

Rockchip Linux Software Developer Guide

ID: RK-KF-YF-902

Release Version: V2.3.0

Release Date: 2024-09-20

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

DISCLAIMER

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

Trademark Statement

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

All rights reserved. ©2024. Rockchip Electronics Co., Ltd.

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

Rockchip Electronics Co., Ltd.

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

Website: www.rock-chips.com

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

Customer service Fax: +86-591-83951833

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

Preface

Overview

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

Date	Author	Version	Change Description
2023-12-20	Caesar Wang	V2.1.0	Update chip system support status Add Roadmap sections Update documentation sections Update SDK compilation instructions Update SDK development sections Update FAQ section
2024-06-20	Caesar Wang	V2.2.0	Add support for the RK3576 chip Update the status of chip support, and remove Linux 4.4 and Linux 4.19 Update more module development instructions, such as post-processing for rootfs, Overlay handling, etc. Update FAQ
2024-09-20	Caesar Wang	V2.3.0	Update Camera Development Introduction Update Security Development Introduction Add Detailed Explanation of Recovery Function Update SDK Launch Method Add Development Introduction for Boot Logo, Animation, and Other Boot Screen Images Add Section on Shared Remote Desktop Development Update Frequently Asked Questions

RK Linux SDK Supported System List

Linux6.1 SDK

Chipset	Buildroot Version	DebianVersion	Yocto Version	Kernel Version	SDK Version	TAG Version
RK3588	2024.02	12	5.0	6.1	V1.1.1_20240920	linux-6.1-stan-rkr4
RK3576	2024.02	12	5.0	6.1	V1.0.1_20240920	linux-6.1-stan-rkr4
RK3568	2024.02	12	5.0	6.1	V1.0.1_20240920	linux-6.1-stan-rkr4
RK3566	2024.02	12	5.0	6.1	V1.0.1_20240920	linux-6.1-stan-rkr4
RK3562	2024.02	12	5.0	6.1	V1.0.0_20240920	linux-6.1-stan-rkr4

Linux5.10 SDK

Chipset	Buildroot Version	DebianVersion	Yocto Version	Kernel Version	SDK Version	TAG Version
RK3588	2021.11	11	4.0	5.10	V1.5.0_20240620	linux-5.10-gen-rkr8
RK3562	2021.11	11	4.0	5.10	V1.2.0_20240620	linux-5.10-stan-rkr3
RK3566	2021.11	11	4.0	5.10	V1.5.0_20240620	linux-5.10-gen-rkr8
RK3568	2021.11	11	4.0	5.10	V1.5.0_20240620	linux-5.10-gen-rkr8
RK3399	2021.11	11	4.0	5.10	V1.5.0_20240620	linux-5.10-gen-rkr8
RK3358	2021.11	N/A	N/A	5.10	V1.4.0_20240620	linux-5.10-gen-rkr8
RK3328	2021.11	N/A	4.0	5.10	V1.1.0_20240620	linux-5.10-gen-rkr8
RK3326	2021.11	N/A	N/A	5.10	V1.5.0_20240620	linux-5.10-gen-rkr8
PX30	2021.11	11	4.0	5.10	V1.5.0_20240620	linux-5.10-gen-rkr8
RK3308	2021.11	N/A	N/A	5.10	V1.5.0_20240620	linux-5.10-gen-rkr8
RK3288	2021.11	11	4.0	5.10	V1.2.0_20240620	linux-5.10-gen-rkr8

Chipset	Buildroot Version	DebianVersion	Yocto Version	Kernel Version	SDK Version	TAG Version
RK312X	2021.11	N/A	N/A	5.10	V1.4.0_20240620	linux- 5.10- gen- rkr8
RK3036	2021.11	N/A	N/A	5.10	V1.4.0_20240620	linux- 5.10- gen- rkr8

2024-2025 Linux SDK Upgrade Schedule

Chipset	Buildroot Version	Debian Version	Yocto Version	Kernel Version	Release schedule
RK3506	2024.02	N/A	N/A	6.1	2024.Q4
RK3399	2024.02	N/A	5.0	6.1	2024.Q4

Contents

Rockchip Linux Software Developer Guide

1. Chapter-1 SDK Development Roadmap
 - 1.1 SDK Roadmap
 - 1.2 Kernel Roadmap
 - 1.3 Buildroot Roadmap
 - 1.4 Yocto Roadmap
 - 1.5 Debian Roadmap
2. Chapter-2 SDK
 - 2.1 Overview
 - 2.2 Obtain the General SDK
 - 2.2.1 Get Source Code from Rockchip Code Server
 - 2.2.1.1 Rockchip Linux6.1 SDK Downloading
 - 2.2.1.2 Rockchip Linux5.10 SDK Downloading
 - 2.2.2 Get Source Code from Local Compression Package
 - 2.3 Summary of SDK Build Commands
3. Chapter-3 Documents Introduction
 - 3.1 General Development Guidance Document (Common)
 - 3.1.1 Asymmetric Multi-Processing System Development Guide (AMP)
 - 3.1.2 Audio Module Document (AUDIO)
 - 3.1.3 Peripheral Components Support List (AVL)
 - 3.1.3.1 DDR Support List
 - 3.1.3.2 eMMC Support List
 - 3.1.3.3 SPI NOR and SLC NAND Flash Support List
 - 3.1.3.4 NAND Flash Support List
 - 3.1.3.5 WIFI/BT Support List
 - 3.1.3.6 Camera Support List
 - 3.1.4 CAN Module Document (CAN)
 - 3.1.5 Clock Module Document (CLK)
 - 3.1.6 CRYPTO Module Document (CRYPTO)
 - 3.1.7 DDR Module Document (DDR)
 - 3.1.8 Debug Module Document (DEBUG)
 - 3.1.9 Display Module Document (DISPLAY)
 - 3.1.10 Dynamic Frequency and Voltage Adjustment Module Documentation (DVFS)
 - 3.1.11 File System Module Documentation
 - 3.1.12 Ethernet Module Document (GMAC)
 - 3.1.13 HDMI-IN Module Document (HDMI-IN)
 - 3.1.14 I2C Module Document (I2C)
 - 3.1.15 IO Power Domain Module Document (IO-DOMAIN)
 - 3.1.16 IOMMU Module Document (IOMMU)
 - 3.1.17 Image Module Document (ISP)
 - 3.1.18 MCU Module Document (MCU)
 - 3.1.19 MMC Module Document (MMC)
 - 3.1.20 Memory Module Document (MEMORY)
 - 3.1.21 MPP Module Document (MPP)
 - 3.1.22 NPU Module Document (NPU)
 - 3.1.23 NVM Module Document (NVM)
 - 3.1.24 PCIe Module Document (PCIe)
 - 3.1.25 Performance Module Document (PERF)
 - 3.1.26 GPIO Module Document (PINCTRL)
 - 3.1.27 PMIC Module Document (PMIC)
 - 3.1.28 Power Module Document (POWER)
 - 3.1.29 PWM Module Document (PWM)
 - 3.1.30 RGA Module Document (RGA)
 - 3.1.31 SARADC Module Document (SARADC)
 - 3.1.32 SPI Module Document (SPI)

- 3.1.33 Thermal Module Document (THERMAL)
 - 3.1.34 Tools Module Document (TOOL)
 - 3.1.35 Security Module Document (TRUST)
 - 3.1.36 UART Module Document (UART)
 - 3.1.37 UBOOT Module Document (UBOOT)
 - 3.1.38 USB Module Document (USB)
 - 3.1.39 Watchdog Module Document (WATCHDOG)
- 3.2 Linux System Development Documents (Linux)
 - 3.2.1 ApplicationNote
 - 3.2.2 Audio Development Documents (Audio)
 - 3.2.3 Camera Development Documents (Camera)
 - 3.2.4 Docker Development Documents (Docker)
 - 3.2.5 Graphics Development Documents (Graphics)
 - 3.2.6 Multimedia
 - 3.2.7 SDK Profile introduction (Profile)
 - 3.2.8 OTA Upgrade (Recovery)
 - 3.2.9 Security Solution (Security)
 - 3.2.10 System Development (System)
 - 3.2.11 UEFI Booting (UEFI)
 - 3.2.12 Network Module (RKWIFIBT)
 - 3.2.13 DPDK Module (DPDK)
- 3.3 Chip Platform Related Documents (Socs)
 - 3.3.1 Release Note
 - 3.3.2 Quick Start
 - 3.3.3 Software Development Guide
- 3.4 Datasheet
 - 3.4.1 Hardware Development Guide
- 3.5 Other Documents (Others)
- 3.6 Documents List (docs_list_en.txt)
- 4. Chapter-4 Tools Introduction
 - 4.1 Driver Installation Tool
 - 4.2 Burning Tools
 - 4.3 Packaging Tools
 - 4.4 Tools used to Make SD Card Upgrading and Booting
 - 4.5 Device Information Writing Tool
 - 4.6 Firmware Signature Tool
 - 4.7 Programmer Upgrade Tool
 - 4.8 PCBA Testing Tools
 - 4.9 DDR Soldering Test Tool
 - 4.10 eFuse Programming Tool
 - 4.11 Mass Production Upgrade Tool
 - 4.12 Partition Modification Tool
- 5. Chapter-5 SDK Software Framework
 - 5.1 Introduction to SDK project Directory
 - 5.1.1 SDK Overview
 - 5.1.2 Buildroot
 - 5.1.3 Yocto
 - 5.1.4 Debian
 - 5.2 SDK Software Architecture
 - 5.3 SDK Development Process
- 6. Chapter-6 SDK Development Environment Setup
 - 6.1 Overview
 - 6.2 Preparation Work before SDK Development
 - 6.2.1 Install and Configure Git
 - 6.2.2 Install and Configure repo
 - 6.2.3 SDK Obtaining
 - 6.2.3.1 SDK Download Command
 - 6.2.3.2 Compressed Package of SDK Code

- 6.2.3.3 Software Update History
 - 6.2.3.4 SDK Update
 - 6.2.3.5 SDK Issue Feedback
 - 6.3 Setting Up a Linux Development Environment
 - 6.3.1 Preparing the Development Environment
 - 6.3.2 Installing Libraries and Toolkits
 - 6.3.2.1 Checking and Upgrading the Host's Python Version
 - 6.3.2.2 Checking and Upgrading the Host's `make` Version
 - 6.3.2.3 Checking and Upgrading the Host's LZ4 Version
 - 6.4 Window PC Development Environment Setup
 - 6.4.1 Development Tool Installation
 - 6.4.2 Rockchip USB Driver Installation
 - 6.4.3 Windows Burning Tool Usage
 - 6.4.4 Target Hardware Board Preparation
 - 6.5 Docker Environment Setup
 - 6.6 Introduction to Cross-compilation Tool Chain
 - 6.6.1 U-Boot and Kernel Compilation Tool Chain
 - 6.6.2 Buildroot Toolchain
 - 6.6.2.1 Configuring the Compilation Environment
 - 6.6.2.2 Packaging Toolchain
 - 6.6.3 Debian Toolchain
 - 6.6.4 Yocto Toolchain
- 7. Chapter-7 SDK Version and Update Instructions
 - 7.1 SDK Naming Rules
 - 7.1.1 SDK tag Naming Rules
 - 7.1.2 SDK Release Document Naming Rules
 - 7.1.3 SDK Compressed Package Naming Rules
 - 7.2 SDK Update Mechanism
 - 7.2.1 SDK Update
 - 7.2.2 SDK Release Patch
 - 7.3 How to Build Server to Sync the SDK
- 8. Chapter-8 SDK Compilation Instructions
 - 8.1 SDK Compilation Command
 - 8.2 SDK Board Level Configuration
 - 8.3 SDK Customization Configuration
 - 8.4 SDK Environment Variable Configuration
 - 8.5 Fully Automatic Compilation
 - 8.6 Build Modules
 - 8.6.1 Build U-Boot
 - 8.6.2 Build Kernel
 - 8.6.3 Build Recovery
 - 8.6.4 Build Buildroot
 - 8.6.5 Build Debian
 - 8.6.6 Build Yocto
 - 8.6.7 Cross-Compilation
 - 8.6.7.1 SDK Directory Built-in Cross-Compilation
 - 8.6.7.2 Built-in Cross Compilation in Buildroot
- 9. Chapter-9 SDK Firmware Upgrade
 - 9.1 Burning Mode Introduction
 - 9.1.1 Windows Flashing Instructions
 - 9.1.2 Linux Flashing Instructions
 - 9.1.3 System Partition Introduction
- 10. Chapter-10 SDK Development
 - 10.1 Bootloader Development
 - 10.1.1 U-Boot Development
 - 10.1.1.1 Introduction to U-Boot
 - 10.1.1.2 Version
 - 10.1.1.3 Preliminary Preparation

- 10.1.1.4 Booting Process
 - 10.1.1.5 Shortcut Keys
- 10.1.2 UEFI Development
- 10.2 Kernel Development
 - 10.2.1 Introduction to DTS
 - 10.2.1.1 DTS Overview
 - 10.2.1.2 Adding a New Product DTS
 - 10.2.2 Module Development Documentation
 - 10.2.3 Common Module Commands
 - 10.2.3.1 CPU-Related Commands
 - 10.2.3.1.1 CPU Frequency Locking Operation
 - 10.2.3.1.2 View Current CPU Frequency
 - 10.2.3.1.3 Checking Current CPU Voltage
 - 10.2.3.1.4 CPU Independent Frequency and Voltage Adjustment
 - 10.2.3.1.5 How to View Frequency and Voltage Table
 - 10.2.3.1.6 How to Check CPU Temperature
 - 10.2.3.2 GPU-Related Commands
 - 10.2.3.2.1 GPU Clock Setting Operation
 - 10.2.3.2.2 View Current GPU Frequency
 - 10.2.3.2.3 Inspect Current GPU Voltage
 - 10.2.3.2.4 GPU Frequency and Voltage Adjustment
 - 10.2.3.2.5 Inspect GPU Utilization
 - 10.2.3.2.6 Inspect GPU Temperature
 - 10.2.3.3 DDR Related Commands
 - 10.2.3.3.1 Inspect DDR Frequency
 - 10.2.3.3.2 DDR Frequency Setting Operation
 - 10.2.3.3.3 View DDR Bandwidth Utilization
 - 10.2.3.3.4 DDR Bandwidth Statistics
 - 10.2.3.3.5 Memory Debugging
 - 10.2.3.3.6 Memory Stress Test
 - 10.2.3.4 NPU-related Commands
 - 10.2.3.4.1 Viewing NPU Frequency
 - 10.2.3.4.2 NPU Frequency Setting Operation
 - 10.2.3.4.3 NPU Support Query Settings
 - 10.2.3.4.4 Related Materials
 - 10.2.3.5 RGA-related Commands
 - 10.2.3.5.1 Query RGA Frequency
 - 10.2.3.5.2 Modifying RGA Frequency
 - 10.2.3.5.3 RGA Driver Version Query
 - 10.2.3.5.4 Query the version number of the librqa library
 - 10.2.3.5.5 Enable librqa Logging
 - 10.2.3.5.6 RGA Debug Node
 - 10.2.3.5.7 RGA Load Query
 - 10.2.3.5.8 RGA Memory Manager Inquiry
 - 10.2.3.5.9 RGA Task Request Inquiry
 - 10.2.3.5.10 RGA Hardware Information Inquiry
 - 10.2.3.5.11 Dump Runtime Data
 - 10.2.3.6 VPU-related Commands
 - 10.2.3.6.1 Querying VPU Driver Version
 - 10.2.3.6.2 Checking VPU Frame Rate
 - 10.2.3.6.3 Viewing Frequency
 - 10.2.3.6.4 Modify Frequency
 - 10.2.3.6.5 Enabling VPU Debug Print
 - 10.2.4 Pin Configuration Control
 - 10.2.4.1 General-Purpose Input/Output (GPIO)
 - 10.2.4.1.1 I/O Multiplexing (IOMUX)
 - 10.2.4.2 PULL (Port Pull-Up and Pull-Down)
 - 10.2.4.3 DRIVE-STRENGTH (Port Drive Strength)

- 10.2.4.4 SMT (Synchronous Metastability Trigger)
 - 10.2.5 Thermal Control Development
 - 10.2.6 DDR Development Guide
 - 10.2.7 SD Card Configuration
- 10.3 Recovery Development
 - 10.3.1 Introduction
 - 10.3.2 Debugging
 - 10.3.3 Recovery Feature Summary
- 10.4 Buildroot Development
- 10.5 Debian Development
- 10.6 Yocto Development
- 10.7 Audio Development
 - 10.7.1 Kernel Audio Driver Development
 - 10.7.2 Audio Pulseaudio Path Adaptation
- 10.8 Multimedia Development
- 10.9 Camera Development
 - 10.9.1 Image Processing Module (ISP)
 - 10.9.2 USB CAMERA
- 10.10 Graphics Development
 - 10.10.1 Boot Screen Development
 - 10.10.1.1 Boot Logo Development
 - 10.10.1.2 Boot Animation Development
 - 10.10.2 Desktop Application Development
 - 10.10.2.1 Weston Desktop Development
 - 10.10.2.2 Enlightenment Desktop Development
 - 10.10.3 LVGL Desktop Development
 - 10.10.3.1 Flutter Linux Desktop Development
- 10.11 Secure Development
 - 10.11.1 Secure Boot Initialization
 - 10.11.2 Compiling Secureboot
 - 10.11.3 Rockchip Anti-Cloning Feature
- 10.12 WIFI/BT Development
- 10.13 Post-Processing Development for ROOTFS
- 10.14 Overlays Development
- 10.15 SDK Launch Method
 - 10.15.1 Device Boot Process
 - 10.15.2 BOOTROM Process
 - 10.15.3 System Boot Methods
- 10.16 Shared Remote Desktop Development
 - 10.16.1 Controlling Remote Desktop with wlfreerdp
 - 10.16.2 Weston VNC Remote Desktop Access
- 10.17 SDK Testing
 - 10.17.1 Integrating Rockchip Stress Test Scripts
 - 10.17.2 Benchmark Testing
 - 10.17.3 Rockchip Modules and Stress Testing
- 11. Chapter-11 SDK System Debugging Tools Introduction
 - 11.1 ADB Tool
 - 11.1.1 Overview
 - 11.1.2 USB ADB Usage Instructions
 - 11.2 Busybox Tool
 - 11.3 GDB Tool
 - 11.4 IO Tool
 - 11.5 kmsgrab Tool
 - 11.6 modetest Tool
 - 11.7 Perf Tool
 - 11.8 Pmap Tool
 - 11.9 ProcrankTool
 - 11.10 PS Tool

- 11.11 Slabtop Tool
- 11.12 Strace Tool
- 11.13 Top Tools
- 11.14 Update Tool
- 11.15 Vendor_storage Tool
- 11.16 Vmstat Tool
- 11.17 Watch Tool
- 11.18 weston Debugging Method
- 11.19 Obtain System Log Information Automatically
- 12. Chapter-12 SDK Software License Instructions
 - 12.1 Copyright Detection Tool
 - 12.1.1 Buildroot
 - 12.1.2 Debian
 - 12.1.3 Yocto
 - 12.2 License List
- 13. Chapter-13 Rockchip open source information
 - 13.1 Github
 - 13.2 Wiki
 - 13.3 Upstream
- 14. Chapter-14 SDK FAQ
 - 14.1 How to Confirm the Current SDK Version and System/Kernel Version
 - 14.2 Issues about SDK Building
 - 14.2.1 Sync Issues Caused by Repo
 - 14.2.2 Abnormal Issues Caused by ./build.sh Building
 - 14.2.3 Compilation issue in the Docker environment with ./build.sh
 - 14.2.4 Buildroot Building Issues
 - 14.3 Recovery FAQ
 - 14.3.1 How to prevent recovery from being compiled into the update.img
 - 14.4 Buildroot FAQ
 - 14.4.1 Dual Screen Asynchronous Clock Times
 - 14.4.2 How to Set Buildroot System to Boot into Command Line Mode
 - 14.5 Debian FAQ
 - 14.5.1 "noexec or nodev" Issue
 - 14.5.2 Failed to Download "Base Debian"
 - 14.5.3 Abnormal Operation Causes an error to Mount /dev
 - 14.5.4 Multiple Mounts lead to /dev error
 - 14.5.5 How to Check System Related Information
 - 14.5.5.1 How to Check the Debian Version of Your System
 - 14.5.5.2 How to Check Whether the Debian Display Uses X11 or Wayland
 - 14.5.5.3 How to Check System Partition Status
 - 14.5.5.4 The ssh.service Is Abnormal in System
 - 14.5.6 Debian11 Base Package Fails to Build
 - 14.5.7 How to Decompress, Modify and Repackage Debian deb Package
 - 14.5.8 How to Add the Swap Partition in Debian
 - 14.5.9 Update Debian System for the First Time Will Restart the Display Service
 - 14.5.10 Errors Occurring in Debian When Calling libGL related dri.so
 - 14.5.11 How to Confirm that the Hardware Mouse Layer is Useful in Debian
 - 14.5.12 The log is Too Large in Debian
 - 14.5.13 Debian Setting Multi User Mode Issue
 - 14.5.14 Debian Username and Password
 - 14.5.15 Debian XFCE Desktop icon Double-click Abnormal
 - 14.5.16 Chromium browsers will have a command line flag: --no-sandbox
 - 14.5.17 Setting Up DRI2 Extension in Debian System's X11
 - 14.5.18 Installing GCC Toolchain on Debian
 - 14.5.19 Auto-Completion for Installing Packages on Debian
 - 14.5.20 Supporting DRI3 Extension in Debian X11 System
 - 14.5.21 Setting System to Boot into Command Line Mode
 - 14.5.22 Configuring Screen Rotation

- 14.5.23 Implementing No Black Screen Function in Debian
- 14.5.24 Removing Desktop Mouse Pointer Display in Debian System
- 14.5.25 Steps for Compiling and Porting the rkaiq/rkisp Repository in Debian
 - 14.5.25.1 Overview of Steps
- 14.5.26 How to Download Offline Deb Packages in Debian
- 14.5.27 How to Check glibc Version in Debian
- 14.5.28 Support for Screen Splitting in Debian Systems
- 14.6 Linux Video Related Issues
 - 14.6.1 When playing videos, there is stuttering, and the logs show frame dropping errors
 - 14.6.2 gst-launch-1.0 Camera Video Preview Command
 - 14.6.3 The Playback Screen Jitters after Turning on AFBC
 - 14.6.4 How does GStreamer implement zero-copy
 - 14.6.5 How to Test the Highest Decoding Performance of gst-launch-1.0
 - 14.6.6 The Screen Jitters or Ripples Appear during Playback
 - 14.6.7 How to Quickly Connect GStreamer to Opengles
- 14.7 Third-party OS Porting Issues
 - 14.7.1 Is There Any Introduction to Kylin System Porting, Which is to Download the Standard iso Image and Extract rootfs.squafs for Porting
 - 14.7.2 Which Domestic OS Have Been Adapted
 - 14.7.3 Whether to Support UEFI Booting
- 14.8 Display Related Issues
 - 14.8.1 Mouse Blinking Issue
 - 14.8.2 How to Send Video to the Video Layer
 - 14.8.3 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
 - 14.8.4 What is the Debian Xserver Version
- 14.9 Browser Related Issues
 - 14.9.1 Chromium R114 Scaling Issue
- 15. Chapter-15 SSH Public Key Operation Introduction
 - 15.1 Multiple Device Use the Same SSH Public Key
 - 15.2 Switches Different SSH Public Keys on One Device
 - 15.3 Key Authority Management
 - 15.4 Reference Documents

1. Chapter-1 SDK Development Roadmap

1.1 SDK Roadmap

Version	SDK Released	Projected EOL (End of Life)
Linux 4.4	2018-06-20	Jun 2023
Linux 4.19	2020-12-11	Dec 2023
Linux 5.10	2022-01-15	Future - Long Term Support (Dec 2025)
Linux 6.1	2023-12-20	Future - Long Term Support (until April 2028)

1.2 Kernel Roadmap

Version	Kernel Released	SDK Released	Projected EOL
4.4	2017-11-12	2018-06-20	Jan 2024
4.19	2018-10-22	2020-12-11	Dec 2024
5.10	2020-12-13	2022-01-15	Dec 2026
6.1	2022-12-11	2023-12-20	Dec 2026

The Linux Kernel's last stable release each year is designated as the LTS version. However, the exact number of versions released annually varies, usually around 5 to 6, making it difficult to predict the specific version number of the next LTS. The maintenance duration for LTS versions also varies, based on their usage extent. Typically, it starts with 2 years and may extend to 4 or 6 years (not guaranteed).

For specific Kernel EOL, refer to the official release:

[Kernel Release](#)

The currently released SDKs for Kernel versions 4.4, 4.19, 5.10, and 6.1 are long-term supported versions by Rockchip.

Minor versions are updated quarterly (e.g., 4.19.27 to 4.19.232).

Major versions are updated biennially (e.g., 4.4 to 4.19 to 5.10 to 6.1).

Regarding the next kernel LTS version: Rockchip's kernel update strategy involves releasing an LTS (Long-Term Support) version every two years, which is offset by one year compared to the mainline kernel's strategy of releasing an LTS version every year. For example, if the mainline releases one LTS version annually, Rockchip releases one every two years. Therefore, the subsequent LTS version for Rockchip will be released in 2024, followed by another in 2026, and so on.

1.3 Buildroot Roadmap

The current officially supported Buildroot versions by RK are `2018.02` and `2021.11`.

Version	Buildroot Released	SDK Released	Projected EOL
2018.02	2018-03-04	2018-06-20	Jan 2024
2021.11	2021-12-05	2022-05-20	Dec 2026
2024.02	Mar 2024	Q4 2024	Future - Long Term Support (until April 2028)

The Buildroot community releases a new version every three months, and one stable version is released annually.

For more details, refer to the official Buildroot release information:

[Buildroot Release](#)

1.4 Yocto Roadmap

The current officially supported Yocto versions by RK range from Dunfell (3.1) to Kirkstone (4.0), with the main maintenance version being Kirkstone (4.0).

Version	Yocto Released	SDK Released	Support Level
Kirkstone (4.0)	May 2022	2022-12-20	Long Term Support (Apr 2026 ¹)
Scarthgap (5.0)	April 2024	2024-12-20	Future - Long Term Support (until April 2028)

The Yocto community typically releases two major versions each year, usually in the spring and autumn seasons.

For more details, refer to the official Yocto release information:

[Yocto Release](#)

1.5 Debian Roadmap

The current officially supported Debian versions by RK range from Stretch (9) to Bookworm (12), with the main maintenance version being Bullseye (11).

Version	Debian Released	SDK Released	EOL LTS
Stretch (9)	2017-06-17	2018-06-20	2022-07-01
Buster (10)	2019-07-06	2020-12-11	2024-06-30
Bullseye (11)	2021-08-14	2022-01-15	~ 2026
Bookworm (12)	2023-06-10	Q4 2023	~ 2028

Debian LTS versions are updated biennially.

For more details, refer to the official Debian release information:

[Debian Release](#)

2. Chapter-2 SDK

2.1 Overview

Rockchip Linux SDK supports Buildroot, Yocto and Debian systems. Kernel is based on Kernel 4.4, Kernel 4.19 or Kernel 5.10, 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/.

2.2 Obtain the General SDK

2.2.1 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](#).

2.2.1.1 Rockchip Linux6.1 SDK Downloading

Chipset	Version	Download Command
RK3588	Linux6.1	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 rk3588 -m \ rk3588_linux6.1_release.xml
RK3576	Linux6.1	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 rk3576 -m \ rk3576_linux6.1_release.xml
RK3566_RK3568	Linux6.1	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 rk3566_rk3568 -m \ rk3566_rk3568_linux6.1_release.xml

2.2.1.2 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
RK3328	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 \ rk3328_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

Chipset	Version	Download Command
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
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
```

2.2.2 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_LINUX6.1_SDK_RELEASE_V1.0.0_20231220.tgz for example, After getting a initialization package, you can get source code by running the following command:

```
mkdir rk3588
tar xvf RK3588_LINUX6.1_SDK_RELEASE_V1.0.0_20231220.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_linux6.1_release.xml
In the following example: the printed version number is v1.0.0 and the update
time is 20231220
<SDK>/.repo/manifests/release/rk3588_linux6.1_release_v1.0.0_20231220.xml
```

The initial compressed SDK packages currently released for Linux6.2 are as follows:

Chipset	Compressed Package	Version
RK3588	RK3588_LINUX6.1_SDK_RELEASE_V1.0.0_20231220.tgz	v1.0.0
RK3576	RK3588_LINUX6.1_SDK_RELEASE_V1.0.0_20240620.tgz	v1.0.0
RK3566、RK3568	RK3566_RK3568_LINUX6.1_SDK_RELEASE_V1.0.0_20240620.tgz	v1.0.0

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!

2.3 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
RK3576	EVB	RK3576 EVB1	./build.sh lunch:rockchip_rk3576_evb1_v10_defconfig && ./build.sh	./build.sh rootfs	make ARCH=arm64 rockchip_linux_defconfig; make ARCH=arm64 rk3576-evb1- v10-linux.img -j24	./make.sh rk3576 --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`

Build Debian system: `RK_ROOTFS_SYSTEM=debian ./build.sh`

Build Yocto system: `RK_ROOTFS_SYSTEM=yocto ./build.sh`

- **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-gnueabi/bin/arm-none-linux-gnueabi-
```


3. Chapter-3 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_en.txt (docs file directory structure), and the detailed introduction as follows:

3.1 General Development Guidance Document (Common)

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

3.1.1 Asymmetric Multi-Processing System Development Guide (AMP)

For details, see the `<SDK>/docs/en/Common/AMP` directory. The AMP system is a general-purpose multicore heterogeneous system solution provided by Rockchip, which has been widely used in industrial applications such as power and industrial control, as well as consumer products like vacuum cleaners.

3.1.2 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/en/Common/AUDIO/  
├── Algorithms  
├── Rockchip_Developer_Guide_Audio_EN.pdf  
└── Rockchip_Developer_Guide_PulseAudio_EN.pdf
```

3.1.3 Peripheral Components Support List (AVL)

Please refer to `<SDK>/docs/en/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
```

3.1.3.1 DDR Support List

For the Rockchip platform DDR chip support list, please refer to "Rockchip_Support_List_DDR_Ver2.61.pdf" in the `<SDK>/docs/en/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

3.1.3.2 eMMC Support List

The eMMC chip support list for Rockchip platform can be found in the `<SDK>/docs/en/Common/AVL` directory in the document titled 'RKeMMCSupportList_Ver1.81_20240329.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.

[Table 23] Performance

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

Figure 1-1 eMMC Performance Example

3.1.3.3 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_V1.47_20240326.pdf" in the `<SDK>/docs/en/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

3.1.3.4 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

3.1.3.5 WIFI/BT Support List

WIFI/BT Support List for Rockchip Platform can be found in the document titled 'Rockchip_Support_List_Linux_WiFi_BT_Ver1.9_20240329.pdf' in the `<SDK>/docs/en/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.

3.1.3.6 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.

