Rockchip Linux

Software Developer Guide

Release version: 1.05

Date: 2018.12

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

Preface

Overview

Rockchip Buildroot Linux software development guide was designed in order to help software development engineers and technical application engineers get start and debug with Rockchip Buildroot Linux more quickly.

Chipset model

Chip Name	Kernel Version	Buildroot Version
PX30	Linux4.4	2018.02_rc3
RK3308	Linux4.4	2018.02_rc3
RK3326	Linux4.4	2018.02_rc3
RK3399	Linux4.4	2018.02_rc3
RK1808	Linux4.4	2018.02_rc3
RK3399PRO	Linux4.4	2018.02_rc3

Applicable object

This document is mainly suitable for the following engineers:

- Field application engineers
- Software development engineers

Revision history

Revision Date	Version No.	Author	Reviewer	Revision Description
2018-05-10	V1.00	yhx	cch	Initial Release
2018-05-28	V1.01	yhx	cch	1. Update DDR support list as described in chapter 1.1. 2. Update eMMC support list as described in chapter 1.2. 3. Update SPI Nor and SLC Nand support lists as described in chapter 1.3. 4. Update SDK documentations/tools index, see chapter 2 for details. 5. Update data partition of partition table to oem partition, cfg partition to userdata partition, see chapter 6.3 for details. 6. Update upgrade_tool upgrading command, as described in chapter 7.4. 7. Flash type is configured as Nand by default. See chapter 10.7.1.1 for details. 8. Add Rootfs changing to ext2 file system instruction. See chapter 10.1.7.2 for details. 9. Modify the preset system application or data instruction as described in chapter 10.3. 10. Update WiFi/BT Development Guide as described in chapter 10.8. 11. Add DLNA usage and development instructions, as described in chapter 10.11.1. 12. Add instructions for RK3308 DuerOS, as described in chapter 10.13.1. 13. Update chapter 11.1.4 ADB window garbled display. 14. Add PCBA test tools instructions, as described in chapter 12.2.
2018-07-10	v1.02	wxt		1.Add rk3399/rk3326/px30 chips instructions

2018-07-16	v1.03	yhx		1. Update the location of fully automatic compiled script. See chapter 6.4 for details. 2. Add new modules compiling instructions, see chapter 6.5 for details. 3. Add misc and recovery partition upgrading description, see chapter 7.4 for details. 4. Add OTA upgrade instructions, see chapter 10.5 for details. 5. Add RK3308 EVB V12 configuration instructions, compatible with V11, see chapter 6.2 for details. 6. Add 32bit compiling instructions for RK3308 voice module boards. See chapter 6.2 for details.
2018-11-05	V1.04	cww		1. Fixed chapter 5.5 directory name error 2. Fixed chapter 6.2 compile command error 3. Fixed chapter 7.6 directory name error 4. Fixed problems described in chapter 9.1.1 5. Fixed ADB function description problems in chapter 11.1.3 6. Fixed chapter 11.5 last_log description problems 7. Fixed the descriptions of firmware unified package in chapter 12.3
2018-11-25	V1.05	hl		1. Added rk1808 instructions
2019-02-17	V1.05	wxt		1. Add rk3399pro instructions 2. Fixed dvfs/rk3399pro in tools index 2.1 3. Fixed 4.2.3 Addition of Dependent Packages 4. Copyright 2018 is changed to Copyright 2019

Table of content

List of figures

List of tables

Chapter 1 Support list

1.1 DDR support list

DDR support list of Rockchip platform, see "RK DDR Support List V2.32 20180525.pdf" in docs\Platform support lists directory. DDR support level is shown in the table below, only symbols marked with √ and T/A are recommended.

Table 1-1 Rockchip DDR Support Symbols

Symbol	**Description**
√	Fully Tested and Mass production
T/A	Fully Tested and Applicable
N/A	Not Applicable

1.2 eMMC support list

eMMC support list of Rockchip platform, see "RKeMMCSupportList Ver1.39_20180515.pdf" in docs\Platform support lists directory. eMMC support level is shown in the table below, only symbols marked with √ and T/A are recommended.

Table 1-2 Rockchip eMMC Support Symbols

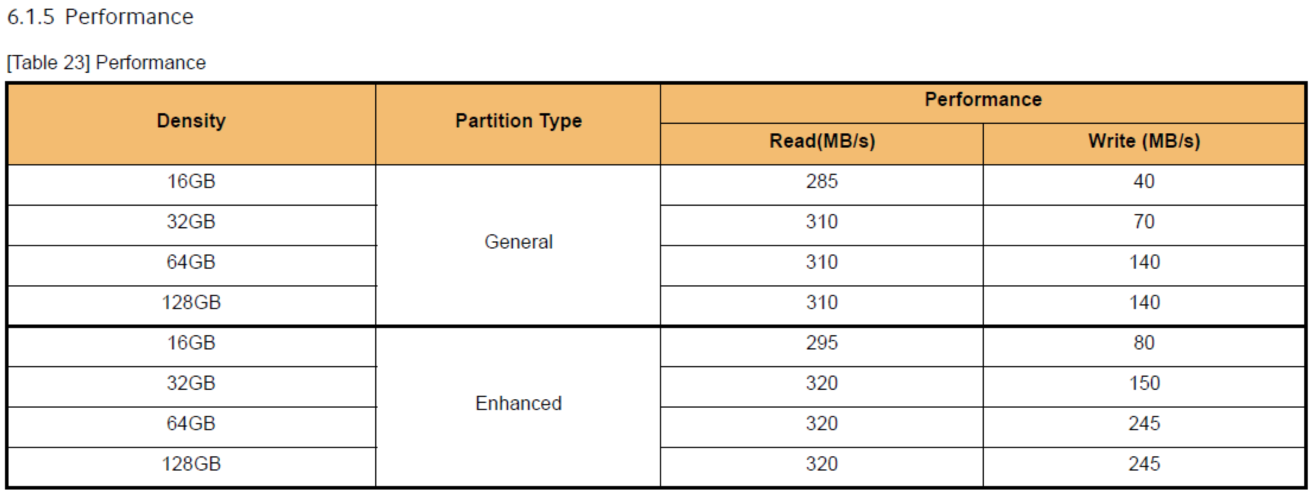
Symbol	**Description**
√	Fully Tested , Applicable and Mass Production
T/A	Fully Tested , Applicable and Ready for Mass Production
D/A	Datasheet Applicable,Need Sample to Test
N/A	Not Applicable

1.2.1 Selection of high performance eMMC symbols

In order to improve system performance, high performance eMMC symbols need to be selected. Please refer to symbols in Rockchip's support list before selecting eMMC symbols, focusing on the performance chapter of vender's Datasheet.

Reference to vender size and rate of eMMC symbols reading and writing to select. It is recommended to select symbols with sequential read rate >200MB/s and sequential write rate of >40MB/s.

If you have questions about selection, you can also contact Rockchip FAE window <fae@rock-chips.com>.



6.1.5 Performance

[Table 23] Performance

Density	Partition Type	Performance	
		Read(MB/s)	Write (MB/s)
16GB	General	285	40
32GB		310	70
64GB		310	140
128GB		310	140
16GB	Enhanced	295	80
32GB		320	150
64GB		320	245
128GB		320	245

Figure 1-1 eMMC Performance examples

1.3 SPI Nor and SLC Nand support list

SPI Nor and SLC Nand support list of Rockchip platform, see “RK SpiNor and SLC Nand SupportList Ver1.07_20180515.pdf” in docs\Platform support lists directory. SPI Nand symbols are also available in the documentation for selection. SPI Nor and SLC Nand support level table is indicated in the table below, only symbols marked with √ and T/A are recommended.

Table 1-3 Rockchip SPI Nor and SLC Nand Support Symbol

Symbol	**Description**
√	Fully Tested , Applicable and Mass Production
T/A	Fully Tested , Applicable and Ready for Mass Production
D/A	Datasheet Applicable,Need Sample to Test
N/A	Not Applicable

1.4 WiFi/BT support list

WiFi / BT support list of Rockchip platform, see "rockchip_Linux_SDK_WiFi_Situation_20180507.pdf" in docs\Platform support lists directory. List of Wifi/Bt chips list in the document is that have been tested on Rockchip platform. It is recommended to select according to modules on the list. If there are other WiFi/BT chips to debug, the corresponding kernel driver is required from WiFI/BT chip vendor.

Maturity is detailed in the documentation:

Perfect > Very Good > Good > Preliminary > X

If you have questions about selection, you can also contact Rockchip FAE window <fae@rock-chips.com>.

1.5 List of hardware suitable for SDK software packages

With SDK release, Rockchip platform will provide related reference hardware. Hardware user guides mainly introduce basic function features, hardware interfaces and instruction of reference hardware boards. It is designed to help developers use EVB faster and more accurately for application development of related products.

Table 1-4 List of Hardware user guides on Rockchip platform

platform	hardware	Corresponding Document
PX30	Evb board	docs\ SoC platform related\PX30\PX30 MINI EVB Hardware Operation Guide_20180710.pdf
RK3308	Evb board	docs\ SoC platform related\RK3308\RK3308_EVB User Guide V001_201805.pdf
RK3399	Evb board	docs\ SoC platform related\RK3399\Rockchip RK3399 Excavators User Guide _V2.0_20180424.pdf
RK1808	Evb board	docs\ SoC platform related\RK1808\RK1808 EVB User Guide_V10_20181226.pdf
RK3399PRO	Evb board	docs\ SoC platform related\RK3399PRO\RK3399PRO EVB User Guide and Schematic and PCB Review Notice _V10_20190111

Chapter 2 Documents and Tools Index

2.1 Documents index

Documents released with Rockchip Linux SDK are designed to help developers start and debug quickly, but contents in documents do not cover all development knowledge and issues. List of documents is also being updated constantly. For questions and requirements obout documents, please contact our FAE window<fae@rock-chips.com>.

Rockchip Linux SDK includes Develop Reference Documents in docs directory, Platform support lists, RKTools manuals, SoC platform related, and Linux reference documents(system develop guide) in docs directory.

```
docs
├── Develop reference documents
│   ├── Audio
│   │   └── Rockchip Audio Developer Guide V1.1-20170215-linux4.4.pdf
│   ├── CAMERA
│   │   ├── CIF_ISP10_Driver_User_Manual_v1.0.pdf
│   │   ├── RK_ISP10_Camera_User_Manual_v2.2.pdf
│   │   └── Rockchip_Camera_AVL_v2.0_Package_20180515.7z
│   ├── CRU
│   │   └── Rockchip Clock Submodule Developer Guide V1.1-20170210.pdf
│   ├── DDR
│   │   ├── DDR Developer Guide.pdf
│   │   ├── DDR Troubleshooting manual 1.0.pdf
│   │   ├── DDR Symbols Verification Process Description.pdf
│   │   └── RK DDR Application Note 5--Customers verify their own DRAM
process.pdf
│   ├── DEBUG
│   │   ├── perf instructions.pdf
│   │   ├── streamline instructions.pdf
│   │   └── systrace instructions.pdf
│   ├── DISPLAY
│   │   ├── rockchip_drm_integration_helper-zh.pdf
│   │   └── Rockchip_DRM_Panel_Porting_Guide_V1.3_20171209.pdf
│   ├── DVFS
│   │   ├── Linux 4.4 kernel DVFS instructions.pptx
│   │   ├── Rockchip CPU-Freq Developer Guide V1.0.1-20170213.pdf
│   │   └── Rockchip DEVFreq Developer Guide V1.0-20160701.pdf
│   ├── GMAC
│   │   └── Rockchip Ethernet Developer Guide V2.3.1-20160708.pdf
│   ├── I2C
│   │   └── Rockchip-Developer-Guide-Linux-I2C.pdf
│   ├── IO-DOMAIN
│   │   └── Rockchip IO-Domain Developer Guide V1.0-20160630.pdf
│   ├── MCU
│    │    └─── Rockchip-Developer-Guide-linux4.4-MCU.pdf
│    ├─── MMC
│    │    └─── Rockchip-Developer-Guide-linux4.4-SDMMC-SDIO-eMMC.pdf
│    ├─── MPP
│    │    └─── MPP Development Reference _v0.3.pdf
│    ├─── NPU
│    │    ├─── RK1808_RKNN_SDK Developer Guide _V0.9.6.pdf
│    │    └─── RKNN-Toolkit instructions _V0.9.6.pdf
│    ├─── NVM
│    │    ├─── appnote rk flash storage 2017.09.06.pdf
│    │    ├─── appnote rk vendor storage 2017.01.23.pdf
│    │    └─── Rockchip-Secure-Boot-Application-Note.pdf
│    ├─── PCIe
│    │    ├─── PCIe Protocol And Linux Driver Introduction.pdf
│    │    └─── Rockchip-Developer-Guide-linux4.4-PCIe.pdf
│    ├─── PERF
│    │    ├─── perf Instructions.pdf
│    │    ├─── streamline Instructions.pdf
│    │    └─── systrace Instructions.pdf
│    ├─── Pin-Ctrl
│    │    ├─── Rockchip GPIO common issues.pdf
│    │    └─── Rockchip Pin-Ctrl Developer Guide V1.0-20160725.pdf
│    ├─── PMIC
│    │    ├─── Rockchip-Developer-Guide-Linux4.4-DC-DC.pdf
│    │    ├─── Rockchip-Developer-Guide-RK805.pdf
│    │    ├─── Rockchip-Developer-Guide-RK818_6-Fuel-Gauge.pdf
│    │    ├─── Rockchip RK805 Developer Guide V1.0-20170220.pdf
│    │    ├─── Rockchip RK818_6 voltmeter Developer Guide V2.0-20170525.pdf
│    │    ├─── Rockchip-RK818-RK816-FG-Log-Description-linux4.4.pdf
│    │    └─── Mimimum System Clock Solution Summary.pptx
│    ├─── PWM
│    │    └─── Rockchip Backlight Control Developer Guide V0.1-20160729.pdf
│    ├─── S2R
│    │    ├─── Rockchip Suspend and Resume Developer Guide V0.1-20160729.pdf
│    │    ├─── Suspend and Resume process.pdf
│    │    └─── Suspend and Resume related Issues Fixing.pdf
│    ├─── SECURITY
│    │    ├─── Rockchip-Secure-Boot-Application-Note-V1.9.pdf
│    │    └─── Rockchip TEE Security SDK Developer Guide V1.0.pdf
│    ├─── SPI
│    │    └─── Rockchip-Developer-Guide-linux4.4-SPI.pdf
│    ├─── THERMAL
│    │    └─── Rockchip Thermal Developer Guide-4.4 V1.0.1-20170428.pdf
│    ├─── TOOL
│    │    ├─── Production-Guide-For-Firmware-Download.pdf
│    │    ├─── RK3288 JTAG Configuration Guide V1.0-20170316.pdf
│     │     ├─── RK3399 JTAG Configuration Guide V1.1-20170327.pdf
│     │     ├─── RK3399-LOG-EXPLANATION.pdf
│     │     ├─── rk_fw_upgrade_issue.pdf
│     │     ├─── RKUpgrade_Dll_UserManual.pdf
│     │     └─── Rockchip-Parameter-File-Format-Version1.4.pdf
│     ├─── TRUST
│     │     ├─── armv8 trusted firmware- Instructions V1.0-20170525.pdf
│     │     └─── Rockchip-Developer-Guide-Trust.pdf
│     ├─── UART
      │     └─── Rockchip-Developer-Guide-linux4.4-UART.pdf
│     ├─── U-Boot
│     │     ├─── Rockchip-Developer-Guide-UBoot-nextdev.pdf
│     │     └─── Rockchip-Developer-Guide-UBoot-rkdevelop-vs-nextdev.pdf
│     └─── USB
│          ├─── RK3399-USB-DTS.pdf
│          ├─── Rockchip-Developer-Guide-linux4.4-USB.pdf
│          ├─── Rockchip-USB-FFS-Test-Demo.pdf
│          ├─── Rockchip-USB-Performance-Analysis-Guide.pdf
│          ├─── Rockchip-USB-SQ-Test-Guide.pdf
│          └─── USB-Initialization-Log-Analysis.pdf
├─── Linux reference documents
│     ├─── camera_engine_rkisp_user_manual_v1.0.pdf
│     ├─── Linux QT Application Development Introduction.pdf
│     ├─── Linux rga Introduction.pdf
│     ├─── Linux SDK Pressure Test Methods.pdf
│     ├─── mipi-dsi.ppt
│     ├─── RK3308_RTL8723DS_WIFI_BT_Instructions_V1.20.pdf
│     ├─── RK_linux_camera_gstreamer_application
development_v1.0_20171212.pdf
│     ├─── RK_Linux_SDK Compiling Development Environment Established.pdf
│     ├─── Use of Rockchip Debian Double Screen Display Audio Instructions.pdf
│     ├─── Rockchip DLNA Developer Guide _V1.00.pdf
│     ├─── Rockchip_Gstreamer Instructions.pdf
│     ├─── Rockchip Linux DS5 Performance Debug Tool Streamline Instructions.pdf
│     ├─── Rockchip Linux WIFI BT Developer Guide V1.1 20180622.pdf
│     ├─── Rockchip Linux Upgrade firmware packaging Guide.pdf
│     ├─── Rockchip Recovery Developer GuideV1.0.3.pdf
│     ├─── Rockchip ROS Instructions.pdf
│     ├─── Rockchip Secureboot Instructions.pdf
│     ├─── Rockchip Linux Audio Developer Guide.pdf
│     ├─── Rockchip Linux Camera Developer Guide.pdf
│     ├─── Rockchip Linux SDK uboot logo display developer guide.pdf
│     ├─── Rockchip PCBA test Developer Guide _1.05.pdf
│     ├─── Rockchip Fragmentation Upgrade Developer Guide -V1.0.0.pdf
│     └─── The Buildroot User Manual.pdf
├─── Platform support lists
```

```
│   ├── RK DDR Support List Ver2.34.pdf
│   ├── RKeMMCSupportList Ver1.41_20181030.pdf
│   ├── RKeMMCSupportList Ver1.42_2018_11_30.pdf
│   ├── RKNandFlashSupportList Ver2.73_20180615.pdf
│   ├── RK SpiNor and SLC Nand SupportList Ver1.10_2018_1101.pdf
│   ├── Rockchip_Camera_Module_AVL_v2.0.pdf
│   └── Rockchip_Linux_SDK_WiFi_Situation_20180507.pdf
├── RKTools manuals
│   ├── Android Adds A Partition Configuration Guide V1.00.pdf
│   ├── Linux Development Tools Manual_v1.32.pdf
│   ├── REPO Mirror Server Setup and Management_V2.2_20131231.pdf
│   ├── RK3308_EQ_DRC_TOOL_v1.22.pdf
│   ├── Rockchip-Parameter-File-Format-Version1.4.pdf
│   ├── Rockchip Bug System Guidelines V1.0-201712.pdf
│   ├── Rockchip Mass Production Upgrade Guide V1.1-20170214.pdf
│   ├── SecureBoot Development Tools Guidelines .pdf
│   ├── WNpctool Brief Instructions_V1.1.0_0920.pdf
│   └── Rockchip Development Tools Manual _v1.1.pdf
├── Rockchip Linux Software Developer Guide.pdf
└── SoC platform related
    ├── PX30
    │   ├── io_list_v0.1_for evb&ref 20180112.xlsx
    │   ├── PX30_LINUX_BETA_V0.2_20180803 Release Notes.pdf
    │   ├── PX30 Linux SDK Release Note.pdf
    │   ├── PX30 MINI EVB Hardware Instruction_20180710.pdf
    │   ├── rk_evb_px30_ddr3p416dd6_v10.pdf
    │   └── Rockchip PX30 Datasheet V1.1-20180918.pdf
    ├── PX3SE
    │   ├── PX3SE_Linux_Beta_V0.3_20180817 Release Notes.pdf
    │   ├── PX3SE Linux SDK Release Note.pdf
    │   ├── RK312x Multimedia Codec Benchmark v1.1.pdf
    │   └── Rockchip PX3 SE Datasheet V1.2-20171226.pdf
    ├── RK1808
    │   ├── RK1808 EVB User Guide_V10_20181226.pdf
    │   ├── RK1808_RKNN_SDK Developer Guide _V0.9.6.pdf
    │   ├── RKNN-Toolkit Guidelines_V0.9.6.pdf
    │   └── Rockchip RKNN_DEMO Module Developer Guide V0.1.pdf
    ├── RK3288
    │   ├── RK3288 Linux SDK Release Note.pdf
    │   ├── Rockchip RK3288 Datasheet V2.4-20180530.pdf
    │   ├── Rockchip_RK3288 Linux_SDK_V2.0_ Release Notes_20180620.pdf
    │   └── Rockchip_RK3288 Linux_SDK_V2.1_ Release Notes_20180926.pdf
    ├── RK3308
    │   ├── RK3308 Asound Configuration Instructions_V1.20.pdf
    │   ├── RK3308 Audio Codec Benchmark v1.0.pdf
    │   ├── RK3308_Audio_Codec_Introduction_v0.3.0_CN.pdf
```

```
|     ├─── RK3308 DuerOS Guidelines V1.00_20180929.pdf
|     ├─── RK3308_EVB User Guide V001_201805.pdf
|     ├─── RK3308 input-event-daemon Instructions.pdf
|     ├─── RK3308_Linux_SDK_For_Robot_V1.00_20181208 Release Notes.pdf
|     ├─── RK3308 SD card & U disk Developer Guide _V1.0_20180907.pdf
|     ├─── RK3308_VAD_Register Configuration _v1.1.pdf
|     ├─── RK3308 VAD+ Iflytek Identification Demo Instructions_v1.1.pdf
|     ├─── RK3308 LED Control Demo Introduction.pdf
|     ├─── RK3308 Local Player Demo Introduction.pdf
|     ├─── RK3308 AISPEECH Voice SDK Instructions V0.01_20180607.pdf
|     ├─── RK3308 Voice Module 32-bit System Compilation Guide _v1.40.pdf
|     ├─── RK3308_Audio Gain Adjustment_ Introduction_V1.1.pdf
|     ├─── Rockchip RK3308 Datasheet V1.4-20180812.pdf
|     └─── Rockchip_ Hardware VAD Module_Instructions _V1.0.pdf
├─── RK3326
|     ├─── RK3326 EVB Development Board User Guide v10_20180705.docx
|     ├─── RK3326 EVB Development Board User Guide v10_20180705.pdf
|     ├─── RK3326_LINUX_SDK_BETA_20180803 Release Note.pdf
|     ├─── RK3326 Linux SDK Release Note.pdf
|     ├─── RK_Linux_SDK Compile and Develop Environment Established.pdf
|     └─── Rockchip RK3326 Datasheet V1.1 20180411.pdf
├─── RK3328
|     ├─── RK3328 BOX EVB Hardware Guidelines V1.0_20170223.pdf
|     ├─── RK3328_Linux_Release_Note.pdf
|     ├─── RK3328_Linux_Release_V1.0_20181102 Release Notes.pdf
|     ├─── RK_Linux_SDK Compile and Develop Environment Established.pdf
|     └─── Rockchip RK3328 Datasheet V1.2-20180205.pdf
├─── RK3399
|     ├─── RK3399_Linux_NN_SDK_V1.0_20180713 Release Notes.pdf
|     ├─── RK3399 Linux SDK Release Note.pdf
|     ├─── RK3399_LINUX_SDK_V2.09_20181102 Release Notes.pdf
|     ├─── RK3399 Multimedia Codec Benchmark v1.0.pdf
|     ├─── RK3399_SSD_Android&Linux_V1.0_20180522.pdf
|     ├─── Rockchip-Developer-Guide-linux4.4-MCU.pdf
|     ├─── Rockchip RK3399 Datasheet V1.8-20180529.pdf
|     └─── Rockchip RK3399 Excavator User Guide _V2.0_20180424.pdf
└─── RK3399PRO
      ├─── RK3399Pro EVB Board Introduction.jpg
      ├─── RK3399PRO_IO_LIST_V11_for EVB 20181203
      ├─── RK3399Pro Schematic and PCB Review Notice_V10_2019*/
      ├─── RK3399PRO_LINUX_SDK_V0.01_20190217 Release Notes.pdf
      ├─── RK3399PRO_NPU Power on and Start-up.pdf
      └─── RK3399PRO-RKNN_DEMO Module Development
GuideV0.1_20181208.pdf
```

2.2 Tools index

Tools released with Rockchip Linux SDK are used for development or debug and mass production. Tools version will be updated with SDK updates. If you have any questions or requirements about tools, please contact our FAE <fae@rock-chips.com>.

