Rockchip PX30 Linux SDK Release Note

ID: RK-FB-CS-005

Release Version: V1.5.0

Release Date: 2021-05-20

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

DISCLAIMER

Trademark Statement

Rockchip Electronics Co., Ltd.

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

Website: www.rock-chips.com

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

Customer service Fax: +86-591-83951833

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

Preface

Overview

The document presents Rockchip PX30 Linux SDK release notes, aiming to help engineers get started with PX30 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 Version	Debian Version	Yocto Version	Kernel Version
PX30	2018.02-rc3	10	3.2	4.4

Revision History

Date	Version	Author	Revision History
2019-04-25	V1.0.0	Ziyuan Xu	Initial version
2019-09-17	V1.2.0	Ziyuan Xu	Update Linux_SDK_V1.2.0 description Update application compile instruction Add the instruction to get source code from github
2020-02-24	V1.3.0	Ziyuan Xu	Improve description for ROS Add description for Debian10
2020-12-03	V1.4.0	Caesar Wang	SDK update to V1.4.0
2021-05-20	V1.5.0	Caesar Wang	Update Hardware/Software Development Guide Add precaution of GPIO power design

Contents

1. Overview

This SDK is based on Buildroot 2018.02-rc3, Yocto 3.2, and Debian 10 or later version with kernel 4.4 and U-boot v2017.09. It is suitable for PX30 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
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 Building Introduction to build a development environment.

3.1 General PX30 Linux SDK Obtain

3.1.1 Get Source Code from Rockchip Code Server

To get PX30 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.

PX30_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/rockchip/platform/manifests -b linux -m \
px30_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 PX30_LINUX_SDK_V1.5_20210520.tgz as an example. After geting a initialization package, you can get source code by running the following command:

```
mkdir px30
tar xvf PX30_LINUX_SDK_V1.5_20210520.tgz -C px30
cd px30
.repo/repo/repo sync -l
.repo/repo/repo sync -c --no-tags
```

Developers can update via `.repo/repo/repo sync -c --no-tags` command according to update instructions that are regularly released by FAE window.

4. Software Development Guide

4.1 Development Guide

Aiming to help engineers get started with SDK development and debugging faster, We have released “Rockchip_Developer_Guide_Linux_Software_CN.pdf” with the SDK, please refer to the documents under the project's docs/ directory.

4.2 Software Update History

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

```
.repo/manifests$ realpath px30_linux_release.xml
# e.g.:printf version v1.5.0, update time is 20210520
# <SDK>/.repo/manifests/px30_linux_release_v1.5.0_20210520.xml
```

Or refer to the project directory:

```
<SDK>/docs/PX30/PX30_Linux_SDK_Note.md
```

5. Hardware Development Guide

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

PX30 EVB hardware design guide:

```
<SDK>/docs/PX30/Rockchip_PX30_Hardware_Design_Guide_V1.3_EN_20191206.pdf
```

PX30 EVB hardware development guide:

```
<SDK>/docs/PX30/Rockchip_PX30_User_Manual_EVB_V1.1_EN.pdf
```

6. The Precaution of IO Power Design

The IO level of the controller power domain must be consistent with the IO level of the connected peripheral chip, and the voltage configuration of software must be consistent with the voltage of hardware to avoid GPIO damage.

 Note:

About matching of GPIO power domain and IO level:

PMUIO0_VDD, PMUIO1_VDD, VCCIO1_VDD, VCCIO2_VDD, VCCIO3_VDD, VCCIO4_VDD, VCCIO5_VDD, VCCIO6_VDD, VCCIO7_VDD, voltage of these GPIO power domain must be consistent with the IO level voltage of the connected peripheral to avoid GPIO damage.

Also need to note that the voltage configuration of software should be consistent with the voltage of hardware: For example, if hardware IO level is connected to 1.8V, the voltage configuration of software should be configured to 1.8V accordingly; if hardware IO level should be connected to 3.3V, and the voltage configuration of software should also be configured to 3.3V to avoid GPIO damage.

