

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

RK3399 Linux SDK Release Note

(Technical Department, Dept.III)

Status: [] draft [] Modifying [<input checked="" type="checkbox"/>] Released	Document ID:	RK-FB-CS-002
	Version:	2.2.1
	Author:	Caesar Wang
	Date:	2019-10-14
	Reviewer:	Eddie Cai
	Date:	2019-10-14

Revision History

Revision Date	Version No.	Revision Description	Author	Reviewer
2017-01-16	V1.0.0	Initial version	Guochun Huang	Yuanbin Lan
2017-02-27	V1.1.0	Add linux pc download tools	Guochun Huang	Yuanbin Lan
2017-06-08	V1.2.0	U-boot release branch	Guochun Huang	Yuanbin Lan
2018-04-08	V1.3.0	Merge U-boot of Android and Linux	Caesar Wang	Eddie Cai
2018-04-11	V1.4.0	SDK obtaining instruction	Caesar Wang	Eddie Cai
2018-04-18	V1.5.0	Modify some wrong words and repository address	Caesar Wang	Eddie Cai
2018-05-17	V2.0.0	1. Buildroot and debian documents integration 2. Add ssh public key instruction	Caesar Wang	Eddie Cai
2018-05-24	v2.0.1	1. Fix the command to compile debian 2. Fix mkrnling incompatibility issue 3. Add recovery	Caesar Wang	Eddie Cai
2018-06-04	V2.0.2	1. Display frame switch to wayland 2. Support uboot boot logo	Caesar Wang	Eddie Cai
2018-06-11	V2.0.3	1. Solve keyboard abnormal problem 2. boot.img replaces resource.img and kernel.img	Caesar Wang	Eddie Cai
2018-07-04	V2.0.4	1. buildroot upgrade to 2018.02.rc3 2. Add warranty disclaimer	Caesar Wang	Eddie Cai
2018-07-19	V2.0.5	1. update the fully automatic compiling method 2. Add debian64 instruction	Caesar Wang	Eddie Cai
2018-08-03	V2.0.6	1. Update dependent libraries 2. Fix the problem when compile debian	Caesar Wang	Eddie Cai
2018-09-07	V2.0.7	Content update to match current version	Caesar Wang	Eddie Cai
2018-09-29	V2.0.8	Update Debian firmware generation instructions	Caesar Wang	Eddie Cai
2018-11-02	V2.0.9	1. Fix compiling rootfs, app instruction 2. SDK v2.09 firmware update	Caesar Wang	Eddie Cai
2019-01-24	V2.1.0	1. Rename project rootfs to debian 2. Uboot config changed from evb_rk3399_defconfig to rk3399_defconfig 3. Added rk_tee and security part	Caesar Wang	Eddie Cai
2019-06-28	V2.2.0	1. Add Yocto instructions 2. EVB renamed to excavator	Caesar Wang	Eddie Cai
2019-10-14	v2.2.1	Update Debian 64bit build rule Update 9.6 content	Caesar Wang	Eddie Cai

Contents

Chapter 1	Overview	4
Chapter 2	Main Functions.....	4
Chapter 3	How to Obtain the SDK.....	4
Chapter 4	Software Development Guide.....	5
4.1	Development Guide	5
4.2	Software Update History.....	5
Chapter 5	RK3399_Linux Project Directory Introduction	6
Chapter 6	SDK Compiling Instruction	6
6.1	Compile U-Boot.....	7
6.2	Kernel Compiling Steps	7
6.3	Recovery Compiling Steps.....	7
6.4	Buildroot Rootfs and APP Compiling.....	7
6.5	Compile Debian Rootfs.....	8
6.5.1	Building Base Debian System.....	8
5.5.2	Building rk-debian Rootfs.....	8
5.5.3	Creating the ext4 Image(linaro-rootfs.img)	8
6.6	Yocto Rootfs Compiling	9
6.7	Fully Automatically Compiling	9
6.8	Firmware Package Steps	10
Chapter 7	Upgrade Instruction.....	11
7.1	Windows Upgrade Instruction	11
7.2	Linux Upgrade Instruction	12
7.3	System Partition Instruction.....	12
Chapter 8	RK3399 SDK Firmware	13
Chapter 9	SSH Public Key Operation Instruction.....	13
9.1	SSH Public Key Generation.....	14
9.2	Use Key-chain to Manage Keys.....	14
9.3	Multiple Machines Use the Same SSH Public Key	15
9.4	One Machine Switches Different SSH Public Keys	15
9.5	Key Authority Management.....	16
9.6	Reference document.....	17

Warranty Disclaimer

This document is provided according to “current situation” and Fuzhou Rockchip Electronics Co., Ltd. (“the company”, the same below) is not responsible for providing any express or implied statement or warranty of accuracy, reliability, completeness, marketability, specific purpose or non-infringement of any statement, 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 Co., 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.