In Rockchip Linux SDK, linux (tools used in Linux operating system) and windows (tools used in Windows operating system) are included in tools directory.

```
tools/
├── linux
│    ├── Linux_Pack_Firmware（Linux Firmware package Tools）
│    ├── Linux_SecureBoot（Linux Firmware Signature Tools）
│    ├── Linux_TA_Sign_Tool.rar
│    ├── Linux_Upgrade_Tool（Linux development tools）
└── windows
     ├── AndroidTool（development tools）
     │    ├── AndroidTool_Release_v2.52
     │    └── rockdev（Firmware packaging Tools）
     ├── DriverAssitant_v4.5.zip（Driver installation assistant）
     ├── PCBATool_Setup_V1.0.6_0516_3308.exe（PCBA test of PC end tools）
     ├── efuse_v1.37.zip（Efuse burning tools）
     ├── FactoryTool_v1.61.zip（Factory mass production tools）
     ├── SecureBootTool_v1.89.zip（Firmware Signature Tools）
     ├── SpiImageTools_v1.36.zip
     ├── EQ_DRC_TOOL_v1.1.zip（Sound effects real-time tuning tools）
     └── WNpctool_Setup_V1.1.9_180118.zip（Vendor information burning tool）
```

Chapter 3. SDK software architecture

3.1 SDK overview

Rockchip Buildroot Linux SDK is a software development package based on Buildroot-2018.02 version, which contains various system source code, drivers, tools, and application packages for Linux-based system development.

Buildroot is an open source embedded linux system automatically built framework on Linux platform. The whole Buildroot is made up of Makefile scripts and Kconfig configuration files. You can compile a complete Linux system software that can be directly written to device through Buildroot configuration.

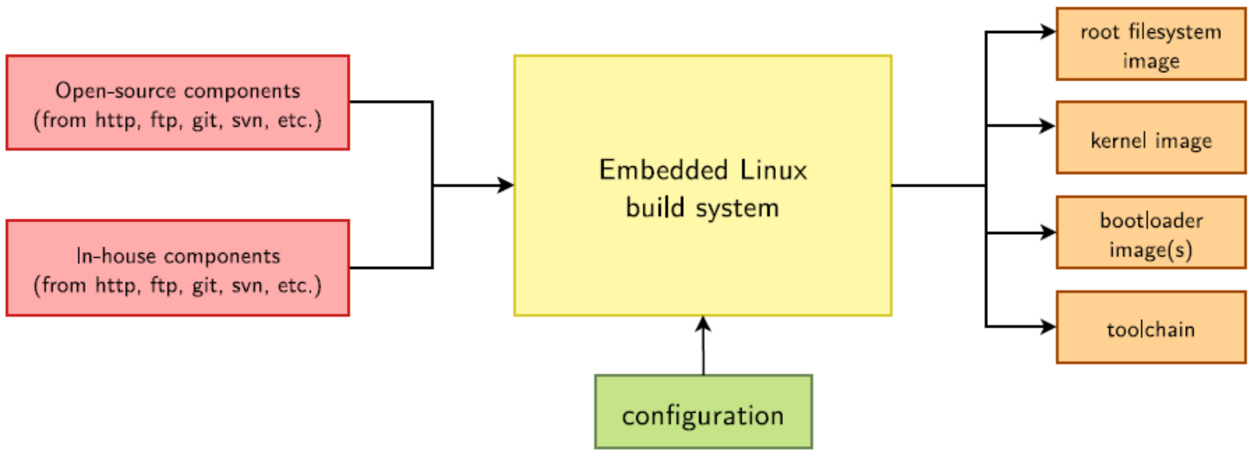

Open-source components (from http, ftp, git, svn, etc.)
In-house components (from http, ftp, git, svn, etc.)
Embedded Linux build system
configuration
root filesystem image
kernel image
bootloader image(s)
toolchain

Figure 3-1 Buildroot compilation block diagram

Buildroot has following advantages:

1. Build through source code with great flexibility;
2. Convenient cross-compilation environment for quick build;
3. Each system component is easy to configure and convenient for custom development.

3.2 SDK software block diagram

The SDK software block diagram 3-2 shows four levels of bootloader, Linux Kernel, Libraries, and Applications from bottom to top.

Contents of each level are as follows:

- Bootloader layer mainly provides underlying system support packages, such as Bootloader, U-Boot, and ATF related support.
- Kernel layer mainly provides a standard implementation of Linux Kernel, and Linux is also an open operating system.
- Linux core of Rockchip platform is standard Linux 4.4 kernel, providing basic support such as security, memory management, process management, network protocol stack, etc. mainly manages device hardware resources such as CPU scheduling, cache, memory, I/O through Linux kernel, etc.
- Libraries layer corresponds to general embedded system, which is equivalent to

middleware layer. It includes various system basic libraries and third-party open source libraries support. It provides API interfaces for application layer. System customizers and application developers can develop new applications based on API of Libraries layer.

- Applications layer mainly implements specific product functions and interaction logic. It requires some system basic libraries and third-party libraries support. Developers can develop and implement their own applications and provide various capabilities of system to end users.

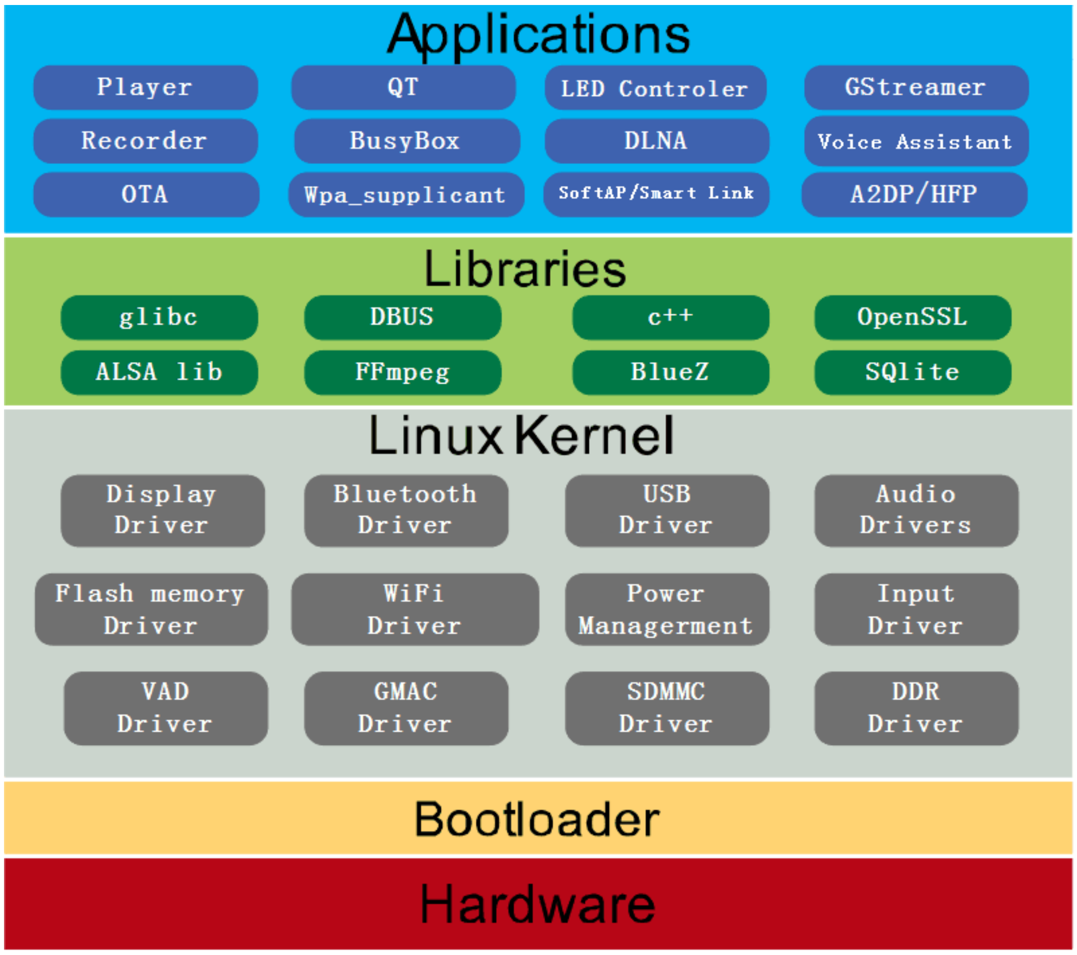


Figure 3-2 Buildroot system software level

SDK system boot up process is shown in Figure 3-3.

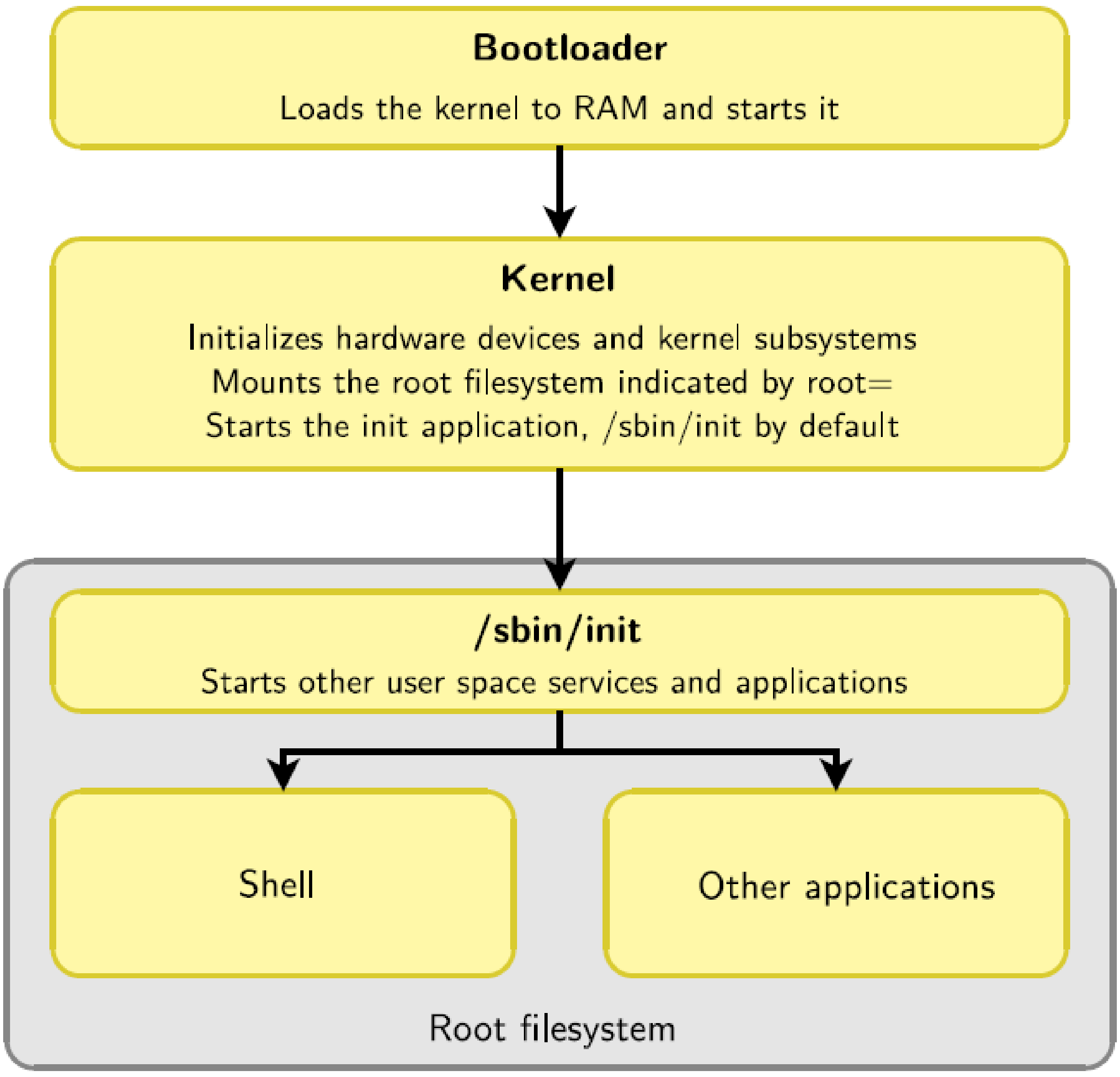

Bootloader
Loads the kernel to RAM and starts it
Kernel
Initializes hardware devices and kernel subsystems
Mounts the root filesystem indicated by root=
Starts the init application, /sbin/init by default
/sbin/init
Starts other user space services and applications
Shell
Other applications
Root filesystem

Figure 3-3 Linux system boot up process

3.3 SDK development process

Rockchip Buildroot Linux system is based on Linux Kernel and is developed for a variety of different products. System customization and application ports development can be effectively implemented based on this SDK.

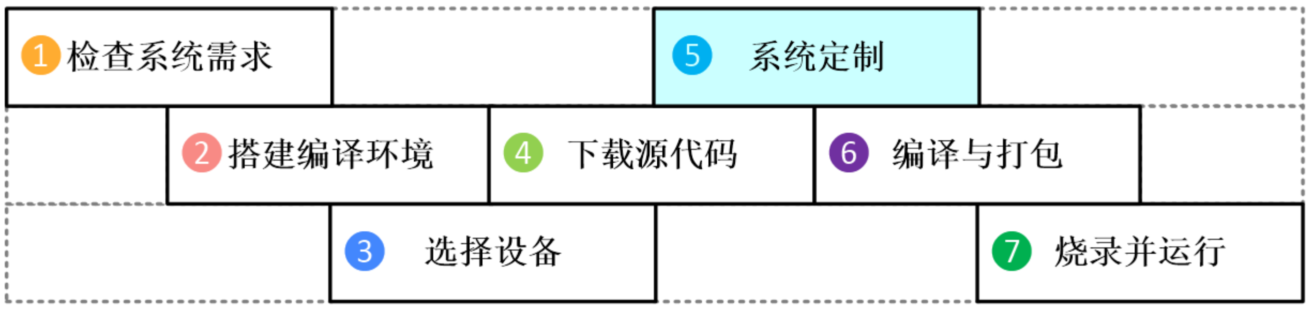


Figure 3-4 development process

As shown in Figure 3-4, developers can follow the above development process to quickly build development environment and compile code of Rockchip Buildroot Linux system locally. The process will be briefly described below:

Check system requirements: Before downloading code and compiling, make sure whether local development equipment meets the requirements, including hardware capabilities of device, software system, toolchain, and so on. At present, SDK supports compiling under Linux system, and only provides toolchains support under Linux environment. Other systems such as MacOS and Windows are not supported at present.

Build compiling environment: about introductions to various software packages and tools that need to be installed on development machine see Chapter 4, Develop Environment Construction for details, to get versions of Linux operating system that Rockchip Buildroot Linux has verified, and libraries files that are dependent when compiling.

Select device: during development process, developers need to select corresponding hardware boards according to their own needs. For details, see chapter 1.5 List of Hardwares Suitable for SDK Software Packages.

Download source code: after selecting device, you need to install repo tool to download source code in mass. For details, see chapter 5.4 To obtain SDK.

System customization: Developers can customize U-Boot (see Chapter 8 U-Boot Develop), Kernel (see Chapter 9 Kernel Develop) and Buildroot (see Chapter 10 Buildroot Develop) according to hardware boards and product definition used. Please refer to the related development guides and configuration instructions in chapters.

Compile and package：After introduction to source code, select product and initialize related compiling environment, and then execute compile command, including the whole or module compiling and compiling cleanup. For details, see Chapter 6 SDK Compile.

Burn and run: After generating image file, it will introduce how to burn image and run it on hardware device. For details, see Chapter 7 SDK Image Upgrade.

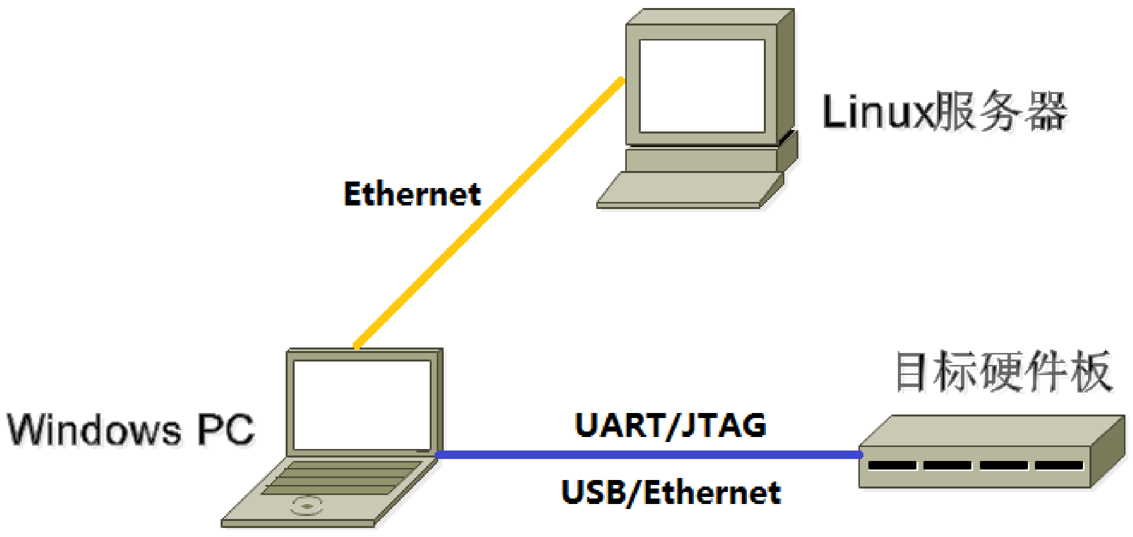
Chapter 4. Develop Environment Construction

4.1 Overview

This chapter mainly describes how to build compiling environment locally to compile Rockchip Buildroot Linux SDK source code. Current SDK only supports compile under Linux environment and provides a cross-compiling toolchain under Linux.

A typical embedded development environment usually includes a Linux server, a Windows PC, and a target hardware board. Take RK3308 as an example. The typical development environment is shown in Figure 4-1.

- A cross-compiling environment is built on Linux server to provide code update and download and code cross-compiling services for software development.
- Windows PC and Linux server share programs, and install SecureCRT or puTTY, remotely log in to Linux server through network to cross-compile, and code develop and debug.
- Windows PC connects to target hardware board through serial port and USB, and can write the compiled image file to target hardware board to debug system or application.



Figrue 4-1 Typical development environment

Note: In fact, a lot of work can be done on the Linux PC which is used in the development environment. For example, if you use minicom instead of SecureCRT or puTTY, you can choose it.

