Rockchip RK3399 Linux SDK 快速入门

文档标识：RK-FB-YF-944

发布版本：V1.3.0

日期：2023-09-20

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RK3399 Linux SDK的基本使用方法，旨在帮助开发者快速了解并使用RK3399 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**
RK3399	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	更新支持Linux5.10版本
2022-11-20	V1.0.2	Caesar Wang	更新Linux刷机说明
2023-04-20	V1.1.0	Caesar Wang	适配新版本SDK
2023-05-20	V1.1.1	Caesar Wang	更新SDK到V1.1.1
2023-06-20	V1.2.0	Caesar Wang	更新SDK到V1.2.0
2023-06-20	V1.3.0	Caesar Wang	更新SDK到V1.3.0

目录

1. 搭建开发环境

我们推荐使用 Ubuntu 22.04 或更高版本的系统进行编译。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 22.04 或更高版本系统。

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 22.04	20.10.21	pass	pass
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)

1.1 安装库和工具集

使用命令行进行设备开发时，可以通过以下步骤安装编译SDK需要的库和工具。

使用如下apt-get命令安装后续操作所需的库和工具：

```
sudo apt-get update && sudo apt-get install git ssh make gcc libssl-dev \
liblz4-tool expect expect-dev 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 libgmp-dev \
libmpc-dev bc python-is-python3 python2
```

> 说明：
> 安装命令适用于Ubuntu22.04，其他版本请根据安装包名称采用对应的安装命令，若编译遇到报错，可以视报错信息，安装对应的软件包。其中：
>
> - 如果PC在编译Buildroot时无法访问Google网站，需设置DNS来支持使用国内镜像kgithub.com下载dl包。
> - python要求安装python 3.6及以上版本，此处以python 3.6为例。
> - make要求安装 make 4.0及以上版本，此处以 make 4.2为例。
> - lz4要求安装 lz4 1.7.3及以上版本。
> - 编译Yocto需要VPN网络，git没有CVE-2022-39253安全检测补丁。

1.1.1 设置DNS支持kgithub.com

```
sudo sed -i '$a 43.154.68.204\tkgithub.com' /etc/hosts
sudo sed -i '$a 43.155.83.75\traw.kgithub.com
objects.githubusercontent.kgithub.com' /etc/hosts
```

1.1.2 检查和升级主机的`python`版本

检查和升级主机的`python`版本方法如下：

- 检查主机`python`版本

```
$ python3 --version
Python 3.10.6
```

如果不满足python>=3.6版本的要求， 可通过如下方式升级：

- 升级`python 3.6.15`新版本

```
PYTHON3_VER=3.6.15
echo "wget
https://www.python.org/ftp/python/${PYTHON3_VER}/Python-${PYTHON3_VER}.tgz"
echo "tar xf Python-${PYTHON3_VER}.tgz"
echo "cd Python-${PYTHON3_VER}"
echo "sudo apt-get install libsqlite3-dev"
echo "./configure --enable-optimizations"
echo "sudo make install -j8"
```

1.1.3 检查和升级主机的`make`版本

检查和升级主机的`make`版本方法如下：

- 检查主机`make`版本

```
$ make -v
GNU Make 4.2
Built for x86_64-pc-linux-gnu
```

- 升级`make 4.2`新版本

```
$ sudo apt update && sudo apt install -y autoconf autopoint

git clone https://gitee.com/mirrors/make.git
cd make
git checkout 4.2
git am $BUILDROOT_DIR/package/make/*.patch
autoreconf -f -i
./configure
make make -j8
sudo install -m 0755 make /usr/bin/make
```

1.1.4 检查和升级主机的`lz4`版本

检查和升级主机的`lz4`版本方法如下：

- 检查主机`lz4`版本

```
$ lz4 -v
*** LZ4 command line interface 64-bits v1.9.3, by Yann Collet ***
```

- 升级`lz4`新版本

```
git clone https://gitee.com/mirrors/LZ4_old1.git
cd LZ4_old1

make
sudo make install
sudo install -m 0755 lz4 /usr/bin/lz4
```