Beyond reasonable use range, 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 without the written permission.

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.: +86-4007-700-590

Customer service Fax: +86-591-83951833

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

Chapter 1 Overview

This SDK is based on linux Buildroot, Ycoto and Debian 9 with kernel 4.4. It is applicable to the development of RK3399 excavator and all other Linux products based on it.

This SDK supports VPU hardware decoding, GPU 3D, Wayland display, QT and other function. For detailed functions 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 the SDK

SDK is released by Rockchip server or obtain from Github open source website. Please refer to [Chapter 6 SDK Compiling Instruction](#) to build a development environment.

First method to obtain SDK: get source code from Rockchip code server:

To get RK3399 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 9 SSH Public Key Instruction](#).

RK3399_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 rk3399_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 the 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 rk3399_linux_sdk_release_v2.2.0_20190628.tgz as an example. After copying initialization package, you can get source code by running the following command:

```
mkdir rk3399
tar xvf rk3399_linux_sdk_release_v2.2.0_20190628.tgz -C rk3399
cd rk3399
.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 rk3399 linux work directory:

```
mkdir rk3399_linux
```

Enter the rk3399 linux work directory

```
cd rk3399_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 rk3399_lin  
ux_release.xml  
Synchronize the whole project:  
../repo/repo sync
```

Chapter 4 Software Development Guide

4.1 Development Guide

RK3399 Linux SDK Kernel version is Linux4.4, Rootfs are Buidlroot (2018.02-rc3), Yocto(thud 2.6)and Debian9 respectively. To help engineers get familiar with SDK development and debugging more quickly, “Rockchip Linux Software Development Guide” is released with SDK. It can be obtained in the “docs/” directory and will be continuously updated.

4.2 Software Update History

The software release version is upgraded through the project xml, please refer to the below method for details:

```
.repo/manifests$ ls -l -h rk3399_linux_release.xml
```

Updated information can be viewed through project text by below method:

```
.repo/manifests$ cat rk3399_linux_v2.00/RK3399_Release_Note.txt
```

Or refer to the project directory:

```
docs/SoC platform related/RK3399/RK3399_Release_Note.pdf
```

Chapter 5 RK3399_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 apps like Camera/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) prebuilts: store cross-compilation toolchain.
- 9) recovery: store recovery project files.
- 10) rkbin: store firmware and tools.
- 11) rockdev: store compiled output firmware.
- 12) tools: store some commonly used tools.
- 13) u-boot: uboot code.
- 14) yocto: yocto root file system.

Chapter 6 SDK Compiling Instruction

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-gnueabi
bihf u-boot-tools device-tree-compiler gcc-aarch64-linux-gnu mtools
parted libudev-dev libusb-1.0-0-dev python-linaro-image-tools lina
ro-image-tools autoconf autotools-dev libsigsegv2 m4 intltool libdr
m-dev curl sed make binutils build-essential gcc g++ bash patch gzi
p bzip2 perl tar cpio python unzip rsync file bc wget libncurses5 l
ibqt4-dev libglib2.0-dev libgtk2.0-dev libglade2-dev cvs git mercur
ial rsync openssh-client subversion asciidoc w3m dlatex graphviz p
ython-matplotlib libc6:i386 libssl-dev texinfo liblz4-tool genext2f
s
```

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-gnueabi
bihf u-boot-tools device-tree-compiler gcc-aarch64-linux-gnu mtools
parted libudev-dev libusb-1.0-0-dev python-linaro-image-tools lina
ro-image-tools gcc-4.8-multilib-arm-linux-gnueabi gcc-arm-linux-g
nueabi libssl-dev gcc-aarch64-linux-gnu g++ conf autotools-dev lib
sigsegv2 m4 intltool libdrm-dev curl sed make binutils build-essent
ial gcc g++ bash patch gzip bzip2 perl tar cpio python unzip rsync
file bc wget libncurses5 libqt4-dev libglib2.0-dev libgtk2.0-dev li
bglade2-dev cvs git mercurial rsync openssh-client subversion asci
idoc w3m dlatex graphviz python-matplotlib libc6:i386 libssl-dev te
xinfo 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-gnueabi)

