Rockchip Linux Software Developer Guide

ID: RK-KF-YF-902

Release Version: V2.0.1

Release Date: 2023-12-05

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

DISCLAIMER

Trademark Statement

Rockchip Electronics Co., Ltd.

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

Website: www.rock-chips.com

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

Customer service Fax: +86-591-83951833

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

Preface

Overview

This document is a guide for Rockchip Buildroot/Debian/Yocto Linux system software and is designed to help software development engineers and technical support engineers get started with the development and debugging of the Rockchip Linux platform faster.

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers

Software development engineers

Revision History

Date	Author	Version	Change Description
2021-04-10	Caesar Wang	V1.0.0	Initial version
2021-05-20	Caesar Wang	V1.1.0	Added support for rk3399, rk3288,rk3326/px30
2021-09-30	Caesar Wang	V1.2.0	Updated support for Linux4.4 and Linux4.19
2022-01-15	Caesar Wang	V1.3.0	Added support for RK3588 Linux5.10 Added support for RK3358 Linux4.19 Updated SDK version information
2022-04-14	Caesar Wang	V1.4.0	Added support for RK3326S Updated RK3588 Updated FAQ
2022-05-20	Caesar Wang	V1.4.1	Updated chip support list and 2022 roadmap
2022-06-20	Caesar Wang	V1.4.2	Update SDK version and support
2022-09-20	Caesar Wang	V1.5.0	Updated support for Linux 5.10
2022-11-20	Caesar Wang	V1.6.0	Update the support status and roadmap of each chip system Secure boot update instructions Update FAQ
2023-04-20	Caesar Wang	V1.7.0	Update the support status and roadmap of each chip system Update the latest SDK directory structure Add support for RK3562 The document is split into multiple chapters
2023-05-20	Caesar Wang	V1.8.0	Corrected some mistake Version update Added SDK version description Updated the SDK development environment setup chapter
2023-06-20	Caesar Wang	V1.9.0	Updated documentation Updated development environment setup chapter
2023-07-20	Caesar Wang	V1.9.1	Updated RK3566/RK3568/RK3399 Linux4.19 SDK version
2023-09-20	Caesar Wang	V2.0.0	Update the contents of each chapter
2023-12-05	Ruby Zhang	V2.0.1	Fix some description

RK Linux SDK Supported System List

Linux5.10 SDK

Chipset	Buildroot Version	DebianVersion	Yocto Version	Kernel Version	SDK Version	TAG Version
RK3588	2021.11	11	4.0	5.10	V1.3.0_20230920	linux-5.10-gen-rkr6
RK3562	2021.11	11	4.0	5.10	V1.0.0_20230620	linux-5.10-stan-rkr1
RK3566	2021.11	11	4.0	5.10	V1.3.0_20230920	linux-5.10-gen-rkr6
RK3568	2021.11	11	4.0	5.10	V1.3.0_20230920	linux-5.10-gen-rkr6
RK3399	2021.11	11	4.0	5.10	V1.3.0_20230920	linux-5.10-gen-rkr6
RK3358	2021.11	N/A	N/A	5.10	V1.2.0_20230920	linux-5.10-gen-rkr6
RK3326	2021.11	N/A	N/A	5.10	V1.2.0_20230920	linux-5.10-gen-rkr6
PX30	2021.11	11	4.0	5.10	V1.3.0_20230920	linux-5.10-gen-rkr6
RK3308	2021.11	N/A	N/A	5.10	V1.2.0_20230920	linux-5.10-gen-rkr6
RK312X	2021.11	N/A	N/A	5.10	V1.2.0_20230920	linux-5.10-gen-rkr6

Chipset	Buildroot Version	DebianVersion	Yocto Version	Kernel Version	SDK Version	TAG Version
RK3036	2021.11	N/A	N/A	5.10	V1.2.0_20230920	linux-5.10-gen-rkr6

Linux4.19 SDK

Chipset	Buildroot Version	DebianVersion	Yocto Version	Kernel Version	SDK Version	TAG Version
RK3566	2021.11	11	4.0	4.19	V1.4.0_20230720	linux-4.19-gen-rkr4
RK3568	2021.11	11	4.0	4.19	V1.4.0_20230720	linux-4.19-gen-rkr4
RK3399	2021.11	11	4.0	4.19	V1.3.0_20230720	linux-4.19-gen-rkr4
RK3326	2018.02-rc3	N/A	N/A	4.19	V1.2.2_20221220	linux-4.19-gen-rkr3.2
RK3358	2018.02-rc3	N/A	N/A	4.19	V1.1.1_20220915	linux-4.19-M-rkr1
RK3288	2018.02-rc3	10	3.4	4.19	V1.2.2_20221220	linux-4.19-gen-rkr3.2
PX30/PX30S	2018.02-rc3	10	3.4	4.19	V1.2.2_20221220	linux-4.19-gen-rkr3.2

Linux4.4 SDK

Chipset	Buildroot Version	Debian Version	Yocto Version	Kernel Version	SDK Version	TAG Version
RK3399PRO	2018.02-rc3	10	3.2	4.4	V1.4.2_20210202	N/A
RK1808	2018.02-rc3	N/A	N/A	4.4	V2.3.8_20210820	N/A
RK3399	2018.02-rc3	10	3.4	4.4	V2.9.0_20220620	linux-4.4-gen-rkr1
RK3326	2018.02-rc3	N/A	N/A	4.4	V1.8.0_20220620	linux-4.4-gen-rkr1
RK3328	2018.02-rc3	N/A	N/A	4.4	V1.1.0_20200603	N/A
RK3288	2018.02-rc3	10	3.4	4.4	V2.6.0_20220620	linux-4.4-gen-rkr1
PX30	2018.02-rc3	10	3.4	4.4	V1.8.0_20220620	linux-4.4-gen-rkr1
RK3308	2018.02-rc3	N/A	N/A	4.4	V1.5.2_20220124	N/A
RK3358	2018.02-rc3	N/A	N/A	4.4	V1.8.0_20220620	linux-4.4-gen-rkr1
RK312X	2018.02-rc3	N/A	N/A	4.4	V1.3.0_20220620	linux-4.4-gen-rkr1
PX3SE	2018.02-rc3	N/A	N/A	4.4	V1.0.0_20190124	N/A

2023-2024 Upgrade Schedule

Chipset	Buildroot Version	Debian Version	Yocto Version	Kernel Version	Release schedule
RK3328	2021.11	N/A	4.0	5.10	2023.Q4
RK3588	2021.11	12	4.0	6.1	2023.Q4

Contents

1. Chapter-1 SDK

1.1 Overview

Rockchip Linux SDK supports Buildroot, Yocto and Debian systems. Kernel is based on Kernel 4.4, Kernel 4.19 or Kernel5.10, and boot is based on U-boot v2017.09. It is suitable for all Linux products developed or secondary developed based on Rockchip EVB development board.

The development kit is suitable for but not limited to industrial applications, smart homes, consumer electronics, office and conference and other AIoT products, and provides flexible data path combination interfaces to meet the customized requirement of customers for free combination. For specific function debugging and interface description, please read the documents under the project directory docs/.

1.2 Obtain the General SDK

1.2.1 Get Source Code from Rockchip Code Server

To get Rockchip Linux SDK, customers need an account to access the source code repository provided by Rockchip. In order to get code synchronization permission, please provide SSH public key for server authentication and authorization when apply for SDK from Rockchip technical window. About Rockchip server SSH public key authorization, please refer to SSH Public Key Operation Introduction.

1.2.1.1 Rockchip Linux4.4 SDK Downloading

Chipset	Version	Download Command
RK3399	Linux4.4	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m \ rk3399_linux_release.xml
RK3326	Linux4.4	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m \ rk3326_linux_release.xml
RK3358	Linux4.4	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m \ rk3358_linux_release.xml
PX30	Linux4.4	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m \ px30_linux_release.xml
RK3308	Linux4.4	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m \ rk3308_linux_release.xml
RK312X	Linux4.4	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m \ rk312x_linux_release.xml
RK3399pro	Linux4.4	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rk/platform/manifests -b linux -m \ rk3399pro_linux_release.xml

1.2.1.2 Rockchip Linux4.19 SDK Downloading

Chipset	Version	Download Command
RK3566、RK3568	Linux4.19	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m rk356x_linux_release.xml
RK3399	Linux4.19	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m \ rk3399_linux4.19_release.xml
RK3326	Linux4.19	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m \ rk3326_linux4.19_release.xml
RK3358	Linux4.19	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m \ rk3358_linux4.19_release.xml
PX30	Linux4.19	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m \ px30_linux4.19_release.xml

1.2.1.3 Rockchip Linux5.10 SDK Downloading

Chipset	Version	Download Command
RK3562	Linux5.10	repo init --repo-url https://gerrit.rock-chips.com:8443/repo-release/tools/repo -u \ https://gerrit.rock-chips.com:8443/linux/rockchip/platform/manifests -b rk3562 -m \ rk3562_linux_release.xml
RK3588	Linux5.10	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m \ rk3588_linux_release.xml
RK3566、RK3568	Linux5.10	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m \ rk356x_linux5.10_release.xml
RK3399	Linux5.10	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m \ rk3399_linux5.10_release.xml
RK3326	Linux5.10	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m \ rk3326_linux5.10_release.xml
RK3358	Linux5.10	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m \ rk3358_linux5.10_release.xml
PX30	Linux5.10	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m \ px30_linux5.10_release.xml
RK3308	Linux5.10	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m \ rk3308_linux5.10_release.xml

Chipset	Version	Download Command
RK312X	Linux5.10	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m \ rk312x_linux5.10_release.xml
RK3036	Linux5.10	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m \ rk3036_linux5.10_release.xml

Repo, a tool built on Python script by Google to help manage git repositories, is mainly used to download and manage software repository of projects. The download address is as follows:

```
git clone ssh://git@www.rockchip.com.cn/repo/rk/tools/repo
```

1.2.2 Get Source Code from Local Compression Package

For quick access to SDK source code, Rockchip Technical Window usually provides corresponding version of SDK initial compression package. In this way, developers can get SDK source code through decompressing the initial compression package, which is the same as the one downloaded by repo.
Take RK3588_LINUX5.10_SDK_RELEASE_V1.0.0_20220520.tgz for example, After getting a initialization package, you can get source code by running the following command:

```
mkdir rk3588
tar xvf RK3588_LINUX5.10_SDK_RELEASE_V1.0.0_20220520.tgz -C rk3588
cd rk3588
.repo/repo/repo sync -l
.repo/repo/repo sync -c
```

Developers can update via `.repo/repo/repo sync -c` command according to update introductions that are regularly released by FAE window.

> Note:
> The software release version can be checked through the project xml file. The detailed way is as follows:

```
.repo/manifests$ realpath rk3588_linux_release.xml
In the following example: the printed version number is v1.1.1 and the update
time is 20230520
<SDK>/.repo/manifests/rk3588_linux_release_v1.1.1_20230520.xml
```

The initial compressed SDK packages currently released for Linux5.10 are as follows:

Chipset	Compressed Package	Version
RK3562	RK3562_LINUX5.10_SDK_RELEASE_V1.0.0_20230520.tgz	v1.0.0
RK3588	RK3588_LINUX5.10_SDK_RELEASE_V1.0.0_20220520.tgz	v1.0.0
RK3566, RK3568	RK356X_LINUX5.10_SDK_RELEASE_V1.0.0_20220920.tgz	v1.0.0
RK3399	RK3399_LINUX5.10_SDK_RELEASE_V1.0.0_20220920.tgz	v1.0.0
RK3326,RK3326S	RK3326_LINUX5.10_SDK_RELEASE_V1.0.0_20220920.tgz	v1.0.0
RK3358	RK3358_LINUX5.10_SDK_RELEASE_V1.0.0_20230420.tgz	v1.0.0
PX30,PX30S	PX30_LINUX5.10_SDK_RELEASE_V1.0.0_20220920.tgz	v1.0.0
RK3308	RK3308_LINUX5.10_SDK_RELEASE_V1.0.0_20220920.tgz	v1.0.0
RK312X	RK312X_LINUX5.10_SDK_RELEASE_V1.0.0_20220420.tgz	v1.0.0
RK3036	RK3036_LINUX5.10_SDK_RELEASE_V1.0.0_20220420.tgz	v1.0.0

> Notice:
> The initial compressed package may be replaced and updated with a new version!

1.3 Summary of SDK Build Commands

Chipset	Type	Reference model	One-click Building	rootfs Building	kernel compilation	uboot compilation
RK3588S	EVB	RK3588S EVB1	./build.sh lunch:rockchip_rk3588s_evb1_lp4x_v10_defconfig && ./build.sh	./build.sh rootfs	make ARCH=arm64 rockchip_linux_defconfig rk3588_linux.config; make ARCH=arm64 rk3588s-evb1-lp4x-v10-linux.img -j24	./make.sh rk3588 --spl-new
RK3588	EVB	RK3588 EVB1	./build.sh lunch:rockchip_rk3588_evb1_lp4_v10_defconfig && ./build.sh	./build.sh rootfs	make ARCH=arm64 rockchip_linux_defconfig rk3588_linux.config; make ARCH=arm64 rk3588-evb1-lp4-v10-linux.img -j24	./make.sh rk3588 --spl-new
RK3588	EVB	RK3588 EVB7	./build.sh lunch:rockchip_rk3588_evb7_lp4_v10_defconfig && ./build.sh	./build.sh rootfs	make ARCH=arm64 rockchip_linux_defconfig rk3588_linux.config; make ARCH=arm64 rk3588-evb7-lp4-v10-linux.img -j24	./make.sh rk3588 --spl-new
RK3568	EVB	RK3568 EVB1	./build.sh lunch:rockchip_rk3568_evb1_ddr4_v10_defconfig && ./build.sh	./build.sh rootfs	make ARCH=arm64 rockchip_linux_defconfig; make ARCH=arm64 rk3568-evb1-ddr4-v10-linux.img -j24	./make.sh rk3568 --spl-new
RK3568	EVB	RK3568 EVB8	./build.sh lunch:rockchip_rk3568_evb8_lp4_v10_defconfig && ./build.sh	./build.sh rootfs	make ARCH=arm64 rockchip_linux_defconfig; make ARCH=arm64 rk3568-evb8-lp4-v10-linux.img -j24	./make.sh rk3568 --spl-new
RK3566	EVB	RK3566 EVB2	./build.sh lunch:rockchip_rk3566_evb2_lp4x_v10_defconfig && ./build.sh	./build.sh rootfs	make ARCH=arm64 rockchip_linux_defconfig; make ARCH=arm64 rk3566-evb2-lp4x-v10-linux.img -j24	./make.sh rk3566 --spl-new
RK3562	Porduct	Dictionary pen	./build.sh lunch:rockchip_rk3562_dictpen_test3_v20_defconfig && ./build.sh	./build.sh rootfs	make ARCH=arm64 rk3562_linux_dictpen_defconfig; make ARCH=arm64 rk3562-dictpen-test3-v20.img -j24	./make.sh rk3562 --spl-new
RK3562	Robot Vacuum Cleaner	RK3562 EVB1	./build.sh lunch:rockchip_rk3562_evb1_lp4x_v10_defconfig && ./build.sh	./build.sh rootfs	make ARCH=arm64 rockchip_linux_defconfig rk3562_robot.config; make ARCH=arm64 rk3562-evb1-lp4x-v10-linux.img -j24	./make.sh rk3562 --spl-new
RK3562	Robot Vacuum Cleaner	RK3562 EVB2	./build.sh lunch:rockchip_rk3562_evb2_ddr4_v10_defconfig && ./build.sh	./build.sh rootfs	make ARCH=arm64 rockchip_linux_defconfig rk3562_robot.config; make ARCH=arm64 rk3562-evb2-ddr4-v10-linux.img -j24	./make.sh rk3562 --spl-new
RK3562	EVB	RK3562 EVB1	./build.sh lunch:rockchip_rk3562_evb1_lp4x_v10_defconfig && ./build.sh	./build.sh rootfs	make ARCH=arm64 rockchip_linux_defconfig; make ARCH=arm64 rk3562-evb1-lp4x-v10-linux.img -j24	./make.sh rk3562 --spl-new
RK3562	EVB	RK3562 EVB2	./build.sh lunch:rockchip_rk3562_evb2_ddr4_v10_defconfig && ./build.sh	./build.sh rootfs	make ARCH=arm64 rockchip_linux_defconfig; make ARCH=arm64 rk3562-evb2-ddr4-v10-linux.img -j24	./make.sh rk3562 --spl-new
RK3399	Industry board	RK3399 EVB IND	./build.sh lunch:rockchip_rk3399_evb_ind_lpddr4_defconfig && ./build.sh	./build.sh rootfs	make ARCH=arm64 rockchip_linux_defconfig; make ARCH=arm64 rk3399-evb-ind-lpddr4-linux.img -j24	./make.sh rk3399
RK3399	excavator	RK3399 SAPPHIRE EXCAVATOR	./build.sh lunch:rockchip_rk3399_sapphire_excavator_defconfig && ./build.sh	./build.sh rootfs	make ARCH=arm64 rockchip_linux_defconfig; make ARCH=arm64 rk3399-sapphire-excavator-linux.img -j24	./make.sh rk3399
RK3326	EVB	RK3326 EVB	./build.sh lunch:rockchip_rk3326_evb_lp3_v12_defconfig && ./build.sh	./build.sh rootfs	make ARCH=arm64 rockchip_linux_defconfig; make ARCH=arm64 rk3399-sapphire-excavator-linux.img -j24	./make.sh rk3399
RK3288	EVB	RK3288 EVB RK808	./build.sh lunch:rockchip_rk3288w_evb_rk808_defconfig && ./build.sh	./build.sh rootfs	make ARCH=arm rockchip_linux_defconfig; make ARCH=arm rk3288-evb-rk808-linux.img -j24	./make.sh rk3288

> Note:
>
> - **Rootfs Building**:
>   The SDK build Buildroot system by default. If you need other systems, you can set environment variables. for example：
>   Build Buildroot system: RK_ROOTFS_SYSTEM=buildroot ./build.sh rootfs

Build Debian system: RK_ROOTFS_SYSTEM=debian ./build.sh rootfs
Build Yocto system: RK_ROOTFS_SYSTEM=yocto ./build.sh rootfs

- **Build** Kernel, U-boot: a tool chain needs to be specified.
- SDK has a built-in 32-bit tool chain, for example:

export CROSS_COMPILE=../prebuilts/gcc/linux-x86/aarch64/gcc-arm-10.3-2021.07-x86_64-aarch64-none-linux-gnu/bin/aarch64-none-linux-gnu-

- SDK has a built-in 64-bit tool chain, for example:
  export CROSS_COMPILE=../prebuilts/gcc/linux-x86/arm/gcc-arm-10.3-2021.07-x86_64-arm-none-linux-gnueabihf/bin/arm-none-linux-gnueabihf-

2. Chapter-2 Documents Introduction

Documents released with Rockchip Linux SDK are designed to help developers quickly get started with development and debugging. The content involved in the documents does not cover all development information and issues. The document list will also be constantly updated. If you have any questions or requirements about documents, please contact our FAE window fae@rock-chips.com.
The docs directory in Rockchip Linux SDK is divided into two parts: Chinese (cn) and English (en) The Chinese directory comes with Common (general development guidance documents), Socs (chip platform related documents), Linux (Linux system development related documents), Others (other reference documents), docs_list_cn.txt (docs file directory structure), and the detailed introduction as follows:

2.1 General Development Guidance Document (Common)

Please see the documentation in each subdirectory of `<SDK>/docs/cn/Common` for details.

2.1.1 Peripheral Components Support List (AVL)

Please refer to `<SDK>/docs/cn/Common/AVL` directory for details, which contains the support list for DDR/eMMC/NAND FLASH/WIFI-BT/CAMERA. The support list is updated in real time on the redmine. The link is as follows:

```
https://redmine.rockchip.com.cn/projects/fae/documents
```

2.1.1.1 DDR Support List

For the Rockchip platform DDR chip support list, please refer to "Rockchip_Support_List_DDR_Ver2.55.pdf" in the `<SDK>/docs/cn/Common/AVL` directory. The following table shows the support level of DDR. It is only recommended to use chips marked with √ and T/A.
Table 1-1 Rockchip DDR Support Symbol

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

2.1.1.2 eMMC Support List

The eMMC chip support list for Rockchip platform can be found in the `<SDK>/docs/cn/Common/AVL` directory in the document titled 'RKeMMCSupportList_V1.73_20230303.pdf'. It is recommended to choose chips marked with √ or T/A in the support level table below.
Table 1-2 Rockchip eMMC 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

- **Selection of High-Performance eMMC Chips**

To enhance system performance, it is necessary to choose high-performance eMMC chips. Before selecting an eMMC chip, please refer to the models in the support list provided by Rockchip and pay close attention to the `performance` section in the manufacturer's datasheet.

Please select eMMC chips based on the manufacturer's size specifications and the read/write speeds. It is recommended to choose chips with sequential read speeds >200MB/s and sequential write speeds >40MB/s.

If there are any doubts about selection, you can also directly contact the Rockchip FAE team fae@rock-chips.com.

Figure 1-1 eMMC Performance Example

2.1.1.3 SPI NOR and SLC NAND Flash Support List

The SPI NOR and SLC NAND Flash support list for Rockchip platform, can be found in the document titled "RK_SpiNor_and_SLC_Nand_SupportList_V1.41_20230303.pdf" in the `<SDK>/docs/cn/Common/AVL` directory, the document also indicates the models of SPI NAND that can be selected. It is recommended to choose chips marked with √ or T/A in the support level table below.

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

2.1.1.4 NAND Flash Support List

Nand Flash support list for Rockchip platform, can be found in the /docs/Common/AVL` directory in the document titled “RKNandFlashSupportList Ver2.73_20180615.pdf”.the document also indicates the models of Nand Flash that can be selected. It is recommended to choose chips marked with √ or T/A in the support level table below.

Table 1-4 Rockchip Nand Flash 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

2.1.1.5 WIFI/BT Support List

WIFI/BT Support List for Rockchip Platform can be found in the document titled 'Rockchip_Support_List_Linux_WiFi_BT_20220828.pdf' in the `<SDK>/docs/cn/Common/AVL` directory. This document contains a comprehensive list of WIFI/BT chips tested extensively on the Rockchip platform. It is advisable to select models based on the list. For the testing of other WIFI/BT chips, corresponding kernel drivers should be provided from the original manufacturers of the WIFI/BT chips.