4.2 Linux server development environment construction

Rockchip Buildroot Linux SDK was developed and tested on Ubuntu 16.04. Therefore, we recommend to compile on Ubuntu 16.04 system. Other versions are not specifically tested and may need to be adjusted accordingly.

In addition to system requirements, there are other software and hardware requirements.

- Hardware requirements: 64-bit system, disk space is greater than 40G. If you do multiple builds, you will need more disk space.
- Package dependencies: In addition to python 2.7, make 3.8, git 1.7, some additional packages need to be installed, which will be listed in the package installation chapter.

4.2.1 Linux server system version of release package

This SDK development environment installed the following version of Linux system, and the SDK is compiled with this Linux system by default.

Ubuntu 16.04.2 LTS

Linux version 4.4.0-62-generic (buildd@lcy01-30) (gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.4) #83-Ubuntu SMP Wed Jan 18 14:10:15 UTC 2017

4.2.2 Network environment construction

Please configure network by youself and install network components such as nfs, samba, ssh

4.2.3 Package installation

After operating system is installed and configured network environment, you can continue to install the related software packages as follows.

1. apt-get update

```
$ sudo apt-get update
```

2. Install the packages that Kernel and U-Boot compile depended on

```
sudo apt-get install git-core gnupg flex bison gperf build-essential zip curl
zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386 lib32ncurses5-dev
x11proto-core-dev libx11-dev lib32z1-dev ccache libgl1-mesa-dev libxml2-utils
xsltproc unzip device-tree-compiler liblz4-tool
```

3. Install the package that Buildroot depended on.

```
sudo apt-get install libfile-which-perl sed make binutils gcc g++ bash patch gzip
bzip2 perl tar cpio python unzip rsync file bc libmpc3 git repo texinfo pkg-config cmake
tree texinfo
```

If compiler encounters errors, install the corresponding software packages according to error message.

4.2.4 Cross-compiling toolchain introduction

Since Rockchip Buildroot SDK is currently only compiled under Linux, we only provide a cross-compiling toolchain under Linux. The compile toolchain prebuilt directory used by U-Boot and Kernel is under prebuilt/gcc, and buildroot uses the toolchain compiled from open source software.

U-Boot and Kernel compile toolchain

```
prebuilts/gcc/linux-x86/aarch64/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-g
```

```
nu/bin/aarch64-linux-gnu-
```

Corresponding version

```
gcc version 6.3.1 20170404 (Linaro GCC 6.3-2017.05)
```

Buildroot compile toolchain

```
buildroot/output/rockchip_rk3308_release/host/bin/aarch64-rockchip-linux-gnu-
```

Corresponding version

```
gcc version 6.4.0 (Buildroot 2018.02-rc3-00017-g9c68ede
```

If you need other platform or version of toolchain, you need to compile it yourself.

After the above environment is ready, Linux server development environment building is complet，and the source code can be downloaded.

4.2.5 Linux upgrade tool

Linux upgrade tool is released in tools\linux\Linux_Upgrade_Tool\Linux_Upgrade_Tool directory. Can be used for development and debugging and firmware upgrading under Linux environment.

See chapter 7.4 Upgrade tool for details.

4.3 Windows PC development environment Construction

4.3.1 Development tools installation

Please install SourceInsight, Notepad++ and other editing software.

4.3.2 Rockchip USB driver installation

During developing and debugging, you need to switch device to Loader mode or Maskrom mode. You need to install Rockusb driver to identify device normally.

Rockchip USB Driver Installation Assistant is stored in tools\windows\DriverAssitant_v4.x.zip. It upports xp, win7_32, win7_64, win8_32, win8_64 operating system.

Installation steps are as follows:

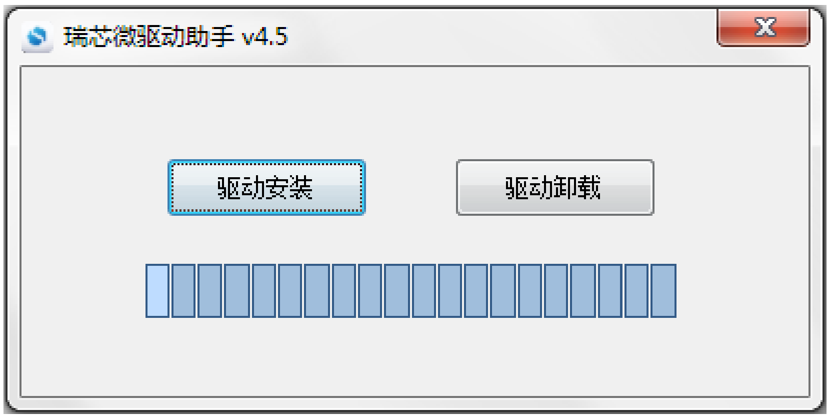
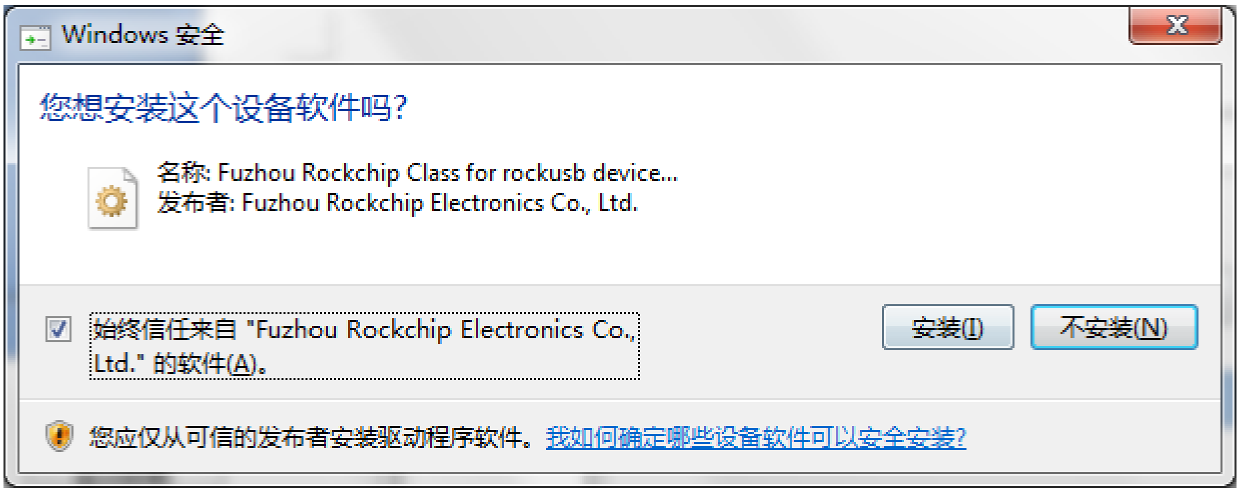


Figure 4-3 Rockchip USB driver installation 1



Windows 安全

您想安装这个设备软件吗?

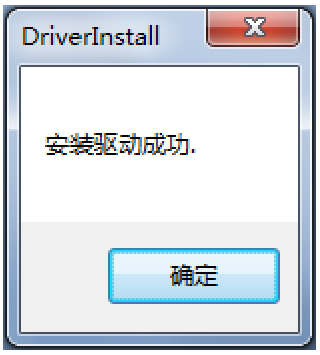
名称: Fuzhou Rockchip Class for rockusb device...
发布者: Fuzhou Rockchip Electronics Co., Ltd.

[✓] 始终信任来自 "Fuzhou Rockchip Electronics Co., Ltd." 的软件(A)。

安装(I) 不安装(N)

您应仅从可信的发布者安装驱动程序软件。我如何确定哪些设备软件可以安全安装?

Figure 4-4 Rockchip USB driver installation 2



DriverInstall

安装驱动成功.

确定

Figure 4-5 Rockchip USB driver installation 3

4.3.3 Windows upgrade tool

Upgrade tool of Windows system is released in tools\windows\AndroidTool/AndroidTool_Release, which can be used for developing and debugging and firmware upgrading under Windows environment.

For detail instructions, see chapter 12.3 Firmware package tools.

4.4 Target hardware board preparation

Please refer to Section 1.5 List of Hardware Suitable for SDK Software Packages, select the corresponding hardware board for further development and debugging.

The corresponding hardware instructions will introduce hardware interfaces, instructions for use, and upgrade methods.

Chapter 5. SDK Installation Preparation

5.1 Brief introduction

The code and related documents of Rockchip Buildroot Linux SDK are divided into several git repositories for version management. Developers can use repo to download, commit, and switch branches for these git repositories uniformly.

5.2 Repo installation

Make sure there is a bin/ directory under home directory and that directory is included in the path:

```
mkdir ~/bin
export PATH=~/bin:$PATH
```

If you can access to google's address, download Repo tool and make sure it's executable:

```
curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
chmod a+x ~/bin/repo
```

If you execute the above command under Chinese domestic environment and find that ~/bin/repo is empty, you can visit the domestic site to download repo tool.

```
curl https://mirrors.tuna.tsinghua.edu.cn/git/git-repo -o ~/bin/repo
chmod a+x ~/bin/repo
```

In addition to the above two methods, you can also use the following command to get repo:

```
sudo apt-get install repo
```

5.3 Git configuration

Please configure your own git information before using repo, otherwise the later operations may occur obstacles in hook checking:

```
git config --global user.name "your name"
git config --global user.email "your mail"
```

5.4 To obtain SDK

SDK is released out through Rockchip code server. Customers applie for SDK from Rockchip technical window, and needs to provide SSH public key to server authentication and authorization simultaneously. After obtaining authorization, code can be synchronized. For SSH public key authorization of Rockchip code server, please refer to Appendix 13.1 SSH public key operation instructions.

5.4.1 SDK download command

Rockchip Buildroot Linux SDK is designed to adapter to different chip platforms, such as RK3308, RK3288, RK3326, RK3399, RK3399pro, RK1808, PX30, PX3-SE, etc. Source code for different chip platforms will vary to some extent. When developers download source code, they should declare the chip platform they want firstly to avoid download the code they don't need.

SDK uses -m <chip platform_linux_release>.xml to declare the corresponding chip platform that you want to download.

RK3308_LINUX_SDK download command is as follows:

```
  mkdir rk3308
  cd rk3308
  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 rk3308_linux_release.xml
     .repo/repo/repo sync -c
```

Code will start to download automatically, just wait patiently. Source code file will be located under the corresponding project name in working directory. The initial sync operation will take one hour or more to complete.

5.4.2 SDK code compression package

To facilitate customers to quickly access SDK source code, Rockchip Technology Window usually provides corresponding version of SDK initial compression package. In this way, developers can obtain SDK source code from decompression the initial compression package, which is the same as the one downloaded by repo.

Take rk3308_linux_v1.00_20180510.tgz as an example. After copying initialization package, you can get source code by the following command:

```
mkdir rk3308
tar xvf rk3308_linux_v1.00_20180510.tgz -C rk3308
cd rk3308
.repo/repo/repo sync -l
.repo/repo/repo sync -c
```

Table 5-1 Rockchip SDK corresponding compression package

Chip platform	compression package	version
PX30	px30_linux_beta_v1.0_20180710.tgz	V1.00
RK3308	rk3308_linux_v1.00_20180510.tgz	V1.00
RK3326	rk3326_linux_sdk_20180426.tgz	V1.00
RK3399	rk3399_linux_sdk_v2.0_20180517.tgz	V2.00
RK1808	Rk1808_linux_v1.00_20181225.tgz	V1.00
RK3399PRO	rk3399pro_linux_sdk_beta_v0.01_20190217.tgz	V0.01

5.4.3 ReleaseNote

To facilitate customers to understand changes covered by SDK update and solved problems, ReleaseNote.txt is added to SDK, which records issues that are resolved by each update and whether customers are advised to update them all. After customers and developers update ReleaseNote, check updates and decide whether to update the SDK code.

ReleaseNote storage directory is shown in Table 5-2.

Table 5-2 Rockchip SDK corresponding ReleaseNote

Chip platform	**ReleaseNote**
PX30	.repo/manifests/px30_linux_v1.00/PX30_Release_Note.txt
RK3308	.repo/manifests/rk3308_linux_v1.00/RK3308_Release_Note.txt
RK3326	.repo/manifests/rk3326_linux_v1.00/RK3326_Release_Note.txt
RK3399	.repo/manifests/rk3399_linux_v2.00/RK3399_Release_Note.txt
RK1808	.repo/manifests/rk1808_linux_v1.00/RK1808_Release_Note.txt
RK3399PRO	.repo/manifests/rk3399pro_linux_v0.01/RK3399PRO_Release_Note.txt

Update ReleaseNote methods separately:

1. First record .repo/manifests/ directory current commit number commit1
2. Execute the following command to update ReleaseNote file.

```
cd .repo/manifests/
git pull origin linux
```

Read ReleaseNote, if you decide to update, execute the following command in root directory:

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

If you do not want to update code, execute the following command in .repo/manifests directory to roll back commit to the previous version:

```
git reset –hard commit1
```

5.5 SDK directory structure

After SDK download is complete, you can see the following directory structure in root directory:

```
    ├── buildroot
    ├── build.sh
    ├── device
    ├── docs
    ├── external
    ├── IMAGE
    ├── kernel
    ├── mkfirmware.sh
    ├── prebuilts
    ├── rkbin
    ├── rockdev
```

```
├── tools
└── u-boot
```

- Buildroot directory store buildroot open source project code, customizable root file system
- Build.sh is system compiling script, and executes the whole compilation of SDK.
- Device directory stores board level configurations and some preset files, boot scripts, etc.
- Docs directory store SDK rlated documents.
- External directory to store SDK related libraries and tools source code.
- IMAGE stores all image file backups and version compilation information after build.sh.
- kernel is kernel source code.
- mkfirmware.sh script can package image file and copy to rocketv/ directory uniformly.
- Prebuilts directory stores cross-compilation toolchains used by U-Boot and Kernel compilation.
- rkbin directory stores some key binary files of Rockchip platform, including ddr.bin, miniloader.bin, and bl31.bin, which are used during U-Boot compilation.
- Rockdev executes mkfirmware.sh to copy the generated image by system compilation to rockdev.
- Tools directory stores development tools, debugging tools, and production tools for Windows and Linux environments.
- u-boot directory stores source code of U-Boot.

5.6 SDK update

Developers can update synchronously by commands according to update instructions released regularly by FAE window.

```
.repo/repo/repo sync -c
```

5.7 SDK issues feedback

In order to help customers develop easily, Rockchip bug system (Redmine) records process and status of user issues, so that both parties can track at the same time, making problem processing more timely and efficient. For the future SDK issues, specific technical issues, technical consultations, etc. can be submitted to this bug system, Rockchip technical services will distribute, process and track issues in time.

See "Rockchip Bug System User Guide V1.0-201712.pdf "in docs\RKTools manuals\ directory for details.

Chapter 6. SDK Compile

6.1 Uboot compile

- **PX30 platform:**

```
cd u-boot
./make.sh evb-px30
```

After compiling, trust.img、px30_loader_v1.07.107.bin、uboot.img three image files will generate.

- **RK3308 platform:**

```
cd u-boot
./make.sh evb-rk3308
```

After compiling, trust.img、rk3308_loader_v1.17.101.bin、uboot.img three image files will generate.

RK3308 32bit compiling

```
cd u-boot
./make.sh evb-aarch32-rk3308
```

- **RK3326 platform:**

```
cd u-boot
./make.sh evb-rk3326
```

After compiling, trust.img, rk3326_loader_v1.07.110.bin, uboot.img three image files will generate.

RK3399 platform:

```
cd u-boot
./make.sh evb-rk3399
```

After compiling, trust.img, rk3399_loader_v1.12.112.bin, uboot.img three image files will generate.

RK1808 platform:

```
cd u-boot
./make.sh rk1808
```

After compiling, trust.img、rk1808_loader_v1.01.101.bin、uboot.img three image files will generate.

RK3399PRO platform:

```
cd u-boot
./make.sh rk3399pro
```

After compiling, trust.img、rk3399pro_loader_v1.15.115.bin、uboot.img three image files will generate.

6.2 Kernel compile

- **PX30 platform:**

PX30 EVB V10 board:

hardware version	Silk screen on board	Reference design
V10	PX30_Mini_EVB_V10_20180526	rk_evb_px30_ddr3p416dd6_v10.pdf

PX30	Board configuration file	Compile command
EVB	px30-evb-ddr3-v10-linux-v10.dts	cd kernel make px30_linux_defconfig make px30-evb-ddr3-v10-linux.img

- **RK3308 platform:**

1. RK3308 EVB V10 board

hardware version	Silk screen on board	Reference design
V10	RK_EVB_RK3308_DDR3P116SD4_V10_20180301	RK3308_AI-VA_BETA_V01_20180307

RK3308 EVB V10 boards are equipped with different microphone array boards. Different board configuration files are required, they are as follows:

Microphone array boards	Board configuration file	Compile commands
I2S digital microphone	rk3308-evb-dmic-i2s-v10.dts	cd kernel make rk3308_linux_defconfig make rk3308-evb-dmic-i2s-v10.img
Analog microphone	rk3308-evb-amic-v10.dts	cd kernel make rk3308_linux_defconfig make rk3308-evb-amic-v10.img
PDM digital microphone	rk3308-evb-dmic-pdm-v10.dts	cd kernel make rk3308_linux_defconfig make rk3308-evb-dmic-pdm-v10.img

2. RK3308 EVB V11 and V12 boards:

hardware version	Silk screen on board	Reference design
V11	RK_EVB_RK3308_DDR3P116SD4_V11_20180420 RK_EVB_RK3308_DDR3P116SD4_V12	RK3308_AI-VA_REF_V10

RK3308 EVB V11 and V12 boards are equipped with different microphone array boards. Different board configuration files are required, and differences are as follows:

Microphone array boards	Board configuration file	Compile commands
I2S digital microphone	rk3308-evb-dmic-i2s-v11.dts	cd kernel make rk3308_linux_defconfig make rk3308-evb-dmic-i2s-v11.img
Analog microphone	rk3308-evb-amic-v11.dts	cd kernel make rk3308_linux_defconfig

		make rk3308-evb-amic-v11.img
PDM digital microphone	rk3308-evb-dmic-pdm-v11.dts	cd kernel make rk3308_linux_defconfig make rk3308-evb-dmic-pdm-v11.img

After compiling is complete, kernel root directory generates a boot.img image file.

3. RK3308 Voice module board：

For details, see” RK3308 Voice Module 32-bit System Compilation Guide_v1.00.pdf “in docs\ SoC platform related\RK3308\ directory.

- **RK3326 platform**：

RK3326 EVB V11 board

hardware version	Silk screen on board	Reference design
V11	RK_EVB_RK3326_LP3178P132SD4_V11_20180301_FZB	RK_EVB_RK3326_LP3S178P132SD4_V11_20180301

RK3326	Board configuration file	Compile commands
EVB	rk3326-evb-linux-lp3-v10.dts	cd kernel make rk3326_linux_defconfig make rk3326-evb-linux-lp3-v10.img

- **RK3399 platform**：

RK3399 EVB V13 board

hardware version	Silk screen on board	Reference design
V13	RK_EXCAVATOR_MAIN_V13_20170911_FZB	RK_SAPPHIRE_SOCBOARD_RK3399_LPDDR3D178P232SD8_V12_20161109 and RK_EXCAVATOR_MAIN_V13_2017091

RK3399	Board configuration file	Compile commands
EVB	rk3399-sapphire-excavator-linux.dts	cd kernel make rockchip_linux_defconfig make rk3399-sapphire-excavator-linux.img

- **RK1808 platform**：

RK1808 EVB V11 board

hardware version	Silk screen on board	Reference design
V11	RK_EVB_RK1808_LP3D1789132SD6_V11_20181107_FINAL_LINT	RK_EVB_RK1808_LP3D1789132SD6_V11_20181107

RK1808	Board configuration file	Compile commands
EVB	rk1808-evb-v10.dts	cd kernel make rk1808_linux_defconfig make rk1808-evb-v10.img

- **RK3399PRO platform**:

RK3399PRO EVB V10 V11 board

hardware version	Silk screen on board	Reference design
V10	RK_EVB_RK3399PRO_LP35178P332SD8_V10	RK_EVB_RK3399PRO_LP3S178P332SD8_V10 20180921_RZF and RK_EVB_RK3399PRO_LP3S178P332SD8_V10 20180925_final_lint.brd
V11	RK_EVB_RK3399PRO_LP35178P332SD8_V11	RK_EVB_RK3399PRO_LP3S178P332SD8_V11 20181113_RZF and rk_evb_rk3399pro_lp3s178p332sd8_v11-20181115.brd

RK3399PRO	Board configuration file	Compile commands
EVB V10	rk3399pro-evb-v10-linux.dts	cd kernel make rockchip_linux_defconfig make rk3399pro-evb-v10-linux.img
EVB V11	rk3399pro-evb-v11-linux.dts	cd kernel make rockchip_linux_defconfig make rk3399pro-evb-v11-linux.img

6.3 Buildroot compiling

After costomers configure compilation dependencies according to actual compilation environment, follow the steps below to configure and execute make.

```
$ source buildroot/build/envsetup.sh

You're building on Linux
Lunch menu...pick a combo:
1. rockchip_rk3308_release
2. rockchip_rk3308_debug
3. rockchip_rk3308_robot_release
4. rockchip_rk3308_robot_debug
5. rockchip_rk3308_mini_release

Which would you like? [1]
```

If rockchip_rk3308_release is selected, enter the corresponding serial number 1.

```
$ make
```

After compiling, execute mkfirmware.sh script in SDK root directory to generate firmware.

```
$ ./mkfirmware.sh
```

All images required for burning will be copied to rockdev directory.

```
rockdev
 ├── boot.img
 ├── misc.img
 ├── parameter.txt
 ├── recovery.img
 ├── MiniLoaderAll.bin（即 rk3308_loader_v1.17.101.bin）
 ├── oem.img
 ├── userdata.img
 ├── rootfs.img
 ├── trust.img
 └── uboot.img
```

After all image files are obtained, in order to facilitate programming and mass production, these separate images can usually be manually packaged into an update.img by script. For packaging method, see Section 12.3 Firmware package tools. If you use fully automatic compilation script, it will automatically package update.img.