3.1.4 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/en/Common/CAN/
├── Rockchip_Developer_Guide_CAN_FD_EN.pdf
└── Rockchip_Developer_Guide_Can_EN.pdf
```

3.1.5 Clock Module Document (CLK)

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

```
docs/en/Common/CLK/
├── Rockchip_Developer_Guide_Clock_EN.pdf
├── Rockchip_Developer_Guide_Gpio_Output_Clocks_EN.pdf
└── Rockchip_Developer_Guide_Pll_Ssmod_Clock_EN.pdf
```

3.1.6 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/en/Common/CRYPTO/
└── Rockchip_Developer_Guide_Crypto_HWRNG_EN.pdf
```

3.1.7 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/en/Common/DDR/  
└─ Rockchip-Developer-Guide-DDR-EN.pdf
```

3.1.8 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/en/Common/DEBUG/  
└─ Rockchip_Developer_Guide_DS5_EN.pdf  
└─ Rockchip_Developer_Guide_FT232H_USB2JTAG.pdf  
└─ Rockchip_Developer_Guide_GDB_Over_ADB_EN.pdf  
└─ Rockchip_Developer_Guide_GNU_MCU_Eclipse_OpenOCD_EN.pdf
```

3.1.9 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/en/Common/DISPLAY/  
└─ DP  
└─ HDMI  
└─ MIPI  
└─ RK628  
└─ Rockchip_BT656_TX_AND_BT1120_TX_Developer_Guide_EN.pdf  
└─ Rockchip_Developer_Guide_Baseparameter_Format_Define_And_Use_EN.pdf  
└─ Rockchip_Developer_Guide_DRM_Display_Driver_EN.pdf  
└─ Rockchip_Developer_Guide_RGB_MCU_EN.pdf  
└─ Rockchip_Develop_Guide_DRM_Direct_Show_EN.pdf  
└─ Rockchip_DRM_Panel_Porting_Guide_V1.6_20190228.pdf  
└─ Rockchip_RK3588_Developer_Guide_MIPI_DSI2_EN.pdf
```

3.1.10 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/en/Common/DVFS/  
├─ Rockchip_Developer_Guide_CPUFreq_EN.pdf  
└─ Rockchip_Developer_Guide_Devfreq_EN.pdf
```

3.1.11 File System Module Documentation

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

```
docs/en/Common/FS/  
└─ Rockchip_Developer_FAQ_FileSystem_EN.pdf
```

3.1.12 Ethernet Module Document (GMAC)

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

```
docs/en/Common/GMAC/  
├─ Rockchip_Developer_Guide_Linux_GMAC_EN.pdf  
├─ Rockchip_Developer_Guide_Linux_GMAC_DPDK_EN.pdf  
├─ Rockchip_Developer_Guide_Linux_GMAC_Mode_Configuration_EN.pdf  
├─ Rockchip_Developer_Guide_Linux_GMAC_RGMII_Delayline_EN.pdf  
└─ Rockchip_Developer_Guide_Linux_MAC_TO_MAC_EN.pdf
```

3.1.13 HDMI-IN Module Document (HDMI-IN)

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

```
docs/en/Common/HDMI-IN/  
├─ Rockchip_Developer_Guide_HDMI_IN_Based_On_CameraHal3_EN.pdf  
└─ Rockchip_Developer_Guide_HDMI_RX_EN.pdf
```

3.1.14 I2C Module Document (I2C)

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

```
docs/en/Common/I2C/  
└─ Rockchip_Developer_Guide_I2C_EN.pdf
```

3.1.15 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/en/Common/IO-DOMAIN/  
└─ Rockchip_Developer_Guide_Linux_IO_DOMAIN_EN.pdf
```

3.1.16 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/en/Common/IOMMU/  
└─ Rockchip_Developer_Guide_Linux_IOMMU_EN.pdf
```

3.1.17 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/en/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 repository) 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".

3.1.18 MCU Module Document (MCU)

MCU development guide on the Rockchip platform.

```
docs/en/Common/MCU/  
└─ Rockchip_RK3399_Developer_Guide_MCU_EN.pdf
```

3.1.19 MMC Module Document (MMC)

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

```
docs/en/Common/MMC/  
├─ Rockchip_Developer_Guide_SDMMC_SDIO_eMMC_EN.pdf  
└─ Rockchip_Developer_Guide_SD_Boot_EN.pdf
```

3.1.20 Memory Module Document (MEMORY)

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

```
docs/en/Common/MEMORY/  
├─ Rockchip_Developer_Guide_Linux_CMA_EN.pdf  
├─ Rockchip_Developer_Guide_Linux_DMABUF_EN.pdf  
├─ Rockchip_Developer_Guide_Linux_Meminfo_EN.pdf  
└─ Rockchip_Developer_Guide_Linux_Memory_Allocator_EN.pdf
```

3.1.21 MPP Module Document (MPP)

MPP development instructions on the Rockchip platform.

```
docs/en/Common/MPP/  
└─ Rockchip_Developer_Guide_MPP_EN.pdf
```

3.1.22 NPU Module Document (NPU)

The SDK provides RKNPU-related development tools as follows:

RKNN-TOOLKIT2:

RKNN-Toolkit2 is a development kit for generating and evaluating RKNN models on a PC:

The development kit is located in the `external/rknn-toolkit2` directory, mainly used to implement a series of functions such as model conversion, optimization, quantization, inference, performance evaluation, and accuracy analysis.

Basic functions are as follows:

Function	Description
Model Conversion	Supports floating-point models of Pytorch / TensorFlow / TFLite / ONNX / Caffe / Darknet Supports quantization-aware models (QAT) of Pytorch / TensorFlow / TFLite Supports dynamic input models (dynamicization/native dynamic) Supports large models
Model Optimization	Constant folding / OP correction / OP Fuse&Convert / Weight sparsification / Model pruning
Model Quantization	Supports quantization types: asymmetric i8/fp16 Supports Layer / Channel quantization methods; Normal / KL/ MMSE quantization algorithms Supports mixed quantization to balance performance and accuracy
Model Inference	Supports model inference through the simulator on the PC Supports model inference on the NPU hardware platform (board-level inference) Supports batch inference, supports multi-input models
Model Evaluation	Supports performance and memory evaluation of the model on the NPU hardware platform
Accuracy Analysis	Supports quantization accuracy analysis function (simulator/NPU)
Additional Features	Supports version/device query functions, etc.

For specific usage instructions, please refer to the current `doc/` directory documentation:

```
└─ 01_Rockchip_RKNPU_Quick_Start_RKNN_SDK_V2.0.0beta0_EN.pdf
...
└─ RKNNToolkit2_API_Difference_With_Toolkit1-V2.0.0beta0.md
└─ RKNNToolkit2_OP_Support-v2.0.0-beta0.md
```

RKNN API:

The development instructions for RKNN API are located in the project directory `external/rknpu2`, used for inferring RKNN-Toolkit2 generated rknn models.

For specific usage instructions, please refer to the current `doc/` directory documentation:

```
...
└─ 02_Rockchip_RKNPU_User_Guide_RKNN_SDK_V2.0.0beta0_EN.pdf
└─ 03_Rockchip_RKNPU_API_Reference_RKNN_Toolkit2_V2.0.0beta0_EN.pdf
└─ 04_Rockchip_RKNPU_API_Reference_RKNNRT_V2.0.0beta0_EN.pdf
```

3.1.23 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/en/Common/NVM/
├─ Rockchip_Application_Notes_Storage_EN.pdf
├─ Rockchip_Developer_FAQ_Storage_EN.pdf
├─ Rockchip_Developer_Guide_Dual_Storage_EN.pdf
└─ Rockchip_Developer_Guide_SATA_EN.pdf
```

3.1.24 PCIe Module Document (PCIe)

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

```
docs/en/Common/PCIe/
├─ Rockchip_Developer_Guide_PCIe_EN.pdf
├─ Rockchip_Developer_Guide_PCIe_EP_Standard_Card_EN.pdf
├─ Rockchip_Developer_Guide_PCIe_Performance_EN.pdf
├─ Rockchip_PCIe_Virtualization_Developer_Guide_EN.pdf
└─ Rockchip_RK3399_Developer_Guide_PCIe_EN.pdf
```

3.1.25 Performance Module Document (PERF)

Introduction to PERF Performance analysis on Rockchip Platform

```
docs/en/Common/PERF/
├─ Rockchip_Develop_Guide_Linux_RealTime_Performance_Test_Report_EN.pdf
├─ Rockchip_Optimize_Tutorial_Linux_IO_EN.pdf
├─ Rockchip_Quick_Start_Linux_Perf_EN.pdf
├─ Rockchip_Quick_Start_Linux_Performance_Analyse_EN.pdf
├─ Rockchip_Quick_Start_Linux_Streamline_EN.pdf
└─ Rockchip_Quick_Start_Linux_Systrace_EN.pdf
```

3.1.26 GPIO Module Document (PINCTRL)

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

```
docs/en/Common/PINCTRL/
└─ Rockchip_Developer_Guide_Linux_Pinctrl_EN.pdf
```

3.1.27 PMIC Module Document (PMIC)

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

```
docs/en/Common/PMIC/
├─ Rockchip_RK805_Developer_Guide_EN.pdf
├─ Rockchip_RK806_Developer_Guide_EN.pdf
├─ Rockchip_RK808_Developer_Guide_EN.pdf
├─ Rockchip_RK809_Developer_Guide_EN.pdf
├─ Rockchip_RK816_Developer_Guide_EN.pdf
├─ Rockchip_RK817_Developer_Guide_EN.pdf
├─ Rockchip_RK818_Developer_Guide_EN.pdf
├─ Rockchip_RK818_RK816_Developer_Guide_Fuel_Gauge_EN.pdf
└─ Rockchip_RK818_RK816_Introduction_Fuel_Gauge_Log_EN.pdf
```

3.1.28 Power Module Document (POWER)

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

```
docs/en/Common/POWER/
└─ Rockchip_Developer_Guide_Power_Analysis_EN.pdf
```

3.1.29 PWM Module Document (PWM)

PWM developer guide on the Rockchip platform.

```
docs/en/Common/PWM
└─ Rockchip_Developer_Guide_Linux_PWM_EN.pdf
```

3.1.30 RGA Module Document (RGA)

RGA developer guide on the Rockchip platform.

```
docs/en/Common/RGA/
├─ Rockchip_Developer_Guide_RGA_EN.pdf
└─ Rockchip_FAQ_RGA_EN.pdf
```

3.1.31 SARADC Module Document (SARADC)

SARADC developer guide on the Rockchip platform.

```
docs/en/Common/SARADC/
└─ Rockchip_Developer_Guide_Linux_SARADC_EN.pdf
```

3.1.32 SPI Module Document (SPI)

SPI developer guide on the Rockchip platform.

```
docs/en/Common/SPI/  
└─ Rockchip_Developer_Guide_Linux_SPI_EN.pdf
```

3.1.33 Thermal Module Document (THERMAL)

Thermal developer guide on the Rockchip platform.

```
docs/en/Common/THERMAL/  
└─ Rockchip_Developer_Guide_Thermal_EN.pdf
```

3.1.34 Tools Module Document (TOOL)

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

```
docs/en/Common/TOOL/  
├─ Production-Guide-For-Firmware-Download.pdf  
├─ RKUpgrade_Dll_UserManual.pdf  
├─ Rockchip-User-Guide-ProductionTool-EN.pdf  
├─ Rockchip_Introduction_Partition_EN.pdf  
└─ Rockchip_User_Guide_Production_For_Firmware_Download_EN.pdf
```

3.1.35 Security Module Document (TRUST)

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

```
docs/en/Common/TRUST/  
├─ Rockchip_Developer_Guide_Trust_EN.pdf  
├─ Rockchip_RK3308_Developer_Guide_System_Suspend_EN.pdf  
├─ Rockchip_RK3399_Developer_Guide_System_Suspend_EN.pdf  
├─ Rockchip_RK356X_Developer_Guide_System_Suspend_EN.pdf  
├─ Rockchip_RK3576_Developer_Guide_System_Suspend_EN.pdf  
└─ Rockchip_RK3588_Developer_Guide_System_Suspend_EN.pdf
```

3.1.36 UART Module Document (UART)

Introduction to serial port functions and debugging on the Rockchip Platform

```
docs/en/Common/UART/  
├─ Rockchip_Developer_Guide_UART_EN.pdf  
└─ Rockchip_Developer_Guide_UART_FAQ_EN.pdf
```

3.1.37 UBOOT Module Document (UBOOT)

Introduction to U-Boot related development on the Rockchip platform

```
docs/en/Common/UBOOT/  
├─ Rockchip_Developer_Guide_Linux_AB_System_EN.pdf  
├─ Rockchip_Developer_Guide_U-Boot_TFTP_Upgrade_EN.pdf  
├─ Rockchip_Developer_Guide_UBoot_MMC_Device_Analysis_EN.pdf  
├─ Rockchip_Developer_Guide_UBoot_MTD_Block_Device_Design_EN.pdf  
├─ Rockchip_Developer_Guide_UBoot_Nextdev_EN.pdf  
└─ Rockchip_Introduction_UBoot_rkdevelop_vs_nextdev_EN.pdf
```

3.1.38 USB Module Document (USB)

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

```
docs/en/Common/USB/  
├─ Rockchip_Developer_Guide_Linux_USB_Initialization_Log_Analysis_EN.pdf  
├─ Rockchip_Developer_Guide_Linux_USB_PHY_EN.pdf  
├─ Rockchip_Developer_Guide_Linux_USB_Performance_Analysis_EN.pdf  
├─ Rockchip_Developer_Guide_USB2_Compliance_Test_EN.pdf  
├─ Rockchip_Developer_Guide_USB_EN.pdf  
├─ Rockchip_Developer_Guide_USB_FFS_Test_Demo_EN.pdf  
├─ Rockchip_Developer_Guide_USB_Gadget_UAC_EN.pdf  
├─ Rockchip_Developer_Guide_USB_SQ_Test_EN.pdf  
├─ Rockchip_Introduction_USB_SQ_Tool_EN.pdf  
├─ Rockchip_RK3399_Developer_Guide_USB_EN.pdf  
├─ Rockchip_RK3399_Developer_Guide_USB_DTS_EN.pdf  
├─ Rockchip_RK356x_Developer_Guide_USB_EN.pdf  
├─ Rockchip_RK3588_Developer_Guide_USB_EN.pdf  
├─ Rockchip_Trouble_Shooting_Linux4.19_USB_Gadget_UVC_EN.pdf  
└─ Rockchip_Trouble_Shooting_Linux_USB_Host_UVC_EN.pdf
```

3.1.39 Watchdog Module Document (WATCHDOG)

Development instructions for Watchdog on the Rockchip platform.

```
docs/en/Common/WATCHDOG/  
└─ Rockchip_Developer_Guide_Linux_WDT_EN.pdf
```

3.2 Linux System Development Documents (Linux)

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

```
|— ApplicationNote
|— Audio
|— Camera
|— DPDK
|— Docker
|— Graphics
|— Multimedia
|— Profile
|— Recovery
|— Security
|— System
|— Uefi
└— Wifibt
```

3.2.1 ApplicationNote

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

```
docs/en/Linux/ApplicationNote/
|— Rockchip_Developer_Guide_Linux_Flash_Open_Source_Solution_EN.pdf
|— Rockchip_Instruction_Linux_ROS2_EN.pdf
|— Rockchip_Instruction_Linux_ROS_EN.pdf
|— Rockchip_Quick_Start_Linux_USB_Gadget_EN.pdf
└— Rockchip_Use_Guide_Linux_RetroArch_EN.pdf
```

3.2.2 Audio Development Documents (Audio)

Self developed audio algorithm on Rockchip platform.

```
docs/en/Linux/Audio/
|— Rockchip_Developer_Guide_Microphone_Array_TEST_EN.pdf
|— Rockchip_Developer_Guide_Microphone_Array_Tuning.pdf
└— Rockchip_Introduction_Linux_Audio_3A_Algorithm_EN.pdf
```

3.2.3 Camera Development Documents (Camera)

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

```
docs/en/Linux/Camera/
|— Rockchip_Developer_Guide_Linux4.4_Camera_EN.pdf
|— Rockchip_Developer_Guide_Linux_RMSL_EN.pdf
└— Rockchip_Trouble_Shooting_Linux4.4_Camera_EN.pdf
└— Rockchip_Trouble_Shooting_Linux5.10_Camera_EN.pdf
```

3.2.4 Docker Development Documents (Docker)

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

```
docs/en/Linux/Docker/
├─ Rockchip_Developer_Guide_Debian_Docker_EN.pdf
├─ Rockchip_Developer_Guide_Linux_Docker_Deploy_EN.pdf
└─ Rockchip_User_Guide_SDK_Docker_EN.pdf
```

3.2.5 Graphics Development Documents (Graphics)

Linux Graphics related development on Rockchip Platform.

```
docs/en/Linux/Graphics/
├─ Rockchip_Developer_Guide_Buildroot_Weston_EN.pdf
├─ Rockchip_Developer_Guide_Linux_Graphics_EN.pdf
└─ Rockchip_Developer_Guide_Linux_LVGL_EN.pdf
```

3.2.6 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/en/Linux/Multimedia/
├─ Rockchip_Developer_Guide_Linux_RKADK_EN.pdf
├─ Rockchip_User_Guide_Linux_Gstreamer_EN.pdf
└─ Rockchip_User_Guide_Linux_Rockit_EN.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_EN.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
RK3576	3840x2160@60fps	7680x4320@30fps	7680x4320@30fps	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
RK3576	4096x2304@60fps	4096x2304@60fps	N/A
RK3566/RK3568	1920x1088@60f	1920x1088@60f	N/A
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

3.2.7 SDK Profile introduction (Profile)

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


```
docs/en/Linux/Profile/  
├─ Rockchip_Developer_Guide_Linux_PCBA_EN.pdf  
├─ Rockchip_Introduction_Linux_Benchmark_KPI_EN.pdf  
├─ Rockchip_Introduction_Linux_PLT_EN.pdf  
└─ Rockchip_User_Guide_Linux_Software_Test_EN.pdf
```

3.2.8 OTA Upgrade (Recovery)

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

```
docs/en/Linux/Recovery/  
├─ Rockchip_Developer_Guide_Linux_DFU_Upgrade_EN.pdf  
├─ Rockchip_Developer_Guide_Linux_Recovery_EN.pdf  
├─ Rockchip_Developer_Guide_Linux_Upgrade_EN.pdf  
└─ Rockchip_Introduction_Smart_Screen_OTA_EN.pdf
```

3.2.9 Security Solution (Security)

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

```
docs/en/Linux/Security/  
├─ Rockchip_Developer_Guide_Linux_Secure_Boot_EN.pdf  
└─ Rockchip_Developer_Guide_TEE_SDK_EN.pdf
```

3.2.10 System Development (System)

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

```
docs/en/Linux/System/  
├─ Rockchip_Developer_Guide_Buildroot_EN.pdf  
├─ Rockchip_Developer_Guide_Debian_EN.pdf  
└─ Rockchip_Developer_Guide_Third_Party_System_Adaptation_EN.pdf
```

3.2.11 UEFI Booting (UEFI)

Introduction to UEFI boot solution on Rockchip Linux platform.

```
docs/en/Linux/Uefi/  
└─ Rockchip_Developer_Guide_UEFI_EN.pdf
```

3.2.12 Network Module (RKWIFI BT)

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

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

3.2.13 DPDK Module (DPDK)

DPDK development guide on Rockchip Linux platform.

```
docs/en/Linux/DPDK/  
└─ Rockchip_Developer_Guide_Linux_DPDK_EN.pdf
```

3.3 Chip Platform Related Documents (Socs)

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

3.3.1 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/en/<chipset_name>` directory:

```
Rockchip_<chipset_name>_Linux_SDK_Release_<version>_EN.pdf
```

3.3.2 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/en/<chipset_name>/Quick-start` directory.

3.3.3 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_EN.pdf" document can be obtained from the `/docs/en/<chip_name>/` directory and will be continuously improved and updated.

3.4 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/en/<chipset_name>/Datasheet` directory.

3.4.1 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/en/<chip_name>/Hardware` directory.

3.5 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/en/Others` directory.

```
docs/en/Others/  
├─ Rockchip_Developer_Guide_Repo_Mirror_Server_Deploy_EN.pdf  
├─ Rockchip_Trouble_Shooting_Linux_Real-Time_Performance_EN.pdf  
├─ Rockchip_User_Guide_Bug_System_EN.pdf  
└─ Rockchip_User_Guide_SDK_Application_And_Synchronization_EN.pdf
```

3.6 Documents List (docs_list_en.txt)

Please refer to the document in the `/docs/en/docs_list_en.txt` directory.

```
├─ Common  
├─ Linux  
├─ Others  
├─ Rockchip_Developer_Guide_Linux_Software_EN.pdf  
├─ <chipset_name>  
└─ docs_list_en.txt
```

4. Chapter-4 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
BoardProofTool	Anti-Piracy Tool
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
programming_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`

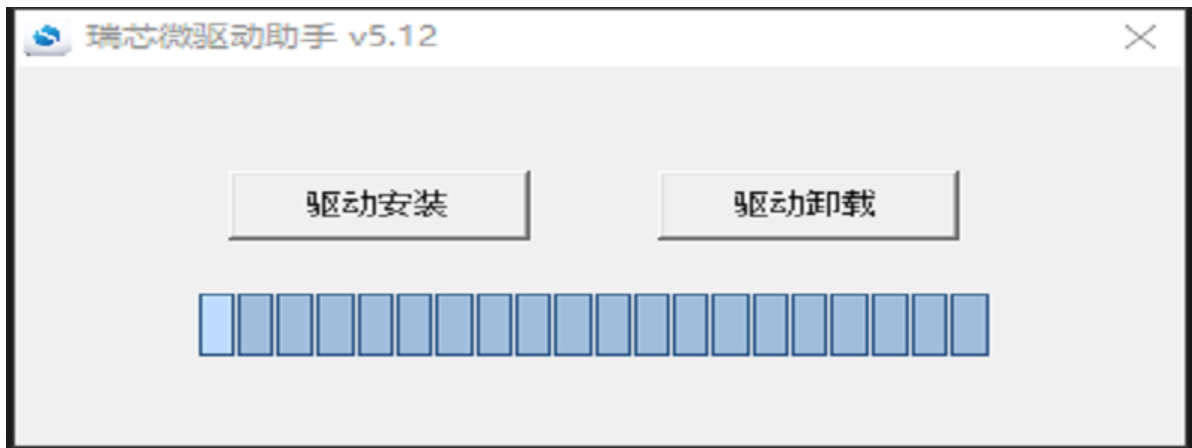
Tool name	Description
boot_merger	Packing or unpacking loader tool
upgrade_tool	Command line upgrade tool
rk_sign_tool	Secureboot signature tool

4.1 Driver Installation Tool

Rockchip USB driver installation assistant is stored in

`<SDK>/tools/windows/DriverAssitant_v5.13.zip`. xp, win7_32, win7_64, win10_32, win10_64 and other operating systems are supported.

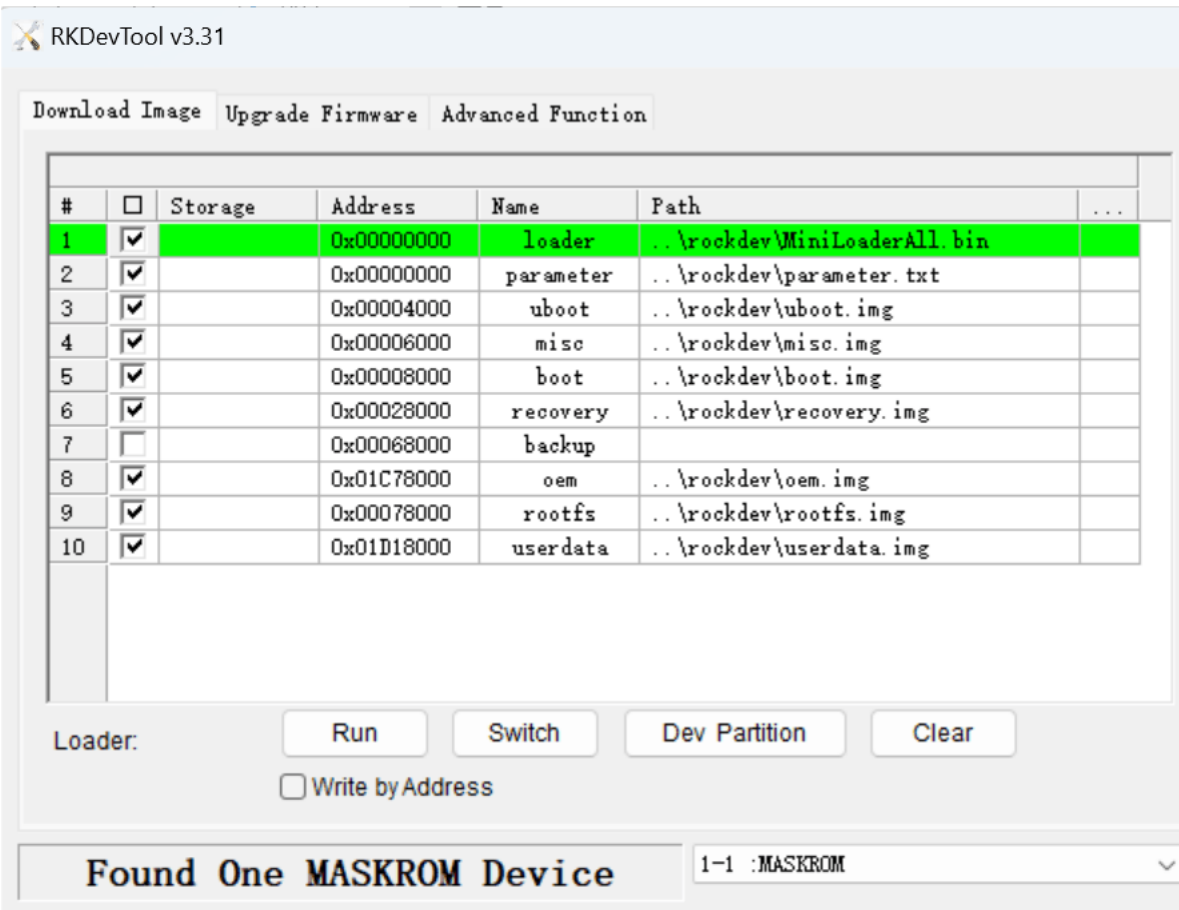
The installation steps are as follows:



4.2 Burning Tools

- The SDK provides Windows burning tools (the tool version requires V3.31 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 V2.26 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> <BlockLen> [--Force]
RunSystem:          RUN  <uboot_addr> <trust_addr> <boot_addr> <uboot> <trust> <boot>
-----