1.1.5 检查和升级主机的`git`版本

- 检查主机`git`版本

```
$ git -v
git version 2.38.0
```

- 升级`git`新版本

```
$ sudo apt update && sudo apt install -y libcurl4-gnutls-dev

git clone https://gitee.com/mirrors/git.git --depth 1 -b v2.38.0
cd git
make git -j8
make install
sudo install -m 0755 git /usr/bin/git
```

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/cn/RK3399`目录下获取，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 RK3399 的开发调试工作，随 SDK 发布《Rockchip_RK3399_Datasheet_V2.1_20200323.pdf》芯片手册。可在`docs/cn/RK3399/Datasheet`目录下获取，并会不断完善更新。

2.3 Debian开发指南

为帮助开发工程师更快上手熟悉 RK3399 Debian的开发调试，随 SDK 发布《Rockchip_Developer_Guide_Debian_CN.pdf》开发指南，可在`docs/cn/Linux/System`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 RK3399 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/cn/Linux/System`下获取，并会不断完善更新。

2.5 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 rk3399_linux5.10_release.xml
# 例如:打印的版本号为v1.3.0，更新时间为20230920
# <SDK>/.repo/manifests/rk3399_linux/rk3399_linux5.10_release_v1.2.0_20230920.xml
```

软件发布版本升级更新内容通过工程文本可以查看，参考工程目录：

```
<SDK>/.repo/manifests/rk3399_linux/RK3399_Linux5.10_SDK_Note.md
或者
<SDK>/docs/cn/RK3399/RK3399_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：

RK3399 硬件设计指南：

```
<SDK>/docs/cn/RK3399/Hardware/Rockchip_RK3399_Hardware_Design_Guide_V1.3_CN.pdf
```

RK3399 挖掘机硬件开发指南：

```
<SDK>/docs/cn/RK3399/Hardware/Rockchip_RK3399_User_Manual_Excavator_EVB_V3.0_CN.p
df
```

RK3399 IND 行业板硬件开发指南：

```
<SDK>/docs/cn/RK3399/Hardware/Rockchip_RK3399_User_Manual_IND_EVB_V1.1.pdf
```

4. IO电源设计注意事项

更多信息参考：

```
<SDK>/docs/cn/RK3399/Rockchip_RK3399_Introduction_IO_Power_Domains_Configuration.
pdf
<SDK>/docs/cn/Common/IO-DOMAIN/Rockchip_Developer_Guide_Linux_IO_DOMAIN_CN.pdf
```

5. SDK 配置框架说明

5.1 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。

SDK 工程目录包含有 buildroot、debian、app、kernel、u-boot、device、docs、external 等目录。采用manifest来管理仓库，用repo工具来管理每个目录或其子目录会对应的 git 工程。

- app：存放上层应用 APP，主要是一些应用Demo。
- buildroot：基于 Buildroot（2021）开发的根文件系统。
- debian：基于 Debian bullseye(11) 开发的根文件系统。
- device/rockchip：存放芯片板级配置以及一些编译和打包固件的脚本和文件。
- 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。
- external：存放第三方相关仓库，包括显示、音视频、摄像头、网络、安全等。
- kernel：存放 Kernel 开发的代码。
- output：存放每次生成的固件信息、编译信息、XML、主机环境等。
- prebuilts：存放交叉编译工具链。
- rkbin：存放 Rockchip 相关二进制和工具。
- rockdev：存放编译输出固件,实际软链接到`output/firmware`。
- 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 4.0开发的根文件系统。

6. SDK 编译说明

SDK可通过`make`或`./build.sh`加目标参数进行相关功能的配置和编译。
具体参考`device/rockchip/common/README.md`编译说明。

