RK3399Pro Linux SDK Release Notes

ID: RK-FB-CS-009

Release Version: V1.4.0

Release Date: 2020-10-10

Security Level: □Top-Secret □Secret □Internal ■Public

DISCLAIMER

Trademark Statement

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: fae@rock-chips.com

Preface

Overview

This document presents an overview of Rockchip RK3399Pro Linux SDK release notes, aiming to help engineers get started with RK3399Pro Linux SDK and related debugging methods faster.

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers

Software development engineers

Chipset	Buildroot	Debian 9	Debian 10	Yocto
RK3399Pro	Y	Y	N	Y

Revision History

Date	Version	Author	Revision History
2019-02-17	V0.0.1	Caesar Wang	Initial Beta version
2019-03-21	V0.0.2	Caesar Wang	Modify method of using ./mkfirmware.sh to generate image in chapter 5.1.3 Change the description of adding Debian to rknn_demo in chapter 8. Change the SDK firmware to v0.02 in chapter 8
2019-06-06	V1.0.0	Caesar Wang	Release version Add NPU related instructions Add Yocto compilation instructions Add github download instructions。
2019-06-21	V1.0.1	Caesar Wang	Update software development guide。
2019-10-14	V1.1.2	Caesar Wang	Update Debian build note
2019-10-23	V1.1.3	Caesar Wang	Support RK3399Pro EVB V13
2019-12-03	V1.2.0	Caesar Wang	Update chapter 3,4,6,7,8,9,10
2020-03-24	V1.3.0	Caesar Wang	Add RK3399Pro EVB V14 support
2020-07-22	V1.3.1	Ruby Zhang	Update the company name, the format and the file name of the document
2020-08-06	V1.3.2	Caesar Wang	Support Debian 10
2020-08-13	V1.3.3	Caesar Wang	Upgrade rknpu to 1.3.4, and update the directory structure and Firmware upgrade
2020-10-10	V1.4.0	Caesar Wang	Upgrade rknpu to 1.4.0, and update the directory structure

Contents

1. Overview

This SDK is based on 3 Linux systems: Buildroot 2018.02-rc3, Yocto Thud 3.0, Debian 9, Debian10, with Kernel 4.4 and U-boot v2017.09. It is suitable to the development of RK3399Pro EVB and all other Linux products developed based on it.
This SDK supports NPU TensorFlow/Caffe model, VPU hardware decoding, GPU 3D, Wayland display, QT and other functions. For detailed function debugging and interface introduction, please refer to related documents under the docs/ directory in the project.

2. Main Functions

Functions	Module Names
Data Communication	Wi-Fi, Ethernet Card, USB, SDCARD, PCI-e
Application	Multimedia playback, settings, browser, file management

3. How to Get the SDK

The SDK is released by Rockchip server. Please refer to Chapter 7 SDK Compilation Introduciton to build a development environment.

3.1 General RK3399Pro Linux SDK Obtain

3.1.1 Get Source Code from Rockchip Code Server

To get RK3399Pro Linux software package, customers need an account to access the source code repository provided by Rockchip. In order to be able to obtain code synchronization, please provide SSH public key for server authentication and authorization when apply for SDK from Rockchip technical window. About Rockchip server SSH public key authorization, please refer to Chapter 10 SSH Public Key Operation Introduction.

RK3399Pro_Linux_SDK download command is as follows：

```
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, a tool built on Python script by Google to help manage git repositories, is mainly used to download and manage software repository of projects. The download address is as follows:

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

3.1.2 Get Source Code from Local Compression Package

For quick access to SDK source code, Rockchip Technical Window usually provides corresponding version of SDK initial compression package. In this way, developers can get SDK source code through decompressing the initial compression package, which is the same as the one downloaded by repo.
Take rk3399pro_linux_sdk_release_v1.4.0_20201010.tgz as an example. After geting a initialization package, you can get source code by running the following command:

```
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
```

Developers can update via `.repo/repo/repo sync -c` command according to update instructions that are regularly released by FAE window.

4. Software Development Guide

4.1 NPU Development Tool

The SDK NPU development tool includes following items:

RKNN_DEMO (MobileNet SSD)：
Please refer to the directory “external/rknn_demo/” for RKNN Demo, and refer to the document in the project directory docs/Linux/ApplicationNote/Rockchip_Developer_Guide_Linux_RKNN_Demo_CN.pdf for detailed operation introduction.

RKNN-TOOLKIT：
Development tools are in project directory “external/rknn-toolkit”. Which is used for model conversion, model reasoning, model performance evaluation functions, etc. Please refer to documents in the docs/ directory for details.

```
├── 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 development materials are in the project directory “external/rknpu”.

RKNPUTools:
RKNN API development materials are in the project directory “external/NRKNPUTools”.

