Rockchip RK3399Pro Linux SDK 发布说明

文档标识：RK-FB-CS-009

发布版本：V1.4.0

日期：2020-10-10

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

文档主要介绍 Rockchip RK3399Pro Linux SDK发布说明，旨在帮助工程师更快上手RK3399Pro Linux SDK开发及相关调试方法。

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 9**	**Debian 10**	**Yocto**
RK3399Pro	Y	Y	Y	Y

修订记录

日期	版本	作者	修改说明
2019-02-17	V0.0.1	Caesar Wang	初始 Beta 版本
2019-03-21	V0.0.2	Caesar Wang	修改5.1.3中./mkfirmware.sh 生成image 的方法。 更改8章节中 rknn_demo 用例，添加Debian 的说明。 更改8章节中SDK固件升级到v0.0.2。
2019-06-06	V1.0.0	Caesar Wang	正式发布版本，添加 NPU 相关说明。 增加 Yocto 的编译说明，增加 github 下载说明。
2019-06-21	V1.0.1	Caesar Wang	修改软件开发指南名字
2019-10-14	V1.1.2	Caesar Wang	修改 Debian 编译说明
2019-10-23	V1.1.3	Caesar Wang	支持 RK3399Pro EVB V13 编译
2019-12-03	V1.2.0	Caesar Wang	章节 3、4、6、7、8、9、10 内容更改
2020-03-24	V1.3.0	Caesar Wang	增加 RK3399Pro V14 的支持
2020-07-22	V1.3.1	Ruby Zhang	更新公司名称，文件名以及文档格式
2020-08-06	V1.3.2	Caesar Wang	Debian 10 的支持
2020-08-13	V1.3.3	Caesar Wang	rknpu版本更新搭配1.3.4，目录结构调整和固件升级
2020-10-10	V1.4.0	Caesar Wang	rknpu版本更新搭配1.4.0，目录结构调整

目录

1. 概述

本 SDK支持三个系统分别基于 Buildroot 2018.02-rc3、Yocto Thud 3.0、Debian9 和 Debian 10 上开发，内核基于 Kernel 4.4，引导基于 U-boot v2017.09，适用于 RK3399Pro EVB 开发板及基于此开发板进行二次开发的所有 Linux 产品。
本 SDK 支持 NPU TensorFlow/Caffe 模型、VPU 硬解码、GPU 3D、Wayland 显示、QT 等功能。具体功能调试和接口说明，请阅读工程目录 docs/ 下文档。

2. 主要支持功能

功能	模块名
数据通信	Wi-Fi、以太网卡、USB、SD 卡、PCI-e 接口
应用程序	多媒体播放、设置、浏览器、文件管理

3. SDK 获取说明

SDK 通过瑞芯微代码服务器对外发布获取。其编译开发环境，参考第 7 节 SDK编译说明。

3.1 RK3399Pro Linux 通用软件包获取方法

3.1.1 通过代码服务器下载

获取 RK3399Pro Linux 软件包，需要有一个帐户访问 Rockchip 提供的源代码仓库。客户向瑞芯微技术窗口申请 SDK，同步提供 SSH公钥进行服务器认证授权，获得授权后即可同步代码。关于瑞芯微代码服务器 SSH公钥授权，请参考第 10 节 SSH 公钥操作说明。

RK3399Pro_Linux_SDK 下载命令如下：

```
repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \
ssh://git@www.rockchip.com.cn/linux/rk/platform/manifests -b linux -m \
rk3399pro_linux_release.xml
```

repo 是 google 用 Python 脚本写的调用 git 的一个脚本，主要是用来下载、管理项目的软件仓库，其下载地址如下：

```
git clone ssh://git@www.rockchip.com.cn/repo/rk/tools/repo
```

3.1.2 通过本地压缩包解压获取

为方便客户快速获取 SDK 源码，瑞芯微技术窗口通常会提供对应版本的 SDK 初始压缩包，开发者可以通过这种方式，获得 SDK 代码的初始压缩包，该压缩包解压得到的源码，进行同步后与通过 repo 下载的源码是一致的。
以 rk3399pro_linux_sdk_release_v1.4.0_20201010.tgz 为例，拷贝到该初始化包后，通过如下命令可检出源码：

```
mkdir rk3399pro
tar xvf rk3399pro_linux_sdk_release_v1.4.0_20201010.tgz -C rk3399pro
cd rk3399pro
.repo/repo/repo sync -l
.repo/repo/repo sync -c
```

后续开发者可根据 FAE 窗口定期发布的更新说明，通过 ".repo/repo/repo sync -c" 命令同步更新。

4. 软件开发指南

4.1 NPU 开发工具

本 SDK NPU 开发工具如下：

RKNN_DEMO (MobileNet SSD)：
RKNN 的 Demo 请参考目录 external/rknn_demo/，相关操作说明详见工程目录 docs/Linux/ApplicationNote/Rockchip_Developer_Guide_Linux_RKNN_Demo_CN.pdf。

