RK3399Pro Linux SDK Release Notes

ID: RK-FB-CS-009

Release Version: V1.3.1

Release Date: 2020-07-22

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

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。
2020-10-14	V1.1.2	Caesar Wang	Update Debian build note
2020-10-23	V1.1.3	Caesar Wang	Support RK3399Pro EVB V13
2020-12-03	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

Contents

RK3399Pro Linux SDK Release Notes
1. Overview
2. Main Functions
3. How to Get the SDK
4. Software Development Guide
4.1 Development Guide
4.2 NPU Development Tool
4.3 Software Update History
5. Hardware Development Guide
6. SDK Project Directory Introduction
7. SDK Compilation Instroduction
7.1 NPU Compilation Introduction
7.1.1 Uboot Compilation
7.1.2 Kernel Compilation Steps
7.1.3 Boot.img and NPU Firmware Generation Steps
7.1.4 Full Automatic Compilation
7.2 RK3399Pro Compilation Instroduction
7.2.1 U-boot Compilation
7.2.2 Kernel Compilation Steps
7.2.3 Recovery Compilation Steps
7.2.4 Buildroot rootfs and APP Compilation
7.2.5 Debian rootfs Compilation
7.2.6 Yocto rootfs Compilation
7.2.7 Full Automatic Compilation
7.2.8 Firmware Package
8. Upgrade Introduction
8.1 Windows Upgrade Introduction
8.2 Linux Upgrade Instruction
8.3 System Partition Introduction
9. RK3399Pro SDK Firmware and Simple Demo Test
9.1 RK3399Pro SDK Firmware
9.2 RKNN_DEMO Test
9.3 N4 Camera Test
10. SSH Public Key Operation Introduction
10.1 Multiple Machines Use the Same SSH Public Key
10.2 One Machine Switches Different SSH Public Keys
10.3 Key Authority Management
10.4 Reference Documents

1. Overview

This SDK is based on 3 Linux systems: Buildroot 2018.02-rc3, Yocto Thud 2.6, Debian 9, 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

SDK is released by Rockchip server or got from Github open source website. Please refer to Chapter 7 SDK Compilation Introduciton to build a development environment.

First method to get the SDK: 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
```

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_v1.3.0_20200324.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_v1.3.0_20200324.tgz -C
rk3399pro
cd rk3399pro
.repo/repo/repo sync -l
.repo/repo/repo sync
```

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

Second method to get the SDK: get source code from Github open source website:

Download repo tools:

```
git clone https://github.com/rockchip-linux/repo.git
```

Make an rk3399pro linux work directory:

```
mkdir rk3399pro_linux
```

Enter the rk3399pro linux work directory

```
cd rk3399pro_linux/
```

Initialize the repo repository:

```
../repo/repo init --repo-url=https://github.com/rockchip-linux/repo -u
https://github.com/rockchip-linux/manifests -b master -m
rk3399pro_linux_release.xml
```

Synchronize the whole project:

```
../repo/repo sync
```

Note: If your project has already started, please choose the first Method to get the code first. Unlike Github, it has passed by internal stress testing and version control. The second method is more suitable for enthusiasts and project evaluation.

4. Software Development Guide

4.1 Development Guide

RK3399Pro Linux SDK Kernel version is Kernel 4.4, Rootfs is Buidlroot (2018.02-rc3), Yocto(thud 2.6) and Debian9 respectively. To help engineers quick start of SDK development and debugging, “Rockchip_Developer_Guide_Linux_Software_EN” is released with the SDK. It can be obtained in the docs/ directory and will be continuously updated.

4.2 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, please refer to the document in the project directory docs/Soc_public/RK3399PRO/ 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.

```
├── Rockchip_Developer_Guide_RKNN_Toolkit_Custom_OP_CN.pdf
├── Rockchip_Developer_Guide_RKNN_Toolkit_Custom_OP_EN.pdf
├── Rockchip_Quick_Start_RKNN_Toolkit_V1.3.0_CN.pdf
├── Rockchip_Quick_Start_RKNN_Toolkit_V1.3.0_EN.pdf
├── Rockchip_Trouble_Shooting_RKNN_Toolkit_V1.3_CN.pdf
├── Rockchip_Trouble_Shooting_RKNN_Toolkit_V1.3_EN.pdf
├── Rockchip_User_Guide_RKNN_Toolkit_V1.3.0_CN.pdf
├── Rockchip_User_Guide_RKNN_Toolkit_V1.3.0_EN.pdf
├── Rockchip_User_Guide_RKNN_Toolkit_Visualization_CN.pdf
├── Rockchip_User_Guide_RKNN_Toolkit_Visualization_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/Soc_public/RK3399PRO/ Rockchip_RK3399Pro_Instruction_Linux_NPU_CN.pdf” for RK3399Pro NPU software setup instructions.

4.3 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_Release_Note.txt
```