NPU software startup instructions:
Please refer to the document in the project directory “docs/RK3399PRO/Rockchip_RK3399Pro_Developer_Guide_Linux_NPU_CN.pdf” for RK3399Pro NPU software setup instructions.

4.2 Software Update History

Software release version upgrade can be checked through project xml file by the following command:

```
.repo/manifests$ ls -l -h rk3399pro_linux_release.xml
```

Software release version updated information can be checked through the project text file by the following command:

```
.repo/manifests$ cat rk3399pro_linux_v0.01/RK3399PRO_Linux_SDK_Release_Note.md
```

Or refer to the project directory:

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

5. Hardware Development Guide

Please refer to user guides in the project directory for hardware development :

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

6. SDK Project Directory Introduction

There are buildroot, debian, recovery, app, kernel, u-boot, device, docs, external and other directories in the project directory. Each directory or its sub-directories will correspond to a git project, and the commit should be done in the respective directory.

- app: store application APPs like qcamera/qfm/qplayer/qseting and other applications.
- buildroot: root file system based on Buildroot (2018.02-rc3).
- debian: root file system based on Debian 9.
- device/rockchip: store board-level configuration for each chip and some scripts and prepared files for compiling and packaging firmware.
- docs: stores development guides, platform support lists, tool usage, Linux development guides, and so on.
- distro: a root file system based on Debian 10.
- IMAGE: stores compilation time, XML, patch and firmware directory for each compilation.
- external: stores some third-party libraries, including audio, video, network, recovery and so on.
- kernel: stores kernel4.4 development code.

- npu: store npu code.
- prebuilts: stores cross-compilation toolchain.
- rkbin: stores Rockchip Binary and tools.
- rockdev: stores compiled output firmware.
- tools: stores some commonly used tools under Linux and Windows system.
- u-boot: store U-Boot code developed based on v2017.09 version.
- yocto: stores the root file system developed based on Yocto Thud 3.0.

7. SDK Build Introduction

7.1 SDK Dependency Packages Installation

This SDK is developed and tested on Ubuntu system. We recommend using Ubuntu 18.04 for compilation. Other Linux versions may need to adjust the software package accordingly. In addition to the system requirements, there are other hardware and software requirements.
Hardware requirements: 64-bit system, hard disk space greater than 40G. If you do multiple builds, you will need more hard drive space

Software requirements: Ubuntu 18.04 system:

Please install software packages with below commands to setup SDK compiling environment:

```
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
```

It is recommended to use Ubuntu 18.04 system or higher version for development. If you encounter an error during compilation, you can check the error message and install the corresponding software packages.

7.2 SDK Board Level Configuration

Enter the project SDK/device/rockchip/rk3399pro directory:

Board level configuration	Note
BoardConfig-rk3399pro_evb_v10-usb.mk	Suitable for RK3399Pro V10 development board
BoardConfig-rk3399pro_evb_v11_v12-usb.mk	Suitable for RK3399Pro V11 and V12 development boards
BoardConfig_rk3399pro_evb_v13_pcie.mk	Suitable for RK3399Pro V13 development board
BoardConfig_rk3399pro_evb_v14-combine.mk	Suitable for RK3399Pro V14 development board
BoardConfig-rk3399pro_evb_lpd4_v11_v12-usb.mk	Suitable for RK3399Pro LPDDR4 development board
BoardConfig-rk3399pro_npu-pcie.mk	Suitable for linking NPU by hardware PCIe
BoardConfig-rk3399pro_npu-usb.mk	Suitable for linking NPU by hardware USB3.0

The first way:

Add board configuration file behind `/build.sh` , for example:

Select the board configuration of **RK3399Pro V10 development board**:

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

Select the board configuration of the **RK3399Pro V11 or V12 development board**:

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

Select the board-level configuration of the **RK3399Pro V13 development board**:

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

Select the board-level configuration of the **RK3399Pro V14 development board**:

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

Select the board configuration of the **RK3399Pro LPDDR4 development board**:

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

Select the board-level configuration of **NPU in the hardware PCIe mode**:

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

Select the board-level configuration of **NPU in the hardware USB3.0 mode**:

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

The second way:

```
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 Compilation Commands

Execute the command in the root directory: `./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'.
```

View detailed build commands for some modules, for example: `./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
```

NPU firmware will be uploaded when RK3399Pro power on. The default NPU firmware is pre-build into "/usr/share/npu_fw" directory of rootfs. For NPU firmware flashing and setup methods, please refer to the document :

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

It is going to introduce NPU and RK3399Pro firmware compiling methods next.

7.4 NPU Build Introduction

7.4.1 Full Automatic Build

Enter root directory of project directory and execute the following commands to automatically complete all build:

RK3399Pro EVB V10/V11/V12 development boards：

```
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 development boards：

```
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
```

After compiling, boot.img, uboot.img, trust.img, MiniLoaderAll.bin are generated in rockdev directory.