Please refer to the following documents for details:

```
<SDK>/docs/PX30/Rockchip_PX30_Introduction_IO_Power_Domains_Configuration.pdf
<SDK>/docs/Common/IO-DOMAIN/Rockchip_Developer_Guide_Linux_IO_DOMAIN_CN.pdf
```

7. 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.
- device/rockchip: store board-level configuration for each chip and some scripts and prepared files for building and packaging firmware.
- docs: stores development guides, platform support lists, tool usage, Linux development guides, and so on.
- IMAGE: stores building time, XML, patch and firmware directory for each building.
- external: stores some third-party libraries, including audio, video, network, recovery and so on.
- kernel: stores kernel4.4 development code.

- prebuilts: stores cross-building toolchain.
- rkbin: stores Rockchip Binary and tools.
- rockdev: stores building 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 3.2.

8. SDK Building Introduction

8.1 SDK Dependency Packages Installation

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

Software requirements: Ubuntu 18.04 system:

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

```
sudo apt-get install repo git ssh make gcc libssl-dev liblz4-tool \
expect g++ patchelf chrpath gawk texinfo chrpath diffstat binfmt-support \
qemu-user-static live-build bison flex fakeroot cmake gcc-multilib g++-multilib
unzip \
device-tree-compiler python-pip ncurses-dev pyelftools \
```

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.

8.2 SDK Board Level Configuration

Enter the project /device/rockchip/px30 directory:

Board level configuration	Note
BoardConfig-px30-evb-ddr3-v10.mk	Suitable for PX30 EVB V10 development board
BoardConfig-px30-evb-ddr3-v10-32bit.mk	Suitable for PX30 EVB V10 development board with 32bit system
BoardConfig-px30-evb-ddr3-v11.mk	Suitable for PX30 EVB V11 development board
BoardConfig-px30-evb-ddr3-v11-32bit.mk	Suitable for PX30 EVB V11 development board with 32bit system
BoardConfig-px30-robot64.mk	Suitable for PX30 robot mini rootfs
BoardConfig-px30-robot64_no_gpu.mk	Suitable for PX30 robot mini rootfs without gpu

The first way:

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

Select the board configuration of **PX30 EVB V10 development board**:

```
./build.sh device/rockchip/px30/BoardConfig-px30-evb-ddr3-v10.mk
or
./build.sh device/rockchip/px30/BoardConfig-px30-evb-ddr3-v10-32bit.mk
```

Select the board configuration of the **PX30 EVB V11 development board**:

```
./build.sh device/rockchip/px30/BoardConfig-px30-evb-ddr3-v11.mk
or
./build.sh device/rockchip/px30/BoardConfig-px30-evb-ddr3-v11-32bit.mk
```

Select the board-level configuration of the **PX30 Robot**:

```
./build.sh device/rockchip/px30/BoardConfig-px30-robot64.mk
or
./build.sh device/rockchip/px30/BoardConfig-px30-robot64_no_gpu.mk
```

The second way:

```
px30$ ./build.sh lunch
processing option: lunch

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

0. default BoardConfig.mk
1. BoardConfig-px30-evb-ddr3-v10-32bit.mk
2. BoardConfig-px30-evb-ddr3-v10.mk
3. BoardConfig-px30-evb-ddr3-v11-32bit.mk
4. BoardConfig-px30-evb-ddr3-v11.mk
5. BoardConfig-px30-robot64.mk
6. BoardConfig-px30-robot64_no_gpu.mk
7. BoardConfig.mk
Which would you like? [0]:
...
```

8.3 Compilation Commands

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

```
px30$ ./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 debian 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`