For any questions about chip selection, it is recommended to contact the Rockchip FAE team at fae@rock-chips.com.

2.1.1.6 Camera Support List

Camera Support List for Rockchip Platform can be found in the Camera Module Support List. This online list contains a comprehensive collection of Camera Modules extensively tested on the Rockchip platform. It is advisable to select models based on the list.

For any questions about module selection, it is recommended to contact the Rockchip FAE team at fae@rock-chips.com.

2.1.2 Audio Module Document (AUDIO)

It includes audio algorithms for microphones and relevant development documents for audio/Pulseaudio modules. The reference documents are as follows:

```
docs/cn/Common/AUDIO/
├── Algorithms
├── Rockchip_Developer_Guide_Audio_CN.pdf
└── Rockchip_Developer_Guide_PulseAudio_CN.pdf
```

2.1.3 CAN Module Document (CAN)

CAN bus, also known as Controller Area Network, is an efficient serial communication network used for distributed control or real-time control. The following documents mainly cover CAN driver development, communication testing tools, common command interfaces, and frequently asked questions.

```
docs/cn/Common/CAN/
├── Rockchip_Developer_Guide_CAN_FD_CN.pdf
└── Rockchip_Developer_Guide_Can_CN.pdf
```

2.1.4 Clock Module Document (CLK)

This document primarily covers clock development on the Rockchip platform, including Clock, GPIO, PLL spreading, etc.

```
docs/cn/Common/CLK/
├── Rockchip_Developer_Guide_Clock_CN.pdf
├── Rockchip_Developer_Guide_Gpio_Output_Clocks_CN.pdf
└── Rockchip_Developer_Guide_Pll_Ssmod_Clock_CN.pdf
```

2.1.5 CRYPTO Module Document (CRYPTO)

The following documents primarily focus on the development of Rockchip Crypto and HWRNG (TRNG), including driver development and upper-layer application development.

```
docs/cn/Common/CRYPTO/
└── Rockchip_Developer_Guide_Crypto_HWRNG_CN.pdf
```

2.1.6 DDR Module Document (DDR)

This module document mainly includes DDR development guide, DDR issue troubleshooting, DDR chip validation process, DDR board layout instructions, DDR bandwidth tool usage, and DDR DQ eye diagram tool, etc. for Rockchip platform.

```
docs/cn/Common/DDR/
├── Rockchip-Developer-Guide-DDR-CN.pdf
```

2.1.7 Debug Module Document (DEBUG)

This module document mainly includes introduction to the use of debugging tools such as DS5, FT232H_USB2JTAG, GDB_ADB, Eclipse_OpenOCD, etc. for Rockchip platform.

```
docs/cn/Common/DEBUG/
├── Rockchip_Developer_Guide_DS5_CN.pdf
├── Rockchip_Developer_Guide_FT232H_USB2JTAG.pdf
├── Rockchip_Developer_Guide_GDB_Over_ADB_CN.pdf
└── Rockchip_Developer_Guide_GNU_MCU_Eclipse_OpenOCD_CN.pdf
```

2.1.8 Display Module Document (DISPLAY)

This module document mainly includes development documents about DRM, DP, HDMI, MIPI, RK628 and other display modules for Rockchip platform.

```
docs/cn/Common/DISPLAY/
├── DP
├── HDMI
├── MIPI
├── RK628
├── Rockchip_BT656_TX_AND_BT1120_TX_Developer_Guide_CN.pdf
├── Rockchip_Developer_Guide_Baseparameter_Format_Define_And_Use_CN.pdf
├── Rockchip_Developer_Guide_DRM_Display_Driver_CN.pdf
├── Rockchip_Developer_Guide_RGB_MCU_CN.pdf
├── Rockchip_Develop_Guide_DRM_Direct_Show_CN.pdf
├── Rockchip_DRM_Panel_Porting_Guide_V1.6_20190228.pdf
└── Rockchip_RK3588_Developer_Guide_MIPI_DSI2_CN.pdf
```

2.1.9 Dynamic Frequency and Voltage Adjustment Module Documentation (DVFS)

This module document primarily covers CPU/GPU/DDR and other dynamic frequency and voltage adjustment modules for Rockchip platform.

Cpufreq and Devfreq are a set of framework models defined by kernel developers that support dynamic frequency and voltage adjustment based on specified governors. It effectively reduces power consumption while balancing performance.

```
docs/cn/Common/DVFS/
├── Rockchip_Developer_Guide_CPUFreq_CN.pdf
└── Rockchip_Developer_Guide_Devfreq_CN.pdf
```

2.1.10 File System Module Documentation

This module document primarily includes the development documentation related to the file system on the Rockchip platform.

```
docs/cn/Common/FS/
└── Rockchip_Developer_FAQ_FileSystem_CN.pdf
```

2.1.11 Ethernet Module Document (GMAC)

This module document primarily includes the development documentation related to the Ethernet GMAC interface on the Rockchip platform.

```
docs/cn/Common/GMAC/
├── Rockchip_Developer_Guide_Linux_GMAC_CN.pdf
├── Rockchip_Developer_Guide_Linux_GMAC_DPDK_CN.pdf
├── Rockchip_Developer_Guide_Linux_GMAC_Mode_Configuration_CN.pdf
├── Rockchip_Developer_Guide_Linux_GMAC_RGMII_Delayline_CN.pdf
└── Rockchip_Developer_Guide_Linux_MAC_TO_MAC_CN.pdf
```

2.1.12 HDMI-IN Module Document (HDMI-IN)

This module document mainly contains development documents related to the HDMI-IN interface of the Rockchip platform.

```
docs/cn/Common/HDMI-IN/
├── Rockchip_Developer_Guide_HDMI_IN_Based_On_CameraHal3_CN.pdf
└── Rockchip_Developer_Guide_HDMI_RX_CN.pdf
```

2.1.13 I2C Module Document (I2C)

This module document mainly contains development documents related to I2C interface of the Rockchip platform.

```
docs/cn/Common/I2C/
└── Rockchip_Developer_Guide_I2C_CN.pdf
```

2.1.14 IO Power Domain Module Document (IO-DOMAIN)

In the Rockchip platform, IO voltages generally include 1.8V, 3.3V, 2.5V, 5.0V, etc. Some IO interfaces support multiple voltage levels. The io-domain is responsible for configuring the IO power domain registers. It configures the corresponding voltage registers based on the actual hardware voltage range. Without proper configuration, the IO interfaces cannot function correctly.

```
docs/cn/Common/IO-DOMAIN/
└── Rockchip_Developer_Guide_Linux_IO_DOMAIN_CN.pdf
```

2.1.15 IOMMU Module Document (IOMMU)

It mainly introduces the Rockchip platform IOMMU for converting 32-bit virtual addresses and physical addresses. It has read and write control bits and can generate page missing exceptions and bus exception interrupts.

```
docs/cn/Common/IOMMU/
└── Rockchip_Developer_Guide_Linux_IOMMU_CN.pdf
```

2.1.16 Image Module Document (ISP)

ISP1.X is mainly suitable for RK3399/RK3288/PX30/RK3326/RK1808, etc.
ISP21 is mainly suitable for RK3566_RK3568, etc.
ISP30 is mainly suitable for RK3588, etc.
ISP32-lite is mainly suitable for RK3562, etc.

It contains ISP development documents, VI driver development documents, IQ Tool development documents, debugging documents and color debugging documents. The reference documents are as follows:

```
docs/cn/Common/ISP/
├── ISP1.X
├── ISP21
├── ISP30
├── ISP32-lite
└── The-Latest-Camera-Documents-Link.txt
```

> **Note：**
> Reference documents about RK3288/RK3399/RK3326/RK1808 Linux(kernel-4.4) rkisp1 driver, sensor driver, vcm driver is: "RKISP_Driver_User_Manual_v1.3_20190919";
> RK3288/RK3399/RK3326/RK1808 Linux(kernel-4.4) camera_engine_rkisp (3A repositry) reference documents is: "camera_engine_rkisp_user_manual_v2.0";
> Reference document for IQ effect file parameters of RK3288/RK3399/RK3326/RK1808 Linux(kernel-4.4) camera_engine_rkisp v2.0.0 version v2.0.0 and above is: "RKISP1_IQ_Parameters_User_Guide_v1.0_20190606".

2.1.17 MCU Module Document (MCU)

MCU development guide on the Rockchip platform.

```
docs/cn/Common/MCU/
└── Rockchip_RK3399_Developer_Guide_MCU_CN.pdf
```

2.1.18 MMC Module Document (MMC)

Development guide for interfaces such as SDIO, SDMMC, and eMMC on the Rockchip platform.

```
docs/cn/Common/MMC/
├── Rockchip_Developer_Guide_SDMMC_SDIO_eMMC_CN.pdf
└── Rockchip_Developer_Guide_SD_Boot_CN.pdf
```

2.1.19 Memory Module Document (MEMORY)

Process memory module mechanisms such as CMA and DMABUF on the Rockchip platform.

```
docs/cn/Common/MEMORY/
├── Rockchip_Developer_Guide_Linux_CMA_CN.pdf
├── Rockchip_Developer_Guide_Linux_DMABUF_CN.pdf
├── Rockchip_Developer_Guide_Linux_Meminfo_CN.pdf
└── Rockchip_Developer_Guide_Linux_Memory_Allocator_CN.pdf
```

2.1.20 MPP Module Document (MPP)

MPP development instructions on the Rockchip platform.

```
docs/cn/Common/MPP/
└── Rockchip_Developer_Guide_MPP_CN.pdf
```

2.1.21 Watchdog Module Document (WATCHDOG)

Development instructions for Watchdog on the Rockchip platform.

```
docs/cn/Common/WATCHDOG/
└── Rockchip_Developer_Guide_Linux_WDT_CN.pdf
```

2.1.22 NPU Module Document (NPU)

Instructions for developing NPU related tools on the Rockchip platform.

RKNN-TOOLKIT2: The development tool is located in the `external/rknn-toolkit2` directory and is primarily used for model conversion, model inference, and model performance evaluation. It is compatible with chips such as RK356X/RK3588.

The development and usage of RKNN API are located in the project directory `external/rknpu2` and are used for inferring RKNN models generated by RKNN-Toolkit2.
For detailed usage instructions, please refer to the documentation in the current directory.

```
docs/cn/Common/NPU/
├── README.md
├── rknn-toolkit2
│   ├── changelog-1.5.2.txt
│   ├── requirements_cp310-1.5.2.txt
│   ├── requirements_cp36-1.5.2.txt
│   ├── requirements_cp38-1.5.2.txt
│   ├── RKNNToolKit2_API_Difference_With_Toolkit1-1.5.2.md
│   ├── RKNNToolKit2_OP_Support-1.5.2.md
│   ├── Rockchip_Quick_Start_RKNN_Toolkit2_CN-1.5.2.pdf
│   ├── Rockchip_Trouble_Shooting_RKNN_Toolkit2_CN-1.5.2.pdf
│   └── Rockchip_User_Guide_RKNN_Toolkit2_CN-1.5.2.pdf
└── rknpu2
    ├── RK3588_NPU_SRAM_usage.md
    ├── RKNN_Compiler_Support_Operator_List_v1.5.2.pdf
    ├── RKNN_Dynamic_Shape_Usage.md
    ├── Rockchip_Quick_Start_RKNN_SDK_V1.5.2_CN.pdf
    ├── Rockchip_RKNPU_User_Guide_RKNN_API_V1.5.2_CN.pdf
    └── Rockchip_RV1106_Quick_Start_RKNN_SDK_V1.5.2_CN.pdf
```

2.1.23 NVM Module Document (NVM)

It mainly introduces the boot process on the Rockchip platform, configuring and debugging storage, OTP OEM area burning and other security interfaces。

```
docs/cn/Common/NVM/
├── Rockchip_Application_Notes_Storage_CN.pdf
├── Rockchip_Developer_FAQ_Storage_CN.pdf
├── Rockchip_Developer_Guide_Dual_Storage_CN.pdf
└── Rockchip_Developer_Guide_SATA_CN.pdf
```

2.1.24 PCIe Module Document (PCIe)

It mainly introduces the development instructions of PCIe on Rockchip platform.

```
docs/cn/Common/PCIe/
├── Rockchip_Developer_Guide_PCIe_CN.pdf
├── Rockchip_Developer_Guide_PCIE_EP_Stardard_Card_CN.pdf
├── Rockchip_Developer_Guide_PCIe_Performance_CN.pdf
├── Rockchip_PCIe_Virtualization_Developer_Guide_CN.pdf
└── Rockchip_RK3399_Developer_Guide_PCIe_CN.pdf
```

2.1.25 Performance Module Document (PERF)

Introduction to PERF Performance analysis on Rockchip Platform

```
docs/cn/Common/PERF/
├── Rockchip_Develop_Guide_Linux_RealTime_Performance_Test_Report_CN.pdf
├── Rockchip_Optimize_Tutorial_Linux_IO_CN.pdf
├── Rockchip_Quick_Start_Linux_Perf_CN.pdf
├── Rockchip_Quick_Start_Linux_Performance_Analyse_CN.pdf
├── Rockchip_Quick_Start_Linux_Streamline_CN.pdf
└── Rockchip_Quick_Start_Linux_Systrace_CN.pdf
```

2.1.26 GPIO Module Document (PINCTRL)

PIN-CTRL driver and DTS usage method on Rockchip platform.

```
docs/cn/Common/PINCTRL/
└── Rockchip_Developer_Guide_Linux_Pinctrl_CN.pdf
```

2.1.27 PMIC Module Document (PMIC)

Developer guide for PMICs such as RK805, RK806, RK808, RK809, and RK817 on the Rockchip platform.

```
docs/cn/Common/PMIC/
├── Rockchip_RK805_Developer_Guide_CN.pdf
├── Rockchip_RK806_Developer_Guide_CN.pdf
├── Rockchip_RK808_Developer_Guide_CN.pdf
├── Rockchip_RK809_Developer_Guide_CN.pdf
├── Rockchip_RK816_Developer_Guide_CN.pdf
├── Rockchip_RK817_Developer_Guide_CN.pdf
├── Rockchip_RK818_Developer_Guide_CN.pdf
├── Rockchip_RK818_RK816_Developer_Guide_Fuel_Gauge_CN.pdf
└── Rockchip_RK818_RK816_Introduction_Fuel_Gauge_Log_CN.pdf
```

2.1.28 Power Module Document (POWER)

Basic concepts and optimization methods for chip power consumption on Rockchip platform.

```
docs/cn/Common/POWER/
└── Rockchip_Developer_Guide_Power_Analysis_CN.pdf
```

2.1.29 PWM Module Document (PWM)

PWM developer guide on the Rockchip platform.

```
docs/cn/Common/PWM
└── Rockchip_Developer_Guide_Linux_PWM_CN.pdf
```

2.1.30 RGA Module Document (RGA)

RGA developer guide on the Rockchip platform.

```
docs/cn/Common/RGA/
├── Rockchip_Developer_Guide_RGA_CN.pdf
└── Rockchip_FAQ_RGA_CN.pdf
```

2.1.31 SARADC Module Document (SARADC)

SARADC developer guide on the Rockchip platform.

```
docs/cn/Common/SARADC/
└── Rockchip_Developer_Guide_Linux_SARADC_CN.pdf
```

2.1.32 SPI Module Document (SPI)

SPI developer guide on the Rockchip platform.

```
docs/cn/Common/SPI/
└── Rockchip_Developer_Guide_Linux_SPI_CN.pdf
```

2.1.33 Thermal Module Document (THERMAL)

Thermal developer guide on the Rockchip platform.

```
docs/cn/Common/THERMAL/
└── Rockchip_Developer_Guide_Thermal_CN.pdf
```

2.1.34 Tools Module Document (TOOL)

Instructions for using tools such as partitioning, mass production burning, and factory line burning on the Rockchip platform.

```
docs/cn/Common/TOOL/
├── Production-Guide-For-Firmware-Download.pdf
├── RKUpgrade_Dll_UserManual.pdf
├── Rockchip-User-Guide-ProductionTool-CN.pdf
├── Rockchip_Introduction_Partition_CN.pdf
└── Rockchip_User_Guide_Production_For_Firmware_Download_CN.pdf
```

2.1.35 Security Module Document (TRUST)

Introduction to functions such as TRUST and sleep & wake-up on the Rockchip platform

```
docs/cn/Common/TRUST/
├── Rockchip_Developer_Guide_Trust_CN.pdf
├── Rockchip_RK3308_Developer_Guide_System_Suspend_CN.pdf
├── Rockchip_RK3399_Developer_Guide_System_Suspend_CN.pdf
└── Rockchip_RK3588_Developer_Guide_System_Suspend_CN.pdf
```

2.1.36 UART Module Document (UART)

Introduction to serial port functions and debugging on the Rockchip Platform

```
docs/cn/Common/UART/
├── Rockchip_Developer_Guide_UART_CN.pdf
└── Rockchip_Developer_Guide_UART_FAQ_CN.pdf
```

2.1.37 UBOOT Module Document (UBOOT)

Introduction to U-Boot related development on the Rockchip platform

```
docs/cn/Common/UBOOT/
├── Rockchip_Developer_Guide_Linux_AB_System_CN.pdf
├── Rockchip_Developer_Guide_U-Boot_TFTP_Upgrade_CN.pdf
├── Rockchip_Developer_Guide_UBoot_MMC_Device_Analysis_CN.pdf
├── Rockchip_Developer_Guide_UBoot_MTD_Block_Device_Design_CN.pdf
├── Rockchip_Developer_Guide_UBoot_Nextdev_CN.pdf
└── Rockchip_Introduction_UBoot_rkdevelop_vs_nextdev_CN.pdf
```

2.1.38 USB Module Document (USB)

Introduction to USB development guide, USB signal testing and debugging tools on the Rockchip platform

```
docs/cn/Common/USB/
├── Rockchip_Developer_Guide_Linux_USB_Initialization_Log_Analysis_CN.pdf
├── Rockchip_Developer_Guide_Linux_USB_Performance_Analysis_CN.pdf
├── Rockchip_Developer_Guide_Linux_USB_PHY_CN.pdf
├── Rockchip_Developer_Guide_USB2_Compliance_Test_CN.pdf
├── Rockchip_Developer_Guide_USB_CN.pdf
├── Rockchip_Developer_Guide_USB_FFS_Test_Demo_CN.pdf
├── Rockchip_Developer_Guide_USB_Gadget_UAC_CN.pdf
├── Rockchip_Developer_Guide_USB_SQ_Test_CN.pdf
├── Rockchip_Introduction_USB_SQ_Tool_CN.pdf
├── Rockchip_RK3399_Developer_Guide_USB_CN.pdf
├── Rockchip_RK3399_Developer_Guide_USB_DTS_CN.pdf
├── Rockchip_RK356x_Developer_Guide_USB_CN.pdf
├── Rockchip_RK3588_Developer_Guide_USB_CN.pdf
└── Rockchip_Trouble_Shooting_Linux4.19_USB_Gadget_UVC_CN.pdf
```

2.2 Linux System Development Documents (Linux)

Please refer to documents under the `<SDK>/docs/cn/Linux`:

```
├── ApplicationNote
├── Audio
├── Camera
├── Docker
├── Graphics
├── Multimedia
├── Profile
├── Recovery
├── Security
├── System
├── Uefi
└── Wifibt
```

2.2.1 ApplicationNote

Development instructions for applications on the Rockchip platform, such as ROS, RetroArch, USB, etc

```
docs/cn/Linux/ApplicationNote/
├── Rockchip_Developer_Guide_Linux_Flash_Open_Source_Solution_CN.pdf
├── Rockchip_Instruction_Linux_ROS2_CN.pdf
├── Rockchip_Instruction_Linux_ROS_CN.pdf
├── Rockchip_Quick_Start_Linux_USB_Gadget_CN.pdf
└── Rockchip_Use_Guide_Linux_RetroArch_CN.pdf
```

2.2.2 Audio Development Documents (Audio)

Self developed audio algorithm on Rockchip platform.

```
docs/cn/Linux/Audio/
├── Rockchip_Developer_Guide_Microphone_Array_TEST_CN.pdf
├── Rockchip_Developer_Guide_Microphone_Array_Tuning.pdf
└── Rockchip_Introduction_Linux_Audio_3A_Algorithm_CN.pdf
```

2.2.3 Camera Development Documents (Camera)

MIPI/CSI Camera and Structured Light Development Guide on the Rockchip Platform

```
docs/cn/Linux/Camera/
├── Rockchip_Developer_Guide_Linux4.4_Camera_CN.pdf
├── Rockchip_Developer_Guide_Linux_RMSL_CN.pdf
└── Rockchip_Trouble_Shooting_Linux4.4_Camera_CN.pdf
```

2.2.4 Docker Development Documents (Docker)

Docker build and development of third-party systems such as Debian/Buildroot on the Rockchip platform.

```
docs/cn/Linux/Docker/
├── Rockchip_Developer_Guide_Debian_Docker_CN.pdf
├── Rockchip_Developer_Guide_Linux_Docker_Deploy_CN.pdf
└── Rockchip_User_Guide_SDK_Docker_CN.pdf
```

2.2.5 Graphics Development Documents (Graphics)

Linux Graphics related development on Rockchip Platform.

```
docs/cn/Linux/Graphics/
├── Rockchip_Developer_Guide_Buildroot_Weston_CN.pdf
└── Rockchip_Developer_Guide_Linux_Graphics_CN.pdf
```

2.2.6 Multimedia

The general process of video encoding and decoding on the Rockchip Linux platform:

```
vpu_service  -->  mpp --> gstreamer/rockit --> app
vpu_service: driver
MPP: A video encoding and decoding middleware for the Rockchip platform. Please
refer to the MPP documentation for detailed instructions
Gstreamer/rockit: used to connect components such as apps
```

Currently, gstreamer is used in Debian/Buildroot systems by default to connect apps and codec components.

Currently, the key development documents are as follows:

```
docs/cn/Linux/Multimedia/
├── Rockchip_Developer_Guide_Linux_RKADK_CN.pdf
├── Rockchip_User_Guide_Linux_Gstreamer_CN.pdf
└── Rockchip_User_Guide_Linux_Rockit_CN.pdf
```

Encoding and decoding functionalities can also be tested directly through the testing interfaces provided by MPP (such as mpi_dec_test\mpi_enc_test...). MPP source code reference is located in `<SDK>/external/mpp/`. For testing demos, please refer to `<SDK>/external/mpp/test`. Please refer to the SDK document `Rockchip_Developer_Guide_MPP_CN.pdf` for details.