Note: the generated NPU firmware under rockdev should be placed in the specified directory of rootfs “/usr/share/npu_fw”.

7.4.2 Build and package each module

7.4.2.1 Uboot Build

```
### U-Boot build command
./build.sh uboot

### To view detailed U-Boot build command
./build.sh -h uboot
```

7.4.2.2 Kernel Build

```
### Kernel build command
./build.sh kernel

### To view detailed Kernel build command
./build.sh -h kernel
```

7.4.2.3 Rootfs Build

```
### Rootfs  build command
./build.sh ramboot

### To view detailed Rootfs build command
./build.sh -h ramboot
```

The way to build Buildroot package:

Note: The projects in the SDK root directory app and external are all Buildroot packages, they are in the same build way.

```
### 1. First check which configuration of the rootfs is corresponding to Board
Config
./build.sh -h rootfs
### Current SDK Default [ rootfs ] Build Command###
source envsetup.sh rockchip_rk3399pro-npu-multi-cam
make

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

### 3. Check the makefile file name of the corresponding module
### eg: buildroot/package/rockchip/rknpu/rknpu.mk
make rknpu-dirclean
make rknpu-rebuild
```

7.4.2.4 Firmware Package

Firmware Package command: `./mkfirmware.sh`

Firmware directory: rockdev

7.5 RK3399Pro Build Introduction

7.5.1 Automatic Build

Enter root directory of project directory and execute the following commands to automatically complete all buid:

```
./build.sh all # Only build module code(u-Boot, kernel, Rootfs, Recovery)
               # Need to execute ./mkfirmware.sh again for firmware package

./build.sh     # Base on ./build.sh all
               # 1. Add firmware package ./mkfirmware.sh
               # 2. update.img package
               # 3. Copy the firmware in the rockdev directory to the
IMAGE/***_RELEASE_TEST/IMAGES directory
               # 4. Save the patches of each module to the
IMAGE/***_RELEASE_TEST/PATCHES directory
               # Note: ./build.sh  and  ./build.sh allsave command are the same
```

It is Buildroot by default, you can specify rootfs by setting the environment variable RK_ROOTFS_SYSTEM. There are four types of system for RK_ROOTFS_SYSTEM: buildroot, Debian, distro and yocto. In which, debian is used to build Debian 9 system, distro is used to build debian10 system

For example, if you need debain, you can generate it with the following command:

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

7.5.2 Build and package each module

7.5.2.1 U-boot Build

```
### U-Boot build command
./build.sh uboot

### To view the detailed U-Boot build command
./build.sh -h uboot
```

7.5.2.2 Kernel Build

```
### Kernel build command
./build.sh kernel

### To view the detailed Kernel build command
./build.sh -h kernel
```

7.5.2.3 Recovery Build

```
### Recovery build command
./build.sh recovery

### To view the detailed Recovery build command
./build.sh -h recovery
```

Note: Recovery is a unnecessary function, some board configuration will not be set

7.5.2.4 Buildroot Build

Enter project root directory and run the following commands to automatically complete compiling and packaging of Rootfs.

```
./build.sh rootfs
```

After build, rootfs.ext4 is generated in Buildroot directory “output/rockchip_chipset/images”.

7.5.2.4.1 Buildroot Cross Compilation

If you need to build a single module or a third-party application, you need to setup the cross compilation environment. Cross compilation tool is located in “buildroot/output/rockchip_rk3399pro_combine/host/usr” directory. You need to set bin/ directory of tools and aarch64-buildroot-linux-gnu/bin/ directory to environment variables, and execute auto-configuration environment variable script in the top-level directory (only valid for current console):

```
source envsetup.sh
```

Enter the command to check:

```
cd buildroot/output/rockchip_rk3399pro_combine/host/usr/bin
./aarch64-linux-gcc --version
```

Then the following logs are printed:

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

7.5.2.4.2 Build Modules in Buildroot

For example, for the qplayer module, commonly used build commands are as follows:

- Build qplayer

```
SDK$make qplayer
```

- Rebuild qplayer

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

- delete qplayer

```
SDK$make qplayer-dirclean
or
SDK$rm -rf /buildroot/output/rockchip_rk3399pro/build/qlayer-1.0
```

7.5.2.5 Debian 9 Build

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

Or enter debian/ directory:

```
cd debian/
```

The following compilation and debian firmware generation, you can refer to "readme.md" in the current directory.