```

4.3 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
```

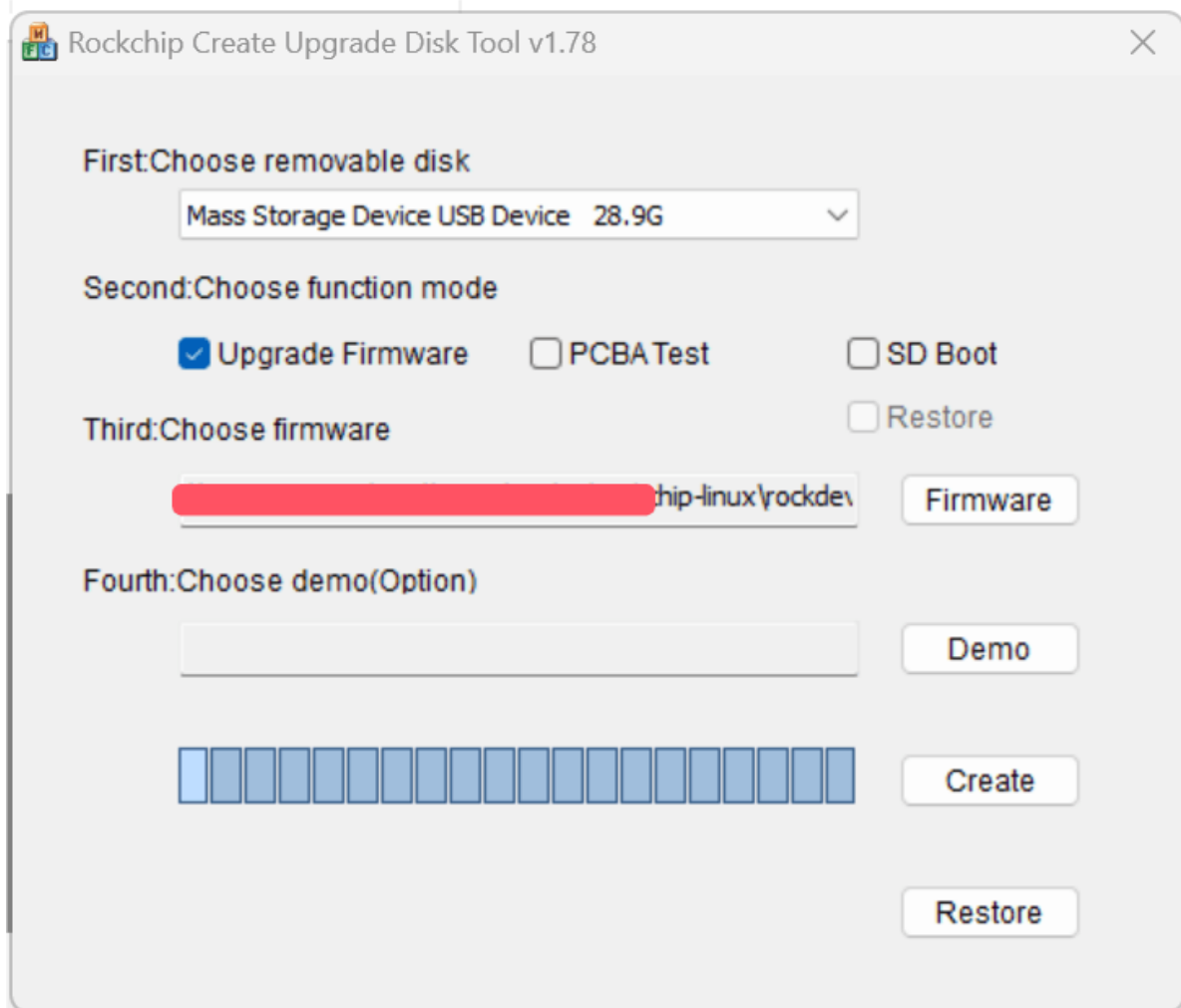
- How to package update.img firmware in Linux environment, run the following command to generate update.img

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

4.4 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.

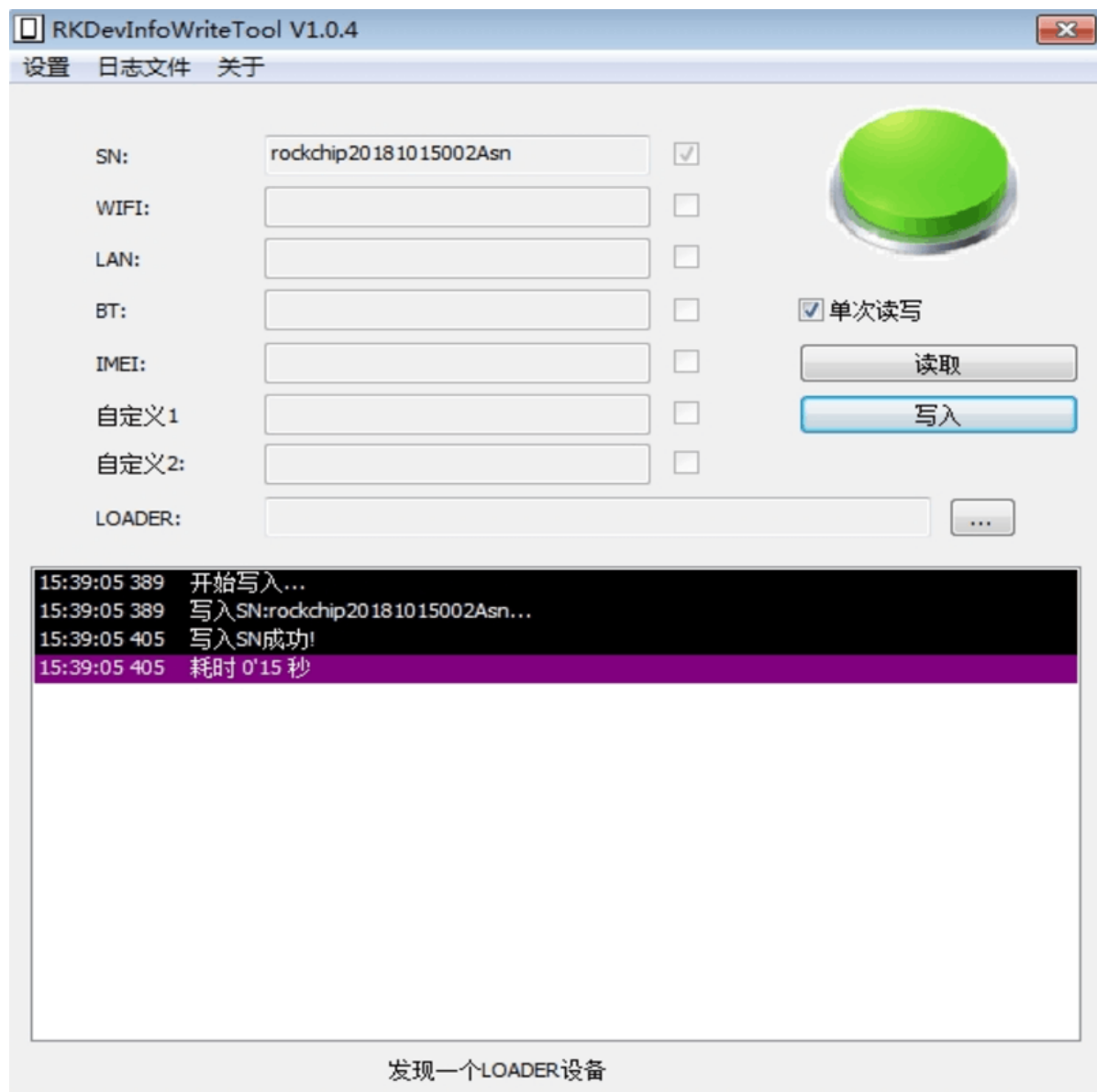
```
<SDK>/tools/windows/SDDiskTool_v1.78.zip
```



4.5 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_EN.pdf](#) in the current directory for tool usage,.

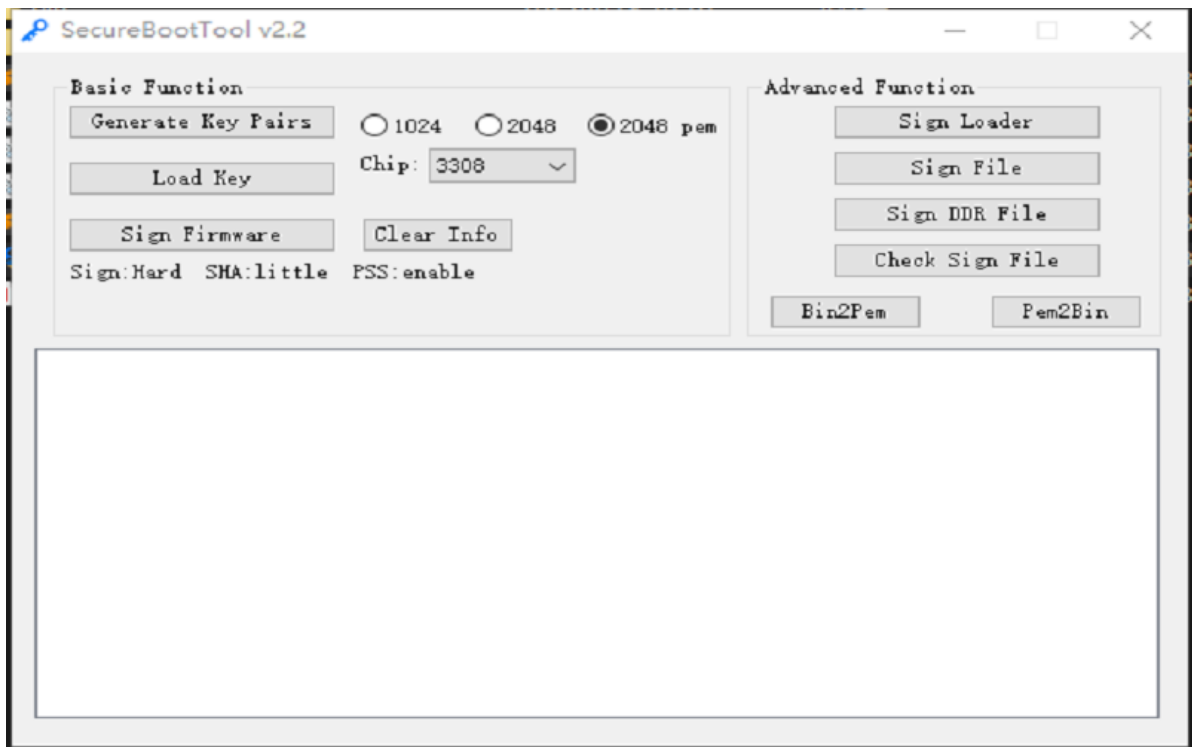


4.6 Firmware Signature Tool

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/en/Linux/Security/Rockchip_Developer_Guide_Linux4.4_SecureBoot_EN.pdf for details.

4.7 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
```

```

PS D:\Rockchip\vs_projects\programmer_image_tool> .\programmer_image_tool -h
NAME
    programmer_image_tool - creating image for programming on flash
SYNOPSIS
    programmer_image_tool [-iotbpsvh]
    This tool aims to convert firmware into image for programming
    From now on, it can support slc nand(rk)|spi nand|nor|emmc.
OPTIONS:
    -i    input firmware
    -o    output directory
    -t    storage type,range in[SLC|SPINAND|SPINOR|EMMC]
    -b    block size,unit KB
    -p    page size,unit KB
    -s    oob size,unit B
    -2    2k data in one page
    -l    using page linked list
    -v    show version

```

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.

4.8 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.

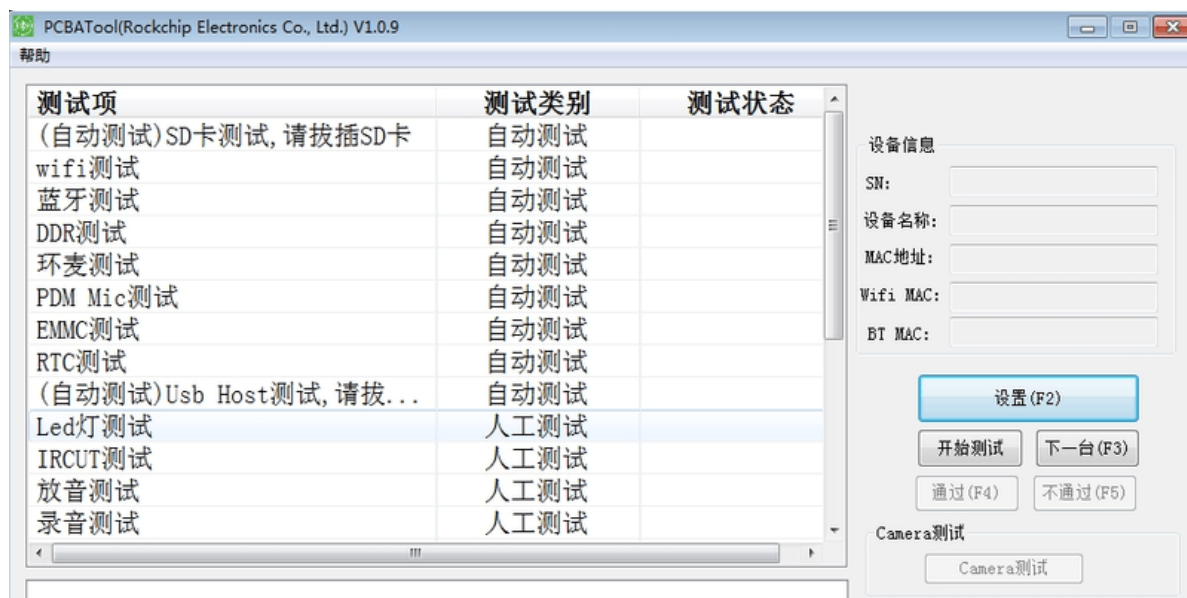
These test items include automatic test items and manual test items. Wireless network, DDR/eMMC, and Ethernet are automatic test items. Button, SD card, USB HOST, Codec are manual test items.

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



4.9 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
```

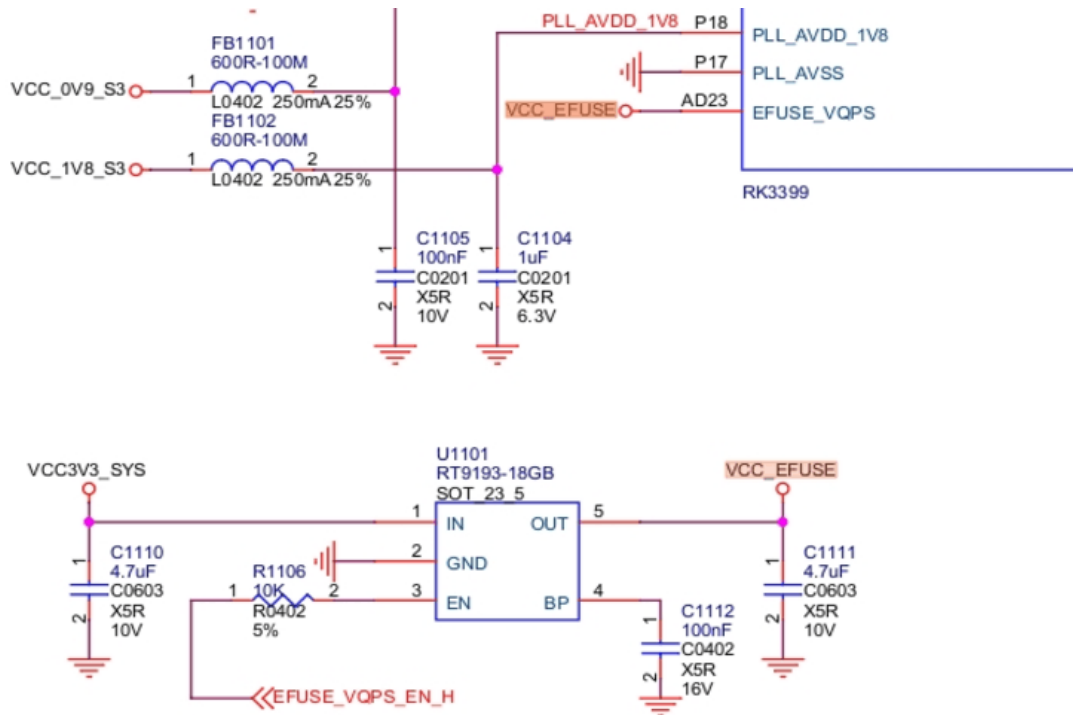


4.10 eFuse Programming Tool

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

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

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.

4.11 Mass Production Upgrade Tool

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

```
<SDK>/tools/windows/FactoryTool_v1.91.zip
```

4.12 Partition Modification Tool

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

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

Parameter:

parameter.txt

Browse

Name	Offset	Secotr Offset	Size	
uboot	0x800000	0x4000	4MB	
trust	0xC00000	0x6000	4MB	
misc	0x1000000	0x8000	4MB	
boot	0x1400000	0xA000	32MB	
recovery	0x3400000	0x1A000	32MB	
backup	0x5400000	0x2A000	32MB	
oem	0x7400000	0x3A000	64MB	
rootfs	0xB400000	0x5A000	6144MB	
userdata:g...	0x18B400000	0xC5A000	-	

Offset:

Size:

☒ KB

☐ MB

Name:

Modify

Revoke

Save

5. Chapter-5 SDK Software Framework

5.1 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.

5.1.1 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:

5.1.2 Buildroot

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.

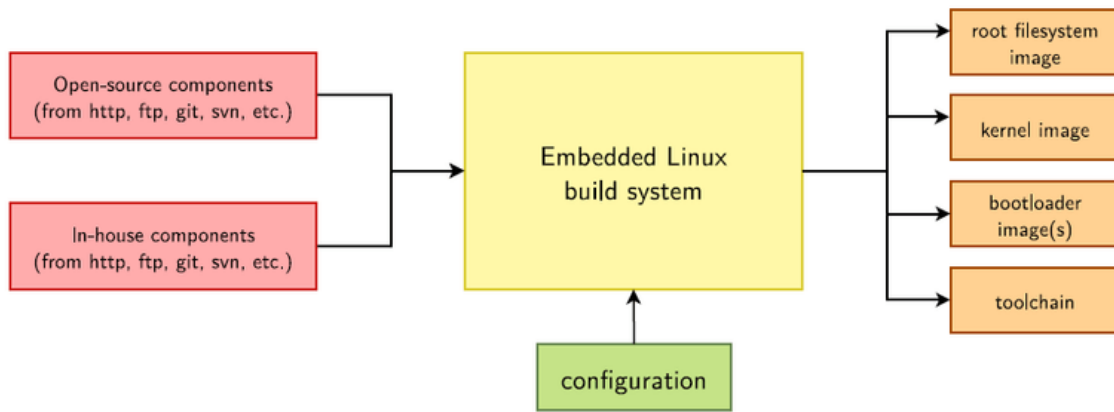


Figure 1-1 Buildroot Compilation Block Diagram

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.

5.1.3 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.

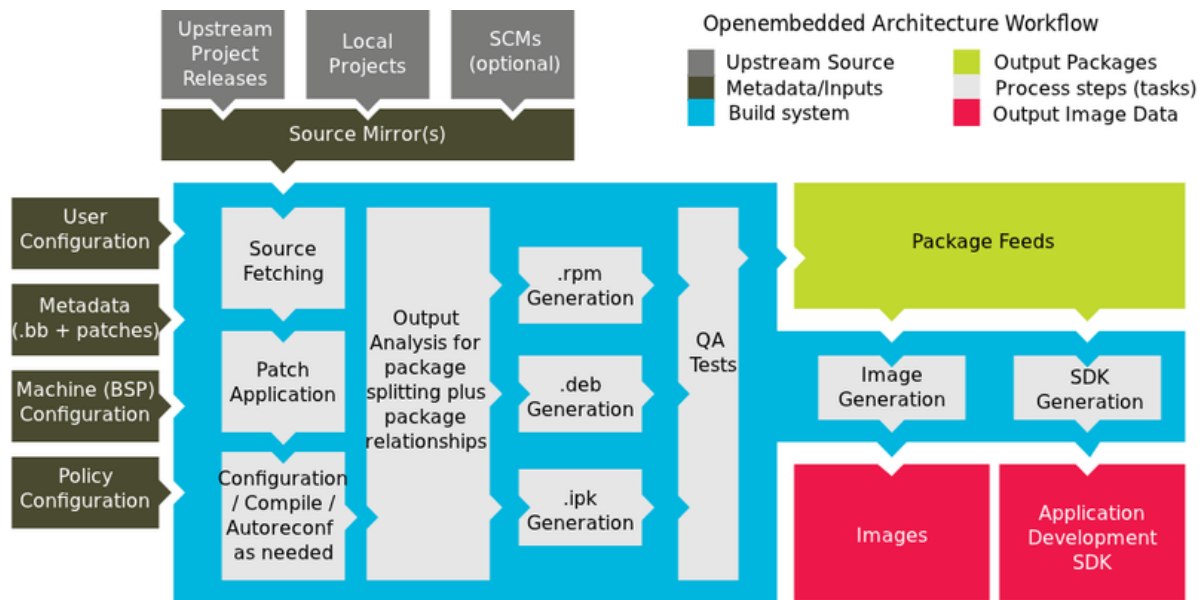


Figure 2-1 Yocto Compilation block diagram

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.

5.1.4 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.

5.2 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.

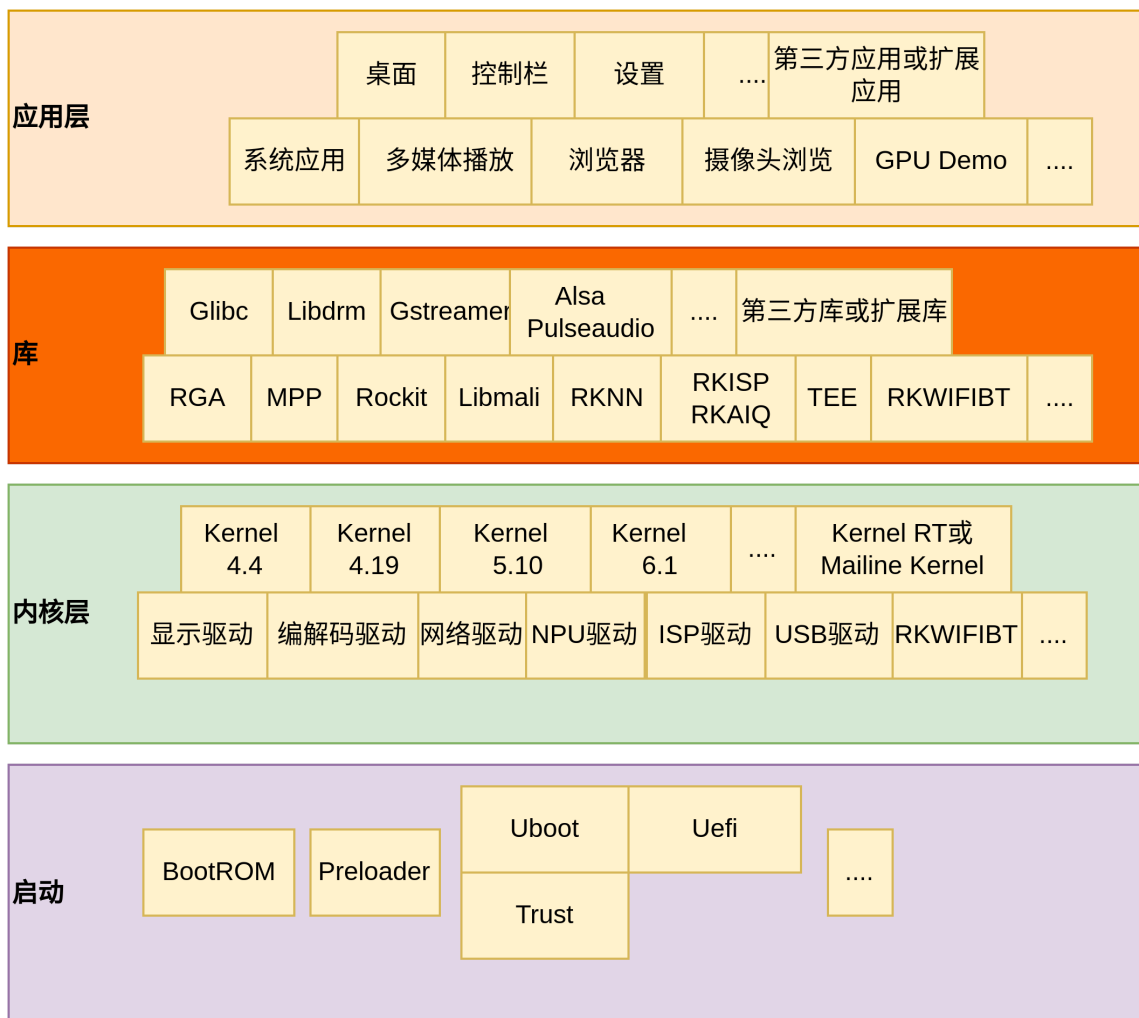
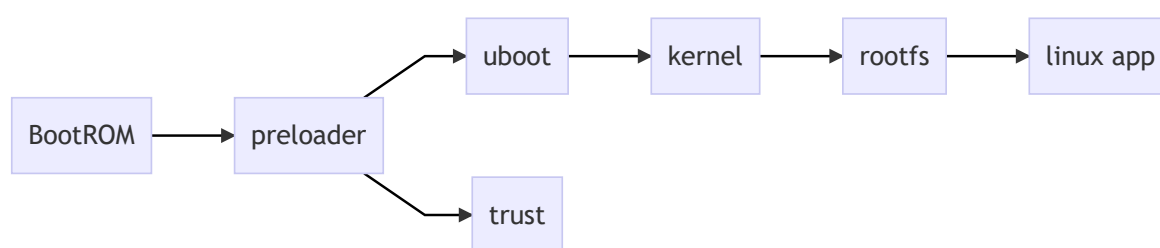


Figure 3-1 SDK Software Block Diagram

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.

5.3 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.

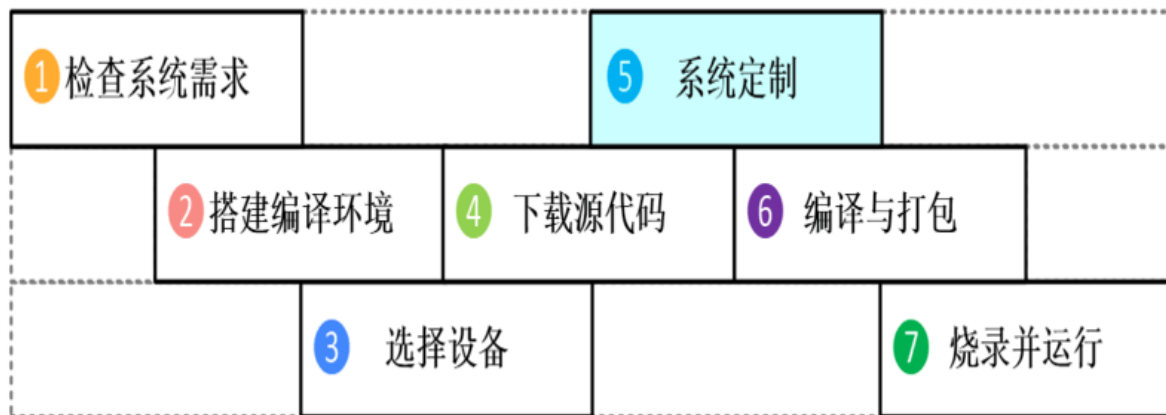


Figure 4-1 SDK Development Process

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](#).

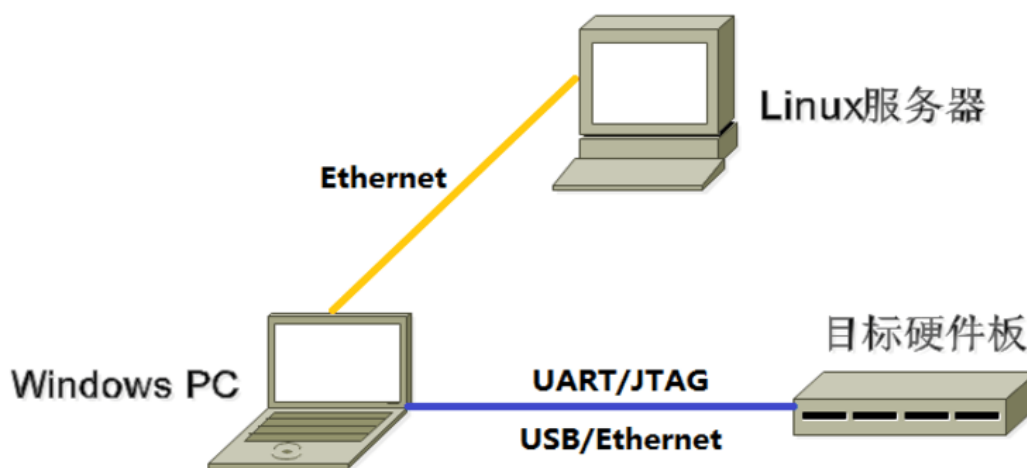
6. Chapter-6 SDK Development Environment Setup

6.1 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.

6.2 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.

6.2.1 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"
```

6.2.2 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
```

6.2.3 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.

6.2.3.1 SDK Download Command

Rockchip Linux SDK can be adapted to different chip platforms, such as RK3588, RK3576, 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.

6.2.3.2 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.

6.2.3.3 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/en/RK3588/RK3588_Linux5.10_SDK_Note.md
```

6.2.3.4 SDK Update

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

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

6.2.3.5 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/en/Others/Rockchip_User_Guide_SDK_Application_And_Synchronization_EN.pdf
```

6.3 Setting Up a Linux Development Environment

6.3.1 Preparing the Development Environment

We recommend using a system with Ubuntu 22.04 or a higher version for compilation. Other Linux versions may require corresponding adjustments to the software packages. In addition to system requirements, there are other hardware and software requirements.

Hardware Requirements: 64-bit system with hard disk space greater than 40GB. If you are performing multiple builds, you will need more hard disk space.

Software Requirements: System with Ubuntu 22.04 or a higher version.

6.3.2 Installing Libraries and Toolkits

When developing for devices using the command line, you can install the necessary libraries and tools required for compiling the SDK by following these steps.

Use the following `apt-get` commands to install the libraries and tools needed for subsequent 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 suitable for Ubuntu 22.04. For other versions, please use the corresponding installation command based on the package name. If you encounter errors during compilation, you can install the corresponding software packages based on the error messages. Specifically:

- Python requires the installation of version 3.6 or higher, with version 3.6 used as an example here.
- Make requires the installation of version 4.0 or higher, with version 4.2 used as an example here.
- LZ4 requires the installation of version 1.7.3 or higher.
- Compiling Yocto requires a VPN network.

6.3.2.1 Checking and Upgrading the Host's Python Version

The method to check and upgrade the host's Python version is as follows:

- Check the host's Python version

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

If the requirement of Python version ≥ 3.6 is not met, you can upgrade by the following method:

- Upgrade to the new version of `Python 3.6.15`

```
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"
```

6.3.2.2 Checking and Upgrading the Host's `make` Version

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

- Checking the host's `make` version

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

- Upgrading to a newer version of `make 4.2`

```
$ 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
```

6.3.2.3 Checking and Upgrading the Host's LZ4 Version

The method to check and upgrade the host's LZ4 version is as follows:

- Check the host's LZ4 version

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

- Upgrade to a 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
```

6.4 Window PC Development Environment Setup

6.4.1 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)

6.4.2 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:



6.4.3 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.

6.4.4 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.

6.5 Docker Environment Setup

To help developers quickly complete the complex development environment preparation mentioned above, we also provide a cross-compiler Docker image. This enables customers to rapidly verify and then shorten the build time of the compilation environment.

Before using the Docker environment, you can refer to the following document for instructions:

```
<SDK>/docs/en/Linux/Docker/Rockchip_Developer_Guide_Linux_Docker_Deploy_EN.pdf.
```

The following systems have been verified:

Distribution Version	Docker Version	Image Loading	Firmware Compilation
ubuntu 22.04	24.0.5	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)

The Docker image can be obtained from the website [Docker Image](#).

6.6 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.

6.6.1 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-

Linux5.10/6.1 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)

Linux5.10/6.1 SDK:
gcc version 10.3.1 20210621 (GNU Toolchain for the A-profile Architecture 10.3-2021.07 (arm-10.29))
```

6.6.2 Buildroot Toolchain

6.6.2.1 Configuring the Compilation Environment

To compile individual modules or third-party applications, a cross-compilation environment must be set up. For instance, for the RK3576, the cross-compilation tools are located in the `buildroot/output/rockchip_rk3576/host/usr` directory. It is necessary to set the `bin/` directory of the tools and the `aarch64-buildroot-linux-gnu/bin/` directory as environment variables. Execute the script for automatic configuration of the environment variables in the top-level directory:

```
source buildroot/envsetup.sh rockchip_rk3576
```

Enter the command to check:

```
cd buildroot/output/rockchip_rk3576/host/usr/bin
./aarch64-linux-gcc --version
```

At this point, the following information will be printed:

```
aarch64-linux-gcc.br_real (Buildroot -gc307c95550) 12.3.0
```

6.6.2.2 Packaging Toolchain

Buildroot supports the packaging of the built-in toolchain into a compressed archive for standalone compilation by third-party applications. For detailed information on how to package the toolchain, please refer to the Buildroot official documentation:

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

In the SDK, you can directly run the following command to generate the toolchain package:

```
./build.sh bmake:sdk
```

The generated toolchain package is located in the `buildroot/output/*/images/` directory, named `aarch64-buildroot-linux-gnu_sdk-buildroot.tar.gz`, for users who have the need. After extraction, the path to `gcc` will be:

```
./aarch64-buildroot-linux-gnu_sdk-buildroot/bin/aarch64-buildroot-linux-gnu-gcc
```

6.6.3 Debian Toolchain

Utilize Docker on the machine side, gcc, or `dpkg-buildpackage` for the relevant compilation.

6.6.4 Yocto Toolchain

Refer to the following:

- [Building your own recipes from first principles](#)
- [New Recipe](#)

7. Chapter-7 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.

7.1 SDK Naming Rules

7.1.1 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

7.1.2 SDK Release Document Naming Rules

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

7.1.3 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:

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

7.2 SDK Update Mechanism

7.2.1 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).

7.2.2 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](#)

7.3 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/en/Others/Rockchip_Developer_Guide_Repo_Mirror_Server_Deploy.pdf
```

8. Chapter-8 SDK Compilation Instructions

The SDK can be configured and compiled with `make` or `./build.sh` along with target parameters. For specific references, see the compilation instructions in `device/rockchip/common/README.md`.

To ensure SDK updates smoothly each time, it is recommended to clean previous compilation products before updating. This avoids potential compatibility issues or compilation errors, as old compilation products may not be suitable for the new version of the SDK. Run the following command to clean these compilation products:

```
./build.sh cleanall.
```

8.1 SDK Compilation Command

`make help`, for example:

```
$ make help
menuconfig          - interactive curses-based configurator
oldconfig           - resolve any unresolved symbols in .config
synconfig           - Same as oldconfig, but quietly, additionally update
deps
olddefconfig        - Same as synconfig 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
##### Rockchip Linux SDK #####

Manifest: rk3562_linux_release_v1.1.0_20231220.xml

Log colors: message notice warning error fatal

Usage: build.sh [OPTIONS]
Available options:
chip[:<chip>[:<config>]]      choose chip
defconfig[:<config>]         choose defconfig
config                        modify SDK defconfig
...
updateimg                    build update image
otapackage                    build A/B OTA update image
all                           build images
release                       release images and build info
save                          alias of release
all-release                   build and release images
allsave                       alias of all-release
shell                         setup a shell for developing
```

```

cleanall          cleanup
clean[:module[:module]]...  cleanup modules
post-rootfs <rootfs dir>    trigger post-rootfs hook scripts
help              usage

Default option is 'all'.

```

8.2 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_evb7_lp4_v11_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 <code>make lunch</code> or <code>./build.sh lunch</code> 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.

