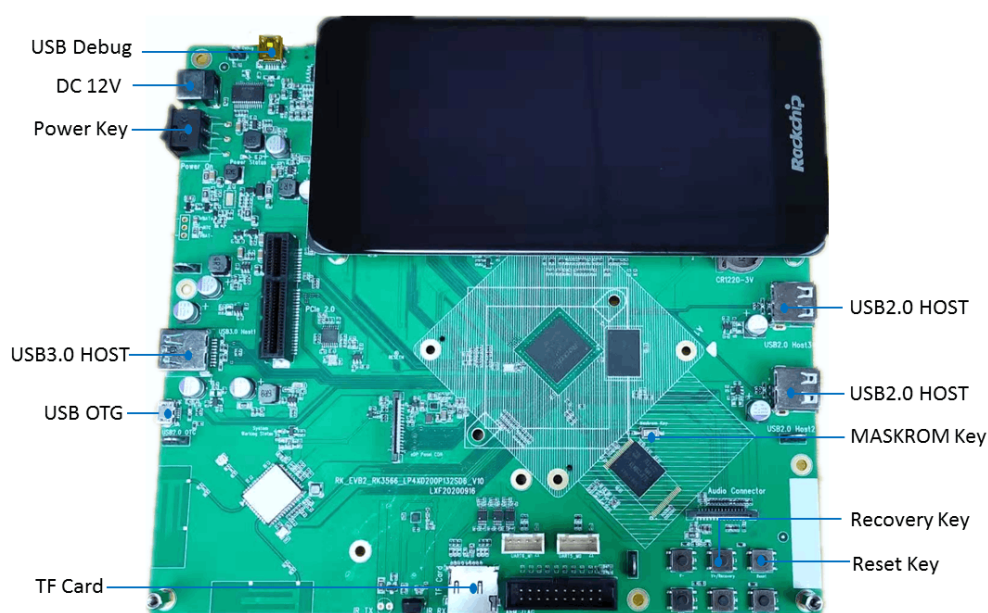
上面 Kernel/U-Boot/Recovery/Rootfs 各个部分的编译后，进入工程目录根目录执行以下命令自动完成所有固件打包到 rockdev 目录下：