Or refer to the project directory:

```
<SDK>/docs/SoC_public/RK3399PRO/RK3399PRO_Linux_SDK_Release_Note.pdf
```

5. Hardware Development Guide

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

```
<SDK>/docs/Soc_public/RK3399PRO/Rockchip_RK3399Pro_User_Guide_Hardware_xx.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 YoctoThud 2.6.

7. SDK Compilation Instroduction

Ubuntu 16.04 system: Please install software packages with below commands to setup Buildroot compiling environment:

```
sudo apt-get install repo git-core gitk git-gui gcc-arm-linux-gnueabihf u-boot-tools device-tree-compiler gcc-aarch64-linux-gnu mtools parted libudev-dev libusb-1.0-0-dev python-linaro-image-tools linaro-image-tools autoconf autotools-dev libsigsegv2 m4 intltool libdrm-dev curl sed make binutils build-essential gcc g++ bash patch gzip bzip2 perl tar cpio python unzip rsync file bc wget libncurses5 libqt4-dev libglib2.0-dev libgtk2.0-dev libglade2-dev cvs git mercurial rsync openssh-client subversion asciidoc w3m dblatex graphviz python-matplotlib libc6:i386 libssl-dev texinfo liblz4-tool genext2fs expect patchelf xutils-dev
```

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

```
sudo apt-get install repo git-core gitk git-gui gcc-arm-linux-gnueabihf u-boot-tools device-tree-compiler gcc-aarch64-linux-gnu mtools parted libudev-dev libusb-1.0-0-dev python-linaro-image-tools linaro-image-tools gcc-arm-linux-gnueabihf libssl-dev gcc-aarch64-linux-gnu g+conf autotools-dev libsigsegv2 m4 intltool libdrm-dev curl sed make binutils build-essential gcc g++ bash patch gzip bzip2 perl tar cpio python unzip rsync file bc wget libncurses5 libqt4-dev libglib2.0-dev libgtk2.0-dev libglade2-dev cvs git mercurial rsync openssh-client subversion asciidoc w3m dblatex graphviz python-matplotlib libc6:i386 libssl-dev texinfo liblz4-tool genext2fs xutils-dev
```

Ubuntu 17.04 or later version system: In addition to the above, the following dependencies is needed:

```
sudo apt-get install lib32gcc-7-dev g++-7 libstdc++-7-dev
```

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.

Note:

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

```
<SDK>/docs/Soc_public/RK3399PRO/Rockchip_RK3399Pro_Instruction_Linux_NPU_CN.pdf。/
```

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

7.1 NPU Compilation Introduction

7.1.1 Uboot Compilation

Enter project npu/u-boot directory and run `make.sh` to get rknpu_lion_loader_v1.03.103.bin trust.img uboot.img:

rk3399pro-npu：

```
./make.sh rknpu-lion
```

The compiled files are in u-boot directory:

```
u-boot/
├── rknpu_lion_loader_v1.03.103.bin
├── trust.img
└── uboot.img
```

7.1.2 Kernel Compilation Steps

Enter project root directory and run the following command to automatically compile and package kernel:

RK3399Pro EVB V10/V11/V12 boards:

```
cd npu/kernel
git checkout remotes/rk/stable-4.4-rk3399pro_npu-linux
make ARCH=arm64 rk3399pro_npu_defconfig
make ARCH=arm64 rk3399pro-npu-evb-v10.img -j12
```

RK3399Pro EVB V13/V14 boards:

```
cd kernel //change to stable-4.4-rk3399pro_npu-pcie-linux branch
make ARCH=arm64 rk3399pro_npu_pcie_defconfig
make ARCH=arm64 rk3399pro-npu-evb-v10-multi-cam.img -j12
```

7.1.3 Boot.img and NPU Firmware Generation Steps

Enter project npu directory and run the following command to automatically compile and package boot.img:

RK3399Pro EVB V10/V11/V12 boards:

```
cd npu
./build.sh ramboot
./mkfirmware.sh rockchip_rk3399pro-npu
```

RK3399Pro EVB V13/V14 boards:

```
cd npu/device/rockchip
cp rk3399pro-npu-multi-cam/BoardConfig.mk .BoardConfig.mk
cd - && cd npu
./build.sh ramboot
./mkfirmware.sh rockchip_rk3399pro-npu-multi-cam
```

7.1.4 Full Automatic Compilation

After compiling various parts of Kernel/U-Boot/Rootfs above, enter root directory of project directory and execute the following commands to automatically complete all compilation:

RK3399Pro EVB V10/V11/V12 boards:

```
cd npu/device/rockchip
cp rk3399pro-npu/BoardConfig.mk .BoardConfig.mk
cd - && cd npu
./build.sh uboot
./build.sh kernel
./build.sh ramboot
./mkfirmware.sh rockchip_rk3399pro-npu
```

RK3399Pro EVB V13/V14 boards:

```
cd npu/device/rockchip
cp rk3399pro-npu-multi-cam/BoardConfig.mk .BoardConfig.mk
cd ../../
./build.sh uboot
./build.sh kernel
./build.sh ramboot
./mkfirmware.sh rockchip_rk3399pro-npu-multi-cam
```

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".

7.2 RK3399Pro Compilation Instroduction

7.2.1 U-boot Compilation

Enter project u-boot directory and execute `make.sh` to get rk3399pro_loader_v1.23.115.bin trust.img uboot.img: RK3399Pro EVB boards：

```
./make.sh rk3399pro
```

The compiled file is in u-boot directory:

```
u-boot/
├── rk3399pro_loader_v1.24.119.bin
├── trust.img
└── uboot.img
```

7.2.2 Kernel Compilation Steps

Enter project root directory and run the following command to automatically compile and package kernel:

RK3399Pro EVB V10 boards：

```
cd kernel
make ARCH=arm64 rockchip_linux_defconfig
make ARCH=arm64 rk3399pro-evb-v10-linux.img -j12
```

RK3399Pro EVB V11/V12 boards：

```
cd kernel
make ARCH=arm64 rockchip_linux_defconfig
make ARCH=arm64 rk3399pro-evb-v11-linux.img -j12
```

RK3399Pro EVB V13 boards：

```
cd kernel
make ARCH=arm64 rockchip_linux_defconfig
make ARCH=arm64 rk3399pro-evb-v13-linux.img -j12
```

RK3399Pro EVB V14 boards：

```
cd kernel
make ARCH=arm64 rockchip_linux_defconfig
make ARCH=arm64 rk3399pro-evb-v14-linux.img -j12
```

After compiling, boot.img which contains image and DTB of kernel will be generated in kernel directory.

7.2.3 Recovery Compilation Steps

Enter project root directory and run the following command to automatically complete compilation and packaging of Recovery.

RK3399Pro EVB boards：

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

The recovery.img is generated in Buildroot directory “output/rockchip_rk3399pro_recovery/images” after compiling.

7.2.4 Buildroot rootfs and APP Compilation

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

RK3399Pro EVB V10/V11/V12 boards:

```
cd device/rockchip/rk3399pro
cp BoardConfig_rk3399pro_usb.mk ../.BoardConfig.mk
cd - && ./build.sh rootfs
```

RK3399Pro EVB V13 boards:

```
cd device/rockchip/rk3399pro
cp BoardConfig_rk3399pro_multi_cam_pcie.mk ../.BoardConfig.mk
cd - && ./build.sh rootfs
```

RK3399Pro EVB V14 boards:

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

After compiling, rootfs.ext4 is generated in Buildroot directory “output/rockchip_rk3399pro/images”.

Note: If you need to compile a single module or a third-party application, you need to setup the cross-compiling environment.Cross-compiling tool is located in “buildroot/output/rockchip_rk3399pro/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:

```
aarch64-linux-gcc --version
```

When the following logs are printed, configuration is successful:

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

7.2.5 Debian rootfs Compilation

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

Enter debian/ directory firstly:

```
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 '/home/foxluo/work3/rockchip/rk_linux/rk3399_linux/rootfs/ubuntu-build-service/stretch-desktop-armhf/chroot' mounted with noexec or nodev
```

Solution：

```
mount -o remount,exec,dev xxx  (xxx is the mount place), then
rebuild it.
```

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, but 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-compiled package is shared in Baidu Cloud Network Disk

(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.

7.2.6 Yocto rootfs Compilation

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.

7.2.7 Full Automatic Compilation

After compiling various parts of Kernel/U-Boot/Recovery/Rootfs above, enter root directory of project directory and execute the following commands to automatically complete all compilation:

```
$./build.sh all
```

It is Buildroot by default, you can specify rootfs by setting the environment variable RK_ROOTFS_SYSTEM. For example, if Yocto is needed, you can generate with the following commands:

```
$export RK_ROOTFS_SYSTEM=yocto
$./build.sh all
```

Detailed parameters usage, you can use help to search, for example