(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
```

Compile 64-bit Debian:

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

After compiling, linaro-stretch-alip-xxxxx-1.tar.gz (xxxxx is generated timestamp will be generated in debian/ directory.)

FAQ:
If you encounter the following problem during above compiling:

```
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
```

Solution：

```
mount -o remount,exec,dev xxx
(xxx is the project directory path, then rebuild)
```

In addition, if there are other compilation issues, please check firstly that the compiler system is not ext2/ext4.

- Building Base Debian need to access to foreign websites, it often fail to download in domestic networks.

Debian 9 uses live build, it can be configured like below to change the image source to domestic

```
+++ 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 \
```

If the package cannot be downloaded for other network reasons, a pre-build package is shared in Baidu Cloud Network Disk, put it in the current directory, and then do the next step directly.

(2) Building rk-debian rootfs

Compile 64-bit Debian:

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

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

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

Will generate linaro-rootfs.img.

7.5.2.6 Debian 10 Build

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

Or enter distro/directory:

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

After building, the rootfs.ext4 will be generated in the distro directory “distro/output/images/”.
Note: The current build of Debian10 Qt also depends on the build of Buildroot qmake, so please build Buildroot before building Debian10.

Please refer to the following document for more introductions about Debian10.

```
<SDK>/docs/Linux/ApplicationNote/Rockchip_Developer_Guide_Debian10_EN.pdf
```

7.5.2.7 Yocto Build

Enter project root directory and execute the following commands to automatically complete compiling and packaging Rootfs.

RK3399Pro EVB boards：

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

After compiling, rootfs.img is generated in yocto directory “/build/lastest”.

FAQ：

If you encounter the following problem during above compiling:

```
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.
```

Solution:

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

Or refer tosetup-locale-python3.The image generated after compiling is in “yocto/build/lastest/rootfs.img”. The default login username is root.

Please refer to Rockchip Wiki for more detailed information of Yocto.

7.5.2.8 Firmware Package

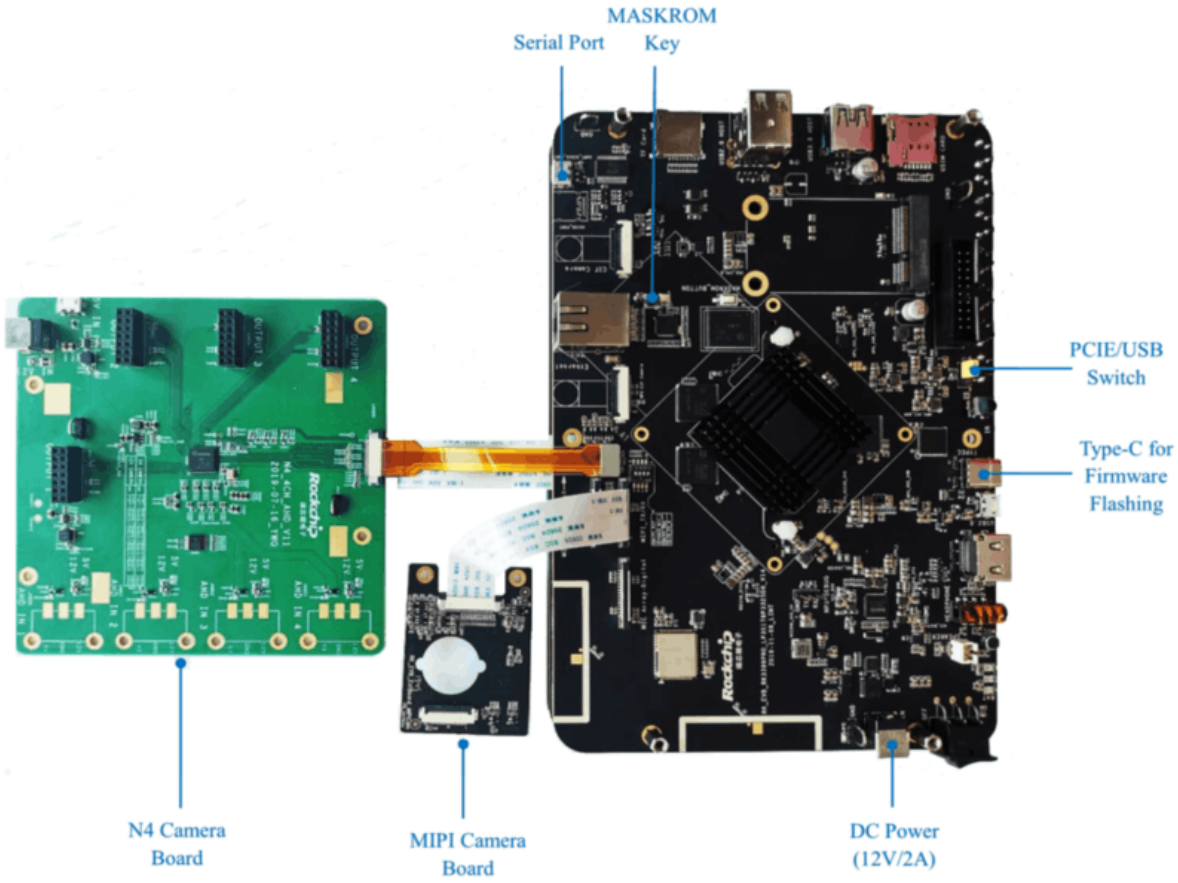
After compiling various parts of Kernel/U-Boot/Recovery/Rootfs above, enter root directory of project directory and run the following command to automatically complete all firmware packaged into rockdev directory:

Firmware generation:

```
./mkfirmware.sh
```

8. Upgrade Introduction

There are V10/V11/V12/V13/V14 Five versions of current RK3399Pro EVB, V10 version board is green, and V10/V11/V12/V13/V14 version board is black. Board function positions are the same. The following is the introduction of RK3399Pro EVB V14 board, as shown in the following figure

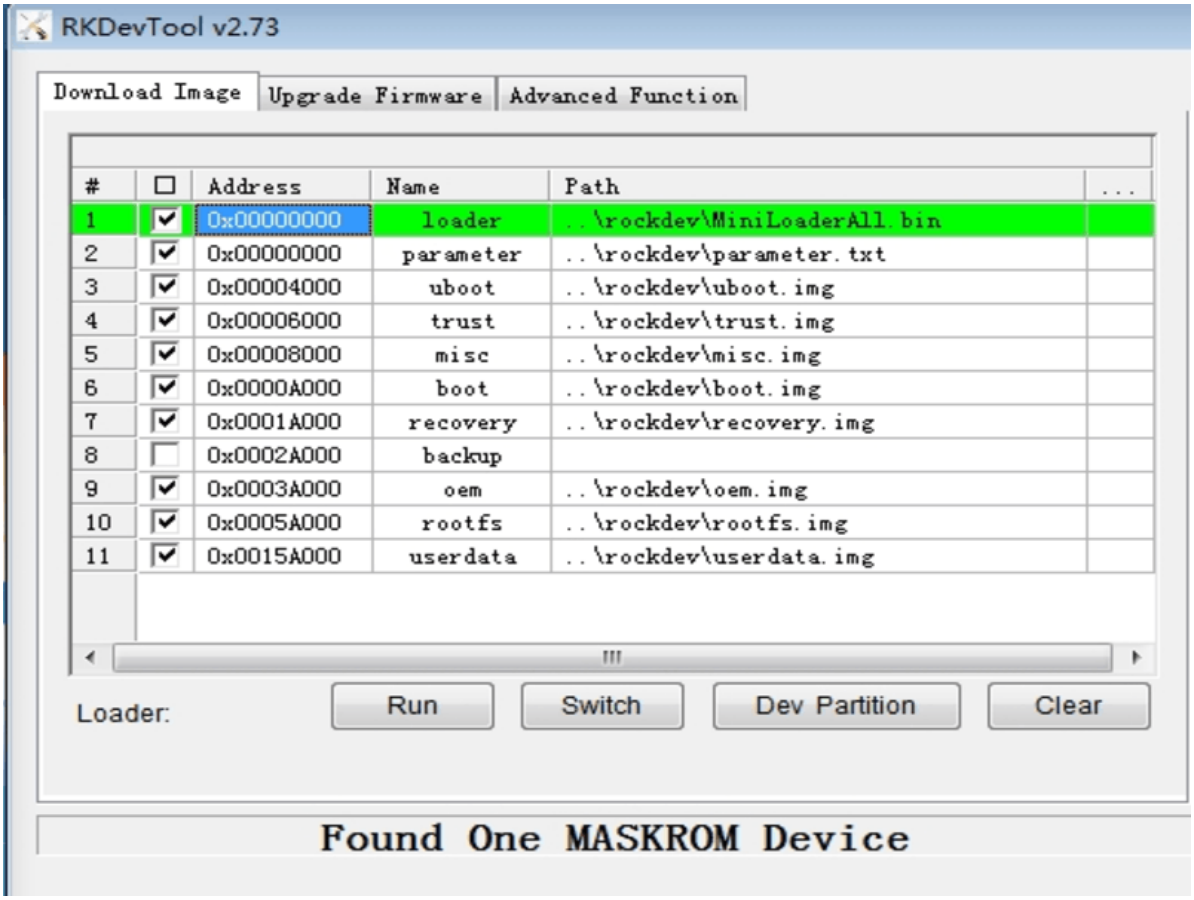


8.1 Windows Upgrade Introduction

SDK provides windows upgrade tool (this tool should be V2.55 or later version) which is located in project root directory:

```
tools/
├── windows/RKDevTool
```

As shown below, after compiling the corresponding firmware, device should enter MASKROM or BootROM mode for update. After connecting USB cable, long press the button “MASKROM” and press reset button “RST” at the same time and then release, device will enter MASKROM Mode. Then you should load the paths of the corresponding images and click “Run” to start upgrade. You can also press the “recovery” button and press reset button “RST”then release to enter loader mode to upgrade. Partition offset and flashing files of MASKROM Mode are shown as follows (Note: Window PC needs to run the tool as an administrator):



Note：Before upgrade, please install the latest USB driver, which is in the below directory:

```
<SDK>/tools/windows/DriverAssitant_v4.91.zip
```

8.2 Linux Upgrade Instruction

The Linux upgrade tool (Linux_Upgrade_Tool should be v1.33 or later versions) is located in “tools/linux” directory. Please make sure your board is connected to MASKROM/loader rockusb, if the compiled firmware is in rockdev directory, upgrade commands are as below:

```
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
```

Or upgrade the whole update.img in the firmware

```
sudo ./upgrade_tool uf rockdev/update.img
```

Or in root directory, run the following command on the machine to upgrade in MASKROM state:

```
./rkflash.sh
```

8.3 System Partition Introduction

Default partition (below is RK3399Pro EVB reference partition) is showed as follows:

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 partition: flashing uboot.img built from uboot.
- trust partition: flashing trust.img built from uboot.
- misc partition: flashing misc.img, for recovery.
- boot partition: flashing boot.img built from kernel.
- recovery partition: flashing recovery.img.
- backup partition: reserved, temporarily useless. Will be used for backup of recovery as in Android in future.
- oem partition: used by manufacturer to store their APP or data, mounted in /oem directory
- rootfs partition: store rootfs.img built from buildroot or debian.
- userdata partition: store files temporarily generated by APP or for users, mounted in /userdata directory

9. RK3399Pro SDK Firmware and Simple Demo Test

9.1 RK3399Pro SDK Firmware

RK3399PRO_LINUX_SDK_V1.3.3_20200813 firmware download links are as follows:
(Including Buildroot,Debian and Yocto firmware)

- Baidu cloud disk

Buildroot:
V10 (green) development board
V11/V12 (black) development board
V13 (black) development board
V14 (black) development board

Debian 9:
Debian9 rootfs

Debian 10:
Debian10 pcie rootfs
Debian10 usb rootfs

Yocto:
Yocto rootfs

- Microsoft OneDriver

Buildroot:
V10 (green) development board
V11/V12 (black) development board
V13 (black) development board
V14 (black) development board

Debian 9:
Debian9 rootfs

Debian 10:
Debian10 pcie rootfs
Debian10 usb rootfs

Yocto:
Yocto rootfs

9.2 RKNN_DEMO Test

Firstly, insert usb camera, run rknn_demo in Buildroot system or run test_rknn_demo.sh in Debian system. Please refer to the project document: SDK/docs/Linux/ApplicationNote/Rockchip_Developer_Guide_Linux_RKNN_Demo_CN/EN.pdf for details, the results of running in Buildroot are as follows:

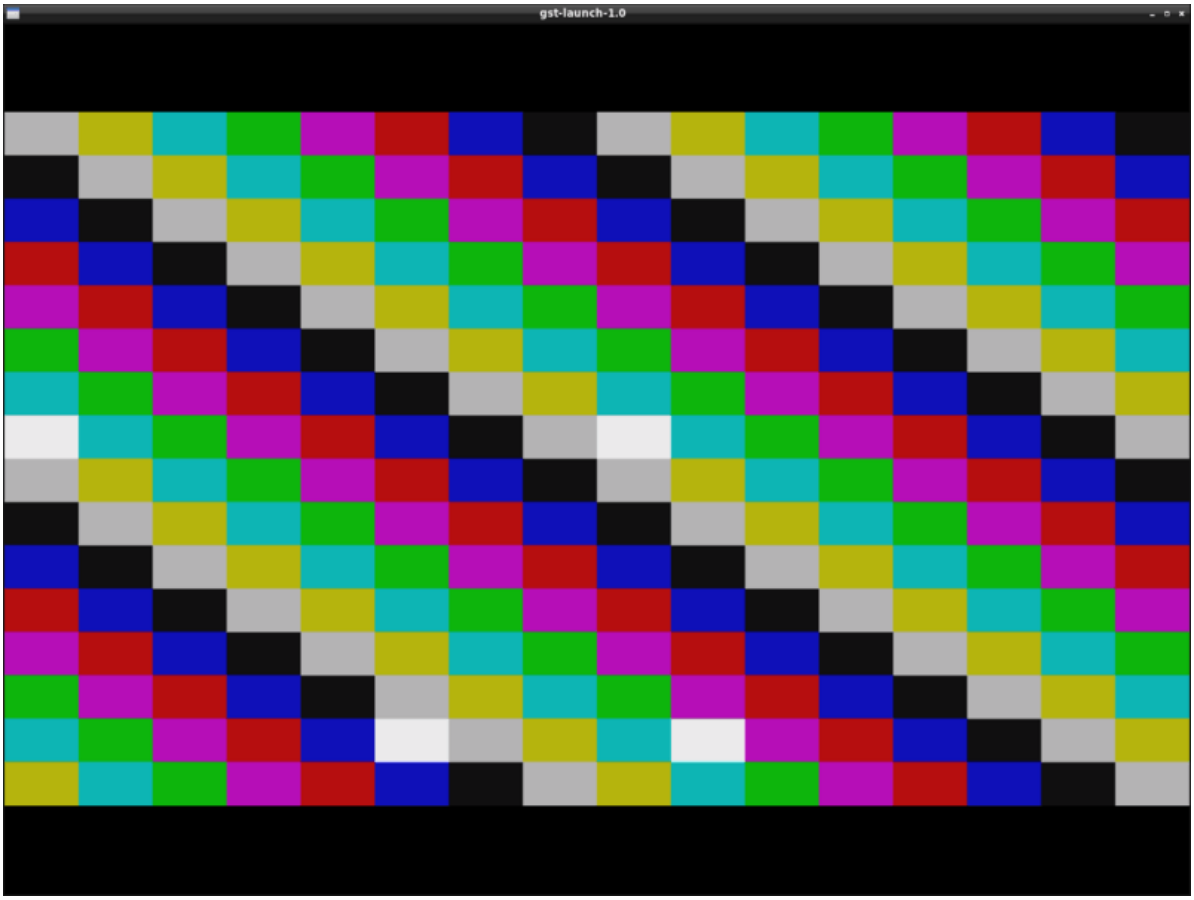
```
[root@rk3399pro:/]# rknn_demo
librga:RGA_GET_VERSION:3.02,3.020000
ctx=0x2a64ac20,ctx->rgaFd=3
Rga built version:version:+2017-09-28 10:12:42
success build
set plane zpos = 3 (0~3)size = 12582988, g_bo.size = 13271040
size = 12582988, cur_bo->size = 6635520
size = 12582988, cur_bo->size = 6635520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