RKNN-TOOLKIT：
开发工具在 external/rknn-toolkit 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：

```
├── changelog.txt
├── Rockchip_Developer_Guide_RKNN_Toolkit_Custom_OP_V1.4.0_CN.pdf
├── Rockchip_Developer_Guide_RKNN_Toolkit_Custom_OP_V1.4.0_EN.pdf
├── Rockchip_Quick_Start_RKNN_Toolkit_V1.4.0_CN.pdf
├── Rockchip_Quick_Start_RKNN_Toolkit_V1.4.0_EN.pdf
├── Rockchip_Trouble_Shooting_RKNN_Toolkit_V1.4.0_CN.pdf
├── Rockchip_Trouble_Shooting_RKNN_Toolkit_V1.4.0_EN.pdf
├── Rockchip_User_Guide_RKNN_Toolkit_Lite_V1.4.0_CN.pdf
├── Rockchip_User_Guide_RKNN_Toolkit_Lite_V1.4.0_EN.pdf
├── Rockchip_User_Guide_RKNN_Toolkit_V1.4.0_CN.pdf
├── Rockchip_User_Guide_RKNN_Toolkit_V1.4.0_EN.pdf
├── Rockchip_User_Guide_RKNN_Toolkit_Visualization_V1.4.0_CN.pdf
└── Rockchip_User_Guide_RKNN_Toolkit_Visualization_V1.4.0_EN.pdf
```

RKNN-DRIVER：
RKNN DRIVER 开发内容在工程目录 external/rknpu 下。

RKNPUTools：
RKNN API的开发使用在工程目录 external/RKNPUTools 下。

NPU 软件启动说明：
RK3399Pro的 NPU 软件启动说明，请参考工程目录 docs/RK3399PRO/Rockchip_RK3399Pro_Developer_Guide_Linux_NPU_CN.pdf。

4.2 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$ ls -l -h rk3399pro_linux_release.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 rk3399pro_linux_v0.01/RK3399PRO_Linux_SDK_Release_Note.md
```

或者参考工程目录：

```
<SDK>/docs/RK3399PRO/RK3399PRO_Linux_SDK_Release_Note.md
```

5. 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：

```
<SDK>/docs/RK3399PRO/Rockchip_RK3399Pro_User_Guide_Hardware_CN.pdf
```

6. 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 9 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步和预备文件。
- 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。
- distro：基于 Debian 10 开发的根文件系统。
- 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 4.4 开发的代码。
- npu：存放 NPU 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 Thud 3.0 开发的根文件系统。

7. SDK 编译说明

7.1 SDK依赖包安装

本 SDK 开发环境是在 Ubuntu 系统上开发测试。我们推荐使用 Ubuntu 18.04 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 18.04 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
```

建议使用 Ubuntu18.04 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。

7.2 SDK板级配置

进入工程/device/rockchip/rk3399pro目录：

板级配置	说明
BoardConfig-rk3399pro_evb_v10-usb.mk	适用于 RK3399Pro V10 开发板
BoardConfig-rk3399pro_evb_v11_v12-usb.mk	适用于 RK3399Pro V11、V12 开发板
BoardConfig_rk3399pro_evb_v13_pcie.mk	适用于 RK3399Pro V13 开发板
BoardConfig_rk3399pro_evb_v14-combine.mk	适用于 RK3399Pro V14 开发板
BoardConfig-rk3399pro_evb_lpd4_v11_v12-usb.mk	适用于 RK3399Pro LPDDR4 开发板
BoardConfig-rk3399pro_npu-pcie.mk	适用于硬件PCIe方式链接NPU
BoardConfig-rk3399pro_npu-usb.mk	适用于硬件USB3.0方式链接NPU

方法1
`./build.sh` 后面加上板级配置文件, 例如：

选择**RK3399Pro V10 开发板**的板级配置：

```
./build.sh device/rockchip/rk3399pro/BoardConfig-rk3399pro_evb_v10-usb.mk
```

选择**RK3399Pro V11/V12 开发板**的板级配置：

```
./build.sh device/rockchip/rk3399pro/BoardConfig-rk3399pro_evb_v11_v12-usb.mk
```

选择**RK3399Pro V13 开发板**的板级配置：

```
./build.sh device/rockchip/rk3399pro/BoardConfig_rk3399pro_evb_v13_pcie.mk
```

选择**RK3399Pro V14 开发板**的板级配置：

```
./build.sh device/rockchip/rk3399pro/BoardConfig-rk3399pro_npu-pcie.mk
```

选择**RK3399Pro LPDDR4 开发板**的板级配置：

```
./build.sh device/rockchip/rk3399pro/BoardConfig-rk3399pro_evb_lpd4_v11_v12-
usb.mk
```

