Rockchip PX30 Linux SDK 快速入门

文件标识：RK-JC-YF-941

发布版本：V1.9.1

日期：2023-04-20

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。

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 PX30 Linux SDK 的基本使用方法，旨在帮助开发者快速了解并使用 PX30 Linux 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	Kernel
PX30/PX30-S	2021.11	N/A	N/A	4.4/4.19/5.10

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	WJL	2022-06-16	初始版本。
V1.8.0	WJL	2022-06-20	版本同步至 V1.8.0。
V1.8.1	LinJianHua	2022-09-20	支持Linux5.10 SDK
V1.8.2	Caesar Wang	2022-11-20	更新刷机说明
V1.9.1	WJL	2023-04-20	更新编译说明

目录

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：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 64位 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 binfmt-support bison bzip2 chrpath cmake \
     cpp-aarch64-linux-gnu device-tree-compiler diffstat expat expect \
     fakeroot flex g++ g++-aarch64-linux-gnu gawk gcc gcc-multilib git \
     g++-multilib gpgv2 libgucharmap-2-90-dev liblz4-tool libssl-dev live-build \
     make ncurses-dev patchelf qemu-user-static ssh texinfo unzip
```

建议使用 Ubuntu 20.04 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。

随 SDK 发布《Rockchip_Developer_Guide_Linux_Docker_Deploy_CN.pdf》，可在 docs/ApplicationNote 下获取，并会不断完善更新。

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	20.10.12	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）

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 PX30 的开发调试工作，随 SDK 发布《Rockchip PX30-S Datasheet V1.0-20220301.pdf》芯片手册。

2.3 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：

```
# Linux4.4 SDK
.repo/manifests$ realpath px30_linux_release.xml
# 例如：打印的版本号为 v1.8.0，更新时间为 20220620
<SDK>/.repo/manifests/px30_linux_release_v1.8.0_20220620.xml

# Linux4.19 SDK
.repo/manifests$ realpath px30_linux4.19_release.xml
# 例如：打印的版本号为 v1.2.0，更新时间为 20220620
<SDK>/.repo/manifests/px30_linux4.19_release_v1.2.0_20220620.xml

# Linux5.10 SDK
.repo/manifests$ realpath px30_linux5.10_release.xml
# 例如：  打印的版本号为 v1.1.0，更新时间为 20230420
<SDK>/.repo/manifests/px30_linux5.10_release_v1.1.0_20230420.xml
```

软件发布版本更新内容通过工程文本可以查看，具体方法如下：

```
# Linux4.4 SDK
.repo/manifests$ cat PX30_Linux_SDK_Release_Note.md

# Linux4.19 SDK
.repo/manifests$ cat PX30_Linux4.19_SDK_Release_Note.md

# Linux5.10 SDK
.repo/manifests$ cat PX30_Linux5.10_SDK_Release_Note.md
```

或者参考工程目录：

```
# Linux4.4 SDK
<SDK>/docs/PX30/PX30_Linux_SDK_Release_Note.md

# Linux4.19 SDK
<SDK>/docs/PX30/PX30_Linux4.19_SDK_Release_Note.md

# Linux5.10 SDK
<SDK>/docs/PX30/PX30_Linux5.10_SDK_Release_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：

PX30 硬件设计指南：

```
<SDK>/docs/PX30/Hardware/Rockchip_PX30_Hardware_Design_Guide_V1.3_20191206_CN.pdf
```

