

Rockchip RK3399 Linux SDK Release Note

ID: RK-FB-CS-002

Release Version: V2.5.0

Release Date: 2020-10-13

Security Level: ☐Top-Secret ☐Secret ☐Internal ☒Public

DISCLAIMER

THIS DOCUMENT IS PROVIDED "AS IS". ROCKCHIP ELECTRONICS CO., LTD. ("ROCKCHIP") DOES NOT PROVIDE ANY WARRANTY OF ANY KIND, EXPRESSED, IMPLIED OR OTHERWISE, WITH RESPECT TO THE ACCURACY, RELIABILITY, COMPLETENESS, MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE OR NON-INFRINGEMENT OF ANY REPRESENTATION, INFORMATION AND CONTENT IN THIS DOCUMENT. THIS DOCUMENT IS FOR REFERENCE ONLY. THIS DOCUMENT MAY BE UPDATED OR CHANGED WITHOUT ANY NOTICE AT ANY TIME DUE TO THE UPGRADES OF THE PRODUCT OR ANY OTHER REASONS.

Trademark Statement

"Rockchip", "瑞芯微", "瑞芯" shall be Rockchip's registered trademarks and owned by Rockchip. All the other trademarks or registered trademarks mentioned in this document shall be owned by their respective owners.

All rights reserved. ©2020. Rockchip Electronics Co., Ltd.

Beyond the scope of fair use, neither any entity nor individual shall extract, copy, or distribute this document in any form in whole or in part without the written approval of Rockchip.

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

The document presents Rockchip RK3399 Linux SDK release notes, aiming to help engineers get started with RK3399 Linux SDK development and debugging faster.

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers

Software development engineers

Chipset and System Support

Chipset	Buildroot	Debian 9	Debian 10	Yocto
RK3399	Y	Y	Y	Y

Revision History

Date	Version	Author	Revision History
2017-01-16	V1.0.0	Guochun Huang	Initial version
2017-02-27	V1.1.0	Guochun Huang	Add Linux PC download tools
2017-06-08	V1.2.0	Caesar Wang	An official release version, adds NPU related instructions. Add Yocto building and github download instructions.
2018-04-08	V1.3.0	Caesar Wang	Update the name of software develop guide
2018-04-11	V1.4.0	Caesar Wang	Update Debian building instructions.
2018-04-18	V1.5.0	Caesar Wang	Fix some mistaken words and repository address
2018-05-17	V2.0.0	Caesar Wang	Integrate Buildroot and Debian documents Add SSH public key operation introduction
2019-01-24	V2.1.0	Caesar Wang	Rename project rootfs chapter to Debian update U-boot config
2019-06-28	V2.2.0	Caesar Wang	Add Yocto introduction ; EVB renamed to excavator
2019-12-03	V2.3.0	Caesar Wang	Update Debian 64 bit building Updated chapters 1, 2, 3 and 9.6 Updated Chapter 5 SDK Directory Introduction Update Chapter 6 Debian10 Building。
2020-04-30	V2.4.0	Caesar Wang	Rewrite the document with Markdown Add and use RK3399 EVB IND by default。
2020-7-22	V2.4.1	Ruby Zhang	Update the company name, the format and the file name of the document
2020-10-13	V2.5.0	Ruby Zhang	Compilation rules for adapting to new version