选择**硬件PCIe方式的NPU**的板级配置：

```
./build.sh device/rockchip/rk3399pro/BoardConfig-rk3399pro_npu-pcie.mk
```

选择**硬件USB3.0方式的NPU**的板级配置：

```
./build.sh device/rockchip/rk3399pro/BoardConfig-rk3399pro_npu-usb.mk
```

方法2

```
rk3399pro$ ./build.sh lunch
processing option: lunch

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

0. default BoardConfig.mk
1. BoardConfig-rk3399pro_evb_lpd4_v11_v12-usb.mk
2. BoardConfig-rk3399pro_evb_v10-usb.mk
3. BoardConfig-rk3399pro_evb_v11_v12-usb.mk
4. BoardConfig-rk3399pro_npu-pcie.mk
5. BoardConfig-rk3399pro_npu-usb.mk
6. BoardConfig.mk
7. BoardConfig_rk3399pro_evb_v13_pcie.mk
8. BoardConfig_rk3399pro_evb_v14-combine.mk
Which would you like? [0]:
...
```

7.3 查看编译命令

在根目录执行命令：./build.sh -h|help

```
rk3399pro$ ./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 debian9 stretch rootfs
distro              -build debian10 buster 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
save                -save images, patches, commands used to debug
allsave             -build all & firmware & updateimg & save

Default option is 'allsave'.
```

查看部分模块详细编译命令，例如：./build.sh -h kernel

```
rk3399pro$ ./build.sh -h kernel
###Current SDK Default [ kernel ] Build Command###
cd kernel
make ARCH=arm64 rockchip_linux_defconfig
make ARCH=arm64 rk3399pro-evb-v14-linux.img -j12
```

RK3399Pro 每次上电启动后会加载 NPU 固件。 默认 NPU 固件都是预编好放到
根文件系统的 /usr/share/npu_fw 目录下, NPU 固件烧写及启动方式请参考文档

```
<SDK>/docs/RK3399PRO/Rockchip_RK3399Pro_Developer_Guide_Linux_NPU_CN.pdf。
```

下面分别对 NPU 和 RK3399Pro 固件编译方法进行介绍：

7.4 NPU 编译说明

7.4.1 全自动编译

进入工程目录根目录执行以下命令自动完成所有的编译：

RK3399Pro EVB V10/V11/V12 开发板：

```
cd npu
./build.sh device/rockchip/rk3399pro/BoardConfig-rk3399pro_npu-usb.mk
cd kernel
git checkout remotes/rk/stable-4.4-rk3399pro_npu-linux
cd -
./build.sh uboot
./build.sh kernel
./build.sh ramboot
./mkfirmware.sh
```

RK3399Pro EVB V13/V14 开发板：

```
cd npu
./build.sh device/rockchip/rk3399pro/BoardConfig-rk3399pro_npu-pcie.mk
cd kernel
git checkout remotes/rk/stable-4.4-rk3399pro_npu-pcie-linux
cd -
./build.sh uboot
./build.sh kernel
./build.sh ramboot
./mkfirmware.sh
```

在 rockdev 目录下生成 boot.img，uboot.img，trust.img，MiniLoaderAll.bin
注意：rockdev下生成 NPU 固件需要存放在 Rootfs 指定位置/usr/share/npu_fw。

7.4.2 各模块编译及打包

7.4.2.1 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
```

7.4.2.2 Kernel编译

```
### Kernel编译命令
./build.sh kernel

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

7.4.2.3 Rootfs编译

```
### Rootfs编译命令
./build.sh ramboot

### 查看Rootfs详细编译命令
./build.sh -h ramboot
```

Buildroot的package编译方法：
注：SDK根目录app和external下的工程都是buildroot的package包，编译方法相同。

```
### 1. 先查看Board Config对应的rootfs是哪个配置
./build.sh -h rootfs
###Current SDK Default [ rootfs ] Build Command###
source envsetup.sh rockchip_rk3399pro-npu-multi-cam
make

### 2. source buildroot对应的defconfig
source envsetup.sh rockchip_rk3399pro-npu-multi-cam

### 3. 查看对应模块的makefile文件名
### 例如：buildroot/package/rockchip/rknpu/rknpu.mk
make rknpu-dirclean
make rknpu-rebuild
```

7.4.2.4 固件打包

固件打包命令：`./mkfirmware.sh`

固件目录：rockdev

7.5 RK3399Pro 编译说明

7.5.1 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目前可设定四个类型：buildroot、debian、distro 和 yocto 。
其中debian是编译Debian9系统，distro是编译debian10系统。

比如需要 debain 可以通过以下命令进行生成：

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

7.5.2 各模块编译及打包

7.5.2.1 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
```

7.5.2.2 Kernel编译

```
### Kernel编译命令
./build.sh kernel

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

7.5.2.3 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

7.5.2.4 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_芯片命令/images下生成 rootfs.ext4。