```
rk3399pro$ ./build.sh --help
Usage: build.sh [OPTIONS]
Available options:
BoardConfig*.mk    -switch to specified board config
uboot              -build uboot
spl                -build spl
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'.
```

Board level configurations of each board need to be configured in the "/device/rockchip/rk3399pro/Boardconfig.mk".

Main configurations of RK3399Pro EVB are as follows:

```
# Target arch
export RK_ARCH=arm64
# Uboot defconfig
export RK_UBOOT_DEFCONFIG=rk3399pro
# Kernel defconfig
export RK_KERNEL_DEFCONFIG=rockchip_linux_defconfig
# Kernel dts
export RK_KERNEL_DTS=rk3399pro-evb-v14-linux
# boot image type
export RK_BOOT_IMG=boot.img
# kernel image path
export RK_KERNEL_IMG=kernel/arch/arm64/boot/Image
# parameter for GPT table
export RK_PARAMETER=parameter.txt
# Buildroot config
export RK_CFG_BUILDROOT=rockchip_rk3399pro_combine
# Recovery config
export RK_CFG_RECOVERY=rockchip_rk3399pro_recovery
```

7.2.8 Firmware Package

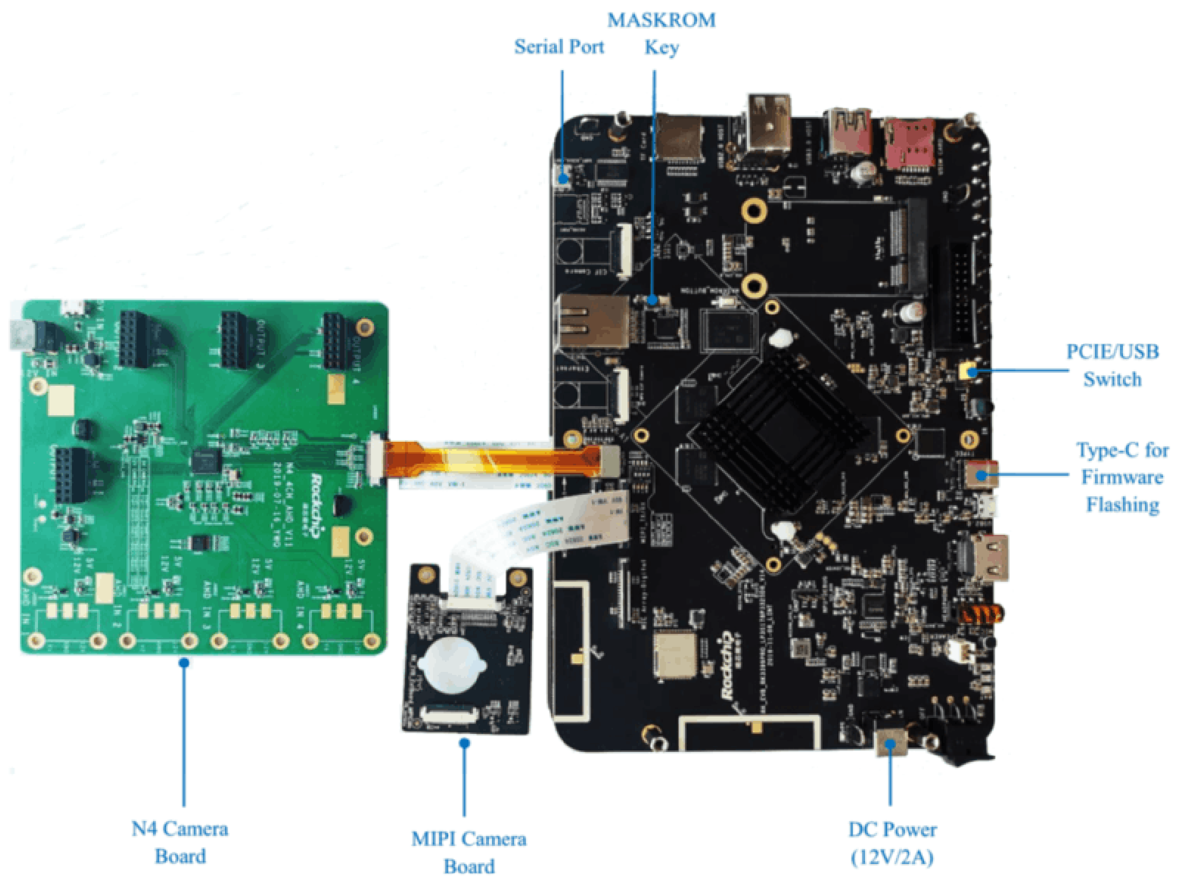
After compiling various parts of Kernel/U-Boot/Recovery/Rootfs above, enter root directory of project directory and run the following command to automatically complete all firmware packaged into rockdev directory:

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 V12 board, as shown in the following figure

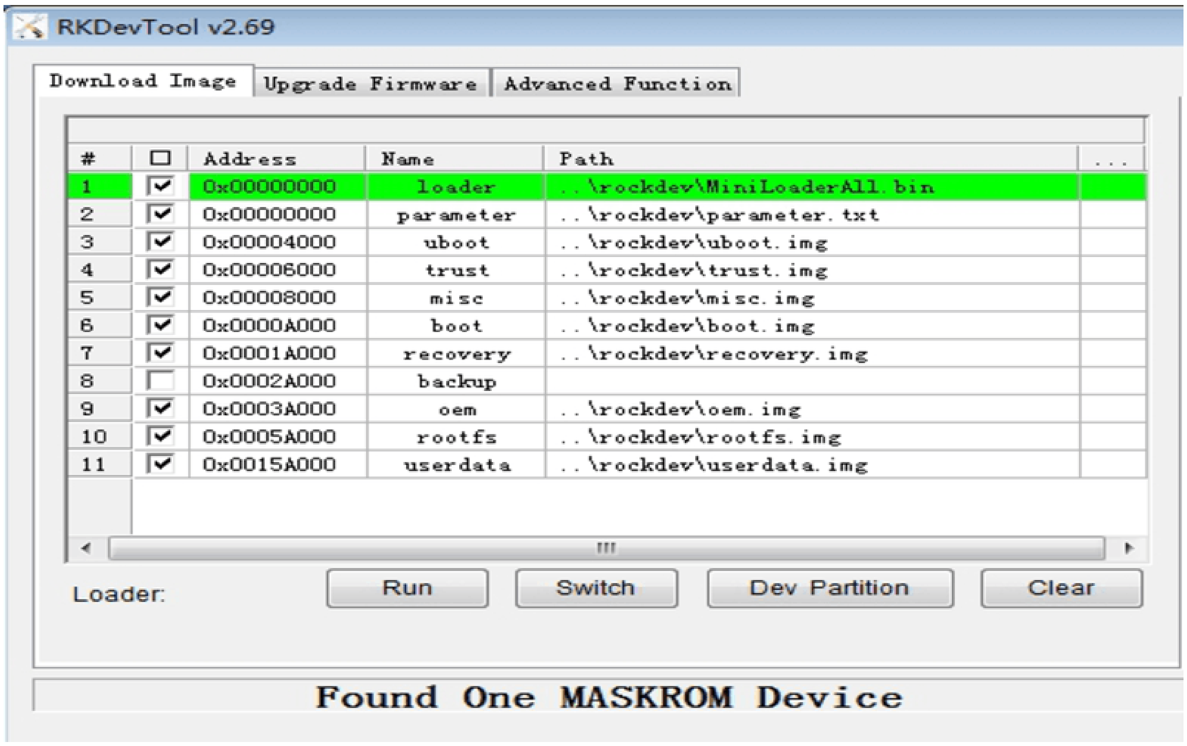


8.1 Windows Upgrade Introduction

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