```
px30$ ./build.sh -h kernel
###Current SDK Default [ kernel ] Build Command###
cd kernel
make ARCH=arm64 px30_linux_defconfig
make ARCH=arm64 px30-evb-ddr3-v11-linux.img -j12
```

8.4 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 three types of system for RK_ROOTFS_SYSTEM: buildroot, Debian, and yocto.

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

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

8.5 Build and package each module

8.5.1 U-boot Build

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

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

8.5.2 Kernel Build

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

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

8.5.3 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

8.5.4 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”.

8.5.4.1 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_px30_64/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_px30_64/host/usr/bin
./aarch64-linux-gcc --version
```

Then the following logs are printed:

```
gcc version 9.3.0 (Buildroot 2018.02-rc3-02723-gd3fbc6ae13)
```

8.5.4.2 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_px30_64/build/qlayer-1.0
```

8.5.5 Debian Building

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

Or enter debian/ directory:

```
cd debian/
```

Please refer to the readme.md in the directory for further building and Debian firmware generation.

(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
```

Build 64 bit Debian:

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

After building, linaro-buster-alip-xxxxx-1.tar.gz (xxxxx is timestamp generated) will be generated in “debian/”:

FAQ:

- If you encounter the following problem during above building:

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

Solution：

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

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

- Because building Base Debian requires to access to foreign websites, and when domestic networks access foreign websites, download failures often occur:

The live build is used in Debian10, you can configure like below to change the image source to domestic:

```
+++ b/ubuntu-build-service/buster-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 "https://mirrors.tuna.tsinghua.edu.cn/debian" \
+ --mirror-chroot "https://mirrors.tuna.tsinghua.edu.cn/debian" \
+ --mirror-chroot-security "https://mirrors.tuna.tsinghua.edu.cn/debian-security"
\
+ --mirror-binary "https://mirrors.tuna.tsinghua.edu.cn/debian" \
+ --mirror-binary-security "https://mirrors.tuna.tsinghua.edu.cn/debian-security"
  --apt-indices false \
  --apt-recommends false \
  --apt-secure false \
```

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

(2) Building rk-debian rootfs

Build 64bit Debian：

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

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

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

The linaro-rootfs.img will be generated.

8.5.6 Yocto Build

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

Px30 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.

8.5.6.1 Firmware Package

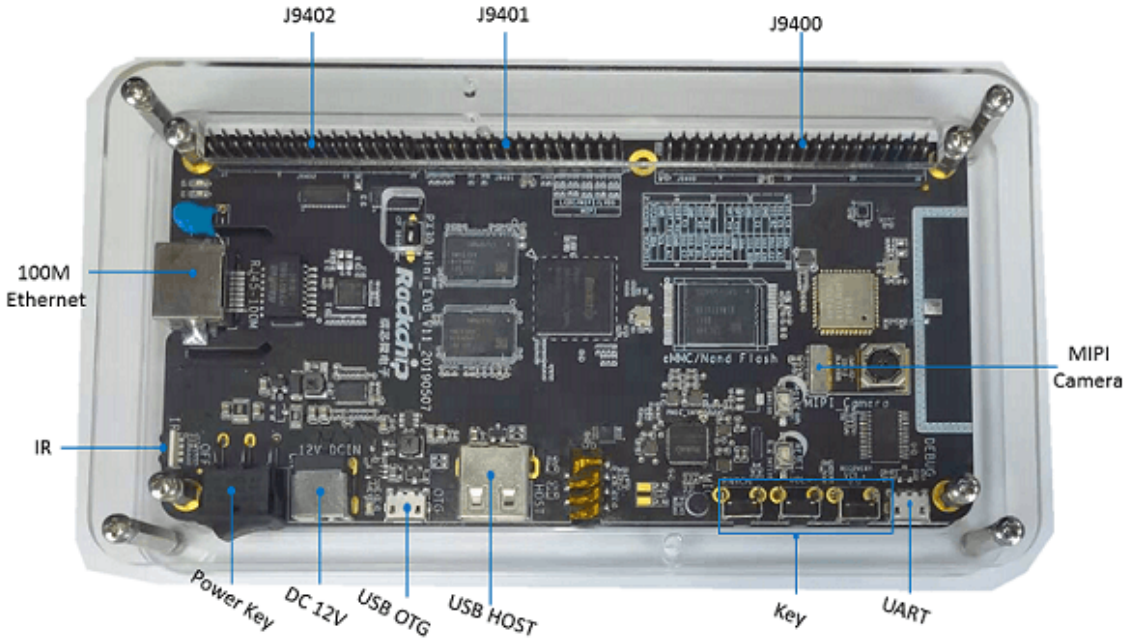
After compiling various parts of Kernel/U-Boot/Recovery/Rootfs above, enter root directory of project directory and run the following command to automatically complete all firmware packaged into rockdev directory:

Firmware generation:

```
./mkfirmware.sh
```

9. Upgrade Introduciton

Interfaces layout of PX30 EVB are showed as follows:

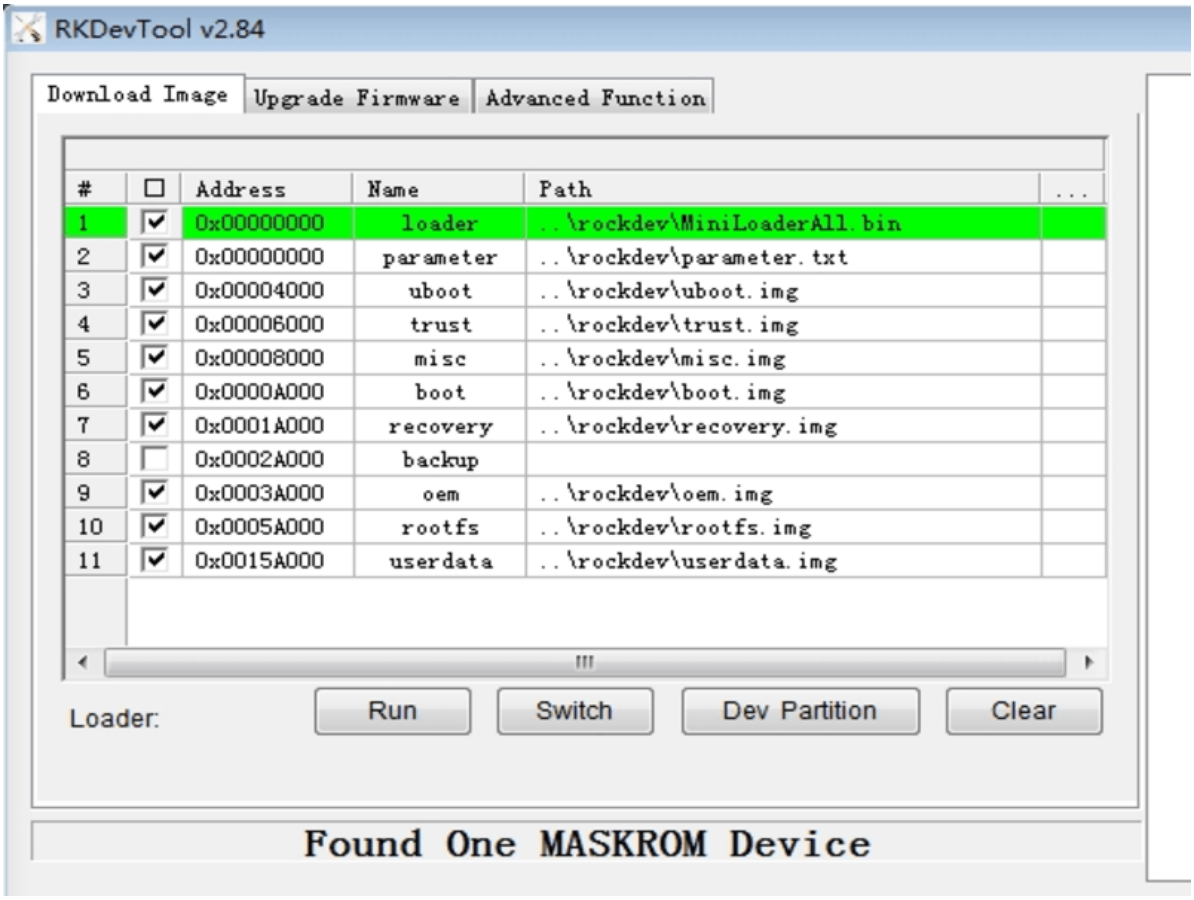


9.1 Windows Upgrade Introduction

SDK provides windows upgrade tool (this tool should be V2.84 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_v5.11.zip
```

9.2 Linux Upgrade Instruction

The Linux upgrade tool (Linux_Upgrade_Tool should be v1.57 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
```

9.3 System Partition Introduction

Default partition introduction (below is PX30 EVB 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

10. PX30 SDK Firmware

- Baidu Cloud Disk

Buildroot

Debian rootfs

Yocto rootfs