固件生成：

```
./mkfirmware.sh
```

## 7. 刷机说明

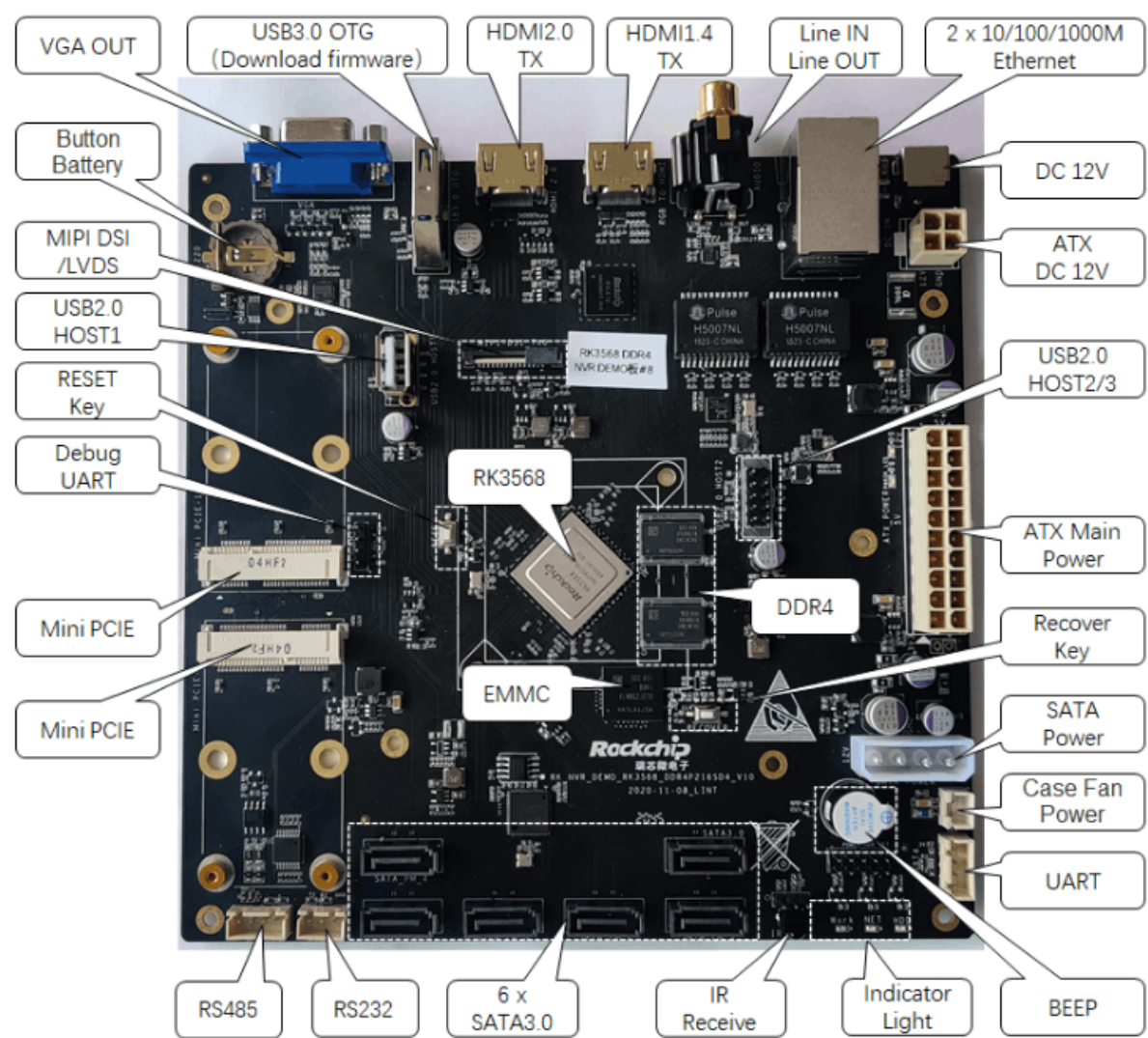
RK3566 EVB2 接口分布图如下：



RK3568 EVB1 开发板接口分布图如下：



RK3568 NVR 开发板接口分布图如下:



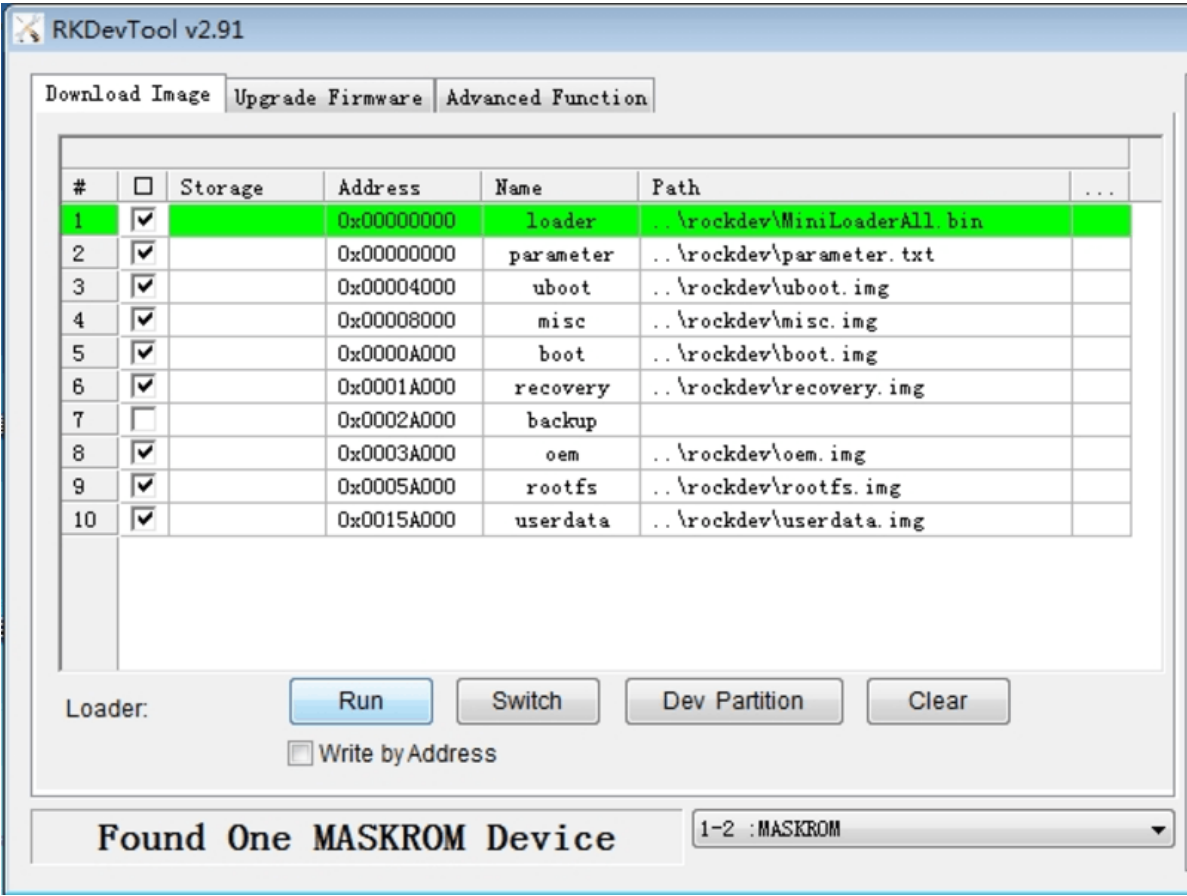


## 7.1 Windows 刷机说明

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
```

## 7.2 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 V2.17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 rockdev/MiniLoaderAll.bin -noreset
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 rockdev/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

```

## 7.3 系统分区说明

默认分区说明 ( 下面是 RK3568 EVB 分区参考)

| Number | Start (sector) | End (sector) | Size  | Name     |
|--------|----------------|--------------|-------|----------|
| 1      | 16384          | 24575        | 4M    | uboot    |
| 2      | 24576          | 32767        | 4M    | misc     |
| 3      | 32768          | 163839       | 64M   | boot     |
| 4      | 163840         | 294911       | 32M   | recovery |
| 5      | 294912         | 360447       | 32M   | bakcup   |
| 6      | 360448         | 12943359     | 6144M | rootfs   |
| 7      | 12943360       | 12943359     | 128M  | oem      |
| 8      | 13205504       | 61120478     | 22.8G | 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。

## 8. RK356X SDK 固件

- 百度云网盘

[Buildroot](#)

[Debian](#)

[Yocto](#)

- 微软 OneDriver

[Buildroot](#)

[Debian](#)

[Yocto](#)