Contents

Rockchip RK3399 Linux SDK Release Note

1. Overview
2. Main Functions
3. How to Get the SDK
 - 3.1 General RK3399 Linux SDK Obtain
 - 3.1.1 Get Source Code from Rockchip Code Server
 - 3.1.2 Get Source Code from Local Compression Package
4. Software Development Guide
 - 4.1 Software Update History
5. Hardware Development Guide
6. SDK Project Directory Introduction
7. SDK Building Introduction
 - 7.1 SDK Dependency Packages Installation
 - 7.2 SDK Board Level Configuration
 - 7.3 Compilation Commands
 - 7.4 Automatic Build
 - 7.5 Build and package each module
 - 7.5.1 U-boot Build
 - 7.5.2 Kernel Build
 - 7.5.3 Recovery Build
 - 7.5.4 Buildroot Build
 - 7.5.4.1 Buildroot Cross Compilation
 - 7.5.4.2 Build Modules in Buildroot
 - 7.5.5 Debian 9 Build
 - 7.5.6 Debian 10 Build
 - 7.5.7 Yocto Build
 - 7.5.7.1 Firmware Package
8. Upgrade Introduction
 - 8.1 Windows Upgrade Introduction
 - 8.2 Linux Upgrade Instruction
 - 8.3 System Partition Introduction
9. RK3399 SDK Firmware
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 Buildroot 2018.02-rc3, Yocto Thud 3.0, Debian 9 and Debian 10 or later version with kernel 4.4 and U-boot v2017.09. It is suitable for RK3399 EVB development boards and all other Linux products developed based on it. This SDK supports VPU hardware decoding, GPU 3D, Wayland/X11 display, Qt and other function. For detailed functions debugging and interface introductions, please refer to the documents under the project's docs/ directory.

2. Main Functions

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

3. How to Get the SDK

The SDK is released by Rockchip server. Please refer to Chapter 7 [SDK Compilation Introduction](#) to build a development environment.

3.1 General RK3399 Linux SDK Obtain

3.1.1 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 10 [SSH Public Key Operation Introduction](#).

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
```

3.1.2 Get Source Code from Local Compression Package

For quick access to SDK source code, Rockchip Technical Window usually provides corresponding version of SDK initial compression package. In this way, developers can get SDK source code through decompressing the initial compression package, which is the same as the one downloaded by repo.

Take rk3399_linux_sdk_release_v2.5.0_20201013.tgz as an example. After getting a initialization package, you can get source code by running the following command:

```
mkdir rk3399
tar xvf rk3399_linux_sdk_release_v2.5.0_20201013.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.

4. Software Development Guide

4.1 Software Update History

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

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

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

```
.repo/manifests$ cat rk3399_linux/RK3399_Linux_SDK_Release_Note.txt
```

Or refer to the project directory:

```
<SDK>/docs/RK3399/RK3399_Linux_SDK_Release_Note.txt
```

5. Hardware Development Guide

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

RK3399 excavator hardware development guide:

```
<SDK>/docs/RK3399/Rockchip_RK3399_User_Manual_Sapphire_EVB_V3.0_CN.pdf
```

RK3399 IND industry board hardware development guide:

6. SDK Project Directory Introduction

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

- app: store application APPs like qcamera/qfm/qplayer/qseting and other applications.
- buildroot: root file system based on Buildroot (2018.02-rc3).
- debian: root file system based on Debian 9.
- device/rockchip: store board-level configuration for each chip and some scripts and prepared files for compiling and packaging firmware.
- docs: stores development guides, platform support lists, tool usage, Linux development guides, and so on.
- distro: a root file system based on Debian 10.
- IMAGE: stores compilation time, XML, patch and firmware directory for each compilation.
- external: stores some third-party libraries, including audio, video, network, recovery and so on.
- kernel: stores kernel4.4 development code.
- npu: store npu code.
- prebuilts: stores cross-compilation toolchain.
- rkbin: stores Rockchip Binary and tools.
- rockdev: stores compiled output firmware.
- tools: stores some commonly used tools under Linux and Windows system.
- u-boot: store U-Boot code developed based on v2017.09 version.
- yocto: stores the root file system developed based on Yocto Thud 3.0.

7. SDK Building Introduction

7.1 SDK Dependency Packages Installation

This SDK is developed and tested on Ubuntu system. We recommend using Ubuntu 18.04 for compilation. Other Linux versions may need to adjust the software package accordingly. In addition to the system requirements, there are other hardware and software requirements.