```
tools/
├── windows/AndroidTool
```

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):



#	☐	Address	Name	Path
1	✓	0x00000000	loader	..\rockdev\MiniLoaderAll.bin
2	✓	0x00000000	parameter	..\rockdev\parameter.txt
3	✓	0x00004000	uboot	..\rockdev\uboot.img
4	✓	0x00006000	trust	..\rockdev\trust.img
5	✓	0x00008000	misc	..\rockdev\misc.img
6	✓	0x0000A000	boot	..\rockdev\boot.img
7	✓	0x0001A000	recovery	..\rockdev\recovery.img
8		0x0002A000	backup	
9	✓	0x0003A000	oem	..\rockdev\oem.img
10	✓	0x0005A000	rootfs	..\rockdev\rootfs.img
11	✓	0x0015A000	userdata	..\rockdev\userdata.img

Found One MASKROM Device

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

```
<SDK>/tools/windows/DriverAssitant_v4.8.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: update uboot.img compiled by uboot.
- trust partition: update trust.img compiled by uboot.
- misc partition: update misc.img for recovery.
- boot partition: update boot.img compiled by kernel.
- recovery partition: update recovery.img.
- backup partition: reserved, temporarily useless. Will be used for backup of recovery as Android in future.
- oem partition: used by manufacturer to store manufacturer's app or data. Read only. Replace the data partition of original speakers. Mounted in /oem directory.
- rootfs partition: store rootfs.img compiled by Buildroot , Yocto or Debian.
- userdata partition: store files temporarily generated by app or for users. Read and write, mounted in /userdata directory.