7.5.2.4.1 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。交叉编译工具位于 buildroot/output/rockchip_rk3399pro_combine/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_rk3399pro_combine/host/usr/bin
./aarch64-linux-gcc --version
```

此时会打印如下信息：

```
aarch64-linux-gcc.br_real (Buildroot 2018.02-rc3-01797-gcd6c508) 6.5.0
```

7.5.2.4.2 Buildroot 中模块编译

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_rk3399pro/build/qlayer-1.0
```

7.5.2.5 Debian 9 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=stretch TARGET=desktop ARCH=arm64 ./mk-base-debian.sh
```

编译完成会在 debian/ 目录下生成：linaro-stretch-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/stretch-desktop-armhf/chroot/test-dev-null:
Permission denied E: Cannot install into target
...
mounted with noexec or nodev
```

解决方法：

```
mount -o remount,exec,dev xxx
(其中xxx 是工程目录路径，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。

- 编译 Base Debian 由于访问国外网站，国内网络会经常出现下载失败的情况:

Debian 9 使用 live build,镜像源改为国内可以这样配置:

```
+++ b/ubuntu-build-service/stretch-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 "http://mirrors.163.com/debian" \
+ --mirror-chroot "http://mirrors.163.com/debian" \
+ --mirror-chroot-security "http://mirrors.163.com/debian-security" \
+ --mirror-binary "http://mirrors.163.com/debian" \
+ --mirror-binary-security "http://mirrors.163.com/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-stretch.sh
```

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

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

此时会生成 linaro-rootfs.img。

7.5.2.6 Debian 10 编译

```
./build.sh distro
```

或进入 distro/ 目录：

```
cd distro/ && make ARCH=arm64 rk3399pro_defconfig && ./make.sh
```

编译后在 distro/output/images/ 目录下生成 rootfs.ext4 。
注意： 目前Debian 10 QT的编译还依赖 Buildroot qmake的编译， 所以编译 Debian 10 前，请先编译 Buildroot。
更多 Debian 10的介绍参考文档：

```
<SDK>/docs/Linux/ApplicationNote/Rockchip_Debian10_Developer_Guide_CN.pdf
```

7.5.2.7 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：
RK3399Pro 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。

7.5.2.8 固件的打包

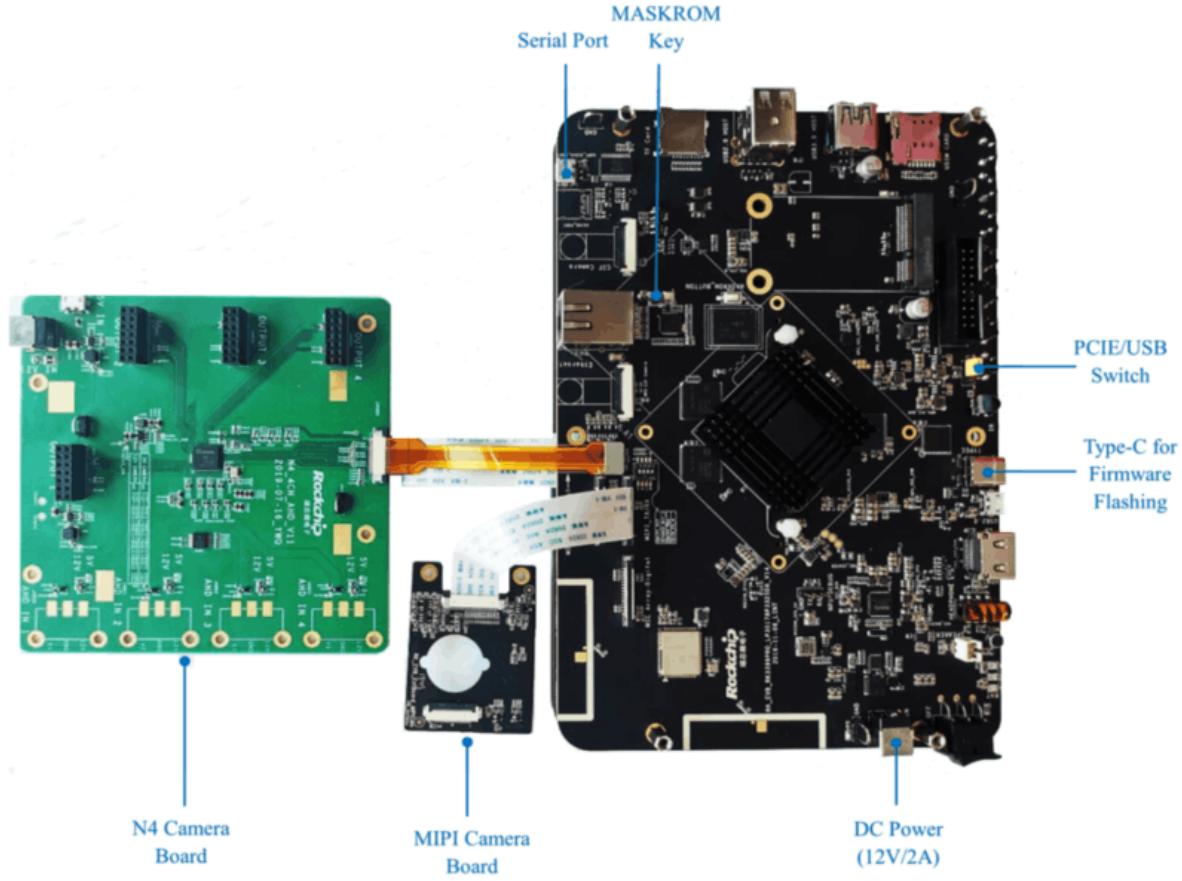
上面 Kernel/U-Boot/Recovery/Rootfs 各个部分的编译后，进入工程目录根目录执行以下命令自动完成所有固件打包到 rockdev 目录下：

固件生成:

```
./mkfirmware.sh
```

8. 刷机说明

目前 RK3399Pro EVB 有 V10/V11/V12/V13/V14，5个版本，绿色板子是 V10 版本，黑色板子是 V11/V12/V13/V14 版本。板子功能位置是一样，下面以 RK3399Pro EVB V14 板子做介绍，如下图说明。

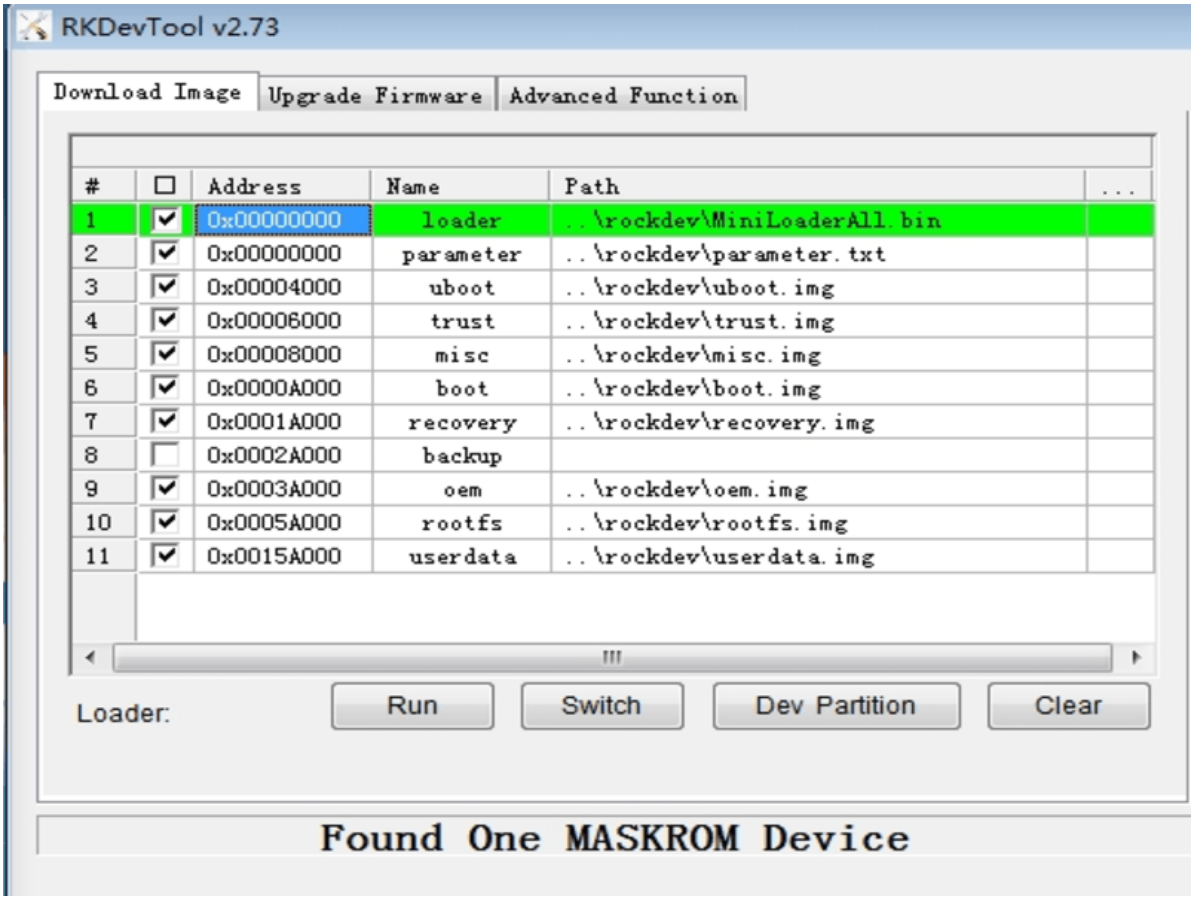


8.1 Windows 刷机说明

SDK 提供 Windows 烧写工具(工具版本需要 V2.55 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： WIndow 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_v4.91.zip
```

