Security Class: Top-Secret (　) 　Secret (　) 　Internal (　) 　Public (√)

RK3399PRO_Linux_SDK_V1.0.1_20190621

(Technical Department, Dept.Ⅲ)

Status： [　] Modifying [√] Released	Version：	V1.0.1
	Author：	Caesar Wang
	Date：	2019-06-21
	Reviewer：	Charles Chen
	Date：	2019-06-21

Fuzhou Rockchip Electronics Co.,Ltd
(All versions, all rights reserved)

Revision History

Revision Date	Version No.	Revision Description	Author	Reviewer
2019-02-17	Beta_V0.01	Initial Beta version	Caesar Wang	Eddie Cai
2019-03-21	Beta_V0.02	1. Modify the method of using ./mkfirmware.sh to generate image in chapter 5.1.3 2. Change the description of adding Debian to rknn_demo in chapter 8. 3. Change the SDK firmware to v0.02 in chapter 8	Caesar Wang	Eddie Cai
2019-06-06	V1.0.0	1. Release version 2. Add NPU related instructions 3. Add Yocto compilation instructions 4. Add github download instructions	Caesar Wang	Charles Chen
2019-06-21	V1.0.1	1. Change Software development name	Caesar Wang	Charles Chen

Table of content

Warranty Disclaimer

This document is provided according to "current situation" and Fuzhou Rockchip Electronics Co., Ltd. (short of "Rockchip" below) is not responsible for providing any express or implied statement or warranty of accuracy, reliability, completeness, marketability, specific purpose or infringement of any statements, information and content of this document. This document is intended as a guide only.

Due to product version upgrades or other reasons, this document may be updated or modified from time to time without notice.

Brand Statement

Rockchip, "瑞芯微", "瑞芯", and other Rockchip trademarks are trademarks of Fuzhou Rockchip electronics., ltd., and are owned by Fuzhou Rockchip electronics co.,ltd.

All other trademarks or registered trademarks mentioned in this document are owned by their respective owners.

Copyright © 2019 Fuzhou Rockchip Electronics Co., Ltd.

Without the written permission, any unit or individual shall not extract or copy part or all of the content of this document, and shall not spread in any form.

Fuzhou Rockchip Electronics Co., Ltd
Address: No.18 Building, A District, No.89 Software Boulevard, FuZhou, FuJian, PRC
Website: www.rock-chips.com
Customer service Tel.：+86-591-83991906
Customer service Fax：+86-591-83951833
Customer service e-Mail: fae@rock-chips.com

Chapter 1 Overview

This SDK is based on Linux Buildroot and Debian 9 system with kernel 4.4, it is applicable to the development of RK3399Pro EVB and all other Linux products 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 instructions, please refer to related documents under the project's docs/ directory.

Chapter 2 Main Functions

Functions	Module Names
Data Communication	Wi-Fi, Ethernet Card, USB, SDCARD
Application	Gallery, settings, video, audio, video playback

Chapter 3 How to Obtain SDK

SDK is released by Rockchip server or obtained from Github open source website. Please refer to chapter 7 SDK compiling instruction to build a development environment.

First method to obtain 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 Instruction.

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 obtain SDK source code through decompressing the initial compression package, which is the same as the one downloaded by repo.

Take rk3399pro_linux_sdk_beta_v0.01_20190217.tgz as an example. After copying initialization package, you can get source code by running the following command:

mkdir rk3399pro

tar xvf rk3399pro_linux_sdk_release_v1.0.0_20190606.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 obtain 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 rk3399pro linux work directory

cd rk3399pro_linux/

Initialize 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

Chapter 4 Software Development Guide

4.1 Development Guide

RK3399Pro Linux SDK Kernel version is Linux4.4, Rootfs is Buidlroot (2018.02-rc3), Yocto(thud 2.6) and Debian9 respectively. To help engineers quick start of SDK development and debugging, "Rockchip_Linux_Software_Development Guide_x.pdf" 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 the following items:

RKNN_DEMO(MobileNet SSD)：

For RKNN Demo, please refer to the directory external/rknn_demo/. See the project directory docs/SoC platform related/RK3399PRO/Rockchip RKNN_DEMO Module Development Guide V0.2.pdf for details.

