Rockchip Linux Software Development Guide

File Ref: RK-KF-YF-902

Release version: V1.6.0

Date: 2022-11-20

File Classification:□ Top Secret□

Secret□ Internal■ Public

Rockchip Electronics Co.

Address: No.18, Zone A, Software Park, Tongpan Road, Fuzhou, Fujian, China

Web site: www.rock-chips.com

Guest Services: +86-4007-700-590

Customer Service Fax: +86-591-83951833

Customer Service Email: fae@rock-chips.com

ahead言

summarize

This file serves as a software development guide for Rockchip Buildroot/Debian/Yocto Linux systems, and is designed to help software developers and technical supporters get up to speed with development and debugging on the Rockchip Linux platform.

Readership

This document (this guide) is mainly for the following practitioners: Technical Support Workers (TSPs)

Software Developer

Revision Record

日period	author	releases	Modification Instructions
2021-04-10	Caesar Wang	V1.0.0	initial version
2021-05-20	Caesar Wang	V1.1.0	Add support for rk3399, rk3288, rk3326/px30.
2021-09-30	Caesar Wang	V1.2.0	Updated support for Linux 4.4 and Linux 4.19.
2022-01-15	Caesar Wang	V1.3.0	Added support for RK3588 Linux 5.10 Added support for RK3358 Linux 4.19 Updated SDK version information
2022-04-14	Caesar Wang	V1.4.0	Added support for RK3326S Updated RK3588 Update FAQ
2022-05-20	Caesar Wang	V1.4.1	Updated core support and 2022 roadmap
2022-06-20	Caesar Wang	V1.4.2	Updated SDK version and support status
2022-09-20	Caesar Wang	V1.5.0	Updating Linux 5.10 to support it

2022-11-20	Caesar Wang	V1.6.0	Updates to all cores片 system support status and roadmap secure boot update instructions Update FAQ

Each core片 system support status

core name	**Buildroot** version	**Debian** version	**Yocto** version	**Kernel** version	**SDK** Version
RK3588	2021.11	11	4.0	5.10	V1.0.5
RK3566	2021.11	11	4.0	5.10	V1.0.2
RK3568	2021.11	11	4.0	5.10	V1.0.2
RK3399	2021.11	11	4.0	5.10	V1.0.2
RK3326/RK3326S	2021.11	N/A	N/A	5.10	V1.0.2
PX30/PX30S	2021.11	N/A	4.0	5.10	V1.0.2
RK3308	2021.11	N/A	N/A	5.10	V1.0.2
RK3566	2018.02-rc3	10	3.4	4.19	V1.3.1
RK3568	2018.02-rc3	10	3.4	4.19	V1.3.1
RK3399	2018.02-rc3	10	3.4	4.19	V1.2.1
RK3326/RK3326S	2018.02-rc3	N/A	N/A	4.19	V1.2.1
RK3358	2018.02-rc3	N/A	N/A	4.19	V1.1.1
RK3288	2018.02-rc3	10	3.4	4.19	V1.2.1
PX30/PX30S	2018.02-rc3	10	3.4	4.19	V1.2.1
RK3399PRO	2018.02-rc3	10	3.2	4.4	V1.4.2
RK1808	2018.02-rc3	N/A	N/A	4.4	V1.1.7
RK3399	2018.02-rc3	10	3.4	4.4	V2.8.0
RK3326/RK3326S	2018.02-rc3	N/A	N/A	4.4	V1.8.0
RK3328	2018.02-rc3	N/A	N/A	4.4	V1.1.0
RK3288	2018.02-rc3	10	3.4	4.4	V2.6.0
PX30/PX30S	2018.02-rc3	10	3.4	4.4	V1.8.0
RK3308	2018.02-rc3	N/A	N/A	4.4	V1.5.2
RK3358	2018.02-rc3	N/A	N/A	4.4	V1.8.0
RK3126C/RK3128	2018.02-rc3	N/A	N/A	4.4	V1.3.0
PX3SE	2018.02-rc3	N/A	N/A	4.4	V1.0.0

2022 Roadmap

Name of the core	**Buildroot** version	**Debian** version	**Yocto** version	**Kernel** version	Expected release date
RK3358	2021.11	N/A	N/A	5.10	2022.Q4
RK3288	2021.11	11	4.0	5.10	2022.Q4
RK312X	2021.11	N/A	N/A	5.10	2022.Q4
RK3036G	2021.11	N/A	N/A	5.10	2023.Q1
RK3399Pro	2021.11	11	4.0	5.10	2023.Q1
RK3328	2021.11	N/A	N/A	5.10	2023.Q1

目diary

Heterodyne mode,比 e.g. left/right or up/down position.

13.6.3 What is the Debian xserver version?

1. Rockchip Linux SDK Packages

1.1 brief

The Rockchip Linux SDK supports Buildroot, Yocto, and Debian, and is based on Kernel 4.4, Kernel 4.19, or Kernel 5.10, with boot based on U-boot v2017.09. It is suitable for all Linux products that utilize the Rockchip EVB board and a two-unit development kit based on this board. All Linux products based on the Rockchip EVB development board and a second development based on this board.

VPU hard decoding, GPU 3D, Wayland\X11 display etc. are supported so far. For detailed debugging and interface descriptions, please read the tutorial docs/file below.

1.2 How to Obtain Rockchip Linux Common Packages

1.2.1 Download via code server

To obtain Rockchip Linux packages, an account is required to access the source code repository provided by Rockchip. Customers can request an SDK from the Rockchip Technical Window, provide the SSH public key for server authentication authorization, and obtain the authorization to synchronize the code. For more information about SSH public key authorization for Rexchip's code server, please refer to Section 15, SSH Public Key Operation.

The Rockchip Linux SDK download command is as follows:

lamp pith片	releases	download command
RK3588	Linux 5.10	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	Linux 5.10	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	Linux 5.10	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests - b linux -m \ rk3399_linux5.10_release.xml
rk3326, rk3326s	Linux 5.10	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
PX30, PX30S	Linux 5.10	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests - b linux -m \ px30_linux5.10_release.xml
RK3308	Linux 5.10	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

RK3566, RK3568	Linux 4.19	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests - b linux -m \ rk356x_linux_release.xml
RK3399	Linux 4.19	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests - b linux -m \ rk3399_linux4.19_release.xml

lamp pith片	releases	download command
RK3326	Linux 4.19	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests - b linux -m \ rk3326_linux4.19_release.xml
RK3358	Linux 4.19	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests - b linux -m \ rk3358_linux4.19_release.xml
RK3288	Linux 4.19	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests - b linux -m \ rk3288_linux4.19_release.xml
PX30	Linux 4.19	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests - b linux -m \ px30_linux4.19_release.xml

RK3399	Linux 4.4	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \\ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/ manifests - b linux -m \\ rk3399_linux_release.xml
RK3358	Linux 4.4	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \\ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/ manifests - b linux -m \\ rk3358_linux_release.xml
RK3326	Linux 4.4	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \\ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/ manifests - b linux -m \\ rk3326_linux_release.xml
RK3288	Linux 4.4	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \\ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/ manifests - b linux -m \\ rk3288_linux_release.xml
lamp pith片	releases	download command
PX30	Linux 4.4	repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u \\ ssh://git@www.rockchip.com.cn/linux/rockchip/platform/ manifests - b linux -m \\ px30_linux_release.xml

The repo is a git script written by Google用 Python Script that is used to download and manage project repositories, and can be downloaded at the following address:

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

1.2.2 Obtained by decompressing a local zip archive

To facilitate quick access to the SDK source code, Rexchip Micro's technical window usually provides the initial zip package of the corresponding version of the SDK. In this way, developers can obtain the initial zip package of the SDK code, which is unzipped and synchronized with the source code downloaded through the repo.
Take RK356X_LINUX_SDK_RELEASE_V1.3.0_20220620.tgz as an example, after copying the initialization package, you can check the source code by the following command:

```
mkdir rk356x
tar xvf RK356X_LINUX_SDK_RELEASE_V1.3.0_20220620.tgz -C rk356x
cd rk356x
.repo/repo/repo sync -l
.repo/repo/repo sync -c
```

Subsequent developers can synchronize updates according to the update instructions periodically published in the FAE window via the `.repo/repo/repo sync -c` command to synchronize the update.
The initial zip package of the SDK released by目前Linux is as follows:

Name of the core	compressed package	releases
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
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
RK3588	RK3588_LINUX_SDK_RELEASE_V1.0.0_20220520.tgz	v1.0.0
RK3566, RK3568	RK356X_LINUX_SDK_RELEASE_V1.3.0_20220620.tgz	v1.3.0
RK3399	RK3399_LINUX4.19_SDK_RELEASE_V1.2.0_2022062020tgz	v1.2.0
RK3326	PK3326_LINUX4.19_SDK_RELEASE_V1.2.0_20220620.tgz	v1.2.0
RK3358	PK3358_LINUX4.19_SDK_RELEASE_V1.1.0_20220620.tgz	v1.1.0
PX30	PX30_LINUX4.19_SDK_RELEASE_V1.2.0_20220620.tgz	v1.2.0
RK3288	RK3288_LINUX4.19_SDK_RELEASE_V1.2.0_20220620.tgz	v1.2.0
RK3399	RK3399_LINUX_SDK_RELEASE_V2.9.0_20220620.tgz	v2.9.0
RK3326	RK3326_LINUX_SDK_RELEASE_V1.8.0_20220620.tgz	v1.9.0
RK3358	PK3358_LINUX_SDK_RELEASE_V1.8.0_20220620.tgz	v1.8.0
PX30	PX30_LINUX_SDK_RELEASE_V1.8.0_20220620.tgz	v1.8.0
RK3288	RK3288_LINUX_SDK_RELEASE_V2.6.0_202206200.tgz	v2.6.0

Note: The initial zip may be updated with a new version of the replacement!

1.3SDK Adaptation Hardware Full Auto-Compile Summary

Execute the following command at the root of the task to automatically complete all compilation:

```
. /build.sh all # Compile only module code (u-Boot, kernel,
Rootfs, Recovery)
                # To package the firmware, you need to run .
                /mkfirmware.sh to perform firmware packaging.
.               # In . /build.sh all
/build.s        # 1. add firmware package .
h               /mkfirmware.sh # 2.
                update.img packaging
                # 3. Copy the firmware from the rootdev directory to
                the IMAGE/***_RELEASE_TEST/IMAGES directory.
                # 4. Save patches for each module to the
                IMAGE/***_RELEASE_TEST/PATCHES directory
                # Note: . /build.sh is the same as . /build.sh
                allsave command.
```

The default is Buildroot, and rootfs can be specified by setting the bad state variable RK_ROOTFS_SYSTEM.目 can be preceded by three types: buildroot, debian, and yocto.

If debain is required, it can be performed using the following commands:

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

The hardware list Buildroot Full AutoCompile is summarized below:

lamp pith片	typology	releases	build	Board Configuration	Compile in One Click
RK3588	classifier for frequency of telegrams, phone calls用	Linux 5.10	ARM64	BoardConfig-rk3588- evb1-lp4-v10.mk	. /build.sh device/rockchip/rk3588/BoardConfig-rk3588-evb1-lp4-v10.mk && . /build.sh
RK3588	classifier for frequency of telegrams, phone calls用	Linux 5.10	ARM64	BoardConfig-rk3588- evb3-lp5-v10.mk	. /build.sh device/rockchip/rk3588/BoardConfig-rk3588-evb3-lp5-v10.mk && . /build.sh
RK3588	classifier for frequency of telegrams, phone calls用	Linux 5.10	ARM64	BoardConfig-rk3588s- evb1-lp4x-v10.mk	. /build.sh device/rockchip/rk3588/BoardConfig-rk3588s-evb1-lp4x-v10.mk && . /build.sh
RK3566	classifier for frequency of telegrams, phone calls用	Linux 4.19/Linux 5.10	ARM32	BoardConfig-rk3566- evb2-lp4x-v10-32bit.mk	. /build.sh device/rockchip/rk356x/BoardConfig-rk3566-evb2-lp4x-v10-32bit.mk && . /build.sh
RK3566	classifier for frequency of telegrams,	Linux 4.19/Linux 5.10	ARM64	BoardConfig-rk3566- evb2-lp4x-v10.mk	. /build.sh device/rockchip/rk356x/BoardConfig-rk3566-evb2-lp4x-v10.mk && . /build.sh

	phone calls用				
RK3568	classifier for frequency of telegrams, phone calls用	Linux 4.19/Linux 5.10	ARM32	BoardConfig-rk3568- evb1-ddr4-v10-32bit.mk	. /build.sh device/rockchip/rk356x/BoardConfig-rk3568-evb1-ddr4-v10-32bit.mk && . /build.sh
RK3568	classifier for frequency of telegrams, phone calls用	Linux 4.19/Linux 5.10	ARM64	BoardConfig-rk3568- evb1-ddr4-v10.mk	. /build.sh device/rockchip/rk356x/BoardConfig-rk3568-evb1-ddr4-v10.mk && . /build.sh
RK3568	classifier for frequency of telegrams, phone calls用 SPI Nor	Linux 4.19/Linux 5.10	ARM64	BoardConfig-rk3568- evb1-ddr4-v10-spi-nor-64M.mk	. /build.sh device/rockchip/rk356x/BoardConfig-rk3568-evb1-ddr4-v10-spi-nor-64M.mk && . /build.sh
RK3568	UVC	Linux 4.19/Linux 5.10	ARM64	BoardConfig-rk3568- uvc-evb1-ddr4-v10.mk	. /build.sh device/rockchip/rk356x/BoardConfig-rk3568-uvc-evb1-ddr4-v10.mk && . /build.sh
RK3326	sweeper	Linux 4.4/Linux 4.19/Linux 5.10	ARM64	BoardConfig-rk3326-robot64.mk	. /build.sh device/rockchip/rk3326/BoardConfig-rk3326-robot64.mk && . /build.sh
RK3326	sweeper	Linux 4.4/Linux 4.19/Linux 5.10	ARM64	BoardConfig-rk3326-robot64_no_gpu.mk	. /build.sh device/rockchip/rk3326/BoardConfig-rk3326-robot64_no_gpu.mk && . /build.sh
PX30	classifier for frequency of telegrams,	Linux 4.4/Linux 4.19/Linux 5.10	ARM32	BoardConfig-px30-evb- ddr3-v10-32bit.mk	. /build.sh device/rockchip/px30/BoardConfig-px30-evb-ddr3-v10-32bit.mk && . /build.sh