8.2 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 V1.33 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
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
```

8.3 系统分区说明

默认分区说明 (下面是 RK3399Pro 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	96M	recovery
6	303104	368639	32M	bakcup
7	368640	499711	64M	oem
8	499712	13082623	6144M	rootfs
9	12082624	30535646	8521M	userdata

- uboot 分区：存放 uboot 编译出来的 uboot.img。
- trust 分区：存放 uboot 编译出来的 trust.img。
- misc 分区：存放 misc.img，给 recovery 使用。
- boot 分区：存放 kernel 编译出来的 boot.img。
- recovery 分区：存放 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：存放 buildroot、debian 编出来的 rootfs.img。
- userdata 分区：存放 APP 临时生成文件或给最终用户使用，挂载在 /userdata 目录下。

9. RK3399Pro SDK 固件及简单 Demo 测试

9.1 RK3399Pro SDK 固件

RK3399PRO_LINUX_SDK_V1.4.0_20201010 固件下载链接如下
（包含 Buildroot/Debian 9/Debian10/Yocto 的固件）

- 百度云网盘

Buildroot:
V10(绿色)板子
V11/V12(黑色)板子
V13(黑色)板子
V14(黑色)板子

Debian 9:
Debian9 rootfs

Debian 10:
Debian10 pcie rootfs
Debian10 usb rootfs

Yocto:
Yocto rootfs

- 微软 OneDriver

Buildroot:
V10(绿色)板子
V11/V12(黑色)板子
V13(黑色)板子
V14(黑色)板子

Debian 9:
Debian9 rootfs

Debian 10:
Debian10 pcie rootfs
Debian10 usb rootfs

Yocto:
Yocto rootfs

9.2 RKNN_DEMO 测试

首先插入 usb camera，
然后在 Buildroot 系统中运行 rknn_demo 或 Debian 系统中运行 test_rknn_demo.sh。
具体参考工程文档/docs/Linux/ApplicationNote/Rockchip_Developer_Guide_Linux_RKNN_Demo_CN/EN.pdf。在 Buildroot 中运行结果如下：

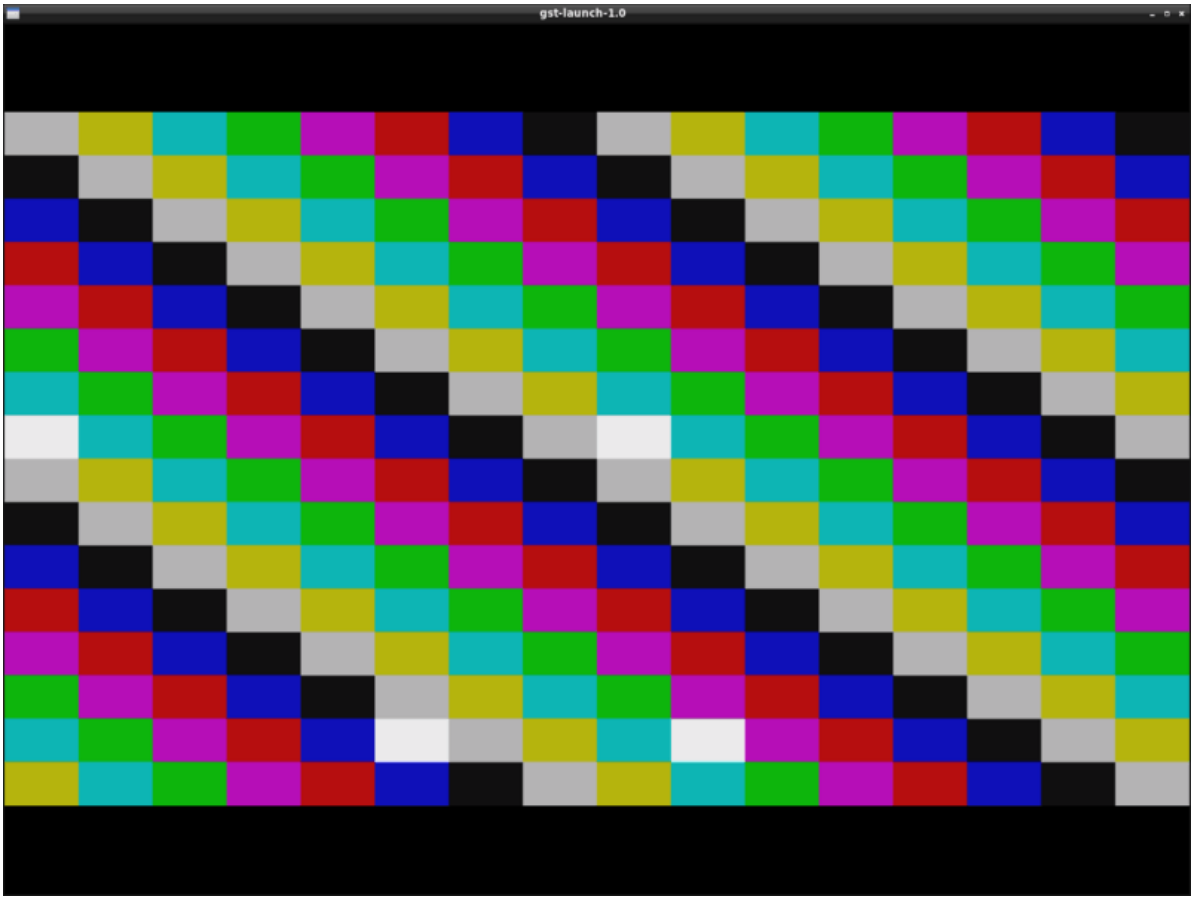
```
[root@rk3399pro:/]# rknn_demo
librga:RGA_GET_VERSION:3.02,3.020000
ctx=0x2a64ac20,ctx->rgaFd=3
Rga built version:version:+2017-09-28 10:12:42
success build
set plane zpos = 3 (0~3)size = 12582988, g_bo.size = 13271040
size = 12582988, cur_bo->size = 6635520
size = 12582988, cur_bo->size = 6635520
size = 12582988, cur_bo->size = 6635520