6.4 Fully automatic compilation script

In order to improve the efficiency of compiling and reduce possible misoperations caused by manual compiling, SDK integrates fully automatic compilation scripts to facilitate firmware compiling and backup.

1）The fully automated compilation script raw file is stored in:

```
device/rockchip/common/build.sh
```

2）when repo sync, copy to project root directory via copy option in manifest

3）Modify the specific variables in device/rockchip/rkxx (chip platform)/BoardConfig.mk script to generate corresponding product firmwares.

For example, RK3308 platform can modify device/rockchip/rk3308/BoardConfig.mk file:

```
#buildroot defconfig
LUNCH=rockchip_rk3308_release
#uboot defconfig
UBOOT_DEFCONFIG=evb-rk3308
#kernel defconfig
KERNEL_DEFCONFIG=rk3308_linux_defconfig
#kernel dts
KERNEL_DTS= rk3308-evb-dmic-pdm-v11
```

The following variables should be modified according to actual project situation:

```
LUNCH variable specifies Buildroot compiles defconfig.
KERNEL_DTS variable specifies product board configuration for compiled kernel.
```

4）Execute automatic compilation script:

```
./build.sh
```

The script automatically configures environment variables, compiles U-Boot, compiles Kernel, compiles Buildroot, compiles Recovery and generates firmware.

5) Script generation content:

Script will copy the compiled firmware to:

IMAGE/RK3308-EVB-DMIC-PDM-V11_****_RELEASE_TEST/IMAGES directory, specific paths are based on actual generation. Each time you compile, a new directory will be saved, the firmware version of debugging and developing process will be automatically backed up, and various information of firmware version will be stored.

6.5 Module compile

In order to facilitate developing and debugging, fully automatic compiling script also supports separate modules for compiling, which facilitates to module debuge, and specify and compile some modules.

Modules compiling can be found in instructions:

```
./build.sh -h
====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
cleanall             -clean uboot, kernel, rootfs, recovery
firmware             -pack all the image we need to boot up system
updateimg            -pack update image
save                 -save images, patches, commands used to debug
default              -build all modules
```

To compile kernel separately, just execute the following command:

```
./build.sh kernel
```

Chapter 7. SDK Image Upgrade

7.1 Overview

This chapter describes how to upgrade and run a compiled image file on a hardware device.

Introductions to several image upgrade tools provided by Rockchip platform are shown in Table 7-1. You can select an appropriate upgrade method for upgrade. Before upgrade, you need to install the latest USB driver. See chapter 4.3.2 for Rockchip USB driver installation.

Table 7-1 Rockchip platform upgrade tool

Tools	Operating system	Description
Rockchip development tools	Windows	Upgrade firmware separatly and the whole update firmware tool
FactoryTool	Windows	Mass production upgrade tool, support USB one-to-many upgrade
upgrade_tool	Linux	Development tools under Linux

7.2 Upgrade modes

Rockchip platform hardware operation modes are shown in Table 7-2. Only when devices are in Maskrom and Loader mode firmware can be burned or updated on board.

Table 7-2 Rockchip platform device modes

mode	Tool burning	Description
Maskrom	supported	When flash is not burning firmware, chip will boot into Maskrom mode, and the initial firmware can be burned. If Loader fails to start normally during developing and debugging process, you can also enter Maskrom mode to burn firmware.
Loader	supported	In Loader mode, firmware can be burned and upgraded. You can use a tool to separately write a partition image file for debugging.
Recovery	not supported	System boot recovery startup, the main role is to upgrade, restore factory settings.
Normal Boot	not supported	System boot rootfs startup and load rootfs. Most of developments are in this mode.

The following ways to enter upgrade mode：

1. Powers up, and enters Maskrom mode if firmware has not been downloaded.
2. After upgrading firmware, press and hold recovery button to power on or reset, system will enter Loader firmware upgrading mode.
3. After upgrading firmware, press and hold Maskrom button to power on or reset, system will enter MaskRom firmware upgrading mode.

4. After upgrading firmware, after power-on or reset, development board enters the system normally. On Rockchip development tool, it will display "Discover an ADB device" or "Discover an MSC device", then click button "Switch" on the tool to enter Loader mode
5. After upgrading firmware, you can enter reboot loader command in serial port or adb command line mode to enter Loader mode.

Take RK3308 EVB board as an example. Maskrom button is shown in Figure 7-1, and Recovery button is shown in Figure 7-2.

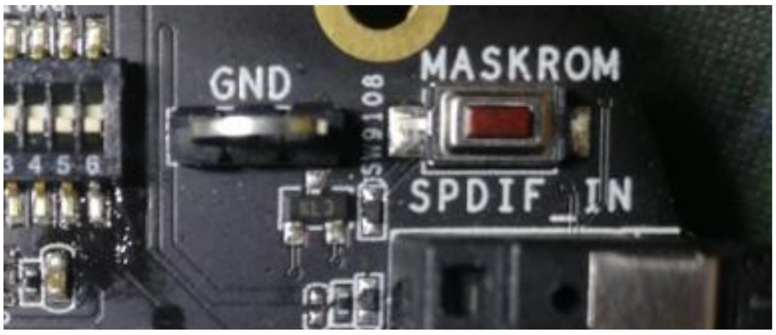


Figure 7-1 Maskrom button of RK3308 EVB board

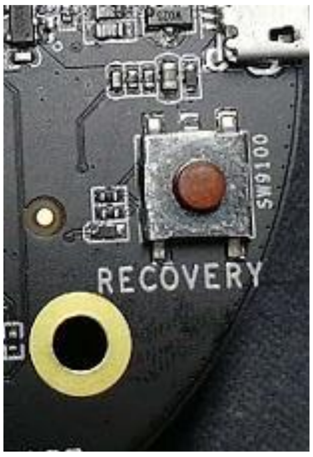


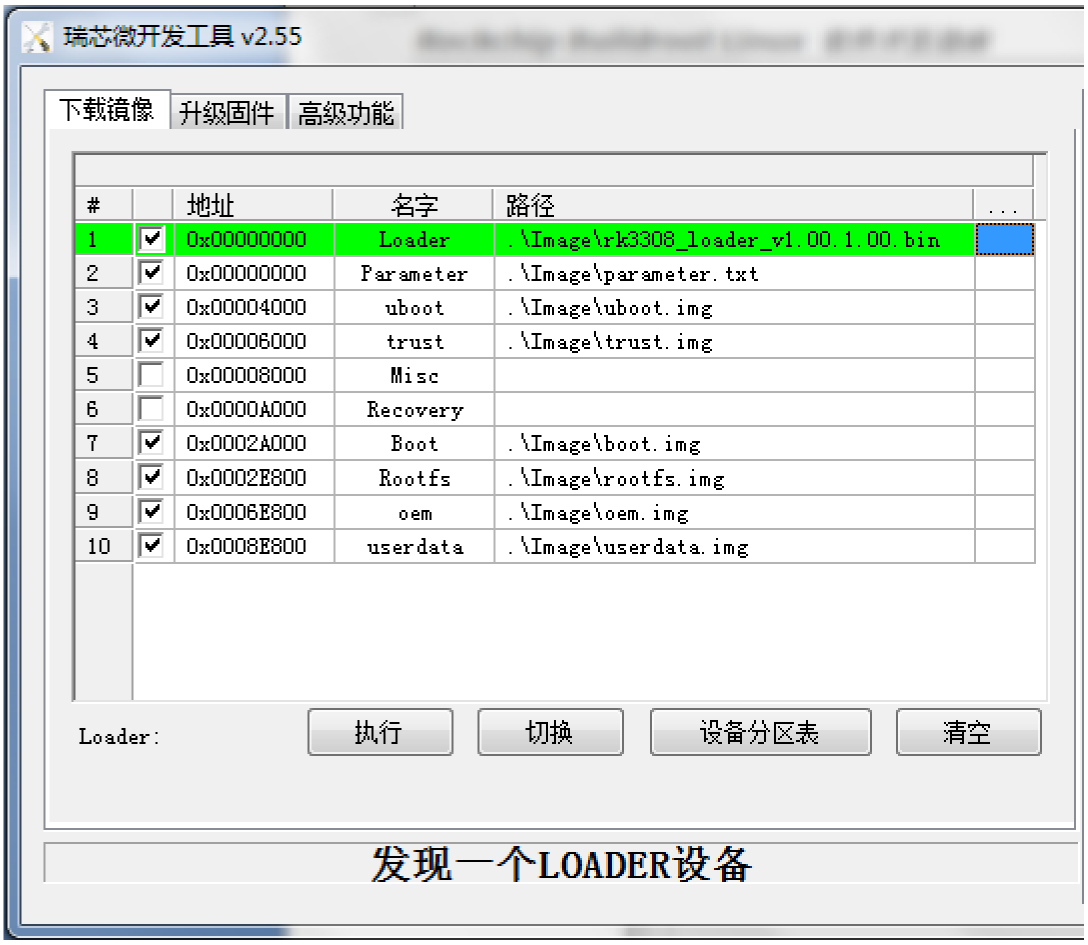
Figure 7-2 Recovery button of RK3308 EVB board

7.3 Rockchip development tools for upgrade

For details obout upgrade, please refer to "Rockchip Development Tool Manual.pdf" in docs\ RKTools manuals directory.

SDK provides an upgrade tool, as shown in Figure 7-3. After compiling and generating the corresponding firmware, you can enter upgrade mode to flash. For machines that have burned other firmware, you can choose to re-burn firmware, or select a low-grid device, erase idb, and then upgrade.

7.3.1 Download image



#		地址	名字	路径	...
1	✓	0x00000000	Loader	.\Image\rk3308_loader_v1.00.1.00.bin	
2	✓	0x00000000	Parameter	.\Image\parameter.txt	
3	✓	0x00004000	uboot	.\Image\uboot.img	
4	✓	0x00006000	trust	.\Image\trust.img	
5		0x00008000	Misc		
6		0x0000A000	Recovery		
7	✓	0x0002A000	Boot	.\Image\boot.img	
8	✓	0x0002E800	Rootfs	.\Image\rootfs.img	
9	✓	0x0006E800	oem	.\Image\oem.img	
10	✓	0x0008E800	userdata	.\Image\userdata.img	

Loader:

发现一个LOADER设备

Figure 7-3 Interface of Rockchip development tool for upgrade

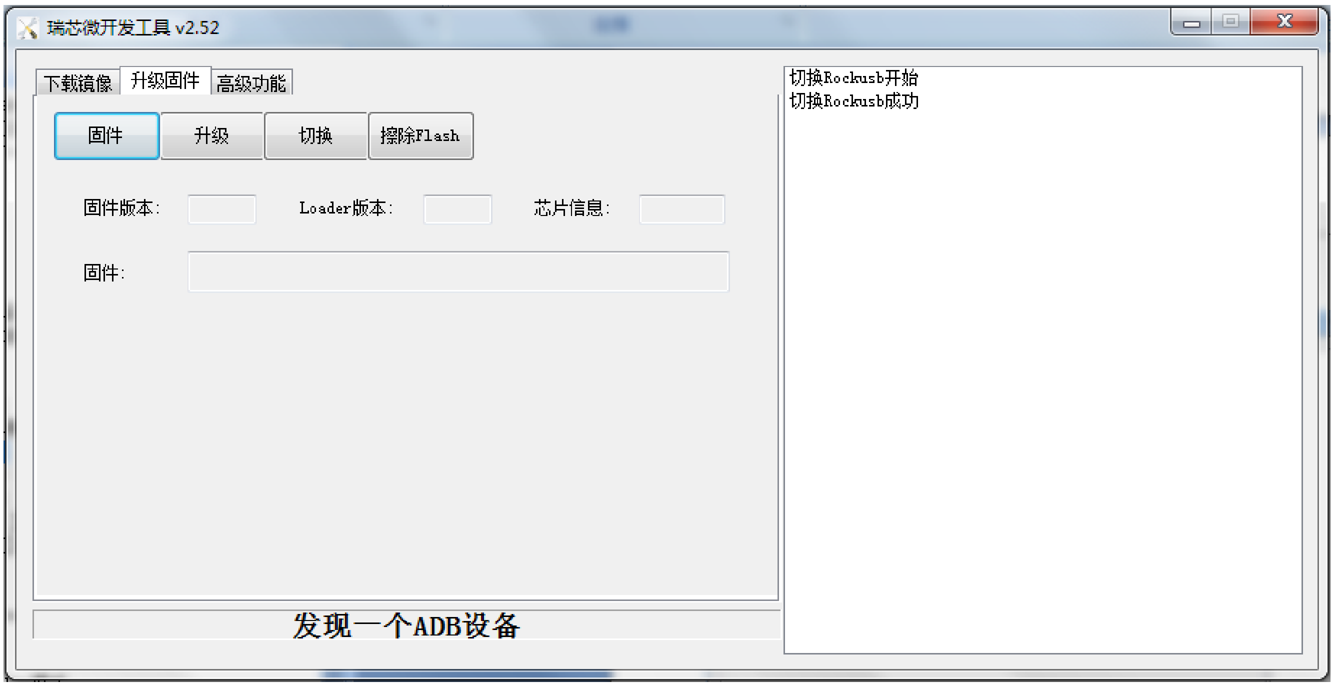
1) Hardware board enters burning mode by connected USB.

2) Open the tool and click download image menu. Click the ... column to pop up a file selection box, you can select img local address of the corresponding partition, and several other items are configured in turn.

3) After configuration, click execute, you will see blank box on the right to enter download prompt.

Click "execute" button to start burning, Click "clear" button is to clear edit box text.

7.3.2 Firmware upgrade

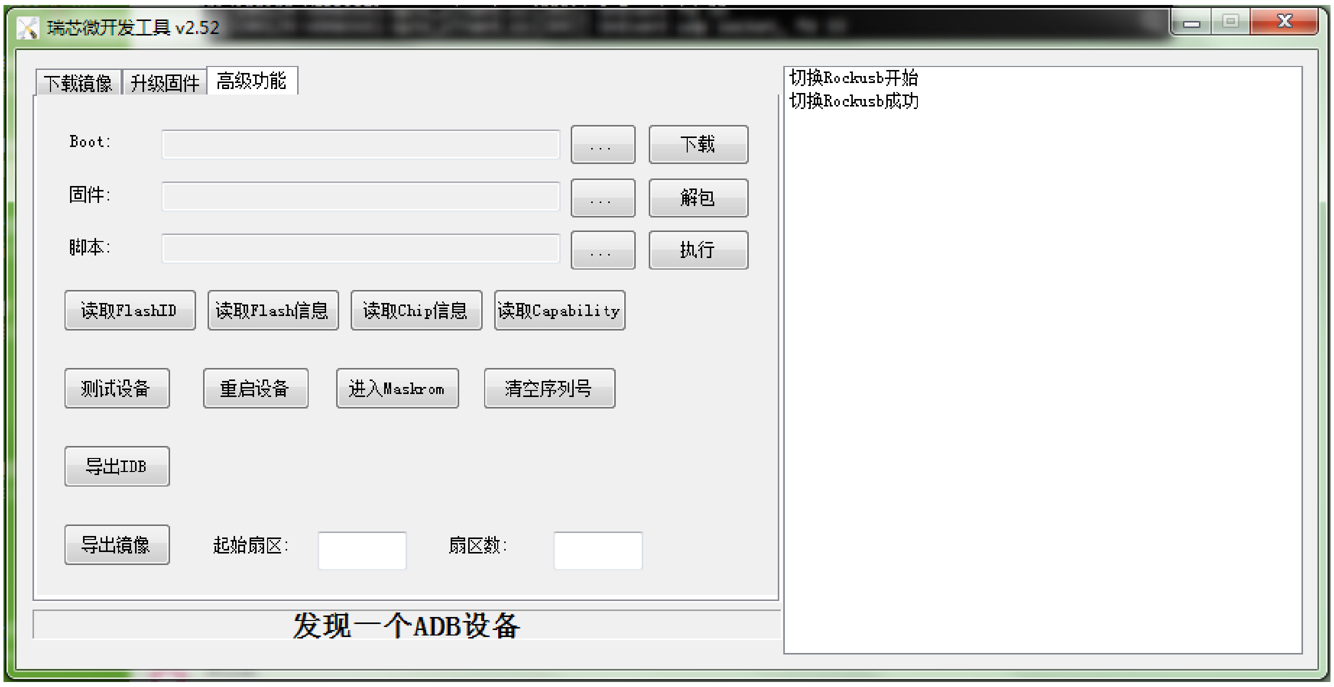


```
切换Rockusb开始
切换Rockusb成功
```

Figure 7-4 Firmware upgrade of android development tool

1) Click on firmware to select the packaged update.img file and click upgrade button to download it. (Note that device must be in download mode).

7.3.3 Advanced features



```
切换Rockusb开始
切换Rockusb成功
```

Figure 7-5 Advanced Features of Android development tool

Boot can only choose the packaged update.img file or loader file;
Firmware must use the packaged update.img;
Unpacking function can disassemble update.img into various partial image files.

7.4 Upgrade_tool

For details about upgrade, please refer to "Linux Development Tools User Manual_v1.32.pdf" in the docs\ RKTools manuals directory.

Connect to Linux host by USB cable. Make sure your board enters upgrade mode. Upgrade command is as follows:

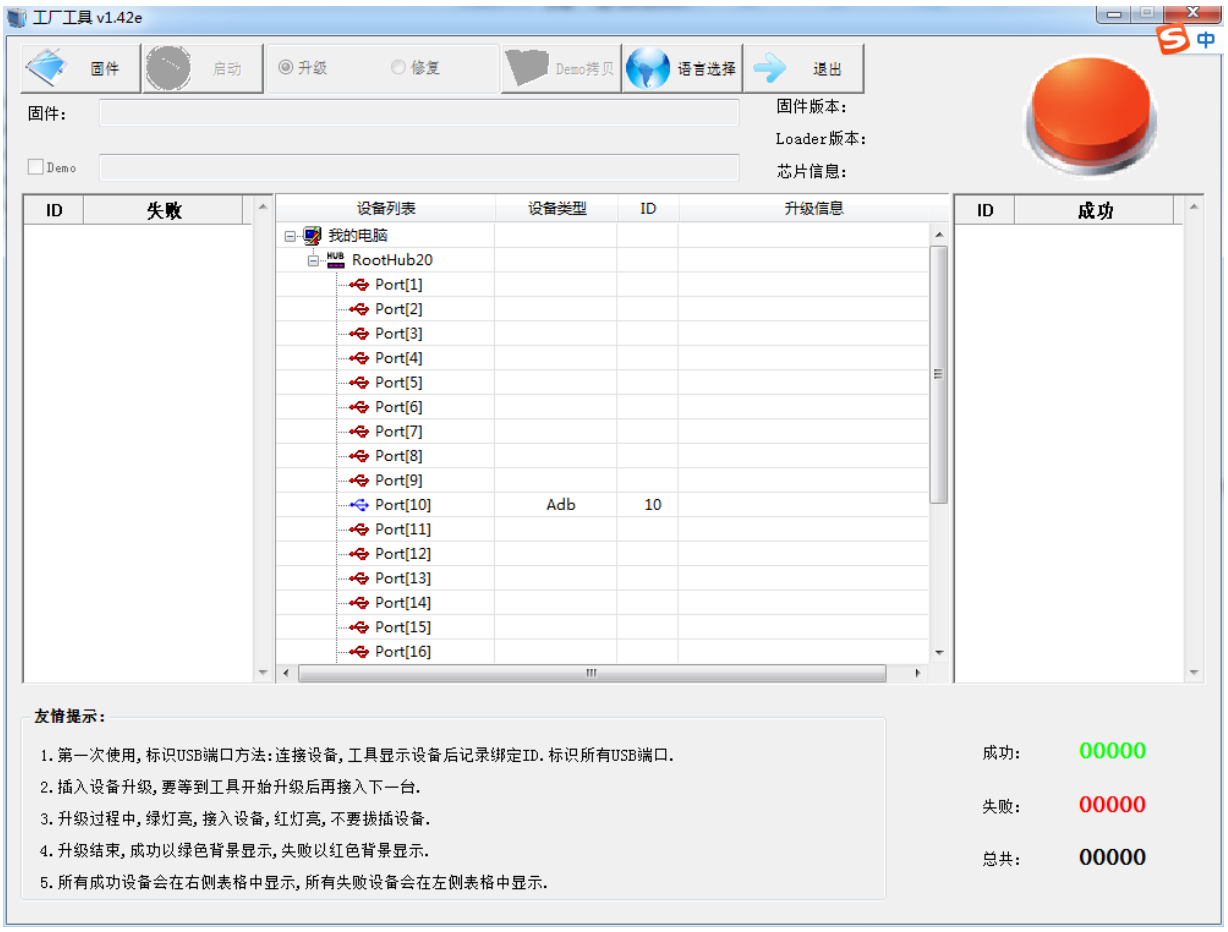
```
sudo ./upgrade_tool ul rk3308_loader_v1.17.101.bin
sudo ./upgrade_tool di -p parameter.txt
sudo ./upgrade_tool di -u uboot.img
sudo ./upgrade_tool di -t trust.img
sudo ./upgrade_tool di -misc misc.img
sudo ./upgrade_tool di -r recovery.img
sudo ./upgrade_tool di -b boot.img
sudo ./upgrade_tool di -rootfs rootfs.img
sudo ./upgrade_tool di -oem oem.img
sudo ./upgrade_tool di -userdata userdata.img
sudo ./upgrade_tool rd
```

7.5 FactoryTool

1) Click firmware button and select update.img packaged by packaging tool, waiting for unpacking to succeed.