8.3 SDK Customization Configuration

The SDK can be configured through `make menuconfig`, with the main components currently available for configuration being:

```
(rk3562) SoC
  Rootfs  --->
  Loader (u-boot)  --->
  RTOS  --->
  Kernel  --->
  Boot  --->
  Recovery (based on buildroot)  --->
  PCBA test (based on buildroot)  --->
  Security  --->
  Extra partitions  --->
  Firmware  --->
  Update (Rockchip update image)  --->
  Others configurations  --->
```

- Rootfs: This represents the "Root File System". Here, you can choose different root file systems like Buildroot, Yocto, Debian, etc.
- Loader (u-boot): This is the boot loader configuration, usually u-boot, which initializes hardware and loads the main operating system.
- RTOS: Real-Time Operating System (RTOS) configuration options, suitable for applications requiring real-time performance.
- Kernel: Configure kernel options here, customizing a Linux kernel suitable for your hardware and application needs.
- Boot: Configure Boot partition support formats here.
- Recovery (based on buildroot): This is the configuration for the recovery environment based on buildroot, used for system recovery and upgrade.
- PCBA test (based on buildroot): This is a configuration for a PCBA (Printed Circuit Board Assembly) testing environment based on buildroot.
- Security: Indicates that certain security features will depend on the buildroot configuration chosen for Rootfs, mainly the Secureboot feature activation.
- Extra partitions: Used to configure additional partitions.
- Firmware: Configure firmware-related options here.
- Update (Rockchip update image): Used to configure options for Rockchip's complete firmware.
- Others configurations: Other additional configuration options.

The `make menuconfig` interface provides a text-based user interface to select and configure various options. Once configuration is complete, use the command `make savedefconfig` to save these settings, so that custom builds will be done based on these settings.

With the above configs, you can choose different Rootfs/Loader/Kernel configurations for various custom builds, allowing flexible selection and configuration of system components to meet specific needs.

8.4 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.

8.5 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
```

8.6 Build Modules

8.6.1 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.

8.6.2 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.

8.6.3 Build Recovery

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

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

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

```
output/rockchip_rk3562_recovery/images.
```

Note: recovery.img includes Kernel, so every time Kernel changes, Recovery needs to be repackaged. The method for repackaging Recovery is as follows:

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

Note: Recovery is not a mandatory feature, some board configurations might not set it

8.6.4 Build Buildroot

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

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

After compilation, different formats of images are generated in the Buildroot directory

```
output/rockchip_rk3562/images, rootfs.ext4 format is used by default.
```

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

```
<SDK>/docs/en/Linux/System/Rockchip_Developer_Guide_Buildroot_EN.pdf
```

8.6.5 Build Debian

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

After compilation, a linaro-rootfs.img is generated in the debian directory.

Note: Related dependencies must be installed beforehand

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

For more information, please refer to the Debian development document:

```
<SDK>/docs/en/Linux/System/Rockchip_Developer_Guide_Debian_EN.pdf
```

8.6.6 Build Yocto

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

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

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

The default login username is root. For more information on Yocto, please refer to [Rockchip Wiki](#).

FAQ:

- If you encounter the following issue during compilation:

```
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](#).

8.6.7 Cross-Compilation

8.6.7.1 SDK Directory Built-in Cross-Compilation

The SDK prebuilts directory built-in cross-compilation are as follows:

Contents	Description
prebuilts/gcc/linux-x86/aarch64/gcc-arm-10.3-2021.07-x86_64-aarch64-none-linux-gnu	gcc arm 10.3.1 64-bit toolchain
prebuilts/gcc/linux-x86/arm/gcc-arm-10.3-2021.07-x86_64-arm-none-linux-gnueabi	gcc arm 10.3.1 32-bit toolchain

The toolchain can be downloaded from the following address:

[Click here](#)

8.6.7.2 Built-in Cross Compilation in Buildroot

Different chip and target feature configurations can be set via `source buildroot/envsetup.sh`

```
$ source buildroot/envsetup.sh  
Top of tree: rk3562  
  
You're building on Linux  
Lunch menu...pick a combo:  
  
44. rockchip_rk3562  
45. rockchip_rk3562_32  
46. rockchip_rk3562_dictpen
```



```
47. rockchip_rk3562_ramboot
48. rockchip_rk3562_recovery
49. rockchip_rk3562_robot
...

Which would you like? [1]:
```

Default choice is 44, `rockchip_rk3562`. Then, enter the RK3562 Buildroot directory to start compiling related modules.

`rockchip_rk3562_32` is for compiling a 32-bit Buildroot system, and `rockchip_rk3562_recovery` is used to compile the Recovery module.

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

Enter the buildroot directory

```
SDK#cd buildroot
```

- Delete and recompile rockchip-test

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

- Recompile rockchip-test

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

- Delete rockchip-test

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

To compile an individual module or third-party application, a cross-compilation environment needs to be configured. For instance, for RK3562, its cross-compilation tools are located in `buildroot/output/rockchip_rk3562/host/usr`. You need to set the `bin/` directory of the tool and `aarch64-buildroot-linux-gnu/bin/` as environment variables and run the script for auto-configuring environment variables in the top-level directory:

```
source buildroot/envsetup.sh rockchip_rk3562
```

Enter the command to check,:

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

The following information will be printed:

```
aarch64-linux-gcc.br_real (Buildroot) 12.3.0
```

```
Save to the rootfs configuration file
buildroot$ make update-defconfig
```

9. Chapter-9 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

9.1 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.

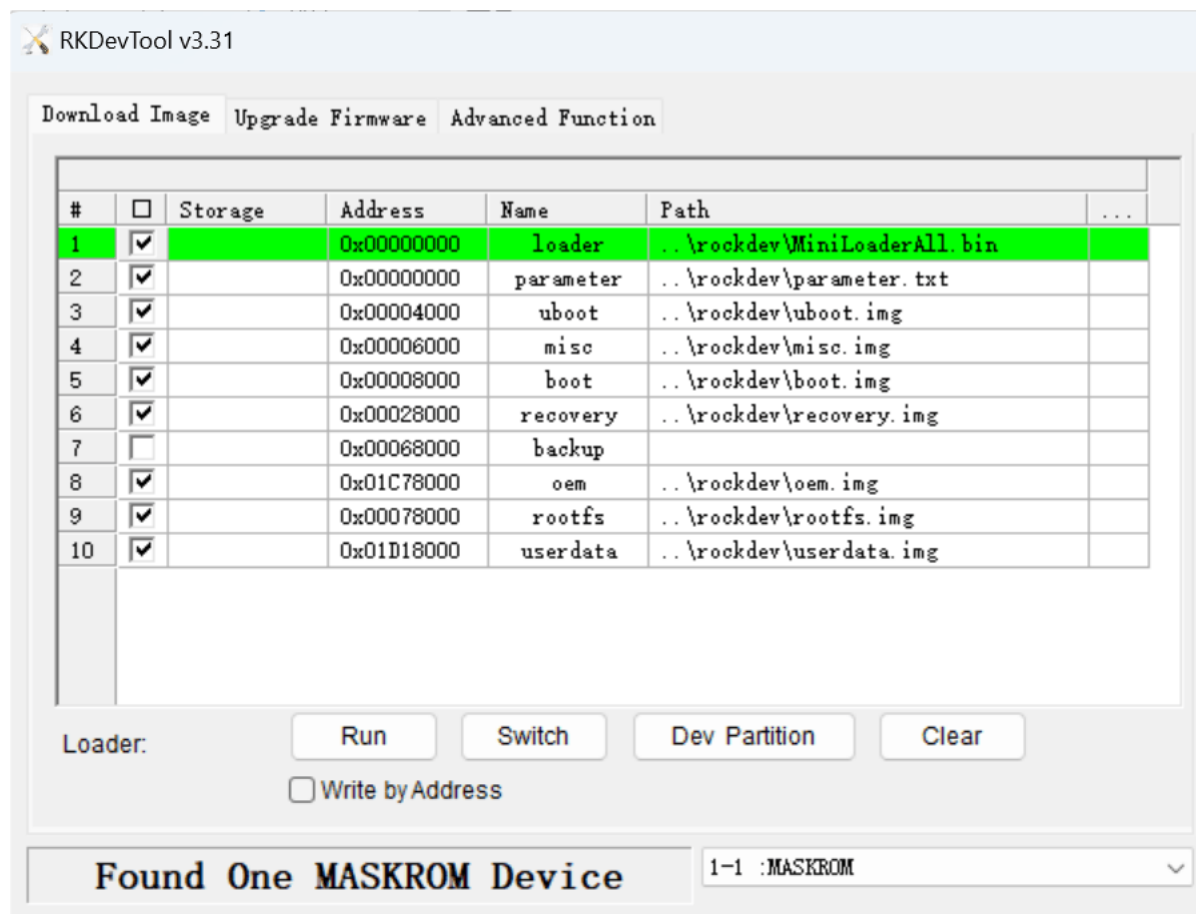
9.1.1 Windows Flashing Instructions

Rockchip's SDK provides a Windows burning tool (the tool version requires V3.31 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_v5.13.zip
```

9.1.2 Linux Flashing Instructions

The Linux burning tool is located in the tools/linux directory (Linux_Upgrade_Tool tool version requires V2.26 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 rockdev/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
```

9.1.3 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.

10. Chapter-10 SDK Development

Core components development in SDK, such as U-Boot, Kernel, Recovery, Buildroot, Debian, Yocto, Multivideo, Graphics, Rootfs post-processing, Overlays, and other key component development.

10.1 Bootloader Development

Bootloader is a type of software that runs before the operating system starts. It is responsible for initializing hardware devices, configuring system parameters, and loading the operating system kernel. Bootloader development on the Rockchip platform typically involves two main components: U-Boot and UEFI (Unified Extensible Firmware Interface).

10.1.1 U-Boot Development

This section provides a brief introduction to the basic concepts of U-Boot and considerations for compilation, assisting customers in understanding the U-Boot framework for RK platforms. For specific details on U-Boot development, refer to `<SDK>/docs/en/Common/U-Boot/Rockchip-Developer-Guide-UBoot-*.pdf`.

10.1.1.1 Introduction to U-Boot

The v2017(next-dev) is a development branch derived from the official v2017.09 release of U-Boot by RK, which now supports all mainstream RK chips currently on the market. The main features supported include:

- Support for booting RK Android firmware;
- Support for booting Android AOSP firmware;
- Support for booting Linux Distro firmware;
- Support for Rockchip miniloader and two types of Pre-loaders, SPL/TPL;
- Support for display devices such as LVDS, EDP, MIPI, HDMI, CVBS, and RGB;
- Support for booting from storage devices including eMMC, Nand Flash, SPI Nand flash, SPI NOR flash, SD card, and USB flash drive;
- Support for FAT, EXT2, and EXT4 file systems;
- Support for GPT and RK parameter partition tables;
- Support for boot logo, charging animation, low battery management, and power management;
- Support for I2C, PMIC, CHARGE, FUEL GUAGE, USB, GPIO, PWM, GMAC, eMMC, NAND, and Interrupt;
- Support for Vendor storage to save user data and configurations;
- Support for RockUSB and Google Fastboot for USB gadget flashing of eMMC;
- Support for USB devices such as Mass storage, ethernet, and HID;
- Support for dynamically selecting kernel DTB based on hardware status.

10.1.1.2 Version

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

Method 1: Confirm whether the version number of the 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 of the formal print at boot is U-Boot 2017.09.

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

Project Open Source: v2017 is open source and regularly updated to Github: <https://github.com/rockchip-linux/u-boot>

Kernel Version: v2017 requires RK kernel version ≥ 4.4

10.1.1.3 Preliminary Preparation

- Download rkbin

This is a repository of tools, used to store the non-open source bin, scripts, and packaging tools of RK. When compiling U-Boot, it indexes the relevant files from this repository to package and generate the loader, trust, and uboot firmware. The rkbin and U-Boot projects must maintain the same directory level relationship.

Repository Download: Please refer to the appendix section.

- Download GCC

The GCC compiler uses gcc-linaro-6.3.1, which is placed inside the prebuilts directory. The prebuilts and U-Boot maintain the same directory level relationship. 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/
```

- Select defconfig

The support status of defconfig for each platform (subject to the SDK release):

"[Chip]_defconfig" or "[Chip].config" are usually full-featured versions, and the rest are specific feature versions.

1. The support status of defconfig for each platform (subject to the SDK release):

"[Chip]_defconfig" or "[Chip].config" are usually full-featured versions, and the rest are specific feature versions.

Chip	defconfig	Support for Kernel DTB	Description
RK3036	rk3036_defconfig	Yes	General version
RK3128x	rk3128x_defconfig	Yes	General version
RK3126	rk3126_defconfig	Yes	General version
RK3288	rk3288_defconfig	Yes	General version
RK3328	rk3328_defconfig	Yes	General version
RK3399	rk3399_defconfig	Yes	General version
RK3399Pro	rk3399pro_defconfig	Yes	General version
RK3399Pro-npu	rknpu-lion_defconfig	Yes	General version
RK3308	rk3308_defconfig rk3308-aarch32_defconfig	Yes	General version Supports aarch32 mode
PX30	px30_defconfig	Yes	General version
RK3326	rk3326_defconfig rk3326-aarch32_defconfig	Yes	General version Supports aarch32 mode
RK3568	rk3568_defconfig rk3568-dfu.config rk3568-nand.config rk3568-spl-spi-nand_defconfig rk3568-aarch32.config rk3568-usbplug.config	Yes	General version Supports DFU Supports MLC/TLC/eMMC SPI-nand specific SPL Supports aarch32 mode Supports usbplug mode
RK3566	rk3566.config rk3566-eink.config	Yes	General version E-ink version
RK3588	rk3588_defconfig rk3588-ramboot.config rk3588-sata.config rk3588-aarch32.config rk3588-ipc.config	Yes	General version No storage device (memory boot) Dual storage support for sata boot Supports aarch32 mode Used on IPC SDK
RK3562	rk3562_defconfig	Yes	General version
RK3576	rk3576_defconfig rk3576-usbplug.config rk3576-car.config rk3576-ab-car.config rk3576-eink.config	Yes	General version Open source usbplug Car version Supports AB system car version E-ink version

Note: If the table and the defconfig released by the SDK are different, please refer to the SDK.

10.1.1.4 Booting Process

The U-Boot booting process for the RK platform is as follows, with 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 needed for
the C environment
    // [First Phase] C environment initialization, initiating a series of
function calls
    => board_init_f: init_sequence_f[]
        initf_malloc
        arch_cpu_init                    // [SoC's lowlevel initialization]
        serial_init                      // Serial port initialization
        dram_init                        // [Obtaining DDR capacity
information]
        reserve_mmu                    // Reserve memory from the end of
the DDR towards lower addresses
        reserve_video
        reserve_uboot
        reserve_malloc
        reserve_global_data
        reserve_fdt
        reserve_stacks
        dram_init_banksize
        system_init
        setup_reloc                    // Determine the relocation
address for U-Boot itself
    // Assembly environment
    => relocate_code                    // Assembly implementation of U-
Boot code relocation
    // [Second Phase] C environment initialization, initiating a series of
function calls
    => board_init_r: init_sequence_r[]
        intr_caches                    // Enable MMU and I/D cache
        intr_malloc
        bidram_intr
        system_intr
        intr_of_live                    // Initialize of_live
        intr_dm                        // Initialize the device model
(dm) framework
        board_init                    // [Platform initialization, the
core part]
        board_debug_uart_init        // Serial port I/O multiplexing,
clock configuration
        init_kernel_dtb                // [Switch to the kernel device
tree blob (dtb)]!
        clks_probe                    // Initialize system clock
frequencies
        regulators_enable_boot_on    // Initialize system power supply
        io_domain_init                // I/O domain initialization
        set_armclk_rate                // __weak, ARM frequency scaling
(only implemented if the platform requires it)
        dvfs_init                    // DVFS for wide-temperature chips
```

```

        rk_board_init                // __weak, Implemented by each
specific platform
        console_init_r
        board_late_init              // [Platform late initialization]
        rockchip_set_ethaddr        // Set MAC address
        rockchip_set_serialno       // Set serial number
        setup_boot_mode             // Parse "reboot xxx" command,
// Identify key presses and loader burn mode, recovery
        charge_display              // U-Boot charging
        rockchip_show_logo          // Display boot logo
        soc_clk_dump                // Print the clock tree
        rk_board_late_init          // __weak, Implemented by each
specific platform
        run_main_loop               // [Enter command line mode, or
execute boot command]

```

10.1.1.5 Shortcut Keys

The RK platform offers a set of serial port combination keys to trigger certain events for debugging and flashing (if they do not trigger, please try multiple times; ineffective when secure-boot is enabled). **Press and hold at startup:**

- Ctrl+C: Enter the U-Boot command line mode;
- Ctrl+D: Enter the loader flashing mode;
- Ctrl+B: Enter the maskrom flashing mode;
- Ctrl+F: Enter the fastboot mode;
- Ctrl+M: Print the info of bidram/system;
- Ctrl+I: Enable the kernel initcall_debug;
- Ctrl+P: Print the cmdline info;
- Ctrl+S: Enter the U-Boot command line after "Starting kernel...";

10.1.2 UEFI Development

Currently, only the RK3588 chip supports UEFI development.

EDK2-RK3588 is an open-source project based on the UEFI Development Kit (EDK II), specifically designed for the Rockchip RK3588 SoC. This project provides a complete development environment to assist developers in firmware development, debugging, and optimization on RK3588.

To familiarize yourself with RK3588 UEFI development and debugging, you can obtain the "Rockchip_Developer_Guide_UEFI_EN.pdf" development guide, which is released with the SDK and located under `docs/en/RK3588/UEFI`.

10.2 Kernel Development

This section aims to provide you with a brief introduction to common kernel configuration modifications, focusing on Device Tree (DTS) configurations, to help you make simple modifications more quickly and conveniently. The following introduction is based on the Linux 5.10 kernel version.

10.2.1 Introduction to DTS

10.2.1.1 DTS Overview

The Device Tree Source (DTS) is a data structure that describes hardware devices, used to define and configure hardware devices within the Linux kernel. The DTS employs a text-like format, representing hardware devices and their relationships through hierarchical structures and property descriptions.

The DTS is compiled into a Device Tree Blob (DTB), which the kernel loads and parses at startup, extracting information about the hardware devices for initialization and configuration.

The introduction of the device tree addresses the inflexibility and maintainability issues of the traditional "hard-coding" approach when dealing with different hardware configurations. With the device tree, hardware descriptions and configuration information are abstracted, allowing the kernel to adapt to different hardware platforms without modifying the kernel code. In this way, the same kernel can run on multiple different hardware devices, simply by loading the corresponding device tree.

This chapter aims to introduce how to add a new board DTS configuration and some common syntax. For more detailed information on DTS, see: [devicetree-specifications](#) and [devicetree-bindings](#).

10.2.1.2 Adding a New Product DTS

- Creating a dts file

The Linux Kernel currently supports multiple platforms using dts files, and the dts files for the Rockchip platform are stored at:

```
ARM 32-bit Kernel: arch/arm/boot/dts/  
ARM 64-bit Kernel: arch/arm64/boot/dts/rockchip
```

The general naming convention 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 generally named according to the silk screen on the board. If your board is an all-in-one board, you only need one dts file to describe it.

If the hardware design consists of a core board and a bottom board, or if the product has multiple product forms, you can place the common hardware descriptions in the dtsti file, and the dts file describes different hardware modules, and include the common hardware descriptions by including "xxx.dtsi".

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

- Modify the Makefile in the directory where the 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 compiling the Kernel, 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 descriptions of dts syntax

The dts syntax can include other common dts data like c/c++ by using `#include xxx.dtsi`. The dts file will inherit all the properties and values of the device nodes included in the dtsi file. If a property is defined in multiple dts/dtsi files, its value will be the definition in the dts. All chip-related controller nodes will be defined in `soc.dtsi`, and if you need to enable the device function, you need to set its status to "okay" in the dts file. To disable 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";
};
```

10.2.2 Module Development Documentation

The directory `<SDK>/docs/en/Common/` contains development documents categorized by functional modules. This section primarily serves as an index for these development documents. When encountering issues during actual development, refer to the following table to read and learn the corresponding development guides. They can be obtained under `docs/en/Common` and will be continuously improved and updated. For more details, see [Documentation Instructions](#).

10.2.3 Common Module Commands

10.2.3.1 CPU-Related Commands

10.2.3.1.1 CPU Frequency Locking Operation

- **Non-Heterogeneous Core Platforms:**

Switch the CPU frequency control policy to `userspace`:

```
echo userspace > /sys/devices/system/cpu/cpufreq/policy0/scaling_governor
```

Set the CPU frequency to 216MHz:

```
echo 216000 > /sys/devices/system/cpu/cpufreq/policy0/scaling_setspeed
```

- **Heterogeneous Core Platforms:**

Switch the little core CPU frequency control policy to `userspace`:

```
echo userspace > /sys/devices/system/cpu/cpufreq/policy0/scaling_governor
```

Set the little core CPU frequency to 216MHz:

```
echo 216000 > /sys/devices/system/cpu/cpufreq/policy0/scaling_setspeed
```

Switch the big core CPU frequency control policy to `userspace`:

```
echo userspace > /sys/devices/system/cpu/cpufreq/policy4/scaling_governor
```

Set the big core CPU frequency to 408MHz:

```
echo 408000 > /sys/devices/system/cpu/cpufreq/policy4/scaling_setspeed
```

10.2.3.1.2 View Current CPU Frequency

- **Non-Heterogeneous Core Platform:**

Using the CPUFreq user-space interface:

```
cat /sys/devices/system/cpu/cpufreq/policy0/scaling_cur_freq
```

Using the Clock Debug interface:

```
cat /sys/kernel/debug/clk/armclk/clk_rate
```

- **Heterogeneous Core Platform:**

Using the CPUFreq user-space interface to view the frequency of little cores and big cores:

```
cat /sys/devices/system/cpu/cpufreq/policy0/scaling_cur_freq # Frequency of
little cores
cat /sys/devices/system/cpu/cpufreq/policy4/scaling_cur_freq # Frequency of
big cores
```

Using the Clock Debug interface to view the frequency of little cores and big cores:

```
cat /sys/kernel/debug/clk/armclk1/clk_rate # Frequency of little cores
cat /sys/kernel/debug/clk/armclkb/clk_rate # Frequency of big cores
```

10.2.3.1.3 Checking Current CPU Voltage

- **Non-Heterogeneous Core Platforms:**

```
cat /sys/kernel/debug/regulator/vdd_core/voltage # Note: vdd_core may need to
be modified according to actual configuration
```

- **Heterogeneous Core Platforms:**

```
cat /sys/kernel/debug/regulator/vdd_core_l/voltage # Little core voltage
cat /sys/kernel/debug/regulator/vdd_core_b/voltage # Big core voltage
```

10.2.3.1.4 CPU Independent Frequency and Voltage Adjustment

- When adjusting frequency and voltage, please follow the sequence below:

Increase Frequency: First adjust the voltage, then increase the frequency.

Decrease Frequency: First decrease the frequency, then adjust the voltage.

- **Setting CPU Frequency via Clock Debug Interface:**

For non-heterogeneous core platforms (e.g., RK3288), set to 216MHz:

```
echo 216000000 > /sys/kernel/debug/clk/armclk/clk_rate
cat /sys/kernel/debug/clk/armclk/clk_rate
```

For heterogeneous core platforms (e.g., RK3399), set the little cores to 216MHz and the big cores to 408MHz:

```
echo 216000000 > /sys/kernel/debug/clk/armclk1/clk_rate
cat /sys/kernel/debug/clk/armclk1/clk_rate
echo 408000000 > /sys/kernel/debug/clk/armclkb/clk_rate
cat /sys/kernel/debug/clk/armclkb/clk_rate
```

- **Setting CPU Voltage via Regulator Debug Interface:**

For non-heterogeneous core platforms (e.g., RK3288), set to 950mV:

```
echo 950000 > /sys/kernel/debug/regulator/vdd_core/voltage
cat /sys/kernel/debug/regulator/vdd_core/voltage
```

For heterogeneous core platforms (e.g., RK3399), set the little cores to 950mV and the big cores to 1000mV:

```
echo 950000 > /sys/kernel/debug/regulator/vdd_core_l/voltage
cat /sys/kernel/debug/regulator/vdd_core_l/voltage
echo 1000000 > /sys/kernel/debug/regulator/vdd_core_b/voltage
cat /sys/kernel/debug/regulator/vdd_core_b/voltage
```

10.2.3.1.5 How to View Frequency and Voltage Table

```
cat /sys/kernel/debug/opp/opp_summary
```

10.2.3.1.6 How to Check CPU Temperature

```
cat /sys/class/thermal/thermal_zone0/temp
```

For more details, please refer to the document:

docs\en\Common\DVFS\Rockchip_Developer_Guide_CPUFreq_EN

10.2.3.2 GPU-Related Commands

10.2.3.2.1 GPU Clock Setting Operation

- Switch the GPU to user space control mode (the path `ff400000.gpu` may vary depending on the platform):

```
echo userspace > /sys/class/devfreq/ff400000.gpu/governor
```

- Set the GPU frequency to 400MHz:

```
echo 400000000 > /sys/class/devfreq/ff400000.gpu/userspace/set_freq
```

- Check the current GPU frequency:

```
cat /sys/class/devfreq/ff400000.gpu/cur_freq
```

10.2.3.2.2 View Current GPU Frequency

- Method One: Through the devfreq user-space interface (the path `ff400000.gpu` may vary depending on the platform):

```
cat /sys/class/devfreq/ff400000.gpu/cur_freq
```

- Method Two: Through the clock debug interface (`aclk_gpu` may vary according to actual configuration):

```
cat /sys/kernel/debug/clk/aclk_gpu/clk_rate
```

10.2.3.2.3 Inspect Current GPU Voltage

- Inspect GPU voltage (`vdd_logic` may vary based on actual regulator configuration):

```
cat /sys/kernel/debug/regulator/vdd_logic/voltage
```

10.2.3.2.4 GPU Frequency and Voltage Adjustment

- **Important:** When increasing frequency, adjust the voltage first; when decreasing frequency, adjust the frequency first.
- Disable GPU automatic frequency scaling (the path `ff400000.gpu` may vary depending on the platform):

```
echo userspace > /sys/class/devfreq/ff400000.gpu/governor
```

- Adjust GPU frequency (`aclk_gpu` may vary depending on the actual configuration):

```
echo 400000000 > /sys/kernel/debug/clk/aclk_gpu/clk_rate  
cat /sys/kernel/debug/clk/aclk_gpu/clk_rate
```

- Adjust GPU voltage (`vdd_logic` may vary depending on the actual regulator configuration):

```
echo 1000000 > /sys/kernel/debug/regulator/vdd_logic/voltage  
cat /sys/kernel/debug/regulator/vdd_logic/voltage
```

10.2.3.2.5 Inspect GPU Utilization

- Inspect GPU Utilization (the path `fb000000.gpu` may vary depending on the platform):

```
cat /sys/class/devfreq/*.gpu/load
```


10.2.3.2.6 Inspect GPU Temperature

- Inspect GPU temperature (the `thermal_zone1` may need to be modified according to the actual hardware configuration):

```
cat /sys/class/thermal/thermal_zone1/temp
```

10.2.3.3 DDR Related Commands

10.2.3.3.1 Inspect DDR Frequency

- Use the following commands to view DDR frequency information:

```
cat /sys/kernel/debug/clk/clk_summary | grep ddr
cat /sys/class/devfreq/dmc/cur_freq
```

10.2.3.3.2 DDR Frequency Setting Operation

- Switch the DDR frequency control strategy to `userspace`:

```
echo userspace > /sys/class/devfreq/dmc/governor
```

- List available DDR frequencies:

```
cat /sys/class/devfreq/dmc/available_frequencies
```

- Set the DDR frequency to 1560 MHz:

```
echo 1560000000 > /sys/class/devfreq/dmc/userspace/set_freq
```

10.2.3.3.3 View DDR Bandwidth Utilization

- To check the current utilization of DDR bandwidth:

```
cat /sys/class/devfreq/dmc/load
```

10.2.3.3.4 DDR Bandwidth Statistics

- Utilize the DDR bandwidth statistics tool `rk-msch-probe` provided by FAE:

```
./rk-msch-probe-for-user-64bit -c rk3576
```

10.2.3.3.5 Memory Debugging

- Viewing Memory Information:

```
cat /proc/meminfo
```

- Viewing Virtual Memory Usage:

```
cat /proc/vmallocinfo
```

- Enabling CMA Debugging Features, which requires setting in the configuration:

```
CONFIG_CMA_DEBUGFS=y    # Enable debug file nodes
CONFIG_CMA_DEBUG=y      # Print CMA log information
```

- Viewing CMA Related Information:

```
ls /sys/kernel/debug/cma/cma-reserved
```

- Viewing DMA Buffer Information:

```
cat /sys/kernel/debug/dma_buf/bufinfo
cat /proc/rk_dmabuf/size
```

10.2.3.3.6 Memory Stress Test

Download DDR related materials from Rockchip Redmine:

[DDR Related Materials](#) Execute memory stress test commands:

```
stressapptest -s 43200 -i 4 -C 4 -W --stop_on_errors -M 128
memtester 128m > /data/memtester_log.txt &
```

For detailed reference:

```
docs\en\Common\DDR\***
docs\en\Common\MEMORY\***
```

10.2.3.4 NPU-related Commands

10.2.3.4.1 Viewing NPU Frequency

Use the following commands to view the NPU frequency (note: the device path `fdab0000.npu` may vary depending on the platform):

```
cat /sys/kernel/debug/clk/clk_summary | grep npu
cat /sys/class/devfreq/*.npu/cur_freq
```

10.2.3.4.2 NPU Frequency Setting Operation

Perform frequency configuration for NPU devices (the device path `*.npu` may need to be modified according to the platform):

```
echo userspace > /sys/class/devfreq/*.npu/governor
echo 1000000000 > /sys/class/devfreq/*.npu/userspace/set_freq
cat /sys/class/devfreq/*.npu/cur_freq
```

10.2.3.4.3 NPU Support Query Settings

The following commands are applicable for the RKNPU2 platform with a driver version above 0.7.2:

- Query NPU Utilization:

```
cat /sys/kernel/debug/rknpu/load
cat /proc/debug/rknpu/load
```

- Query NPU Driver Version:

```
cat /sys/kernel/debug/rknpu/driver_version
cat /proc/debug/rknpu/driver_version
```

- Manage NPU Power State:

```
echo on > /sys/kernel/debug/rknpu/power # Turn on NPU power
echo off > /sys/kernel/debug/rknpu/power # Turn off NPU power
```

- Query NPU Operating Voltage:

```
cat /sys/kernel/debug/rknpu/volt
```

- Dynamically Manage NPU Power and Delay Shutdown Time (unit: ms):

```
cat /sys/kernel/debug/rknpu/delayms # Query power delay shutdown time
echo 2000 > /sys/kernel/debug/rknpu/delayms # Set power delay shutdown time to
2000ms
```

10.2.3.4.4 Related Materials

- **RKNPU Development Materials**

Tools: [rknn-toolkit](#)

Runtime: [rknpu](#), [RK3399Pro_npu](#)

- **RKNPU2 Development Materials**

Tools: [rknn-toolkit2](#)

Runtime: [rknpu2](#)

rknn_model_zoo: [rknn_model_zoo](#)

10.2.3.5 RGA-related Commands

10.2.3.5.1 Query RGA Frequency

```
cat /sys/kernel/debug/clk/clk_summary | grep rga # Query RGA frequency,
including aclk frequency
```

10.2.3.5.2 Modifying RGA Frequency