RKNN-TOOLKIT：

Development tools are in project directory external/rknn-toolkit. Please refer to the document in the docs/Develop reference documents/NPU/ for details.

-RK3399Pro_Linux&Android_RKNN_API_V0.9.4_20190311.pdf

-RKNN-Toolkit FAQ.pdf

-RKNN-Toolkit Quick Start Guide_V1.0.0.pdf

-RKNN-Toolkit User Guide_V1.0.0.pdf

- Deep neural network model design recommendations based on RKNN.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/NRKNPUTool

NPU software startup instructions：

RK3399PRO NPU software start instructions, please refer to the project directory docs/SoC platform related/RK3399PRO/RK3399PRO_NPU power on and startup instructions.pdf

4.3 Software Update History

Software release version upgrade can be viewed through project xml, the detailed method is as follows:

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

Software updated information can be viewed through the project text, as follows:

```
.repo/manifests$ cat rk3399pro_linux_v0.01/RK3399PRO_Release_Note.txt
```

Or refer to the project directory:

docs/SoC platform related/RK3399PRO/RK3399PRO_Release_Note.pdf

Chapter 5 Hardware Development Guide

Hardware development can refer to user guides in the project directory

docs/SoC platform related/RK3399PRO/

-RK3399Pro EVB User Guide_V10_20190401.pdf

-RK3399Pro_EVB introduction_20181212.pdf

-RK3399PRO_IO_LIST_V11_for EVB 20181203.pdf

-RK3399Pro Hardware Design Guide, Schematic and PCB Review

Notes_V10_20190111/*

Chapter 6 RK3399Pro_Linux 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.

1) app: store application apps like Camer/Video/Music and other applications.

2) buildroot: customize buildroot root file system.

3) debian: debian root file system.

4) device/rockchip: store some scripts and prepared files for compiling and packaging firmware.

5) docs: store project help files.

6) external: related libraries, including audio, video, network and so on.

7) kernel: kernel source code.

8) npu: store npu code.

9) prebuilts: store cross-compilation toolchain.

10) recovery: store recovery project files.

11) rkbin: store firmware and tools.

12) rockdev: store compiled output firmware.

13) tools: store some commonly used tools.

14) u-boot: uboot code.

15) yocto：yocto root file system.

Chapter 7 SDK Compiling Instructions

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

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-4.8-multilib-arm-linux-gnueabihf 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

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

(No need to install gcc-4.8-multilib-arm-linux-gnueabihf)

Note: NPU firmware will be uploaded when RK3399pro Power on. The default NPU firmware is pre-programmed into /usr/share/npu_fw directory of rootfs. For NPU firmware programming and startup methods, please refer to the document under docs/Soc Platform related/RK3399PRO /RK3399PRO_NPU power on and start.pdf

NPU and RK3399pro firmware compiling methods will be described below:

7.1 NPU Compiling Instruction

7.1.1 Uboot Compiling

Enter project npu/u-boot directory and run make.sh to get

rknpu_lion_loader_v1.02.103.bin trust.img uboot.img:

rk3399pro-npu：

./make.sh rknpu-lion or ./build.sh uboot

The compiled files are in u-boot directory:

u-boot/

├── rknpu_lion_loader_v1.02.103.bin

├── trust.img

└── uboot.img

7.1.2 Kernel Compiling Steps

Enter project root directory and run the following command to automatically

compile and package kernel:

rk3399pro evb board：

cd kernel
make ARCH=arm64 rk3399pro_npu_defconfig
make ARCH=arm64 rk3399pro-npu-evb-v10.img -j12
Or ./build.sh kernel

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:

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

After compiling, boot.img, uboot.img, trust.img, MiniLoaderAll.bin are

generated in rockdev directory.