- Microsoft OneDriver

Buildroot

Debian rootfs

Yocto rootfs

11. SSH Public Key Operation Introduction

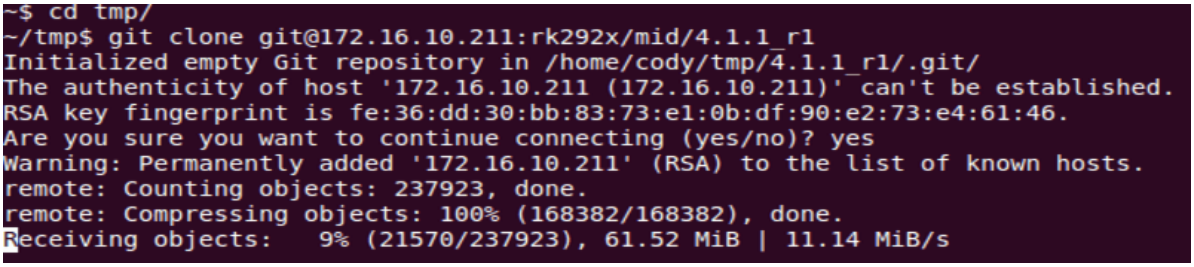
Please follow the introduction in the “Rockchip_User_Guide_SDK_Application_And_Synchronization_CN” to generate an SSH public key and send the email to fae@rock-chips.com, to get the SDK code.
This document will be released to customers during the process of applying for permission.

11.1 Multiple Machines Use the Same SSH Public Key

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

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

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



Adding ssh private key may result in the following error.