Rockchip chips like RK3588 support powerful multimedia capabilities:

- Supports H.265/H.264/AV1/VP9/AVS2 video decoding, up to 8K60FPS, also supports 1080P multi-format video decoding (H.263, MPEG1/2/4, VP8, JPEG).
- Supports 8K H.264/H.265 video encoding and 1080P VP8, JPEG video encoding.
- Video post-processing: deinterlacing, noise reduction, edge/detail/color optimization.

Below are the reference specifications for common chip encoding and decoding capabilities on each platform.

> **Note:** The maximum testing specifications are related to numerous factors, so the same decoding IP specifications might differ among different chips. Chip support may vary in different systems, leading to differences in supported formats and performance.

- **Decoding Capability Specification Table**

Chip	H264	H265	VP9	JPEG
RK3588	7680X4320@30f	7680X4320@60f	7680X4320@60f	1920x1088@200f
RK3566/RK3568	4096x2304@60f	4096x2304@60f	4096x2304@60f	1920x1080@60f
RK3562	1920x1088@60f	2304x1440@30f	4096x2304@30f	1920x1080@120f
RK3399	4096x2304@30f	4096x2304@60f	4096x2304@60f	1920x1088@30f
RK3328	4096x2304@30f	4096x2304@60f	4096x2304@60f	1920x1088@30f
RK3288	3840x2160@30f	4096x2304@60f	N/A	1920x1080@30f
RK3326	1920x1088@60f	1920x1088@60f	N/A	1920x1080@30f
PX30	1920x1088@60f	1920x1088@60f	N/A	1920x1080@30f
RK312X	1920x1088@30f	1920x1088@60f	N/A	1920x1080@30f

- **Coding Capability Specification Table**

Chip	H264	H265	VP8
RK3588	7680x4320@30f	7680x4320@30f	1920x1088@30f
RK3566/RK3568	1920x1088@60f	1920x1088@60f	1920x1088@30f
RK3562	1920x1088@60f	N/A	N/A
RK3399	1920x1088@30f	N/A	1920x1088@30f
RK3328	1920x1088@30f	1920x1088@30f	1920x1088@30f
RK3288	1920x1088@30f	N/A	1920x1088@30f
RK3326	1920x1088@30f	N/A	1920x1088@30f
PX30	1920x1088@30f	N/A	1920x1088@30f
RK312X	1920x1088@30f	N/A	1920x1088@30f

2.2.7 SDK Profile introduction (Profile)

Including software testing, benchmarks, etc. on Rockchip Linux platform.

```
docs/cn/Linux/Profile/
├── Rockchip_Developer_Guide_Linux_PCBA_CN.pdf
├── Rockchip_Introduction_Linux_Benchmark_KPI_CN.pdf
├── Rockchip_Introduction_Linux_PLT_CN.pdf
└── Rockchip_User_Guide_Linux_Software_Test_CN.pdf
```

2.2.8 OTA Upgrade (Recovery)

An introduction to the recovery development process and upgrade during OTA upgrade of Rockchip Linux platform.

```
docs/cn/Linux/Recovery/
├── Rockchip_Developer_Guide_Linux_DFU_Upgrade_CN.pdf
├── Rockchip_Developer_Guide_Linux_Recovery_CN.pdf
├── Rockchip_Developer_Guide_Linux_Upgrade_CN.pdf
└── Rockchip_Introduction_Smart_Screen_OTA_CN.pdf
```

2.2.9 Security Solution (Security)

Introduction to the secure boot solution of Securboot and TEE on Rockchip Linux platform

```
docs/cn/Linux/Security/
├── Rockchip_Developer_Guide_Linux_Secure_Boot_CN.pdf
└── Rockchip_Developer_Guide_TEE_SDK_CN.pdf
```

2.2.10 System Development (System)

Introduction to the porting and development guide for Debian and other third-party systems on the Rockchip Linux platform

```
docs/cn/Linux/System/
├── Rockchip_Developer_Guide_Debian_CN.pdf
└── Rockchip_Developer_Guide_Third_Party_System_Adaptation_CN.pdf
```

2.2.11 UEFI Booting (UEFI)

Introduction to UEFI boot solution on Rockchip Linux platform。

```
docs/cn/Linux/Uefi/
└── Rockchip_Developer_Guide_UEFI_CN.pdf
```

2.2.12 Network Module (RKWIFIBT)

Introduction to the development of WIFI, BT, etc. on Rockchip Linux platform。

```
docs/cn/Linux/Wifibt/
├── AP module RF test document
├── REALTEK module RF test document
├── Rockchip_Developer_Guide_Linux_WIFI_BT_CN.pdf
├── WIFIBT programming interface
└── WIFI performance testing PC tool
```

2.2.13 DPDK Module (DPDK)

DPDK development guide on Rockchip Linux platform.

```
docs/cn/Linux/DPDK/
└── Rockchip_Developer_Guide_Linux_DPDK_CN.pdf
```

2.3 Chip Platform Related Documents (Socs)

Refer to the documentation in the `<SDK>/docs/cn/<chipset_name>` directory. Normally, it will include the release notes, quick start, software development guide, hardware development guide, Datasheet, etc.

2.3.1 Release Note

It contains an overview of the chip, supported main functions, instructions for obtaining the SDK, etc.

Reference document is in the `<SDK>/docs/cn/<chipset_name>` directory:
`Rockchip_<chipset_name>_Linux_SDK_Release_<version>_CN.pdf`

2.3.2 Quick Start

Normally, it will include software and hardware development guide, SDK build, SDK pre-build firmware, SDK burning, etc.
Please refer to the documentation in the `<SDK>/docs/cn/<chipset_name>/Quick-start` directory.

2.3.3 Software Development Guide

In order to help developers get familiar with the development and debugging of the SDK faster, the "Rockchip_Developer_Guide_Linux_Software_CN.pdf" document can be obtained from the /docs/cn/<chip_name>/ directory and will be continuously improved and updated.

2.4 Datasheet

In order to help developers get familiar with chip development and debugging faster, a chip datasheet is released with the SDK.

Please refer to the documentation in the `<SDK>/docs/cn/<chipset_name>/Datasheet` directory.

2.4.1 Hardware Development Guide

Rockchip platform will have corresponding hardware reference documents released with the SDK software package. The hardware user guide mainly introduces the basic features, hardware interfaces, and usage methods of the reference hardware board. It aims to assist developers in using the EVB more quickly and accurately, and in developing related products. For more details, please refer to the documents in the `<SDK>/docs/cn/<chip_name>/Hardware` directory.

2.5 Other Documents (Others)

For other reference documents, such as Repo mirror environment construction, Rockchip SDK application and synchronization guide, Rockchip Bug system usage guide, etc., please refer to the documents in the `<SDK>/docs/cn/Others` directory.

```
docs/cn/Others/
├── Rockchip_Developer_Guide_Repo_Mirror_Server_Deploy_CN.pdf
├── Rockchip_Trouble_Shooting_Linux_Real-Time_Performance_CN.pdf
├── Rockchip_User_Guide_Bug_System_CN.pdf
└── Rockchip_User_Guide_SDK_Application_And_Synchronization_CN.pdf
```

2.6 Documents List (docs_list_cn.txt)

Please refer to the document in the /docs/cn/docs_list_cn.txt` directory.

```
├── Common
├── Linux
├── Others
├── Rockchip_Developer_Guide_Linux_Software_CN.pdf
├──<chipset_name>
└── docs_list_cn.txt
```

3. Chapter-3 Tools Introduction

Tools released with the Rockchip Linux SDK are used for development, debugging, and mass production process. The tool versions will be continuously updated along with SDK updates. If you have any questions or requirements about the tools, please contact our FAE team: fae@rock-chips.com.

In the Rockchip Linux SDK, tools are included in the `tools` directory for use in Linux (tools for Linux operating system environment), Mac (tools for macOS operating system environment), and Windows (tools for Windows operating system environment).

- Windows Tools

Tool usage instructions: `<SDK>/tools/windows/ToolsRelease.txt`

Tool name	Description
boot_merger	Packing or unpacking loader tool
DDR_UserTool	DDR user testing tool
DriverAssitant	Driver installation tool
EfuseTool	efuse burning tool
FactoryTool	Mass production upgrade tool
ParameterTool	Partition table modification tool
pin_debug_tool	GPIO debugging tool
programmer_image_tool	Programmer upgrade tool
rk_ddrbin_tool	RK ddrbin debugging tool
RKDevInfoWriteTool	Device information writing tool
RKDevTool	Discrete firmware upgrade and entire update firmware tool
RKDevTool_Release	Firmware burning tool
RKPCBATool	PCBA board testing tool
rk_sign_tool	Secureboot signature tool
Rockchip_HdcpKey_Writer	HDCP key burning tool
Rockchip_USB_SQ_Tool	USB PHY signal quality debugging work
SDDiskTool	SD card boot or upgrade image creation
SecureBootTool	Firmware Signature Tool
upgrade_tool	Command line upgrade tool
RKImageMaker	Command line packaging tool

- Linux Tools

Tool usage instructions: `<SDK>/tools/linux/ToolsRelease.txt`

Tool name	Description
boot_merger	Packing or unpacking loader tool
Firmware_Merger	SPI NOR firmware packing tool (the generated firmware can be used in the programmer)
Linux_DDR_Bandwidth_Tool	DDR bandwidth statistics tool
Linux_Diff_Firmware	OTA differential package tool
Linux_Pack_Firmware	Firmware packaging tool (packaged into updata.img)
Linux_SecureBoot	Firmware Signature Tool
Linux_SecurityAVB	AVB Signature tool
Linux_SecurityDM	DM Signature Tool
Linux_Upgrade_Tool	Firmware burning tool
pin_debug_tool	GPIO debugging tool
programmimg_image_tool	Packaged SPI NOR/SPI NAND/SLC NAND/eMMC programmer firmware
rk_ddrbin_tool	RK ddrbin debugging tool
rk_sign_tool	Secureboot signature tool

- Mac tools

Tool documentation: `<SDK>/tools/mac/ToolsRelease.txt`

Tool name	Description
boot_merger	Packing or unpacking loader tool
upgrade_tool	Command line upgrade tool
rk_sign_tool	Secureboot signature tool

3.1 Driver Installation Tool

Rockchip USB driver installation assistant is stored in `<SDK>/tools/windows/DriverAssitant_v5.12.zip`. xp, win7_32, win7_64, win10_32, win10_64 and other operating systems are supported.

The installation steps are as follows:

3.2 Burning Tools

- The SDK provides Windows burning tools (the tool version requires V3.15 or above), which is located in the project root directory:

```
<SDK>/tools/windows/RKDevTool/
```

- The SDK provides Linux burning tools (the version of Linux_Upgrade_Tool requires V2.17 or above), the tool is located in the project root directory:

```
<SDK>/tools/linux/Linux_Upgrade_Tool/Linux_Upgrade_Tool
```

```
Linux_Upgrade_Tool$ sudo ./upgrade_tool -h

--------------------Tool Usage --------------------
Help:             H
Version:          V
Log:              LG
------------------Upgrade Command ------------------
ChooseDevice:       CD
ListDevice:     LD
SwitchDevice:       SD
UpgradeFirmware:    UF <Firmware> [-noreset]
UpgradeLoader:      UL <Loader> [-noreset] [FLASH|EMMC|SPINOR|SPINAND]
DownloadImage:      DI <-p|-b|-k|-s|-r|-m|-u|-t|-re image>
DownloadBoot:       DB <Loader>
EraseFlash:     EF <Loader|firmware>
PartitionList:      PL
WriteSN:        SN <serial number>
ReadSN:         RSN
ReadComLog:     RCL <File>
CreateGPT:      GPT <Input Parameter> <Output Gpt>
SwitchStorage:      SSD
----------------Professional Command -----------------
TestDevice:     TD
ResetDevice:        RD [subcode]
ResetPipe:      RP [pipe]
ReadCapability:     RCB
ReadFlashID:        RID
ReadFlashInfo:      RFI
ReadChipInfo:       RCI
ReadSecureMode:     RSM
WriteSector:        WS  <BeginSec> <PageSizeK> <PageSpareB> <File>
ReadLBA:        RL  <BeginSec> <SectorLen> [File]
WriteLBA:       WL  <BeginSec> [SizeSec] <File>
EraseLBA:       EL  <BeginSec> <EraseCount>
EraseBlock:     EB <CS> <BeginBlock> <BlokcLen> [--Force]
RunSystem:      RUN <uboot_addr> <trust_addr> <boot_addr> <uboot> <trust> <boot>
-------------------------------------------------------
```

3.3 Packaging Tools

Mainly used to package each discrete firmware into a complete update.img firmware for easy upgrade.

- How to package update.img firmware in Windows environment, run the following command to generate update.img

```
<SDK>/tools/windows/RKDevTool/rockdev/mkupdate.bat
```

- How to package update.img firmware in Linux environment, run the following command to generate update.img

```
<SDK>/tools/linux/Linux_Pack_Firmware/rockdev/mkupdate.sh
```

3.4 Tools used to Make SD Card Upgrading and Booting

It is used for making SD card upgrades, SD card booting, and SD card PCBA testing.

```
<SDK>/tools/windows/SDDiskTool_v1.74.zip
```

3.5 Device Information Writing Tool

`<SDK>/tools/windows/RKDevInfoWriteTool*`
Unzip RKDevInfoWriteTool-1.3.0.7z and install it. Open the software with administrator rights. Please refer to the `Rockchip_User_Guide_RKDevInfoWriteTool_CN.pdf` in the current directory for tool usage,.

3.6 Firmware Signature Tool

It is used for efuse/otp signing of firmware.

- The SDK provides Windows signature tool which is located in the project root directory:

```
<SDK>/tools/windows/SecureBootTool_v2.2
```

- The SDK provides Linux signing tool which is located in the project root directory:

```
<SDK>/tools/linux/rk_sign_tool_v1.31_linux.zip

rk_sign_tool_v1.3_linux$ ./rk_sign_tool
rk_sign_tool is a tool signing firmware and loader for secureboot
usage of rk_sign_tool v1.3:
CC <--chip chip_id> //select sign options by chip
KK [--bits default=2048] <--out> //generating rsa key pairs
LK <--key> <--pubkey> //loading rsa key pairs
SI [--key] <--img> [--pss] //signing image like boot uboot trust
SL [--key] [--pubkey] <--loader> [--little] [--pss] //signing loader like
RKXX_loader.bin
```

```
SF [--key] [--pubkey] <--firmware> [--little] [--pss] //signing firmware like
update.img
SB [--key] <--bin> [--pss] //signing binary file
GH <--bin> <--sha 160|256> [--little] //computing sha of binary file
******rk_sign_tool XX -h to get more help******
```

Please refer to the document:
/docs/cn/Linux/Security/Rockchip_Developer_Guide_Linux4.4_SecureBoot_CN.pdf for details.

3.7 Programmer Upgrade Tool

It is used for mass production of programmer image making tools, this tool is located at:

```
<SDK>/tools/windows/programmer_image_tool or
<SDK>/tools/linux/programmer_image_tool
```

Programmer image creation steps:

- Burn image to eMMC

```
./programmer_image_tool -i update.img -t emmc
```

- Burn image to SPI Nor Flash

```
./programmer_image_tool -i update.img -t spinor
```

Please refer to the `user_manual.pdf` document in the tool directory for more instructions.

3.8 PCBA Testing Tools

PCBA testing tools are used to help quickly identify the quality of product functions during mass production and improve production efficiency. Currently includes screen(LCD), wireless (Wi-Fi), Bluetooth (bluetooth), DDR/eMMC storage, SD card (sdcard), USB HOST, key (KEY), speaker and earphone (Codec) and other test items.

These test items include automatic test items and manual test items. Wireless network, DDR/eMMC, and Ethernet are automatic test items. Button, SD card, USB HOST, Codec are manual test items.

PCBA tools are located at:

```
<SDK>/tools/windows/RKPCBATool_V1.0.9.zip
```

For detailed PCBA function configuration and usage instructions, please refer to:

/tools/windows/RKPCBATool_V1.0.9/Rockchip PCBA Test Development Guide_1.10.pdf

3.9 DDR Soldering Test Tool

It is used to test the hardware connection of DDR and troubleshoot hardware problems such as false soldering:

```
<SDK>/tools/windows/DDR_UserTool_v1.41.zip
```

3.10 eFuse Programming Tool

It is used for eFuse programming, suitable for RK3288/RK3368/RK3399/RK3399Pro and other platforms.

```
<SDK>/tools/windows/EfuseTool_v1.42zip
```

If the chip uses eFuse to enable the SecureBoot function, please ensure that there is no problem with the hardware connection, because when eFuse is programmed, Kernel has not been started yet, so please ensure that VCC_EFUSE has power in MaskRom state before it can be used.

The /Tools/windows/EfuseTool_v1.4.zip is used to let the board enter MaskRom state.
Click "Firmware", select the signed update.img, or Miniloader.bin, click "Start" to start programming eFuse.

3.11 Mass Production Upgrade Tool

It is used for burning firmware during mass production in factory:

```
<SDK>/tools/windows/FactoryTool_v1.76.zip
```

3.12 Partition Modification Tool

The tool is used for partition modification in Parameter.txt:

```
<SDK>/tools/windows/ParameterTool_v1.2.zip
```

4. Chapter-4 SDK Software Framework

4.1 Introduction to SDK project Directory

A typical Linux SDK project directory includes buildroot, debian, app, kernel, u-boot, device, docs, external, etc. Repositories are managed by manifests, and the `repo` tool is utilized to manage each directory or its corresponding Git project, including the following subdirectories:

- `app`: Contains upper-level application apps, mainly various application demos.
- `buildroot`: Contains the root file system developed with Buildroot.
- `debian`: Contains the root file system developed with Debian.
- `device/rockchip`: Contains chip board-level configurations and scripts/files for compiling and packaging firmware.
- `docs`: Contains general development guides, Linux system development guides, chip platform-related documents, etc.
- `external`: Contains third-party related repositories, including display, audio/video, camera, network, security, etc.
- `kernel`: Contains Kernel development code.
- `output`: Contains firmware information, compilation details, XML files, host environment, etc., generated during each build.
- `prebuilts`: Contains cross-compilation toolchains.
- `rkbin`: Contains Rockchip-related binaries and tools.
- `rockdev`: Contains compiled output firmware, actually, it soft link to xxx `output/firmware`.
- `tools`: Contains commonly used tools for both Linux and Windows operating systems.
- `u-boot`: Contains U-Boot code developed based on version v2017.09.
- `yocto`: Contains the root file system developed with Yocto.

SDK Overview

The general Linux SDK currently integrates several mainstream Linux distributions. Distributions refer to the systems we commonly use on Linux hosts, providing users with pre-integrated Linux operating systems and various application software. Linux distributions come in various forms, ranging from fully-featured desktop systems to server versions and lightweight systems for small devices. For example, Debian supports desktop versions, and there are customizable lightweight systems like Buildroot and Yocto.

Customers can choose the system of product based on detailed product requirements. The specific system description are as follows:

4.1.1 Buildroot System

The Buildroot system in Rockchip Linux SDK includes various system source code, drivers, tools, and application software packages used for Linux system development. Buildroot is an open-source embedded Linux system automatic build framework on the Linux platform. The entire Buildroot consists of Makefile scripts and Kconfig configuration files. By configuring Buildroot, a complete Linux system software that can be directly flashed and run on the device.

Figure 1-1 Buildroot Compilation Block Diagram

Advantages of Buildroot:

- **Source Code Building:** Buildroot offers great flexibility through source code building.
- **Convenient Cross-Compilation Environment:** It provides an easy-to-use cross-compilation environment for quick and efficient building, making it beginner-friendly.
- **Component Configuration:** Buildroot allows easy customization of various system components, facilitating tailored development.

One of the most famous projects using Buildroot is OpenWrt. Images created with OpenWrt can run on routers with 16 M SPI NOR flash memory, containing only essential components. This simplicity is achieved thanks to Buildroot. The entire Buildroot project is maintained in a single Git repository.

Buildroot uses kconfig and make, a defconfig configuration represents a BSP support.

Buildroot itself does not have the ability to expand, and users need to complete the work through scripts. These listed features are all different from Yocto.

> At present, Buildroot is a critical system that we develop and maintain internally.

4.1.2 Yocto

Yocto, like Buildroot, is a set of tools for building embedded systems, but their styles are completely different. Yocto projects are maintained through individual packages (meta), with some packages responsible for the core and others for peripherals. Some packages are designed for running Rockchip chips, some for installing Weston, and others for running Debian. Similar mechanisms are used for Node.js. The Yocto community is highly active and expansive, allowing everyone to share their achievements on GitHub. This mechanism ensures that we can reuse others' work from the internet, which is very valuable. Compared to the Buildroot system, Yocto has a stronger compilation mechanism. It automatically handles dependencies and recompilation of third-party packages. However, it is relatively complex and requires a VPN network.

Figure 2-1 Yocto Compilation block diagram

Yocto is a very flexible build system that allows users to use the shell and Python to handle a variety of special cases. Currently, the system is mainly targeted at foreign customers. If you need more information about Yocto, check out the following references:

Yocto

Rockchip Yocto

Yocto stable branch

> At present, Yocto system is mainly used by overseas, enthusiasts or customers with secondary development capabilities.

4.1.3 Debian

Debian is a Linux operating system that is completely free, open-source, and widely used on various devices. The reasons for choosing Debian include:

- **Debian is Free Software:**
  Debian is composed of free and open-source software and will always remain 100% free. Everyone can use, modify, and distribute it freely. Developers can build upon the Debian system created based on Rockchip.