```
echo 400000000 > /sys/kernel/debug/clk/aclk_rga/clk_rate # Modify the frequency
value to the desired frequency
```

10.2.3.5.3 RGA Driver Version Query

```
cat /sys/kernel/debug/rkrga/driver_version
cat /proc/rkrga/driver_version
```

10.2.3.5.4 Query the version number of the librqa library

- **Linux System**

```
strings /usr/lib/librqa.so | grep rqa_api | grep version
```

10.2.3.5.5 Enable librqa Logging

- **Linux System (librqa version 1.9.0 and above)**

```
export ROCKCHIP_RGA_LOG=1
export ROCKCHIP_RGA_LOG_LEVEL=6
```

10.2.3.5.6 RGA Debug Node

```
cat /sys/kernel/debug/ampak/debug
```

10.2.3.5.7 RGA Load Query

```
cat /sys/kernel/debug/AMPAK/load
```

10.2.3.5.8 RGA Memory Manager Inquiry

```
cat /sys/kernel/debug/rkrga/mm_session
```

10.2.3.5.9 RGA Task Request Inquiry

```
cat /sys/kernel/debug/AMPAK/request_manager
```

10.2.3.5.10 RGA Hardware Information Inquiry

```
cat /sys/kernel/debug/rkrga/hardware
```

10.2.3.5.11 Dump Runtime Data

```
/*Use the following command to dump runtime data for debugging purposes*/
echo /data/rga_image > /sys/kernel/debug/rkrga/dump_path
echo 1 > /sys/kernel/debug/rkrga/dump_image
```

For specific details, refer to the documentation:

`docs\en\Common\RGA***` or <https://github.com/airockchip/librqa/tree/main/docs>

10.2.3.6 VPU-related Commands

10.2.3.6.1 Querying VPU Driver Version

Use the following command to query the version information of the VPU driver:

```
cat /proc/mpp_service/version
```

10.2.3.6.2 Checking VPU Frame Rate

To view the frame rate data of the VPU, use the following command:

```
cat /proc/mpp_service/sessions-summary
```

10.2.3.6.3 Viewing Frequency

To view frequency information related to rkvinc, execute the following command:

```
cat /sys/kernel/debug/clk/clk_summary | grep rkvinc
```

10.2.3.6.4 Modify Frequency

To modify the frequency of the VPU to 600MHz, you can use the following command:

```
echo 600000000 > /proc/mpp_service/rkvinc/clk_core
```

10.2.3.6.5 Enabling VPU Debug Print

To enable the debug print feature of VPU:

```
echo 0x100 > /sys/module/rk_vcodec/parameters/mpp_dev_debug
```

For more information on VPU documentation, please refer to:

```
docs\en\Common\MPP\Rockchip_Developer_Guide_MPP_EN.pdf
```

10.2.4 Pin Configuration Control

10.2.4.1 General-Purpose Input/Output (GPIO)

For instance, the RK3399/RK3399Pro chip offers 5 sets of GPIOs (GPIO0 to GPIO4), totaling 122 GPIO pins. All GPIOs can serve as interrupt sources, with GPIO0 and GPIO1 also capable of acting as system wake-up pins. Each GPIO can be configured via software for pull-up or pull-down, with the default state being input mode. Additionally, the driving capability of the GPIO can also be configured by software.

Regarding the correspondence between GPIO on the schematic and GPIO in the Device Tree Source (DTS), for example, GPIO4C0 should be represented as "gpio4 16" in the DTS. This is because GPIO4A has 8 pins, and GPIO4B also has 8 pins, with counting starting from 0, so the C0 pin corresponds to the 16th pin, the C1 pin corresponds to the 17th pin, and so on.

10.2.4.1.1 I/O Multiplexing (IOMUX)

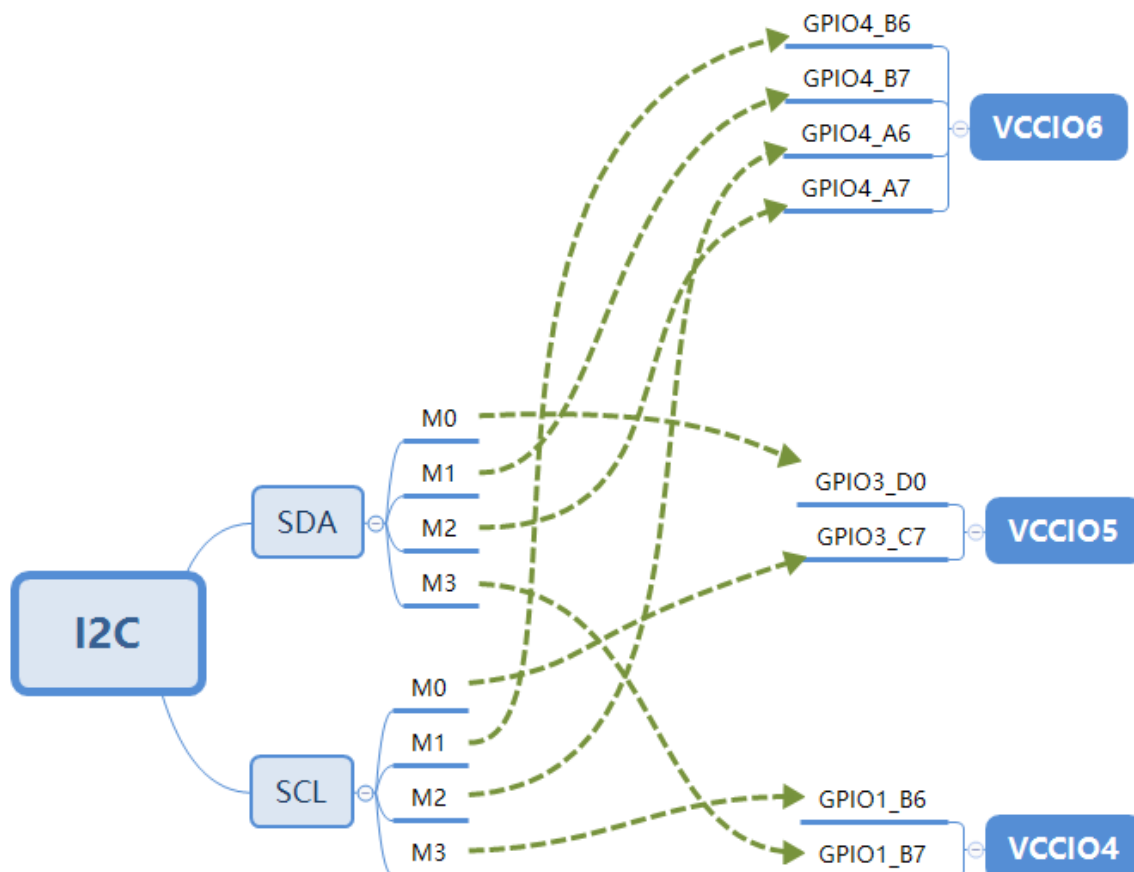
The SOC pins of Rockchip can be multiplexed into various functions. If a controller has multiple multiplexing pins, they are generally referred to as m0, m1, m2, etc. For instance, the I2C controller has two sets of multiplexing pins, which are named 2cm0 and i2cm1 respectively.

The configuration registers for pin multiplexing are located in GRF/PMUGRF (on RK3588, it is referred to as IOC).

As an example, the RK3588 BUS_IOC_GPIO1B_IOMUX_SEL_H Address: Operational Base + offset (0x002C)

```
gpio1b7_sel
4'h0: GPIO
4'h2: MIPI_CAMERA2_CLK_M0
4'h3: SPDIF1_TX_M0
4'h4: PCIE30X2_PERSTN_M3
4'h5: HDMI_RX_CEC_M2
4'h6: SATA2_ACT_LED_M1
4'h9: I2C5_SDA_M3
4'ha: UART1_RX_M1
4'hb: PWM13_M2
```

The following is the IOMUX for RK3588 I2C5:



Multiplexing support provides greater flexibility in hardware design. When the peripheral working voltage is 1.8V or 3.3V, one can select pins with different voltage domains VCCIO.

Note: The register configuration for multi-path multiplexing is ineffective for TX-type pins and effective for RX-type pins.

10.2.4.2 PULL (Port Pull-Up and Pull-Down)

The bias of Rockchip IO PAD generally supports three modes:

- bias-disable
- bias-pull-up
- bias-pull-down

Pull-up and pull-down configurations are applied to the IO PAD and take effect for both GPIO and IOMUX.

10.2.4.3 DRIVE-STRENGTH (Port Drive Strength)

The drive strength of Rockchip IO PAD supports different intensity configurations based on various processes. For chips before RK3399, the drive strength configuration is set in units of mA, while for chips after RK1808, it is generally set in levels, with the numerical value of the gear representing the register configuration value.

For example, in the RK3588 TRM, the drive strength levels for GPIO0_C7 are as follows:

```
gpio0c7_ds
GPIO0C7 DS control Driver Strength Selection
3'b000: 100ohm
3'b100: 66ohm
3'b010: 50ohm
3'b110: 40ohm
3'b001: 33ohm
3'b101: 25ohm
```

The software driver still processes according to the level, that is, the above register description corresponds to:

```
3'b000: Level0
3'b100: Level4
3'b010: Level2
3'b110: Level6
3'b001: Level1
3'b101: Level5
```

In the DTS, `drive-strength=<5>;` indicates the configuration is set to Level5, which means writing `3'b101` to the register.

10.2.4.4 SMT (Synchronous Metastability Trigger)

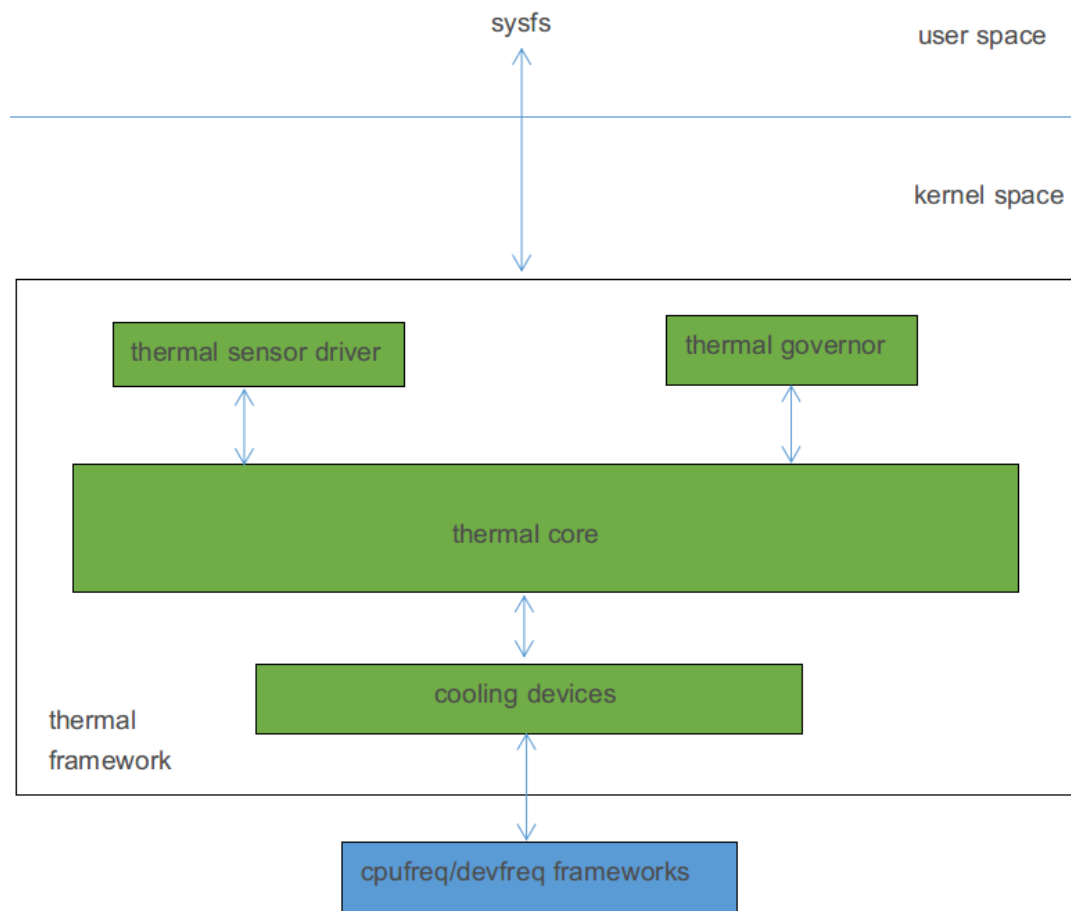
Most Rockchip IO PAD chips support the SMT function, which is disabled by default; enabling SMT can eliminate edge jitter, increase the voltage range of VIH and VIL, and enhance the signal stability of IOs. Generally, the I2C SCL/SDA will have the SMT function enabled by default.

For more details on Pinctrl usage, please refer to the document

`<SDK>/docs/en/Common/PINCTRL/Rockchip_Developer_Guide_Linux_Pinctrl_EN.pdf`.

10.2.5 Thermal Control Development

The Thermal framework is a model defined by kernel developers to support the control of system temperature based on a specified governor, in order to prevent the chip from overheating. The Thermal framework consists of governors, cores, cooling devices, and sensor drivers, with the software architecture as follows:



Thermal Governor: It is used to determine whether the cooling device needs to reduce frequency and to what extent. The following types of governors are included in Linux versions 4.4 and above:

- **power_allocator:** Introduces PID (Proportional-Integral-Derivative) control, dynamically allocates power to each cooling device based on the current temperature, and converts power into frequency, thereby achieving the effect of limiting frequency according to temperature.
- **step_wise:** The cooling device reduces frequency step by step according to the current temperature.
- **fair_share:** Cooling devices with more frequency bands are prioritized for frequency reduction.
- **userspace:** Does not limit frequency.

Thermal Core: It encapsulates and abstracts thermal governors and thermal drivers, and defines a clear interface.

Thermal Sensor Driver: A sensor driver used to obtain temperature, such as `tsadc`.

Thermal Cooling Device: A device that generates heat or can reduce temperature, such as CPU, GPU, DDR, etc.

The Rockchip chip has thermal control sensors on its ARM core and GPU core, which can monitor the temperature of the CPU and GPU in real time, and control the frequency of the CPU and GPU through algorithms to control the temperature of the CPU and GPU. The hardware design and mold of each product are different, and the corresponding heat dissipation conditions are also different. Appropriate adjustments to thermal control parameters can be made through the following configurations in dts to adapt to the product:

Set the temperature at which thermal control is activated:


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

Set the maximum temperature for thermal control:

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

Set the software shutdown temperature:

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

Configure the 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 information on thermal control, refer to the relevant documentation in the

`<SDK>/docs/en/Common/THERMAL` directory.

10.2.6 DDR Development Guide

During the DDR development process, you may encounter various issues. Below are some key development resources and steps to help you efficiently resolve problems:

- **Frequently Asked Questions:** Understand the typical issues that may arise in DDR development and their solutions.
- **Troubleshooting Methods:** Master the systematic problem diagnosis process to quickly locate the root cause of the problem.
- **Particle Verification Process:** Follow detailed verification steps to ensure the performance and compatibility of DDR particles.
- **PCB Design Considerations:** Pay attention to the key elements of PCB design to avoid potential layout issues.
- **Eye Diagram Analysis Tools:** Learn how to use eye diagram tools for signal integrity analysis to optimize DDR performance.
- **Bandwidth Statistics Tools:** Utilize bandwidth statistics tools to assess the data transmission capabilities of DDR.

For more detailed information and guidance, please refer to the documents under the path

`<SDK>/docs/en/Common/DDR`. These documents will provide you with comprehensive development instructions and practical guides.

10.2.7 SD Card Configuration

For specific instructions on SD card configuration, please refer to the relevant documents in the directory

`<SDK>/docs/en/Common/MMC`.

For some chips, such as RK3326/RK3399PRO, the UART for debugging and the SD card are multiplexed by default. The default configuration enables debugging. If you want to use the SD card, you need the following configuration:

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

10.3 Recovery Development

10.3.1 Introduction

The development of the Recovery mechanism, akin to the Recovery feature in Android, primarily serves to erase user data and facilitate system upgrades. Currently, it mainly supports upgrades in Recovery mode, upgrades in A/B partition mode, and factory reset.

In Linux, Recovery mode involves an additional Recovery partition on the device, composed of the kernel, resources, and ramdisk, mainly used for upgrade operations. The u-boot will determine whether to boot into the Normal system or the Recovery system based on the fields stored in the misc partition. Due to the system's independence, Recovery mode ensures the integrity of the upgrade process, meaning that if the upgrade is interrupted, such as by an abnormal power loss, the upgrade can still continue to be executed.

10.3.2 Debugging

Common debugging methods involve enabling debug mode.

Create a hidden file named `.rkdebug` in the directory

`buildroot/output/rockchip_xxx_recovery/target`:

```
touch .rkdebug
```

Logs for upgrades in Recovery mode are printed out through the serial port. Another method is to inspect the file `userdata/recovery/Log`.

For more information on Recovery development, refer to the document:

`<SDK>/docs/en/Linux/Recovery/Rockchip_Developer_Guide_Linux_Recovery_EN.pdf`

10.3.3 Recovery Feature Summary

In summary, all Recovery features are summarized in the following table:

Feature	Description	Involved Components
Support for passing commands to recovery upgrade in normal mode	Enter recovery mode through shell commands in normal mode to complete partition upgrades	update/updateEngine
Direct system upgrade in normal mode, Linux A/B partition upgrade	Directly complete upgrades in normal mode, requiring the enablement of A/B partitions, Linux A/B, which means preparing two independent system firmwares stored separately on flash, the system can boot from one slot, if the current slot fails to boot, it can boot from the other slot, directly upgrade the system in Normal mode without entering the system upgrade mode, just restart the system to enter the upgraded system	updateEngine
bootcontrol	The bootcontrol program can be set in two boot modes, successful_boot and reset retry, used to set which partition is bootable	updateEngine
Upgrade recovery partition in normal mode	To prevent issues with other partitions not being able to upgrade normally due to power loss during the upgrade of recovery.img, this partition upgrade is placed under the normal system upgrade, that is, during the upgrade, it will first detect whether the update.img upgrade package includes a packaged recovery.img, if so, then upgrade the recovery partition, and then enter Recovery mode to upgrade other partition firmwares	update/updateEngine
Print log information	The upgrade log can be output through the serial port or stored under userdate, if it is a device with a screen, you can view the log print through the UI interface	Recovery
Command to obtain version information	Obtain the current upgrade program version information	update/updateEngine
Obtain data from the misc partition	Obtain data from the misc partition, successful upgrade can verify successful data acquisition	Recovery
Parse commands within the misc partition	Parse data within the misc partition, successful upgrade can verify successful data parsing	Recovery
Reboot Recovery command parsing	The upgrade program supports rebooting	Recovery
Provide command usage instructions	Obtain detailed instructions for the current command usage	update/updateEngine
Full package upgrade package production	Produce update.img upgrade packages	update/updateEngine
Differential package upgrade package production	Differential upgrade package production	update/updateEngine
A/B package	Produce A/B partition upgrade packages	update/updateEngine
Upgrade via USB	Store the upgrade package in a USB flash drive, and perform partition upgrades through the upgrade program	Recovery

Feature	Description	Involved Components
Upgrade via SD card	Create a bootable disk upgrade using SDDiskTool, can normally read firmware from SD, write to Flash, and start normally (the capacity of the SD card must not exceed 32G, otherwise formatting will fail), if it is in A/B mode, you need to export RK_UPDATE_SDCARD_ENABLE_FOR_AB=true, and then compile	Recovery
Unpack (support for update.img, A/B, differential)	During the upgrade process, the img upgrade file will be unpacked and analyzed	update/updateEngine
Verify upgrade data	After a successful upgrade, the partition will be verified	rkupdate/updateEngine
Block device	Upgrade of storage devices such as EMMC and other block devices	rkupdate/updateEngine
MTD device	MTD is a device for managing flash storage, such as NAND flash and NOR flash	rkupdate/updateEngine
Partition verification after upgrade completion	After the upgrade is completed, the partition needs to be verified	rkupdate/updateEngine
Factory reset	Read and write configuration files are stored in the userdata partition, the factory firmware will default some configuration parameters, users will generate or modify configuration files after using for a period of time, sometimes users need to clear this data, we need to restore to factory settings	Recovery
Implement a simple UI display	UI display	Recovery
Compatible with projects without UI	Compatible with projects without screens	Recovery
UI prints upgrade log	UI display	Recovery
Dual screen display	Can display on dual screens	Recovery
Screen rotation	Implement UI screen rotation	Recovery
Support for serial port printing to Console	Allow log output to the console	Recovery
Log storage in userdata/recovery/log (enable this feature through commands)	Store logs in the userdata partition, which can be viewed in normal mode after a successful upgrade	Recovery
Abnormal power off	Ensure that the upgrade can continue after restarting in case of unexpected power off during the upgrade process	Recovery
SD card abnormally removed	Directly remove the SD card during the upgrade process started via SD card	Recovery
USB flash drive abnormally removed	Abnormally remove the USB flash drive during the upgrade process via USB flash drive	Recovery

For more Recovery development materials, refer to the document

`<SDK>/docs/en/Linux/Recovery/Rockchip_Developer_Guide_Linux_Recovery_EN.pdf`

10.4 Buildroot Development

Buildroot is a tool that facilitates the construction of a complete Linux system for embedded systems through cross-compilation, offering a simple and automated process.

On the native Buildroot, Rockchip has integrated the BSP configuration for related chips, configurations for hardware module acceleration features, and in-depth customization development of third-party packages, making it convenient for customers to deeply customize and develop their products.

For specific Buildroot development documentation, refer to:

`<SDK>/docs/en/Linux/System/Rockchip_Developer_Guide_Buildroot_EN.pdf`

10.5 Debian Development

Rockchip offers support for Debian 10/11 based on the X11 display architecture and is based on the Linaro version. This support also includes graphics and video acceleration capabilities. The relevant packages include libmali, xserver, and gstreamer rockchip, among others. These packages can be compiled 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 compile deb packages, please refer to the document:

`<SDK>/docs/en/Linux/Docker/Rockchip_Developer_Guide_Debian_Docker_EN.pdf`

Debian Development Documentation Reference

`<SDK>/docs/en/Linux/System/Rockchip_Developer_Guide_Debian_EN.pdf`

10.6 Yocto Development

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

10.7 Audio Development

10.7.1 Kernel Audio Driver Development

The kernel driver utilizes `soc/rockchip/rockchip_multicodecs.c` to replace `soc/generic/simple-card.c`, as the simple card in the kernel driver is merely for simplicity and cannot meet the complex requirements of some products.

The rockchip_multicodecs driver differs from the simple-card mainly in the following aspects:

- Primarily, it is compatible with the support for one sound card with one or multiple codecs.
- It supports ADC for headphones and other detections, whereas the simple-card commonly uses GPIO.
- The I2S TDM controller has separate TX/RX, and multicodecs support them independently, which would affect third-party implementations if added to the simple-card.

10.7.2 Audio Pulseaudio Path Adaptation

The default audio system utilizes Pulseaudio, and typically, it is only necessary to configure

```
/etc/pulse/default.pa.
```

For example, in RK3588, adaptation for ES8388 and RK809 codecs is performed as follows:

```
+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
```

To add support for additional codecs, obtain relevant information through the following commands:

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

For detailed reference, consult the Debian official [PulseAudio](#) and

```
<SDK>/docs/Common/AUDIO/Rockchip_Developer_Guide_PulseAudio_EN.pdf.
```

10.8 Multimedia Development

Development of multimedia on the Rockchip platform through GStreamer/Rockit.

The general pathway is as follows:

```
vpu_service  -->  mpp -->  GStreamer/Rockit --> app  
  
vpu_service: Driver  
mpp: The video codec middleware for the Rockchip platform, refer to the mpp  
documentation for details  
GStreamer/Rockit: Component interfacing with the app
```

The complete solution currently provided by the Rockchip Linux general SDK is based on GStreamer. The advantage of using GStreamer is the convenience of building players, encoders, and other applications based on the pipeline approach. For customized development based on Rockit, refer to the relevant Rockit release documentation.

For specific materials, refer to:

```
<SDK>/docs/en/Linux/Multimedia  
├── Rockchip_Developer_Guide_Linux_RKADK_EN.pdf  
├── Rockchip_User_Guide_Linux_Gstreamer_EN.pdf  
└── Rockchip_User_Guide_Linux_Rockit_EN.pdf  
  
<SDK>/docs/en/Common/MPP/Rockchip_Developer_Guide_MPP_PDF
```

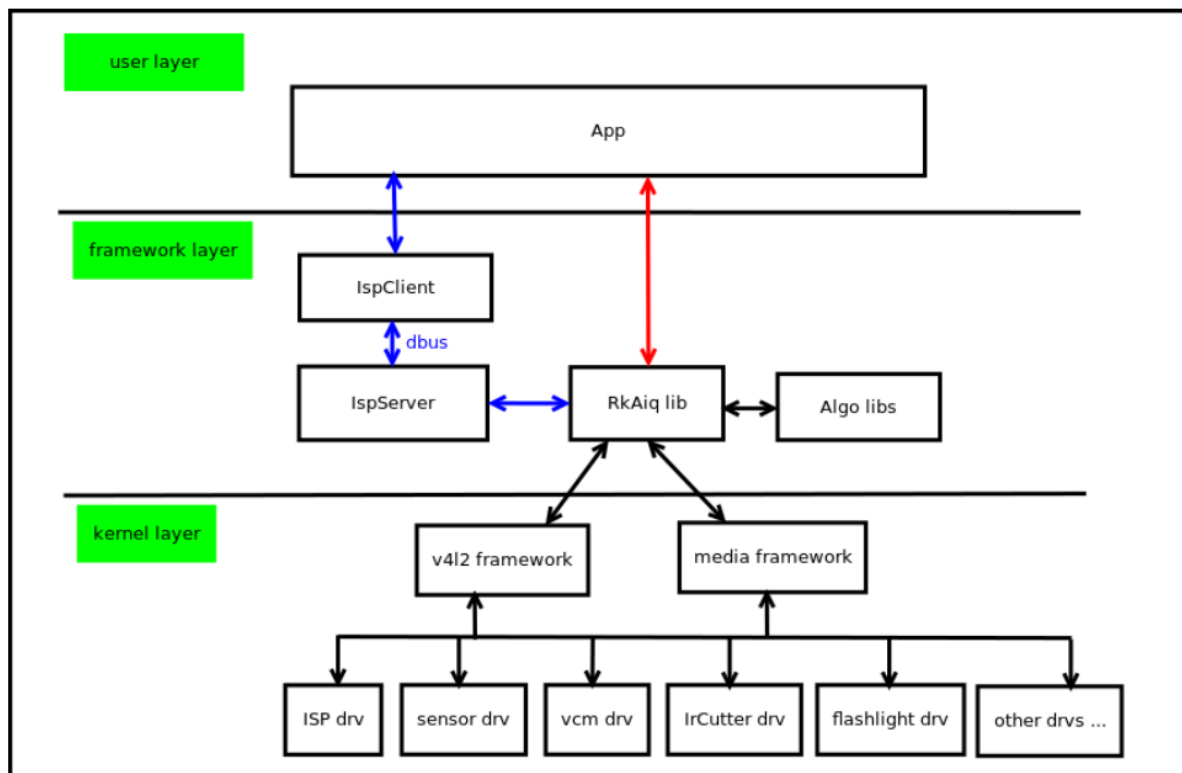
10.9 Camera Development

10.9.1 Image Processing Module (ISP)

The ISP encompasses a series of image processing algorithm modules, primarily including dark current correction, hot pixel correction, 3A, HDR, lens shading correction, lens distortion correction, 3DLUT, denoising (including RAW domain denoising, multi-frame noise reduction, color denoising, etc.), and sharpening. It includes both hardware algorithm implementation and software logic control parts, with RKAIQ being the implementation of the software logic control part.

The RkAIQ software module mainly implements the functionality of obtaining image statistics from the ISP driver, combining IQ Tuning parameters, and using a series of algorithms to calculate new ISP, Sensor, and other hardware parameters. This process is iterated continuously until the optimal image effect is achieved.

On the Rockchip platform, the RKISP or RKAIQ (Rockchip Auto Image Quality) module is mainly used for ISP function development. The ISP software framework is referenced as follows:



ISP has different versions, where:

ISP1.X is primarily applicable to RK3399/RK3288/PX30/RK3326/RK1808, etc.

ISP21 is primarily applicable to RK3566_RK3568, etc.

ISP30 is primarily applicable to RK3588, etc.

ISP32-lite is primarily applicable to RK3562, etc.

ISP39 is primarily applicable to RK3576, etc.

ISP development documentation includes ISP development documentation, VI driver development documentation, IQ Tool development documentation, debugging documentation, and color debugging documentation. The specific documents are as follows:


```
docs/en/Common/ISP/
├─ ISP1.X
├─ ISP21
├─ ISP30
├─ ISP32-lite
├─ ISP39
└─ The-Latest-Camera-Documents-Link.txt
```

10.9.2 USB CAMERA

- Supports standard UVC Camera functionality, up to 4k preview (Rockchip RK356X)
- Supports stable transmission with USB composite devices
- Compatible with smart TVs or PCs for various terminal device previews

Enable this configuration in Buildroot

```
+BR2_PACKAGE_ROCKIT_UVC=y
+BR2_PACKAGE_ROCKCHIP_UVC_APP=y
+BR2_PACKAGE_ROCKIT=y
```

You can refer to the following methods for testing:

```
Board-side configuration enabled:
root@rk3588-buildroot:/# /etc/init.d/S50usbdevice.sh stop
root@rk3588-buildroot:/# usb_config.sh
root@rk3588-buildroot:/# rockit_uvc
```

On the PC side, use applications like AMCap or PotPlayer for USB camera, and open it to see the preview. Switch formats or resolutions according to the application used on the upper machine.

Rockchip RK356X supports UVC Camera functionality and requires the following patch to be applied to the kernel:

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3568-evb.dtsi
b/arch/arm64/boot/dts/rockchip/rk3568-evb.dtsi
index 18d6115341cd..268f5d2b1c05 100644
--- a/arch/arm64/boot/dts/rockchip/rk3568-evb.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3568-evb.dtsi
@@ -1778,6 +1778,8 @@
    &usbdrd_dwc3 {
        dr_mode = "otg";
+       snps,tx-fifo-resize;
+       snps,dis-ulu2-quirk;
        extcon = <&usb2phy0>;
        status = "okay";
    };
```

If the memory exceeds 4G, it needs to be limited to 4G because the USB controller can only access memory within 4G.

```

--- a/arch/arm64/boot/dts/rockchip/rk3568-linux.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3568-linux.dtsi
@@ -13,7 +13,7 @@ aliases {
    };

    chosen: chosen {
-       bootargs = "earlycon=uart8250,mmio32,0xfe660000 console=ttyFIQ0
root=PARTUUID=614e0000-0000 rw rootwait";
+       bootargs = "earlycon=uart8250,mmio32,0xfe660000 console=ttyFIQ0
root=PARTUUID=614e0000-0000 mem=4g rw rootwait";
    };

```

10.10 Graphics Development

Rockchip Linux graphics system development involves utilizing DRM (Direct Rendering Manager) and DMA-BUF (Direct Memory Access - Buffers) technologies for ARM Linux graphics development on the Rockchip Linux platform. The advantage of this platform lies in its universal architecture, which makes custom development based on this architecture relatively easy and allows for the full utilization of existing components. Currently, many basic open-source projects have begun to adopt the Rockchip platform as an adaptation platform for the ARM end.