Note that generated npu firmware under rockdev should be placed in the rootfs

location /usr/share/npu_fw, or manually placed in rootfs.

7.2 RK3399pro Compiling Instruction
7.2.1 Uboot Compiling

Enter project u-boot directory and execute make.sh to get

rk3399pro_loader_v1.22.115.bin trust.img uboot.img:

rk3399pro evb：

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

The compiled file is in u-boot directory:

```
u-boot/
 ├─── rk3399pro_loader_v1.22.115.bin

 ├─── trust.img

 └─── uboot.img
```

7.2.2 Kernel Compiling 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
```

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

generated in kernel directory.

7.2.3 Recovery Compiling Steps

Enter project root directory and run the following command to automatically

complete compiling and packaging of Recovery.

rk3399pro evb board：

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

Recovery.img is generated in Buildroot directory

/output/rockchip_rk1808_recovery/images after compiling.

7.2.4 Buildroot rootfs and app Compiling

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

rk3399pro evb board：

```
./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 configure 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 view:

aarch64**-linux-gcc --version**

When the following logs are printed, configuration is successful:

aarch64-linux-gcc.br_real (Buildroot 2018.02-rc3-00218-gddd64f1) 6.4.0

7.2.5 Debian rootfs Compiling

Enter rootfs/directory firstly:

cd debian/   && ./build.sh debian

Then compile and generate debian firmware, you can refer to debian/readme.md in the current directory

7.2.5.1 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 32-bit debian:

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

Or 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/.

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 d enied E: Cannot install into target '/home/foxluo/work3/rockchip/rk_linux/rk33 99_linux/rootfs/ubuntu-build-service/stretch-desktop-armhf/chroot' mounted w ith noexec or nodev

Solution：

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

In addition, if there are other compiling issues, Please check firstly that the compiler system is not ext2/ext4

7.2.5.2 Building rk-debian rootfs

Compile 32-bit debian:

VERSION=debug ARCH=armhf ./mk-rootfs-stretch.sh

 (A "debug" behind is recommended in the development process)

Compile 64-bit debian:

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

(A "debug" behind is recommended in the development process)

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

./mk-image.sh

Will generate linaro-rootfs.img.

7.2.6 Yocto rootfs Compiling

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

RK3399PRO EVB board：

./build.sh yocto

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

FAQ:

If you encounter the following problem during the 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 to https://webkul.com/blog/setup-locale-python3

The image generated after compiling is in yocto/build/lastest/rootfs.img

The default login username is root

Refer to Rockchip Wiki for more detailed information of Yocto

7.2.7 Fully Automatic Compiling

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

$./build.sh all

Is buildroot by default, you can specify rootfs by setting the environment variable RK_ROOTFS_SYSTEM.

If Yocto is needed, you can generate with the following commands:

$export RK_ROOTFS_SYSTEM=yocto

$./build.sh all

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

rk3399pro$ ./build.sh --help

Can't found build config, please check again

====USAGE: build.sh modules====

uboot   -build uboot

kernel   -build kernel

rootfs   -build default rootfs, currently build buildroot as default

buildroot   -build buildroot rootfs

yocto   -build yocto rootfs, currently build ros as default

ros   -build ros rootfs

debian   -build debian rootfs

pcba   -build pcba

recovery   -build recovery

all   -build uboot, kernel, rootfs, recovery image

....