- **Debian is Stable and Secure:**
  Debian is a stable and secure operating system based on Linux, widely used across devices, including laptops, desktops, and servers. Since 1993, its stability and reliability have made it popular among users. Debian provides reasonable default configurations for each software package, and developers strive to provide security updates throughout its lifecycle.
- **Wide Hardware Support:**
  Most hardware is supported by the Linux kernel. When free software cannot provide sufficient support, dedicated hardware drivers can be used. Currently, chips such as Rockchip RK3588/RK3568/RK3566/RK3399/RK3288 have been adapted and supported.
- **Smooth Updates:**
  Debian is known for its smooth updates within its release cycle, allowing easy transitions to the next major version. Rockchip has upgraded from Debian Stretch (9) to Debian Buster (10) and Bullseye (11).
- **Foundation for Many Other Distributions:**
  Debian serves as the foundation for many popular Linux distributions such as Ubuntu, Knoppix, PureOS, SteamOS, and Tails. Debian provides all the tools so that anyone can extend the software packages in the Debian archive to meet their requirements.
- **A Community Project:**
  Debian is not just a Linux operating system; it is created collaboratively by hundreds of volunteers from around the world. Anyone can be a part of the Debian community, even if they are not a programmer or system administrator.

The customized version of Debian for Rockchip is created through shell scripts, which involve obtaining Debian distribution source code, compiling, and installing the Rockchip hardware acceleration package into the operating system.

> Currently, Debian system is mainly aimed at customer preliminary evaluations, third-party system porting references, and products with complex desktop requirements.

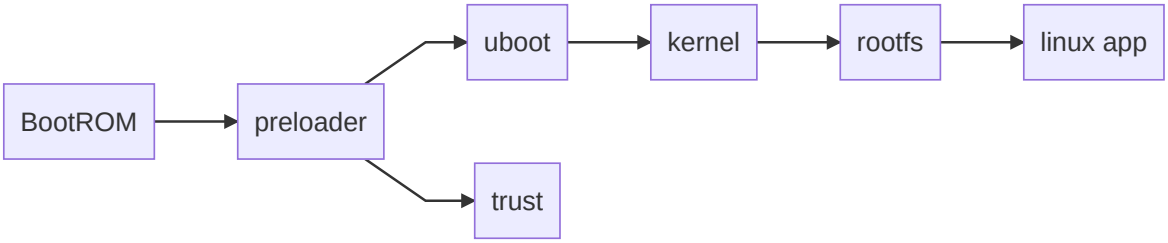
4.2 SDK Software Architecture

As illustrated in Figure 3-1 of the SDK software architecture, it is divided into four layers from bottom to top: the Boot Layer, Kernel Layer, Libraries, and Application Layer. Each layer is described as follows:

- **Boot Layer:** This layer primarily handles system booting processes and provides support for components such as BootROM, U-Boot, ATF, and related functionalities.
- **Kernel Layer:** The Kernel Layer provides the standard implementation of the Linux Kernel. Linux is an open-source operating system. The core of the Rockchip platform's Linux system is based on the standard Linux 4.4/4.19/5.10 kernel versions. It offers foundational support for security, memory management, process management, network protocol stack, etc. The Linux kernel manages hardware resources of devices, such as CPU scheduling, caching, memory, I/O, etc.
- **Libraries:** This layer corresponds to general embedded systems and functions as middleware. It includes various system basic libraries and support for third-party open-source libraries. Libraries provide API interfaces to the application layer. System customizers and application developers can develop new applications based on the API provided by the Libraries layer.
- **Application Layer:** The Application Layer implements specific product features and interaction logic. It requires support from system basic libraries and third-party libraries. Developers can create their own applications, leveraging various system capabilities to deliver the final user experience.

Figure 3-1 SDK Software Block Diagram

The SDK system startup process refers to a software process from system power on to completion of system startup. The following is the Linux system startup process:



> Description:
> Both AP and MCU have a BOOTROM integrated inside. When the system is powered on, it will first run the BOOTROM code, and then the BOOTROM code will detect the peripheral memory and load the Loader code.
> There are currently 3 Pre Loaders: miniloader (non-open source), uboot spl and loader.

4.3 SDK Development Process

The Rockchip Linux system is based on the Buildroot/Yocto/Debian system, and kernel is developed based on kernel 4.4/4.19/5.10. It is an SDK developed for a variety of different product forms. Based on this SDK, system customization and application porting development can be effectively realized.

Figure 4-1 SDK Development Process

As shown in Figure 4-1, developers can follow the development process described above to quickly set up the Rockchip Linux system development environment and build code locally. Here is a brief overview of this process:

- **Check System Requirements:** Before downloading code and compiling, ensure that the local development device meets the requirements, including hardware capabilities, software system, toolchain, etc. Currently, the SDK supports compilation under the Linux operating system environment and provides toolchain support only for Linux. Other systems like MacOS and Windows are not supported at the moment.
- **Set Up Compilation Environment:** Provide information about the various software packages and tools that need to be installed on the development device. For details, please refer to SDK Development Environment Setup.
- **Select Device:** During the development process, developers need to choose the appropriate hardware board based on their requirements. Please refer to SDK Adapted Hardware Automatic Compilation Summary.
- **Download Source Code:** After selecting the device type, install the repo tool for batch downloading of source code. For details, please refer to SDK Software Packages.
- **System Customization:** Developers can customize U-Boot, Kernel, and Rootfs based on the hardware board and product definition. Please refer to SDK Development.
- **Compile and Package:** After setting up the source code and selecting the product, initialize the relevant compilation environment. Then execute compilation commands, including overall or module compilation, and cleanup work. For further details, please refer to SDK Compilation Instructions.
- **Flash and Run:** After generating the image files, learn how to flash the image and run it on the hardware device. For further details, please refer to SDK Firmware Upgrade.

5. Chapter-5 SDK Development Environment Setup

5.1 Overview

This section mainly introduces how to set up a local compilation environment to build Rockchip Linux SDK source code. Currently, the SDK only supports compilation and secondary development in a Linux environment.

A typical embedded development environment typically includes a Linux server, PC, and target hardware board, as shown in the diagram below.

- Set up a cross-compilation environment on the Linux server to provide services such as code provision, updates downloading, and compilation for software development.
- Share programs between the PC and Linux server, and install Putty or Minicom. Remote login to the Linux server over the network is used for cross-compilation and development code debugging.
- Connect the PC to the target hardware board via serial ports and USB. The compiled image file can be flashed onto the target hardware board, and the system or application program can be debugged.

> Note: Windows PC is used in the development environment. In fact, many tasks can also be completed on Linux PC, such as using Minicom instead of Putty. Users can choose by themselves.

5.2 Preparation Work before SDK Development

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

5.2.1 Install and Configure Git

Please configure your own git information before using repo, otherwise subsequent operations may encounter hook checking errors:

- Update sources and install git

```
sudo apt update && sudo apt-get install git
```

- Configure user information

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

5.2.2 Install and Configure repo

repo is a script written by Google using Python script to call git. It is mainly used to download and manage the project's software repositories.

The installation path in the following commands takes "~/bin" as an example. Users should create the required directories themselves.

```
mkdir ~/bin
cd ~/bin
git clone ssh://git@www.rockchip.com.cn/repo/rk/tools/repo
chmod a+x ~/bin/repo/repo
export PATH=~/bin/repo:$PATH
```

In addition to the above methods, you can also use the following command to obtain repo

- In foreign regions, you can download the repo tool from the Google mirror site:

```
mkdir ~/bin
curl https://storage.googleapis.com/git-repo-downloads/repo -o ~/bin/repo
chmod a+x ~/bin/repo
export PATH=~/bin:$PATH
```

- In domestic areas, you can download the repo tool from the tsinghua.edu.cn site:

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

5.2.3 SDK Obtaining

The SDK is released through Rockchip code server. When customers apply for the SDK through Rockchip's technical window, they need an account to access the source code repository provided by Rockchip. In order to be able to obtain code synchronization, please provide SSH public key for server authentication and authorization when apply for SDK from Rockchip technical window. About Rockchip server SSH public key authorization, please refer to the Chapter SSH Public Key Operation Instructions in this document.

5.2.3.1 SDK Download Command

Rockchip Linux SDK can be adapted to different chip platforms, such as RK3588, RK3562, RK3566, RK3568, RK3308, RK3288, RK3326/PX30, RK3399, RK3399Pro, RK1808, etc. The source code of different chip platforms will be different to a certain extent. Developers need to declare the chip platform they want when downloading the source code, so as to avoid downloading code they don't need.

SDK uses different XML to declare the corresponding chip platform you want to download.

Please refer to the Chapter Download through code server in this document.

When the code start to download automatically, just wait patiently. The source code files will be located in the working directory under the corresponding project name. The initial sync operation will take an hour or more to complete.

5.2.3.2 Compressed Package of SDK Code

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

Please refer to the Chapter Obtain by decompressing the local compressed package in this document.

5.2.3.3 Software Update History

The software release version upgrade can be viewed through the project xml. For example, the way for the RK3588 chip is as follows:

```
.repo/manifests$ realpath rk3588_linux_release.xml
In the example: the printed version number is v1.3.0 and the update time is
20230920
<SDK>/.repo/manifests/rk3588_linux_release_v1.3.0_20230920.xml
```

Software release version upgrade and update content can be viewed through the project text. Please refer to the project directory. For example, check the version of RK3588 as follows:

```
<SDK>/.repo/manifests/rk3588_linux/RK3588_Linux5.10_SDK_Note.md

<SDK>/docs/cn/RK3588/RK3588_Linux5.10_SDK_Note.md
```

5.2.3.4 SDK Update

Developers can synchronize updates through commands according to the update instructions published regularly in the FAE window.

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

5.2.3.5 SDK Issue Feedback

In order to better assist customer development, the Rockchip bug system (Redmine) records the user problem handling process and status, making it easier for both parties to track at the same time, making issue processing more timely and efficient. SDK issues, specific technical issues, technical consultation, etc. can be submitted to this bug system, and Rockchip technical services will distribute, handle and track the issues in a timely manner. For more detailed instructions, please refer to the documentation

`<SDK>/docs/cn/Others/Rockchip_User_Guide_SDK_Application_And_Synchronization_CN.pdf`

。

5.3 Linux Server Development Environment Setup

It is recommended to use Ubuntu 22.04 for compilation. Other Linux versions may need to adjust the software package accordingly. In addition to the system requirements, there are other hardware and software requirements. Hardware requirements: 64-bit system, hard disk space should be greater than 40G. If you do multiple builds, you will need more hard drive space

Considering the time cost of setting up customer's development environment, we also provide the image mode of cross compiler docker for customer verification, so as to shorten the time-consuming of setting up the compilation environment.

Reference documents `Docker/Rockchip_Developer_Guide_Linux_Docker_Deploy_EN.pdf`.

- **The compatibility test results of docker compilation image system are as follows**:

Release version	Docker version	Image loading	Firmware compilation
ubuntu 22.04	20.10.21	pass	pass
ubuntu 21.10	20.10.12	pass	pass
ubuntu 21.04	20.10.7	pass	pass
ubuntu 18.04	20.10.7	pass	pass
fedora35	20.10.12	pass	NR (not run)

5.3.1 Install Libraries and Toolsets

When using the command line for device development, you can install the libraries and tools required for compiling the SDK by the following steps.

Use the following `apt-get` command to install the libraries and tools required for the following operations:

```
sudo apt-get update && sudo apt-get install git ssh make gcc libssl-dev \
liblz4-tool expect expect-dev g++ patchelf chrpath gawk texinfo chrpath \
diffstat binfmt-support qemu-user-static live-build bison flex fakeroot \
cmake gcc-multilib g++-multilib unzip device-tree-compiler ncurses-dev \
libgucharmap-2-90-dev bzip2 expat gpgv2 cpp-aarch64-linux-gnu libgmp-dev \
libmpc-dev bc python-is-python3 python2
```

> **Note:**
>
> The installation command is applicable to Ubuntu22.04. For other versions, please use the corresponding installation command according to the name of the installation package. If you encounter an error when compiling, you can install the corresponding software package according to the error message.
>
> - If a PC cannot access the Google website while compiling Buildroot, DNS needs to be set up to support downloading dl packages using the domestic image kgithub.com
> - Python 3.6 or later versions is required to be installed, and python 3.6 is used as an example here.
> - It is requires `make 4.0` and above version to be installed, take `make 4.2` as an example here.
> - lz4 1.7.3 or later versions is required to be installed.
> - Compiling yocto requires a VPN network, and git does not have the CVE-2022-39253 security detection patch.

5.3.1.1 Setting up DNS to support for kgithub.com

```
sudo sed -i '$a 43.154.68.204\tkgithub.com' /etc/hosts
sudo sed -i '$a 43.155.83.75\traw.kgithub.com
objects.githubusercontent.kgithub.com' /etc/hosts
```

5.3.1.2 Check and Upgrade the `python` Version of the Host

The method to check and upgrade the `python` version of the host is as follows:

- Check the `python` version of your host

```
$ python3 --version
Python 3.10.6
```

If it does not meet the requirements of Python>=3.6 version, you can upgrade as follows:

- Upgrade to `python 3.6.15` new version

```
PYTHON3_VER=3.6.15
echo "wget
https://www.python.org/ftp/python/${PYTHON3_VER}/Python-${PYTHON3_VER}.tgz"
echo "tar xf Python-${PYTHON3_VER}.tgz"
echo "cd Python-${PYTHON3_VER}"
echo "sudo apt-get install libsqlite3-dev"
echo "./configure --enable-optimizations"
echo "sudo make install -j8"
```

5.3.1.3 Check and Upgrade the `make` Version of the Host

The method to check and upgrade the `make` version of the host is as follows:

- Check the `make` version

```
$ make -v
GNU Make 4.2
Built for x86_64-pc-linux-gnu
```

- Upgrade to `make 4.2` new version

```
$ sudo apt update && sudo apt install -y autoconf autopoint

git clone https://gitee.com/mirrors/make.git
cd make
git checkout 4.2
git am $BUILDROOT_DIR/package/make/*.patch
autoreconf -f -i
./configure
make make -j8
sudo install -m 0755 make /usr/bin/make
```

5.3.1.4 Check and Upgrade the `lz4` Version of the Host

The method to check and upgrade the `lz4` version of the host is as follows:

- Check the `lz4` version of the host

```
$ lz4 -v
*** LZ4 command line interface 64-bits v1.9.3, by Yann Collet ***
```

- Upgrade to new version of `lz4`

```
git clone https://gitee.com/mirrors/LZ4_old1.git
cd LZ4_old1

make
sudo make install
sudo install -m 0755 lz4 /usr/bin/lz4
```

5.3.1.5 Check and Upgrade the `git` Version of the Host

- Check the `git` version of the host

```
$ git -v
git version 2.38.0
```

- Upgrade `git` to new version

```
$ sudo apt update && sudo apt install -y libcurl4-gnutls-dev

git clone https://gitee.com/mirrors/git.git --depth 1 -b v2.38.0
cd git
make git -j8
make install
sudo install -m 0755 git /usr/bin/git
```

5.3.2 Linux Server System Version

This SDK development environment installs the following versions of Linux systems. The SDK is compiled with this Linux system by default:

```
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=22.04
DISTRIB_CODENAME=jammy
DISTRIB_DESCRIPTION="Ubuntu 22.04 LTS"
Linux version 5.15.0-46-generic (buildd@lcy02-amd64-115) (gcc (Ubuntu 11.2.0-
19ubuntu1) 11.2.0, GNU ld (GNU Binutils for Ubuntu) 2.38) #49-Ubuntu SMP Thu Aug
4 18:03:25 UTC 2022
```

5.3.3 Introduction to Cross-compilation Tool Chain

Since Rockchip Linux SDK is currently only built in the Linux PC environment, we only provide the cross-compilation tool chain under Linux. The tool chain in the prebuilt directory is used by U-Boot and Kernel. Specifically, Rootfs needs to use its corresponding tool chain, or use a third-party tool chain for building.

5.3.3.1 U-Boot and Kernel Compilation Tool Chain

```
Linux4.4/4.19 SDK:
prebuilts/gcc/linux-x86/aarch64/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-

Linux5.10 SDK:
prebuilts/gcc/linux-x86/aarch64/gcc-arm-10.3-2021.07-x86_64-aarch64-none-linux-gnu/bin/aarch64-none-linux-gnu-
```

Corresponding version:

```
Linux4.4/4.19 SDK:
gcc version 6.3.1 20170404 (Linaro GCC 6.3-2017.05)

Linux5.10 SDK:
gcc version 10.3.1 20210621 (GNU Toolchain for the A-profile Architecture 10.3-2021.07 (arm-10.29))
```

5.3.3.2 Buildroot Tool Chain

If you need to build a single module or third-party application, you need to configure the cross-compilation environment. For example, RK3588, its cross-compilation tool is located in the `buildroot/output/rockchip_rk3588/host/usr` directory, you need to add the tool's bin/ directory and set the `aarch64-buildroot-linux-gnu/bin/` directory as an environment variable, and execute a script to automatically configure environment variables in the top-level directory:

```
source buildroot/envsetup.sh
```

Enter the command to view:

```
cd buildroot/output/rockchip_rk3588/host/usr/bin
./aarch64-linux-gcc --version
```

The following information will be printed:

```
aarch64-linux-gcc.br_real (Buildroot linux-5.10-gen-rkr6) 12.3.0
```

If you need tool chains for other platforms or versions, you need to build them by yourself.
After the above environment is prepared, the Linux server development environment has been set up and the source code can be downloaded and built.

Please refer to the following documents for details:

```
buildroot/docs/manual/using-buildroot-toolchain.txt
buildroot/docs/manual/adding-packages.txt
```

5.3.3.3 Debian Toolchain

Use docker on the device, gcc or `dpkg-buildpackage` to build.

5.3.3.4 Yocto Toolchain

Reference is as follows:

```
https://wiki.yoctoproject.org/wiki/Building_your_own_recipes_from_first_principle
s#Adding_new_recipes_to_the_build_system
https://docs.yoctoproject.org/dev/dev-manual/new-recipe.html
```

5.4 Window PC Development Environment Setup

5.4.1 Development Tool Installation

- Users should install editing software such as Vim and Notepad++ by themselves.
- Download Virtual-Box (software that provides a virtual development environment)
- Download XShell6 (used to establish interaction with Linux system)

5.4.2 Rockchip USB Driver Installation

During the development and debugging process, the device needs to be switched to Loader mode or Maskrom mode, and Rockusb driver should be installed, then the device can be recognized normally.
Rockchip USB driver installation assistant is stored in tools/windows/DriverAssitant_v5.x.zip. support xp, win7_32, win7_64, win10_32, win10_64 and other operating systems.
The installation steps are as follows:

5.4.3 Windows Burning Tool Usage

The burning tool for Windows systems is released in `<SDK>/tools/windows/RKDevTool/RKDevTool_Release`, which can be used for development, debugging and firmware burning in Windows environment. Please refer to Development and programming tools for details.

5.4.4 Target Hardware Board Preparation

Please refer to the SDK software package applicable hardware list SDK software package applicable hardware list to select the corresponding hardware board for development and debugging. The corresponding hardware instruction document will introduce the hardware interface, usage instructions and burning operation methods.

6. Chapter-6 SDK Version and Update Instructions

The Linux SDK currently has different versions such as Linux 4.4, Linux 4.19, and Linux 5.10. Use repo to manage the SDK and use tags to manage versions. The various components and modules of the SDK are stored in different Git repositories, and the repo tool is used to coordinate version and code synchronization between them. Each component's Git repository has its own branch and commit history.

6.1 SDK Naming Rules

6.1.1 SDK tag Naming Rules

System - System version - form - rkr Release version

For example: Linux5.10 SDK tag is linux-5.10-gen-rkr1`

Note:

- System: Linux, Android, RTOS, etc.
- System version: Linux4.4, Linux4.19, Linux5.10, etc.
- Form: gen, common, standard, ipc, chip model, etc.
- Release version: external: alpha, beta,release; internal: rka1, rkb1, rkr1

6.1.2 SDK Release Document Naming Rules

Rockchip[1]*RK3399[2]*Linux5.10[3]*SDK[4]*Release[5]*V1.0.0[6]*20220920[7]_CN [8]

> Note:
> [1]: Rockchip, required. Documents released by RK must begin with Rockchip.
> [2]: Chip model. The initial letter of the chip is uniformly capitalized, the suffix depends on the chip and remains consistent with the official website definition. If there are two chips, connect them with an underscore such as (RK3566_RK3568).
> [3]: Linux, Android, FreeRTOS, etc., the first letter is capitalized.
> [4]: Optional, indicating the target market. If it is a general SDK, this item is omitted.
> [5] and [6]: Release type (Alpha/Beta/Release) and version number. The version number style is defined as follows: "Vn.n.n" . The letter "V" remains capitalized and n represents an integer. It can be seen that the version number consists of three numbers with 2 dots in the middle.
>
> - Alpha version starts from V0.0.1, that is, the first two digits keep the number 0, and the last digit is incremented according to the actual testing/release situation;
> - The Beta version starts from V0.1.1, that is, the first digit is kept as 0, the increment of the second digit indicates the addition of functional modules or major updates of functions, and the increment of the third digit indicates bug correction;
> - Release version starts from V1.0.0, the first digit indicates a major update, and the second and third digits refer to the Beta version.

> [7]: Release date. Use the format 20220920 uniformly.
> [8]: Document language version information. CN: Chinese, EN: English

6.1.3 SDK Compressed Package Naming Rules

Chip platform - software version - [target market] + SDK + version number + release date

Example:
RK3399_LINUX5.10_SDK_RELEASE_V1.0.0_20220920

> Note:
>
> 1. All English letters are capitalized
> 2. Software platform: LINUX5.10
> 3. The target market is enclosed in square brackets to indicate optional options. If it is a general SDK, this item is omitted.
> 4. The version number of the SDK compressed package continues to retain 3 digits with two dots in the middle.

6.2 SDK Update Mechanism

6.2.1 SDK Update

If there is an update to the SDK, it will be updated around the 20th of each month, and an Change Notice will be sent out at the end of the month. The update mechanism is a one-month small update (such as V1.0.x), and the version has been fully tested by QA (such as V1.x.0).

6.2.2 SDK Release Patch

If any important issues are encountered when releasing the SDK, they will be pushed to customers in the form of the following patches:

rockchip-patch

6.3 How to Build Server to Sync the SDK

How to build a Gitolite server and use it for management and maintenance of SDK code. Please refer to the following documents for details.

```
<SDK>/docs/cn/Others/Rockchip_Developer_Guide_Repo_Mirror_Server_Deploy.pdf
```

7. Chapter-7 SDK Build Instructions

SDK can be configured and built related functions through `make` or `./build.sh` and target parameters. For details, please refer to `device/rockchip/commo/README.md` compilation instructions.