2) Connect device and put the device into Loader or Maskrom mode, tool will automatically download.

3) It is possible to connect multiple devices at the same time, and one-to-many upgrade improve efficiency of factory upgrade.



固件:

Demo

固件版本:

Loader版本:

芯片信息:

设备列表	设备类型	ID	升级信息
我的电脑			
RootHub20			
Port[1]			
Port[2]			
Port[3]			
Port[4]			
Port[5]			
Port[6]			
Port[7]			
Port[8]			
Port[9]			
Port[10]	Adb	10	
Port[11]			
Port[12]			
Port[13]			
Port[14]			
Port[15]			
Port[16]			

友情提示:

1. 第一次使用, 标识USB端口方法:连接设备, 工具显示设备后记录绑定ID. 标识所有USB端口.
2. 插入设备升级, 要等到工具开始升级后再接入下一台.
3. 升级过程中, 绿灯亮, 接入设备, 红灯亮, 不要拔插设备.
4. 升级结束, 成功以绿色背景显示, 失败以红色背景显示.
5. 所有成功设备会在右侧表格中显示, 所有失败设备会在左侧表格中显示.

成功:	00000
失败:	00000
总共:	00000

Figure 7-6 Factory Tool

7.6 Mass production upgrade

For mass production, taking into account production efficiency and factory station arrangement, please refer to "Rockchip mass production upgrade Guide V1.1-20170214.pdf" in docs\RKTools manuals directory for details.

Rockchip Linux SDK currently supports USB upgrades and burner upgrades.

If you have problems with tools during mass production, you can contact our FAE window <fae@rock-chips.com>.

Chapter 8. U-Boot Develop

This chapter briefly introduces basic concepts of U-Boot and precautions for compiling, helps customers understand U-Boot framework of RK platform. For details on U-Boot development, please refer to docs\Develop reference documents directory "Rockchip-Developer-Guide-UBoot-nextdev.pdf".

8.1 Rockchip U-Boot

Rockchip U-Boot next-dev branch is the version Rockchip cut out from official version of U-Boot v2017.09 for development. Currently, all mainstream RK chips are already supported on the platform.

- Supported chips：rk3288, rk3036, rk312x, rk3288, rk3308, rk3326, rk3399, rk3399pro, rk1808, px30, etc;
- Support firmware booting up of RK Android platform;
- Support the latest Android AOSP (such as GVA) firmware booting up;
- Support for Linux Distro firmware booting up;
- Supports Rockchip miniloader and SPL/TPL pre-loader booting;
- Support LVDS, EDP, MIPI, HDMI and other display devices
- Support eMMC, Nand Flash, SPI NOR flash, SD cards, U disk and other storage devices booting up；
- Support FAT, EXT2, EXT4 file system；
- Support GPT, RK parameter partition format；
- Support boot logo display, charging animation display, low power management, power management;
- Support I2C, PMIC, CHARGE, GUAGE, USB, GPIO, PWM, GMAC, EMMC, NAND, interrupt and other drivers;
- Supports two USB gadgets to burn EMMC: RockUSB and Google Fastboot;
- Support USB devices such as Mass storage, ethernet, HID;
- Support use of Kernel's dtb;
- Support dtbo function;

8.2 Platform compile

8.2.1 rkbin

rkbin project mainly stores bin files (trust, loader, etc.), scripts, package tools, etc. that Rockchip does not open source, so rkbin is just a "toolkit " project.

rkbin project needs to maintain the same directory level with U-Boot project, otherwise rkbin repository will not be found when compiling. When compiling in U-Boot project, compiling script will index the related bin files and packaging tools from rkbin repository, and finally generate the related firmware such as trust.img, uboot.img, loader and so on in U-Boot root directory.

8.2.2 gcc Tool chain

The default compiler version used is gcc-linaro-6.3.1:

```
  32 bit compiler: gcc - linaro - 6.3.1 - 2017.05 - x86_64_arm - linux - gnueabihf
  64 bit compiler: gcc - linaro - 6.3.1 - 2017.05 - x86_64_aarch64 - linux - gnu
```

By default, tool package provided by Rockchip is prebuilts, please ensure that it maintains the same directory level with U-Boot project.

If you need to change compiler path, modify contents of the compiling script ./make.sh:

```
  GCC_ARM32=arm - linux - gnueabihf -
  GCC_ARM64=aarch64 - linux - gnu -
  TOOLCHAIN_ARM32=../prebuilts/gcc/linux - x86/arm/gcc - linaro - 6.3.1 - 2017.05
 - x86_64_arm - linuxgnueabihf/
  bin
  TOOLCHAIN_ARM64=../prebuilts/gcc/linux - x86/aarch64/gcc - linaro - 6.3.1 -
2017.05 - x86_64_aarch64 -
  linux - gnu/bin
```

8.2.3 Compiling

Compiling command:

```
  ./make.sh [board] - - - The source of [board] name is: configs/[board]_defconfig
file
```

Command examples:

```
  ./make.sh evb- rk3308 - - -  build for evb- rk3308_defconfig
  ./make.sh firefly- rk3288 - - -  build for firefly- rk3288_defconfig
```

Chapter 9. Kernel Develop

This chapter briefly introduces some common configuration changes in kernel, mainly configuration of dts, to help customers make some simple modifications faster and more convenient. Kernel version is introduced based on 4.4.

9.1 DTS introduction

9.1.1 Overview

Earlier versions of Linux Kernel were configured to directly configure board-related information in board configuration file, such as IOMUX, default pull-high/low GPIO, and client device information under each I2C/SPI bus. To abandon this 'hard code' method, Linux introduced the concept of Device Tree to describe different hardware structures.

Device Tree data is highly readable, conforms to DTS specification, and is usually described in .dtsi and .dts source files. In the process of kernel compiling, it is compiled into a .dtb binary. During boot-up phase, dtb is loaded into a RAM address space by bootloader (such as U-Boot) and passed to Kernel space as a parameter. Kernel parses the entire dtb file and refines each device information for initialization.

This document aims to introduce how to add a board dts configuration and some common dts syntax descriptions. More detailes about dts syntax is beyond the scope of this document. If you are interested, please refer to:

1. https://www.devicetree.org/specifications/
2. Documentation/devicetree/bindings

9.1.2 Add a new product DTS

9.1.2.1 Create a dts file

Linux Kernel currently supports multi-platform using dts, RK platform dts files are stored in：

```
ARM:arch/arm/boot/dts/
ARM64    :arch/arm64/boot/dts/rockchip
```

The general dts file naming rule is "soc-board_name.dts", such as rk3308-evb-dmic-i2s-v10.dts.

Soc refers to chip name, board_name is generally named according to board silk screen.

If your board is an integrated board, you only need a dts file to describe it.

```
rk3308-ai-va-v10.dts
   └─── rk3308.dtsi
```

If hardware is a structure of a core board and a subboard, or if product has multiple product forms, common hardware descriptions can be placed in dtsi file, and dts file describes different hardware modules, and a common hardware description is included by including "xxx.dtsi".

```
   ├─── rk3308-evb-amic-v10.dts
```

```
    |     ├─── rk3308-evb-ext-v10.dtsi
    |     └─── rk3308-evb-v10.dtsi
    |          └─── rk3308.dtsi
    └─── rk3308-evb-dmic-i2s-v10.dts
          ├─── rk3308-evb-ext-v10.dtsi
          └─── rk3308-evb-v10.dtsi
               └─── rk3308.dtsi
```

9.1.2.2 Modify Makefile of the directory where dts is located

```
  diff --git a/arch/arm64/boot/dts/rockchip/Makefile
b/arch/arm64/boot/dts/rockchip/Makefile
  index 073281d..7c329d4 100644
  --- a/arch/arm64/boot/dts/rockchip/Makefile
  +++ b/arch/arm64/boot/dts/rockchip/Makefile
  @@ -1,6 +1,9 @@
   dtb-$(CONFIG_ARCH_ROCKCHIP) += px30-evb-ddr3-v10.dtb
   dtb-$(CONFIG_ARCH_ROCKCHIP) += px30-evb-ddr3-lvds-v10.dtb
   dtb-$(CONFIG_ARCH_ROCKCHIP) += px30-evb-ddr4-v10.dtb
  +dtb-$(CONFIG_ARCH_ROCKCHIP) += rk3308-evb-amic-v10.dtb
```

When compiling kenrel, you can directly make dts-name.img (such as rk3308-evb-amic-v10.img) to generate the corresponding resource.img (including dtb data).

9.1.2.3 A few notes on dts syntax

Dts syntax can be like c/c++, including other public dts data via #include xxx.dtsi. Dts file will inherit attributes and values of all device nodes of included dtsi file.

If property is defined in multiple dts/dtsi files, its value is finally defined by dts. All chip-related controller nodes are defined in soc.dtsi. To enable device function, you need to set its status to "okay" in dts file.

```
    /dts-v1/;
    #include "rk3308-evb-v10.dtsi"

    / {
        /* rk3308-evb-v10.dtsi There is also a definition of this attribute, and the final
value is the defined value of dts. */
        model = "Rockchip RK3308 evb analog mic board";
        compatible = "rockchip,rk3308-evb-amic-v10", "rockchip,rk3308";

        /* The node under root node is a new device node. */
        sound {
            compatible = "simple-audio-card";
            ...
        };
    };
    /* The node under non-root node is device node described by reference dtsi, and
can reset value. */
    &i2s_8ch_2 {
        status = "okay";

        #sound-dai-cells = <0>;
    };
```

9.2 Reference documents of kernel modules develop

In docs\ Develop reference documents\ directory, the corresponding development documents are released according to function modules. In this section, we mainly conclude an index on these development documents. Combination of practical development problems, refer to the following table to read corresponding development guides.

function modules	Subdirectory	Related documents
PWM、Backlight	PWM	Rockchip Backlight Control Developer Guide V0.1-20160729.pdf
Secure, encrypted、OPTEE	SECURITY	Rockchip TEE Security SDK Developer Guide V1.0.pdf Rockchip-Secure-Boot-Application-Note.pdf
USB	USB	USB-Initialization-Log-Analysis.pdf Rockchip-USB-SQ-Test-Guide.pdf Rockchip-USB-Performance-Analysis-Guide.pdf Rockchip-Developer-Guide-linux4.4-USB.pdf RK3399-USB-DTS.pdf
UART develop configuration	UART	Rockchip-Developer-Guide-linux4.4-UART.pdf
Trust、ATF	TRUST	Rockchip-Developer-Guide-Trust.pdf
Tsadc,Temperature control, THERMAL	THERMAL	Rockchip Thermal Developer Guide-4.4 V1.0.1-20170428.pdf
SPI	SPI	Rockchip-Developer-Guide-linux4.4-SPI.pdf
PMIC、Voltmeter、DC-DC	PMIC	Rockchip-Developer-Guide-Linux4.4-DC-DC.pdf Rockchip RK818_6 Voltmeter Developer Guide V2.0-20170525.pdf Rockchip RK805 Developer Guide V1.0-20170220.pdf
GPIO 、 IOMUX 、 Pin-Ctrl	Pin-Ctrl	Rockchip Pin-Ctrl Developer Guide V1.0-20160725.pdf
PCIe	PCIe	Rockchip-Developer-Guide-linux4.4-PCIe.pdf
eMMC 、 SDIO 、 SDMMC	MMC	Rockchip-Developer-Guide-linux4.4-SDMMC-SDIO-eMMC.pdf
IO-DOMAIN 、 Power domain	IO-DOMAIN	Rockchip IO-Domain Developer Guide V1.0-20160630.pdf

configuration		
I2C	I2C	Rockchip I2C Developer Guide V1.0-20160629.pdf
GMAC、Ethernet	GMAC	Rockchip Ethernet Developer Guide V2.3.1-20160708.pdf
DVFS、CPU Freq	DVFS	Rockchip CPU-Freq Developer Guide V1.0.1-20170213.pdf
CLK 、 Clock configuration	CRU	Rockchip Clock Submodule Developer Guide V1.1-20170210.pdf
Audio	Audio	Rockchip Audio Developer Guide V1.1-20170215-linux4.4.pdf RK3308_Audio_Introduction_v0.1.0_CN.pdf
DDR 、 DDR stability	DDR	RK DDR Application Note 5 -- Customers verify their own DRAM process.pdf DDR issues debug Manual 1.0.pdf DDR Developer Guide.pdf
Display、DRM	DISPLAY	Rockchip HDMI based on DRM framework Developer Guide v1.1-20180322.pdf rockchip_drm_integration_helper-zh.pdf Rockchip_DRM_Panel_Porting_Guide_V1.3_20171209

Chapter 10. Buildroot Develop

This chapter briefly introduce some common configuration changes in Buildroot development. Rockchip Buildroot Linux SDK is Buildroot-2018.02 version.

10.1 Buildroot Basic Development

Buildroot Basic Development Guide, see"The Buildroot User Manual.pdf" under docs\Linux reference documents\directory.

Or visit the official website directly:
http://buildroot.org/downloads/manual/manual.html

10.1.1 Default configuration selection and compiling

After you configure compilation dependencies according to actual compilation environment, follow the steps below to configure and execute make.

```
$ source buildroot/build/envsetup.sh

You're building on Linux
Lunch menu...pick a combo:
1. rockchip_rk3308_release
2. rockchip_rk3308_debug
3. rockchip_rk3308_robot_release
4. rockchip_rk3308_robot_debug
5. rockchip_rk3308_mini_release

Which would you like? [1]
```

If you select rockchip_rk3308_release, enter the corresponding serial number 1.

```
$ make
```

After compiling, execute mkfirmware.sh script in SDK root directory to generate firmware.

Make to execute the following process:

- Download source code;
- Configure, compile, and install cross toolchains;
- Configure, compile, and install selected packages;
- Install the selected format to generate root file system;

Output of buildroot is saved in output directory. The specific directory is determined by configuration file. The above example is saved in buildroot/output/rockchip_rk3308_release directory. The following compiling can be executed in buildroot/output/rockchip_rk3308_release directory or project root directory (make menuconfig can also be executed in project root directory), this directory contains several subdirectories:

- image: Contains the compressed root file system image file.
- build/: Contains all source files, including host tools and selected packages for Buildroot. This directory contains all module source code.

- staging/: This directory is similar to the directory structure of root file system. It contains all header files and libraries generated by compiling, as well as other development files. However, they are not cut and are not suitable for target file system.
- target/: Contains complete root file system. Compared to staging, it has no development files, no header files, and binary files are also processed by strip.
- host/: Tools required for host compilation include cross-compilation tools.

10.1.2 Module configuration compile

The above steps select a default configuration, but not always meet our needs. We may need to add some third-party packages, or modify configuration options of package. Buildroot supports graphical way to select configurations.

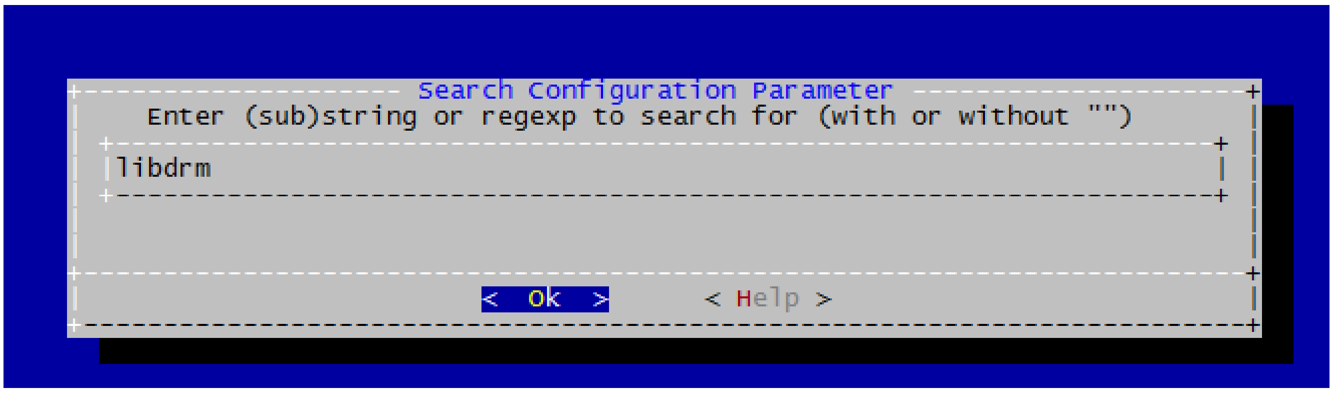
Buildroot support make menuconfig|nconfig|gconfig|xconfig

For example, add libdrm support

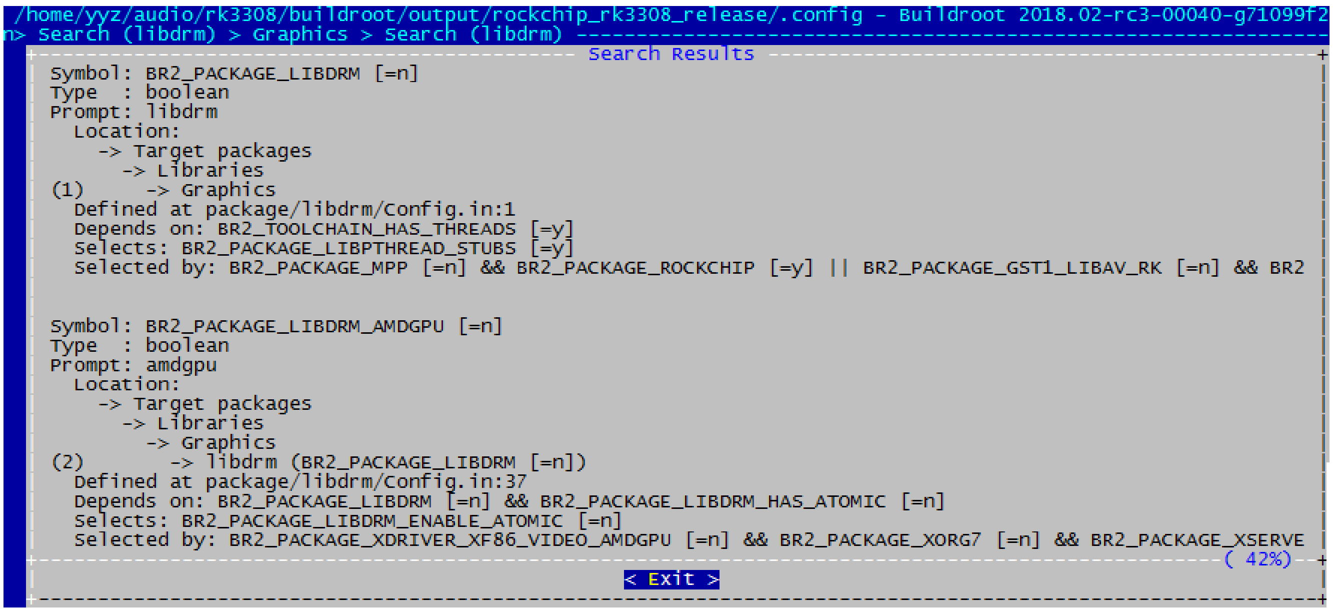
After executing source command and configuring environment variables, in root directory:

```
Make menuconfig
```

Enter /, pop-up search interface as follows, enter libdrm, hit Enter:

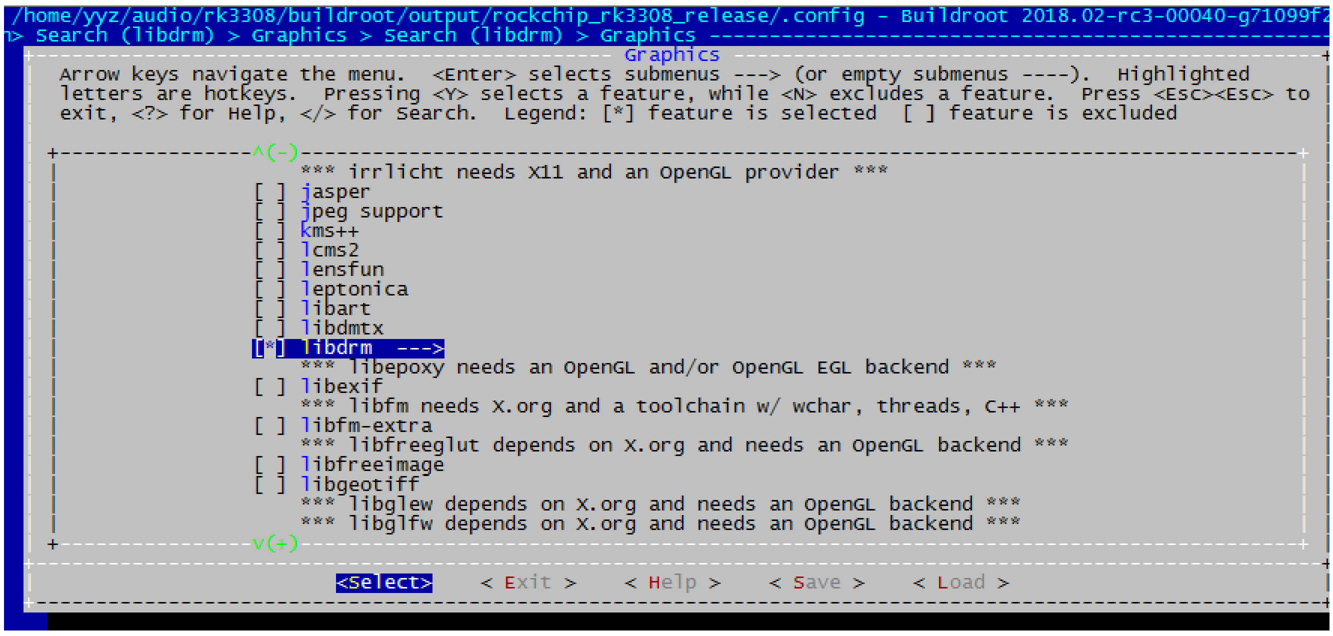


```
Search Configuration Parameter
Enter (sub)string or regexp to search for (with or without "")
libdrm
```



```
/home/yyz/audio/rk3308/buildroot/output/rockchip_rk3308_release/.config - Buildroot 2018.02-rc3-00040-g71099f2
> Search (libdrm) > Graphics > Search (libdrm)
                              Search Results
Symbol: BR2_PACKAGE_LIBDRM [=n]
Type  : boolean
Prompt: libdrm
  Location:
    -> Target packages
      -> Libraries
(1)     -> Graphics
  Defined at package/libdrm/Config.in:1
  Depends on: BR2_TOOLCHAIN_HAS_THREADS [=y]
  Selects: BR2_PACKAGE_LIBPTHREAD_STUBS [=y]
  Selected by: BR2_PACKAGE_MPP [=n] && BR2_PACKAGE_ROCKCHIP [=y] || BR2_PACKAGE_GST1_LIBAV_RK [=n] && BR2


Symbol: BR2_PACKAGE_LIBDRM_AMDGPU [=n]
Type  : boolean
Prompt: amdgpu
  Location:
    -> Target packages
      -> Libraries
        -> Graphics
(2)       -> libdrm (BR2_PACKAGE_LIBDRM [=n])
  Defined at package/libdrm/Config.in:37
  Depends on: BR2_PACKAGE_LIBDRM [=n] && BR2_PACKAGE_LIBDRM_HAS_ATOMIC [=n]
  Selects: BR2_PACKAGE_LIBDRM_ENABLE_ATOMIC [=n]
  Selected by: BR2_PACKAGE_XDRIVER_XF86_VIDEO_AMDGPU [=n] && BR2_PACKAGE_XORG7 [=n] && BR2_PACKAGE_XSERVE
                                                                                             ( 42%)
```

Select 1, then hit space to select on.


```
/home/yyz/audio/rk3308/buildroot/output/rockchip_rk3308_release/.config - Buildroot 2018.02-rc3-00040-g71099f2
> Search (libdrm) > Graphics > Search (libdrm) > Graphics ----------------------------------------------------
+----------------------------------------- Graphics ----------------------------------------------------------+
|  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted
|  letters are hotkeys.  Pressing <Y> selects a feature, while <N> excludes a feature.  Press <Esc><Esc> to
|  exit, <?> for Help, </> for Search.  Legend: [*] feature is selected  [ ] feature is excluded
|
| +---------------^(-)------------------------------------------------------------------------------------+
| |                    *** irrlicht needs X11 and an OpenGL provider ***
| |                [ ] jasper
| |                [ ] jpeg support
| |                [ ] kms++
| |                [ ] lcms2
| |                [ ] lensfun
| |                [ ] leptonica
| |                [ ] libart
| |                [ ] libdmtx
| |                [*] libdrm  --->
| |                    *** libepoxy needs an OpenGL and/or OpenGL EGL backend ***
| |                [ ] libexif
| |                    *** libfm needs X.org and a toolchain w/ wchar, threads, C++ ***
| |                [ ] libfm-extra
| |                    *** libfreeglut depends on X.org and needs an OpenGL backend ***
| |                [ ] libfreeimage
| |                [ ] libgeotiff
| |                    *** libglew depends on X.org and needs an OpenGL backend ***
| |                    *** libglfw depends on X.org and needs an OpenGL backend ***
| +---------------v(+)------------------------------------------------------------------------------------+
+-------------------------------------------------------------------------------------------------------------+
|                    <Select>    < Exit >    < Help >    < Save >    < Load >
+-------------------------------------------------------------------------------------------------------------+
```

Then save and exit, save the configuration command to modify the configuration file in buildroot/configs/directory;

```
make savedefconfig
```

Compile libdrm and generate libdrm

```
make libdrm
```

10.1.3 Busybox configuration modification

Configuration command:

```
make busybox-menuconfig
```

After modification is complete, save the configuration by command:

```
make busybox-update-config
```

10.1.4 Cross compilation tool

After Buildroot compiling, a cross compilation tool will be generated in the specified output host directory, which we can use to compile target program. The cross compilation tool directory generated by default configuration is:

```
buildroot/output/rockchip_rk3308_release/host/usr/bin/output/host/usr/bin/
```

We can compile program directly with cross compilation tool:

```
./buildroot/output/rockchip_rk3308_release/host/usr/bin/output/host/usr/bin/aarch64-rockchip-linux-gnu-gcc main.c -o test
```

Floating point support (the following configuration opens neon support), RK3308 supports crc/crypto/fp/simd these features, the configuration is as follows:

```
CFLAGS += -mcpu=cortex-a35+crc+crypto
```

10.1.5 Add local source package

The above introductions are in the case of Buildroot obtained source package, we just choose to open compilation, if Buildroot do not or how our own application integrate into Buildroot?

Buildroot supports a variety of module compilation methods, including generic-package, cmake-package, autotools-package, etc. We use generic-package as an example;

Example：package/rockchip/alsa_capture

1. Create a directory: package/rockchip/alsa_capture
2. Create Config.in

```
config BR2_PACKAGE_ALSA_CAPTURE
    bool "Simple ALSA Capture Demo"
```

3. Create alsa_capture.mk，Where the source directory points to external/alsa_capture/src

```
###############################################################
#
# alsa_capture
#
###############################################################
ifeq ($(BR2_PACKAGE_ALSA_CAPTURE), y)
ALSA_CAPTURE_VERSION:=1.0.0
ALSA_CAPTURE_SITE=$(TOPDIR)/../external/alsa_capture/src
ALSA_CAPTURE_SITE_METHOD=local

define ALSA_CAPTURE_BUILD_CMDS
    $(TARGET_MAKE_ENV) $(MAKE) CC=$(TARGET_CC) CXX=$(TARGET_CXX) -C $(@D)
endef

define ALSA_CAPTURE_CLEAN_CMDS
    $(TARGET_MAKE_ENV) $(MAKE) -C $(@D) clean
endef

define ALSA_CAPTURE_INSTALL_TARGET_CMDS
    $(TARGET_MAKE_ENV) $(MAKE) -C $(@D) install
endef

define ALSA_CAPTURE_UNINSTALL_TARGET_CMDS
    $(TARGET_MAKE_ENV) $(MAKE) -C $(@D) uninstall
endef

$(eval $(generic-package))
endif
```

4. Create a directory: external/alsa_capture/src, write: alsa_capture.c

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
printf(“hello world\n”);
return 0;
}
```

5. Write Makefile

```
DEPS =
OBJ = alsa_capture.o
CFLAGS = -std=c++11 -lasound
%.o: %.cpp $(DEPS)
	$(CC) -c -o $@ $< $(CFLAGS)

alsa_capture: $(OBJ)
	$(CXX) -o $@ $^ $(CFLAGS)

.PHONY: clean

clean:
	rm -f *.o *~ alsa_capture

.PHONY: install

install:
	cp -f alsa_capture $(TARGET_DIR)/usr/bin/

.PHONY: uninstall

uninstall:
	rm -f $(TARGET_DIR)/usr/bin/alsa_capture
```

6. Add the new package to Buildroot compile system;
7. Modify package/rockchip/Config.in to add the following line

```
source "package/rockchip/alsa_capture/Config.in"
```

8. Configuration the selected package

```
make menuconfig, then choosealsa_capture package;
```

9. compiling

```
make alsa_capture
```

10. Packaged into file system

```
make
```

11. Recompile the package after modifying source code

```
make alsa_capture-rebuild
```

10.1.6 fs-overlay

Some configuration files of the root file system is compiled by default may not meet customization requirements. At this time, fs-overlay can be used. The fs-overlay directory will be replaced in file system directory at the final stage of compilation, and packaged into root file system. The fs-overlay path is specified by default configuration file.

```
BR2_ROOTFS_OVERLAY="board/rockchip/rk3308/fs-overlay"
```

For example, we will modify fstab file to add debugfs and pstore:

```
vi board/rockchip/rk3308/fs-overlay/etc/fstab
# <file system> <mount pt>        <type>   <options>         <dump>   <pass>
/dev/root          /                  ext2      rw,noauto          0          1
proc               /proc              proc      defaults           0          0
devpts             /dev/pts           devpts   defaults,gid=5,mode=620 0         0
tmpfs              /dev/shm           tmpfs     mode=0777          0          0
tmpfs              /tmp               tmpfs     mode=1777          0          0
tmpfs              /run               tmpfs     mode=0755,nosuid,nodev   0         0
sysfs              /sys               sysfs     defaults           0          0
debug              /sys/kernel/debug          debugfs defaults           0          0
pstore             /sys/fs/pstore   pstore   defaults           0          0
```

10.1.7 SDK common configuration modification

10.1.7.1 Flash type modification

Configuration file: device/rockchip/rk3308/BoardConfig.mk, the default configuration is nand device.

```
# Set flash type.
# support <emmc, nand, spi_nand, spi_nor>
FLASH_TYPE := emmc
```

Configuration instructions:

- EMMC device

```
FLASH_TYPE=emmc
```

- NAND device

```
FLASH_TYPE=nand
```

10.1.7.2 Rootfs change to ext2

Rootfs can be configured to readable and writable ext2 file systems for system debugging.

1. Modify bootargs configuration in Kernel:

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3308-evb-v10.dtsi
b/arch/arm64/boot/dts/rockchip/rk3308-evb-v10.dtsi
index d39faf8..549af2e 100644
--- a/arch/arm64/boot/dts/rockchip/rk3308-evb-v10.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3308-evb-v10.dtsi
@ -12,7 +12,7 @
compatible = "rockchip,rk3308-evb", "rockchip,rk3308";
chosen {
-                    bootargs = "earlycon=uart8250,mmio32,0xff0c0000 swiotlb=1
console=ttyFIQ0 root=PARTUUID=614e0000-0000 rootfstype=squashfs rootwait";
+                    bootargs = "earlycon=uart8250,mmio32,0xff0c0000 swiotlb=1
console=ttyFIQ0 root=PARTUUID=614e0000-0000 rootfstype=ext2 rootwait";
          };
```

2. Modify parameter file corresponding to device\rockchip\rk3308\rockimg\ to ensure that rootfs partition size is sufficient to store partition image.
3. Modify rootfs file system type in device\rockchip\rk3308\BoardConfig.mk:

```
diff --git a/BoardConfig.mk b/BoardConfig.mk
index 8adbcdd..3a2be4c 100755
--- a/BoardConfig.mk
+++ b/BoardConfig.mk
@@ -2,8 +2,8 @@
 SDK_ROOT := $(PWD)/../../..

  # Set rootfs type, see buildroot.
-# ext4 squashfs
-ROOTFS_TYPE := squashfs
+# ext4 ext2 squashfs
+ROOTFS_TYPE := ext2
```

4. rootfs partition ext2 file system image is automatically packaged and generated, or it can be obtained directly in the following path:

```
buildroot/output/rockchip_rk3308_release/images/rootfs.ext2
```

10.2 Buildroot RK software package introduction

Rockchip platform adds a number of packages support to Buildroot, which are placed in buildroot/package/rockchip directory. The following is a brief introduction to SDK packages:

```
buildroot/package/rockchip
 ├── adbd------------------------adb daemon
 ├── alexaClientSDK------------ alexa audio SDK
 ├── alsa_capture---------------alsa device recording demo program
 ├── cypress_bsa--------------- cypress Bluetooth protocol stack and demo
 ├── DuerClientSDK-------------Baidu audio DuerOS SDK
 ├── gst1-libav-rk---------------gstreamer libav plugin
 ├── gstreamer1-iep------------gstreamer iep plugin
 ├── gstreamer1-rockchip------gstreamer rk plugin
 ├── libcutils---------------------cutils library ported from Android
 ├── libiep------------------------iep library
 ├── libion------------------------ion library
 ├── liblog------------------------log library ported from Android
 ├── LocalPlayer-----------------Local screenless music player
 ├── mdev_mount---------------mdevAutomatic mount script
 ├── mpp-------------------------mpp library
 ├── pcba-------------------------pcba test program
 ├── pm-suspend-api-------------suspend resume multi-process api and demo
program
```

```
├── rkwifibt----------------------wifi, bt firmware and chip configuration
├── rockchip_modules----------- Kernel modules driver
├── rockchip_test---------------- Test script
├── rockchip_utils--------------- Some general programs, such as io command
├── rv1108-firmware------------rv1108 firmware
├── softap------------------------wifi start ap mode
├── softapServer-----------------wifi network configuration service
├── wakeWordAgent-------------alexa wake up words agent
├── wifiAutoSetup----------------wifi smart config network program
└── wpa_supplicant_realtek-----wpa_supplicant realtek version
```

10.3 Preset system applications or data

10.3.1 oem.img instruction

Oem.img: oem partition image file is ext2 file system format by default, used to preset the costomer's application; directly execute package command to package oem partition image.

```
./mkfirmware.sh
```

To generate rockdev/oem.img;

10.3.2 oem image package directory modification

oem partition packages device/rockchip/oem/oem_normal by default. You can modify package directory and oem partition file system by modifying the following configuration files.

The oem path currently has three configurations:

- oem: does not package voice algorithm program, only basic functions
- dueros: Baidu voice SDK, default configuration
- aispeech: AISpeech Voice SDK
- iflytekSDK: iflytek Voice SDK

Configuration file：device/rockchip/rk3308/BoardConfig.mk

```
# Set oem partition type.
# ext2 squashfs
OEM_PARTITION_TYPE=ext2
OEM_PATH=oem_normal
```

10.3.3 userdata.img instruction

Userdata.img: userdata partition image file, is ext2 file system format by default, used to store runtime data, usually is readable and writable; directly execute package command to package userdata partition image.

```
mkfirmware.sh
```

To generate rockdev/userdata.img;

10.4 Add partition configuration

Please refer to \RKDocs\RKTools manuals\Android to add a partition configuration guide V1.00.pdf

10.5 OTA upgrade

10.5.1 Compile dependencies

Rootfs system dependencies configuration：

```
BR2_PACKAGE_RECOVERYSYSTEM=y
```

Recovery compile command：

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

After compiling, misc.img and recovery.img will be generated in rockdev directory.

10.5.2 Upgrade firmware

Refer to chapter 12.3 firmware package tools. Upgrade firmware can support full partition upgrade, or specify partition upgrade. You can modify package-file file and remove partitions that you do not want to upgrade. This will reduce the size of upgrade package (update.img).

Note: recovery.img cannot be packaged for it does not support upgrades now

10.5.3 Upgrade

Under rootfs, run command:

```
recoverySystem ota /xxx/update.img
```

Device will reboot into recovery and upgrade, if upgrade package is in U disk, you can execute the following command:

```
recoverySystem ota /mnt/usb_storage/update.img
```

Available path

```
U disk mount path：/mnt/usb_storage/
sdcard mount path：/mnt/external_sd/
flash mount path：/userdata/
```

10.6 Restore the factory configuration

We can save configuration file that can be read and written in userdata partition. The factory firmware will default some configuration parameters. After users use it for a period of time, it will generate or modify configuration file. Sometimes users need to clear these data, we need to restore the factory configuration.

- Solution: When firmware is packaged, userdata.img is generated. If users request to restore the factory configuration, userdata partition is formatted.
- SDK implementation: Function key RECOVERY + VOLUMEUP trigger to restore the factory configuration, please refer to the code:

```
buildroot/board/rockchip/rk3308/fs-overlay/etc/input-event-daemon.conf
board/rockchip/rk3308/fs-overlay/usr/sbin/factory_reset_cfg
```

10.7 OPTEE develop

This section describes Rockchip TEE security-related firmware description, TEE environment build, CA/TA development test, TA debug method, TA signature method, and considerations.

Detailed development guide can be found in "Rockchip TEE Security SDK Development Guide V1.0.pdf" under docs\ Develop reference documents\SECURITY directory.

10.8 WiFi/WiFi configure network/BT

WiFi configuration, WiFi configure network, BT related functions, detailed development guide can be found in "Rockchip Linux WIFI BT Developers Guide_V1.00.pdf" under docs\Linux reference documents directory.

10.9 Audio player

10.9.1 Audio player Demo

The SDK provides a local music player demo, which contains API interfaces for initializing, playing, pausing, pausing recovery, and stopping playback. It can be used as a reference for music playback.

For detailed Demo source location and instructions, see”RK3308 Local Player Demo instruction.pdf” in the docs\SoC platform related\RK3308\ directory.

10.10 LED control

The SDK provides an LED control demo program that includes instructions for LED device initialization, individual mode switching, and de-initialization.

This LED control DemoAPI supports the following modes:

- Mode 1 all are on
- Mode 2 all are off
- Mode 3 rotation
- Mode 4 breathing light
- Mode 5 lights fixed number of lights

For detailed Demo source location and instructions, see “ RK3308 LED Control Demo Documentation.pdf” in the docs\SoC platform related\RK3308\ directory.

10.11 Interconnect function

10.11.1 DLNA

The full name of DLNA is DIGITAL LIVING NETWORK ALLIANCE. Designed to address the interconnection of wireless and wired networks, including personal PCs, consumer electronics, and mobile devices, enabling unrestricted sharing and growth of digital media and content services.

For details on the configuration, usage, and development of DLNA, see “Rockchip DLNA Development Guide_V1.00.pdf” in the docs\Linux reference documents\

directory.

10.12 Suspend and resume

10.12.1 Overview

●Suspend: System freezes process, and then suspends power of device to stop working and enters low power mode; suspend command:

```
echo mem > /sys/power/state
```

● Resume: Resume from suspend mode to normal operating mode. The power button on EVB has resume function. When power button is pressed, system will resume.

Some applications need to do some processing before suspend, shut down resources, save state, and then recover after resuming. This requires system to provide a suspend and resume interfaces for application develop.

10.12.2 Single application mode

Single application means that only one application in system needs to pay attention to suspend and resume process. In this case, the application can directly take over suspend and resume. When entering suspend, application can directly call suspend interface to enter suspend. After resuming by events trigger, the application continues to execute code. Similar to the following logic:

```
int main (int argc, char *argv[])
{
    printf("running...\n");
    printf("enter sleep mode...\n");
    system("echo mem > /sys/power/state");
    printf("app resume running...");
    return 0;
}
```

10.12.3 Multi-application mode

Multi-application means that there are multiple applications in system that need to monitor suspend and resume actions; before suspend, these applications need to do suspend processing, and resume also need to do resume actions. We introduce pm-utils to manage suspend and resume actions. Pm-utils is a lightweight set of scripts that is equivalent to suspend and resume HAL.

Pm-utils provides suspend command pm-suspend. After pm-suspend is executed, the scripts in /usr/lib/pm-utils/sleep.d/ directory are executed in sequence before suspend. We added a script to synchronously call application's suspend interface in dbus-send way. After that, system goes to suspend; similarly, after system resume, the script in this directory is executed in turn, and resume interface of application is called in the way of dbus-send with resume irq as a parameter.

We package a suspend and resume api based on dbus interface.

Code directory：package/rockchip/pm-suspend-api

Configuration option：BR2_PACKAGE_PM_SUSPEND_API=y

API interface：

```
  SUSPEND_API int request_system_suspend(void);

  SUSPEND_API int register_suspend_listener(char *bus_name, CALLBACK
suspend_callback, CALLBACK resume_callback);
```

Demo application 1: Every 10 seconds of operation, application for system going to suspend.

```
#include <stdio.h>
#include "dbus_suspend_api.h"

int main()
{
    printf("hello sleep_test\n");
    while (1) {
        sleep(10);
        printf("time elapse 10 seconds,request sleep\n");
        request_system_suspend();
    }
    return 0;
}
```

Demo application 2: Register suspend and resume interface, print status before suspend, print status after resume:

```
#include <stdio.h>
#include "dbus_suspend_api.h"

int suspend(void *data)
{
    printf("app_test1 suspending\n");
    usleep(100*1000);
    //to do...
    printf("app_test1 suspend succeed\n");
    return 0;
}

int resume(void *data)
{
    printf("app_test1 resumeing \n");
    usleep(100*1000);
    //to do...
    printf("app_test1 resume succeed\n");
    return 0;
}

int main()
{
      printf("hello world\n");
      register_suspend_listener("com.rockchip.suspend.app_test1", suspend,
resume);
      while(1)
          sleep(2);
      return 0;
  }
```

you need to install dbus configuration file to dbus directory /etc/dbus-1/system.d/

```
<!DOCTYPE busconfig PUBLIC
 "-//freedesktop//DTD D-BUS Bus Configuration 1.0//EN"
 "http://www.freedesktop.org/standards/dbus/1.0/busconfig.dtd">
<busconfig>
    <policy context="default">
        <allow own="com.rockchip.suspend.app_test1"/>
        <allow send_destination="com.rockchip.suspend.app_test1"/>
        <allow own="com.rockchip.suspend.app_test2"/>
        <allow send_destination="com.rockchip.suspend.app_test2"/>
    </policy>
</busconfig>
```

Specific demo please refer to the code: package/rockchip/pm-suspend-api/src/。

10.13 Speech recognition SDK

At present, Buildroot sdk already supports three speech suites: Dueros, AI Speech., and Iflytek.