However, there are also some challenges in this field. Many developers may not be familiar enough with the concepts and applications of DRM and DMA-BUF, which requires them to invest a certain amount of time and energy to learn and master. To gain an in-depth understanding of Rockchip graphics system development, you can refer to the [Rockchip wiki](#) and the following related technical documents.

```

<SDK>/docs/en/Linux/Graphics/
├─ Rockchip_Developer_Guide_Buildroot_Weston_EN.pdf
├─ Rockchip_Developer_Guide_Linux_Graphics_EN.pdf
└─ Rockchip_Developer_Guide_Linux_LVGL_EN.pdf

<SDK>/docs/en/Common/DISPLAY/
├─ DP
├─ HDMI
├─ MIPI
├─ RK628
├─ Rockchip_BT656_TX_AND_BT1120_TX_Developer_Guide_EN.pdf
├─ Rockchip_DRM_Panel_Porting_Guide_V1.6_20190228.pdf
├─ Rockchip_Develop_Guide_DRM_Direct_Show_EN.pdf
├─ Rockchip_Developer_Guide_Baseparameter_Format_Define_And_Use_EN.pdf
├─ Rockchip_Developer_Guide_DRM_Display_Driver_EN.pdf
├─ Rockchip_Developer_Guide_DisplayPort_EN.pdf
├─ Rockchip_Developer_Guide_HDMI_EN.pdf
├─ Rockchip_Developer_Guide_RGB_MCU_EN.pdf
├─ Rockchip_Developer_Guide_RK628_For_All_Porting_EN.pdf
├─ Rockchip_RK3588_Developer_Guide_MIPI_DSI2_EN.pdf
└─ Rockchip_RK3588_Developer_Guide_Vsync_Adjust_EN.pdf

```

10.10.1 Boot Screen Development

10.10.1.1 Boot Logo Development

- Enable U-Boot Logo

The logo is controlled through the corresponding display interface's `route_XXX` node in the Linux kernel dts (the U-Boot display module and the Linux kernel share the same dtb), where `XXX` can be `dsi`, `edp`, `hdmi`, `lvds`, etc. For specifics, please search for the keyword "route" in the dts.

Taking MIPI DSI0 as an example, find the `route_dsi0` node in the corresponding board-level dts file and set the status to "okay":

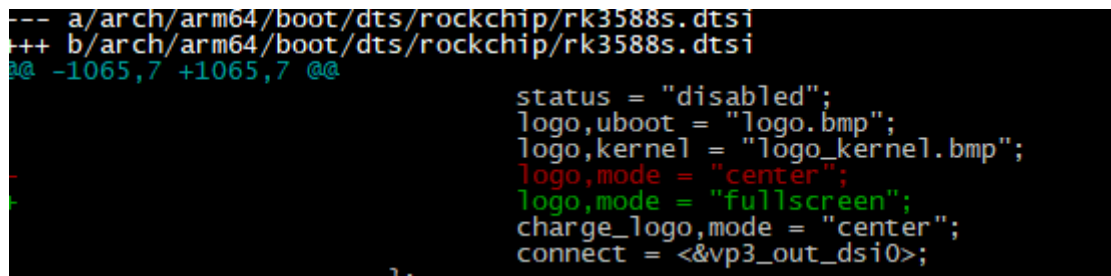
```
&route_dsi0 {
    status = "okay";
    connect = <&vp3_out_dsi0>;
};
```

The `connect` property is detailed in the subsequent section **DTS Configuration**.

- Configure U-Boot Logo for Full-Screen Display

The logo is displayed centered by default. If a full-screen display is required, modify the route node for the corresponding interface.

For example, to display the logo on DSI0 in full-screen mode, modify the `logo,mode` attribute of `route_dsi0` to "fullscreen".



```
--- a/arch/arm64/boot/dts/rockchip/rk3588s.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3588s.dtsi
@@ -1065,7 +1065,7 @@
     status = "disabled";
     logo,uboot = "logo.bmp";
     logo,kernel = "logo_kernel.bmp";
     logo,mode = "center";
     logo,mode = "fullscreen";
     charge_logo,mode = "center";
     connect = <&vp3_out_dsi0>;
```

- Logo Image Requirements

The U-Boot logo and the Linux Kernel logo have the same resolution, and the resolution must be an even number;

It only supports bmp images with 8bit, 16bit, 24bit, and 32bit;

Both the U-Boot logo and the Linux Kernel logo must be enabled simultaneously, meaning `logo.bmp` and `logo_kernel.bmp` must be provided together, not just one.

- Boot Log Confirmation

If the logo function is properly enabled, you will see logs similar to the following during the U-Boot stage:

```
Rockchip UBOOT DRM driver version: v1.0.1
vp0 have layer nr:2[0 2 ], primary plane: 2
vp1 have layer nr:2[1 3 ], primary plane: 3
vp2 have layer nr:2[6 8 ], primary plane: 8
vp3 have layer nr:2[7 9 ], primary plane: 9
Using display timing dts
dsi@fde20000: detailed mode clock 132000 kHz, flags[a]
    H: 1080 1095 1099 1129
    V: 1920 1935 1937 1952
bus_format: 100e
```

```

VOP update mode to: 1080x1920p0, type: MIPI0 for VP3
VOP VP3 enable Esmart3[654x270->654x270@213x825] fmt[2] addr[0xedf04000]
final DSI-Link bandwidth: 880000 Kbps x 4
.....
hdmi_select_link_config use tmds mode
mode:1920x1080 bus_format:0x100a
hdmi@fde80000: detailed mode clock 148500 kHz, flags[5]
    H: 1920 2008 2052 2200
    V: 1080 1084 1089 1125
bus_format: 100a
VOP update mode to: 1920x1080p0, type: HDMI0 for VP0
.....
VOP VP0 enable Esmart0[654x270->654x270@633x405] fmt[2] addr[0xedf04000]
.....
CLK: (uboot. arm: enter 1200000 KHz, init 1200000 KHz, kernel ON/A)

```

It can be observed that the logo display has been initiated on both VP0 and VP3.

The display resolution for VP0 is 1920 x 1080, with the display interface being HDMI.

The display resolution for VP3 is 1080 x 1920, with the display interface being MIPI DSI.

The displayed logo size is 654 x 270.

10.10.1.2 Boot Animation Development

Utilize the SDK's mak config to configure RK_ROOTFS_BOOTANIM.

Relevant settings include: RK_ROOTFS_BOOTANIM_TIMEOUT for animation timeout, default is 3 seconds.

`device/rockchip/common/overlays/rootfs/bootanim/etc/bootanim.d/gst-video.sh` serves as an example script for playing animations, which by default plays videos located in `/etc/bootanim.d` (if no video is available, it plays snowflakes) to the first screen in full screen.

For special requirements such as multi-screen, you can also modify the `gst-video.sh` script to achieve this. The process involves disabling Weston display at boot time, then calling this script to display the animation, waiting for 3 seconds before killing the script and restoring Weston display.

```

| Symbol: RK_ROOTFS_BOOTANIM [=y]
| Type   : bool
| Prompt: bootanim (Boot-time animation)
|   Location:
|     -> Rootfs
|       -> rootfs (RK_ROOTFS [=y])
|         -> Post rootfs installs
|           -> rootfs overlay (RK_ROOTFS_OVERLAY [=y])

```

- FAQ

If issues arise during the boot animation process, you can add the following line to the video playback script:

```
echo 0xff > /sys/module/drm/parameters/debug
```

This enables the debug log of the display driver (prints all relevant upper-level call information). After reproducing the issue, package and send the `/var/log` directory.

The general logic of the boot animation mechanism is as follows:

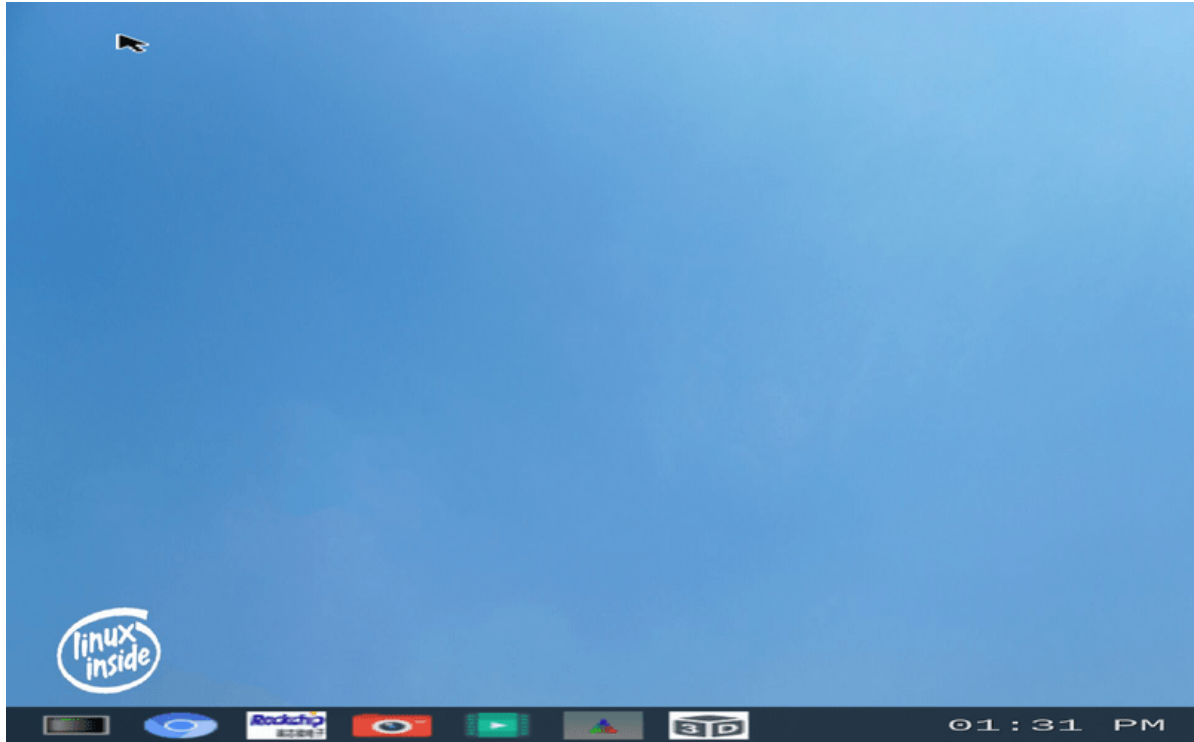
1. bootanim creates a marker file to block Weston from sending display.
2. bootanim plays the animation.
3. Weston starts, draws, but does not send display.

4. When bootanim exits, it pauses the animation, notifies Weston to redraw and send display.
5. bootanim ends the animation.

Some issues may be related to the boot logo. You can add `echo 3 > /sys/class/graphics/fb0/blank` at the beginning of the animation playback script to disable the logo display.

10.10.2 Desktop Application Development

The SDK uses desktop applications based on the X11 protocol such as XFCE, LXDE, GNOME, etc., by default in Debian, while Buildroot/Yocto uses Weston DRM based on the Wayland protocol as the display backend by default. As shown in the figure below:



These Weston applications provide basic functionality configurations such as status bars, backgrounds, Chromium browser, Terminal terminal, Launchers configuration, camera preview, multi-channel video, GPU, mouse, and other demos. For more demos, they can be added through `/etc/xdg/weston/weston.ini.d/*` configuration. For more Weston parameter configurations and usage, refer to [Weston Development](#).

Note: Third-party UI frameworks may involve infringement risks without commercial authorization. If using UI frameworks such as Buildroot QT/Enlightenment/Minigui on the Rockchip platform, third-party authorization and technical support are required, as Rockchip officially does not provide related technical support and maintenance.

The SDK provides three customized UI options: Weston+Wayland, LVGL UI, EFL, Flutter, etc., for desktop development.

10.10.2.1 Weston Desktop Development

Applications related to the Weston client used on Wayland:

```
/usr/bin/#
├─ weston-calibrator
├─ weston-clickdot
├─ weston-cliptest
```

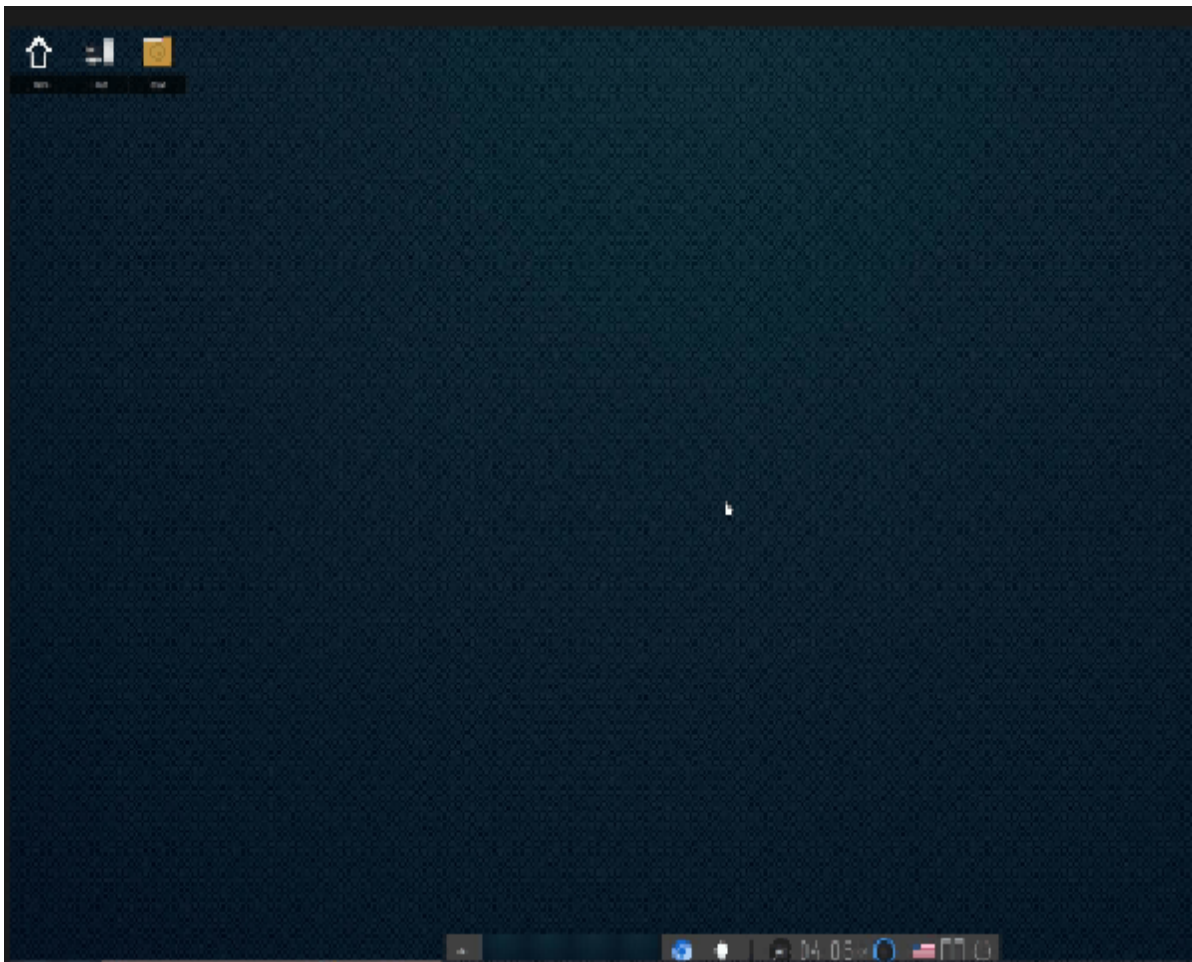
```
|— weston-confine
|— weston-content_protection
|— weston-debug
|— weston-dnd
|— weston-editor
|— weston-eventdemo
|— weston-flower
|— weston-fullscreen
|— weston-image
|— weston-multi-resource
|— weston-presentation-shm
|— weston-resizor
|— weston-scaler
|— weston-screenshooter
|— weston-simple-damage
|— weston-simple-dmabuf-egl
|— weston-simple-dmabuf-v4l
|— weston-simple-egl
|— weston-simple-shm
|— weston-simple-touch
|— weston-smoke
|— weston-stacking
|— weston-surfaces
|— weston-tablet
|— weston-terminal
|— weston-touch-calibrator
|— weston-transformed
```

10.10.2.2 Enlightenment Desktop Development

To enable the necessary configurations, simply set them as follows:

```
BR2_PACKAGE_EFL=y
BR2_PACKAGE_LUAJIT=y
BR2_PACKAGE_EFL_GSTREAMER1=y
BR2_PACKAGE_ENLIGHTENMENT=y
BR2_PACKAGE_EFL_WAYLAND=y
```

The actual running desktop is shown in the figure below:



If your PC prompts this error:

```
2024-03-06T15:20:29 ERR<1307441>:emile ../src/lib/emile/emile_image.c:705
_emile_image_jpeg_error_exit_cb() Wrong JPEG library version: library is 80,
caller expects 90
```

Solution:

```
sudo apt remove libjpeg8-dev
```

It appears there is a conflict with your PC's libjpeg.so*. You can try deleting this library or the corresponding package libjpeg-turbo8.

```
$ whereis libjpeg.so
```

Find libjpeg.so and create a soft link to libjpeg.so.9

```
/usr/lib/x86_64-linux-gnu$ ln -s libjpeg.so.9 libjpeg.so
```

10.10.3 LVGL Desktop Development

To enable the relevant configurations, for example:

```
BR2_PACKAGE_LVGL=y
BR2_PACKAGE_LVGL_COLOR_DEPTH=32
BR2_PACKAGE_LV_DRIVERS=y
BR2_PACKAGE_LV_DRIVERS_USE_DRM=y
BR2_PACKAGE_LVGL_DEMO=y
BR2_PACKAGE_RK_DEMO=y
BR2_PACKAGE_LVGL_DEMO_USE_DRM=y
BR2_PACKAGE_FREETYPE=y
```

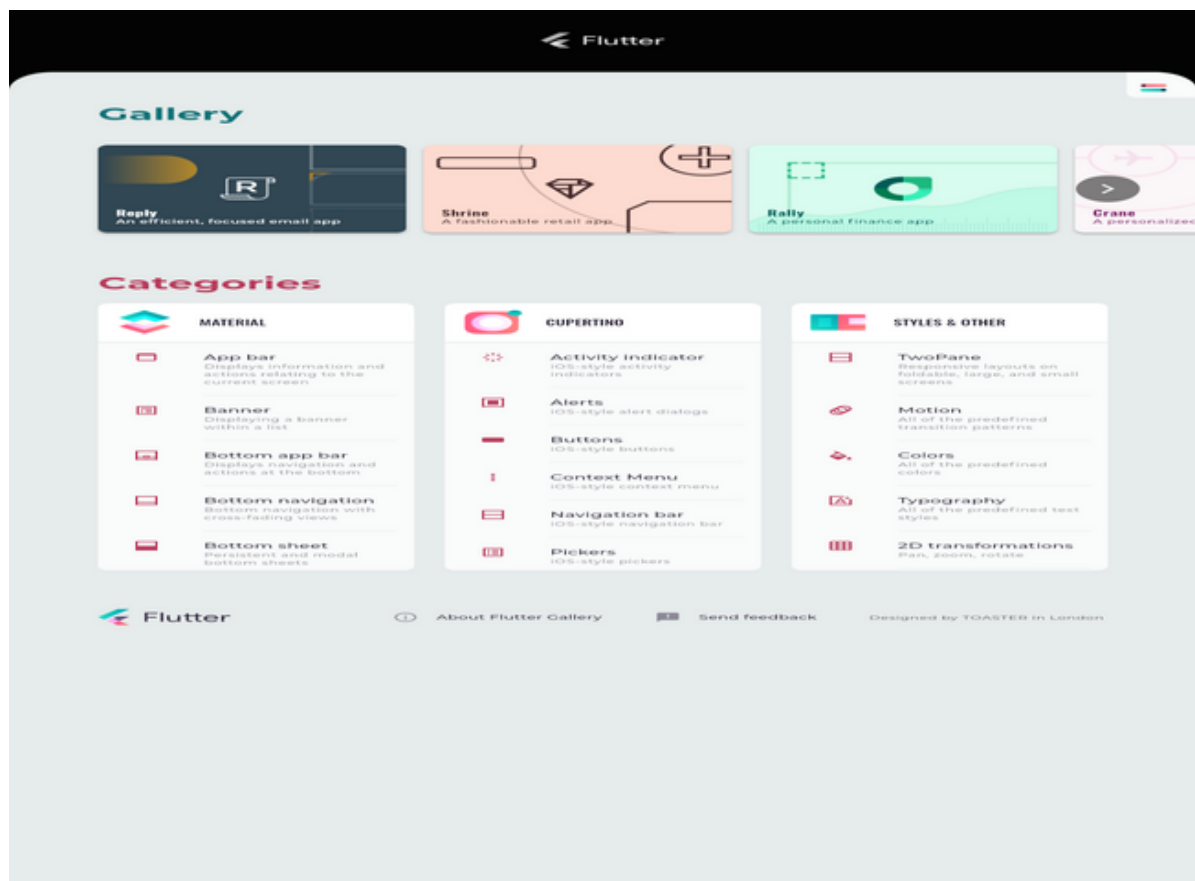


10.10.3.1 Flutter Linux Desktop Development

Ensure that the relevant configurations are enabled, such as:

```
BR2_PACKAGE_FLUTTER_EMBEDDED_LINUX=y  
BR2_PACKAGE_FLUTTER_GALLERY=y
```

```
## Chapter-10 flutter-client -f -b /usr/share/flutter/gallery/release/  
arm_release_ver: g13p0-01eac0, rk_so_ver: 10  
[11:38:01.054] seeing the first app
```

10.11 Secure Development

Security solutions include hardware security (AMPAK, DICE), boot security (Secure Boot, UBM), firmware security (CFI, OP-TEE, CA/TA), secure storage (OPTEE REE, OP-TEE Secure, OP-TEE RPMB), secure verification (Dm-Verity), integrity and anti-tampering (Anti-Rollback, Anti-Copy, dual-key signature verification), hardware isolation (FIREWALL), and other functional developments. For details, refer to the documents in the `<SDK>/docs/en/Linux/Security` directory.

Current Security Features Supported by Each Chip

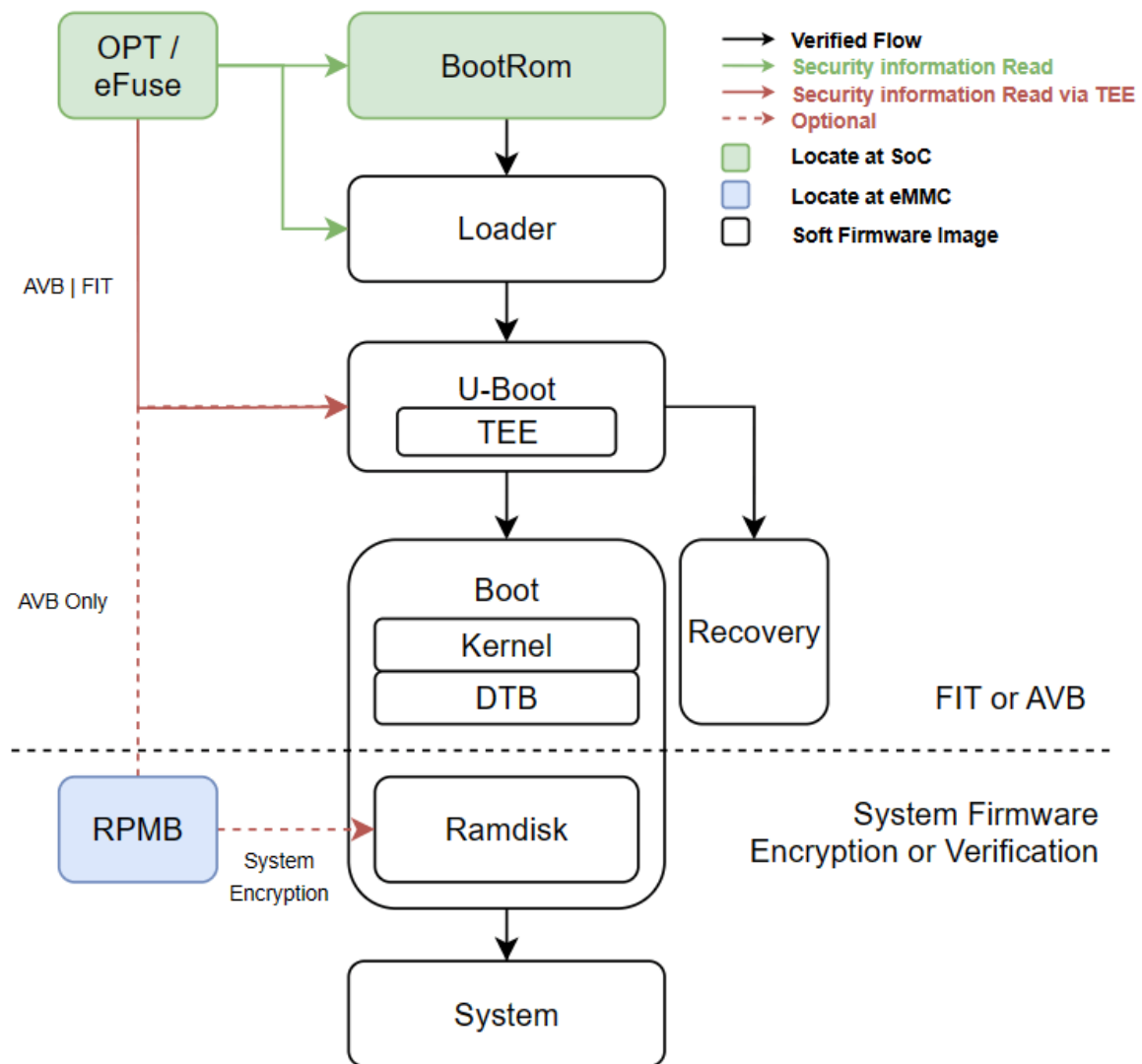
Feature	RK3588	RK3576	RK3568/RK3566	RK3562	Remarks
Secure Boot	√	√	√	√	Supports FIT and AVB schemes, full chain verification. Requires SecureBoot + AB system (mandatory)
CFI	-	-	-	-	Bootrom Control Flow Integrity, prevents hijacking control flow attacks
KEYLAD	-	-	-	-	Hardware Protect Key, CPU cannot read the key
Anti-Rollback	-	-	-	-	Anti-rollback protection (including Bootrom, TA, Uboot, etc., at all levels)
Anti-Copy	√	√	√	√	Anti-cloning protection (secure OTP writes private data, software verification, independent of SecureBoot)
Dual-key Signature Verification	√	√	√	√	BL_KEY0 verifies BL_KEY1, BL_KEY1 verifies the firmware
DICE	√	√	√	√	Device Identifier Composition Engine, only implemented in bootrom
UBM	-	-	-	-	Bootrom Unique Boot Mode OTP configures unique boot storage
FIREWALL	-	√	-	-	Configures secure and non-secure isolation, such as isolating OP-TEE secure memory, non-secure cannot access
OP-TEE	√	√	√	√	Version 3.13.0 (tls implemented in teeos, customers share related documents)

Feature	RK3588	RK3576	RK3568/RK3566	RK3562	Remarks
OP-TEE REE Secure Storage	√	√	√	√	-
OP-TEE RPMB Secure Storage	√	√	√	√	-
OP-TEE Secure Partition Secure Storage	√	√	√	√	Suitable for flash without RPMB, such as nand/nor flash OP-TEE secure memory, non-secure cannot access
TA/CA	√	√	√	√	Version 3.13.0 (tls implemented in teeos, customers share related documents)
TA Encryption	√	√	√	√	-
TA Anti-Rollback	√	√	√	√	-
KEYBOX	√	√	√	√	System encryption: ramdisk reads the secret key (maintain current)
DM-VERIFY	√	√	√	√	-
DM-CRYPT	√	√	√	√	-
Partition Encryption	√	√	√	√	Only supports system partition encryption, Loader/Uboot/Kernel: TBD
SELinux	-	-	-	-	Kernel support, SDK not deployed

PS: A checkmark for RK3588 indicates that the chip supports it; black highlighting indicates that the current SDK has been supported.

10.11.1 Secure Boot Initialization

Secure Boot is a mechanism designed to protect systems from malicious software and unauthorized modifications, ensuring that only software signed by the manufacturer or developer can run at startup. Below are the general steps of the Secure Boot process.



By default, Secure Boot is not enabled. If you need to verify, follow these steps:

- SDK to enable security configuration

```
$ make menuconfig
Enable Security ---> [*] security feature
    --- Security feature (secureboot, encryption, verity, etc.)
        security check method (base|system-encryption|system-verity) (system-
encryption) --->
```

There are three security mechanisms by default: base, system-encryption, and system-verity. Base is the basic version without encryption, system-encryption provides encryption functionality, and system-verity provides verification functionality.