PX30 EVB 硬件开发指南：

```
<SDK>/docs/PX30/Hardware/Rockchip_PX30_User_Manual_EVB_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 app、buildroot、device、docs、external、kernel、output、prebuilts、rkbin、tools、u-boot 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。
- 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。
- external：存放第三方相关仓库，包括音频、视频、网络、recovery 等。
- kernel：Kernel 代码目录的软连接，默认链接到 kernel-5.10 目录。
- 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 4.0 开发的根文件系统。

4.2 SDK 板级配置

进入工程 `<SDK>/device/rockchip/px30` 目录：

板级配置	说明
rockchip_defconfig	默认配置
rockchip_px30_evb_ddr3_v10_defconfig rockchip_px30_evb_ddr3_v10_32bit_defconfig	适用于 PX30 EVB V10 搭配 DDR3 开发板
rockchip_px30_evb_ddr3_v11_defconfig rockchip_px30_evb_ddr3_v11_32bit_defconfig	适用于 PX30 EVB V11 搭配 DDR3 开发板
rockchip_px30_evb_ddr4_v10_defconfig rockchip_px30_evb_ddr4_v10_32bit_defconfig	适用于 PX30 EVB V10 搭配 DDR4 开发板
rockchip_px30_robot64_defconfig rockchip_px30_robot64_no_gpu_defconfig	适用于 PX30 Robot 小系统开发

方法一：

`./build.sh` 后面加上板级配置文件，例如：

选择 **PX30 EVB DDR3 V10** 的板级配置：

```
./build.sh rockchip_px30_evb_ddr3_v10_defconfig
或
./build.sh rockchip_px30_evb_ddr3_v10_32bit_defconfig
```

选择 **PX30 EVB DDR3 V11** 的板级配置：

```
./build.sh rockchip_px30_evb_ddr3_v11_defconfig
或
./build.sh rockchip_px30_evb_ddr3_v11_32bit_defconfig
```

选择 **PX30 EVB DDR4 V10** 的板级配置：

```
./build.sh rockchip_px30_evb_ddr4_v10_defconfig
或
./build.sh rockchip_px30_evb_ddr4_v10_32bit_defconfig
```

选择 **PX30 Robot** 的板级配置：

```
./build.sh rockchip_px30_robot64_defconfig
或
./build.sh rockchip_px30_robot64_no_gpu_defconfig
```

方法二：

```
px30$ ./build.sh lunch

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

1. rockchip_defconfig
2. rockchip_px30_evb_ddr3_v10_32bit_defconfig
3. rockchip_px30_evb_ddr3_v10_defconfig
4. rockchip_px30_evb_ddr3_v11_32bit_defconfig
5. rockchip_px30_evb_ddr3_v11_defconfig
6. rockchip_px30_evb_ddr4_v10_32bit_defconfig
7. rockchip_px30_evb_ddr4_v10_defconfig
8. rockchip_px30_robot64_defconfig
9. rockchip_px30_robot64_no_gpu_defconfig
Which would you like? [1]:
...
```

4.3 查看编译命令

在根目录执行命令：`./build.sh -h|help`

```
px30$ ./build.sh -h
```

```
Usage: build.sh [OPTIONS]
Available options:
chip                - choose chip
lunch               - choose defconfig
*_defconfig         - switch to specified defconfig
    Available defconfigs:
    rockchip_defconfig
    rockchip_px30_evb_ddr3_v10_32bit_defconfig
    rockchip_px30_evb_ddr3_v10_defconfig
    rockchip_px30_evb_ddr3_v11_32bit_defconfig
    rockchip_px30_evb_ddr3_v11_defconfig
    rockchip_px30_evb_ddr4_v10_32bit_defconfig
    rockchip_px30_evb_ddr4_v10_defconfig
    rockchip_px30_robot64_defconfig
    rockchip_px30_robot64_no_gpu_defconfig
olddefconfig        - resolve any unresolved symbols in .config
savedefconfig       - save current config to defconfig
menuconfig          - interactive curses-based configurator
kernel-4.19         - build kernel 4.19
kernel-4.4          - build kernel 4.4
kernel-5.10         - build kernel 5.10
kernel              - build kernel
modules             - build kernel modules
linux-headers       - build linux-headers
loader              - build loader (uboot|spl)
uboot               - build u-boot
spl                 - build spl
uefi                - build uefi
wifibt              - build Wifi/BT
rootfs              - build default rootfs
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)
firmware            - generate and check firmwares
updateimg           - build update image
otapackage          - build OTA update image
sdpackage           - build SDcard update image
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'.
```

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 和 ./build.sh allsave 命令效果一致
# ./build.sh 命令将按顺序完成以下操作：
#     1/ ./build.sh all 编译模块代码（U-boot、Kernel、Rootfs、Recovery）
#         编译后的镜像会存放在 output/firmware 目录中
#     2/ ./build.sh firmware 打包额外的固件（oem、userdata 等镜像）
#         打包的镜像会存放在 output/firmware 目录中
#     3/ ./build.sh updateimg 对所有镜像文件打包
#         打包的 update.img 存放在 output/firmware 目录中
#     4/ ./build.sh save 保存本次编译的固件、日志、补丁、配置
#         上述内容将存放在 output/$defconfig 目录中，$defconfig 为使用的编译配置的名称
./build.sh
```