9. RK3399Pro SDK Firmware and Simple Demo Test

9.1 RK3399Pro SDK Firmware

RK3399PRO_LINUX_SDK_V1.3.0_20200324 firmware download links are as follows: (Including Buildroot,Debian and Yocto firmware)

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

Debian 9: Suitable for all development boards

Yocto: Suitable for all development boards

9.2 RKNN_DEMO Test

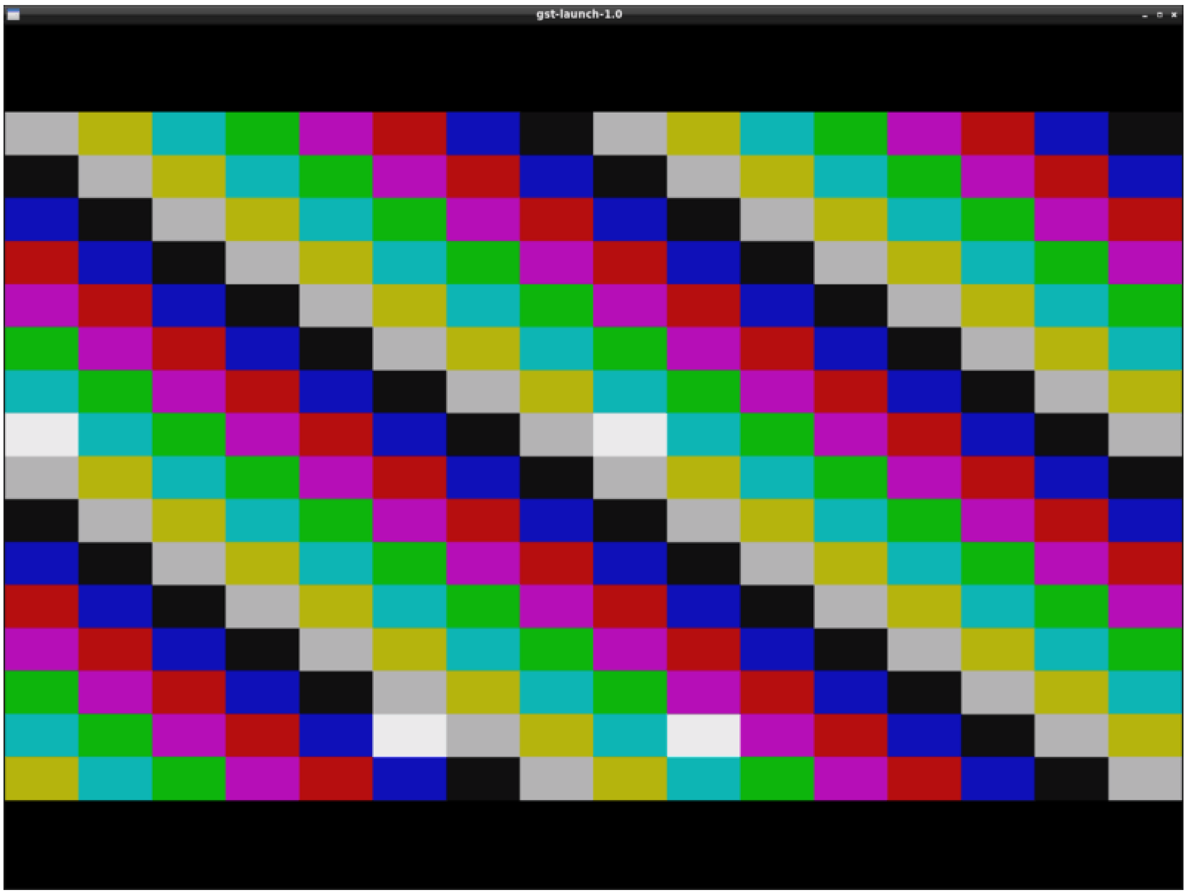
Firstly, insert usb camera, run rknn_demo in Buildroot system or run test_rknn_demo.sh in Debian system. Please refer to the project document: docs/Soc_public/RK3399PRO/Rockchip_Developer_Guide_Linux_RKNN_DEMO_CN.pdf for details, the results of running in Buildroot are as follows:

```
[root@rk3399pro:/]# rknn_demo
librga:RGA_GET_VERSION:3.02,3.020000
ctx=0x2e834c20,ctx->rgaFd=3
Rga built version:version:+2017-09-28 10:12:42
Success build
size = 12582988, g_bo.size = 13271040
size = 12582988, cur_bo->size = 13271040
size = 12582988, cur_bo->size = 13271040
...
read model:/usr/share/rknn_demo/mobilenet_ssd.rknn, len:32002449
Please configure uvc...
D RKNNAPI: ==============================================
D RKNNAPI: RKNN VERSION:
D RKNNAPI: API: 1.3.0 (933b767 build: 2019-11-27 14:43:32)
D RKNNAPI: DRV: 1.3.0 (c4f8c23 build: 2019-11-25 10:39:29)
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)



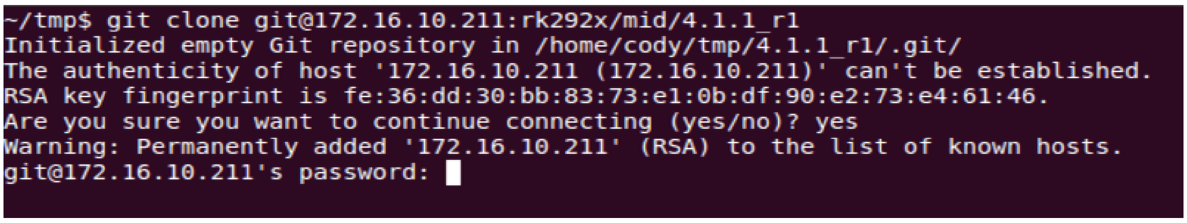
10. SSH Public Key Operation Introduction

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

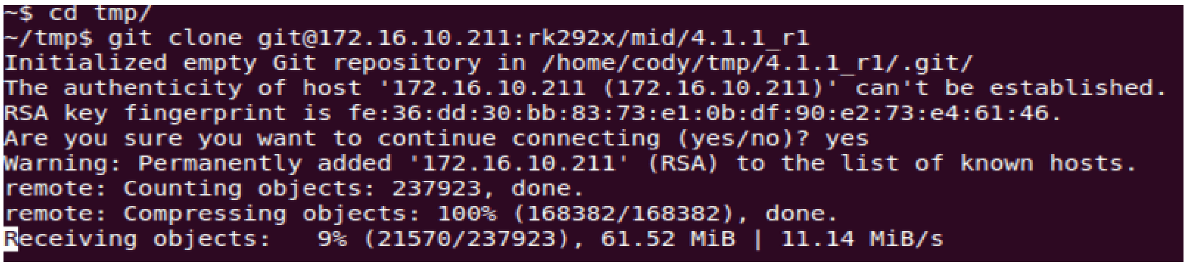
If the same SSH public key should be used in different machines, you can copy the SSH private key file id_rsa to “~/.ssh/id_rsa” of the machine you want to use.

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