...
get device /dev/video10
Please configure uvc...
read model:/usr/share/rknn_demo/mobilenet_ssd.rknn, len:32002449
set plane zpos = 3 (0~3)D RKNNAPI: ==========================================
D RKNNAPI: RKNN VERSION:
D RKNNAPI:   API: 1.3.3 (f20f0bd build: 2020-05-14 14:14:51)
D RKNNAPI:   DRV: 1.3.4 (399a00a build: 2020-07-24 14:09:19)
D RKNNAPI: ==========================================
```

最终在屏幕显示效果如下：

9.3 N4 Camera 测试

首先接入 N4 Camera 模块（需要接入 12V 电源）， 然后在 Buildroot 系统中直接打开 Camera 应用或在 Debian 系统中运行 test_camera-rkisp1.sh。
在 Buildroot中运行结果如下：(没有接入具体Camera Sensor)



10. SSH 公钥操作说明

请根据《Rockchip SDK 申请及同步指南》文档说明操作，生成 SSH 公钥，发邮件至fae@rock-chips.com，申请开通 SDK 代码。
该文档会在申请开通权限流程中，释放给客户使用。

10.1 多台机器使用相同 SSH 公钥

在不同机器使用，可以将你的 SSH 私钥文件 id_rsa 拷贝到要使用的机器的 “~/.ssh/id_rsa” 即可。
在使用错误的私钥会出现如下提示，请注意替换成正确的私钥

```
~/tmp$ git clone git@172.16.10.211:rk292x/mid/4.1.1_r1
Initialized empty Git repository in /home/cody/tmp/4.1.1_r1/.git/
The authenticity of host '172.16.10.211 (172.16.10.211)' can't be established.
RSA key fingerprint is fe:36:dd:30:bb:83:73:e1:0b:df:90:e2:73:e4:61:46.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.10.211' (RSA) to the list of known hosts.
git@172.16.10.211's password:
```

添加正确的私钥后，就可以使用 git 克隆代码，如下图。

```
~$ cd tmp/
~/tmp$ git clone git@172.16.10.211:rk292x/mid/4.1.1_r1
Initialized empty Git repository in /home/cody/tmp/4.1.1_r1/.git/
The authenticity of host '172.16.10.211 (172.16.10.211)' can't be established.
RSA key fingerprint is fe:36:dd:30:bb:83:73:e1:0b:df:90:e2:73:e4:61:46.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.10.211' (RSA) to the list of known hosts.
remote: Counting objects: 237923, done.
remote: Compressing objects: 100% (168382/168382), done.
Receiving objects:   9% (21570/237923), 61.52 MiB | 11.14 MiB/s
```

添加 ssh 私钥可能出现如下提示错误。