	phone calls用				
PX30	classifier for frequency of telegrams, phone calls用	Linux 4.4/Linux 4.19/Linux 5.10	ARM64	BoardConfig-px30-evb- ddr3-v10.mk	. /build.sh device/rockchip/px30/BoardConfig-px30-evb-ddr3-v10.mk && . /build.sh
PX30	classifier for frequency of telegrams, phone calls用	Linux 4.4/Linux 4.19/Linux 5.10	ARM32	BoardConfig-px30-evb- ddr3-v11-32bit.mk	. /build.sh device/rockchip/px30/BoardConfig-px30-evb-ddr3-v11-32bit.mk && . /build.sh
PX30	classifier for frequency of telegrams, phone calls用	Linux 4.4/Linux 4.19/Linux 5.10	ARM64	BoardConfig-px30-evb- ddr3-v11.mk	. /build.sh device/rockchip/px30/BoardConfig-px30-evb-ddr3-v11.mk && . /build.sh
PX30	sweeper	Linux 4.4/Linux 4.19/Linux 5.10	ARM64	BoardConfig-px30-robot64.mk	. /build.sh device/rockchip/px30/BoardConfig-px30-robot64.mk && . /build.sh
PX30	sweeper	Linux 4.4/Linux 4.19/Linux 5.10	ARM64	BoardConfig-px30-robot64_no_gpu.mk	. /build.sh device/rockchip/px30/BoardConfig-px30-robot64_no_gpu.mk && . /build.sh
RK3358	classifier for frequency of telegrams, phone calls用	Linux 4.4/Linux 4.19	ARM64	BoardConfig-rk3358- evb-ddr3-v10.mk	. /build.sh device/rockchip/rk3358/BoardConfig-rk3358-evb-ddr3-v10.mk && . /build.sh
RK3288	classifier for frequency of telegra	Linux 4.4/Linux 4.19	ARM32	BoardConfig-rk3288- evb-rk808.mk	. /build.sh device/rockchip/rk3288/BoardConfig-rk3288-evb-rk808.mk && . /build.sh

	ms, phone calls用				
RK3288	development board	Linux 4.4/Linux 4.19	ARM32	BoardConfig-rk3288-firefly.mk	. /build.sh device/rockchip/rk3288/BoardConfig-rk3288-firefly.mk && . /build.sh
RK3288	classifier for frequency of telegrams, phone calls用	Linux 4.4/Linux 4.19	ARM32	BoardConfig_rk3288- evb-act8846.mk	. /build.sh device/rockchip/rk3288/BoardConfig_rk3288-evb- act8846.mk && . /build.sh

2. File Description

The files released with the Rockchip Linux SDK are intended to help developers get up to speed quickly with development and debugging. The content covered in the files does not cover all development knowledge and issues. The list of files will be updated continuously. For questions and requests for files, please contact our FAE window fae @rock-chips.com.
The Linux SDK comes with Common (common development guide files), Socs (core platform related files), Linux (Linux system development guide), Others (other reference files), and docs_list.txt (docs file destination structure) under the docs directory.

2.1Common Development Guide File (Common)

For details, please refer to the files under each sub-project /docs/Common.

2.2Peripheral Support List (AVL)

The list of DDR/eMMC/NAND FLASH/WIFI-BT/CAMERA support is updated on redmine in real time, the link is as below:

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

2.2.1 DDR支 Holding List

Rockchip Platform DDR Particle Support List, under the /docs/Common/AVL directory Rockchip_Support_List_DDR_V2.50.pdf'', the DDR support levels shown in the table below are only recommended for particles labeled with√ and T/A.
Table 1-1 Rockchip DDR Support Symbol

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

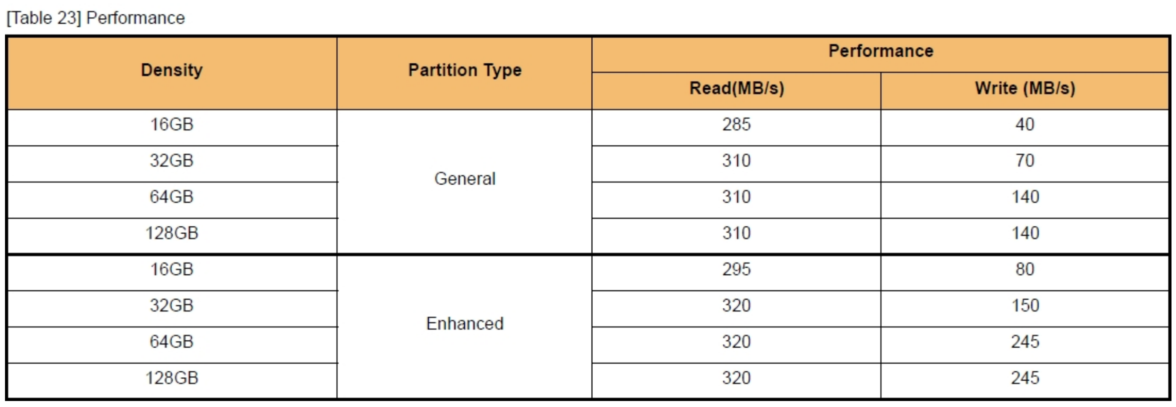
2.2.2 eMMC支 Holding List

Rockchip Platform eMMC Particle Support List, described under /docs/Common/AVL RKeMMCSupportList_Ver1.62_20211213.pdf", the EMMC support table shown below is only recommended to be selected from√ ,
T/A labeled particles.
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

2.2.2.1 Selection of high performance eMMC particles

High-performance eMMC particles are required for high system performance. Before selecting an eMMC pellet, refer to the model number in the Rockchip support list, with a focus on the performance section of the vendor Datashet. Selection is based on the vendor's size and the eMMC particle read/write rate. It is recommended to select a sequential read rate >200MB/s, a sequential write rate >200MB/s, and a sequential write rate >200MB/s.
>40MB/s.
For selection questions, you can also contact the Rockchip FAE window directly atfae@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 Example of eMMC Performance

2.2.3 SPI Nor and SLC Nand支 Holding List

Rockchip Platform SPI Nor and SLC Nand Support List, under /docs/Common/AVL RK_SpiNor_and_SLC_Nand_SupportList_Ver1.34_20211018.pdf'', which also has SPI Nand models labeled in the file, is available for selection. The Nand support levels shown in the table below are recommended only for particles labeled with√ and T/A.

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

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

2.2.4 Nand Flash支 Holding List

The Rockchip platform Nand Flash support list is detailed in

"RKNandFlashSupportList Ver2.73_20180615.pdf" under the "/docs/Common/AVL" directory, and the Nand Flash model numbers are labeled in the file for selection. The Nand Flash support levels shown in the table below are recommended only for particles labeled with√ and T/A.

Table 1-4 Rockchip Nand Flash Support Symbol

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

2.2.5 WIFI/BT支 Holding List

Rockchip Platform WIFI/BT Support List, under the /docs/Common/AVL directory The Rockchip_Support_List_WiFi_and_BT_20190801_EN.pdf file lists the WIFI/BT cards tested on the current Rockchip platform, and it is recommended that you select the cards according to the models on the list. For debugging of other WIFI/BT chips, the corresponding kernel drivers should be provided by the original WIFI/BT chips.
The maturity level is described in the file:
Perfect> Very Good> Good> Preliminary> X
For selection questions, it is recommended to contact the Rockchip FAE window atfae@rock-chips.com .

2.2.6 Camera支 Holding List

Rockchip Platform Camera Support List, detailed under /docs/Common/AVL Rockchip_Camera_Module_AVL_v2.1.pdf", the file list is the list of Camera Modules tested on Rockchip platforms so far, and it is recommended to select the model according to the list.
Notes.
1. For the modules in this table, Rockchip purchases the modules directly from the module manufacturer to perform debugging, and the customer purchases the modules from the module manufacturer according to the module number (with the possibility of changing the FPC connecting wires), and Rockchip and the module manufacturer try to ensure that the results can be used directly without changing the structure or configuration of the modules;
2. Commissioning status: (Y: commissioning complete, N: not yet completed)
IQ: Effect debugging status
driver(Linux): Sensor driver adapted to Rockchip Linux system,
please refer to the description in the driver documentation for details; if you have any questions about the selection, it is recommended to contact Rockchip FAE window atfae@rock-chips.com .