Hardware requirements: 64-bit system, hard disk space greater than 40G. If you do multiple builds, you will need more hard drive space

Software requirements: Ubuntu 18.04 system:

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

```
repo git ssh make gcc libssl-dev liblz4-tool expect g++ patchelf chrpath gawk
texinfo chrpath diffstat binfmt-support qemu-user-static live-build bison flex
fakeroot cmake
```

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

7.2 SDK Board Level Configuration

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

Board level configuration	Note
BoardConfig-rk3399-evb-ind-lpddr4.mk	Suitable for RK3399 industry development board
BoardConfig-rk3399-firefly.mk	Suitable for RK3399 firefly development boards
BoardConfig-rk3399-sapphire-excavator-lp4.mk	Suitable for RK3399 sapphire excavator with lpddr4 development board
BoardConfig-rk3399-sapphire-excavator.mk	Suitable for RK3399 sapphire excavator development board

The first way:

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

Select the board configuration of **RK3399 industry development board**:

```
./build.sh device/rockchip/rk3399/BoardConfig-rk3399-evb-ind-lpddr4.mk
```

Select the board configuration of the **RK3399 firefly development board**:

```
./build.sh device/rockchip/rk3399/BoardConfig-rk3399-firefly.mk
```

Select the board-level configuration of the **RK3399 sapphire excavator with lpddr4 development board**:

```
./build.sh device/rockchip/rk3399/BoardConfig-rk3399-sapphire-excavator-lp4.mk
```

Select the board-level configuration of the **RK3399 sapphire excavator development board**:

```
./build.sh device/rockchip/rk3399/BoardConfig-rk3399-sapphire-excavator.mk
```

The second way:

```
rk3399$ ./build.sh lunch
processing option: lunch

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

0. default BoardConfig.mk
1. BoardConfig-rk3399-evb-ind-lpddr4.mk
2. BoardConfig-rk3399-firefly.mk
3. BoardConfig-rk3399-sapphire-excavator-lp4.mk
4. BoardConfig-rk3399-sapphire-excavator.mk
5. BoardConfig.mk
Which would you like? [0]:
...
```


7.3 Compilation Commands

Execute the command in the root directory: `./build.sh -h|help`

```
rk3399pro$ ./build.sh -h
Usage: build.sh [OPTIONS]
Available options:
BoardConfig*.mk    -switch to specified board config
lunch               -list current SDK boards and switch to specified board config
uboot               -build uboot
spl                 -build spl
loader              -build loader
kernel              -build kernel
modules             -build kernel modules
toolchain           -build toolchain
rootfs              -build default rootfs, currently build buildroot as default
buildroot           -build buildroot rootfs
ramboot             -build ramboot image
multi-npu_boot      -build boot image for multi-npu board
yocto               -build yocto rootfs
debian              -build debian9 stretch rootfs
distro              -build debian10 buster rootfs
pcba                -build pcba
recovery            -build recovery
all                 -build uboot, kernel, rootfs, recovery image
cleanall            -clean uboot, kernel, rootfs, recovery
firmware            -pack all the image we need to boot up system
updateimg           -pack update image
otapackage          -pack ab update otapackage image
save                -save images, patches, commands used to debug
allsave             -build all & firmware & updateimg & save

Default option is 'allsave'.
```

View detailed build commands for some modules, for example: `./build.sh -h kernel`

```
rk3399$ ./build.sh -h kernel
###Current SDK Default [ kernel ] Build Command###
cd kernel
make ARCH=arm64 rockchip_linux_defconfig
make ARCH=arm64 rk3399-sapphire-excavator-linux.img -j12
```

7.4 Automatic Build

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

```

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

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

```

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

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

```

$export RK_ROOTFS_SYSTEM=debian
$./build.sh

```

7.5 Build and package each module

7.5.1 U-boot Build

```

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

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

```

7.5.2 Kernel Build

```

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

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

```

7.5.3 Recovery Build

```

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

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

```

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

7.5.4 Buildroot Build

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

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

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

7.5.4.1 Buildroot Cross Compilation

If you need to build a single module or a third-party application, you need to setup the cross compilation environment. Cross compilation tool is located in “buildroot/output/rockchip_rk3399/host/usr” directory. You need to set bin/ directory of tools and aarch64-buildroot-linux-gnu/bin/ directory to environment variables, and execute auto-configuration environment variable script in the top-level directory (only valid for current console):

```
source envsetup.sh
```

Enter the command to check:

```
cd buildroot/output/rockchip_rk3399/host/usr/bin  
./aarch64-linux-gcc --version
```