- SDK to save security configuration

```
$ make savedefconfig
```

For example, using RK3576 as a reference:

```
--- a/.chips/rk3576/rockchip_rk3576_evb1_v10_defconfig
+++ b/.chips/rk3576/rockchip_rk3576_evb1_v10_defconfig
@@ -1,4 +1,5 @@
 RK_ROOTFS_PREBUILT_TOOLS=y
-RK_UBOOT_SPL=y
 RK_KERNEL_DTS_NAME="rk3576-evb1-v10-linux"
+RK_SECURITY=y
+RK_SECURITY_CHECK_SYSTEM_ENCRYPTION=y
```

10.11.2 Compiling Secureboot

- Directly run `./build.sh` and follow the compilation error messages to proceed step by step.

```
./build.sh
```

```
=====
system-encryption
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
      some operations need super user permission when create encrypt image

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

For example, if the PC password is test0000:
echo -n "test0000" > u-boot/keys/root_passwd
```

```
./build.sh
```

```
=====
system-encryption
ERROR: No enc key(u-boot/keys/system_enc_key) found in u-boot
      Create it by ./build.sh security-createkeys or move your key to it
```

```
./build.sh security-createkeys
```

```
./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_DM_VERITY
CONFIG_TEE
CONFIG_OPTEE
```

Add the relevant configurations to the 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_DM_VERITY=y
+CONFIG_TEE=y
+CONFIG_OPTEE=y
```

```
./build.sh
```

Security: No found config CONFIG_FIT_SIGNATURE in /home/wxt/linux-develop/rockchip-linux/u-boot/.config
make sure your config include this list

```
-----
CONFIG_FIT_SIGNATURE
CONFIG_SPL_FIT_SIGNATURE
```

Add the relevant configurations to u-boot:

```
u-boot$ git diff
diff --git a/configs/rk3576_defconfig b/configs/rk3576_defconfig
index e9b5dbf1210..35925f8d5df 100644
--- a/configs/rk3576_defconfig
+++ b/configs/rk3576_defconfig
@@ -23,6 +23,8 @@ CONFIG_DEBUG_UART=y
 CONFIG_FIT=y
 CONFIG_FIT_IMAGE_POST_PROCESS=y
 CONFIG_FIT_HW_CRYPT=y
+CONFIG_FIT_SIGNATURE=y
+CONFIG_SPL_FIT_SIGNATURE=y
 CONFIG_SPL_LOAD_FIT=y
 CONFIG_SPL_FIT_IMAGE_POST_PROCESS=y
 CONFIG_SPL_FIT_HW_CRYPT=y
```

```
./build.sh
```

CONFIG_DM_VERITY is enabled in kernel!
Please set "GROW_ALIGN: 1" in rockchip-linux/device/rockchip/.chip/parameter.txt:

```
device/rockchip$ git diff
diff --git a/.chips/rk3576/parameter.txt b/.chips/rk3576/parameter.txt
```

```

index a2c21a9d..dec0979d 100644
--- a/.chips/rk3576/parameter.txt
+++ b/.chips/rk3576/parameter.txt
@@ -8,7 +8,7 @@ MACHINE: 0xffffffff
CHECK_MASK: 0x80
PWR_HLD: 0,0,A,0,1
TYPE: GPT
-GROW_ALIGN: 0
+GROW_ALIGN: 1

```

```
./build.sh
```

No found optee node in dts

Please add:

```

    optee: optee {
        compatible = "linaro,optee-tz";
        method = "smc";
        status = "okay";
    };

```

To kernel dts

```

--- a/arch/arm64/boot/dts/rockchip/rk3576-linux.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3576-linux.dtsi
@@ -22,6 +22,13 @@ fiq_debugger: fiq-debugger {
    status = "okay";
};

+    firmware {
+        optee: optee {
+            compatible = "linaro,optee-tz";
+            method = "smc";
+        };
+    };
+

```

```
./build.sh
```

Security: No found BR2_ROOTFS_OVERLAY+="board/rockchip/common/security-system-overlay/" in system config

Add the relevant configuration to buildroot:

```

buildroot$ git diff
diff --git a/configs/rockchip_rk3576_defconfig
b/configs/rockchip_rk3576_defconfig
index 5b47b07c43..9dd11af1cc 100644
--- a/configs/rockchip_rk3576_defconfig
+++ b/configs/rockchip_rk3576_defconfig
@@ -23,3 +23,4 @@
#include "npu2.config"
#include "powermanager.config"
#include "weston.config"
+BR2_ROOTFS_OVERLAY+="board/rockchip/common/security-system-overlay"

```

```
Security: No found config BR2_PACKAGE_RECOVERY in rockchip-  
linux/buildroot/output/rockchip_rk3576_ramboot/.config
```

Add the relevant configuration to buildroot:

```
diff --git a/configs/rockchip_rk3576_ramboot_defconfig  
b/configs/rockchip_rk3576_ramboot_defconfig  
index 46a27b3a9e..9b344004fe 100644  
--- a/configs/rockchip_rk3576_ramboot_defconfig  
+++ b/configs/rockchip_rk3576_ramboot_defconfig  
@@ -1,6 +1,9 @@  
 #include "chips/rk3576_aarch64.config"  
 #include "base/kernel.config"  
+#include "tee_aarch64_v2.config"  
 BR2_TARGET_ROOTFS_CPIO=y  
 BR2_TARGET_ROOTFS_CPIO_GZIP=y  
 BR2_PACKAGE_LUKSMETA=y  
+BR2_PACKAGE_RECOVERY=y  
+BR2_PACKAGE_RECOVERY_UPDATEENGINEBIN=y
```

```
./build.sh
```

After running the `./build.sh` command, the generated `update.img` firmware in the `rockdev` directory can be used for system updates. Simply start the system normally, and as long as the `vroot` partition is properly mounted, the secureboot feature is enabled. You can check the mounting status of the `vroot` partition by executing the `df -h` command:

```
root@rk3576-buildroot:/# df -h  
Filesystem      Size  Used Avail Use% Mounted on  
/dev/mapper/vroot 924M  799M   62M  93% /
```

- FAQ

1. Environment Installation

If the compilation reports an error `No module named Crypto.Signature`, this is due to the development computer not having the python cryptographic library installed. Execute the following commands:

```
pip uninstall Crypto  
pip uninstall pycrypto  
pip install pycrypto
```

If the following error occurs: `ModuleNotFoundError: No module named 'Cryptodome'`
Please install the python package on the development host: `pip3 install [--user] pycryptodomex`

2. When verifying, find a machine that has not been burned with `rpmb`. If there is an encryption key in the machine, use the encryption key in the machine first. If the machine's encryption key is different from the one we compiled, the system will fail to start.
3. The encryption key is placed in the `rpmb` area and can be rewritten repeatedly. If you encounter a situation where the key is different and the system cannot be loaded, you can open `FORCE_KEY_WRITE=true` in `buildroot/board/rockchip/common/security-ramdisk-overlay/init.in` to force an update of the key.

For more details, refer to:

Linux Secureboot Software Development Guide:

`<SDK>/docs/en/Linux/Security/Rockchip_Developer_Guide_Linux_Secure_Boot_EN.pdf`

U-Boot Secureboot Software Development Guide:

`<SDK>/docs/en/Common/SECURITY/Rockchip_Developer_Guide_Secure_Boot_for_UBoot_Next_De
v_EN.pdf`

TEE Development Guide:

`<SDK>/docs/en/Common/SECURITY/Rockchip_Developer_Guide_TEE_SDK_EN.pdf`

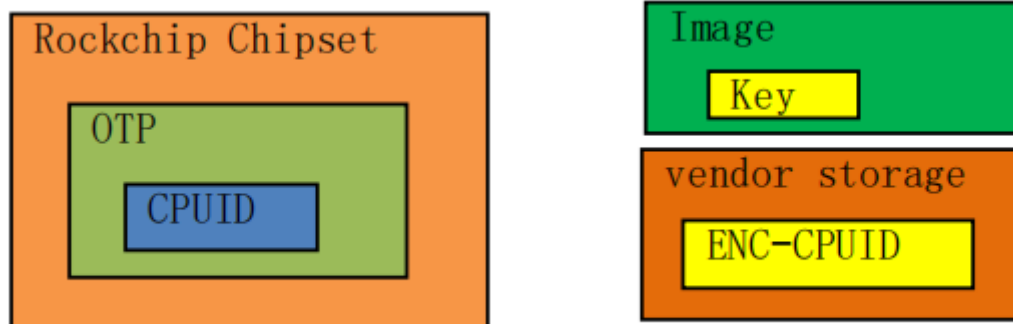
10.11.3 Rockchip Anti-Cloning Feature

Rockchip chips offer anti-cloning technology to protect customers' firmware, private data, and core code. This technology is primarily used to prevent unauthorized users from illegally copying and using customers' firmware and private data, thus avoiding commercial losses due to cloning.

Currently, there are three schemes provided for anti-cloning technology:

- Basic Scheme

During production, Rockchip chips burn a CPUID into the OTP, and each chip has a unique CPUID. Customers can read the CPUID, encrypt it with a symmetric key Key to obtain ENC-CPUID, and store the ENC-CPUID in the vendor storage partition. After the system boots up, it reads the ENC-CPUID from the vendor storage partition, decrypts it to obtain DEC-CPUID, and compares DEC-CPUID with CPUID. If the match fails, it is considered illegal, and the device is restarted to achieve the anti-cloning purpose.



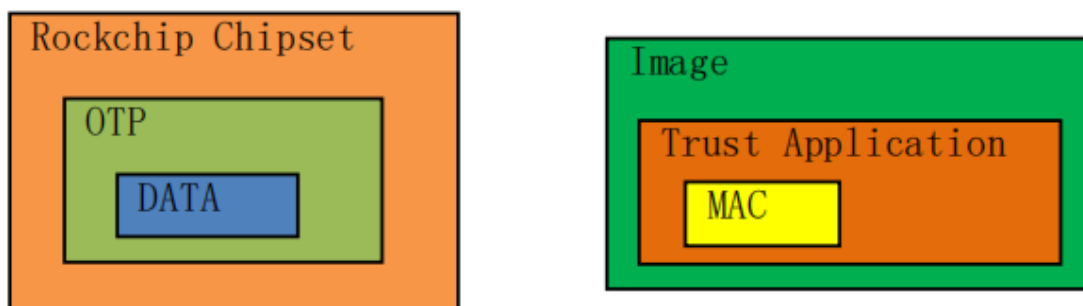
The security of this scheme depends on the protection of the key Key. Customers should avoid hardcoding the plaintext key directly in the code. It is recommended to obfuscate the key before hardcoding it into the code to prevent illegal users from obtaining the plaintext key through disassembly.

Additionally, since ENC-CPUID is stored in the vendor storage partition, erasing the flash will clear the vendor storage partition and result in the loss of ENC-CPUID. Therefore, after erasing the flash, it is necessary to rewrite the ENC-CPUID.

- Intermediate Scheme

Rockchip chips have reserved space in the OTP for customers to store anti-cloning private data. Customers generate custom private data DATA and MAC, and a custom-specific function func is used, where DATA and MAC satisfy a specific functional transformation relationship, such as $MAC = func(DATA)$. Customers can write anti-cloning private data DATA into the OTP area and hardcode MAC into the trusted application TA (Trust Application) code. After the system boots up, the trusted application TA reads the anti-cloning private data DATA from the OTP, and the trusted application TA checks if there is a given functional transformation relationship

between DATA and MAC. If the match fails, it is considered illegal, and the device is restarted to achieve the anti-cloning purpose.



The security of this scheme depends on the protection of the anti-cloning private data DATA. The kernel cannot read this data, only the trusted application TA (Trust Application) can read it. Customers should refer to the "TA Signature" section in the document "Rockchip_Developer_Guide_TEE_SDK_EN.md" and sign the trusted application TA (Trust Application) with their own key to prevent illegal TA from running and causing leakage of anti-cloning private data DATA.

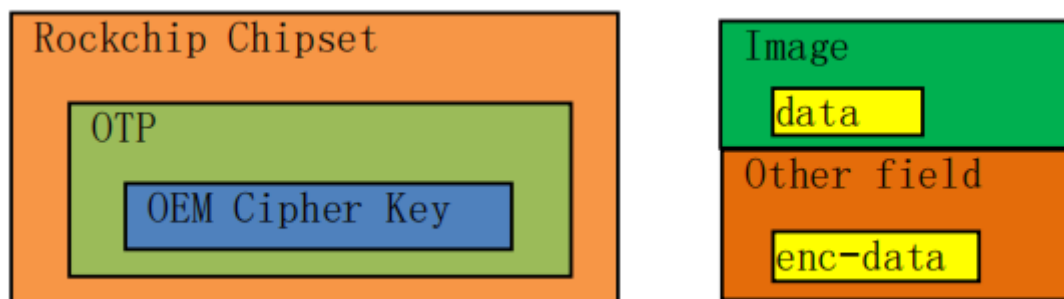
- Advanced Scheme

Rockchip chips have reserved space in the OTP for customers to store custom private keys. For usage, please refer to the "OEM Cipher Key" section in

[docs/en/Common/SECURITY/Rockchip_Developer_Guide_OTP_EN.pdf](#).

The anti-cloning technology relies on the customer's private key OEM Cipher Key stored in the chip's OTP. This key is written during the factory production phase. To ensure the key is not leaked, the system only provides a writing interface without a reading interface. Some platforms also support the Hardware Read feature to lock the key, making it unreadable by the CPU, thus offering higher security.

Customers can customize plaintext data data, use the private key OEM Cipher Key, and encrypt data with the AES algorithm to obtain encrypted data enc-data, which is then stored in a specified location on the flash or hardcoded in the code. The application code sets plaintext data data. After the system boots up, it first decrypts the encrypted data enc-data with the private key OEM Cipher Key to obtain decrypted data dec-data and compares it with the plaintext data data in the application code. If the plaintext data data or encrypted data enc-data is tampered with, or if the flash firmware is illegally copied to other unauthorized chip platforms, the decrypted data dec-data will not match the plaintext data data. A match failure is considered illegal, and the device is restarted to achieve the anti-cloning purpose.



The advantage of this scheme is that the customer's private key OEM Cipher Key, once locked, cannot be read by the CPU, offering higher confidentiality, integrity, and non-tampering security features.

Customers can freely choose to use any one of the schemes.

For more details, please refer to the anti-cloning document

[docs/en/Common/SECURITY/Rockchip_Developer_Guide_Anti_Copy_Board_EN.pdf](#).

The anti-cloning tool BoardProofTool is located at

```
tools/windows/BoardProofTool_v1.0_20231201.zip.
```

10.12 WIFI/BT Development

Refer to the documentation in the `<SDK>/docs/en/Linux/Wifibt` directory.

10.13 Post-Processing Development for ROOTFS

The Post rootfs option is primarily used for post-processing of the root file system, including performance tuning, feature support, and debugging assistance, to meet various development and deployment needs. Functional configuration can be done through the SDK menuconfig. Below is an example of using menuconfig in the SDK directory:

```
<SDK>$ make menuconfig
[*] Rootfs (Buildroot|Debian|Yocto) --->
  Post rootfs installs --->
    [*] create /etc/ld.so.cache
        hostname (auto) --->
        locale (auto) --->
    [*] strip kernel modules
  [*] async-commit DRM driver hack
  [*] debug information dir (/info/)
  [*] Rockchip udev rules
  [*] Wi-Fi/BT (Kernel modules, firmwares and scripts) --->
      disk helpers (Boot-time mounting and resizing) (auto) --->
  [ ] format extra partitions when needed
  [ ] bypass boot time fsck
  [*] log guardian (Truncate logs when disk is full) --->
```

Rockchip Post rootfs development options provide a variety of features for customizing and optimizing the root file system, mainly including:

1. System Configuration

- Create dynamic link library cache to accelerate program loading
- Set hostname and localization environment
- Remove debugging symbols from kernel modules to reduce module size

2. Driver Support

- Apply a temporary fix for DRM drivers
- Add udev rules for Rockchip devices to enable device recognition and configuration
- Install Wi-Fi and Bluetooth related kernel modules, firmware, and scripts

3. File System Management

- Create a debug information directory to store related files
- Disk helper for mounting and resizing file systems at boot time
- Option to format additional partitions when needed
- Option to bypass file system integrity check at boot time

4. Log Management

- Log monitoring feature to truncate log files when disk space is insufficient

Through these options, you can customize and optimize the root file system on Rockchip devices according to actual needs, improving system performance, feature support, and debugging capabilities. These options provide developers with flexible customization space, helping to meet special needs in different scenarios.

10.14 Overlays Development

Rockchip Overlays offers a variety of feature options, including USB device support, fonts, input event handling, boot animation, interrupt balancing, file system optimization, virtual terminals, prebuilt tools, and custom overlays. You can configure these features through the SDK menuconfig.

Here is an example of using menuconfig in the SDK directory:

\$ make menuconfig

```
[*] Rootfs (Buildroot|Debian|Yocto) --->
  [*] Overlays --->
    [*] USB gadget --->
      extra fonts (auto) --->
      input-event-daemon (power-key handling) (auto) --->
      [ ] bootanim (Boot-time animation)
      [*] irqbalance (Balance hardware IRQs)
      [*] fstrim (Discard unused blocks on all filesystems) --->
      [ ] frecon virtual terminal (VT)
      [*] prebuilt tools
      () extra overlay dirs
```

For instance, if you need to enable the MTP feature in USB device support, you can configure it as follows:

```
[*] Rootfs (Buildroot|Debian|Yocto) --->
  [*] Overlays --->
    [*] USB gadget --->
      [*] Media Transfer Protocol (MTP) --->
        (devicon.ico) device icon (NEW)
        (umtprd.conf) umtprd config file
```

After the configuration is complete, save the configuration and compile:

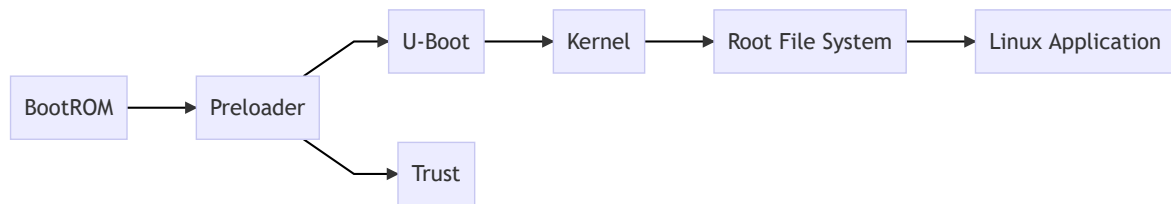
make savedefconfig && ./build.sh

In this way, you can enable or disable different Overlays features according to your needs and make corresponding configurations to meet specific development requirements.

10.15 SDK Launch Method

10.15.1 Device Boot Process

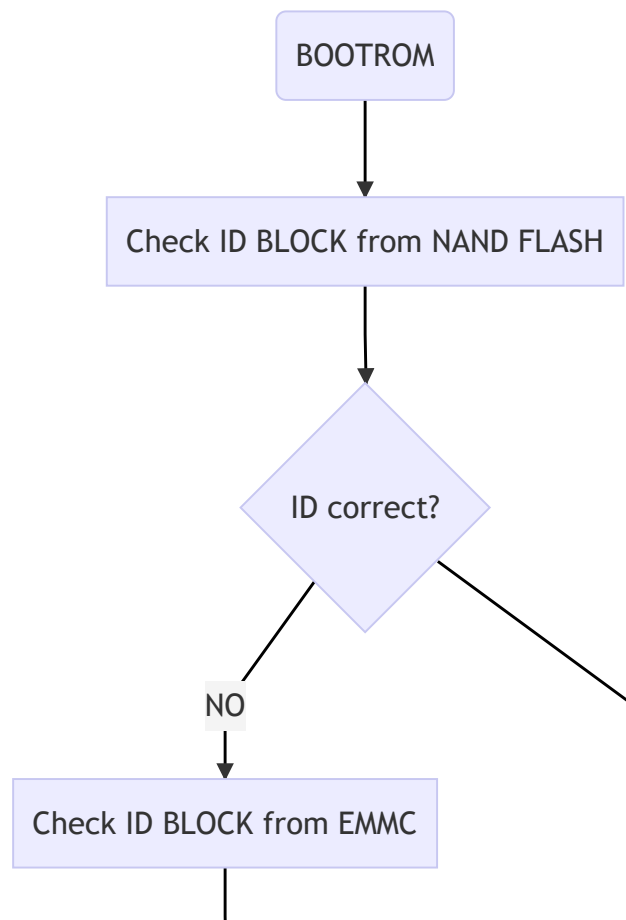
The boot process refers to the software procedure from system power-on to system startup completion. Below is the Linux system boot process:

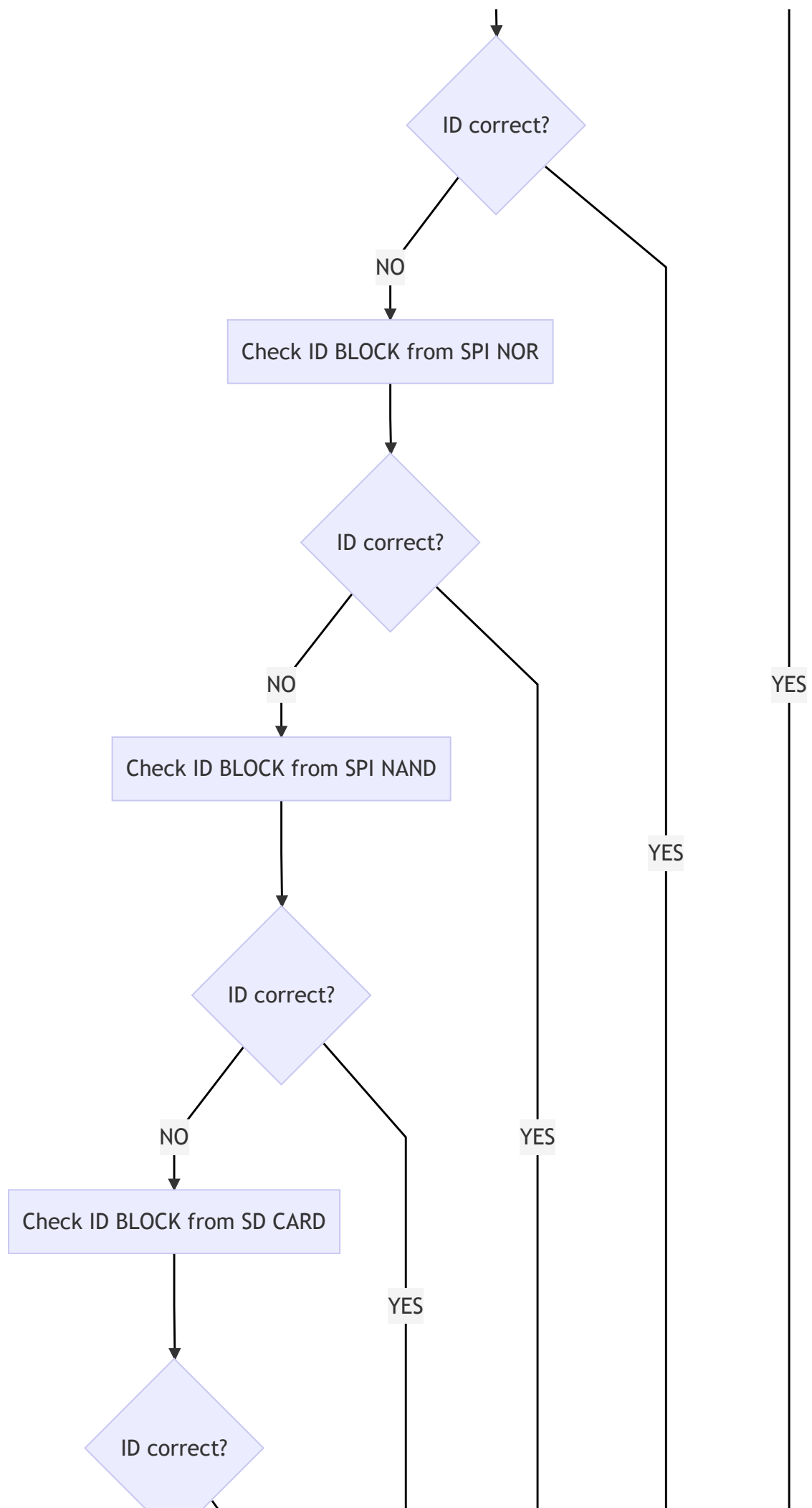


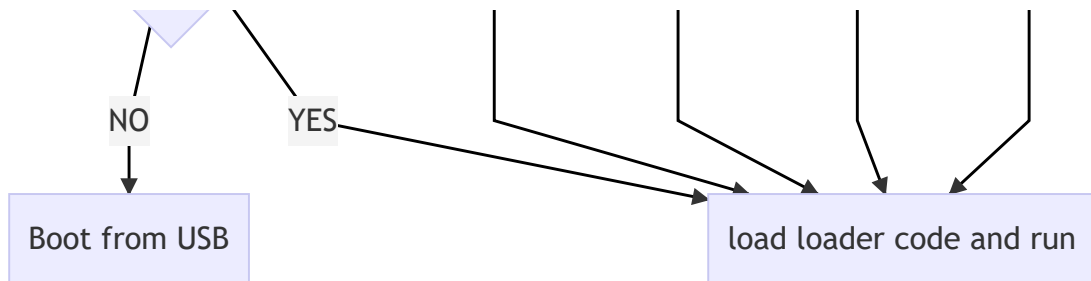
10.15.2 BOOTROM Process

After power-on, the latest executable code on the Rockchip chip is the mask code integrated within the chip, which cannot be changed. This is the BOOTROM code, which is integrated into both the AP and MCU. The BOOTROM code runs first upon system power-up, and then it detects peripheral storage devices and loads the Loader code from them.

Different chips have different sequences for detecting peripheral storage devices. The following diagram is an example of the BOOTROM boot process:







Explanation:

- Some chips support ADC Key with different input voltage levels to specify the storage device detection order in BOOTROM. If the storage detection fails, the device directly enters maskrom mode.
- Detecting storage devices is usually done by detecting the device ID to confirm the presence of an external device.
- If all devices fail to detect valid firmware, the device enters maskrom mode and waits for specific interfaces such as USB/UART to download firmware. Not all chips support USB/UART interface upgrades.

BootROM Boot Order for Various Chips

AP	No.1	No.2	No.3	No.4	No.5	No.6
RK3128	NAND	EMMC	SPI NOR(SFC)	SPI NAND(SFC)	SD0	USB
RK3126(B)	NAND	EMMC	SPI NOR(SFC)	SPI NAND(SFC)	SD0	USB
RK3036	NAND	EMMC	SPI NOR(SFC)	SPI NAND(SFC)	SD0	USB
RK3288	NAND	EMMC	SPI NOR(SPI2)	SPI NAND(SPI2)	SD0	USB
RK3399	SPI NOR(SPI2)	SPI NAND(SPI2)	EMMC	SD0	USB	--
RK3328	EMMC	SPI NOR(SPI2)	SPI NAND(SPI2)	SD0	USB	--
RK3128X/H	NAND	EMMC	SPI NOR(SPI2)	SPI NAND(SPI2)	SD0	USB
RK3308	NAND	EMMC	SPI NOR(SFC)	SPI NAND(SFC)	SD0	USB
RK3326	NAND	EMMC	SPI NOR(SFC)	SPI NAND(SFC)	SD0	USB
RK1808	SPI SLAVE	SPI NOR(SFC)	SPI NAND(SFC)	EMMC	USB	--
RK3399PRO	SPI NOR(SPI2)	SPI NAND(SPI2)	EMMC	SD0	USB	--
RK3568	SPI NOR(SFC)	SPI NAND(SFC)	NAND	EMMC	SD0	USB
RK3566	SPI NOR(SFC)	SPI NAND(SFC)	NAND	EMMC	SD0	USB
RK3588	SPI NOR(SFC)	SPI NAND(SFC)	EMMC--	SD0	USB	
RK3528	SPI NOR(SFC)	SPI NAND(SFC)	EMMC--	SD0	USB	
RK3562	SPI NOR(SFC)	SPI NAND(SFC)	EMMC--	SD0	USB	

10.15.3 System Boot Methods

Currently, the Linux SDK system provides three boot methods: Sysv, Busybox, and Systemd init.

Yocto systems default to using the Sysv init method for managing boot startup, utilizing the `update-rc.d.bbclass` to manage service boot startup configurations.

Buildroot systems default to using the Busybox init method for managing boot startup, which reads the `/etc/inittab` file after booting.

Debian systems default to using the Systemd method for managing boot startup. Systemd init reads related services in the `/etc/systemd/system/` directory after booting.

The SDK has different processing for different startup 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
```

10.16 Shared Remote Desktop Development

Remote desktop is a technology that allows users to access and control the desktop of another computer (server) through a computer (client). This technology enables users to run applications on the remote computer, manipulate files, and use devices as if they were operating the computer directly.

10.16.1 Controlling Remote Desktop with wlfreerdp

wlfreerdp is a Wayland Remote Desktop Protocol (RDP) client. It enables users to remotely connect to Windows servers or personal computers from Linux systems, allowing them to access the Windows desktop environment remotely.

PC End: Ubuntu22.04 xrdp (0.9.17)

Device: RK3588 + buildroot + wlfreerdp (2.11.5)

PC End: Share --> Remote Desktop, enable remote desktop, the device name is domain, the username is Username, and the password is Password.

```
root@rk3588-buildroot:/# wlfreerdp /f/v:172.16.15.86 /u:wxt /p:test0000 /d:pc
```

```
[08:00:27:928] [1872:1872] [WARN][com.freerdp.crypto] - Certificate verification failure 'self-signed certificate (18)' at stack position 0
```

```
[08:00:27:928] [1872:1872] [WARN][com.freerdp.crypto] - CN = GNOME, C = US
```

```
[08:00:28:237] [1872:1872] [INFO][com.freerdp.gdi] - Local framebuffer format PIXEL_FORMAT_BGRA32
```

```
[08:00:28:237] [1872:1872] [INFO][com.freerdp.gdi] - Remote framebuffer format PIXEL_FORMAT_BGRA32
[08:00:28:258] [1872:1872] [INFO][com.freerdp.channels.rdpnd.client] - [static] Loaded fake backend for
rdpsnd
[08:00:28:259] [1872:1872] [INFO][com.freerdp.channels.drdynvc.client] - Loading Dynamic Virtual Channel
rdpgfx
..
```

In buildroot, enabling `BR2_PACKAGE_FREERDP=y` is sufficient.

10.16.2 Weston VNC Remote Desktop Access

- Enable Weston VNC

To enable Weston VNC in Buildroot:

Ensure that the configuration option `BR2_PACKAGE_WESTON_VNC` is enabled in Buildroot.

Start the Weston VNC service:

On your system development board, run the following command to start the Weston VNC service:

```
weston -B drm,vnc &
```

This will launch the Weston display server and attach the VNC service.

- Remote Access:

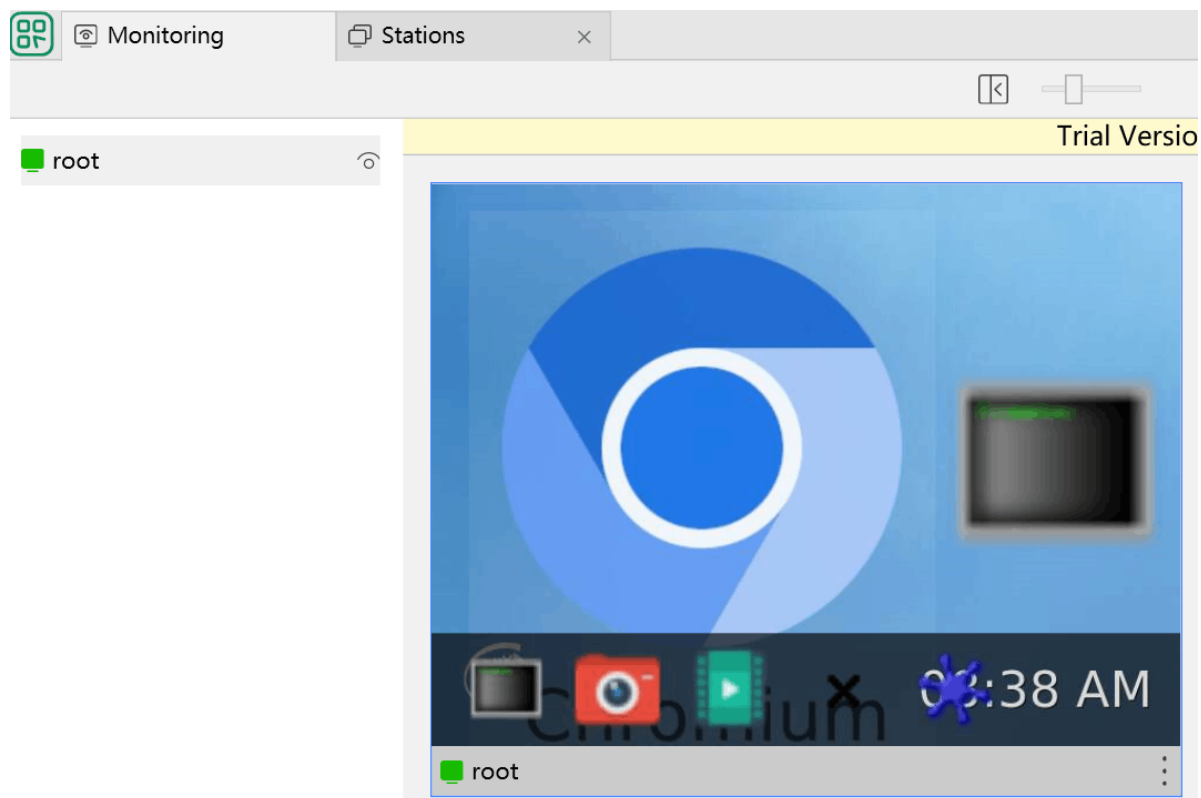
To access the Weston VNC service from a remote location, you can install the MightyViewer software on your Windows system, which is a VNC client.