6.1 Compile U-Boot

Enter project U-Boot directory and execute make.sh to get rk3399_loader_v1.22.119.bin trust.img uboot.img:

Rk3399 excavator development boards:

```
./make.sh rk3399
```

Rk3399 Firefly development board:

```
./make.sh firefly-rk3399
```

Generated files are in u-boot directory after compilation

```
u-boot/
├── rk3399_loader_v1.17.115.bin
├── trust.img
└── uboot.img
```

6.2 Kernel Compiling Steps

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

Rk3399 excavator v11 development boards:

```
cd kernel
make ARCH=arm64 rockchip_linux_defconfig
make ARCH=arm64 rk3399-sapphire-excavator-linux.img -j12
```

Rk3399 excavator v10 development boards:

```
cd kernel
make ARCH=arm64 rockchip_linux_defconfig
make ARCH=arm64 rk3399-sapphire-excavator-v10-linux.img -j12
```

Rk3399 Firefly development boards:

```
cd kernel
make ARCH=arm64 rockchip_linux_defconfig
make ARCH=arm64 rk3399-firefly-linux.img -j12
```

The boot.img includes Image and DTB of kernel will be generated after compiling in the kernel directory.

6.3 Recovery Compiling Steps

Enter project root directory and execute the following command to automatically complete compiling and packaging of Recovery.

Rk3399 excavator /firefly development boards:

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

The recovery.img is generated in Buildroot directory “output/rockchip_rk3399_recovery/images” after compiling.

6.4 Buildroot Rootfs and APP Compiling

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

Rk3399 excavator development boards:

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

After compiling, rootfs.ext4 is generated in Buildroot directory “output/rockchip_rk3399/images”.

Note:

If you need to compile a single module or a third-party application, you need to configure the cross-compilation environment.

Cross-compilation tool is located in “buildroot/output/rockchip_rk3399/host/usr” directory. You need to set the “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 is printed, configuration is successful:

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

6.5 Compile Debian Rootfs

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

Or enter “debian/” directory:

```
cd debian/
```

Please refer to the readme.md in the directory, detailed steps are as follows:

6.5.1 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 timestamp generated) 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:
.../debian/ubuntu-build-service/stretch-desktop-armhf/chroot/test
-dev-null: Permission denied E: Cannot install into target '/home/f
oxluo/work3/rockchip/rk_linux/rk3399_linux/denian/ubuntu-build-serv
ice/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 compiling issues, Please check firstly that the compiler system is not ext2/ext4.

5.5.2 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.sh
```

(A “debug” behind is recommended in the development process)

5.5.3 Creating the ext4 Image(linaro-rootfs.img)

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

Will generate linaro-rootfs.img.

6.6 Yocto Rootfs Compiling

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

rk3399 excavator development boards:

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

After compiling, the 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

6.7 Fully Automatically 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
```

It is buildroot by default, you can specify rootfs by setting the environment variable

RK_ROOTFS_SYSTEM.

For example, if you need Yocto, you can generate by the following commands:

```
$export RK_ROOTFS_SYSTEM=yocto
```

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

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

```
rk3399$ ./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/rockchip/Boardconfig.mk.

Main configurations of rk3399 excavator are as follows:

```
# Target arch
export RK_ARCH=arm64
# Uboot defconfig
export RK_UBOOT_DEFCONFIG=rk3399
# Kernel defconfig
export RK_KERNEL_DEFCONFIG=rockchip_linux_defconfig
# Kernel dts
export RK_KERNEL_DTS=rk3399-sapphire-excavator-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_rk3399
# Recovery config
export RK_CFG_RECOVERY=rockchip_rk3399_recovery
# ramboot config
export RK_CFG_RAMBOOT=
# Pcba config
export RK_CFG_PCBA=rockchip_rk3399_pcba
```

6.8 Firmware Package Steps

After compiling various parts of Kernel/Uboot/Recovery/Rootfs above, enter root directory of project directory and execute 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 7 Upgrade Instruction

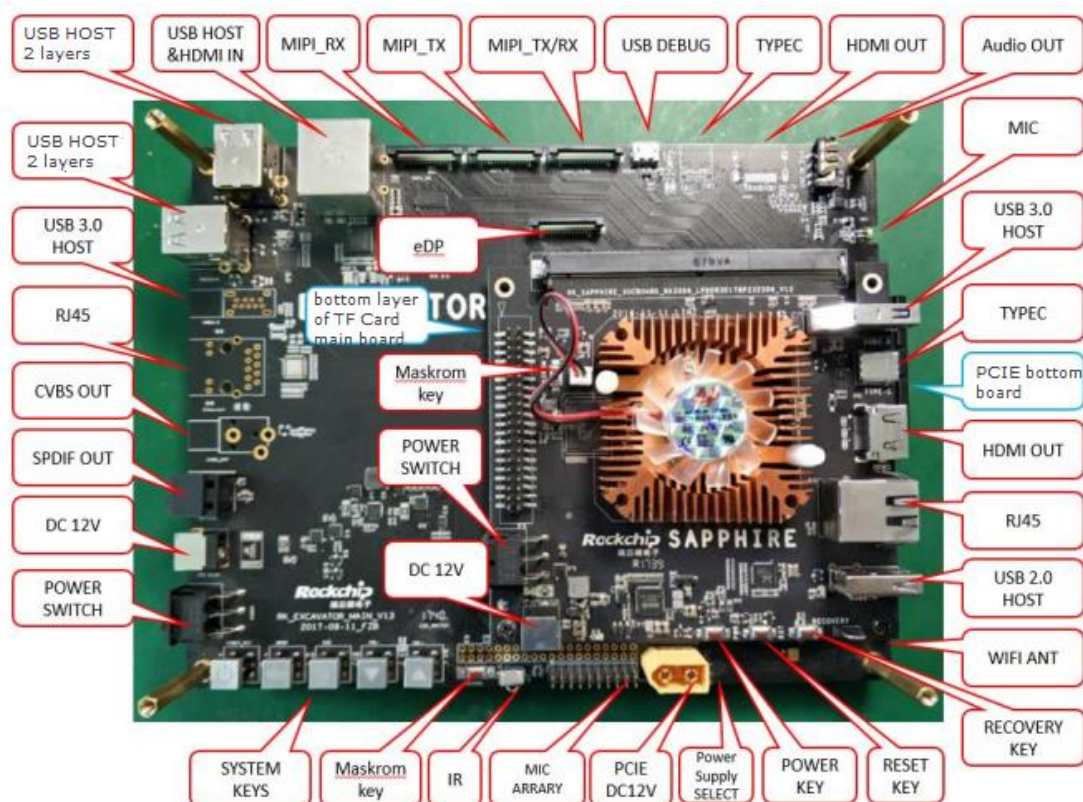


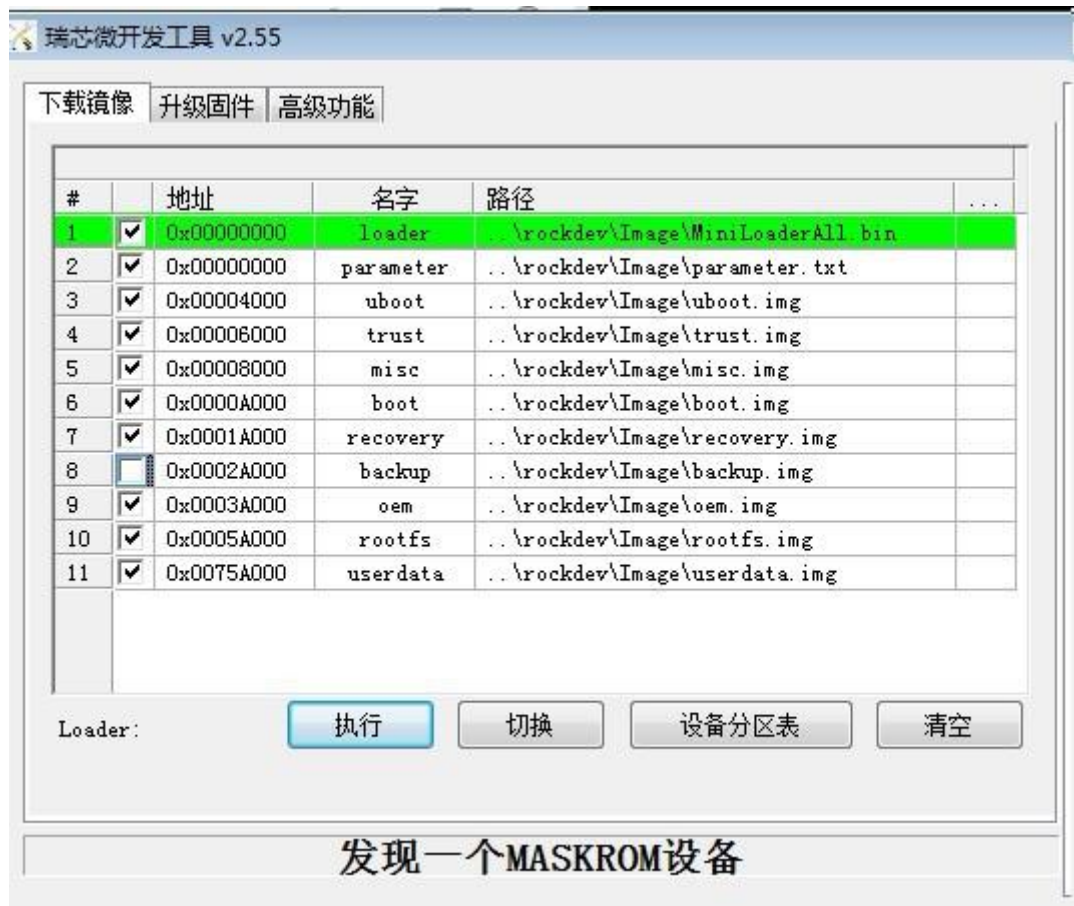
Figure 1 RK3399 excavator

7.1 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 needs to enter MASKROM mode for upgrade. 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 update. You can also press the "recovery" button and press reset button "RST" then release to enter loader mode to update. Partition offset and update files of MASKROM Mode are shown as follows (Note: if the PC is Windows 7, Windows 8 or Windows 10 system, you 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_v4.7.zip

7.2 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         rockdev/rootfs.img
sudo ./upgrade_tool di -userdata       rockdev/userdata.img
sudo ./upgrade_tool rd