It will display as follows：

9.3 N4 Camera Test

First connect the N4 camera module (need a 12V power supply), then directly open the camera application in the Buildroot system or run test_camera-rkisp1.sh in the Debian system.
The running results in Buildroot are as follows: (when no camera sensor is connected)



10. SSH Public Key Operation Introduction

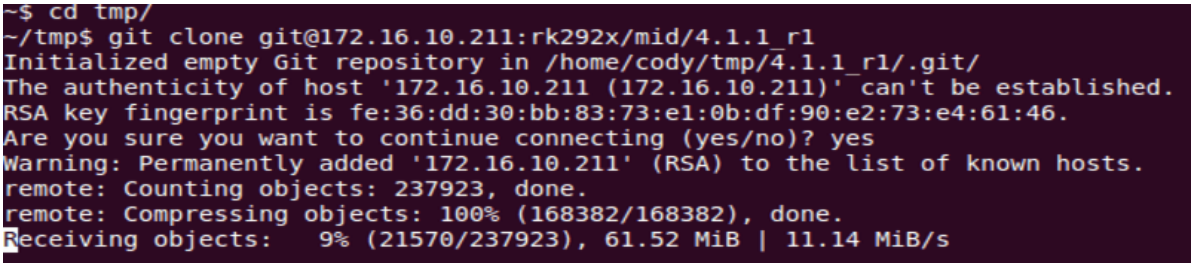
Please follow the introduction in the “Rockchip SDK Application and Synchronization Guide” to generate an SSH public key and send the email to fae@rock-chips.com, to get the SDK code.
This document will be released to customers during the process of applying for permission.

10.1 Multiple Machines Use the Same SSH Public Key

If the same SSH public key should be used in different machines, you can copy the SSH private key file id_rsa to “~/.ssh/id_rsa” of the machine you want to use.

The following prompt will appear when using a wrong private key, please be careful to replace it with the correct private key.

After adding the correct private key, you can use git to clone code, as shown below.



Adding ssh private key may result in the following error.