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 。

比如需要 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
```

4.5.2 Kernel 编译

```
### Kernel编译命令
./build.sh kernel
```

4.5.3 Recovery 编译

```
### Recovery编译命令
./build.sh 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_px30_64/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=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 (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，可以从以下链接获取预编生成的包，放在当前目录直接执行下一步操作。

- Debian10 Base 32bit
- Debian10 Base 64bit

(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。

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：

PX30 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 内置的交叉编译链

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 64位工具链

4.5.7.2 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 PX30，其交叉编译工具位于 `buildroot/output/rockchip_px30_64/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_px30_64/host/usr/bin
./aarch64-linux-gcc --version
```

此时会打印如下信息：

```
aarch64-linux-gcc.br_real (Buildroot -g61d86e7108) 11.3.0
```

4.5.7.3 Buildroot 中模块编译

比如编译 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 /buildroot/output/rockchip_px30_64/build/rockchip-test-master
```

4.5.8 固件的打包

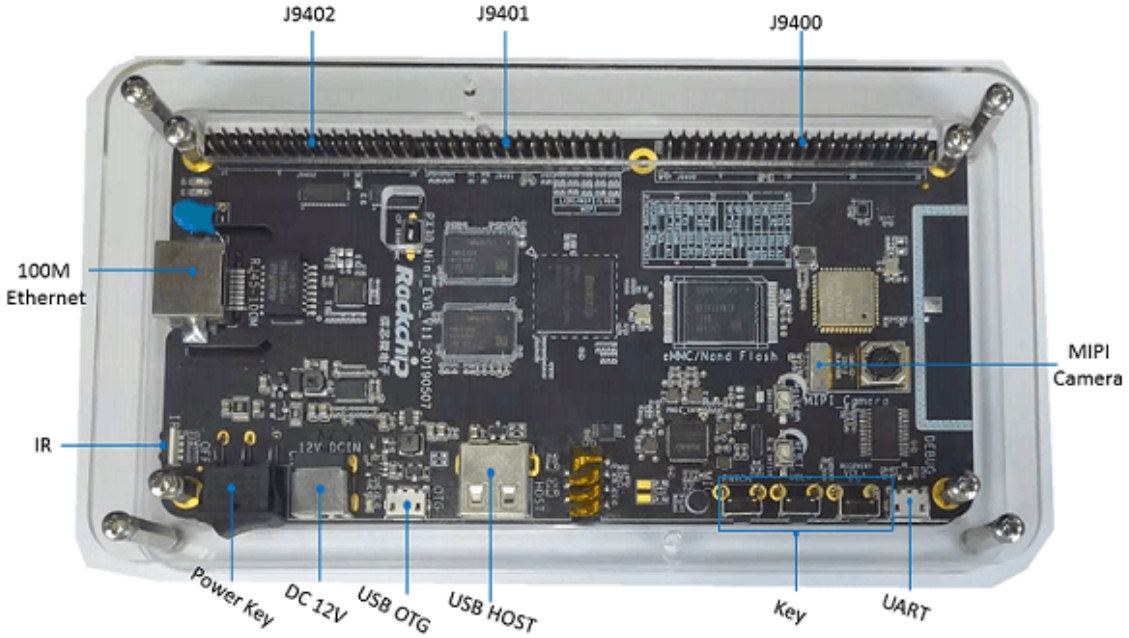
上面 Kernel、U-Boot、Recovery、Rootfs 各个部分的编译后，进入工程目录根目录执行以下命令自动完成所有固件打包到 rockdev 目录下：