Then the following logs are printed:

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

7.5.4.2 Build Modules in Buildroot

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

- Build qplayer

```
SDK$make qplayer
```

- Rebuild qplayer

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

- delete qplayer

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

7.5.5 Debian 9 Build

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

Or enter debian/ directory:

```
cd debian/
```

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

(1) Building Base Debian System

```
sudo apt-get install binfmt-support qemu-user-static live-build  
sudo dpkg -i ubuntu-build-service/packages/*  
sudo apt-get install -f
```

Compile 64-bit Debian:

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

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

FAQ:

If you encounter the following problem during above compiling:

```
noexec or nodev issue /usr/share/debootstrap/functions: line 1450:  
....../rootfs/ubuntu-build-service/stretch-desktop-arm64/chroot/test-dev-null:  
Permission denied E: Cannot install into target  
...  
mounted with noexec or nodev
```

Solution:

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

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

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

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

```

+++ b/ubuntu-build-service/stretch-desktop-arm64/configure
@@ -11,6 +11,11 @@ set -e
echo "I: create configuration"
export LB_BOOTSTRAP_INCLUDE="apt-transport-https gnupg"
lb config \
+ --mirror-bootstrap "http://mirrors.163.com/debian" \
+ --mirror-chroot "http://mirrors.163.com/debian" \
+ --mirror-chroot-security "http://mirrors.163.com/debian-security" \
+ --mirror-binary "http://mirrors.163.com/debian" \
+ --mirror-binary-security "http://mirrors.163.com/debian-security" \
--apt-indices false \
--apt-recommends false \
--apt-secure false \

```

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

(2) Building rk-debian rootfs

Compile 64-bit Debian:

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

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

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

Will generate linaro-rootfs.img.

7.5.6 Debian 10 Build

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

Or enter distro/directory:

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

After building, the rootfs.ext4 will be generated in the distro directory “distro/output/images”.

Note: The current build of Debian10 Qt also depends on the build of Buildroot qmake, so please build Buildroot before building Debian10.

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

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

7.5.7 Yocto Build

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

RK3399Pro EVB boards:

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

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

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 to [setup-locale-python3](#). The image generated after compiling is in “yocto/build/latest/rootfs.img”. The default login username is root.

Please refer to [Rockchip Wiki](#) for more detailed information of Yocto.

7.5.7.1 Firmware Package

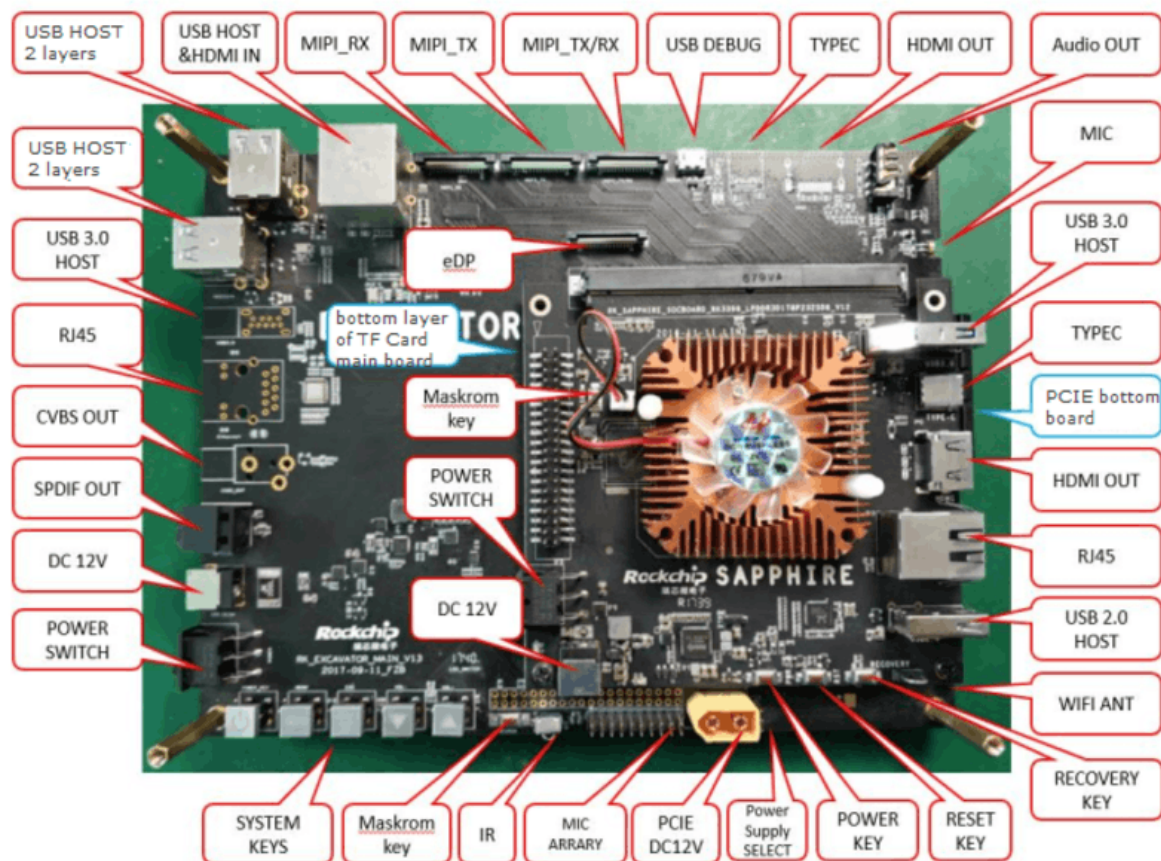
After compiling various parts of Kernel/U-Boot/Recovery/Rootfs above, enter root directory of project directory and run the following command to automatically complete all firmware packaged into rockdev directory:

Firmware generation:

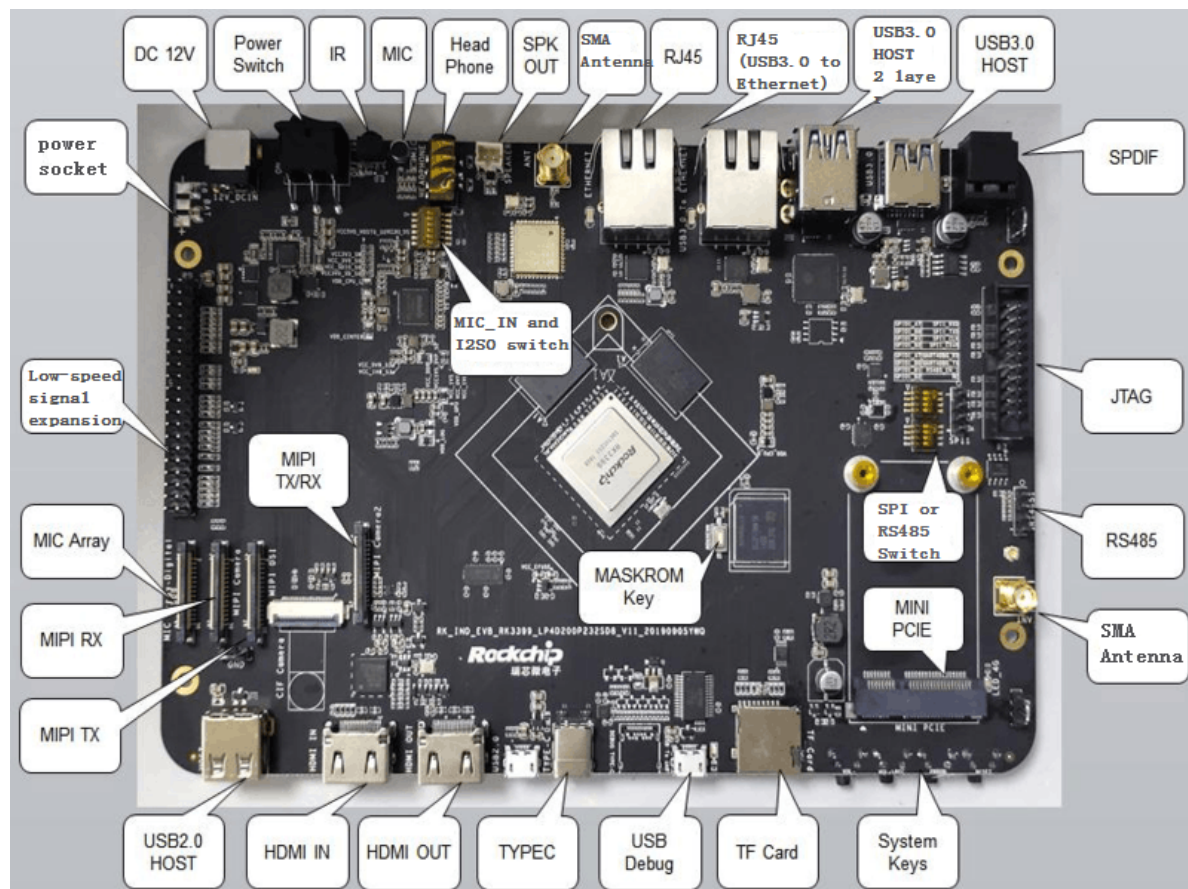
```
./mkfirmware.sh
```

8. Upgrade Introdution

Interfaces layout of RK3399 excavator are showed as follows:



Interfaces layout of RK3399 IND industry board are showed as follows:

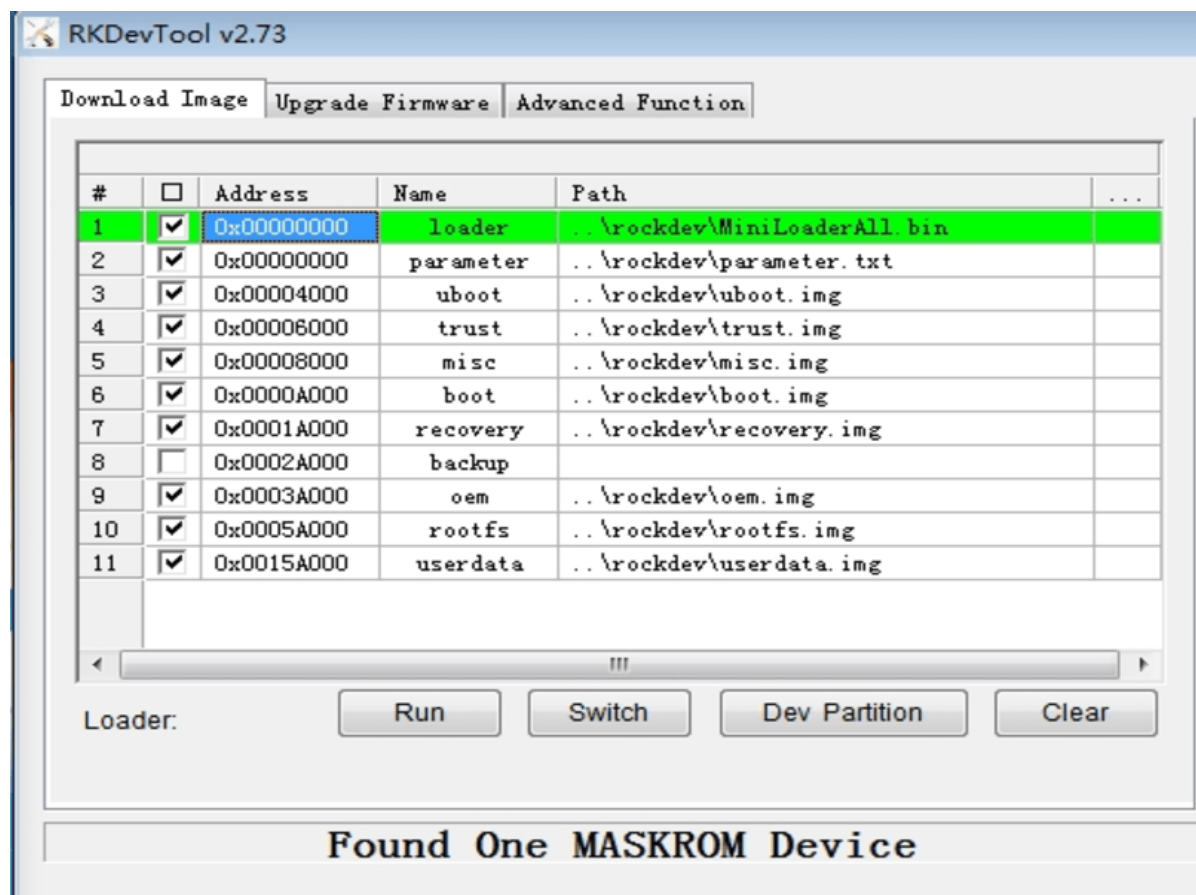


8.1 Windows Upgrade Introduction

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

```
tools/  
└─ windows/RKDevTool
```

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



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

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

8.2 Linux Upgrade Instruction

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


```

sudo ./upgrade_tool ul rockdev/MiniLoaderAll.bin
sudo ./upgrade_tool di -p rockdev/parameter.txt
sudo ./upgrade_tool di -u rockdev/uboot.img
sudo ./upgrade_tool di -t rockdev/trust.img
sudo ./upgrade_tool di -misc rockdev/misc.img
sudo ./upgrade_tool di -b rockdev/boot.img
sudo ./upgrade_tool di -recovery rockdev/recovery.img
sudo ./upgrade_tool di -oem rockdev/oem.img
sudo ./upgrade_tool di -rootfs rockdev/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 introduction (below is RK3399 IND reference partition):