6.1 SDK编译命令查看

`make help`，例如：

```
$ make help
  menuconfig              - interactive curses-based configurator
  oldconfig               - resolve any unresolved symbols in .config
  syncconfig              - Same as oldconfig, but quietly, additionally update
deps
  olddefconfig            - Same as syncconfig but sets new symbols to their
default value
  savedefconfig           - Save current config to RK_DEFCONFIG (minimal config)
  ...
```

make实际运行是`./build.sh`

即也可运行`./build.sh <target>`来编译相关功能，具体可通过`./build.sh help`查看具体编译命令。

```
$ ./build.sh -h
Usage: build.sh [OPTIONS]
Available options:
lunch               - choose defconfig
*_defconfig         - switch to specified defconfig
olddefconfig        - resolve any unresolved symbols in .config
savedefconfig       - save current config to defconfig
menuconfig          - interactive curses-based configurator
kernel-5.10         - build kernel 5.10
kernel              - build kernel
modules             - build kernel modules
loader              - build loader (uboot|spl)
uboot               - build u-boot
spl                 - build spl
uefi                - build uefi
wifibt              - build Wifi/BT
rootfs              - build rootfs (default is buildroot)
buildroot           - build buildroot rootfs
yocto               - build yocto rootfs
debian              - build debian rootfs
recovery            - build recovery
pcba                - build PCBA
security_check      - check contidions for security features
createkeys          - build secureboot root keys
security_uboot      - build uboot with security paramter
security_boot       - build boot with security paramter
security_recovery   - build recovery with security paramter
security_rootfs     - build rootfs and some relevant images with security paramter
(just for dm-v)
updateimg           - build update image
otapackage          - build OTA update image
sdpackage           - build SDcard update image
firmware            - generate and check firmwares
all                 - build all basic image
save                - save images and build info
allsave             - build all & firmware & updateimg & save
cleanall            - cleanup
```

```
post-rootfs        - trigger post-rootfs hook scripts
shell              - setup a shell for developing
help               - usage

Default option is 'allsave'.
```

6.2 SDK板级配置

进入工程`<SDK>/device/rockchip/rk3399`目录：

板级配置	说明
rockchip_rk3399_evb_ind_lpddr4_defconfig	适用于 RK3399 IND 行业板
rockchip_rk3399_firefly_defconfig	适用于 RK3399 Firefly 开发板
rockchip_rk3399_sapphire_excavator_lp4_defconfig	适用于 RK3399 挖掘机搭配LPDDR4板
rockchip_rk3399_sapphire_excavator_defconfig	适用于 RK3399 挖掘机开发板

方法1
`./build.sh` 后面加上板级配置文件, 例如：

选择**RK3399 IND**的板级配置：

```
./build.sh device/rockchip/rk3399/rockchip_rk3399_evb_ind_lpddr4_defconfig
```

选择**RK3399 Firefly 开发板**的板级配置：

```
./build.sh device/rockchip/rk3399/rockchip_rk3399_firefly_defconfig
```

选择**RK3399 挖掘机搭配LPDDR4板**的板级配置：

```
./build.sh
device/rockchip/rk3399/rockchip_rk3399_sapphire_excavator_lp4_defconfig
```

选择**RK3399 挖掘机开发板**的板级配置：

```
./build.sh device/rockchip/rk3399/rockchip_rk3399_sapphire_excavator_defconfig
```

方法2

```
rk3399$ ./build.sh lunch
Pick a defconfig:

1. rockchip_defconfig
2. rockchip_rk3399_evb_ind_lpddr4_defconfig
3. rockchip_rk3399_firefly_defconfig
4. rockchip_rk3399_sapphire_excavator_defconfig
5. rockchip_rk3399_sapphire_excavator_lp4_defconfig
Which would you like? [1]:
...
```

6.3 SDK配置不同启动/内核/系统等组件

SDK可通过`make menuconfig`进行相关配置，目前可配组件主要如下：

```
(rk3399) SoC
   Rootfs  --->
   Loader (u-boot)  --->
   Kernel  --->
   Boot  --->
   Recovery (buildroot)  --->
   PCBA test (buildroot)  --->
   Security  --->
   Update (OTA and A/B)  --->
   Firmware  --->
   Extra partitions  --->
   Others configurations  --->
```

通过以上config，可选择不同rootfs/loader/kernel等配置，进行各种定制化编译。另外还带有强大命令行切换功能。

> 注意：menuconfig配置之后，需要`make savedefconfig`保存配置

6.4 全自动编译

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`./build.sh ./mkfirmware.sh 进行固件打包