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

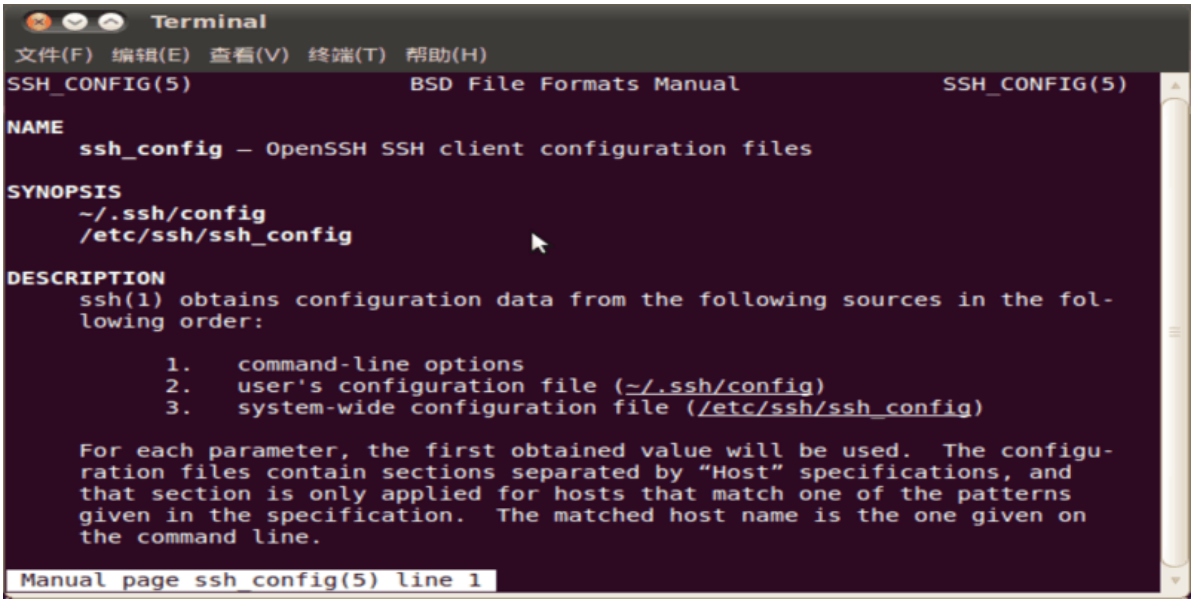
Enter the following command in console to solve:

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

10.2 One Machine Switches Different SSH Public Keys

You can configure SSH by referring to ssh_config documentation.

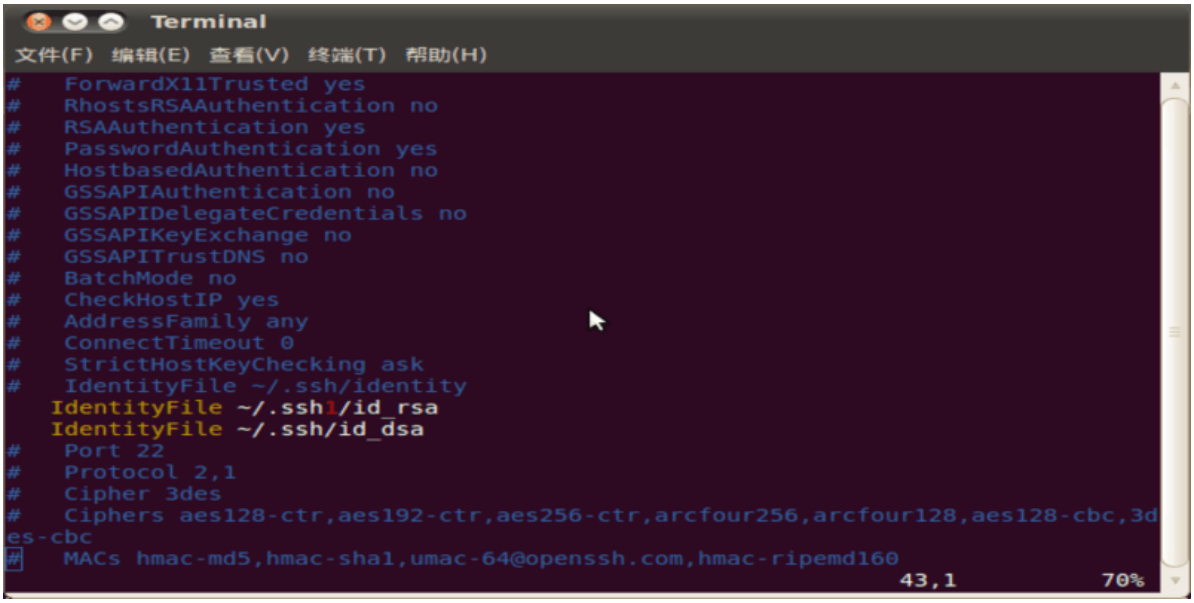
```
~$ man ssh_config
```



Run the following command to configure SSH configuration of current user.

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

As shown in the figure, SSH uses the file "~/.ssh1/id_rsa" of another directory as an authentication private key. In this way, different keys can be switched.



10.3 Key Authority Management

Server can monitor download times and IP information of a key in real time. If an abnormality is found, download permission of the corresponding key will be disabled.

Keep the private key file properly. Do not grant second authorization to third parties.

10.4 Reference Documents

For more details, please refer to document
“/docs/Others/Rockchip_User_Guide_SDK_Application_And_Synchronization_CN.pdf"