2.3Dioxide Platform Related Files (Socs)

For details, refer to the files under ObjectName.

2.3.1 Hardware Development Guide for SDK Packages

The Rockchip platform will have corresponding hardware reference files distributed with the SDK package. The Hardware User's Guide introduces the basic features of the reference boards, the hardware interfaces, and how to use them. The guide is designed to help developers use the EVB faster and more accurately for product development, and can be found in the related files under the following link: /docs/card-name/ object.

2.3.2 Multimedia Codec Support List

Rockchip 片比 such as the RK3399/RK3399Pro support powerful multimedia features: 4K VP9 and 4K 10bits H265/H264 video decoding up to 60fps, 1080P multi-format video decoding (WMV, MPEG-1/2/4, VP8), 1080P video encoding with H.264, VP8, video post-processor: anti-interlacing, denoising, edge/detail/color optimization, and video post-processor: anti-interlacing, denoising, edge/detail/color optimization. VP8 format, video post-processor: anti-interlacing, denoising, edge/detail/color optimization.

Note: This is a snapshot of what is supported on the chip, and the actual support format and performance may vary with different systems.

2.4Linux System Development Files (Linux)

For details, refer to the files under "/docs/Linux".

2.5Other Reference Files (Others)

Other reference files,比 , such as Linux Software Test Guide, Rockchip SDK Application and Synchronization Guide, Rockchip Bug System User Guide, etc., are available in the files under /docs/Others.

2.6File Object Structure (docs_list.txt)

For details, to file /docs/docs_list.txt.

```
docs
├── Common
├── docs_list.txt
├── Linux
├── Others
├── Rockchip_Developer_Guide_Linux_Software_CN.pdf
├── Socs
```

3.Work instructions

The tools released with Rockchip Linux SDK are used in the development and debugging stage as well as the mass production stage. The version of the tools will be updated continuously with the SDK update, please contact our FAE window

atfae@rock-chips.com for any questions or requirements on the tools.
The Rockchip Linux SDK comes with tools linux, windows, and other operating systems under the tools directory.

(Windows operating system environment using tools).

- Windows work

Work instructions file: tools/windows/ToolsRelease.txt

Name of the workpiece	Use of tools
RKDevTool	Upgrading Firmware Computationally and the Entire Update Firmware Tool
FactoryTool	Mass Production Upgrade Tools
SecureBootTool	Firmware Signature Tool
efuseTool	efuse Burning Tool
RKDevInfoWriteTool	Marking Tools
SDDiskTool	SD card image creation
programmer_image_tool	Burner Upgrade Tool
pin_config_tool	IO Configuration Tool
DriverAssitant	Driver Installation Tools
RKImageMaker	Packing Tools (packed into updata.img)
SpeakerPCBATool	Sound Box PCBA Test Tools
RKDevTool_Release	Firmware Burn-in Tool
ParameterTool	Partition Table Modification Tool
CameraFactoryTestTool	Camera Module Tester

- Linux Tools

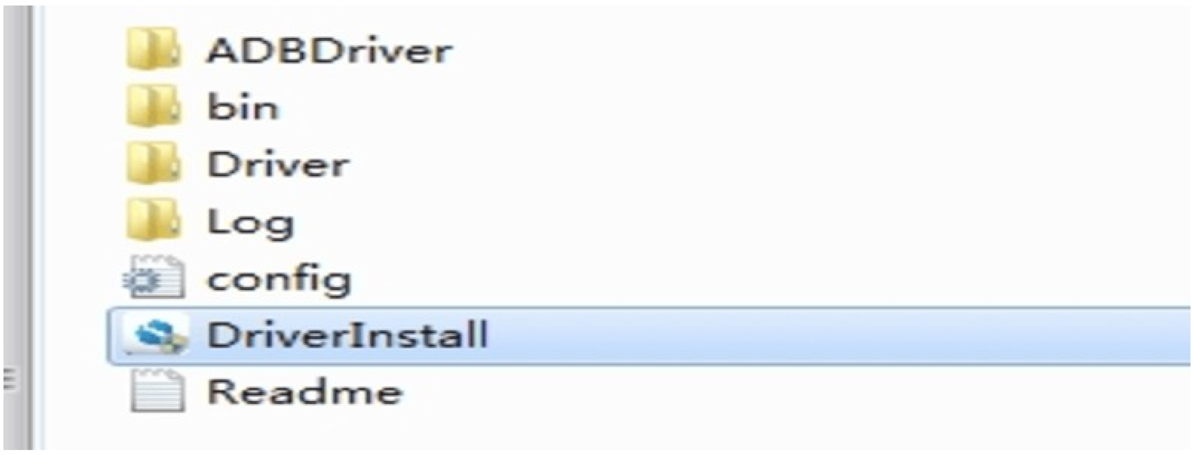
Work instructions file: tools/linux/ToolsRelease.txt

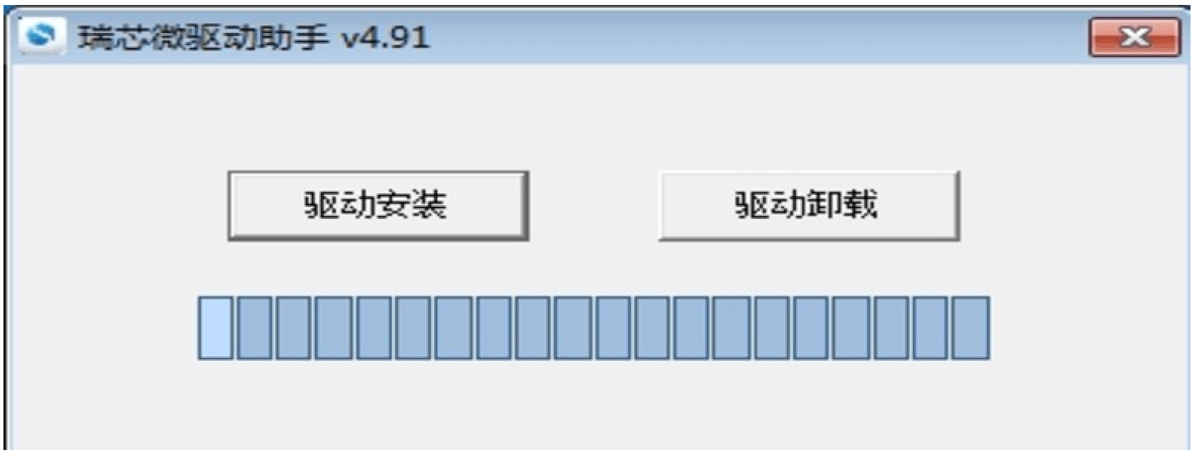
Name of the workpiece	Use of tools
Linux_Pack_Firmware	Firmware Packager (packaged as updata.img)
Linux_Upgrade_Tool	Firmware Burner
Linux_SecureBoot	Firmware Signature Tool
Linux_TA_Sign_Tool	loader (miniloader/trust/uboot) signing utility
Linux_SecurityAVB	boot/recovery signing utility
Linux_SecurityDM	rootfs signing utility
programmer_image_tool	Packaging SPI NOR/SPI NAND/SLC NAND/eMMC Burner Firmware