./build.sh     # 编译模块代码（u-Boot，kernel，Rootfs，Recovery）
               # 打包成update.img完整升级包
               # 所有编译信息复制和生成到out目录下
```

默认是 Buildroot，可以通过设置坏境变量 `RK_ROOTFS_SYSTEM` 指定不同 rootfs。`RK_ROOTFS_SYSTEM`目前可设定三种系统：buildroot、debian、 yocto 。

比如需要 debain 可以通过以下命令进行生成：

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

> 注意：
> SDK每次更新建议都清理之前的编译产物，直接运行`./build.sh cleanall`即可 。

6.5 各模块编译

6.5.1 U-Boot编译

```
./build.sh uboot
```

6.5.2 Kernel编译

- 方法一

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

- 方法二

```
cd kernel
export CROSS_COMPILE=../prebuilts/gcc/linux-x86/aarch64/gcc-arm-10.3-2021.07-
x86_64-aarch64-none-linux-gnu/bin/aarch64-none-linux-gnu-
make ARCH=arm64 rockchip_linux_defconfig
make ARCH=arm64 rk3399-evb-ind-lpddr4-linux.img -j
```

- 方法三

```
cd kernel
export CROSS_COMPILE=aarch64-linux-gnu-
make ARCH=arm64 rockchip_linux_defconfig
make ARCH=arm64 rk3399-evb-ind-lpddr4-linux.img -j
```

6.5.3 Recovery编译

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

注：Recovery是非必需的功能，有些板级配置不会设置

6.5.4 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_rk3399/images`下生成不同格式的镜像, 默认使用rootfs.ext4格式。

6.5.5 Debian编译

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

编译后在 debian 目录下生成 linaro-rootfs.img。

> 说明：需要预先安装相关依赖包
> sudo apt-get install binfmt-support qemu-user-static live-build
> sudo dpkg -i ubuntu-build-service/packages/*
> sudo apt-get install -f

具体可参考Debian开发文档参考：

`<SDK>/docs/cn/Linux/System/Rockchip_Developer_Guide_Debian_CN.pdf`

6.5.6 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：

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

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

默认用户名登录是 root。Yocto 更多信息请参考 Rockchip Wiki。

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

6.5.7 交叉编译

6.5.7.1 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

可从以下地址下载工具链：

点击这里

6.5.7.2 Buildroot内置交叉编译

可通过 `source buildroot/envsetup.sh` 来设置不同芯片和目标功能的配置

```
$ source buildroot/envsetup.sh
Top of tree:
Pick a board:
24. rockchip_rk3399
25. rockchip_rk3399_base
26. rockchip_rk3399_recovery
```

默认选择24，`rockchip_rk3399` 。然后进入RK3399的Buildroot目录，开始相关模块的编译。

> 其中`rockchip_rk3399_base`是Buildroot 系统基础组件编译，`rockchip_rk3399_recovery`是用来编译Recovery模块。

比如编译 rockchip_test 模块，常用相关编译命令如下：

进入 buildroot目录

```
SDK#cd buildroot
```

- 编译 rockchip_test

```
buildroot#make rockchip_test
```

- 重编 rockchip_test

```
buildroot#make rockchip_test-rebuild
```

- 删除 rockchip_test

```
buildroot#make rockchip_test-dirclean
或者
buildroot#rm -rf output/rockchip_rk3399/build/rockchip-test-master/
```

若需要编译单个模块或者第三方应用，需交叉编译环境进行配置。比如RK3399其交叉编译工具位于`buildroot/output/rockchip_rk3399/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 buildroot/envsetup.sh rockchip_rk3399
```

输入命令查看：

```
cd buildroot/output/rockchip_rk3399/host/usr/bin
./aarch64-linux-gcc --version
```