10.13.1 Dueros

DuerOS configuration:
Buildroot: BR2_PACKAGE_DUERCLIENTSDK=y
Device/rockchip/rk3308/BoardConfig.mk configuration OEM_PATH=dueros
For the configuration, usage and development instructions of Dueros, see "RK3308 DuerOS Instructions for Use.pdf" in the docs\SoC platform related\RK3308\ directory.

10.13.2 AI Speech

DuerOS configuration:
device/rockchip/rk3308/BoardConfig.mk configure OEM_PATH=aispeech

10.13.3 IFLYTEK

IFLYTEK　configuration:
Buildroot：BR2_PACKAGE_IFLYTEKSDK=y
device/rockchip/rk3308/BoardConfig.mk configure OEM_PATH=iflytekSDK

10.13.3.1 VAD+ Iflytek recongnition Demo

RK3308 VAD+Iflytek recognition demo program integrates voice detection (VAD, Voice Activity Detection) and Iflytek front-end processing identification module, which can realize VAD wake-up and send it to IFLYTEK to recognize and wake up.

For detailed demo source location and instructions, see "RK3308 VAD+ IFLYTEK recognition Demo Documentation.pdf "in the docs\ SoC platform related\RK3308\ directory.

Chapter 11. System Debug

This chapter focuses on some debug tools and methods during SDK develop, and it will constantly supplement and complete, aim to helping developers quickly get start with basic system debugging and make correct analysis.

11.1 ADB tool

11.1.1 Overview

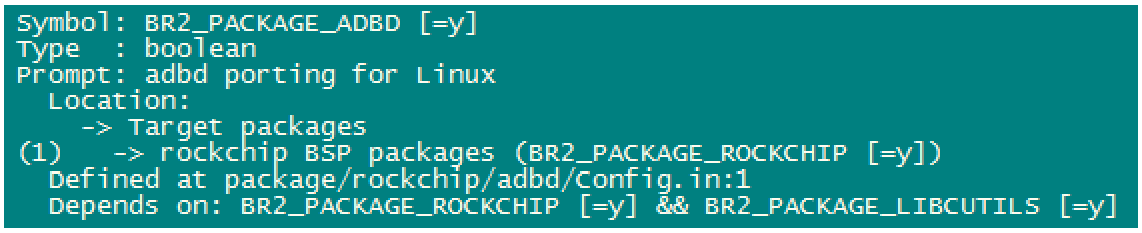
ADB（Android Debug Bridge）is a tool in Android SDK that allows you to operate and manage Android emulators or real Android devices. The main functions are:

- Run shell of device (command line)
- Manage port mapping for emulators or devices
- Upload/download files between computer and device
- Install local apk software to emulator or hardware device

ADB is a "client-server" program, where the client is mainly a PC, and the server is a physical machine or virtual machine of an Android device. Depending on how PC is connected to Box machine, ADB can be divided into two type:

- Network ADB: Host is connected to hardware device through a wired/wireless network (same LAN)
- USB ADB: Host is connected to hardware device via a USB cable

11.1.2 Buildroot configuration


```
Symbol: BR2_PACKAGE_ADBD [=y]
Type  : boolean
Prompt: adbd porting for Linux
  Location:
    -> Target packages
(1)    -> rockchip BSP packages (BR2_PACKAGE_ROCKCHIP [=y])
  Defined at package/rockchip/adbd/Config.in:1
  Depends on: BR2_PACKAGE_ROCKCHIP [=y] && BR2_PACKAGE_LIBCUTILS [=y]
```
Figure 11-1 adb buildroot configuration

11.1.3 USB ADB instruction

There are following restrictions when using USB ADB

- Only supports USB OTG port
- Does not support multiple clients at the same time (such as cmd window, eclipse, etc.)
- Only supports one device connected to host, and does not support connecting multiple devices.

Connection steps are as follows:

1. Device is already running Android system, Settings ->Developer Options -> Connected to Computer and then open to turn on usb debug switch (If it is a Linux system, the configuration is turned on by default)

2. PC host is connected to USB OTG port only through USB cable, and then the computer is connected to device through the following commands.

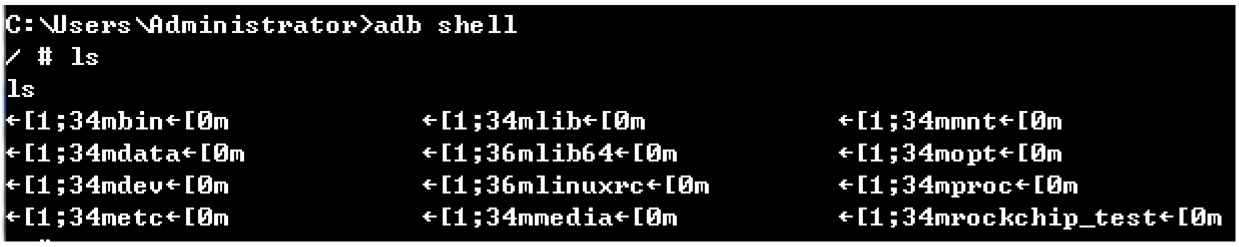
```
Adb shell
```

3. Test whether connection is successful. Run “adb devices” command. If serial

number of the device is displayed, connection is successful.

11.1.4 ADB window garbled display

If you encounter garbled situation shown in the figure below, mainly because Linux system outputs color characters which the tool cannot be parsed normally.



```
C:\Users\Administrator>adb shell
/ # ls
ls
←[1;34mbin←[0m            ←[1;34mlib←[0m            ←[1;34mmnt←[0m
←[1;34mdata←[0m           ←[1;36mlib64←[0m          ←[1;34mopt←[0m
←[1;34mdev←[0m            ←[1;36mlinuxrc←[0m        ←[1;34mproc←[0m
←[1;34metc←[0m            ←[1;34mmedia←[0m          ←[1;34mrockchip_test←[0m
```

Figure 11-2 adb buildroot configuration

There are two solutions:

- Use puTTY tool for adb debugging.
- Enter the following command to remove color display.

```
alias ls='ls --color=never'
```

11.1.5 Details of ADB common commands

（1）**View device status**

View device or emulator connected to your computer:

```
adb devices
```

It will return the serial number or IP and port number (Port) of the device connected to host.

（2）**Enter shell of device and emulator**

Enter shell environment of the device or emulator:

```
adb shell
```

（3）**Upload files from computer to device**

Use push command to upload any file or folder on your local computer to your device. The local path generally refers to local computer; remote path generally refers to the single board device connected to adb.

```
adb push < local path > < remote path >
```

For example：

```
adb push "F:\WishTV\WishTV.apk"  "/system/app"
```

Example note: Upload local "WishTV.apk" file to "/system/app" directory of device system.

（4）**Download files from device to computer**

Pull command is to download files or folders from device to local computer.

```
adb pull < remote path > < local path >
```

For example：

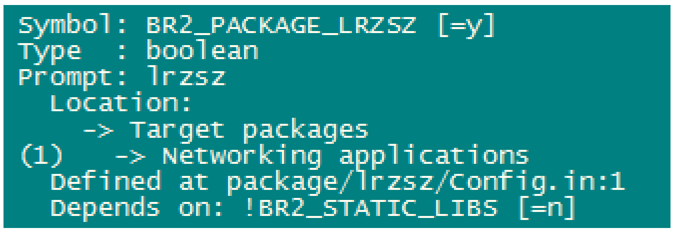
```
adb pull  /system/app/Contacts.apk F:\
```

Example note: Download files or folders in /system/app directory on device system to local F:\ directory.

11.2 Lrzsz tool

Lrzsz is a file transfer tool under Linux. The implementation principle is to transfer files through Xmodem / Ymodem / Zmodem protocol. Lrzsz can work under tools of Shell interface that support these three protocols, such as SecureCRT, puTTY, etc.

11.2.1 Buildroot configuration



```
Symbol: BR2_PACKAGE_LRZSZ [=y]
Type  : boolean
Prompt: lrzsz
  Location:
    -> Target packages
(1)    -> Networking applications
  Defined at package/lrzsz/Config.in:1
  Depends on: !BR2_STATIC_LIBS [=n]
```

Figure 11-3 lrzsz buildroot configuration

11.2.2 Instructions

sz command sends files to local::

```
sz filename
```

rz command locally uploads files to server:

```
rz
```

After executing this command, select the file to be uploaded in pop-up box.

Note: Open SecureCRT software -> Options -> session options -> X/Y/Zmodem to set directory for uploading and downloading.

11.3 Procrank tool

Procrank is a debugging tool that outputs a memory snapshot of a process to effectively observe memory usage of process.

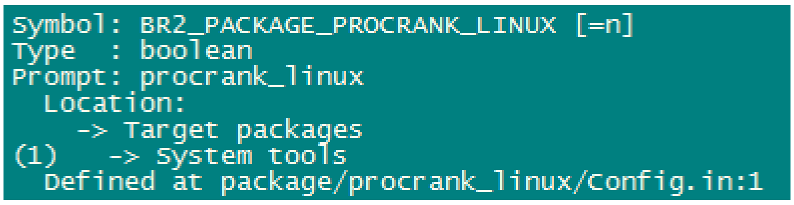
Includes the following memory information:

- VSS: Virtual Set Size, virtual memory consumption (including memory occupied by shared libraries)
- RSS: Resident Set Size, actual physical memory size (including memory occupied by shared library)
- PSS: Proportional Set Size, actual physical memory size used (proportional allocation of shared memory)
- USS: Unique Set Size, physical memory size occupied by processes alone (excluding memory occupied by shared library)

Note:

- USS size represents memory size that is only used by process, and process is completely recovered after it is killed.
- VSS/RSS contains memory used by shared library and has no reference value for viewing memory state of a single process;
- PSS is occupation of a shared memory area by a single process after partitioning shared memory.

11.3.1 Buildroot configuration



```
Symbol: BR2_PACKAGE_PROCRANK_LINUX [=n]
Type  : boolean
Prompt: procrank_linux
  Location:
    -> Target packages
(1)   -> System tools
  Defined at package/procrank_linux/Config.in:1
```

Figure 11-4 procrank buildroot configuration

11.3.2 procrank

Before executing procrank, you need to let terminal get root permissions:

```
su
```

Command format:

```
procrank [ -W ] [ -v | -r | -p | -u | -h ]
```

Common instructions:

- -v: Sort by VSS
- -r: Sort by RSS
- -p: Sort by PSS
- -u: Sort by USS
- -R: Convert to incremental [decrement] order
- -w: Show statistics count of working set only
- -W: Reset statistics count of working set
- -h: Help

Example:

- Output memory snapshot:

```
procrank
```

- Output memory snapshots in descending order of VSS

```
procrank –v
```

The default procrank output is sorted by PSS.

11.3.3 Search specified content information

View memory usage status of specified process. The command format is as follows:

```
procrank | grep [cmdline | PID]
```

cmdline represents name of application that needs to be searched, and PID represents application process that needs to be searched.

Output memory usage status of systemUI process:

```
procrank | grep "com.android.systemui"
```

Or :

```
procrank  | grep 3396
```

11.3.4 Track process memory status

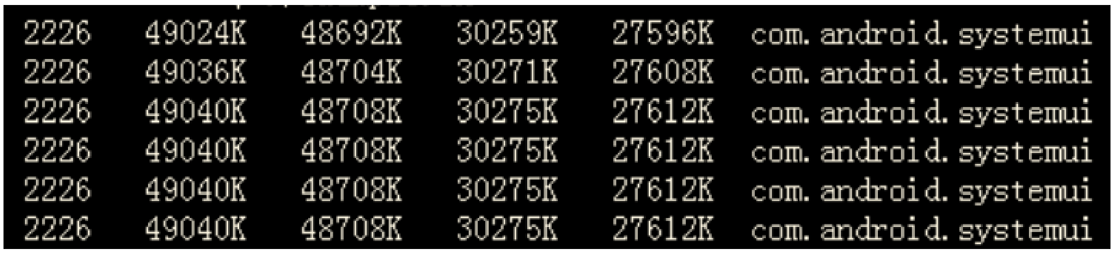
By tracking memory usage status, it analyzes whether there is a memory leak situation in process. Use scripting method to continuously output memory snapshot of a process. By comparing USS segment, you can find out whether the process has a memory leak.

Example: Output application memory usage status of a process named com.android.systemui to see if there is a leak:

1、Writing the script test.sh

```
#!/bin/bash
while true;do
adb shell procrank | grep “com.android.systemui”
sleep 1
done
```

2、After connecting to device via adb tool, run this script: ./test.sh. As show in the figure below:



```
2226   49024K   48692K   30259K   27596K   com.android.systemui
2226   49036K   48704K   30271K   27608K   com.android.systemui
2226   49040K   48708K   30275K   27612K   com.android.systemui
2226   49040K   48708K   30275K   27612K   com.android.systemui
2226   49040K   48708K   30275K   27612K   com.android.systemui
2226   49040K   48708K   30275K   27612K   com.android.systemui
```

Figure 11-5 Track process memory status

11.4 FIQ

FIQ debugger is a system debug method integrated into kernel.

In general, serial port is in normal console mode. Input switch command "f+i+q" under SecureCRT or puTTY, then serial port will switch to FIQ debugger mode.

Because FIQ is a non-maskable interrupt, this debug method is suitable for debug the situation when CPU is hanged. You can use FIQ debugger to print out fault scene of CPU, the common command is sysrq.

Fiq debugger related commands:

```
debug> help
FIQ Debugger commands:
 pc            PC status
 regs          Register dump
 allregs       Extended Register dump
 bt            Stack trace
 reboot [<c>]  Reboot with command <c>
 reset [<c>]   Hard reset with command <c>
 irqs          Interupt status
 sleep         Allow sleep while in FIQ
 nosleep       Disable sleep while in FIQ
 console       Switch terminal to console
 cpu           Current CPU
 cpu <number>  Switch to CPU<number>
 ps            Process list
 sysrq         sysrq options
 sysrq <param> Execute sysrq with <param>
```

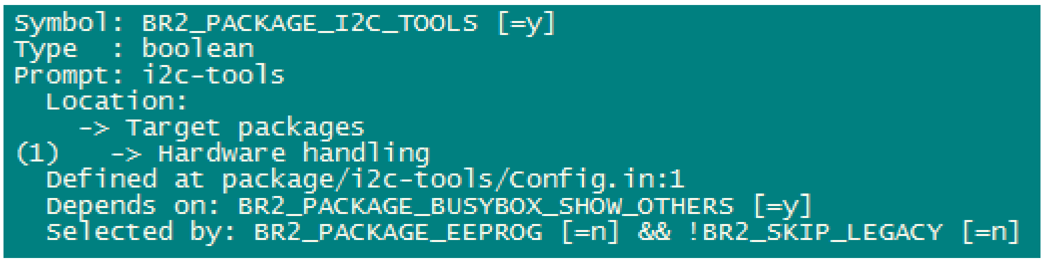
11.5 Last_log

```
cat /sys/fs/pstore/console-ramoops-0
```

Print out device information before the last system reset. If there is an abnormal copying or abnormal power failure, you can use this command to print logs of the last system running status.

11.6 i2c-tools

11.6.1 Buildroot configuration



```
Symbol: BR2_PACKAGE_I2C_TOOLS [=y]
Type  : boolean
Prompt: i2c-tools
  Location:
    -> Target packages
(1)   -> Hardware handling
  Defined at package/i2c-tools/Config.in:1
  Depends on: BR2_PACKAGE_BUSYBOX_SHOW_OTHERS [=y]
  Selected by: BR2_PACKAGE_EEPROG [=n] && !BR2_SKIP_LEGACY [=n]
```

Figure 11-6 i2c-tool buildroot configuration

11.6.2 Instruction

After Buildroot configures i2c-tools, rootfs, it will integrate the following four tools:

- i2cdetect
- i2cdump
- i2cget
- i2cset

i2cdetect lists I2C bus:

```
# i2cdetect -l
i2c-0   i2c         imx-i2c                       I2C adapter
i2c-1   i2c         imx-i2c                       I2C adapter
i2c-2   i2c         imx-i2c                       I2C adapter
```

List I2C bus i2c-1 all devices connected above

```
# i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- UU -- -- -- -- -- -- --
10: -- UU -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- 3a -- -- -- -- --
40: -- -- -- -- -- -- -- -- UU -- -- -- -- -- -- --
50: UU -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

The location of I2C device is found to be UU or a value indicating address of device. UU indicates that device is used in driver.

I2cdumpdump I2C device register value in large quantities

```
# i2cdump -y -f 1 0x3a
No size specified (using byte-data access)
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: eb 00 7f 05 3d 00 00 00 08 06 00 00 00 00 00 00    ?.??=...??......
10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
20: 01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10    ?#Eg????????vT2?
30: f0 e1 d2 c3 00 00 00 00 00 00 00 00 00 00 00 00    ????............
40: 80 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00    ?.?.............
50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
f0: 00 00 00 00 00 00 00 00 00 00 00 30 00 00 00 00    ...........0....
```

I2cget reads value of a register of an I2C device:

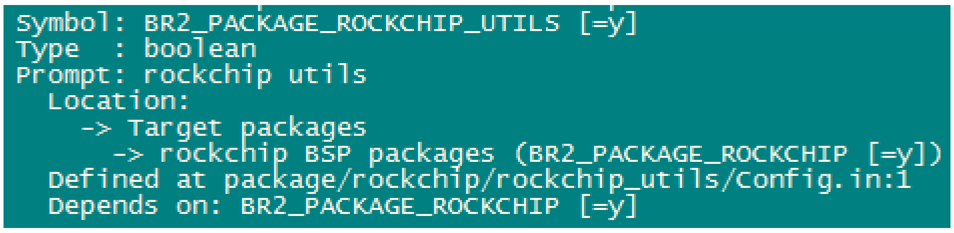
```
# i2cget -y -f 1 0x3a 0x02
0x7f
```

I2cset sets value of a register of an I2C device:

```
[cpp] view plain copy
# i2cset -y -f 1 0x3a 0x02 0x05
```

11.7 io

11.7.1 Buildroot configuration



```
Symbol: BR2_PACKAGE_ROCKCHIP_UTILS [=y]
Type  : boolean
Prompt: rockchip utils
  Location:
    -> Target packages
      -> rockchip BSP packages (BR2_PACKAGE_ROCKCHIP [=y])
  Defined at package/rockchip/rockchip_utils/Config.in:1
  Depends on: BR2_PACKAGE_ROCKCHIP [=y]
```

Figure 11-7 io buildroot configuration

11.7.2 Instruction

After Buildroot configured io, rootfs will integrate io tools. io command can dynamically read or configure value of register. The instructions are as follows:

```
io 0x1000                Reads one byte from 0x1000
io 0x1000 0x12           Writes 0x12 to location 0x1000
```

```
  io -2 -l 8 0x1000            Reads 8 words from 0x1000
  io -r -f dmp -l 100 200      Reads 100 bytes from addr 200 to file
  io -w -f img 0x10000         Writes the whole of file to memory
```

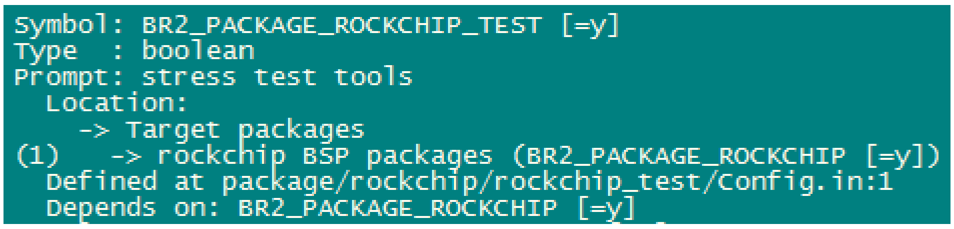
Chapter 12. Common Tools Instruction

This chapter briefly describes some of development and production tools that come with SDK, so that developers can understand usage of RK platform tools. For detailed tool usage instructions, please see documentations in tools directory and tool documentations in docs\ RKTools manuals directory.

12.1 Stress test tool –rockchip_test

SDK integrates stress test tool under Linux platform, stress testing of various functions of device to be tested to ensure stability of the whole system.

12.1.1 Buildroot configuration