```
~/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: 
```

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



```
~$ 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
```

Adding ssh private key may result in the following error.

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

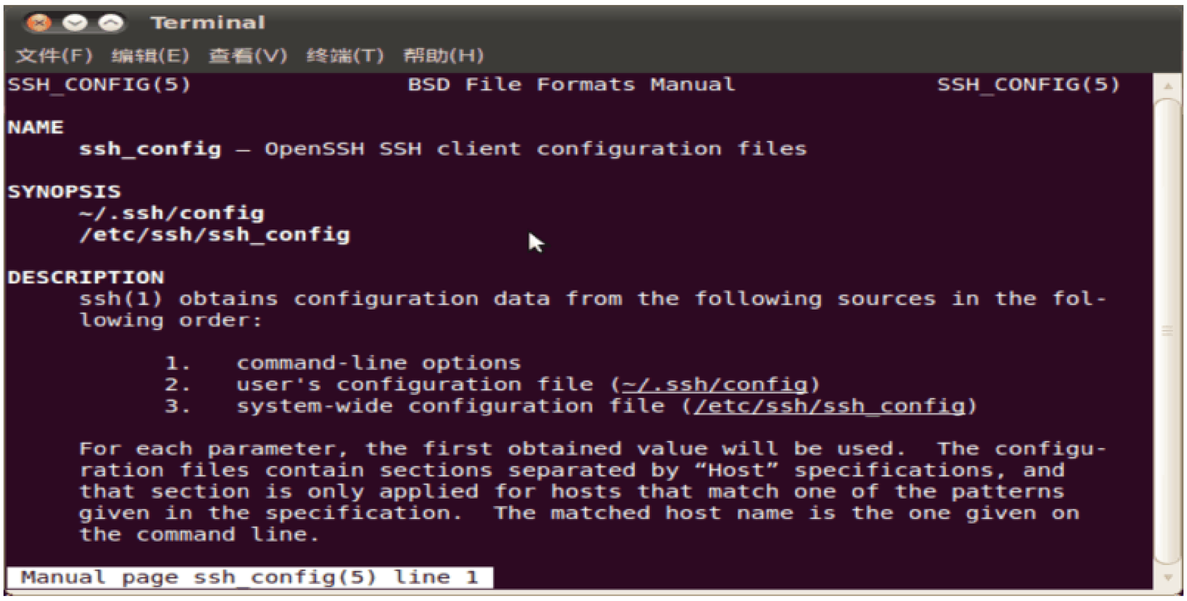
Enter the following command in console to solve:

```
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
```

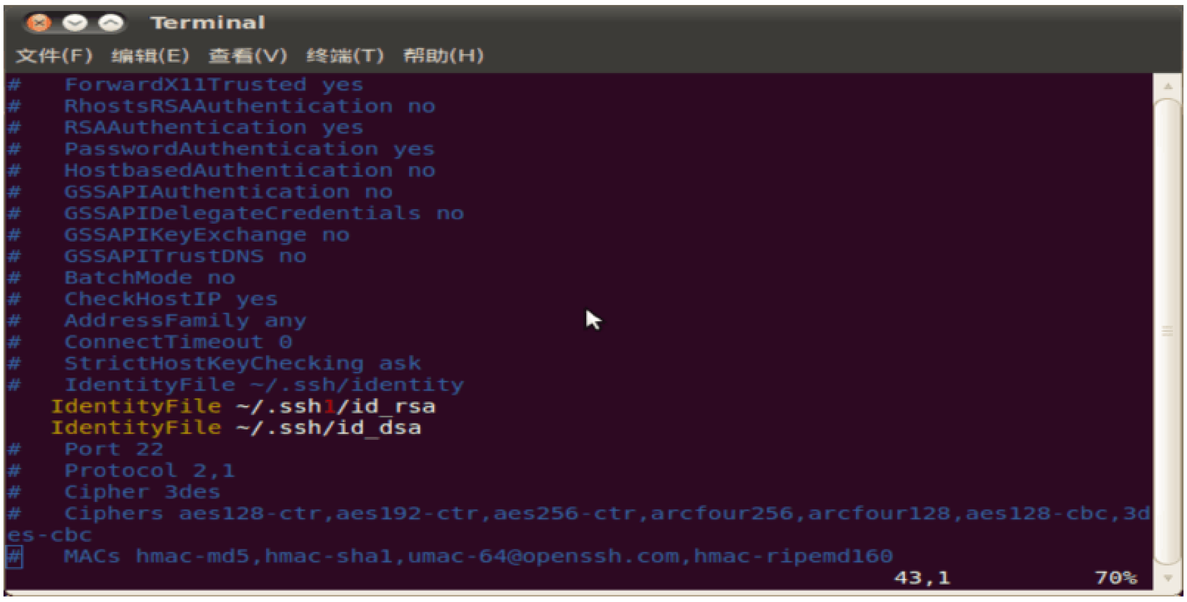


```
SSH_CONFIG(5)              BSD File Formats Manual              SSH_CONFIG(5)

NAME
     ssh_config — OpenSSH SSH client configuration files

SYNOPSIS
     ~/.ssh/config
     /etc/ssh/ssh_config

DESCRIPTION
     ssh(1) obtains configuration data from the following sources in the fol-
     lowing order:

           1.   command-line options
           2.   user's configuration file (~/.ssh/config)
           3.   system-wide configuration file (/etc/ssh/ssh_config)

     For each parameter, the first obtained value will be used.  The configu-
     ration files contain sections separated by “Host” specifications, and
     that section is only applied for hosts that match one of the patterns
     given in the specification.  The matched host name is the one given on
     the command line.

 Manual page ssh_config(5) line 1
```

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.



```
#   ForwardX11Trusted yes
#   RhostsRSAAuthentication no
#   RSAAuthentication yes
#   PasswordAuthentication yes
#   HostbasedAuthentication no
#   GSSAPIAuthentication no
#   GSSAPIDelegateCredentials no
#   GSSAPIKeyExchange no
#   GSSAPITrustDNS no
#   BatchMode no
#   CheckHostIP yes
#   AddressFamily any
#   ConnectTimeout 0
#   StrictHostKeyChecking ask
#   IdentityFile ~/.ssh/identity
   IdentityFile ~/.ssh1/id_rsa
   IdentityFile ~/.ssh/id_dsa
#   Port 22
#   Protocol 2,1
#   Cipher 3des
#   Ciphers aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,aes128-cbc,3d
es-cbc
#   MACs hmac-md5,hmac-sha1,umac-64@openssh.com,hmac-ripemd160
                                                             43,1          70%
```

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 “sdk/docs/RKTools manuals/Rockchip SDK Kit Application GuideV1.6-201905.pdf”.