7.1 SDK Compilation Command

`make help` , for example：

```
$ make help
  menuconfig             - interactive curses-based configurator
  oldconfig              - resolve any unresolved symbols in .config
  syncconfig             - Same as oldconfig, but quietly, additionally update
deps
  olddefconfig           - Same as syncconfig but sets new symbols to their
default value
  savedefconfig          - Save current config to RK_DEFCONFIG (minimal config)
  ...
```

`make` is used to run `./build.sh`

That is, you can also run `./build.sh <target>` to build related functions. You can view the detailed compilation commands through `./build.sh help`.

```
$ ./build.sh -h
Usage: build.sh [OPTIONS]
Available options:
lunch               - choose defconfig
*_defconfig         - switch to specified defconfig
olddefconfig        - resolve any unresolved symbols in .config
savedefconfig       - save current config to defconfig
menuconfig          - interactive curses-based configurator
kernel-5.10         - build kernel 5.10
kernel              - build kernel
modules             - build kernel modules
loader              - build loader (uboot|spl)
uboot               - build u-boot
spl                 - build spl
uefi                - build uefi
wifibt              - build Wifi/BT
rootfs              - build rootfs (default is buildroot)
buildroot           - build buildroot rootfs
yocto               - build yocto rootfs
debian              - build debian rootfs
recovery            - build recovery
pcba                - build PCBA
security_check      - check contidions for security features
createkeys          - build secureboot root keys
security_uboot      - build uboot with security paramter
security_boot       - build boot with security paramter
```

```
security_recovery  - build recovery with security paramter
security_rootfs    - build rootfs and some relevant images with security paramter
(just for dm-v)
updateimg          - build update image
otapackage         - build OTA update image
sdpackage          - build SDcard update image
firmware           - generate and check firmwares
all                - build all basic image
save               - save images and build info
allsave            - build all & firmware & updateimg & save
cleanall           - cleanup
post-rootfs        - trigger post-rootfs hook scripts
shell              - setup a shell for developing
help               - usage

Default option is 'allsave'.
```

7.2 SDK Board Level Configuration

`make rockchip_defconfig`, the detailed board-level configuration instructions are as follows:

Enter the project `<SDK>/device/rockchip/<chipset_name>` directory:

Board Level Configuration	**Description**
rockchip_rk3588_evb1_lp4_v10_defconfig	Suitable for RK3588 EVB1 with LPDDR4 development board
rockchip_rk3588_evb3_lp5_v10_defconfig	Suitable for RK3588 EVB3 with LPDDR5 development board
rockchip_rk3588_evb7_lp4_v10_defconfig	Suitable for RK3588 EVB7 with LPDDR4 development board
rockchip_rk3588s_evb1_lp4x_v10_defconfig	Suitable for RK3588S EVB1 with LPDDR4 development board
rockchip_rk3562_evb1_lp4x_v10_defconfig	Suitable for RK3562 EVB1 with LPDDR4 development board
rockchip_rk3562_evb2_ddr4_v10_defconfig	Suitable for RK3562 EVB2 with DDR4 development board
rockchip_defconfig	The default configuration will be linked to the default board level configuration through `make lunch` or `./build.sh lunch` for configuration
Take RK3562 for example:	

```
$ ./build.sh lunch

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

1. rockchip_defconfig
2. rockchip_rk3562_evb1_lp4x_v10_defconfig
3. rockchip_rk3562_evb2_ddr4_v10_defconfig
Which would you like? [1]:
```

The configuration of other functions can be configured through `make menuconfig` to configure related properties.

7.3 Configures Different boot, kernel, system or other Modules of the SDK

The SDK can be configured through `make menuconfig`. The currently available modules are as follows:

```
(rk3562) SoC
   Rootfs  --->
   Loader (u-boot)  --->
   Kernel  --->
   Boot  --->
   Recovery (buildroot)  --->
   PCBA test (buildroot)  --->
   Security  --->
   Update (OTA and A/B)  --->
   Firmware  --->
   Extra partitions  --->
   Others configurations  --->
```

Through the above config, you can choose different rootfs/loader/kernel and other configurations to perform various customized compilations. It also comes with a powerful command line switching function.

> Note that after menuconfig is configured, you should use `make savedefconfig` to save the configuration.

7.4 SDK Environment Variable Configuration

Configurations for different chips and target functions can be set via `source envsetup.sh`.

```
$ source envsetup.sh
Top of tree: /home/wxt/linux-develop/rk3562

You're building on Linux
Pick a board:

1. rockchip_px30_32
2. rockchip_px30_64
3. rockchip_px30_recovery
4. rockchip_rk3036
```

```
5. rockchip_rk3036_recovery
6. rockchip_rk3126c
7. rockchip_rk312x
8. rockchip_rk312x_recovery
9. rockchip_rk3288
10. rockchip_rk3288_recovery
11. rockchip_rk3308_32_release
12. rockchip_rk3308_b_32_release
13. rockchip_rk3308_b_release
14. rockchip_rk3308_bs_32_release
15. rockchip_rk3308_bs_release
16. rockchip_rk3308_h_32_release
17. rockchip_rk3308_recovery
18. rockchip_rk3308_release
19. rockchip_rk3326_64
20. rockchip_rk3326_recovery
21. rockchip_rk3358_32
22. rockchip_rk3358_64
23. rockchip_rk3358_recovery
24. rockchip_rk3399
25. rockchip_rk3399_base
26. rockchip_rk3399_recovery
27. rockchip_rk3399pro
28. rockchip_rk3399pro-multi-cam
29. rockchip_rk3399pro-npu
30. rockchip_rk3399pro-npu-multi-cam
31. rockchip_rk3399pro_combine
32. rockchip_rk3399pro_recovery
33. rockchip_rk3528
34. rockchip_rk3528_recovery
35. rockchip_rk3562
36. rockchip_rk3562_32
37. rockchip_rk3562_recovery
38. rockchip_rk3566
39. rockchip_rk3566_32
40. rockchip_rk3566_recovery
41. rockchip_rk3566_rk3568_base
42. rockchip_rk3566_rk3568_ramboot
43. rockchip_rk3568
44. rockchip_rk3568_32
45. rockchip_rk3568_recovery
46. rockchip_rk3588
47. rockchip_rk3588_base
48. rockchip_rk3588_ramboot
49. rockchip_rk3588_recovery
Which would you like? [1]:
```

For example, select 35 `rockchip_rk3562`

Then go to the Buildroot directory of RK3562 and start compiling the relevant modules.

7.5 Fully Automatic Compilation

Enter the project root directory and execute the following command to automatically complete all compilation:

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

./build.sh     # build module code (u-Boot, kernel, Rootfs, Recovery)
               # Package into update.img complete upgrade package
               #Copy and generate all compilation information to the out
directory
```

It is Buildroot by default, you can specify different rootfs by setting the environment variable `RK_ROOTFS_SYSTEM`, which can set three systems currently: Buildroot, Debian, and Yocto.

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

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

7.6 Build Modules

7.6.1 Build U-Boot

Enter the SDK project. Run the following command to build:

```
<SDK>#./build.sh uboot
```

For detailed board-level build, please refer to the build instructions in the SDK release document.

7.6.2 Build Kernel

Enter the root directory of the project directory and execute the following command to automatically build and package kernel.

```
<SDK>#./build.sh kernel
```

For detailed board-level compilation, please refer to the release notes or the compilation instructions in Quick Start.

7.6.3 Build Recovery

Enter the project root directory and execute the following command to automatically build and package of Recovery.

```
<SDK>#./build.sh recovery
```

After the compilation, recovery.img is generated in the Buildroot directory:

`output/rockchip_<chipset_name>_recovery/images`.

> Note: recovery.img contains the kernel (kernel.img), so every time the Kernel changes, Recovery needs to be repackaged and generated. Recovery repackaging method is as follows:

```
<SDK>#source buildroot/envsetup.sh
<SDK>#cd buildroot
<SDK>#make recovery-reconfigure
<SDK>#cd -
<SDK>#./build.sh recovery
```

For more compilation instructions, please refer to the SDK release documentation.

> Note: Recovery is a non-essential function, and some board-level configurations will not set it.

7.6.4 Build Buildroot

Enter the root directory of the project directory and execute the following command to automatically complete the compilation and packaging of Rootfs:

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

After compilation, images in different formats are generated in the Buildroot directory: `output/rockchip_<target>/images`. The `rootfs.ext4` format is used by default.

7.6.4.1 Cross-compilation of Buildroot

If you need to build a single module or third-party application, you need to configure the cross-compilation environment. For example, cross-compilation tool of RK3562 is located in the `buildroot/output/rockchip_rk3562/host/usr` directory. You need to set the tool's bin/ directory and `aarch64-buildroot-linux-gnu/bin/` directory as environment variables, and execute the script to automatically configure environment variables in the top directory (only valid for the current console).

```
source buildroot/envsetup.sh
```

Enter the command to view:

```
cd buildroot/output/rockchip_rk3562/host/usr/bin
./aarch64-linux-gcc --version
```

The following information will be printed:

```
aarch64-linux-gcc.br_real (Buildroot linux-5.10-gen-rkr6) 12.3.0
```

7.6.4.2 Buildroot Module Compilation

For example, for the rockchip-test module, the commonly used related compilation commands are as follows:

Enter the buildroot directory

```
SDK#cd buildroot
```

- Build rockchip-test

```
buildroot#make rockchip-test
```

- Re-build rockchip-test

```
buildroot#make rockchip-test-reconfigure
```

- Delete rockchip-test

```
buildroot#make rockchip-test-dirclean
or
buildroot#rm -rf output/rockchip_rk3562/build/rockchip-test-master/
```

7.6.5 Build Debian

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

> **Note: The following related dependency packages need to be installed in advance**

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

Or enter the debian/ directory:

```
cd debian/
```

For subsequent compilation and Debian firmware generation, please refer to the readme.md in the current directory.

- **(1) Building base Debian system**

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

Build 64-bit Debian:

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

After compilation is completed, the following will be generated in the debian/ directory: linaro-buster-alip-xxxxx-1.tar.gz (xxxxx represents the generation timestamp).

FAQ:

- If the above compilation encounters the following problems:

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

Solution:

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

In addition, if you encounter other compilation exceptions, first check that the compilation system used is not ext2/ext4.

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

Debian uses live build, and if the image source is changed to domestic, it can be configured like this:

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

If the package cannot be downloaded due to other network reasons, there are pre-generated packages shared in Debian Base Download Package, and placed in the current directory to directly and go directly to the next step.

- **(2) Building rk-debian rootfs**

Build 64-bit Debian:

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

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

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

At this point, linaro-rootfs.img will be generated.

7.6.6 Build Yocto

Enter the root directory of the project directory and execute the following command to automatically complete the compilation and packaging of Rootfs:

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

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

FAQ:
If you encounter the following problems during compiling the above steps:

```
Please use a locale setting which supports UTF-8 (such as LANG=en_US.UTF-8).
Python can't change the filesystem locale after loading so we need a UTF-8
when Python starts or things won't work.
```

Solution:

```
locale-gen en_US.UTF-8
export LANG=en_US.UTF-8 LANGUAGE=en_US.en LC_ALL=en_US.UTF-8
```

Or refer to setup-locale-python3. The image generated after compilation is in `yocto/build/lastest/rootfs.img`. The default username login is root.

For more information about Yocto, please refer to Rockchip Wiki.

8. Chapter-8 SDK Firmware Upgrade

This chapter mainly introduces the process of building a complete image file (image), burning it and running it on a hardware device.
The introduction of several image burning tools provided by Rockchip platform is as follows. You can choose the appropriate burning method for burning. Before burning, you need to install the latest USB driver. For details, please see Rockchip USB Driver Installation.

Tools	Run System	Description
RKDevTool	Windows	Rockchip development tools, discrete firmware upgrade and entire update firmware upgrade tools
FactoryTool	Windows	Mass production upgrade tool, supports USB one-to-multiple burning
Linux_Upgrade_Tool	Linux	A tool developed under Linux to support firmware upgrades

8.1 Burning Mode Introduction

The several modes of Rockchip platform hardware operation are as shown in the following table. Firmware can only be burned or updated on the board when the device is in Maskrom or Loader mode.

Mode	Tool Burn	Description
Maskrom	Support	When firmware is not yet burned into the Flash, the Chip will enter Maskrom mode, allowing the initial firmware burning. During the development and debugging process, if Loader fails to start normally, firmware can also be burned in Maskrom mode.
Loader	Support	In Loader mode, firmware can be burned and upgraded. You can use tools to burn a certain partition image file separately to facilitate debugging.
Recovery	Not supported	System boot recovery starts, its main function is to upgrade and restore factory settings.
Normal Boot	Not supported	System boot rootfs starts, loads rootfs, most development is debugged in this mode.

There are several ways to enter the burning mode:

- The board has not been burned with firmware. Power on and enter Maskrom mode.
- The board has been burned with firmware, press and hold the recovery button to power on or reset, and the system will enter the Loader firmware burning mode.
- The board has been burned with firmware, press and hold the Maskrom button to power on or reset, and the system will enter the MaskRom firmware burning mode.

- The board has been burned with firmware, and the board enters the system normally after powering on or resetting, Rockchip's development tool displays "An ADB device was found" or "An MSC device was found", and then click the "Switch" button on the tool to enter Loader mode.
- The board has been burned with firmware, you can enter `reboot loader` command in the serial port or ADB command line mode to enter the Loader mode.

8.1.1 Windows Flashing Instructions

Rockchip's SDK provides a Windows burning tool (the tool version requires V3.15 or above). The tool is located in the project root directory:

```
tools/
├── windows/RKDevTool
```

As shown in the figure below, after compiling and generating the corresponding firmware, the device needs to enter the MASKROM or BootROM burning mode.
After connecting the USB download cable, press and hold the "MASKROM" button and press the reset button "RST" and then release it to enter MASKROM mode, after loading the corresponding path of the compiled and generated firmware, click "Execute" to burn. You can also press and hold the "recovery" button and press the reset button "RST" and then release it to enter the loader mode for burning. The following is the partition offset and burning file in the MASKROM mode . (Note: Windows PC requires administrator rights to run the tool before it can be executed)

Note: Before burning, you need to install the latest USB driver. For driver details, please see:

```
<SDK>/tools/windows/DriverAssitant_v5.12.zip
```

8.1.2 Linux Flashing Instructions

The Linux burning tool is located in the tools/linux directory (Linux_Upgrade_Tool tool version requires V2.17 or above), please make sure your board is connected to MASKROM/loader rockusb. For example, if the generated firmware is in the rockdev directory, the upgrade command is as follows:

```
sudo ./upgrade_tool ul rockdev/MiniLoaderAll.bin -noreset
sudo ./upgrade_tool di -p rockdev/parameter.txt
sudo ./upgrade_tool di -u rockdev/uboot.img
sudo ./upgrade_tool di -trust rockdev/trust.img ##For new chips, trust has been
merged into the uboot partition
sudo ./upgrade_tool di -misc rockdev/misc.img
sudo ./upgrade_tool di -b rockdev/boot.img
sudo ./upgrade_tool di -recovery rockdev/recovery.img
sudo ./upgrade_tool di -oem rockdev/oem.img
sudo ./upgrade_tool di -rootfs rocdev/rootfs.img
sudo ./upgrade_tool di -userdata rockdev/userdata.img
sudo ./upgrade_tool rd
```

Or upgrade the packaged complete firmware:

```
sudo ./upgrade_tool uf rockdev/update.img
```

Or in the root directory, run the following command to upgrade when the device is in MASKROM mode:

```
./rkflash.sh
```

8.1.3 System Partition Introduction

Default partition description is showed as follows: (below is the RK3562 EVB partition reference)

Number	Start (sector)	End (sector)	Size	Name
1	16384	24575	4M	uboot
2	24576	32767	4M	misc
3	32768	163839	64M	boot
4	163840	294911	32M	recovery
5	294912	360447	32M	bakcup
6	360448	12943359	6144M	rootfs
7	12943360	12943359	128M	oem
8	13205504	61120478	22.8G	userdata

- uboot partition: uboot.img compiled by uboot.
- misc partition: for misc.img, used by recovery.
- boot partition: boot.img compiled by kernel.
- recovery partition: recovery.img compiled by recovery.
- backup partition: reserved, temporarily unused.
- rootfs partition: rootfs.img compiled by Buildroot, Debian or Yocto.
- oem partition: used by manufacturers to store their APP or data. Mounted in the /oem directory.
- userdata partition: used for APP to temporarily generate files or for end users, mounted in the /userdata directory.

9. Chapter-9 SDK Development

This chapter mainly introduces the development of some core modules in SDK development, such as U-Boot, Kernel, Recovery, Buildroot, Debian, Yocto, Multivideo, Graphics

9.1 U-Boot Development

This section briefly introduces the basic concepts of U-Boot and compilation precautions to help customers understand the U-Boot framework of the RK platform. For detailed U-Boot development details, please refer to the document `Rockchip-Developer-Guide-UBoot-*.pdf` in the `<SDK>/docs/cn/Common/U-Boot` directory.

9.1.1 U-Boot Introduction

The v2017 (next-dev) is a version developed by Rockchip from the official v2017.09 official version of U-Boot. It currently supports all mainstream chips on sale in Rockchip. The supported functions mainly include:

- Support RK Android firmware booting;
- Support Android AOSP firmware booting;
- Support Linux Distro firmware booting;
- Supports Rockchip miniloader and SPL/TPL pre-loader boot;
- Supports LVDS, EDP, MIPI, HDMI, CVBS, RGB and other display devices;
- Supports booting from storage devices such as eMMC, Nand Flash, SPI Nand Flash, SPI NOR Flash, SD card, U disk, etc.;
- Support FAT, EXT2, EXT4 file systems;
- Support GPT, Rockchip parameter partition table;
- Supports boot LOGO, charging animation, low battery management, and power management;
- Supports I2C, PMIC, CHARGE, FUEL GUAGE, USB, GPIO, PWM, GMAC, eMMC, NAND, Interrupt, etc.;
- Support Vendor storage to save user data and configuration;
- Supports RockUSB and Google Fastboot USB gadgets to burn eMMC;
- Supports USB devices such as Mass storage, ethernet, HID, etc.;
- Support dynamic selection of kernel DTB based on hardware status;

9.1.2 Version

There are two versions of RK U-Boot: the old version v2014 and the new version v2017. The internal names are rkdevelop and next-dev respectively. Users have two ways to confirm whether the current U-Boot is version v2017.

Method 1: Confirm whether the version number of Makefile in the root directory is 2017.

```
#
### Chapter-1 SPDX-License-Identifier:      GPL-2.0+
#

VERSION = 2017
PATCHLEVEL = 09
SUBLEVEL =
EXTRAVERSION =
NAME =
......
```

Method 2: Confirm whether the first line officially printed on boot is U-Boot 2017.09.

```
U-Boot 2017.09-01818-g11818ff-dirty (Nov 14 2019 - 11:11:47 +0800)
......
```

> Open source project : v2017 has been open source and regularly updated to Github: https://github.com/rockchip-linux/u-boot
>
> Kernel version: v2017 requires RK kernel version >= 4.4

9.1.3 Preparation in Advance

- Download rkbin

  This is a toolkit repository used to store bins, scripts, and packaging tools of RK that are not open source. When U-Boot is compiled, it will index relevant files from this repository and package them to generate loader, trust, and uboot firmware. The rkbin and U-Boot projects should keep in the a same level of directory.

  > Repository downloading: please refer to the appendix chapter.

- Download GCC

  The GCC compiler uses gcc-linaro-6.3.1 and is placed in the prebuilts directory. prebuilts and U-Boot should keep in the a same level of directory. as follows:

  ```
  // 32 bit:
  prebuilts/gcc/linux-x86/arm/gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf
  // 64 bit:
  prebuilts/gcc/linux-x86/aarch64/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/
  ```

> GCC downloading: please refer to the appendix chapter

- Select defconfig

Defconfig support status of each platform (subject to release SDK):

> "[Chip]_defconfig" or "[Chip].config" are usually full-featured versions, and the rest are Version of specific features.

Chip	defconfig	Supported kernel dtb	Description
RK1808	rk1808_defconfig	Y	General version
RK1806	rk1806_defconfig	Y	General version
RK3036	rk3036_defconfig	Y	General version
RK3128x	rk3128x_defconfig	Y	General version
RK3126	rk3126_defconfig	Y	General version
RK322x	rk322x_defconfig	Y	General version
RK3288	rk3288_defconfig	Y	General version
RK3368	rk3368_defconfig	Y	General version
RK3328	rk3328_defconfig	Y	General version
RK3399	rk3399_defconfig	Y	General version
RK3399Pro	rk3399pro_defconfig	Y	General version
RK3399Pro-npu	rknpu-lion_defconfig	Y	General version
RK3308	rk3308_defconfig rk3308-aarch32_defconfig	Y	General version Support aarch32 mode
PX30	px30_defconfig	Y	General version
RK3326	rk3326_defconfig rk3326-aarch32_defconfig	Y	General version Support aarch32 mode
RK3568	rk3568_defconfig rk3568-dfu.config rk3568-nand.config rk3568-spl-spi-nand_defconfig rk3568-aarch32.config rk3568-usbplug. config		
RK3566	rk3566.config rk3566-eink.config	Y	General version E-book version
RK3588	rk3588_defconfig rk3588-ramboot.config rk3588-sata.config rk3588-aarch32.config rk3588-ipc.config	Y	General version None Storage device (memory boot) Dual storage supports sata boot Supports aarch32 mode Used in ipc sdk
RK3528	rk3528_defconfig	Y	General version
RK3562	rk3562_defconfig	Y	General version

> Note: If the table is different from the defconfig released by the SDK, please refer to the SDK.

9.1.4 Start the process

The U-Boot startup process of the RK platform is as follows, below are only some important steps listed:

```
start.s
    // Assembly environment
    => IRQ/FIQ/lowlevel/vbar/errata/cp15/gic   // ARM architecture related
lowlevel initialization
    => _main
        => stack                               // Prepare the stack required for
the C environment
        // 【Phase 1】 C environment initialization, initiating a series of
function calls
        => board_init_f: init_sequence_f[]
            initf_malloc
            arch_cpu_init                      // 【SoC lowlevel initialization】
            serial_init                        // Serial port initialization
            dram_init                          // 【Get ddr capacity information】
            reserve_mmu                        // Reserve memory from the end of
ddr to lower addresses
            reserve_video
            reserve_uboot
            reserve_malloc
            reserve_global_data
            reserve_fdt
            reserve_stacks
            dram_init_banksize
            sysmem_init
            setup_reloc                        // Determine the address to be
relocated by U-Boot itself
        // Assembly environment
        => relocate_code                       // Implements U-Boot code
relocation by assembly
        // 【Phase 2】 C environment is initialized and a series of function calls
are initiated.
        => board_init_r: init_sequence_r[]
            initr_caches                       // Enable MMU and I/Dcache
            initr_malloc
            bidram_initr
            sysmem_initr
            initr_of_live                      // Initialize of_live
            initr_dm                           // Initialize dm framework
            board_init                         // 【Platform initialization, this
is the core part】
                board_debug_uart_init          // Serial port iomux, clk
configuration
                init_kernel_dtb                // 【switch to kernel dtb】!
                clks_probe                     // Initialize system frequency
                regulators_enable_boot_on      // Initialize system power
                io_domain_init                 // Initialize io-domain
                set_armclk_rate                // __weak, ARM frequency increase
(implemented only when the platform requires it)
                dvfs_init                      // Frequency and voltage
regulation of wide temperature chip
```

```
                rk_board_init                   // __weak, implemented by each
chip platform
            console_init_r
            board_late_init                     // 【Initialize Platform late】
                rockchip_set_ethaddr            // Set mac address
                rockchip_set_serialno           // Set serialno
                setup_boot_mode                 // Parse "reboot xxx" command、
                                                // Identify buttons and loader
burning mode, recovery
                charge_display                  // U-Boot charging
                rockchip_show_logo              // Show boot logo
                soc_clk_dump                    // print clk tree
                rk_board_late_init              // __weak, implemented by each
chip platform
            run_main_loop                       // 【Enter command line mode, or
execute the boot command】
```

9.1.5 Shortcut Keys

RK platform provides serial port key combinations to trigger some events for debugging and burning (if it cannot be triggered, please try a few more times; it will not work when secure-boot is enabled). **Long press when powering on**:

- ctrl+c: Enter U-Boot command line mode;
- ctrl+d: Enter loader burning mode;
- ctrl+b: Enter maskrom burning mode;
- ctrl+f: enter fastboot mode;
- ctrl+m: print bidram/system information;
- ctrl+i: enable kernel initcall_debug;
- ctrl+p: print cmdline information;
- ctrl+s: after "Starting kernel...", enter the U-Boot command line;

9.2 Kernel Development

This section briefly introduces some common configuration modifications of the kernel, mainly the configuration of DTS, to help customers make some simple modifications faster and more conveniently. The Kernel version is based on 4.4 and will be introduced accordingly.

9.2.1 DTS Introduction

9.2.1.1 DTS Overview

Device Tree Source (DTS) files are a data structure used to describe hardware devices, defining and configuring hardware devices in the Linux kernel. These files utilize a text-like format, representing hardware devices and their relationships through a hierarchical structure and attribute descriptions.

Device Tree Source files are built into Device Tree Blob (DTB) files, which the kernel loads and parses during startup. The kernel extracts hardware device information from these files for initialization and configuration.

The introduction of Device Tree addresses solves the inflexibility and maintainability issues of traditional "hard-coded" methods when dealing with different hardware configurations. By abstracting hardware descriptions and configurations, the kernel can adapt to various hardware platforms without modifying the kernel code. In this way, the same kernel can run on multiple different hardware devices, requiring only the loading of the appropriate device tree.

This chapter aims to introduce how to add a new board DTS configuration and provides some common syntax introduction. For more detailed information about DTS, please refer to devicetree-specifications and devicetree-bindings.

9.2.1.2 Add a DTS of a new product

- Create dts file

Currently, Linux Kernel supports using dts on multiple platforms. The dts files of RK platform are stored in:

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

The general naming rule for dts files is "soc-board-name.dts", such as rk3399-evb-ind-lpddr4-linux.dts.
soc refers to the chip model, and board_name is usually named according to the silk of the board.
If your board is an integrated board, you only need a dts file to describe it.

If the hardware is designed with a core board and base board, or the product has multiple product forms, you can put the common hardware description in the dtsi file, the dts file describes different hardware modules, and includes common hardware descriptions through include "xxx.dtsi".

```
├──rk3399-evb-ind-lpddr4-linux.dts
│ ├── rk3399-evb-ind.dtsi
│ └── rk3399-linux.dtsi
```

- Modify the Makefile in the directory where dts is located

```
--- a/arch/arm64/boot/dts/rockchip/Makefile
+++ b/arch/arm64/boot/dts/rockchip/Makefile
@@ -50,6 +50,7 @@ dtb-$(CONFIG_ARCH_ROCKCHIP) += rk3368-tablet.dtb
 dtb-$(CONFIG_ARCH_ROCKCHIP) += rk3399-evb.dtb
 dtb-$(CONFIG_ARCH_ROCKCHIP) += rk3399-evb-ind-lpddr4-android.dtb
 dtb-$(CONFIG_ARCH_ROCKCHIP) += rk3399-evb-ind-lpddr4-android-avb.dtb
+dtb-$(CONFIG_ARCH_ROCKCHIP) += rk3399-evb-ind-lpddr4-linux.dtb
```

When building Kenrel, you can directly `make dts-name.img` (such as rk3399-evb-ind-lpddr4-linux.img) to generate the corresponding boot.img (including dtb data).

- Several description on dts syntax

The dts syntax just like c/c++, by #include xxx.dtsi to include other public dts data. The dts file will inherit all the device nodes' properties and values included in the dtsi files. If a property is defined in multiple dts/dtsi files, its value is ultimately the definition of dts. All controller nodes related to the chip will be defined in soc.dtsi. If you need to enable the device function, you need to set its status to "okay" in the dts file. To shut down the device, you need to set its status to "disabled" in the dts file.

```
/dts-v1/;
```

```
#include "rk3399-evb-ind.dtsi"
#include "rk3399-linux.dtsi"
...
&i2s2 {
        #sound-dai-cells = <0>;
        status = "okay";
};

&hdmi_sound {
        status = "okay";
};
```

9.2.2 Kernel Module Development Documentation

The corresponding development documents are released for functional modules under the `<SDK>/docs/cn/Common/` directory. This section mainly make a summary index of these development documents. Based on the issues encountered in actual development, refer to the following table to read and learn the corresponding development guide. It can be obtained under `docs/cn/Common` and will be continuously improved and updated. For more details, please refer to Documentation Description.

9.2.3 GPIO

For example, RK3399/RK3399Pro provides 5 groups of GPIOs (GPIO0~GPIO4), a total of 122. All GPIOs can be used as interrupts. GPIO0/GPIO1 can be used as system wake-up pins. All GPIOs can be configured as pull-up or pull-down by software. All GPIOs are default for input, GPIO drive capabilities are software configurable. About the correspondence relationship between the GPIO on the schematic diagram and the GPIO in the dts, for example, GPIO4C0, then the corresponding dts should be "gpio4 16". Because GPIO4A has 8 pins and GPIO4B also has 8 pins, so the c0 port is 16, the c1 port is 17, and so on; for the use of GPIO, please refer to `<SDK>/docs/cn/Common/PinCtrl/ Rockchip_Developer_Guide_Linux_Pinctrl_CN.pdf` for details.

9.2.4 CPU, GPU, DDR Frequency Modification

Dynamic Voltage and Frequency Scaling(DVFS) is a real-time voltage and frequency adjustment technology. Currently, the modules supporting DVFS in the 4.4 kernel include CPU, GPU, and DDR. CPUFreq is a set of framework models defined by kernel developers that support dynamic scaling of CPU frequency and voltage. It can effectively reduce CPU power consumption while taking into account CPU performance. CPUFreq selects a suitable frequency for CPU to use through different frequency conversion strategies. The current kernel version provides the following strategies:

- interactive: Dynamically adjust frequency and voltage according to CPU load;
- conservative: Conservative strategy, adjust frequency and voltage step by step;
- ondemand: Dynamic frequency and voltage adjustment based on CPU load, slower response than interactive strategy;
- userspace: User set the voltage and frequency by himself, and the system will not automatically adjust;
- powersave: power consumption priority and always set the frequency to the lowest value;
- performance: Performance priority, always set the frequency to the highest value;

For detailed module functions and configuration, please refer to the documents in the `<SDK>/docs/cn/Common/DVFS/` directory. ARM/GPU/DDR have corresponding debugging interfaces respectively, which can be operated through ADB commands. The corresponding interface directories are as follows:

```
CPU small core: /sys/devices/system/cpu/cpu0/cpufreq/
CPU big:  /sys/devices/system/cpu/cpu4/cpufreq/
GPU: /sys/class/devfreq/ff9a0000.gpu/
DDR:  /sys/class/devfreq/dmc/
```

There are nodes similar to the following in these directories:

- available_frequencies: displays supported frequencies
- available_governors: displays supported frequency conversion strategies
- cur_freq: displays the current frequency
- governor: displays the current frequency conversion strategy
- max_freq: Displays the current highest running frequency
- min_freq: Displays the current lowest running frequency

Taking RK3399/RK3399pro GPU as an example to perform fixed frequency operation, the process is as follows:

Check which frequencies are supported:

```
cat /sys/class/devfreq/ff9a0000.gpu/available_frequencies
```

Switch frequency conversion strategy:

```
echo userspace > /sys/class/devfreq/ff9a0000.gpu/governor
```

Fixed frequency:

```
echo 400000000 > /sys/class/devfreq/ff9a0000.gpu/userspace/set_freq
cat /sys/class/devfreq/ff9a0000.gpu/cur_freq
```

9.2.5 Temperature Control Configuration

There are temperature control sensors in the ARM core and GPU core of RK3399/RK3399Pro chips respectively, which can monitor the temperature of CPU and GPU in real time, and control the frequencies of CPU and GPU through algorithms to control the temperatures of CPU and GPU. The heat dissipation conditions corresponding to different hardware designs and molds of each product are also different. The temperature control parameters can be appropriately adjusted through the following configuration in dts to adapt to the product.

Set the temperature at which the temperature control is turned on:

```
&threshold {
    temperature = ; /* millicelsius */
};
```

Set the upper limit of temperature control

```
&target {
    temperature = ; /* millicelsius */
};
```

Set software shutdown temperature:

```
&soc_crit {
    temperature = ; /* millicelsius */
};
```

Configure hardware shutdown temperature:

```
&tsadc {
    rockchip,hw-tshut-mode = ; /* tshut mode 0:CRU 1:GPIO */
    rockchip,hw-tshut-polarity = ; /* tshut polarity 0:LOW 1:HIGH */
    rockchip,hw-tshut-temp = ;
    status = "okay";
};
```

For detailed instructions on temperature control, please refer to the relevant documents in the `<SDK>/docs/cn/Common/THERMAL` directory.

9.2.6 LPDDR4 Configuration

For DDR configuration instructions, please refer to the relevant documents in the `<SDK>/docs/cn/Common/DDR` directory.

For the dts configuration of RK3399Pro using LPDDR4, please refer to the file: `arch/arm64/boot/dts/rockchip/rk3399pro-evb-lp4-v11-avb.dts`. Just copy the following 3 nodes in the file to the corresponding product dts:

```
&dfi {
    status = "okay";
};
&dmc {
    status = "okay";
    center-supply = <&vdd_center>;//Customers need to configure according to the
actual hardware circuit here
    upthreshold = <40>;
    downdifferential = <20>;
    system-status-freq = <
        /*system status freq(KHz)*/
        SYS_STATUS_NORMAL 856000
        SYS_STATUS_REBOOT 416000
        SYS_STATUS_SUSPEND 416000
        SYS_STATUS_VIDEO_1080P 416000
        SYS_STATUS_VIDEO_4K 856000
        SYS_STATUS_VIDEO_4K_10B 856000
        SYS_STATUS_PERFORMANCE 856000
        SYS_STATUS_BOOST 856000
        SYS_STATUS_DUALVIEW 856000
        SYS_STATUS_ISP 856000
    >;
```

```
    vop-pn-msch-readlatency = <
        /* plane_number readlatency */
        0 0
        4 0x20
    >;
    vop-bw-dmc-freq = <
        /* min_bw(MB/s) max_bw(MB/s) freq(KHz) */
        763 1893 416000
        3013 99999 856000 \
    >;
    auto-min-freq = <0>;
};

&dmc_opp_table {
    compatible = "operating-points-v2";
    opp-200000000 {
        opp-hz = /bits/ 64 ;
        opp-microvolt = <900000> ;
        status = "disabled";
    };
    opp-300000000 {
        opp-hz = /bits/ 64 ;
        opp-microvolt = <900000> ;
        status = "disabled";
    };
    opp-400000000 {
        opp-hz = /bits/ 64 ;
        opp-microvolt = <900000> ;
        status = "disabled";
    };
    opp-416000000 {
        opp-hz = /bits/ 64 ;
        opp-microvolt = <900000> ;
    };
    opp-528000000 {
        opp-hz = /bits/ 64 ;
        opp-microvolt = <900000> ;
        status = "disabled";
    };
    opp-600000000 {
        opp-hz = /bits/ 64 ;
        opp-microvolt = <900000>;
        status = "disabled";
    };
    opp-800000000 {
        opp-hz = /bits/ 64 ;
        opp-microvolt = <900000>;
        status = "disabled";
    };
    opp-856000000 {
        opp-hz = /bits/ 64 ;
        opp-microvolt = <900000>;
    };
    opp-928000000 {
        opp-hz = /bits/ 64 ;
        opp-microvolt = <900000>;
        status = "disabled";
    };
```

```
    opp-1056000000 {
        opp-hz = /bits/ 64 ;
        opp-microvolt = <900000>;
        status = "disabled";
    };
};
```

What needs to be noted here is:

For LPDDR4, only the two frequencies: 416M and 856M are supported. Other frequencies have been disabled, so if customers want to use the same dts to support LPDDR4 and other types of DDR will only be able to support 416M and 856M frequencies. At this time please be sure to configure the DDR frequency conversion function to be enabled by default. The frequency conversion function of LPDDR4 limits the number of sound cards for the following reasons:

- If LPDDR4 requires frequency conversion function, the audio buffer needs to be moved to sram. The sram memory of RK3399Pro is limited, and the available memory is 128k. Currently, the memory pre-allocated to a single audio stream is 32k, so the maximum number of sound cards supported by the system is only 2 (32k*2*2, each sound card includes playback and recording). More sound cards cannot be created successfully unless the pre-allocated memory size for a single audio stream is reduced. but this also relatively reduces the maximum buffer size supported underlying. If the user layer uses a sound card and wants to set a larger buffer, it will be limited. Please note that USB sound cards are not within the restriction because they do not use dma. In other words, you can have 2 sound cards (including sound cards with HDMI, spdif, i2s and other interfaces) and multiple USB sound cards.
- If LPDDR4 frequency conversion is required, the audio buffer needs to be moved to sram. At this time, the system can only support up to 2 sound cards.

Please configure as follows:

Add sram node in dts

```
/* first 64k(0xff8c0000~0xff8d0000) for ddr and suspend */
iram: sram@ff8d0000 {
    compatible = "mmio-sram";
    reg = ; /* 128k */
};
```

Reference the iram node in the corresponding product dts.

```
&dmac_bus {
    iram = <&iram>;
    rockchip,force-iram;
};
```

- If you do not need LPDDR4 frequency conversion, since LPDDR4 frequency conversion is limited to 2 sound cards, if you need more than 3 sound cards, you need to turn off LPDDR4 frequency conversion, that is, disable the dmc node in the dts of the corresponding product, as shown below:

```
&dmc {
    status = "disabled";
    … …
};
```

In addition, you need to make sure that the following two configurations in the kernel are deleted:

Delete the following configuration in dts:

```
/* first 64k(0xff8c0000~0xff8d0000) for ddr and suspend */
iram: sram@ff8d0000 {
    compatible = "mmio-sram";
    reg = ; /* 128k */
};

&dmac_bus {
    iram = <&iram>;
    rockchip,force-iram;
};
```

9.2.7 SD Card Configuration

For detailed instructions on SD Card configurationl, please refer to the relevant documents in the `<SDK>/docs/cn/Common/MMC` directory.

The UART debug of some chips such as RK3326/RK3399PRO is multiplexed with SD Card. The default configuration is to turn on debug. If you want to use SD Card, the following configuration is needed:

```
&fiq_debugger {
    status = "disabled";
    pinctrl-0 = <&uart2a_xfer>;
};
&sdmmc {
    ...
    sd-uhs-sdr104;
    status = "okay";
};
```

9.3 Recovery Development

9.3.1 Introduction

The development of Recovery mechanism is similar to the development of Android's Recovery function. The main function is to erase user data and upgrade system.

Recovery mode in Linux is to add an additional Recovery partition on the device. This partition is composed of kernel+resource+ramdisk and is mainly used for upgrade operations. u-boot will determine whether the system to be booted is a Normal system or a Recovery system based on the fields stored in the misc partition. Due to the independence of the system, Recovery mode can ensure the integrity of the upgrade. That is, if the upgrade process is interrupted, such as an abnormal power off, the upgrade can still continue.

9.3.2 Debug

A common debugging method is to enable debug

Create a hidden file .rkdebug in the buildroot/output/rockchip_xxx_recovery/target directory,

```
touch .rkdebug
```

The log upgraded in Recovery mode is printed on the serial port. Another way is to check the userdata/recovery/Log file

For more Recovery development information, please refer to the documentation：
`<SDK>/docs/cn/Linux/Recovery/Rockchip_Developer_Guide_Linux_Recovery_CN.pdf`

9.4 Buildroot Development

Buildroot is a tool that uses cross-compilation to build a complete Linux system for embedded systems. The operation is simple and automatic.

Based on the native Buildroot, Rockchip has integrated BSP configuration of relevant chips, the configuration of the acceleration functions of each hardware module, and the in-depth customization development of third-party packages to facilitate customers to deeply customize and secondary develop products.

For more Buildroot development information, please refer to the documentation：

`<SDK>/docs/cn/Linux/System/Rockchip_Developer_Guide_Buildroot_CN.pdf`

9.5 Debian Development

Rockchip provides support for Debian 10/11 based on X11 display architecture and is based on the Linaro version. The support also includes graphics and video acceleration functions. Related software packages include libmali, xserver, gstreamer rockchip, etc. These software packages can be built by setting up an environment in Docker, and the generated deb packages are stored in the `<SDK>/debian/packages/*` directory.

For detailed steps on how to use Docker to build the deb packages, please refer to the documentation:

`<SDK>/docs/cn/Linux/Docker/Rockchip_Developer_Guide_Debian_Docker_CN.pdf`

Debian development documentation reference:

`<SDK>/docs/cn/Linux/System/Rockchip_Developer_Guide_Debian_CN.pdf`

9.6 Yocto Development

For more information, please refer to: http://opensource.rock-chips.com/wiki_Yocto

9.7 Audio Development

9.7.1 Kernel Audio Driver Development

The kernel driver uses `soc/rockchip/rockchip_multicodecs.c` instead of `soc/generic/simple-card.c`, In other words, the kernel-driven simple card is just a sample and cannot meet the requirement of some complex products.

The difference between rockchip_multicodecs driver and simple-card is as follows:

- Mainly compatible with one sound card and one or more codec support.
- Headphones and other tests support adc. Currently, gpio is commonly used in simple-card.
- The i2stdm controller is divided into tx/rx, multicodecs supports tx/rx independently, and adding it to simple-card will affect third parties.

9.7.2 Audio Pulseaudio Channel Adaptation

Audio uses pulseaudio by default. Normally, you only need to configure `/etc/pulse/default.pa`

For example, the two Codecs of ES8388 and RK809 is adapted in RK3588.

```
+set-default-source alsa_input.platform-es8388-
sound.HiFi__hw_rockchipes8388__source
+set-default-sink alsa_output.platform-es8388-sound.HiFi__hw_rockchipes8388__sink
+set-default-source alsa_input.platform-rk809-
sound.HiFi__hw_rockchiprk809__source
+set-default-sink alsa_output.platform-rk809-sound.HiFi__hw_rockchiprk809__sink
```

If you need to add more Codec support, obtain relevant information through the following command:

```
pactl list sinks short
pactl list sources short
```

For more reference, please refer to Debian official Pulseaduio and
`<SDK>/docs/Common/AUDIO/Rockchip_Developer_Guide_PulseAudio_CN.pdf`

9.8 Multimedia Development

GStreamer or rockit is used to develop multimedia on the rockchip platform.

```
vpu_service --> mpp --> gstreamer/rockit --> app

vpu_service: driver
mpp: Video encoding and decoding middleware for the rockchip platform. For
related instructions, please refer to the mpp document.
gstreamer/rockit: components for docking apps
```

Currently, the complete solution provided by Rockchip Linux Universal SDK is based on GStreamer. The advantage of using GStreamer is that it can more conveniently complete applications such as players and encoders based on pipelines. If you need customized development based on rockit, please refer to the relevant release documents of rockit.

Reference documents:

```
<SDK>/docs/cn/Linux/Multimedia
├── Rockchip_Developer_Guide_Linux_RKADK_CN.pdf
├── Rockchip_User_Guide_Linux_Gstreamer_CN.pdf
└── Rockchip_User_Guide_Linux_Rockit_CN.pdf
```

9.9 Grahpics Development

Graphics of Rockchip Linux Platform is an ARM Linux platform that utilizes DRM and DMA BUF. The advantage is that with a universal architecture, customized development based on this architecture is easier, and many existing components can be utilized. The development of many existing open source projects has started to use the Rockchip platform as an adaptation platform for ARM. But the disadvantage is that many people do not fully understand these contents, and practical application requires a learning process. For more information, please refer to the Rockchip wiki and the following documents.