default   -build all modules

Board level configurations of each board need to be configured in

/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-v11-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-buildroot.txt
# Buildroot config
export RK_CFG_BUILDROOT=rockchip_rk3399pro
# Recovery config
export RK_CFG_RECOVERY=rockchip_rk3399pro_recovery
# ramboot config
```

7.2.8 Firmware Package Steps

After compiling various parts of Kernel/Uboot/Recovery/Rootfs above, enter

root directory of project directory and run the following command to

automatically complete all firmware packaged into rockdev directory:

 Generate Buildroot firmware:

 ./mkfirmware.sh

 Generate Debian firmware：

 ./build.sh BoardConfig_debian.mk

./mkfirmware can generate debian firmware.

Chapter 8 Upgrade Instruction

There are two versions of current rk3399pro evb, the green board is v10 version, and the black board is v11 version. Board function positions are the same. The following is the introduction of rk3399pro evb v10 board, as shown in the following figure.
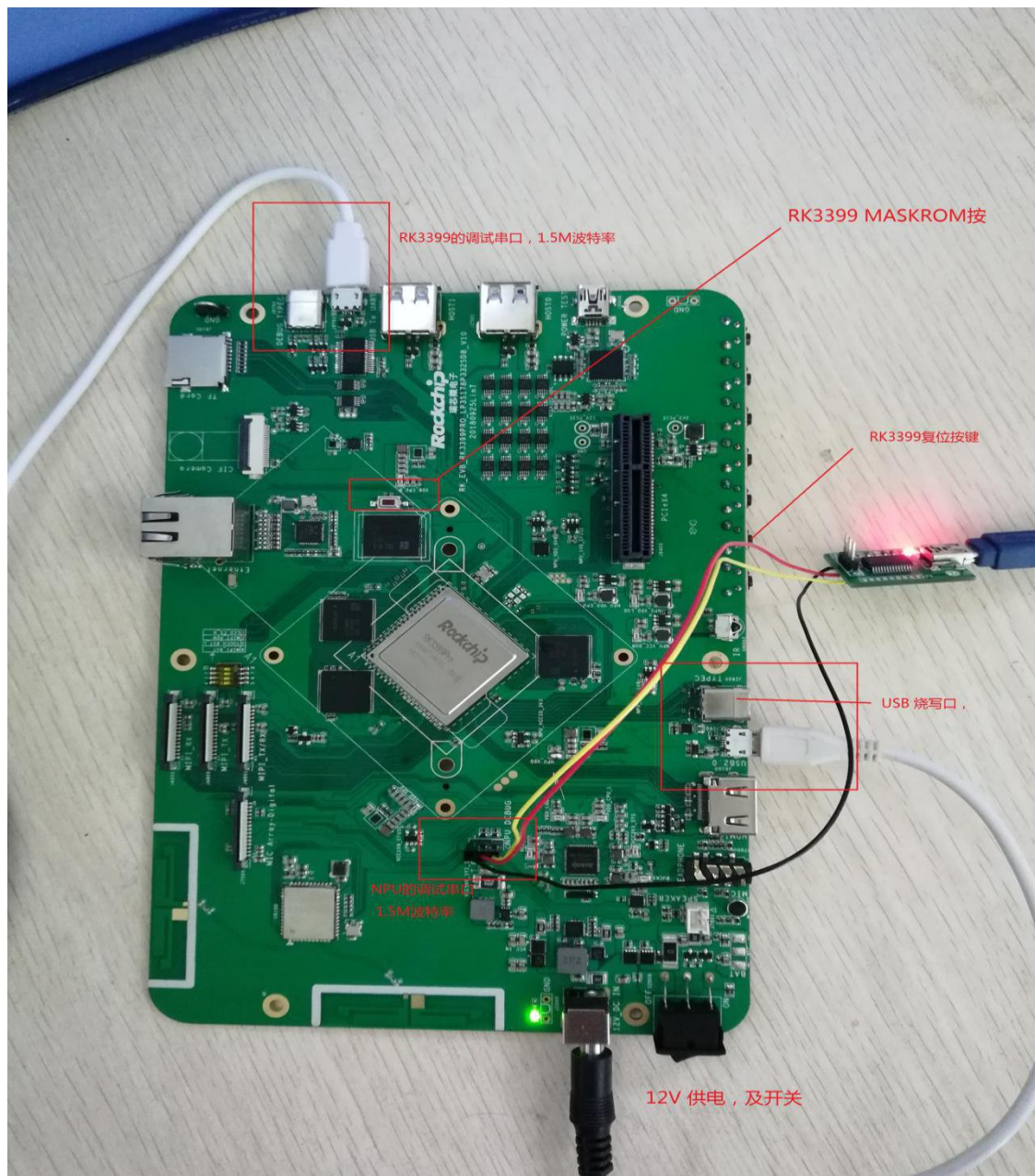

Figure 1 RK3399PRO EVB

8.1 Windows Upgrade Instruction

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 (aka 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 update files of MASKROM Mode are shown as follows (Note: Window PC needs to run the tool as an administrator):
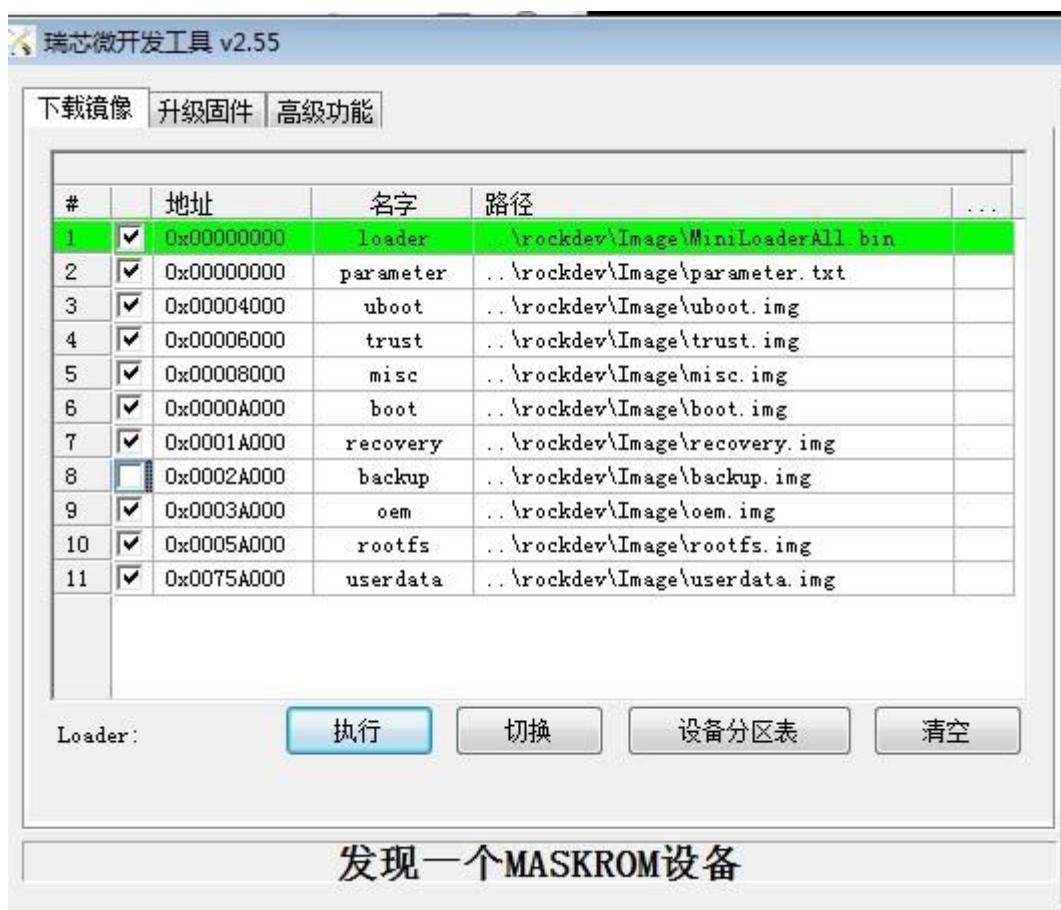
Figure 2 Upgrade tool **AndroidTool.exe**

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

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:

```
udo ./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 in root directory, machine run the following command to upgrade in MASKROM state:

```
./rkflash.sh
```

8.3 System Partition Instruction

Default partition (below is RK3399PRO EVB reference partition):

Number	Start (sector)	End (sector)	Size	Code	Name
1	16384	24575	4096K	0700	uboot
2	24576	32767	4096K	0700	trust
3	32768	40959	4096K	0700	misc
4	40960	106495	32.0M	0700	boot
5	106496	303104	96.0M	0700	recovery
6	303104	368639	32.0M	0700	backup
7	368640	499711	64.0M	0700	oem
8	499712	25696863	1024M	0700	rootfs
9	2596864	30535646	13.3G	0700	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 in 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.

Chapter 9 RK3399Pro SDK Firmware and Simple Demo Test

9.1 RK3399Pro SDK Firmware

RK3399PRO_LINUX_SDK_V1.0.0_20190528 firmware download links are as follows:

(include Debian, Buildroot and Yocto firmware)

V10 (green) development board:

Buildroot：https://eyun.baidu.com/s/3qZ3Wn72

Debian：https://eyun.baidu.com/s/3c3mJHWc

V11 (black) development board:

Buildroot： https://eyun.baidu.com/s/3c4iWmFe

Debian：https://eyun.baidu.com/s/3bqyXe8J

9.2 RKNN_DEMO Test

Firstly, insert usb camera, run **rknn_demo** in buildroot system and run **test_rknn_demo.sh** in Debian system.

Refer to project document docs/Soc Platform Related/RK3399PRO/ Rockchip RKNN_DEMO module developer guide.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
...
get device /dev/video10
read model:/usr/share/rknn_demo/mobilenet_ssd.rknn, len:32002449
spec = local:transfer_proxy
```