- Connection and Control:

Use MightyViewer to connect to the IP address of your development board and begin remote control and viewing of the desktop environment.

Screenshot Example

Below is a screenshot example of connecting to the Weston VNC service using MightyViewer:



10.17 SDK Testing

10.17.1 Integrating Rockchip Stress Test Scripts

The `rockchip_test` integrates functionality, stress, and performance-related testing.

```

      ROCKCHIPS TEST TOOLS
172.16.15.86
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 module:

```

10.17.2 Benchmark Testing

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

```
<SDK>/docs/en/Linux/Profile/Rockchip_Introduction_Linux_Benchmark_KPI_EN.pdf
```

10.17.3 Rockchip Modules and Stress Testing

Below are some common module functionalities and stress testing methods. You can find more detailed information in the following documentation:

```
<SDK>/docs/en/Linux/Profile/Rockchip_User_Guide_Linux_Software_Test_EN.pdf
```

11. Chapter-11 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/
├─ adb
├─ 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
```

11.1 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](#).

11.1.1 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

11.1.2 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. 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.

11.2 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-11 ./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, bzip2, 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, fallocation,
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, rmmmod, route, run-init, run-parts, runlevel, sed, seedrng, seq,
setarch, setconsole, setfattr, setkeycodes, setlogcons, setpriv,
setserial, setsid, sh, shasum, 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
```

11.3 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.

11.4 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.

11.5 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.

11.6 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.
```


11.7 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.

11.8 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
```

11.9 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.

11.10 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
```

11.11 Slabtop Tool

Display kernel slab buffer information in real time.

Refresh the slab buffer information every ten seconds:

```
slabtop -d 10
```

11.12 Strace Tool

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

11.13 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.

11.14 Update Tool

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

11.15 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

```

11.16 Vmstat Tool

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

11.17 Watch Tool

Watch can help monitor the results of a command.

```
## Chapter-11 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

```

11.18 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

```

11.19 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
...

```

12. Chapter-12 SDK Software License Instructions

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

12.1 Copyright Detection Tool

12.1.1 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.

12.1.2 Debian

For license detection, please refer to [Official Tool](#)

```
licensecheck --check '.*' --recursive --copyright --deb-fmt \
--lines 0 * | /usr/lib/cdb/lscdb/lscdb2dep5
```

The relevant copyright instructions for each source code package are located in

```
/usr/share/doc/*/copyright
```

12.1.3 Yocto

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

12.2 License List

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

[license-list](#)

13. Chapter-13 Rockchip open source information

13.1 Github

Rockchip Code Open Source repositories [rockchip-github](#)。

13.2 Wiki

Rockchip open source materials [rockchip-wiki](#)

13.3 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
```


14. Chapter-14 SDK FAQ

The following offers answers and clarifications to common questions about various aspects of the SDK. Here are some potential questions that might be included in an SDK FAQ.

14.1 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"
```

14.2 Issues about SDK Building

14.2.1 Sync Issues Caused by Repo

During the update with `repo sync -c`, the prompt `No module named formatter` occurs because your host machine is using a newer version of Python, such as Python 3.8+, which has completely removed the `formatter` module. This issue is caused by the outdated version of the `repo` tool in the external SDK. To resolve this problem, updating the version of `repo` is necessary. An example of how to do this is as follows:

```
$ 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
```

14.2.2 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"
```

14.2.3 Compilation issue in the Docker environment with ./build.sh

During compilation, a permission issue is prompted:

rm: cannot remove '/home/zora/myjob/linux/rk3568/output/sessions/latest': Permission denied

Sometimes due to issues with copying SDK source code or the PC environment, it is necessary to change the default permissions to 'zora' (a regular username):

```
chown zora:zora -R .
```

Then proceed with other operations.

14.2.4 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.

14.3 Recovery FAQ

14.3.1 How to prevent recovery from being compiled into the update.img

Newer SDKs support disabling recovery; to do this, navigate to make config and deselect the recovery option.

The packaging of update.img is based on the package-file description. Early SDKs used a predefined package-file located in the tools directory. For specific instructions, refer to the relevant documentation and remove the recovery entry.

In newer SDKs, the package-file is automatically generated before packaging the update.img. It is based on the partition descriptions in the parameter partition and the partition images with the same name in the output/firmware directory during the packaging process. If they match, they are automatically added to the package-file and then packaged into the update.img. Therefore, you can either delete the recovery partition in the used parameter.txt or manually delete the output/firmware/recovery.img before running make updateimg to package it.

Newer SDKs also allow you to use make edit-package-file to directly edit the package-file and adjust which images need to be packaged.

14.4 Buildroot FAQ

14.4.1 Dual Screen Asynchronous Clock Times

The clock format can be configured in `/etc/xdg/weston/weston.ini` to compare in seconds:

clock-format=seconds

The default configuration updates the time every minute separately, initiating the update when a specific screen is enabled. The first update aligns with the current time to the nearest whole minute.

From the previous description, it seems likely that the screens were not enabled together, and then the system time changed (either through network synchronization or manual setting), causing the two to be out of sync.

If minute-level accuracy is essential and this scenario needs to be considered, you could modify Weston. Otherwise, using the second-level format should suffice.

```
diff --git a/clients/desktop-shell.c b/clients/desktop-shell.c
index bf75927..67289e2 100644
--- a/clients/desktop-shell.c
+++ b/clients/desktop-shell.c
@@ -531,12 +531,16 @@ panel_launcher_tablet_tool_button_handler(struct widget
*widget,
```

```

        launcher_activate(launcher->argp, launcher->envp);
    }

+static int clock_timer_reset(struct panel_clock *clock);
+
    static void
    clock_func(struct toytimer *tt)
    {
        struct panel_clock *clock = container_of(tt, struct panel_clock, timer);

        widget_schedule_redraw(clock->widget);
+
+    clock_timer_reset(clock);
    }

    static void
@@ -589,7 +593,7 @@ clock_timer_reset(struct panel_clock *clock)
        clock_gettime(CLOCK_REALTIME, &ts);
        tm = localtime(&ts.tv_sec);

-    its.it_interval.tv_sec = clock->refresh_timer;
+    its.it_interval.tv_sec = 0;
        its.it_interval.tv_nsec = 0;
        its.it_value.tv_sec = clock->refresh_timer - tm->tm_sec % clock-
>refresh_timer;
        its.it_value.tv_nsec = 10000000; /* 10 ms late to ensure the clock digit has
        actually changed */

```

14.4.2 How to Set Buildroot System to Boot into Command Line Mode

To disable Weston in buildroot (remove the include of weston.config in the corresponding board defconfig), then:

- Enable CONFIG_VT, CONFIG_VT_CONSOLE, CONFIG_FRAMEBUFFER_CONSOLE, CONFIG_DRM_FBDEV_EMULATION in kernel defconfig.
- Enable BR2_PACKAGE_FRECON, BR2_PACKAGE_FRECON_VT in buildroot.

The difference is that frecon supports rotation, scaling, and Chinese characters, but it can only display on a single screen.

14.5 Debian FAQ

This chapter is going to answer some frequently asked questions about Debian GNU/Linux based on the Rockchip platform. For other questions, please refer to the official website [Debian FAQ](#).

14.5.1 "noexec or nodev" Issue

```

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
(The xxx is the path of project's directory, and then rebuild)
```

In addition, if other compilation exceptions are encountered, first check that the compilation system used is not the system type of ext2/ext4.

14.5.2 Failed to Download "Base Debian"

- Since building Base Debian needs to visit foreign websites, and when using domestic networks to visit foreign websites, download failures often occur:

To use live build in Debian, configured like followings to change the image source to domestic:

```
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 is a pre-built package to share in [Baidu Cloud Network Disk](#), place it in the current directory and execute the next step directly .

14.5.3 Abnormal Operation Causes an error to Mount /dev

For example, like "askpass command or cannot use one" appears

It may be frequent abnormal operations (CTRL+C) during the compilation process, and the above errors can be fixed by the following way:

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

14.5.4 Multiple Mounts lead to /dev error

For example: sudo: unable to allocate pty: No such device appears

The reason may be that the compilation process has been mounted multiple times, resulting in the above error, which can be fixed by the following way:

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

14.5.5 How to Check System Related Information

14.5.5.1 How to Check the Debian Version of Your System

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

14.5.5.2 How to Check Whether the Debian Display Uses X11 or Wayland

On X11 systems:

```
$ echo $XDG_SESSION_TYPE
x11
```

On X11 systems:

```
$ echo $XDG_SESSION_TYPE
wayland
```

14.5.5.3 How to Check 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	

14.5.5.4 The ssh.service Is Abnormal in System

This is a problem of Debian10 or earlier version, please add the following code in the /etc/rc.local:

```
#!/bin/sh -e
#
## Chapter-14 rc.local
#
## Chapter-14 This script is executed at the end of each multiuser runlevel.
## Chapter-14 Make sure that the script will "exit 0" on success or any other
## Chapter-14 value on error.
#
## Chapter-14 In order to enable or disable this script just change the execution
## Chapter-14 bits.
#
## Chapter-14 By default this script does nothing.
## Chapter-14 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
```

14.5.6 Debian11 Base Package Fails to Build

An error similar to the following will be encountered:

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

It is mainly required that the kernel version of the PC should be 5.10+, which is a bug in the old QEMU. There are two solutions:

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

The way to check PC's Kernel Version:

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

- Update system's qemu

Please refer to [qemu](#).

14.5.7 How to Decompress, Modify and Repackage Debian deb Package

If you want to modify and repackage on the original deb, please refer to the follow way:

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

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

#Modify the file XXX

## Chapter-14 Repackage the modified content to generate a deb package
dpkg-deb -b extract/ .
```

14.5.8 How to Add the Swap Partition in Debian

When the physical memory of the system is not enough, you can add Debian's swap virtual memory partition for the current running program. 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-14 count represents the size, here is 2G.
```

- Convert files to swap files

```
sudo mkswap swapfile
```

- Activate the 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 is in 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

```

14.5.9 Update Debian System for the First Time Will Restart the Display Service

In general, in order to be compatible with different chips, when Debian starts for the first time, /etc/init.d/rockchip.sh will install various differential packages according to the chip, 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 process this difference.

14.5.10 Errors Occurring in Debian When Calling libGL related dri.so

Introduction as follows:

- EGL is an extension of OpenGL on the ARM platform for the x window system, and 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 which detecting through glx environment fail to start), libglx.so will search for the dri library in the system. However, Xorg 2D acceleration is implemented directly based on DRM and does not implement the dri library, so libglx.so will report the following error during booting.

```

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

```

It has no influence on system operation, please ignore it.

Similarly, the following errors will also be reported during the booting process of some applications, please ignore it for it has not influence on applications operation.

```

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

```

14.5.11 How to Confirm that the Hardware Mouse Layer is Useful in Debian

- Configure kernel dts

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

```

- modetest test whether the layer has been 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 if the summary has called the hardware mouse layer

```

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

```

If there are steps 1/2, and there are still problems, then check whether /var/log/drm-cursor.log has abnormalities.

14.5.12 The log is Too Large in Debian

Debian provides **logrotate** to manage log files. Logrotate is intended to simplify log file management for systems that will generate many log files. Logrotate supports automatic rotation compression, deletion and sending log related emails. Logrotate can be run daily, weekly, monthly or when the log file size reaches a certain value. Typically, logrotate is run as a daily cron job.

```

apt install -fy logrotate cron

```

14.5.13 Debian Setting Multi User Mode Issue

When the system starts, the 'systemd' process will try to start '/lib/systemd/system/default.target' (usually the graphical interface system is a symbolic link to "graphical.target"). The status can be obtained through the following command:

```

systemctl get-default
graphical.target

```

Setting multi user mode (command line system):

```

systemctl set-default multi-user.target

```

After restarting, it was found that the interface was suspended at the logo and could not access the system. The normal startup sequence of the system is 'sysinit ->multi user ->graphic '. If it is set to 'multi user. target ', it means that the graphical interface has not been started. In this case, VT2 (terminal interaction needs to be enabled) is required, which means that the kernel needs to open the following two macros

```

CONFIG_FRAMEBUFFER_CONSOLE=y
CONFIG_VT=y

```

14.5.14 Debian Username and Password

The default username and password for the system are 'linaro' and 'linaro',

```
Root 'can log in without a password, and can be accessed through the command line  
to' sudo su '
```

14.5.15 Debian XFCE Desktop icon Double-click Abnormal

XFCE desktop native bug can be resolved by checking 'Settings ->Desktop ->Icon' and clicking on 'Activate Project'.

14.5.16 Chromium browsers will have a command line flag: --no-sandbox

Unless using non-hardware accelerated, official native browser version. Otherwise, the customized browser version needs to be started with the `-no-sandbox` parameter, because the sandbox is a permission management and controls file access, and only without a sandbox can access to the hardware node is allowed to achieve hardware acceleration.

14.5.17 Setting Up DRI2 Extension in Debian System's X11

In the SDK, `glmark2-es2` uses the `dri2` interface for display with the Mali GPU library.

For specific implementation, refer to the relevant code in `xserver`:

- File Path: `./hw/xfree86/drivers/modesetting/dri2.c`
- Functions: `ms_dri2_get_msc` and `ms_dri2_schedule_wait_msc`

To confirm if Xserver supports DRI2, check the following log:

```
root@linaro-alip:/# grep -i dri2 /var/log/Xorg.0.log  
[ 47.696] (II) modeset(0): [DRI2] Setup complete  
[ 47.699] (II) modeset(0): [DRI2] DRI driver: rockchip  
[ 47.712] (II) modeset(0): [DRI2] VDPAU driver: rockchip  
[ 48.502] (II) Initializing extension DRI2
```

Below is a segment of the test code named `dri2-test.c` for DRI2.

```
#include <stdlib.h>  
#include <stdio.h>  
#include <xcb/xcb.h>  
#include <xcb/dri2.h>  
#include <X11/Xlib.h>  
#include <X11/Xlib-xcb.h>  
  
int main(void)  
{  
    xcb_connection_t *c;  
    xcb_dri2_connect_cookie_t cookie;  
    xcb_dri2_connect_reply_t *reply;  
    Display *display = XOpenDisplay(NULL);  
    Window window = DefaultRootWindow(display);
```

```

    c = XGetXCBConnection(display);
    cookie = xcb_dri2_connect(c, window, XCB_DRI2_DRIVER_TYPE_DRI);
    reply = xcb_dri2_connect_reply(c, cookie, 0);
    printf("%s[%d] device(%s)\n", __func__, __LINE__,
xcb_dri2_connect_device_name (reply));
    c = xcb_connect(NULL, NULL);
    xcb_screen_t *screen = xcb_setup_roots_iterator(xcb_get_setup(c)).data;
    cookie = xcb_dri2_connect(c, screen->root, XCB_DRI2_DRIVER_TYPE_DRI);
    reply = xcb_dri2_connect_reply(c, cookie, 0);
    printf("%s[%d] device(%s)\n", __func__, __LINE__,
xcb_dri2_connect_device_name (reply));
    return 0;
}

```

build and test:

```

## Chapter-14 gcc dri2-test.c -lxcb -lxcb-dri2 -lX11 -lX11-xcb -o dri2-test
## Chapter-14 ./dri2-test
main[21] device(/dev/dri/card0)
main[27] device(/dev/dri/card0)

```

14.5.18 Installing GCC Toolchain on Debian

To install, run the following command:

```
apt update && apt install -y build-essential manpages-dev
```

After installation, confirm the GCC version:

```

root@linaro-alip:/# gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/aarch64-linux-gnu/10/lto-wrapper
Target: aarch64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Debian 10.2.1-6' --with-
bugurl=file:///usr/share/doc/gcc-10/README.Bugs --enable-
languages=c,ada,c++,go,d,fortranx
Thread model: posix
Supported LTO compression algorithms: zlib zstd
gcc version 10.2.1 20210110 (Debian 10.2.1-6)

```

14.5.19 Auto-Completion for Installing Packages on Debian

This is a common operation in Linux systems. Normally, you need to install the `bash-completion` package.

For specific instructions, search online, but it generally involves:

```

sudo apt-get install bash-completion
source /etc/bash_completion

```

14.5.20 Supporting DRI3 Extension in Debian X11 System

DRI3 extension, part of X11, provides improved support for direct rendering. It's usually enabled by default in Debian. As a low-level protocol, it's not directly executed without external calls.

To use the DRI3 extension in applications, refer to DRI3 documentation and write code to call the respective interfaces. For example, use DRI3 interfaces provided by the XCB (X protocol C-language Binding) library. Here's how to install and view these interfaces in Debian:

- **Installing XCB DRI3 Development Libraries**

Install the `libxcb-dri3-dev` library with:

```
sudo apt-get install libxcb-dri3-dev
```

This installs the necessary libraries and headers for using DRI3 functions in your applications.

- **Viewing DRI3 Interfaces**

To view the definitions and functions of DRI3 interfaces, check the `dri3.h` header file:

```
vi /usr/include/xcb/dri3.h
```

This file contains specific implementations of the DRI3 interface, helping you understand how to use the DRI3 extension in your applications.

Follow these steps to start using the DRI3 extension in Debian for developing or improving your graphics applications. Remember to refer to official XCB and DRI3 documentation for correct and efficient use of these interfaces.

14.5.21 Setting System to Boot into Command Line Mode

The general method involves:

- Uninstalling xserver
- Enabling `CONFIG_FRAMEBUFFER_CONSOLE`, `CONFIG_DRM_FBDEV_EMULATION`, `CONFIG_VT`, `CONFIG_VT_CONSOLE` in the kernel

14.5.22 Configuring Screen Rotation

Refer to `/etc/X11/xorg.conf.d/20-modesetting.conf`

```
...
Section "Screen"
    Identifier "Default Screen"
    Device     "Rockchip Graphics"
    Monitor    "Default Monitor"
    DefaultDepth      24
    SubSection "Display"
        Depth      24
        Modes       "1024x600"
    EndSubSection
EndSection
```

```
#### Valid values for rotation are "normal", "left", "right"
Section "Monitor"
    Identifier   "HDMI-A-1"
    Option       "Rotate" "inverted"
    Option       "Position" "0x0"
EndSection
Section "Monitor"
    Identifier   "DSI-1"
    Option       "Rotate" "left"
    Option       "Position" "0x0"
EndSection
```

14.5.23 Implementing No Black Screen Function in Debian

Black screen refers to the time from X service startup to desktop application display (dependent on the desktop application itself).

To maintain a logo during this time, add the following before executing `Xorg.wrap` in `/usr/bin/X:`

```
export XSERVER_FREEZE_DISPLAY=./freeze_xserver
touch $XSERVER_FREEZE_DISPLAY
$(sleep 6; rm $XSERVER_FREEZE_DISPLAY) &
```

Freeze for 6 seconds, then display the desktop. Adjust the freeze duration as needed.

Alternatively:

```
{
    export XSERVER_FREEZE_DISPLAY=./freeze_xserver
    touch $XSERVER_FREEZE_DISPLAY
    while sleep .5; do
        pgrep panel && break # Waiting for status bar service
    done
    sleep 2 # Waiting for status bar rendering
    rm $XSERVER_FREEZE_DISPLAY
}&
```

14.5.24 Removing Desktop Mouse Pointer Display in Debian System

The native mechanism is designed for compatibility with some non-touch applications, where Xserver converts the first touch event into a mouse event.

To bypass, try:

- Using a transparent mouse theme (search online for specifics)
- Modify `hw/xfree86/drivers/modesetting/drmmode_display.c` source code, removing `drmModeSetCursor`, `drmModeMoveCursor` calls
- If the SDK includes a `drm-cursor` library, modify `/etc/drm-cursor.conf` and add `hide=1`

14.5.25 Steps for Compiling and Porting the rkaiq/rkisp Repository in Debian

To port the `rkaiq/rkisp` functionality from Buildroot to Debian:

14.5.25.1 Overview of Steps

For chips like RK3588, switch to a lower version of GCC and GLIBC for porting to a third-party system.

Step 1: Modify the Buildroot configuration to support GCC 8.

```
[Various changes to the configuration files, including switching from  
BR2_cortex_a76_a55 to BR2_cortex_a72_a53 and setting the GCC version to 8.x]
```

Step 2: Set the Buildroot configuration to use GCC 8 and GLIBC 2.28 by default.

```
buildroot# cat configs/rockchip_rk3588_glibc2.28_defconfig  
  
[Configuration for rockchip_rk3588 with GCC version 8.x and GLIBC 2.28]
```

Step 3: Compile with Buildroot using the configured settings.

Set up the Buildroot environment and compile the `camera-engine-rkaiq` module.

```
<SDK># source buildroot/envsetup.sh rockchip_rk3588_glibc2.28  
<SDK>## cd buildroot  
buildroot# make camera-engine-rkaiq
```

After compilation, port the generated files (e.g., `output/rockchip_rk3588_glibc2.28/build/camera-engine-rkaiq-1.0/camera-engine-rkaiq-1.0.tar`) to the Debian system.

14.5.26 How to Download Offline Deb Packages in Debian

```
root@linaro-alip:/# apt-get download <package name>
```

14.5.27 How to Check glibc Version in Debian

```
root@linaro-alip:/# ldd --version  
ldd (Debian GLIBC 2.31-13+deb11u7) 2.31  
or  
## Chapter-14 /lib/libc.so.6  
GNU C Library (GNU libc) stable release version 2.35.
```

14.5.28 Support for Screen Splitting in Debian Systems

For a physically split screen with a resolution of 3840x2160 cut into 3840x720, the output resolution must be 3840x2160 to light up the screen, but only the 720 part is displayed.

The latest xserver added padding support.

Setting method:

```
+++ b/overlay/etc/X11/xorg.conf.d/20-modesetting.conf
[Configuration changes to set VirtualSize and Padding for the display]
```

Modify `/etc/X11/xorg.conf.d/20-modesetting.conf` as above. `VirtualSize` is configured as the desired resolution, and `Padding` is configured for the pixels cut off in the order top, bottom, left, right.

For unclear text on the screen, modify the padding order.

```
Option "VirtualSize" "DSI-1:1920x316"
Option "Padding" "DSI-1:0,0,0,764"
```

Mouse movement is relative and usually doesn't require special modifications. If there are ratio or position issues, consider using a software cursor (remove the mouse layer in kernel dts configuration or configure SWcursor in the modesetting conf).

To remove the cursor in kernel dts, search and delete, like:

```
kernel/arch/arm64/boot/dts/rockchip# ag cursor
rk3566-evb2-lp4x-v10-linux.dts
12:      cursor-win-id = <ROCKCHIP_VOP2_CLUSTER0>;
```

Or configure SWcursor in modesetting conf:

```
--- a/overlay/etc/X11/xorg.conf.d/20-modesetting.conf
+++ b/overlay/etc/X11/xorg.conf.d/20-modesetting.conf
@@ -2,6 +2,8 @@ Section "Device"
     Identifier   "Rockchip Graphics"
     Driver       "modesetting"

+    Option       "SWcursor"          "TRUE"
+
```

14.6 Linux Video Related Issues

14.6.1 When playing videos, there is stuttering, and the logs show frame dropping errors

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
```


14.6.2 gst-launch-1.0 Camera Video Preview Command

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

14.6.3 The Playback Screen Jitters after Turning on AFBC

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).

14.6.4 How does GStreamer implement zero-copy

Use dmabuf related interfaces for data processing to achieve zero copy

14.6.5 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.

14.6.6 The Screen Jitters or Ripples Appear during Playback

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

14.6.7 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)

14.7 Third-party OS Porting Issues

14.7.1 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.

14.7.2 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.

14.7.3 Whether to Support UEFI Booting

RK3588 will give priority to supporting UEFI

14.8 Display Related Issues

14.8.1 Mouse Blinking Issue

"Refer to the SDK documentation on VOP configuration to set the primary and mouse layers to non-cluster layers."

Some chips support types of layers such as smart, esmart, and cluster, with the cluster layer being the default setting for the mouse layer, relying on some special processing in the SDK.

Some customer applications use methods that do not support cluster layers and require the use of other types of layers (such as smart, esmart) as the mouse layer, for example:

```
&vp0 {  
    cursor-win-id = <ROCKCHIP_VOP2_SMART0>;  
};
```

14.8.2 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.

14.8.3 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

docs/en/Linux/*/Rockchip_Developer_Guide_Buildroot_Weston_EN.pdf 2.9 Multi-screen configuration for details.

14.8.4 What is the Debian Xserver Version

Debian10 uses xserver1.20.4, Debian11 uses xserver1.20.11

14.9 Browser Related Issues

14.9.1 Chromium R114 Scaling Issue

Using the parameter `--force-device-scale-factor=1.5` will make the entire screen display smaller, resulting in an inability to display in full screen.

For the Chromium Wayland interface, this feature has only been adapted for Google's customized Aura Wayland protocol in newer versions, and there are bugs present under the standard Weston.

15. Chapter-15 SSH Public Key Operation Introduction

Please follow the introduction in the “Rockchip_User_Guide_SDK_Application_And_Synchronization_EN” 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.

15.1 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.

```
~/tmp$ git clone git@172.16.10.211:rk292x/mid/4.1.1_r1
Initialized empty Git repository in /home/cody/tmp/4.1.1_r1/.git/
The authenticity of host '172.16.10.211 (172.16.10.211)' can't be established.
RSA key fingerprint is fe:36:dd:30:bb:83:73:e1:0b:df:90:e2:73:e4:61:46.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.10.211' (RSA) to the list of known hosts.
git@172.16.10.211's password: █
```

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

```
~$ cd tmp/
~/tmp$ git clone git@172.16.10.211:rk292x/mid/4.1.1_r1
Initialized empty Git repository in /home/cody/tmp/4.1.1_r1/.git/
The authenticity of host '172.16.10.211 (172.16.10.211)' can't be established.
RSA key fingerprint is fe:36:dd:30:bb:83:73:e1:0b:df:90:e2:73:e4:61:46.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.10.211' (RSA) to the list of known hosts.
remote: Counting objects: 237923, done.
remote: Compressing objects: 100% (168382/168382), done.
Receiving objects: 9% (21570/237923), 61.52 MiB | 11.14 MiB/s
```

Adding ssh private key may result in the following error.

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

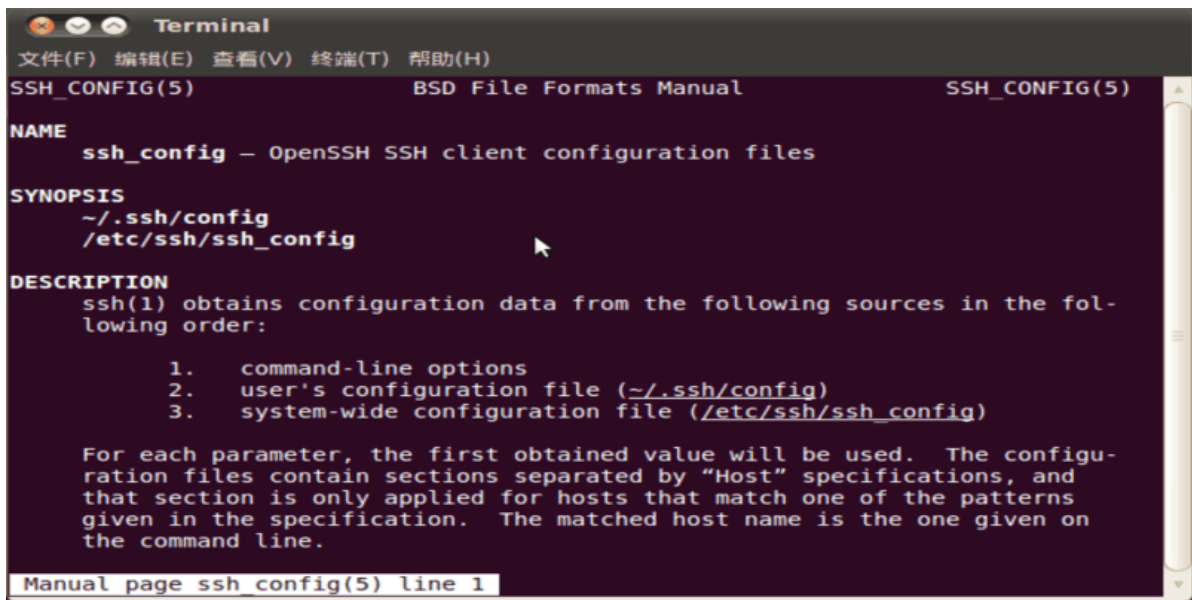
Enter the following command in console to solve:

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

15.2 Switches Different SSH Public Keys on One Device

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

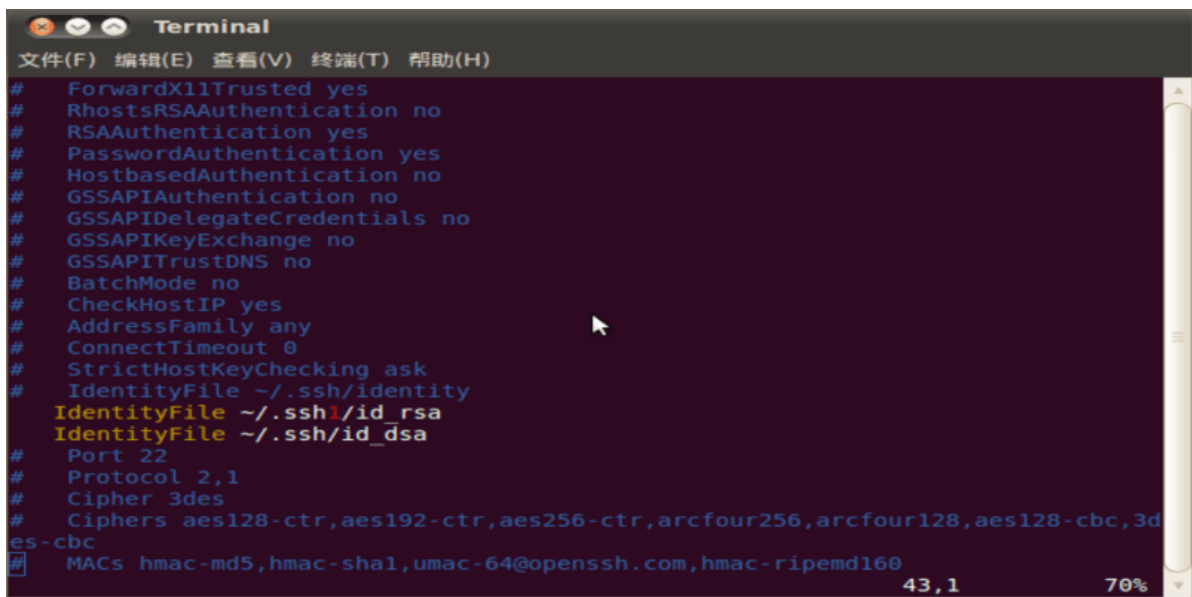
```
~$ man ssh_config
```



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.



15.3 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.

15.4 Reference Documents

For more details, please refer to document

`<SDK>/docs/en/Others/Rockchip_User_Guide_SDK_Application_And_Synchronization_EN.pdf`.