```
<SDK>/docs/cn/Linux/Graphics/
├── Rockchip_Developer_Guide_Buildroot_Weston_CN.pdf
├── Rockchip_Developer_Guide_Linux_Graphics_CN.pdf
```

9.10 Application Development

Commonly used applications of SDK include Weston, EFL, ROS and other application development. Please refer to the documents in the /docs/cn/Linux/Graphics/ApplicationNote directory.

9.11 Security Mechanism Development

Please refer to the documentation in the `<SDK>/docs/cn/Linux/Security` directory

9.12 Secureboot

Secureboot is not enabled by default. If verification is required, the steps are as follows:

9.12.1 Enables Security Configuration in the SDK

$make menuconfig
Enable Security ---> [*] security feature

9.12.2 Saves Security Configuration in the SDK

$ make savedefconfig

Take RK3568 as an example:

```
--- a/.chips/rk3566_rk3568/rockchip_rk3568_evb1_ddr4_v10_defconfig
+++ b/.chips/rk3566_rk3568/rockchip_rk3568_evb1_ddr4_v10_defconfig
@@ -3,3 +3,4 @@ RK_KERNEL_DTS_NAME="rk3568-evb1-ddr4-v10-linux"
 RK_USE_FIT_IMG=y
 RK_WIFIBT_TTY="ttyS8"
 RK_PARAMETER="parameter-buildroot-fit.txt"
+RK_SECURITY=y
```

9.12.3 Build Secureboot

- Directly run `./build.sh` and follow the compilation error instructions and operate step by step:

$ ./build.sh

```
ERROR: No root keys(u-boot/keys) found in u-boot
       Create it by ./build.sh createkeys or move your key to it

$ ./build.sh createkeys

$ ./build.sh

============================================
ERROR: No root passwd(u-boot/keys/root_passwd) found in u-boot
       echo your root key for sudo to u-boot/keys/root_passwd

Write your PC root password into `u-boot/keys/root_passwd`

$ ./build.sh
Security: No found config CONFIG_BLK_DEV_DM in
kernel/arch/arm64/configs/rockchip_linux_defconfig
make sure your config include this list
---------------------------------------

    CONFIG_BLK_DEV_DM
    CONFIG_DM_CRYPT
    CONFIG_BLK_DEV_CRYPTOLOOP
    CONFIG_DM_VERITY
    CONFIG_TEE
    CONFIG_OPTEE

Add related configuration to kernel

kernel$ git diff
diff --git a/arch/arm64/configs/rockchip_linux_defconfig
b/arch/arm64/configs/rockchip_linux_defconfig
index d8757f713ec4..7beca18172e0 100644
--- a/arch/arm64/configs/rockchip_linux_defconfig
+++ b/arch/arm64/configs/rockchip_linux_defconfig
@@ -590,3 +590,9 @@ CONFIG_RCU_CPU_STALL_TIMEOUT=60
 CONFIG_FUNCTION_TRACER=y
 CONFIG_BLK_DEV_IO_TRACE=y
 CONFIG_LKDTM=y
+CONFIG_BLK_DEV_DM=y
+CONFIG_DM_CRYPT=y
+CONFIG_BLK_DEV_CRYPTOLOOP=y
+CONFIG_DM_VERITY=y
+CONFIG_TEE=y
+CONFIG_OPTEE=y

Security: No found config CONFIG_FIT_SIGNATURE in u-boot/configs/rk3568_defconfig
make sure your config include this list
---------------------------------------

    CONFIG_FIT_SIGNATURE
    CONFIG_SPL_FIT_SIGNATURE
```

```
u-boot adds related configuration:

u-boot$ git diff
diff --git a/configs/rk3568_defconfig b/configs/rk3568_defconfig
index fbd9820acc..6efdaac1d6 100644
--- a/configs/rk3568_defconfig
+++ b/configs/rk3568_defconfig
@@ -220,3 +220,5 @@ CONFIG_RK_AVB_LIBAVB_USER=y
 CONFIG_OPTEE_CLIENT=y
 CONFIG_OPTEE_V2=y
 CONFIG_OPTEE_ALWAYS_USE_SECURITY_PARTITION=y
+CONFIG_FIT_SIGNATURE=y
+CONFIG_SPL_FIT_SIGNATURE=y

Security: No found config BR2_PACKAGE_RECOVERY in
buildroot/configsrockchip_rk3566_rk3568_ramboot_defconfig
Security: No found config BR2_PACKAGE_RECOVERY_UPDATEENGINEBIN in
buildroot/configs/rockchip_rk3566_rk3568_ramboot_defconfig

buildroot adds related configuration:

--- a/configs/rockchip_rk3566_rk3568_ramboot_defconfig
+++ b/configs/rockchip_rk3566_rk3568_ramboot_defconfig
@@ -4,6 +4,8 @@
 BR2_PACKAGE_LUKSMETA=y
+BR2_PACKAGE_RECOVERY=y
+BR2_PACKAGE_RECOVERY_UPDATEENGINEBIN=y

Security: No found string BR2_ROOTFS_OVERLAY=".*board/rockchip/common/security-
system-overlay.* in buildroot/configs/rockchip_rk3568_defconfig
ERROR: Running check_security_condition failed!
ERROR: exit code 255 from line 982:

buildroot adds related configuration:

buildroot$ git diff
diff --git a/configs/rockchip_rk3568_defconfig
b/configs/rockchip_rk3568_defconfig
index 6cdd795c64..5244694c7d 100644
--- a/configs/rockchip_rk3568_defconfig
+++ b/configs/rockchip_rk3568_defconfig
@@ -23,3 +23,4 @@
 BR2_PACKAGE_RKWIFIBT_AP6398S=y
 BR2_PACKAGE_RKWIFIBT_BTUART="ttyS8"
 BR2_PACKAGE_RKNPU2=y
+BR2_ROOTFS_OVERLAY="board/rockchip/common/security-system-overlay"

Environment installation:

If the compilation error is `No module named Crypto.Signature`, this is because
the computer used for development does not have a Python algorithm library
installed. Simply execute the following command:
pip uninstall Crypto
pip uninstall pycrypto
pip install pycrypto
2. If the following error occurs: ModuleNotFoundError: No module named
'Cryptodome'
```

```
Please install the python package on the development host: pip3 install [--user]
pycryptodomex
```

- When verifying, find a device that has not burned rpmb. If there is an encryption key in the device, it will be used first. If the device encryption key is different from the encryption key we compiled, the startup will fail.
- The encryption key is placed in the rpmb area and can be rewritten. If the key is different and the system cannot be loaded, you can turn on FORCE_KEY_WRITE=true in `buildroot/board/rockchip/common/security-ramdisk-overlay/init.in` to force the key to be updated.

For details, please refer to

- Linux secureboot software developer guid: `<SDK>/docs/cn/Linux/Security/Rockchip_Developer_Guide_Linux_Secure_Boot_CN.pdf`
- TEE developer guid: `<SDK>/docs/cn/Linux/Security/Rockchip_Developer_Guide_TEE_SDK_CN`

9.13 WIFI/BT Development

Please refer to the document in the directory `<SDK>/docs/cn/Linux/Wifibt`

9.14 SDK Booting Way

Currently, there are three booting ways provided in the Linux SDK: Sysv, Busybox, and Systemd init.

Yocto system uses Sysv init by default to manage booting, and `update-rc.d.bbclass` is used to manage the booting configuration of services.

Buildroot system uses the Busybox init way to manage booting by default. Busybox init will read the inittab file in the /etc/ directory after booting.

Debian system uses Systemd way to manage booting by default. Systemd init will read related services in the `/etc/systemd/system/` directory after booting.

SDK has some different processing for different booting services. For example:

```
<SDK>/device/rockchip/common/scripts/post-disk.sh

if [ "$POST_INIT_BUSYBOX" ]; then
        mkdir -p "$TARGET_DIR/etc/init.d"
        install -m 0755 external/rkscript/$SCRIPT "$TARGET_DIR/etc/init.d/"
fi

if [ "$POST_INIT_SYSTEMD" ]; then
        mkdir -p "$TARGET_DIR/lib/systemd/system"
        install -m 0755 external/rkscript/$DISK_HELPER_TYPE-all.service \
                "$TARGET_DIR/lib/systemd/system/"
        mkdir -p "$TARGET_DIR/etc/systemd/system/sysinit.target.wants"
        ln -sf /lib/systemd/system/$DISK_HELPER_TYPE-all.service \
                "$TARGET_DIR/etc/systemd/system/sysinit.target.wants/"
fi

if [ "$POST_INIT_SYSV" ]; then
```

```
        mkdir -p "$TARGET_DIR/etc/init.d"
        install -m 0755 external/rkscript/$SCRIPT \
                "$TARGET_DIR/etc/init.d/${DISK_HELPER_TYPE}all.sh"
        mkdir -p "$TARGET_DIR/etc/rcS.d"
        ln -sf ../init.d/${DISK_HELPER_TYPE}all.sh \
                "$TARGET_DIR/etc/rcS.d/$SCRIPT"
fi
```

9.15 SDK Test

9.15.1 Integrated Rockchip Stress Test Script

`rockchip_test` integrates functional, stress, and performance related tests

```
       ROCKCHIPS TEST TOOLS

ddr test:              1 (ddr stress test)
cpu test:              2 (cpu stress test)
gpu test:              3 (gpu stress test)
npu test:              4 (npu stress test)
suspend_resume test:   5 (suspend resume)
reboot test:           6 (auto reboot test)
power lost test:       7 (power lost test)
flash stress test:     8 (flash stress test)
recovery test:         9 (recovery wipe all test)
audio test:           10 (audio test)
camera test:          11 (camera test)
video test:           12 (video test)
bluetooth test:       13 (bluetooth on off test)
wifi test:            14 (wifi on off test)
chromium test:        15 (chromium with video test)
**************************************************

please input your test moudle:
```

9.15.2 Benchmark Test

The following are reference data for some commonly used benchmark tests. You can find more information in the following test documents:

```
<SDK>/docs/cn/Linux/Profile/Rockchip_Introduction_Linux_Benchmark_KPI_CN.pdf
```

9.15.3 Rockchip Modules and Stress Testing

Some common module functions and stress testing methods are provided below. You can find more detailed information in the following documents:

```
<SDK>/docs/cn/Linux/Profile/Rockchip_User_Guide_Linux_Software_Test_CN.pdf
```

10. Chapter-10 SDK System Debugging Tools Introduction

There are some commonly used debugging tools for static compilation are pre-installed in the SDK , as follows:

```
device/rockchip/common/tools/
├── adbd
├── busybox
├── gdb
├── io
├── kmsgrab
├── modetest
├── perf-4.19
├── perf-4.4
├── perf-5.10
├── pmap
├── procrank
├── ps
├── slabtop
├── strace
├── top
├── update
├── vendor_storage
├── vmstat
└── watch
```

Just `make menuconfig` in the SDK directory and select the relevant configuration.

```
  | Symbol: RK_ROOTFS_PREBUILT_TOOLS [=y]
                                                |
  | Type  : bool
                                                 |
  | Prompt: prebuilt tools
                                                 |
  |   Location:
                                                |
  |     -> Rootfs
                                                |
  |       -> Post rootfs installs
                                                |
  |   Defined at Config.in.post-rootfs:263
```

Save the configuration, for example, the detailed modifications of RK3588 EVB1 are as follows:

```
device/rockchip/rk3588/rockchip_rk3588_evb1_lp4_v10_defconfig
@@ -1,4 +1,7 @@
 RK_YOCTO_CFG="rockchip-rk3588-evb"
 RK_WIFIBT_TTY="ttyS8"
+RK_ROOTFS_PREBUILT_TOOLS=y
```

10.1 ADB Tool

You can open the following macros in Buildroot to build and obtain:

```
BR2_PACKAGE_ANDROID_TOOLS=y
BR2_PACKAGE_ANDROID_TOOLS_STATIC=y
```

For details usage of ADB tools, please refer to ADB official guidance document.

10.1.1 Overview

- Run the device's shell (command line)
- Manage port mapping for emulators or devices
- Upload/download files between computer and device
- Install local software to Debian, Buildroot and other Linux devices
- ADB is a "client-server" program, where the client mainly refers to a PC and a server is an entity device or a virtual device of Debian. Depending on how the PC connects to Debian devices, ADB can be divided into two types:
  Network ADB: The host is connected to the STB device via a wired/wireless network (same LAN)
  USB ADB: The host is connected to the STB device through a USB cable

10.1.2 USB ADB Usage Instructions

USB adb usage has the following restrictions:

- Only supports USB OTG port
- Does not support using of multiple clients at the same time
- Only supports the host to connect to one device, does not support connecting to multiple devices
  The connection steps are as follows:
  To test whether the connection is successful, run the "adb devices" command. If the serial number of the device is displayed, the connection is successful.

10.2 Busybox Tool

You can open the following macros in Buildroot to build and obtain:

```
BR2_PACKAGE_BUSYBOX=y
BR2_PACKAGE_BUSYBOX_STATIC=y
```

Busybox is similar to a Linux toolbox. It integrates and compresses many tools and commands of Linux.

```
## Chapter-10 ./busybox
BusyBox v1.36.0 (2023-05-20 11:25:39 CST) multi-call binary.
BusyBox is copyrighted by many authors between 1998-2015.
Licensed under GPLv2. See source distribution for detailed
copyright notices.

Usage: busybox [function [arguments]...]
```

```
   or: busybox --list[-full]
   or: busybox --show SCRIPT
   or: busybox --install [-s] [DIR]
   or: function [arguments]...

        BusyBox is a multi-call binary that combines many common Unix
        utilities into a single executable.  Most people will create a
        link to busybox for each function they wish to use and BusyBox
        will act like whatever it was invoked as.

Currently defined functions:
        [, [[, addgroup, adduser, ar, arch, arp, arping, ascii, ash, awk,
        base32, base64, basename, bc, blkid, bunzip2, bzcat, cat, chattr,
        chgrp, chmod, chown, chroot, chrt, chvt, cksum, clear, cmp, cp, cpio,
        crc32, crond, crontab, cut, date, dc, dd, deallocvt, delgroup, deluser,
        devmem, df, diff, dirname, dmesg, dnsd, dnsdomainname, dos2unix, du,
        dumpkmap, echo, egrep, eject, env, ether-wake, expr, factor, fallocate,
        false, fbset, fdflush, fdformat, fdisk, fgrep, find, flock, fold, free,
        freeramdisk, fsck, fsfreeze, fstrim, fuser, getopt, getty, grep,
        groups, gunzip, gzip, halt, hdparm, head, hexdump, hexedit, hostid,
        hostname, hwclock, i2cdetect, i2cdump, i2cget, i2cset, i2ctransfer, id,
        ifconfig, ifdown, ifup, inetd, init, insmod, install, ip, ipaddr,
        ipcrm, ipcs, iplink, ipneigh, iproute, iprule, iptunnel, kill, killall,
        killall5, klogd, last, less, link, linux32, linux64, linuxrc, ln,
        loadfont, loadkmap, logger, login, logname, losetup, ls, lsattr, lsmod,
        lsof, lspci, lsscsi, lsusb, lzcat, lzma, lzopcat, makedevs, md5sum,
        mdev, mesg, microcom, mim, mkdir, mkdosfs, mke2fs, mkfifo, mknod,
        mkpasswd, mkswap, mktemp, modprobe, more, mount, mountpoint, mt, mv,
        nameif, netstat, nice, nl, nohup, nologin, nproc, nslookup, nuke, od,
        openvt, partprobe, passwd, paste, patch, pidof, ping, pipe_progress,
        pivot_root, pmap, poweroff, printenv, printf, ps, pwd, rdate, readlink,
        readprofile, realpath, reboot, renice, reset, resize, resume, rm,
        rmdir, rmmod, route, run-init, run-parts, runlevel, sed, seedrng, seq,
        setarch, setconsole, setfattr, setkeycodes, setlogcons, setpriv,
        setserial, setsid, sh, sha1sum, sha256sum, sha3sum, sha512sum, shred,
        sleep, sort, start-stop-daemon, strings, stty, su, sulogin, svc, svok,
        swapoff, swapon, switch_root, sync, sysctl, syslogd, tail, tar,
        taskset, tc, tee, telnet, test, tftp, time, timeout, top, touch, tr,
        traceroute, tree, true, truncate, ts, tsort, tty, ubirename, udhcpc,
        uevent, umount, uname, uniq, unix2dos, unlink, unlzma, unlzop, unxz,
        unzip, uptime, usleep, uudecode, uuencode, vconfig, vi, vlock, w,
        watch, watchdog, wc, wget, which, who, whoami, xargs, xxd, xz, xzcat,
        yes, zcat
```

10.3 GDB Tool

You can open the following macros in Buildroot to build and obtain:

```
BR2_PACKAGE_GDB=y
BR2_PACKAGE_GDB_STATIC=y
BR2_PACKAGE_GDB_DEBUGGER=y
```

GDB, a commonly used debugging tool in Linux, is a powerful program debugger. Please refer to GDB for more usage instructions.

10.4 IO Tool

You can open the following macros in Buildroot to build and obtain:

```
BR2_PACKAGE_RKTOOLKIT=y
BR2_PACKAGE_RKTOOLKIT_STATIC=y
```

Memory can be accessed through /dev/mem, with permission to read and write chip registers.

10.5 kmsgrab Tool

In the SDK project, you can build and obtain it as follows.

```
For building kmsgrab:
1/ ./build.sh shell
2/ source ./buildroot/output/$RK_BUILDROOT_CFG/host/environment-setup
3/ $CC $RK_DATA_DIR/kmsgrab.c $(pkg-config --cflags --libs libdrm) -static -o
kmsgrab
```

Captures the KMS scan-out framebuffer associated with the specified CRTC or screen as a DRM object that can be passed to other hardware functions.

10.6 modetest Tool

Modetest is a debugging tool that comes with the libdrm source code, which can perform some basic debugging on drm.

The help information of modetest is as follows:

```
(shell)# modetest -h
usage: modetest [-cDdefMPpsCvw]
 Query options:
    -c    list connectors
    -e    list encoders
    -f    list framebuffers
    -p    list CRTCs and planes (pipes)
 Test options:
    -P <crtc_id>:<w>x<h>[+<x>+<y>][*<scale>][@<format>]    set a plane
    -s <connector_id>[,<connector_id>][@<crtc_id>]:<mode>[-<vrefresh>][@<format>]
   set a mode
    -C    test hw cursor
    -v    test vsynced page flipping
    -w <obj_id>:<prop_name>:<value>    set property
 Generic options:
    -d    drop master after mode set
    -M module    use the given driver
    -D device    use the given device
    Default is to dump all info.
```

10.7 Perf Tool

You can open the following macros in Buildroot to build and obtain:

```
For building perf:
1/ Build dynamic version firstly
2/ Enable BR2_PACKAGE_LINUX_TOOLS_PERF_STATIC
3/ Run: make linux-tools-reconfigure
```

Perf is a Linux performance analysis tool.

10.8 Pmap Tool

To see how much memory a process uses, pmap provides a memory mapping of a process. The pmap command is used to display the memory status of one or more processes. It reports the address space and memory status information of the process.

```
#pmap PID
```

10.9 ProcrankTool

You can open the following macros in Buildroot to build and obtain:

```
BR2_PACKAGE_PROCRANK_LINUX=y
BR2_PACKAGE_PROCRANK_LINUX_STATIC=y
```

Use procrank to analyze memory utilization and analyze source code. procrank is a tool that counts memory usage, including detailed parameters of VSS, PSS, PSS and USS. It is commonly used memory analysis tool.

10.10 PS Tool

The Linux PS (process status) command is used to display the status of the current process, similar to the Windows Task Manager.

There are many parameters of PS. Here we only list a few commonly used parameters and briefly introduce them.
-A: lists all processes
-w: display more information by widening the display
-au: displays more detailed information
-aux: displays all processes containing other users

```
au(x) output format:
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

USER: process owner
PID: pid
%CPU: occupied CPU usage
%MEM: occupied memory usage
VSZ: size of virtual memory occupied
```

```
RSS: memory size occupied
TTY: minor device number of tty
STAT: The status of the process:

D: Uninterruptible sleep state (usually, it is IO process)
R: Running
S: stationary state
T: pause execution
Z: Does not exist but cannot be eliminated temporarily
W: Not enough memory pages available to allocate
<: High priority itinerary
N: low priority itinerary
L: There are memory pages allocated and locked in the memory
START: The start time of the process
TIME: execution time
COMMAND: command executed
```

10.11 Slabtop Tool

Display kernel slab buffer information in real time.

Refresh the slab buffer information every ten seconds:

```
slabtop -d 10
```

10.12 Strace Tool

Strace is a Linux user space tracer that can be used for tracking and debugging.

10.13 Top Tools

The top command is a commonly used performance analysis tool under Linux. It can display the resource usage of each process in the system in real time and is often used for server-side performance analysis.

10.14 Update Tool

Enter the recovery program to update and format, then reboot into the normal system.

10.15 Vendor_storage Tool

Read and write on the Vendor partition.

```
root@rk3588-buildroot:/armhf# ./vendor_storage --help
vendor storage tool - Revision: 2.0

./vendor_storage [-r/w <vendor_id> -t <pr_type> -i <input>] [-R]
  -r           Read specify vendor_id
```

```
  -R           Read common vendor_id
  -w           Write specify vendor_id
  -t           print type
  -i           input string
  <vendor_id>  There are 16 types
               "VENDOR_SN_ID"
               "VENDOR_WIFI_MAC_ID"
               "VENDOR_LAN_MAC_ID"
               "VENDOR_BT_MAC_ID"
               "VENDOR_HDCP_14_HDMI_ID"
               "VENDOR_HDCP_14_DP_ID"
               "VENDOR_HDCP_2x_ID"
               "VENDOR_DRM_KEY_ID"
               "VENDOR_PLAYREADY_Cert_ID"
               "VENDOR_ATTENTION_KEY_ID"
               "VENDOR_PLAYREADY_ROOT_KEY_0_ID"
               "VENDOR_PLAYREADY_ROOT_KEY_1_ID"
               "VENDOR_SENSOR_CALIBRATION_ID"
               "VENODR_RESERVE_ID_14"
               "VENDOR_IMEI_ID"
               "VENDOR_CUSTOM_ID"
               And custom can define other id like
               VENDOR_CUSTOM_ID_1A (define ID = 26)
  <pr_type>    In write case, used with -i <input>
               There are 3 types
               "hex": <input> must be hex form like 0123
               "string": <input> must be ASCII string
               "file": <input> must be path to file
               Note: If use "file" and -i with read, it means save storage to
file
Examples:
  ./vendor_storage -w VENDOR_CUSTOM_ID_1A -t file -i /userdata/test.bin
                       write userdata/test.bin to storage
                       Or -t string -i test_storage
                       write string "test_storage" to storage
                       ID = 26
  ./vendor_storage -r VENDOR_CUSTOM_ID_1A -t file -i /userdata/read.bin
                       read storage(ID=26) to userdata/read.bin
                       Or -t string
                       print storage(ID=26) with ASCII string
```

10.16 Vmstat Tool

The vmstat command reports statistics on kernel threads, virtual memory, disks, hypervisor pages, traps, and processor activity.

10.17 Watch Tool

Watch can help monitor the results of a command.