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

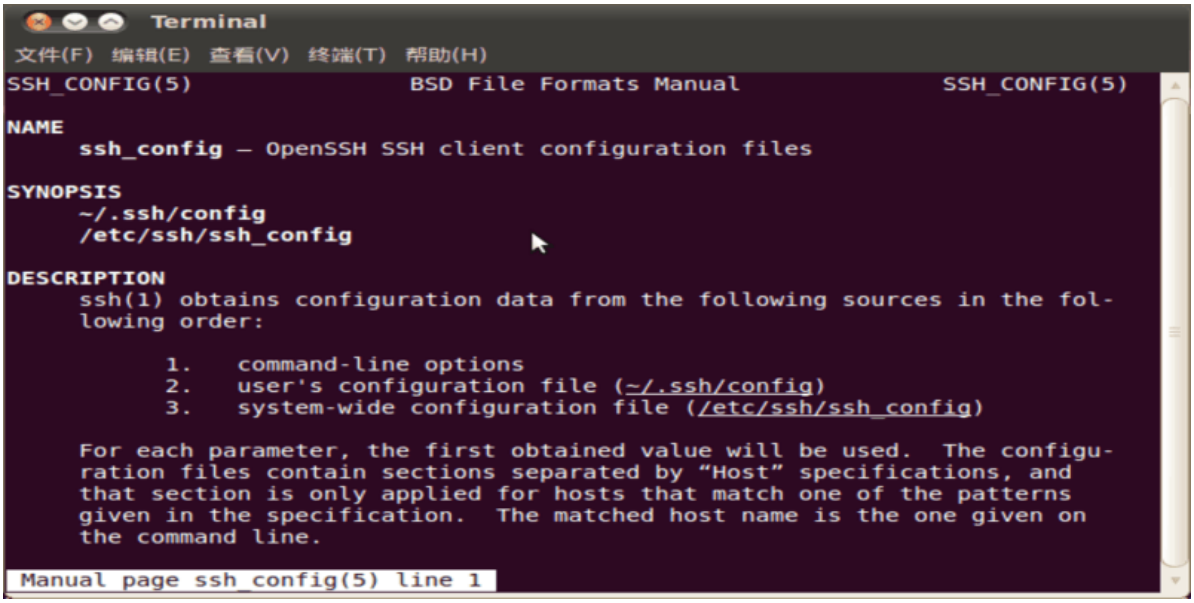
Enter the following command in console to solve:

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

11.2 One Machine Switches Different SSH Public Keys

You can configure SSH by referring to ssh_config documentation.

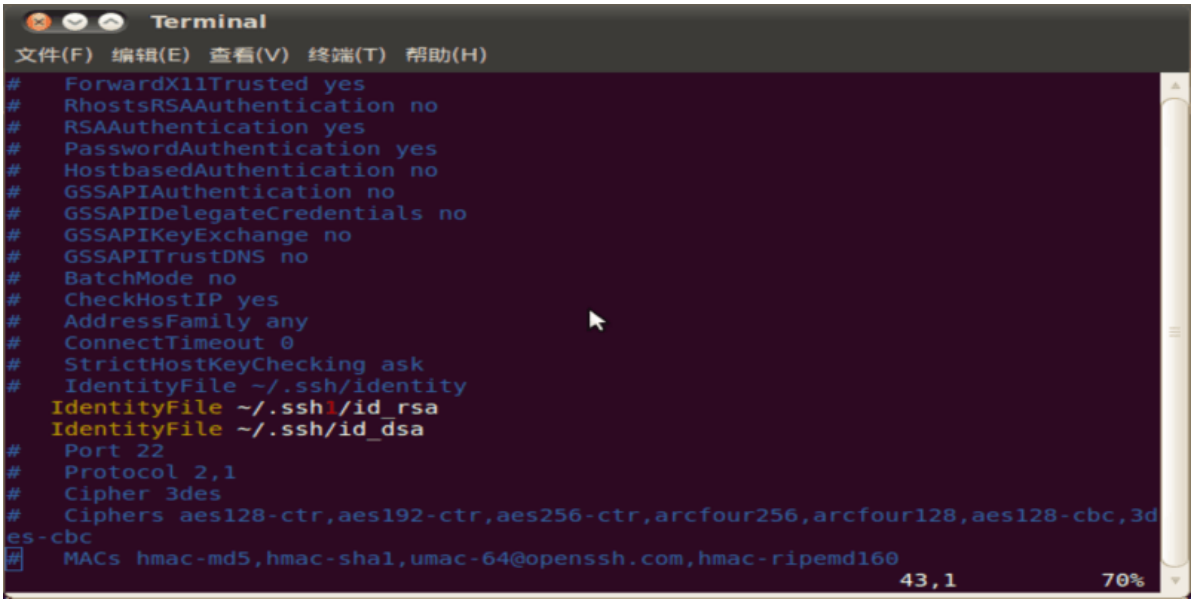
```
~$ man ssh_config
```



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

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

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



11.3 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.

11.4 Reference Documents

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