此时会打印如下信息：

```
aarch64-linux-gcc.br_real (Buildroot) 12.3.0
```

> 保存到rootfs配置文件
> buildroot$ make update-defconfig

6.5.8 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`output/firmware`目录下：

固件生成:

```
./build.sh firmware
```

7. 刷机说明

RK3399 挖掘机接口分布图如下：

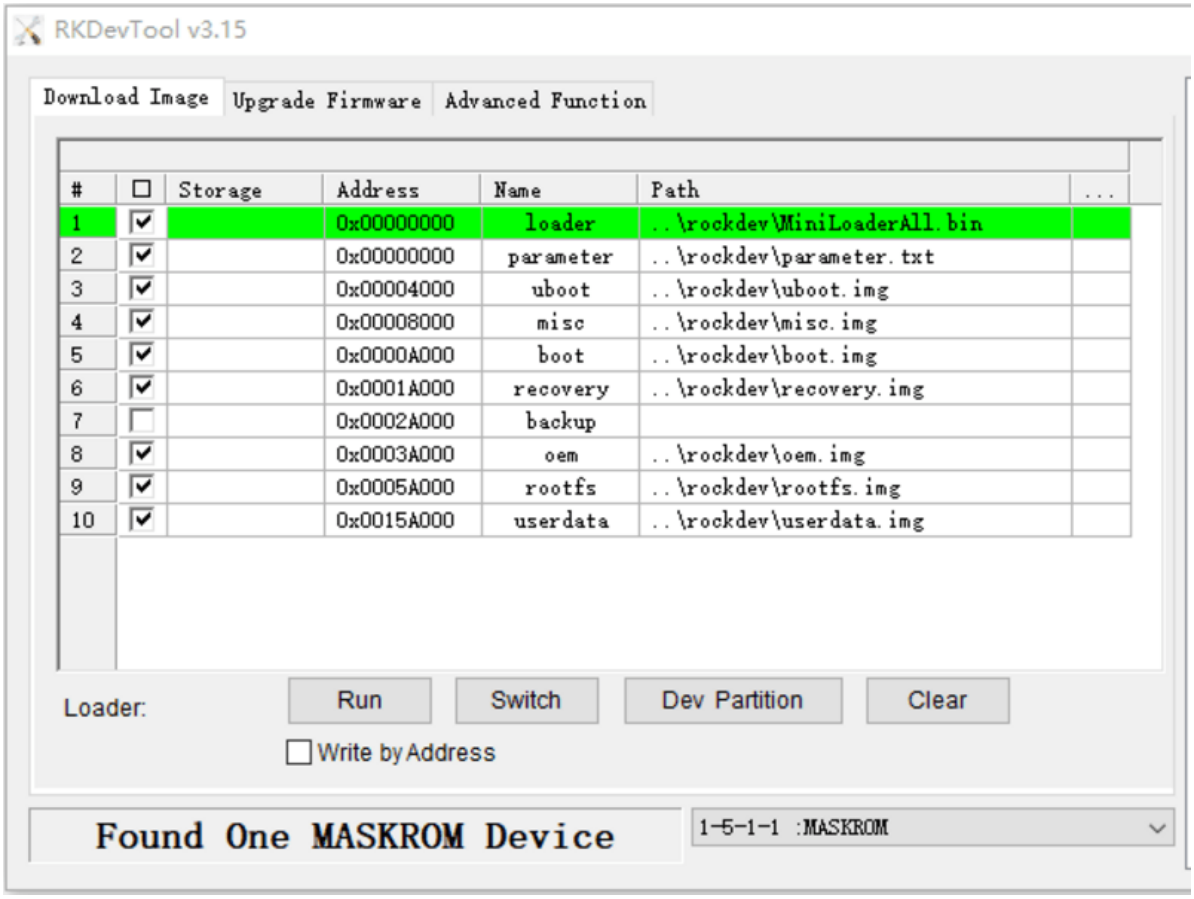
RK3399 IND 行业板接口分布图如下：

7.1 Windows 刷机说明

SDK 提供 Windows 烧写工具(工具版本需要 V3.15 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_v5.12.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 -trust 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
```

7.3 系统分区说明

默认分区说明 (下面是 RK3399 EVB 分区参考）

Number	Start (sector)	End (sector)	Size	Name
1	16384	24575	4096K	uboot
2	24576	32767	4096K	trust
3	32768	40959	4096K	misc
4	40960	106495	32M	boot
5	106496	303104	32M	recovery
6	172032	237567	32M	bakcup
7	237568	368639	64M	oem
8	368640	12951551	6144M	rootfs
9	12951552	30535646	8585M	userdata

- 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。
- 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. RK3399 SDK 固件

点击这里