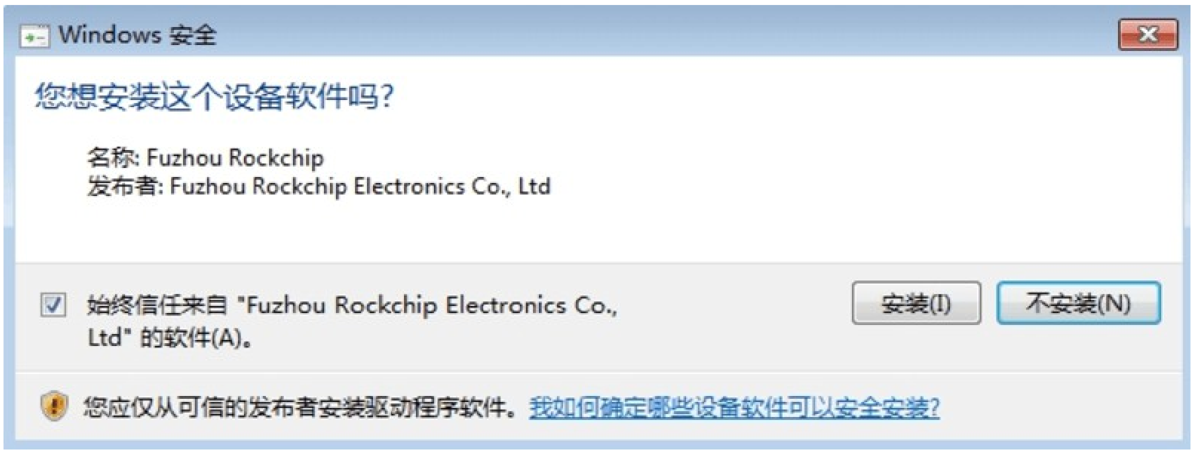
3.1Driver Installation Tools

The Rockchip USB driver installation helpers are stored in `<SDK>/tools/windows/DriverAssitant_<version>.zip`. Support xp,win7_32,win7_64,　　win10_32,win10_64 and other operating systems.

The installation steps are as follows:






Windows 安全

您想安装这个设备软件吗?

名称: Fuzhou Rockchip
发布者: Fuzhou Rockchip Electronics Co., Ltd

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

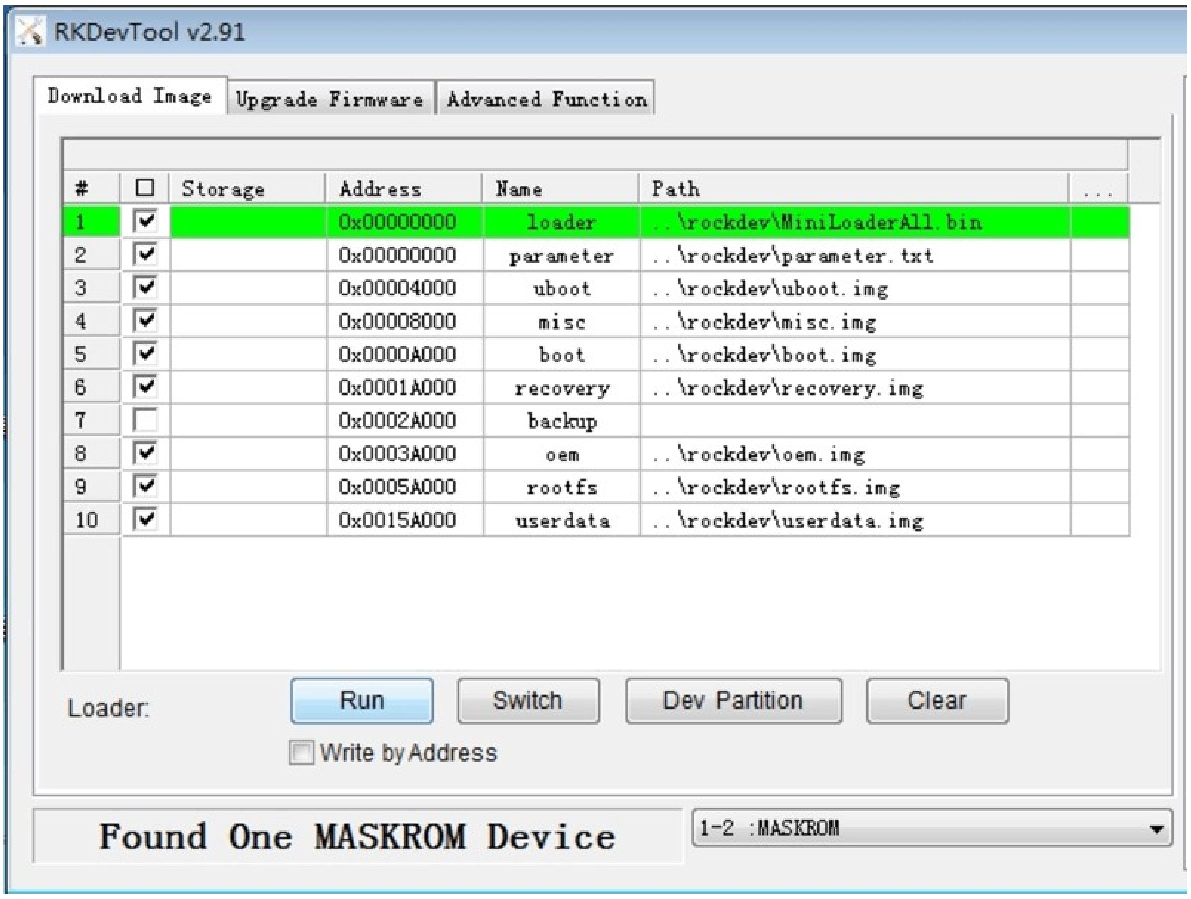
安装(I) 不安装(N)

您应仅从可信的发布者安装驱动程序软件。我如何确定哪些设备软件可以安全安装?

3.2Developing Burn-in Tools

- The SDK is supplied with a Windows burner (version V2.84 or higher), which is located at the root of the workbench:

```
<SDK>/Tools/windows/RKDevTool/
```


RKDevTool v2.91

Download Image | Upgrade Firmware | Advanced Function

#	☐	Storage	Address	Name	Path	...
1	✓		0x00000000	loader	..\rockdev\MiniLoaderAll.bin	
2	✓		0x00000000	parameter	..\rockdev\parameter.txt	
3	✓		0x00004000	uboot	..\rockdev\uboot.img	
4	✓		0x00008000	misc	..\rockdev\misc.img	
5	✓		0x0000A000	boot	..\rockdev\boot.img	
6	✓		0x0001A000	recovery	..\rockdev\recovery.img	
7			0x0002A000	backup		
8	✓		0x0003A000	oem	..\rockdev\oem.img	
9	✓		0x0005A000	rootfs	..\rockdev\rootfs.img	
10	✓		0x0015A000	userdata	..\rockdev\userdata.img	

Loader:

Write by Address

Found One MASKROM Device

1-2 :MASKROM

- The SDK provides a Linux burner (Linux_Upgrade_Tool tool version needs to be V2.17 or above), which is located at the root of the task:

```
<SDK>/Tools/linux/Linux_Upgrade_Tool/
```

```
Linux_Upgrade_Tool$ sudo .
/upgrade_tool -h
-----------------------          -----------------------
Help:            H  Tool Usage
Version:         V
Log.             L
--------------------               --------------------
                 G
ChooseDevic        Upgrade
e:              L  Command
                D    SD
ListDevice:          CD
UpgradeFirmwar       UF <Firmware> [-noreset]
SwitchDevic
e:
e:                   UL <Loader> [-noreset]
UpgradeLoader        [FLASH|EMMC|SPINOR|SPINAND] DI <-p|-b|-k|-
:                    s|-r|-m|-u|-t|-re image>
EraseFlash.     EF
DownloadImage        DB <Loader>
<Loader|firmware>
:
PartitionList.      PL
DownloadBoot.
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:     RI
ReadFlashInf     D
o:               RF
ReadChipInfo     I
:              R RC
ReadSecureMod  L I
e:             W RS
               EB <CS> <BeginBlock> <BlokcLen> [--Force]
WriteSector:   L M
               RUN <uboot_addr> <trust_addr> <boot_addr> <uboot>
ReadLBA:       E WS <BeginSec> <PageSizeK> <PageSpareB>
------------------<trust> <boot>------------------------
WriteLBA:      L <File>
EraseLBA:        <BeginSec> <SectorLen> [File]
EraseBlock.      <BeginSec> [SizeSec] <File>
RunSystem.       <BeginSec> <EraseCount>
```

3.3Baling Tools

It is mainly used to package the discrete firmware into a complete update.img firmware to facilitate upgrading.