```
## Chapter-10 watch
```

```
Usage:
 watch [options] command

Options:
  -b, --beep             beep if command has a non-zero exit
  -c, --color            interpret ANSI color and style sequences
  -d, --differences[=<permanent>]
                         highlight changes between updates
  -e, --errexit          exit if command has a non-zero exit
  -g, --chgexit          exit when output from command changes
  -n, --interval <secs>  seconds to wait between updates
  -p, --precise          attempt run command in precise intervals
  -t, --no-title         turn off header
  -w, --no-wrap          turn off line wrapping
  -x, --exec             pass command to exec instead of "sh -c"

 -h, --help     display this help and exit
 -v, --version  output version information and exit
```

10.18 weston Debugging Method

root@rk3288-buildroot:/# WAYLAND_DEBUG=1 weston --debug

Weston related parameters are as follows:

```
root@rk3288-buildroot:/# weston --help
Usage: weston [OPTIONS]

This is weston version 12.0.1, the Wayland reference compositor.
Weston supports multiple backends, and depending on which backend is in use
different options will be accepted.

Core options:

  --version             Print weston version
  -B, --backend=BACKEND Backend module, one of
                                drm
  --renderer=NAME       Renderer to use, one of
                                auto    Automatic selection of one of the below
renderers
                                gl      OpenGL ES
                                noop    No-op renderer for testing only
                                pixman  Pixman software renderer
  --shell=NAME  Shell to load, defaults to desktop
  -S, --socket=NAME     Name of socket to listen on
  -i, --idle-time=SECS  Idle time in seconds
  --modules             Load the comma-separated list of modules
  --log=FILE            Log to the given file
  -c, --config=FILE     Config file to load, defaults to weston.ini
  --no-config           Do not read weston.ini
  --wait-for-debugger   Raise SIGSTOP on start-up
  --debug               Enable debug extension
  -l, --logger-scopes=SCOPE
                        Specify log scopes to subscribe to.
                        Can specify multiple scopes, each followed by comma
```

```
  -f, --flight-rec-scopes=SCOPE
                        Specify log scopes to subscribe to.
                        Can specify multiple scopes, each followed by comma
  -w, --warm-up         Hold display for the first app
  -h, --help            This help message

Options for drm:

  --seat=SEAT           The seat that weston should run on, instead of the seat
defined in XDG_SEAT
  --drm-device=CARD     The DRM device to use, e.g. "card0".
  --use-pixman          Use the pixman (CPU) renderer (deprecated alias for --
renderer=pixman)
  --current-mode        Prefer current KMS mode over EDID preferred mode
  --continue-without-input      Allow the compositor to start without input
devices
```

10.19 Obtain System Log Information Automatically

The log information will be automatically obtained in the /info directory. The main log information is as follows:

```
/info/
├── clk_summary -> /sys/kernel/debug/clk/clk_summary
├── cmdline -> /proc/cmdline
├── cpuinfo -> /proc/cpuinfo
├── device-tree -> /proc/device-tree
├── diskstats -> /proc/diskstats
├── dma_buf -> /sys/kernel/debug/dma_buf
├── dri -> /sys/kernel/debug/dri
├── fstab -> /etc/fstab
├── gpio -> /sys/kernel/debug/gpio
├── interrupts -> /proc/interrupts
├── iomem -> /proc/iomem
├── kallsyms -> /proc/kallsyms
├── log -> /var/log
├── meminfo -> /proc/meminfo
├── mountall.log -> /tmp/mountall.log
├── os-release -> /etc/os-release
├── partitions -> /proc/partitions
├── pinctrl -> /sys/kernel/debug/pinctrl/
├── rkcif-mipi-lvds -> /proc/rkcif-mipi-lvds
├── rk_dmabuf -> /proc/rk_dmabuf
├── rkisp0-vir0 -> /proc/rkisp0-vir0
├── slabinfo -> /proc/slabinfo
├── softirqs -> /proc/softirqs
├── version -> /proc/version
└── wakeup_sources -> /sys/kernel/debug/wakeup_sources
...
```

11. Chapter-11 SDK Software License Instructions

Before customers obtain the SDK, they need to sign the SDK agreement. This agreement states specific rights and obligations.

11.1 Copyright Detection Tool

11.1.1 Buildroot

```
make legal-info
```

The SDK project can automatically generate relevant License information by running the `make legal-info` command, Like RK3566 project:

```
$ source buildroot/envsetup.sh rockchip_rk3566
buildroot$ make legal-info
```

Executing the above command will generate the following information:

```
buildroot$ tree -L 1 output/rockchip_rk3566/legal-info/
output/rockchip_rk3566/legal-info/
├── buildroot.config
├── host-licenses
├── host-manifest.csv
├── host-sources
├── legal-info.sha256
├── licenses
├── manifest.csv
├── README
└── sources
```

The `manifest.csv` is the license information of the relevant repository used by RK3566 project.
The `host-manifest.csv` is the license information of the third-party tools of the PC used by RK3566 project.

11.1.2 Debian

For license detection, please refer to Official Tool

```
licensecheck --check '.*' --recursive --copyright --deb-fmt \
        --lines 0 * | /usr/lib/cdbs/licensecheck2dep5
```

The relevant copyright instructions for each source code package are located in `/usr/share/doc/*/copyright`

11.1.3 Yocto

The relevant copyright instructions for each source code package are located in `build/tmp/deploy/licenses/*/recipeinfo`

11.2 License List

For the current mainstream License list, please refer to the following link

license-list

12. Chapter-12 Rockchip open source information

12.1 Github

Rockchip Code Open Source repositories rockchip-github。

12.2 Wiki

Rockchip open source materials rockchip-wiki

12.3 Upstream

- Rockchip upstream uboot:

```
git clone https://gitlab.denx.de/u-boot/custodians/u-boot-rockchip.git
```

Upstream U-Boot support the following Rockchip SoCs:

`RK3036, RK3188, RK3288, RK3328, RK3399, RK3566, RK3568`

- Rockchip upstream kernel

```
git clone git://git.kernel.org/pub/scm/linux/kernel/git/mmind/linux-rockchip.git
```

Mainline kernel supports:

`RV1108, RK3036, RK3066, RK3188, RK3228, RK3288, RK3368, RK3399, RK3566, RK3568`

13. Chapter-13 SDK FAQ

13.1 How to Confirm the Current SDK Version and System/Kernel Version?

Relevant information can be obtained through the `/info/ directory` or `/etc/os-release`, such as

```
root@rk3588:/# cat /etc/os-release
NAME=Buildroot
VERSION=linux-5.10-gen-rkr3.4-48-gb0d2bfa6
ID=buildroot
VERSION_ID=2021.11
PRETTY_NAME="Buildroot 2021.11"
BUILD_INFO="wxt@ubuntu-191 Wed Nov 23 09:17:44 CST 2022 -
/home/wxt/test/rk3588/buildroot/configs/rockchip_rk3588_defconfig"
KERNEL="5.10 - rockchip_linux_defconfig"
```

13.2 Issues about SDK Building

13.2.1 Sync Issues Caused by Repo

When using `repo sync -c` to update, prompts `No module named formatter`. This is because your host uses a new version of python. For example, python3.8+ completely removes formatter, and the repo version of the released SDK is too old. This is only fixed by updating the repo version, for example, through the following ways:

```
$ cd .repo/repo/
$ git checkout origin/stable

This branch repo is the 2022 version, and the default master branch is the 2020
version. Or add `--repo-rev=stable` during the repo init project to switch to the
new version repo.

repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo --repo-
rev=stable \
-u ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests -b linux -m \
rk3588_linux_release.xml
```

13.2.2 Abnormal Issues Caused by ./build.sh Building

The following error will be prompted:

```
Adding information to /etc/os-release...
Traceback (most recent call last):
file "rk356x/.repo/repo/main.py", line 56, in <module>
from subcmds.version import Version
File "rk356x/.repo/repo/subcmds/__init__.py", line 35, in <module>
ModuleNotFoundError: No module named 'formatter'
```

Due to the matching problem between python3 and project repo versions in buildroot, `device/rockchip` can be modified as follows:

```
--- a/common/post-build.sh
+++ b/common/post-build.sh
@@ -171,8 +171,6 @@ function add_build_info()

     mkdir -p "$INFO_DIR"

-    yes | python3 .repo/repo/repo manifest -r -o "$INFO_DIR/manifest.xml"
```

13.2.3 Buildroot Building Issues

If some network reasons cause buildroot compilation to fail. This can be solved by using the following ways:

Preset dl directory, dl is a pre-compiled package that can be integrated into buildroot in advance. Reduce download time and improve compilation efficiency.

For example, rk3588 Linux SDK can be modified to add the dl directory as follows

```
$ cd .repo/manifests
$ Change the rk3588_linux_release.xml file as follows:

diff --git a/rk3588_linux_release.xml
b/rrk3588_linux_release.xml
index 5082dea..626f094 100644
--- a/rk3588_linux/rk3588_linux_release.xml
+++ b/rk3588_linux/rk3588_linux_release.xml
@@ -14,6 +14,7 @@

+  <project name="linux/buildroot/dl" path="buildroot/dl" revision="next"/>

$ cd -
$.repo/repo sync -c
```

The modification methods for other chip SDKs are similar.

13.3 Debian Related Issues

13.3.1 "noexec or nodev" Issues

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

Solution：

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

In addition, if you encounter other compilation exceptions, firstly, make sure that the compilation system used is not ext2/ext4.

13.3.2 Failed to Download "Base Debian"

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

Debian uses live build, and if the image source is changed to domestic, it can be configured like this:

```
32 bit system:

+++ b/ubuntu-build-service/{buster/bullseye}-desktop-armhf/configure
@@ -11,6 +11,11 @@ set -e
 echo "I: create configuration"
 export LB_BOOTSTRAP_INCLUDE="apt-transport-https gnupg"
 lb config \
+ --mirror-bootstrap "http://mirrors.ustc.edu.cn/debian" \
+ --mirror-chroot "http://mirrors.ustc.edu.cn/debian" \
+ --mirror-chroot-security "http://mirrors.ustc.edu.cn/debian-security" \
+ --mirror-binary "http://mirrors.ustc.edu.cn/debian" \
+ --mirror-binary-security "http://mirrors.ustc.edu.cn/debian-security" \
   --apt-indices false \
   --apt-recommends false \
   --apt-secure false \

  64 bit system:
  --- a/ubuntu-build-service/{buster/bullseye}-desktop-arm64/configure
+++ b/ubuntu-build-service/{buster/bullseye}-desktop-arm64/configure
@@ -11,6 +11,11 @@ set -e
 echo "I: create configuration"
 export LB_BOOTSTRAP_INCLUDE="apt-transport-https gnupg"
 lb config \
+ --mirror-bootstrap "http://mirrors.ustc.edu.cn/debian" \
+ --mirror-chroot "http://mirrors.ustc.edu.cn/debian" \
+ --mirror-chroot-security "http://mirrors.ustc.edu.cn/debian-security" \
+ --mirror-binary "http://mirrors.ustc.edu.cn/debian" \
+ --mirror-binary-security "http://mirrors.ustc.edu.cn/debian-security" \
   --apt-indices false \
   --apt-recommends false \
   --apt-secure false \
```

If the package cannot be downloaded due to other network reasons, there are pre-generated packages shared in Debian Base Download Package, and placed in the current directory to directly go to the next step.

13.3.3 Abnormal Operations Lead to Errors in Mounting /dev

For example, `askpass command or cannot use one` appears.

The error may have been caused by frequent abnormal operations (CTRL+C) during the compilation process. It can be fixed as follows:

```
 sudo -S  umount  /dev
```

13.3.4 /dev Issue Caused by Multiple Mounts

For example, `sudo: unable to allocate pty: No such device` occurs.

The error may have been caused by multiple mounts during the compilation process. It can be fixed as follows:

```
ssh <username>@<IP address> -T sudo -S umount /dev -l
```

13.3.5 How to Check System Information

13.3.5.1 How to Check the System Debian Version?

```
root@linaro-alip:~# cat /etc/debian_version
11.1
```

13.3.5.2 How to Check Whether Debian uses X11 or Wayland?

In X11 systems:

```
$ echo $XDG_SESSION_TYPE
x11
```

In Wayland systems:

```
$ echo $XDG_SESSION_TYPE
wayland
```

13.3.5.3 How to Check the System Partition Status

```
root@linaro-alip:~# parted -l

Model: MMC BJTD4R (sd/mmc)
Disk /dev/mmcblk0: 31.3GB
```

```
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name      Flags
 1      8389kB  12.6MB  4194kB               uboot
 2      12.6MB  16.8MB  4194kB               misc
 3      16.8MB  83.9MB  67.1MB               boot
 4      83.9MB  218MB   134MB                recovery
 5      218MB   252MB   33.6MB               backup
 6      252MB   15.3GB  15.0GB  ext4         rootfs
 7      15.3GB  15.4GB  134MB   ext2         oem
 8      15.6GB  31.3GB  15.6GB  ext2         userdata
```

13.3.5.4 ssh.service is Abnormal in the System

This is a problem with Debian10 or earlier. Add the following to /etc/rc.local:

```
#!/bin/sh -e
#
## Chapter-13 rc.local
#
## Chapter-13 This script is executed at the end of each multiuser runlevel.
## Chapter-13 Make sure that the script will "exit 0" on success or any other
## Chapter-13 value on error.
#
## Chapter-13 In order to enable or disable this script just change the execution
## Chapter-13 bits.
#
## Chapter-13 By default this script does nothing.
## Chapter-13 Generate the SSH keys if non-existent
if [ ! -f /etc/ssh/ssh_host_rsa_key ]
then
    # else ssh service start in dpkg-reconfigure will fail
    systemctl stop ssh.socket||true
    dpkg-reconfigure openssh-server
fi

exit 0
```

13.3.6 Debian11 Base Package Cannot be Built

You will encounter errors similar to the following:

```
W: Failure trying to run:  /sbin/ldconfig
W: See //debootstrap/debootstrap.log for details
```

The main requirement is that the PC kernel version is 5.10+. This is a bug in the old QEMU. There are two main solutions:

- The kernel version that comes with the PC needs to meet the requirements of 5.10+.

Check PC kernel version:

```
cat /proc/version
Linux version 5.13.0-39-generic
```

- Update system qemu

Refer to qemu.

13.3.7 How to Decompress, Modify and Repackage the Debian deb Package

If you want to modify and repackage the original deb, the method is as follows:

```
#Extract the files in the package to the extract directory
dpkg -X xxx.deb extract/

#Extract the control information of the package under extract/DEBIAN/:
dpkg -e xxx.deb extract/DEBIAN/

#Modify file XXX

## Chapter-13 Repackage the modified content to generate a deb package
dpkg-deb -b extract/ .
```

13.3.8 How to Add Swap Partition in Debian

When the system's physical memory is not enough, you can add Debian's swap virtual memory partition for use by currently running programs. For example, create a 2G virtual memory

- Create a swap file

```
cd /opt
mkdir swap
dd if=/dev/zero of=swapfile bs=1024 count=2000000
## Chapter-13 count represents the size, it is 2G in this example
```

- Convert files to swap files

```
sudo mkswap swapfile
```

- Activate swap file

```
swapon /opt/swapfile

uninstall:
swapoff /opt/swapfile
```

If it is automatically mounted after booting, you can add it to the /etc/fstab file.
eg: /opt/swapfile swap swap defaults 0 0

- Verify whether it takes effect

```
root@linaro-alip:/opt# free -h
               total        used        free      shared  buff/cache   available
Memory:       1.9Gi       390Mi        91Mi        75Mi       1.5Gi       1.4Gi
Swap:         1.9Gi          0B       1.9Gi
e =h
```

13.3.9 When Updating the Debian System for the First Time, It Will Restart the Display Service.

In order to be compatible with different chips, general Debian will install various differential packages according to the chip when /etc/init.d/rockchip.sh is started for the first time, such as libmali isp and other packages. After installation, the display service will be restarted. If it is an independent project, it can be placed in the image to handle this part of the difference.

13.3.10 Issues Caused by Calling libGL Related dri.so in Debian

The reasons are mainly as follows:

- EGL is an extension of OpenGL on the ARM platform for the x window system. Its function is equivalent to the glx library under x86.
- Since the Driver modesettings used by Xorg will load libglx.so by default (disabling glx will cause some applications that detect the glx environment to fail to start), libglx.so will search for the dri implementation library in the system. However, Xorg 2D acceleration is implemented directly based on DRM and does not implement the dri library, so during the booting process, libglx.so will report the following error.

```
(EE) AIGLX error: dlopen of /usr/lib/aarch64-linux-gnu/dri/rockchip_dri.so failed
```

It has no impact on system operation and does not require processing.

Based on the same reason, the following errors will be reported during the startup process of some applications, which do not need to be processed and will not affect the operation of the application.

```
libGL error: unable to load driver: rockchip_dri.so
libGL error: driver pointer missing
libGL error: failed to load driver: rockchip
```

13.3.11 How to Confirm that the Hardware Mouse Layer is Working in Debian

- The kernel dts is configured

Similar to the following log:

```
root@linaro-alip:~# dmesg |grep cursor
[    2.062561] rockchip-vop2 fe040000.vop: [drm:vop2_bind] Cluster1-win0 as
cursor plane for vp0
[    2.062669] rockchip-vop2 fe040000.vop: [drm:vop2_bind] Cluster0-win0 as
cursor plane for vp1
```

- whether modetest tests layer is reported

```
102     0       0       0,0             0,0     0               0x00000002
  formats: XR30 AR30 XB30 AB30 XR24 AR24 XB24 AB24 RG24 BG24 RG16 BG16 YU08 YU10
YUYV Y210
  props:
        8 type:
                flags: immutable enum
                enums: Overlay=0 Primary=1 Cursor=2
                value: 2
```

- Check whether `summary` call the hard mouse layer or not

root@linaro-alip:~# cat /sys/kernel/debug/dri/0/summary |grep 64x64

If there are still problems after steps 1/2, check /var/log/drm-cursor.log to see if there are any abnormalities.

13.4 Linux Video Related Issues

13.4.1 Video Playback Freezes and Frame Drop Errors Appear in the log. How to Fix the Issue?

Using fpsdisplaysink to confirm the maximum frame rate. If the maximum frame rate is close to the expected frame rate, you can specify `sync=false` to solve the problem, some platforms can enable AFBC. About video playback freezes, you can also check the hardware running time by `echo 0x100 > /sys/module/rk_vcodec/parameters/mpp_dev_debug`

13.4.2 gst-launch-1.0 Camera Video Preview Command

You can use v4l2src plugin, such as `gst-launch-1.0 v4l2src ! autovideosink`

13.4.3 The Playback Screen Jitters after Turning on AFBC, How to Fix the Issue?

Screen jitter is generally caused by insufficient DDR bandwidth. You can try the fixed performance mode. In addition, due to the display hardware implementation, If there is scaling in the vertical direction, the required performance and bandwidth will be relatively high, which is easily insufficient, and other scaling methods should to be used (such as rga/gpu).

13.4.4 Is the Gstreamer Framework Buffer Zero Copy?

Use dmabuf related interfaces for data processing to achieve zero copy

13.4.5 How to Test the Highest Decoding Performance of gst-launch-1.0 ?

Set the performance mode, use the official fpsdisplaysink to check the frame rate, and the driver's debugging interface to check the driver frame rate.

13.4.6 If the Screen Jitters or Ripples Appear during Playback, How to Fix the Issue?

Jitter is generally caused by insufficient performance of the display hardware, mostly due to insufficient DDR bandwidth. DDR performance mode can be used to fix the highest frequency

13.4.7 How to Quickly Connect GStreamer to Opengles

Generally, the synthesis is supported by the gl plug-in, and the display is supported by a third-party display service. Display services that can be synthesized internally through opengles (such as Weston)

13.5 Third-party OS Porting Issues

13.5.1 Is There Any Introduction to Kylin System Porting, Which is to Download the Standard iso Image and Extract rootfs.squafs for Porting?

You can refer to the introduction of the porting document in the SDK, which is also applicable to Kirin os, but Kirin os is relatively closed. If you want better performance, you can ask Kirin for images of RK platform.

13.5.2 Which Domestic OS Have Been Adapted?

Tongxin and Kirin have adapted. In addition, the HarmonyOS community is currently working on the adaptation of RK3588, RK356X, and RK3399. For detailed progress, you can also consult our company's business, or follow the HarmonyOS open source community.

13.5.3 Whether to Support UEFI Booting

RK3588 will give priority to supporting UEFI

13.6 Display Related Issues

13.6.1 How to Send Video to the Video Layer

There are interface of drm to query the type of plane. For details, please refer to kmssink method of gstreamer.

13.6.2 How to Configure the Position of Each Screen in Wayland Multi-screen with Differential Display Mode, Such As Left and Right or Up and Down Positions

Weston's differential display only supports left and right arrangement, according to the screen loading order. please refer to `<SDK>/docs/cn/Linux/*/Rockchip_Developer_Guide_Buildroot_Weston_CN.pdf` 2.9 Multi-screen configuration for details.

13.6.3 What is the Debian Xserver Version?

Debian10 uses xserver1.20.4, Debian11 uses xserver1.20.11

14. Chapter-14 SSH Public Key Operation Introduction

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

14.1 Multiple Device Use the Same SSH Public Key

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

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

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

Adding ssh private key may result in the following error.

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

Enter the following command in console to solve:

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

14.2 Switches Different SSH Public Keys on One Device

You can configure SSH by referring to ssh_config documentation.

```
~$ man ssh_config
```

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

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

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

14.3 Key Authority Management

Server can monitor download times and IP information of a key in real time. If an abnormality is found, download permission of the corresponding key will be disabled.

Keep the private key file properly. Do not grant second authorization to third parties.

14.4 Reference Documents

For more details, please refer to document `<SDK>/docs/en/Others/Rockchip_User_Guide_SDK_Application_And_Synchronization_EN.pdf`.