```

Or in root directory, run the following command on your machine to upgrade in maskrom state:
./rkflash.sh

7.3 System Partition Instruction

Default partition (below is RK3399 excavator 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	172031	32.0M	0700	recovery
6	172032	237567	32.0M	0700	backup
7	237568	368639	64.0M	0700	oem
8	368640	12951551	6049M	0700	rootfs
9	12951552	15269854	1132M	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 compiled by recovery.

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. Mounted in /oem directory

rootfs partition: store rootfs.img compiled by buildroot or debian.

userdata partition: store files temporarily generated by app or for users. mounted in /userdata directory.

Chapter 8 RK3399 SDK Firmware

RK3399_LINUX_SDK_V2.3.0_20191014 firmware download link is as below(including Debian and Buildroot and Yocto firmwares):

RK3399 excavator:

Buildroot: <https://eyun.baidu.com/s/3ggdSnFt>
 Yocto: <https://eyun.baidu.com/s/3o9XiCSM>
 Debian: <https://eyun.baidu.com/s/3edCGYa>

RK3399 Firrefly:

Buildroot: <https://eyun.baidu.com/s/3kWycPtt>
 Yocto: <https://eyun.baidu.com/s/3snlVBbB>
 Debian: <https://eyun.baidu.com/s/3dX7bYm>