```
Symbol: BR2_PACKAGE_ROCKCHIP_TEST [=y]
Type  : boolean
Prompt: stress test tools
  Location:
    -> Target packages
(1)   -> rockchip BSP packages (BR2_PACKAGE_ROCKCHIP [=y])
  Defined at package/rockchip/rockchip_test/Config.in:1
  Depends on: BR2_PACKAGE_ROCKCHIP [=y]
```

Figure 12-1 rockchip_test configuration

12.1.2 Instruction

After Buildroot configures rockchip_test, rootfs integrates all scripts and dependencies used in rockchip_test stress test.

Execute rockchip_test/rockchip_test.sh script for testing.

Module related

- Bluetooth stress test: Includes Bluetooth on and off
- Wifi stress test: Including Wifi on and off, (ping test and iperf test to be added)

Non-module related

- Suspend and resume stress test.
- Restart stress test
- Restore the factory settings stress test.
- Arm frequency conversion test
- DDR stress test
- Flash read and write stress test

12.2 PCBA test tools

PCBA testing is used to help quickly identify product features during mass production, that is, to focus on FCT (Functional Test) testing to increase productivity. PCBA test tool is developed under Window system and only supports running under Window system. With device-side test program, it can verify the integrity and quality of functions or devices that need to be focused on.

Current test items include: SD card test, wifi test, Bluetooth test, DDR test, ring microphones test, USB host test, led light test, playback test, recording test, button test, PDM Mic test, Audio Line in test, SPDIF For IN/OUT tests, etc., it will add extended test items later.

Automatic test items do not require manual intervention. After test is over, test result will be reported directly and pass or fail will be displayed. Manual test items need to manually judge whether test items are correctly completed and gives a judgment (pass or fail).

For instructions about how to use the tool, see Rockchip_PCBA Test Development Guide_1.01.pdf in docs\Linux reference documents\ directory.

12.3 Firmware package tools

Firmware package tool can package each fragmented image file into a completed update.img for mass production and upgrade.

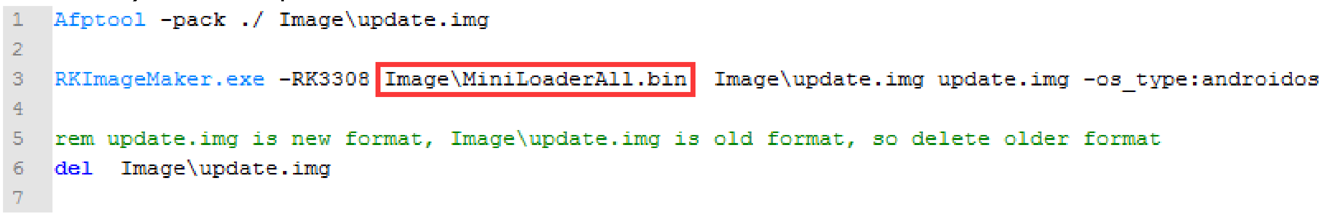
12.3.1 Package under windows

Under windows, package tool is stored in tools\windows\AndroidTool\rockdev. The packaging steps are as follows:

1) Open rockdev directory and edit package-file

Configure according to package-file, "*.img" image in package-file is placed under Image directory, copy the image that needs to be placed in Image directory. And pay attention to whether image name is accurate when configuration. Note that bootloader option should be modified according to the name of loader generated by itself.

2) Edit mkupdate.bat



```
Afptool -pack ./ Image\update.img

RKImageMaker.exe -RK3308 Image\MiniLoaderAll.bin  Image\update.img update.img -os_type:androidos

rem update.img is new format, Image\update.img is old format, so delete older format
del  Image\update.img

```

Figure 12-2 update.img package script

Need to modify loader name to actual stored loader name.

3) Click mkupdate.bat to run, after running, it will generate an update.img in current directory.

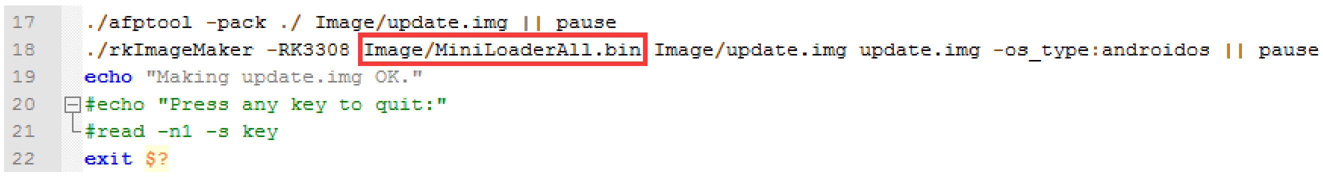
12.3.2 Package under Linux

Under Linux，package tool is stored in w:\RK3308\tools\linux\Linux_Pack_Firmware\rockdev\, package steps are as follows:

1) Open rockdev directory and edit package-file

Configure according to package-file, "*.img" image in package-file is placed under Image directory, copy the image that needs to be placed in Image directory. And pay attention to whether image name is accurate when configuration. Note that bootloader option should be modified according to the name of loader generated by itself.

2）Edit mkupdate.sh



```
./afptool -pack ./ Image/update.img || pause
./rkImageMaker -RK3308 Image/MiniLoaderAll.bin Image/update.img update.img -os_type:androidos || pause
echo "Making update.img OK."
#echo "Press any key to quit:"
#read -n1 -s key
exit $?
```

Figure 12-3 update.img package script

Loader name should be modified to actual loader name.

3）Execute the following command in rockdev directory. After running, an update.img will be generated in current directory.

```
./mkupdate.sh
```

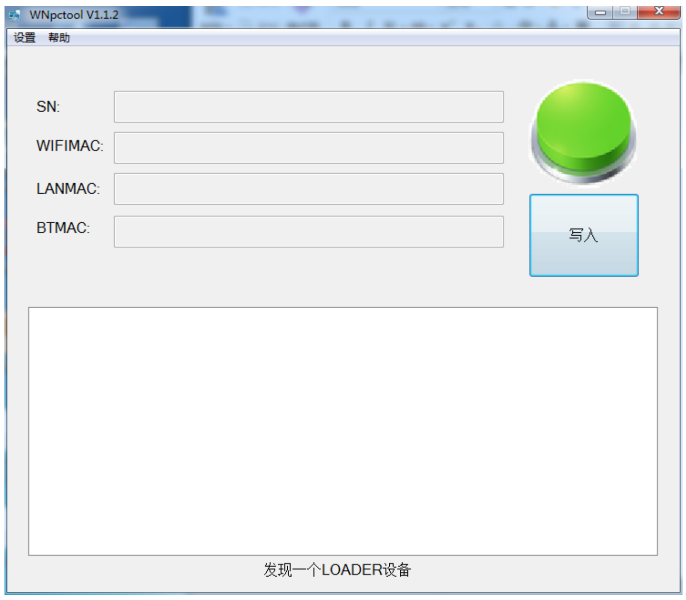
12.4 Firmware signature tool

Firmware signature tool to be updated.

12.5 Serial Number/Mac/Vendor Information writing - WNpctool Tools

Serial number/Mac/vendor information writing all use WNpctool tool. The basic usage of the tool is described below.

12.5.1 WNpctool Write steps



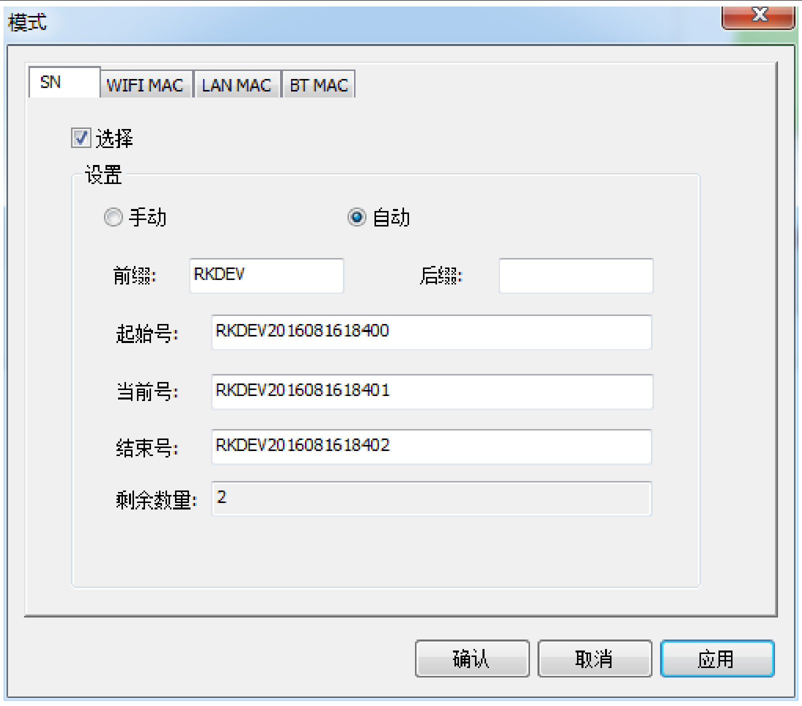
发现一个LOADER设备

Figure 12-4 WNpctool tool

1) Enter loader mode.

2) Click Settings button, there will be a drop-down box button, click "Read" button to switch whether it is a write or read function. Switch to write function.

3) Click mode, the following window appears, used to set SN/WIFI/LAN/BT



☑ 选择

设置

手动 [✓ 自动]

前缀:	RKDEV	后缀:	
起始号:	RKDEV2016081618400		
当前号:	RKDEV2016081618401		
结束号:	RKDEV2016081618402		
剩余数量:	2		

Figure 12-5 WNpctool tool mode setting

4) After setting, click Apply button to close the window, return to the main window, and click write button.

12.5.2 WNpctool Read steps

1) Enter loader mode.

2) Click Settings button, there will be a drop-down box button, click "Read" button to switch whether it is a write or read function. Switch to read function.

3) Click read button

12.6 EQ_DRC tool

EQ_DRC tool is RK3308 sound effects real-time tuning tool, which can be used to adjust various audio parameters of board end in real time.

Instructions for using the tool can be found in docs\ RKTools manuals\ directory.

"RK3308_EQ_DRC_TOOL_V1.1_20180510.pdf", through examples to introduce use of EQ_DRC tuning tools and precautions.

Chapter 13. Appendix

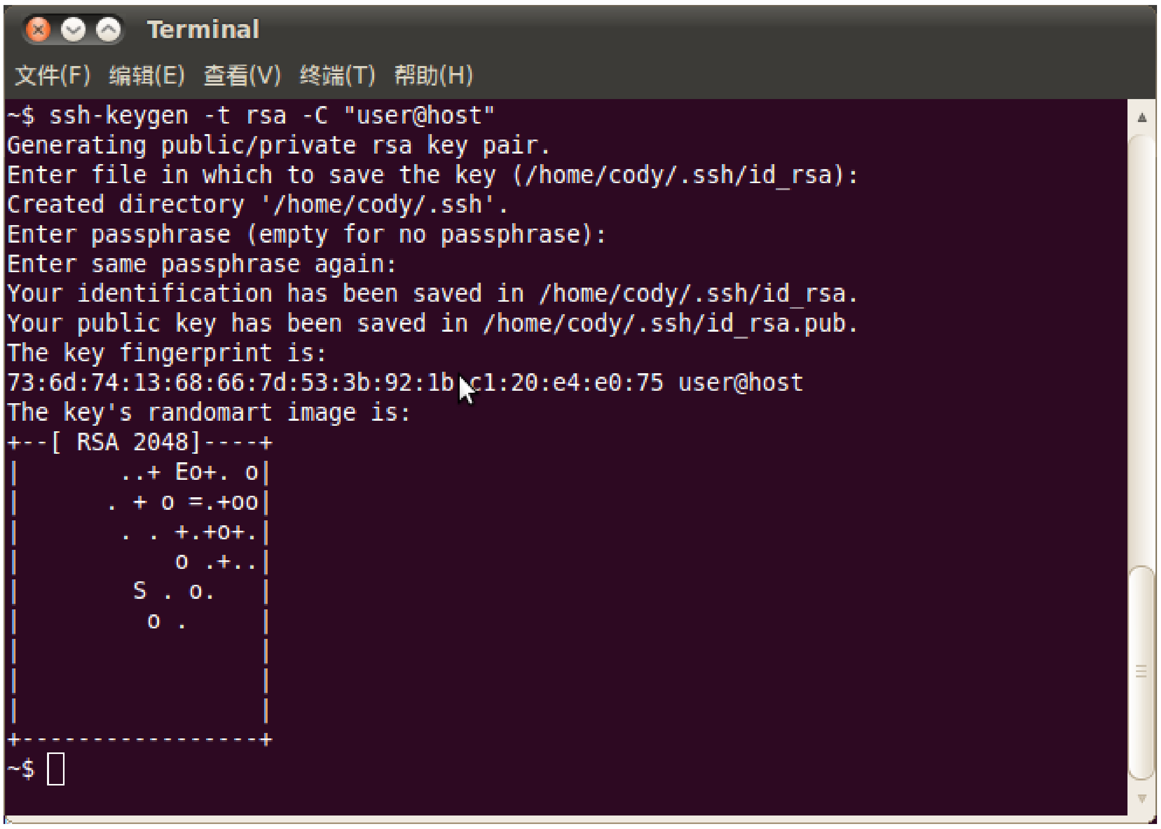
13.1 SSH Public Key Operation Instruction

13.1.1 SSH Public Key Generation

Use the following command to generate:

```
ssh-keygen -t rsa -C "user@host"
```

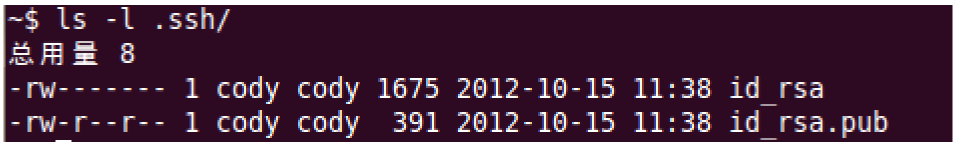
Please replace user@host with your email address。



```
~$ 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:c1:20:e4:e0:75 user@host
The key's randomart image is:
+--[ RSA 2048]----+
|        ..+ Eo+. o|
|       . + o =.+oo|
|        . . +.+o+.|
|            o .+..|
|         S . o.   |
|          o .     |
|                  |
|                  |
|                  |
+-----------------+
~$ 
```

Figure 13-1 Public Key Generation

Completing Command will generate public key files in your directory.



```
~$ 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
```

Figure 13-2 .ssh directory file

Please keep the private key file id_rsa and password generated, and email id_rsa.pub to fae@rock-chips.com to obtain SDK code download permission.

13.1.2 Use key-chain to manage keys

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

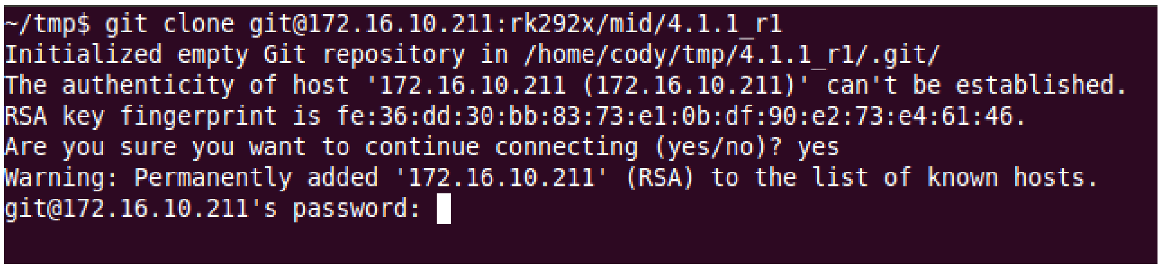
id_rsa is private key file name among them.。

After the above configuration, log in to console again and it will prompt to enter password. Just enter password used to generate the key. If there is no password, you can not enter it.

In addition, try not to use sudo or root users unless you know how to deal with them, which will lead to permission and key management confusion

13.1.3 Multiple machines use the same SSH public key

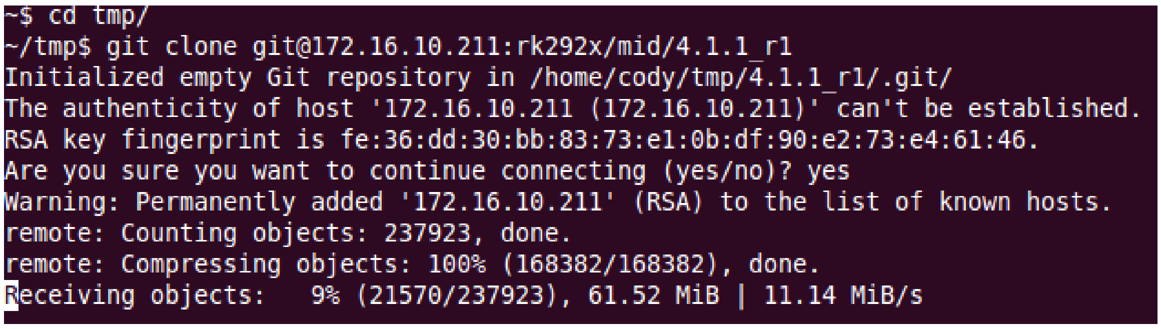
Use on different machines, you can copy ssh private key file id_rsa to "~/.ssh/id_rsa" of machines you want to use.The following prompt will appear when using a wrong private key, please be careful to replace it with the correct private key.



```
~/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:
```

Figure 13-3 wrong private key error

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
```

Figure 13-4 correct clone code

Adding ssh private key may result in following error.

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

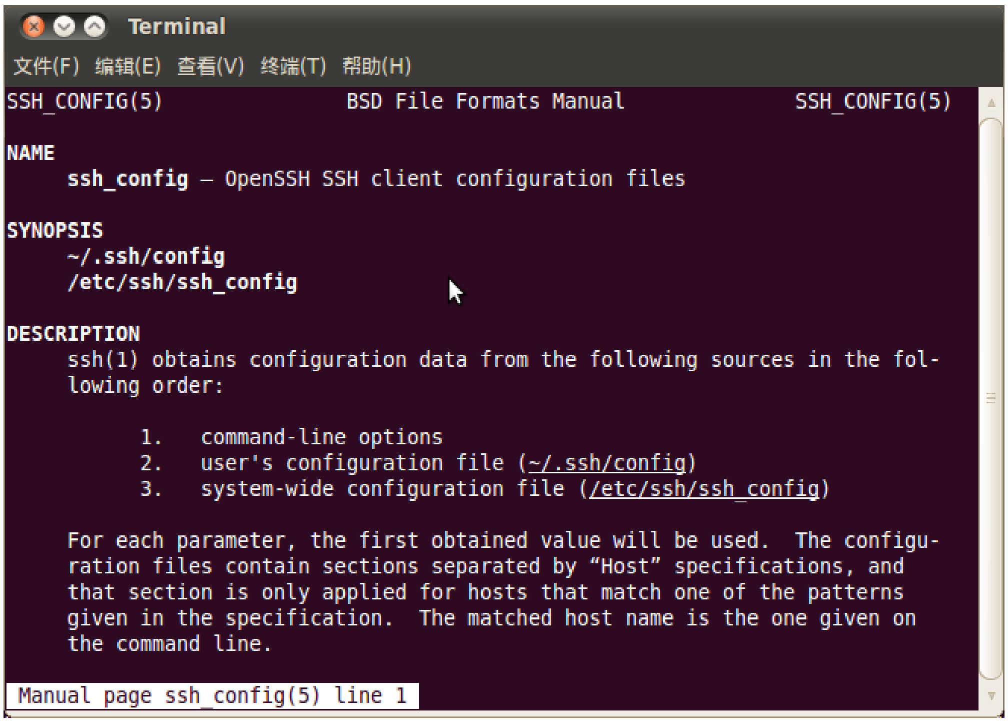
Enter the following command in console to solve

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

13.1.4 One machine switches different SSH public keys

You can configure SSH by referring to ssh_config documentation

```
~$ man ssh_config
```



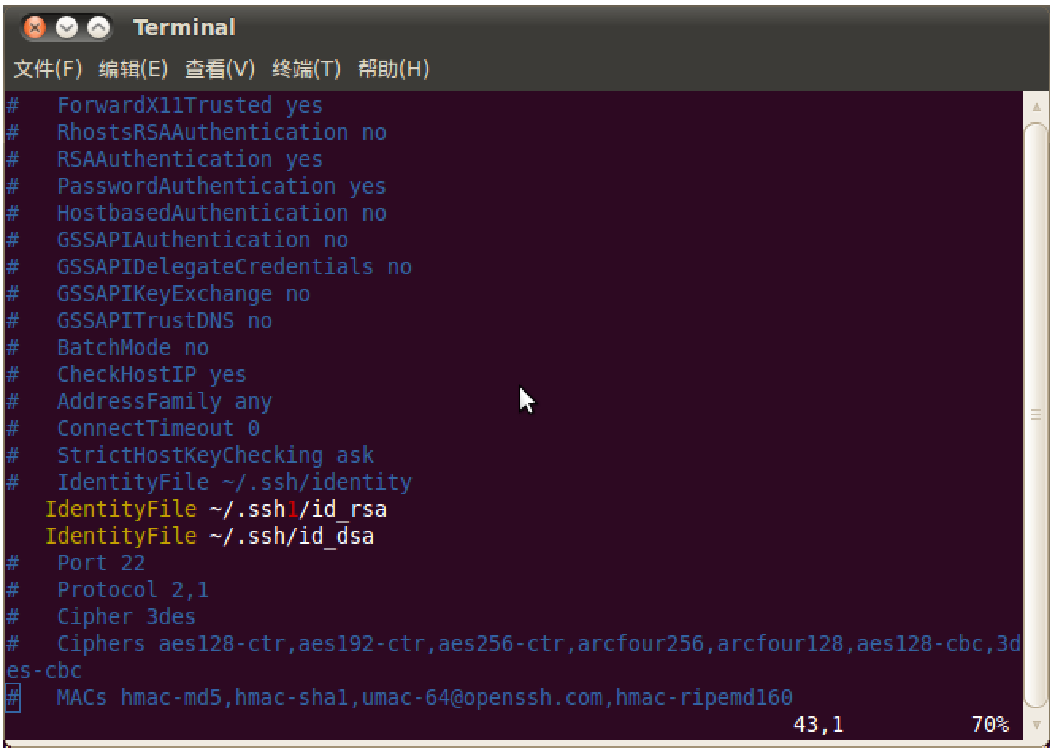
```
SSH_CONFIG(5)             BSD File Formats Manual            SSH_CONFIG(5)

NAME
     ssh_config — OpenSSH SSH client configuration files

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

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

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

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

 Manual page ssh_config(5) line 1
```

Figure 13-5 ssh_config configuration instruction

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

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

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



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

Figure 13-6 .ssh/config configuration modification

13.1.5 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.

Please keep your private key file in a safe place. Do not grant second authorization to third parties.

13.1.6 Git Access Application Instruction

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