- To package the update.img firmware for Windows, run the following command to create update.img

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

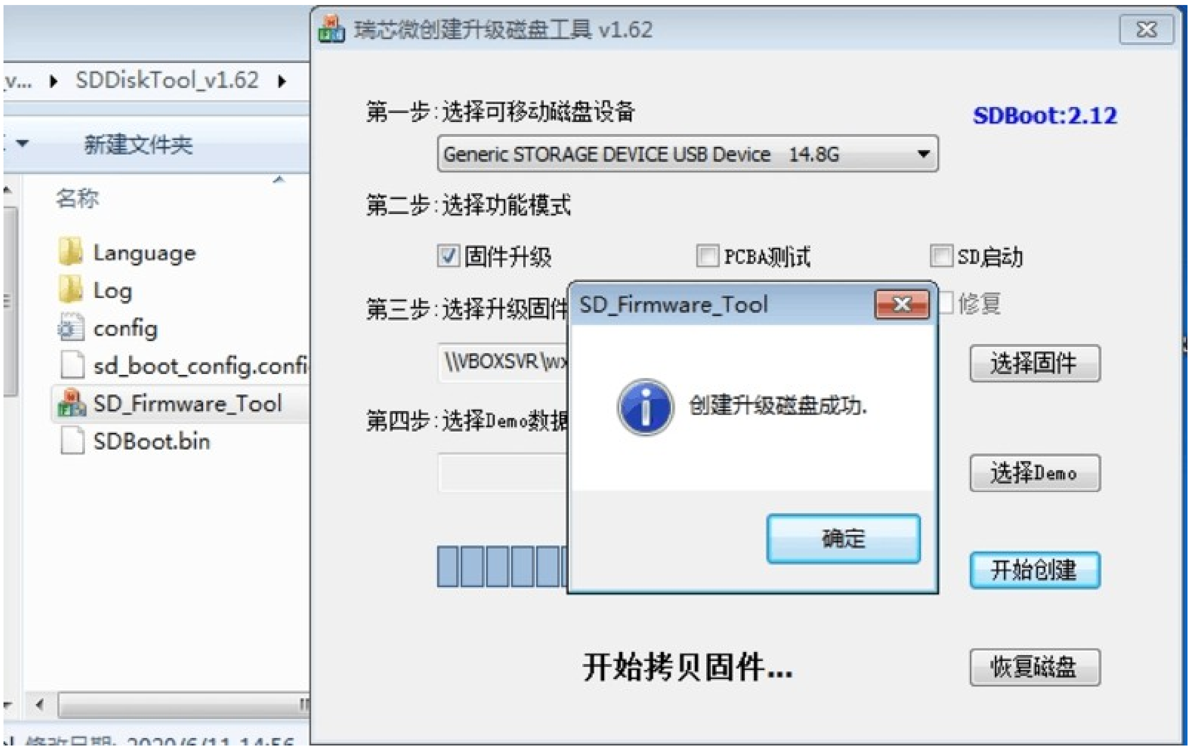
To package the update.img firmware in a Linux environment, run the following command to create update.img

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

3.4SD Upgrade Boot Utility

For SD card upgrade, SD card boot, and SD card PCBA testing.

```
<SDK>/tools/windows/SDDiskTool_v1.62.zip
```



瑞芯微创建升级磁盘工具 v1.62

SDBoot:2.12

第一步：选择可移动磁盘设备

Generic STORAGE DEVICE USB Device 14.8G

第二步：选择功能模式

☑ 固件升级 ☐ PCBA测试 ☐ SD启动 ☐ 修复

第三步：选择升级固件

\\VBOXSVR\wx

第四步：选择Demo数据

SD_Firmware_Tool

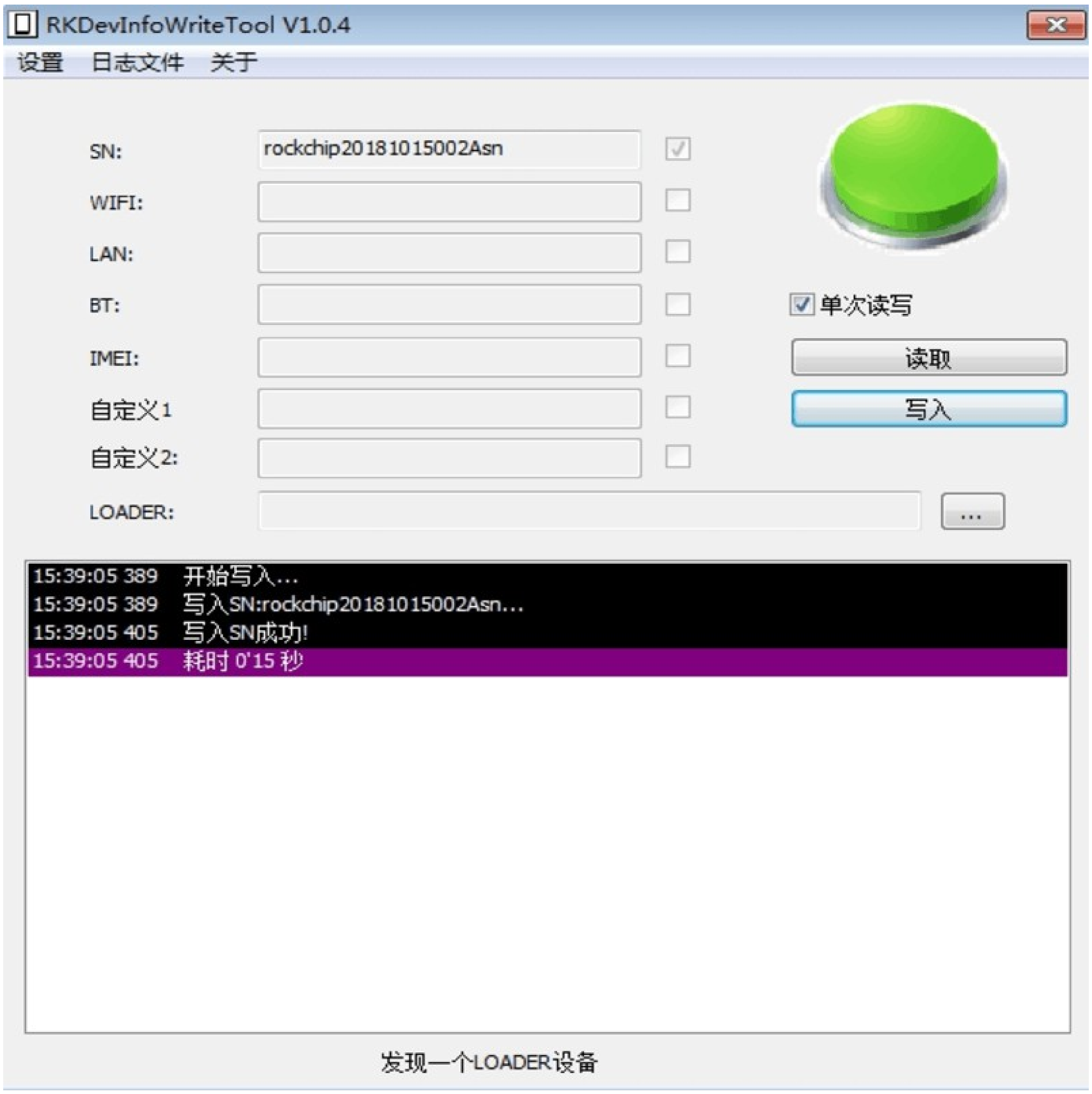
创建升级磁盘成功.