固件生成：

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

5. 刷机说明

PX30-EVB 接口分布图如下：

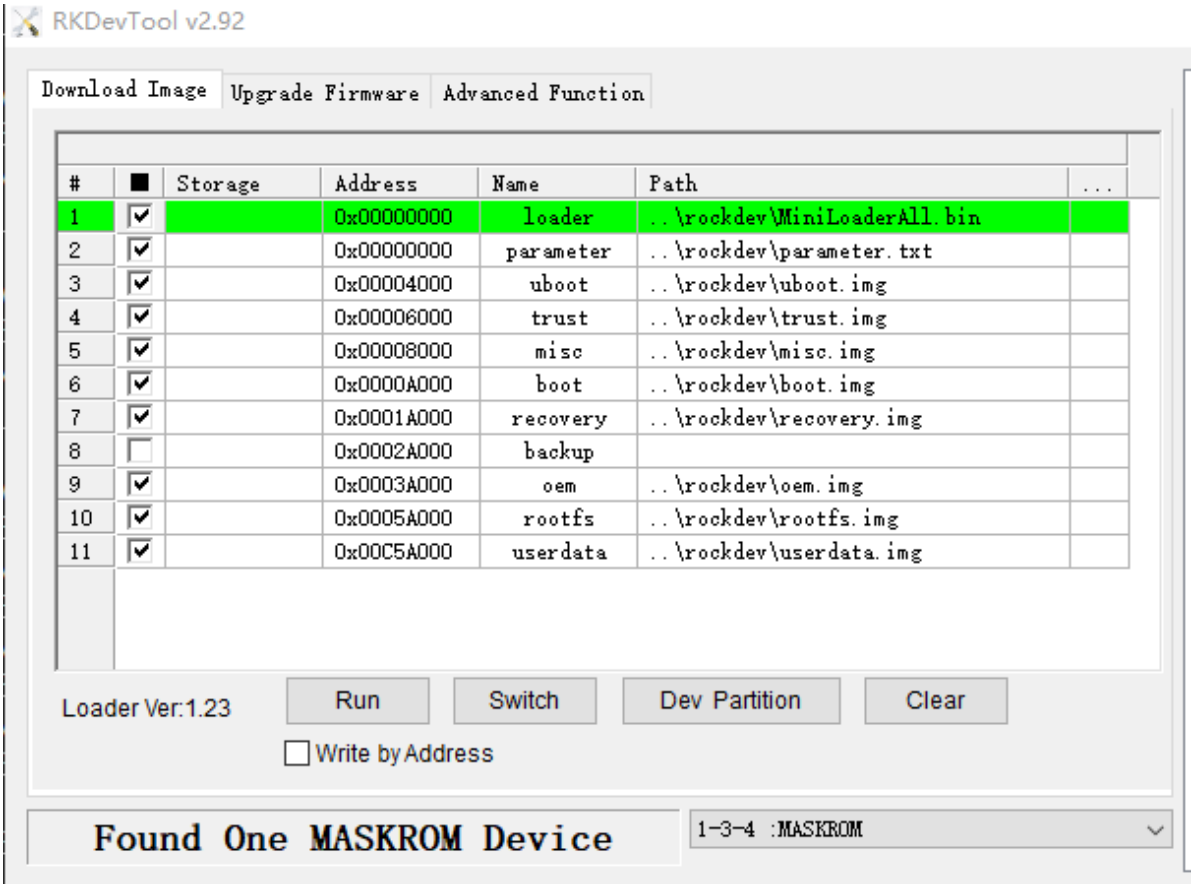


5.1 Windows 刷机说明

SDK 提供 Windows 烧写工具(工具版本需要 V2.92 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
```

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 rockdev/MiniLoaderAll.bin -noreset
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
```

5.3 系统分区说明

默认分区说明（下面是 PX30 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，后续跟 Android 一样作为 recovery 的 backup 使用。
- oem 分区：给厂家使用，存放厂家的 APP 或数据。挂载在 /oem 目录。
- rootfs 分区：供 buildroot、debian 或 yocto 编出来的 rootfs.img。
- 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. PX30 SDK 固件

- 百度云网盘

  Buildroot

  Debian

  Yocto
- 微软 OneDriver

  Buildroot

  Debian

  Yocto