```
    D RKNNAPI:
========================================================
    D RKNNAPI: RKNN VERSION:
    D RKNNAPI:   API: 0.9.6 (47a27f6 build: 2019-06-02 22:29:50)
    D RKNNAPI:   DRV: 0.9.7 (d65a37f build: 2019-05-31 14:23:01)
    D RKNNAPI:
========================================================
    Using UVC media node
...
```

The effect on screen is as follows：
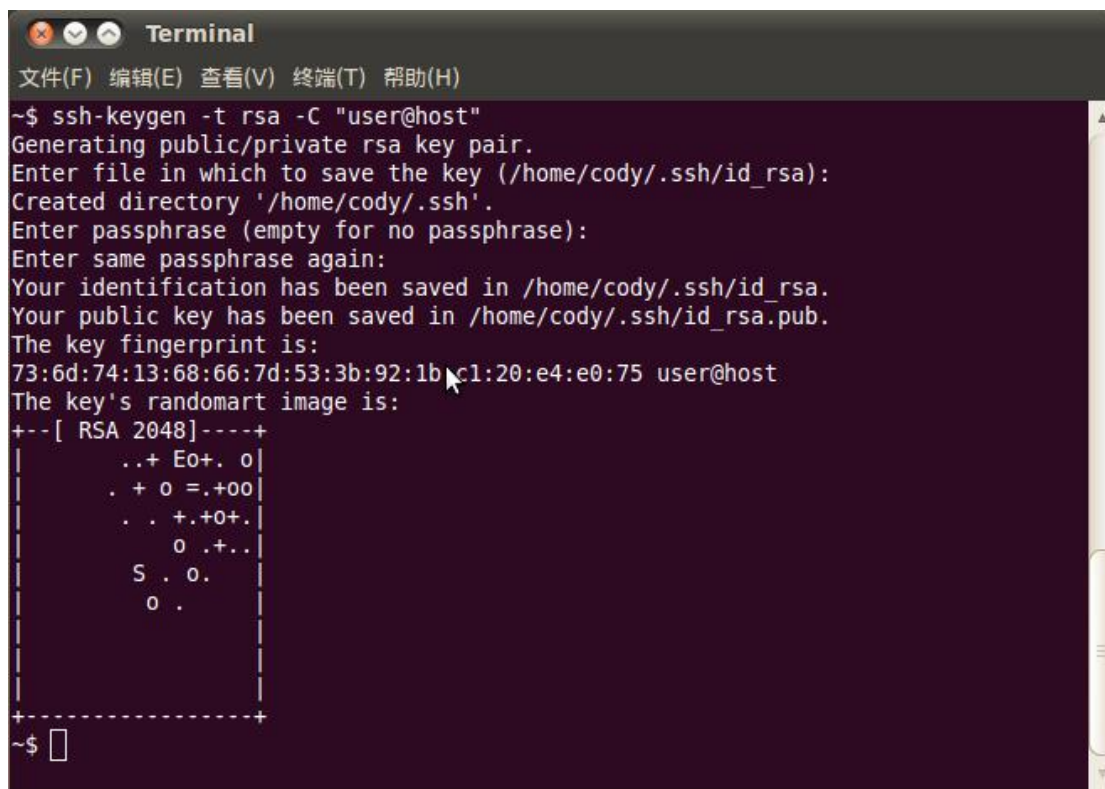
Chapter 10   SSH Public Key Operation Instruction

10.1 SSH Public Key Generation

Use the following command to generate:：
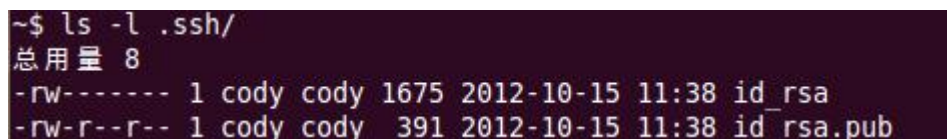
ssh-keygen -t rsa -C "user@host"

Please replace user@host with your email address

A public key file will be generated in your directory after running the command.

Please keep the generated private key file id_rsa and password properly, and email the public key id_rsa.pub to SDK server administrator.

10.2 Use key-chain to Manage Keys

It is recommended to use a simple tool keychain to manage keys.

The detailed usage is as follows:

1. Install keychain package:

$sudo aptitude install keychain

2. Configure the key:

$vim ~/.bashrc

Add the following line:

eval `keychain --eval ~/.ssh/id_rsa`