开始拷贝固件...

3.5Marking Tools

`<SDK>/tools/windows/RKDevInfoWriteTool*`

Unzip RKDevInfoWriteTool-1.3.0.7z and install it, open the software with administrator privileges, and refer to the current entry for work usage. `Rockchip_User_Guide_RKDevInfoWriteTool_CN.pdf` .



SN: rockchip20181015002Asn ✓
WIFI:
LAN:
BT:
IMEI:
自定义1
自定义2:
LOADER:

```
15:39:05 389   开始写入...
15:39:05 389   写入SN:rockchip20181015002Asn...
15:39:05 405   写入SN成功!
15:39:05 405   耗时 0'15 秒
```

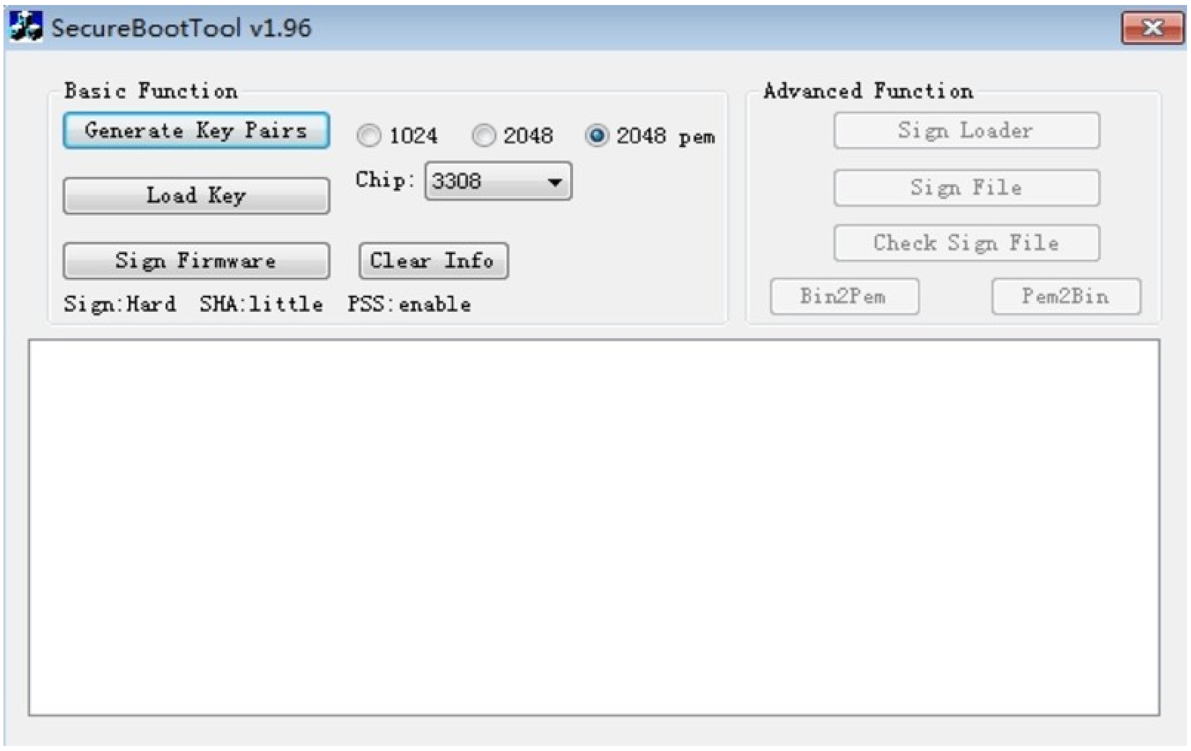
发现一个LOADER设备

3.6Firmware Signature Tool

Use efuse/otp signature for firmware.

- The SDK provides a Windows signing utility located at the root of the task:

```
<SDK>/tools/windows/SecureBootTool_v1.96
```



1024 2048 [✓ 2048 pem]
Chip: 3308
Sign:Hard SHA:little PSS:enable

- The SDK provides a Linux signing utility located at the root of the workbench:

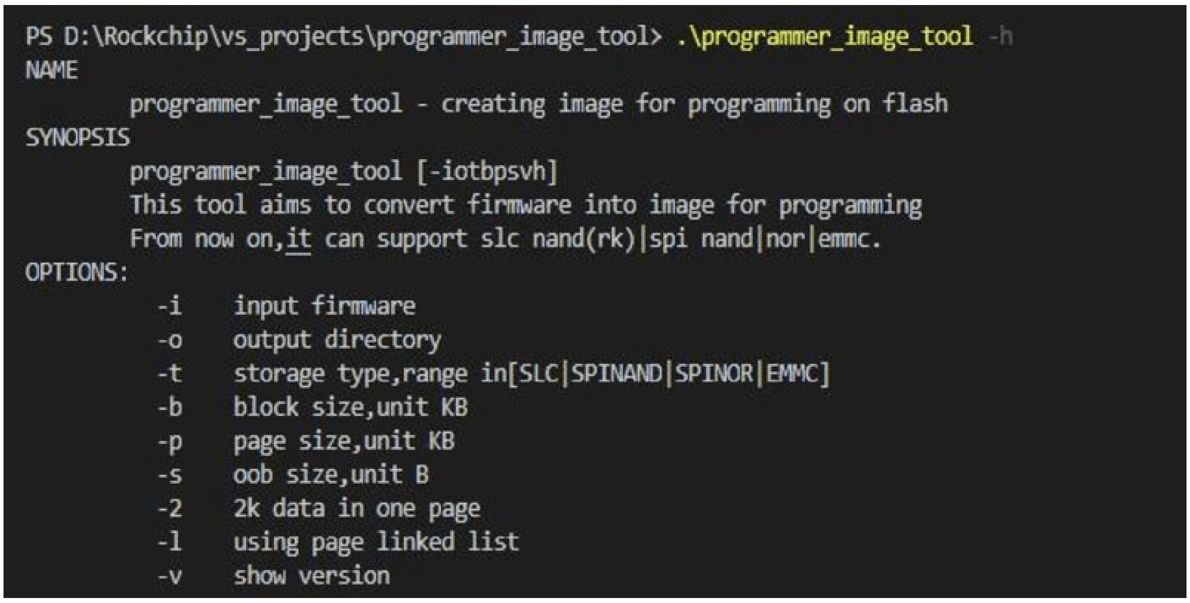
```
<SDK>/tools/linux/rk_sign_tool_v1.31_linux.zip

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******
```

For tool usage, please refer to the instructions for using the signature tool in the document /docs/Linux/Security/Rockchip_Developer_Guide_Linux4.4_SecureBoot_CN.pdf.