```
Agent admitted failture to sign using the key
```

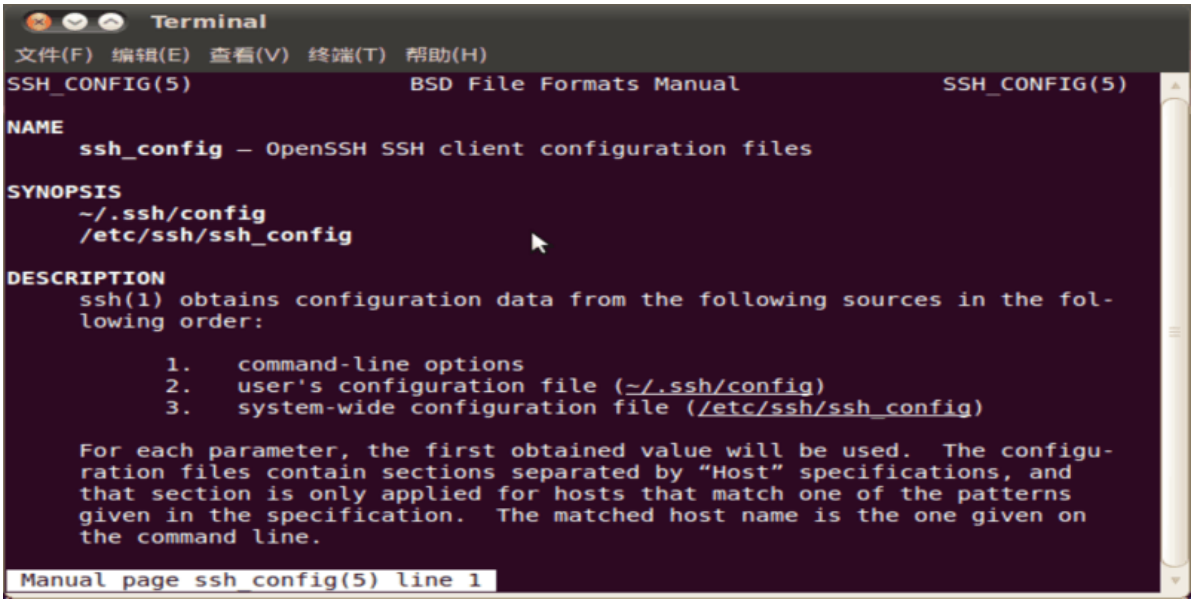
在 console 输入如下命令即可解决。

```
ssh-add ~/.ssh/id_rsa
```

10.2 一台机器切换不同 SSH 公钥

可以参考 ssh_config 文档配置 SSH。

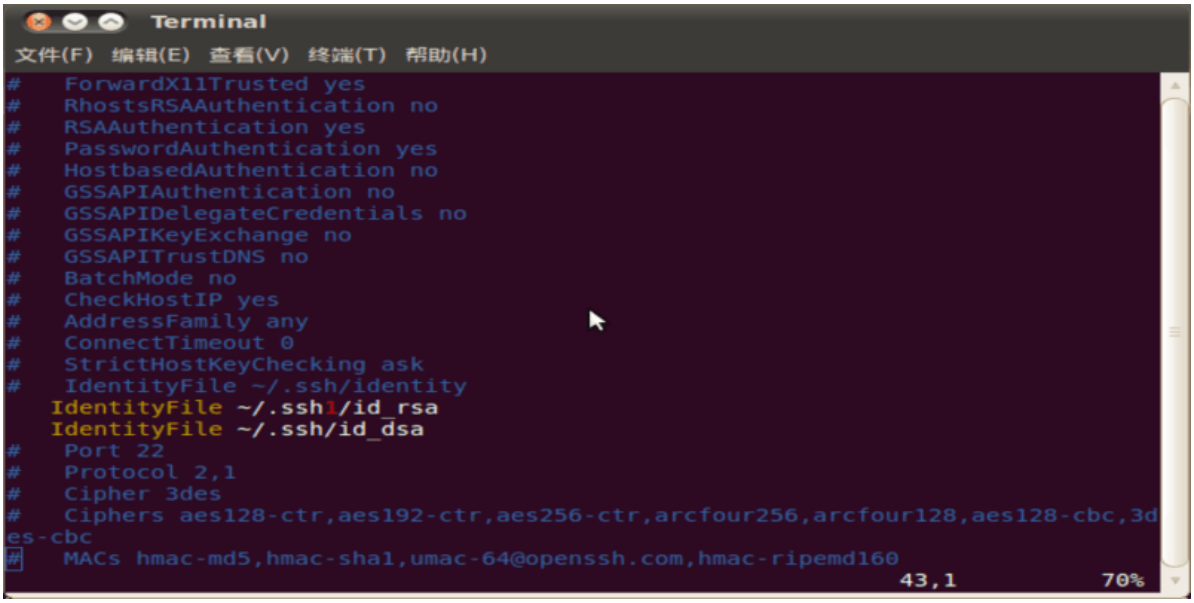
```
~$ man ssh_config
```



通过如下命令，配置当前用户的 SSH 配置。

```
~$ cp /etc/ssh/ssh_config ~/.ssh/config
~$ vi .ssh/config
```

如图，将 SSH 使用另一个目录的文件 “~/.ssh1/id_rsa” 作为认证私钥。通过这种方法，可以切换不同的的密钥。



10.3 密钥权限管理

服务器可以实时监控某个 key 的下载次数、IP 等信息，如果发现异常将禁用相应的 key 的下载权限。

请妥善保管私钥文件。并不要二次授权与第三方使用。

10.4 参考文档

更多详细说明，可参考文档/docs/Others/Rockchip_User_Guide_SDK_Application_And_Synchronization_CN.pdf。