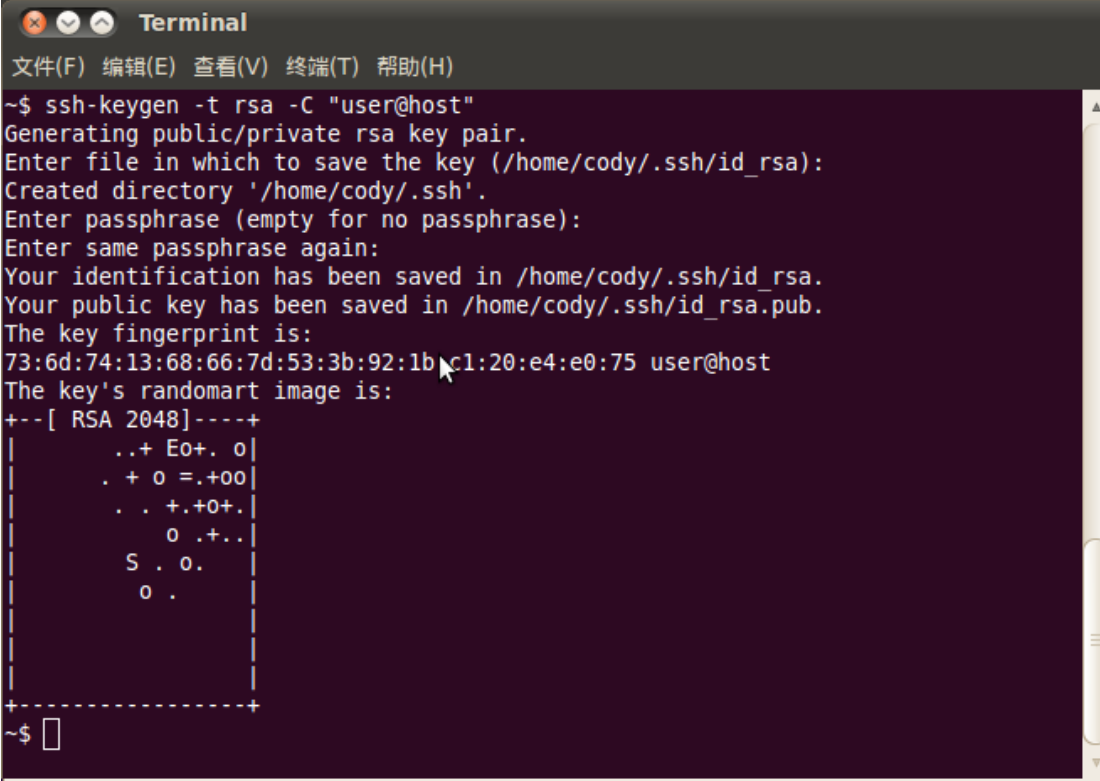
Chapter 9 SSH Public Key Operation Instruction

9.1 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



```

Terminal
文件(F) 编辑(E) 查看(V) 终端(T) 帮助(H)
~$ ssh-keygen -t rsa -C "user@host"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/cody/.ssh/id_rsa):
Created directory '/home/cody/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/cody/.ssh/id_rsa.
Your public key has been saved in /home/cody/.ssh/id_rsa.pub.
The key fingerprint is:
73:6d:74:13:68:66:7d:53:3b:92:1b:1c:1:20:e4:e0:75 user@host
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      .+ Eo+. o|
|    . + o =.+oo|
|   . . +.+o+. |
|      o .+..   |
|     S . o.    |
|      o .      |
+-----+
~$ 

```

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

```

~$ ls -l .ssh/
总用量 8
-rw----- 1 cody cody 1675 2012-10-15 11:38 id_rsa
-rw-r--r-- 1 cody cody 391 2012-10-15 11:38 id_rsa.pub

```

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

9.2 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 configuration, 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.

9.3 Multiple Machines Use the Same SSH Public Key

If the same SSH public key needs to 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 failure to sign using the key