id_rsa is the private key file name.

After the above configurations, log in to console again and you will be prompted to enter password. Just enter password used to generate the key. If there is no password, you can skip it.

In addition, try not to use sudo or root users unless you know how to handle, otherwise it will lead to permission and key management confusion.

10.3 Multiple Machines Use The Same SSH Public Key

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

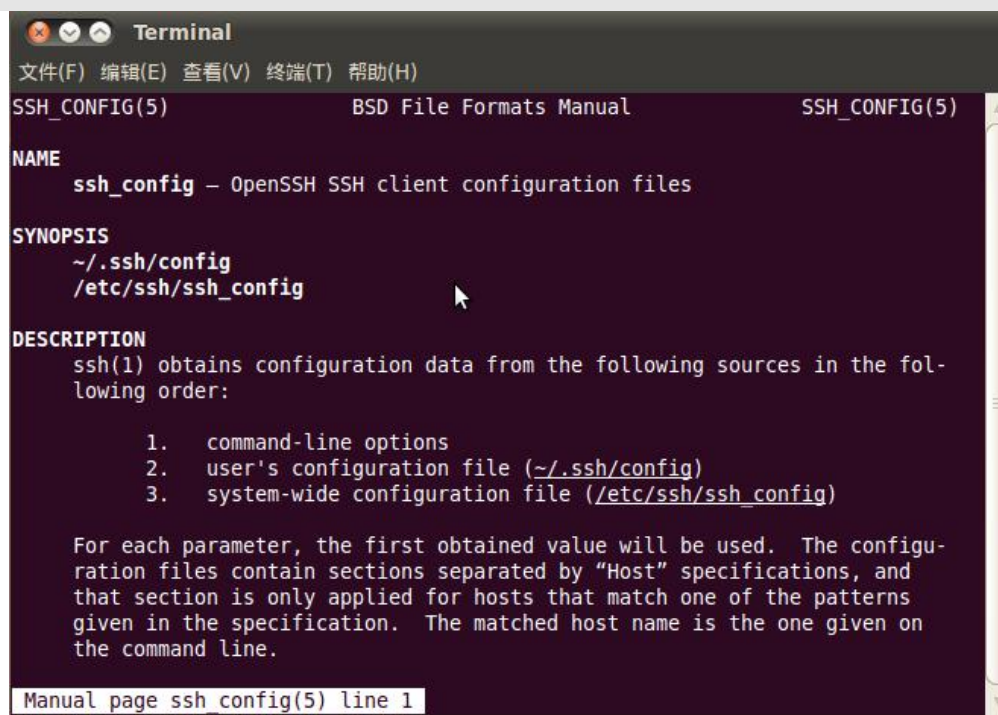Agent admitted failture to sign using the key

Enter the following command in console to solve

ssh-add ~/.ssh/id_rsa

10.4 One Machine Switches Different SSH Public Keys

You can configure SSH by referring to ssh_config documentation.

~$ man ssh_config

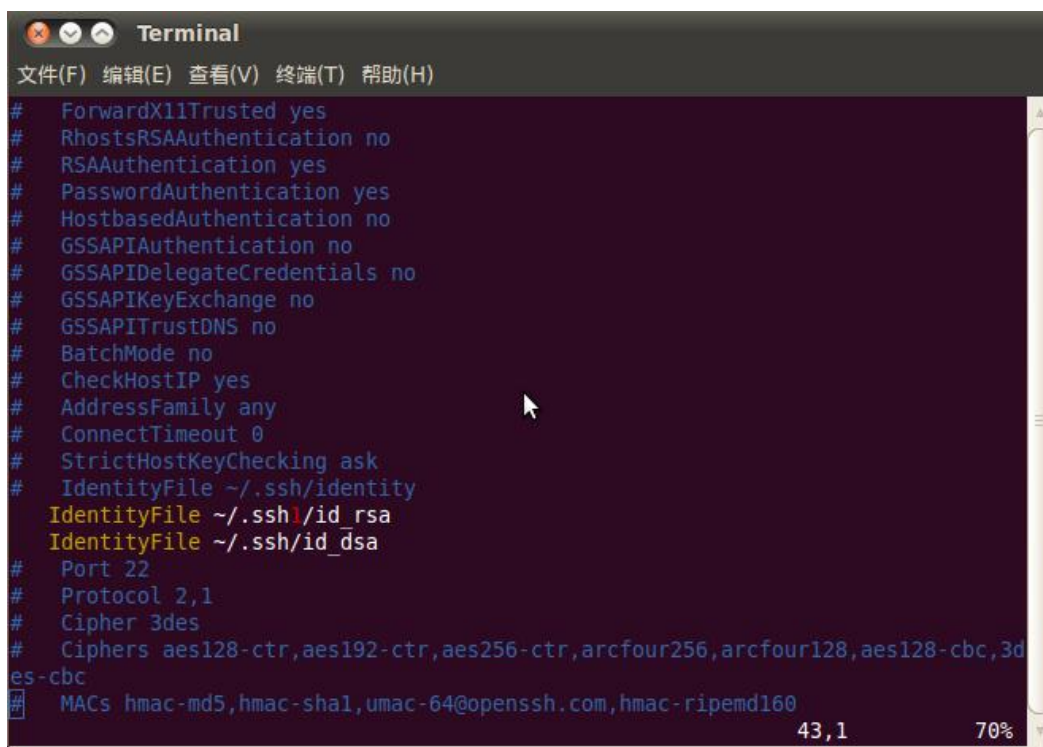Run the following command to configure SSH configuration of current user.

~$ 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.

10.5 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.

10.6 Git Access Application Instruction

Please email the public key file created according to the above chapter to fae@rock-chips.com, to apply for SDK code download permission.