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	32M	recovery
6	172032	237567	32M	bakcup
7	237568	368639	64M	oem
8	368640	12951551	6144M	rootfs
9	12951552	30535646	8585M	userdata

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

9. RK3399 SDK Firmware

RK3399_LINUX_SDK_V2.4.0_20200430 firmware download address is as follows (including Buildroot/Debian 9/Debian 10/Yocto firmwares):

- RK3399 IND industry development boards:

[Buildroot](#)

[Yocto](#)

[Debian9](#)

[Debian10](#)

- RK3399 excavator development boards:

[Buildroot](#)

[Yocto](#)

[Debian9](#)

[Debian10](#)

- RK3399 Firefly development boards:

[Buildroot](#)

[Yocto](#)

[Debian9](#)

[Debian10](#)

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 failure to sign using the key
```

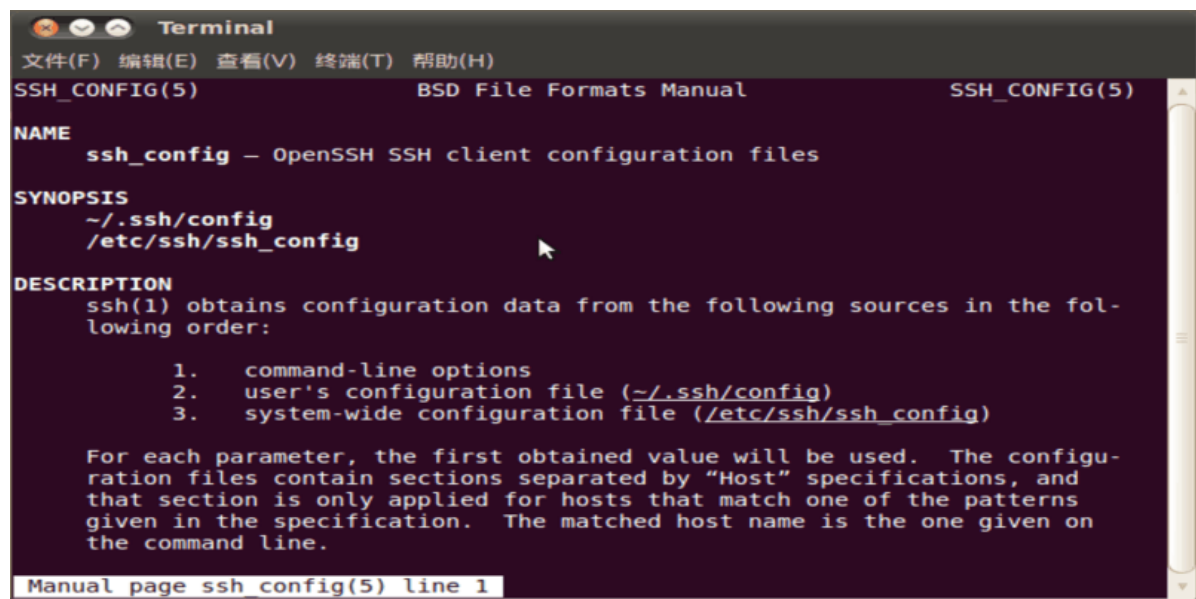
Enter the following command in console to solve:

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

10.2 One Machine Switches Different SSH Public Keys

You can configure SSH by referring to `ssh_config` documentation.

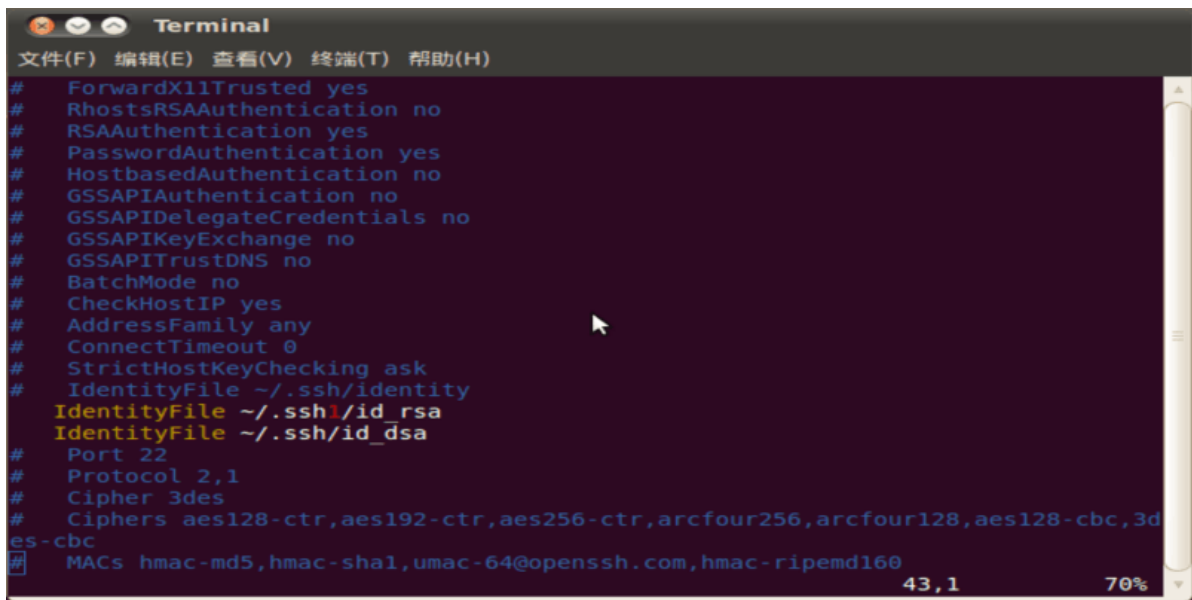
```
~$ man ssh_config
```



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

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

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

A screenshot of a macOS Terminal window titled "Terminal". The menu bar at the top shows "文件(F)", "编辑(E)", "查看(V)", "终端(T)", and "帮助(H)". The terminal content displays the output of the 'ssh -v' command, listing various configuration options and their values. The options include ForwardX11Trusted, RhostsRSAAuthentication, RSAAuthentication, PasswordAuthentication, HostbasedAuthentication, GSSAPIAuthentication, GSSAPIDelegatedCredentials, GSSAPIKeyExchange, GSSAPITrustDNS, BatchMode, CheckHostIP, AddressFamily, ConnectTimeout, StrictHostKeyChecking, IdentityFile, Port, Protocol, Cipher, Ciphers, and MACs. The IdentityFile option is highlighted in yellow for both RSA and DSA keys. The terminal shows the first 43 lines of output, with a scroll bar on the right indicating 70% of the total content is visible.

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

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/Others/Rockchip_User_Guide_SDK_Application_And_Synchronization_CN.pdf