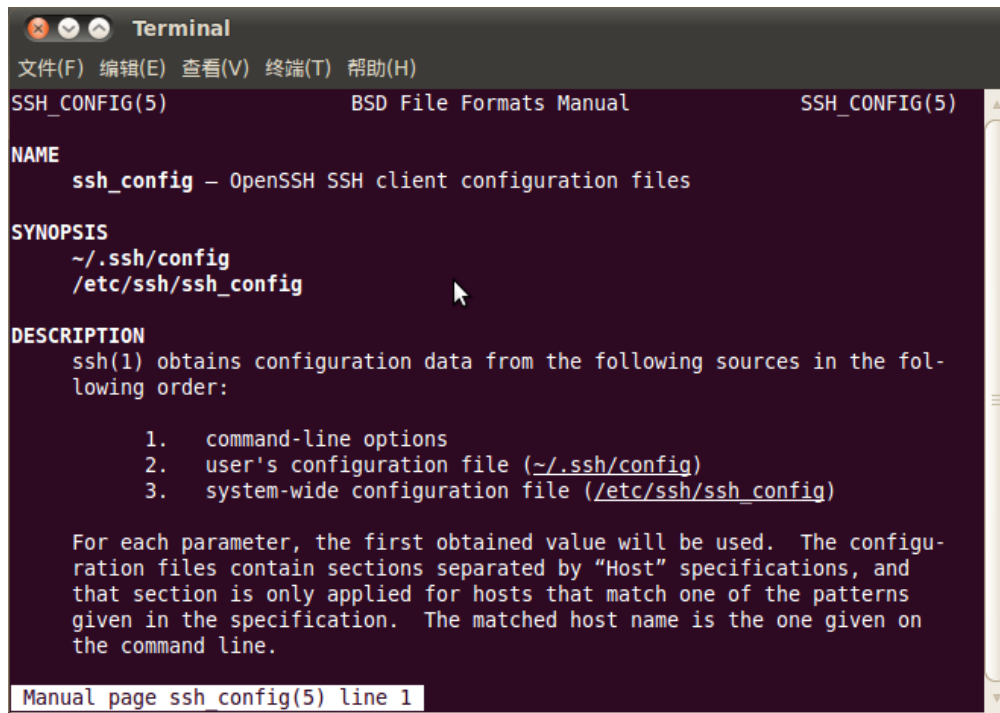
Enter the following command in console to solve

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

9.4 One Machine Switches Different SSH Public Keys

You can configure SSH by referring to ssh_config documentation.

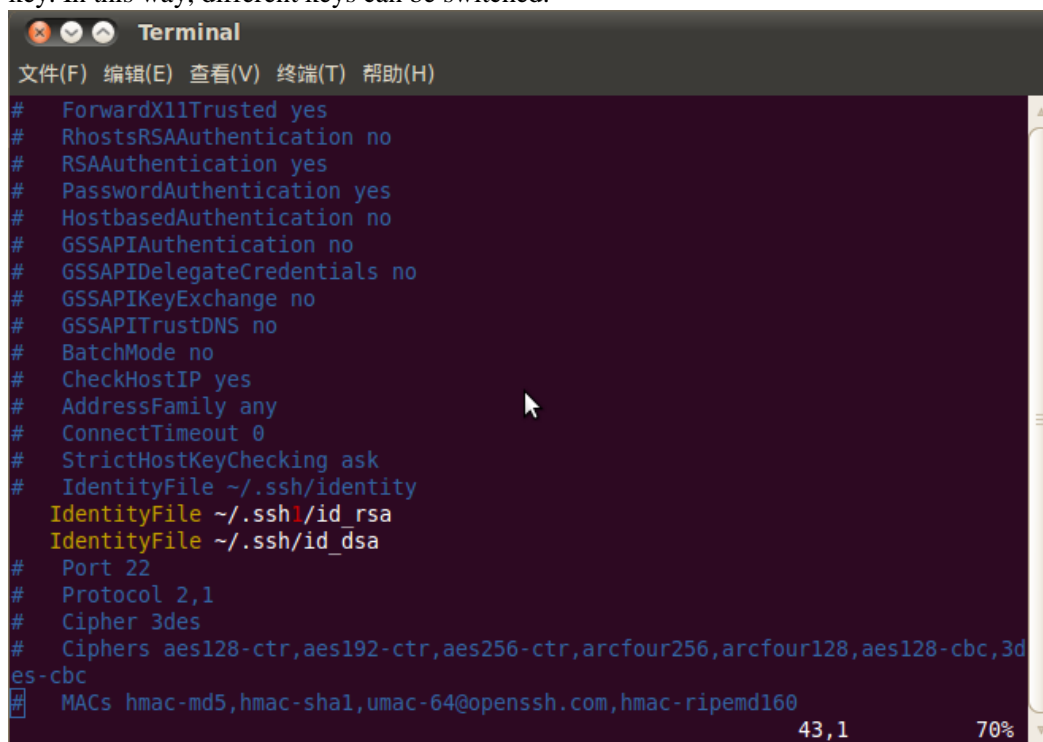
```
~$ man ssh_config
```

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.



9.5 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.

9.6 Reference document

For more details, refer to document `sdk/docs/RKTools manuals/Rockchip SDK Kit 申请指南V1.6-201905.pdf`.