3.7Burner Upgrade Tool

A tool for mass-production burner image creation is located:

```
<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
```

Steps to make a burner image.

- Burning an image to emmc

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

- Burn image to spi nor

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

Refer to the manual at `user_manual`.pdf for additional usage instructions.

3.8PCBA Test Tools

PCBA testing tools are used to help quickly identify the functionality of products during mass production and to enhance production efficiency. Currently, test items include screen (LCD), wireless (Wi-Fi), bluetooth, DDR/EMMC memory, SD card, USB host, key (KEY), and speaker/headphone (Codec).

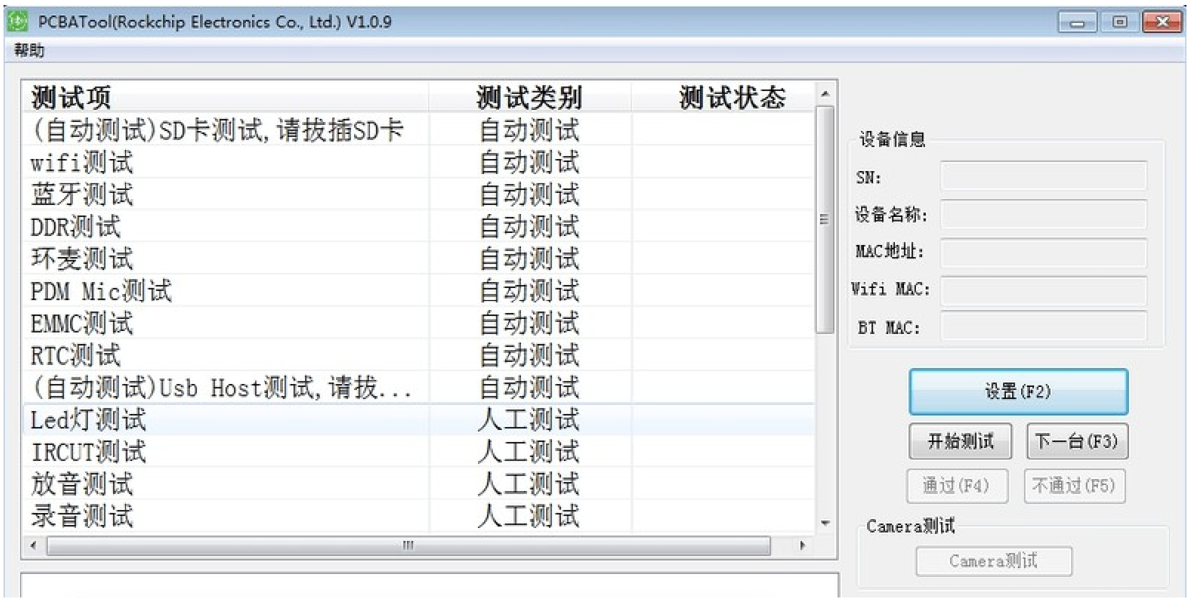
These test items include automatic test items and manual test items, wireless network, DDR/EMMC, Ethernet as automatic test items, and keys, SD Card, USB HOST, Codec, are manual test items.

PCBA tools are located:

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

For specific PCBA function configuration and usage instructions, please refer to.

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


PCBATool(Rockchip Electronics Co., Ltd.) V1.0.9

帮助

测试项	测试类别	测试状态
(自动测试)SD卡测试，请拔插SD卡	自动测试	
wifi测试	自动测试	
蓝牙测试	自动测试	
DDR测试	自动测试	
环麦测试	自动测试	
PDM Mic测试	自动测试	
EMMC测试	自动测试	
RTC测试	自动测试	
(自动测试)Usb Host测试，请拔...	自动测试	
Led灯测试	人工测试	
IRCUT测试	人工测试	
放音测试	人工测试	
录音测试	人工测试	

设备信息

SN:

设备名称:

MAC地址:

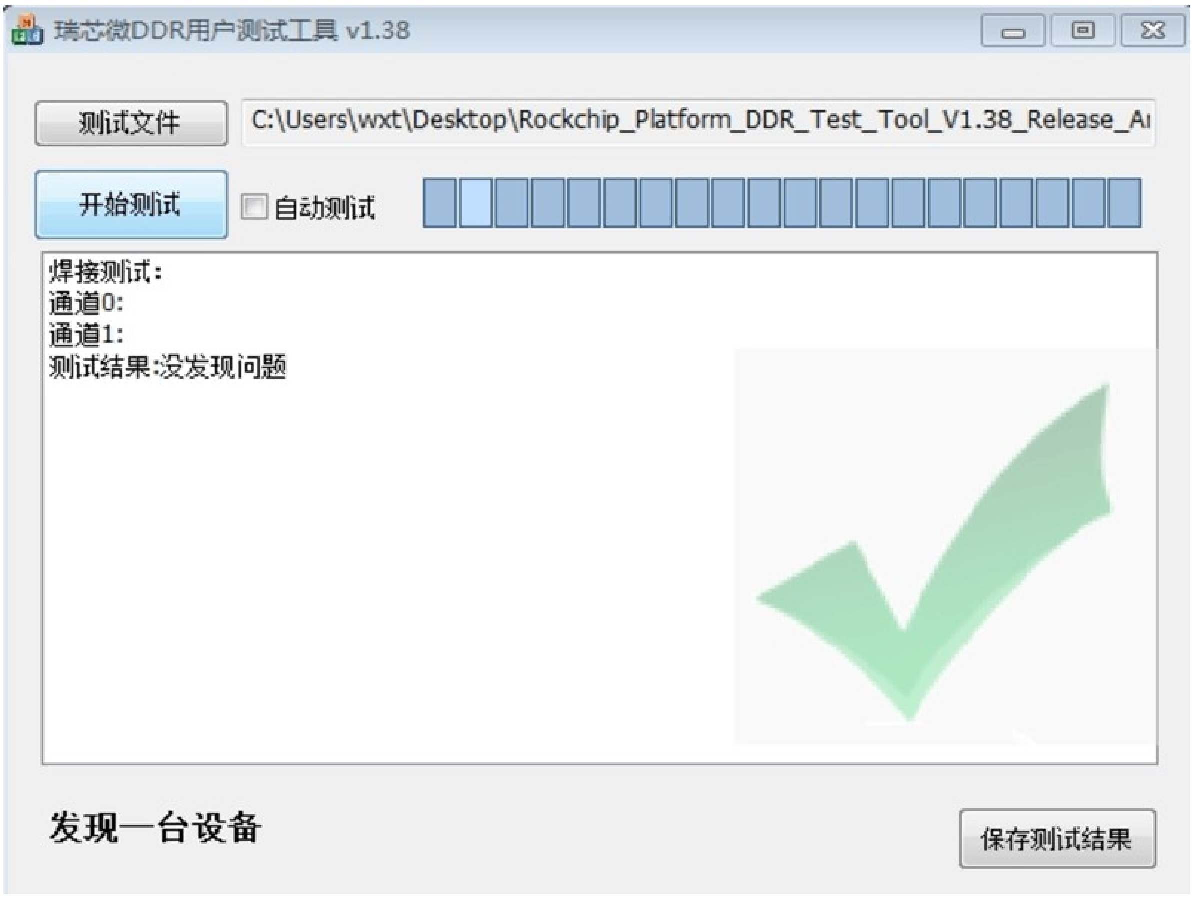
Wifi MAC:

BT MAC:

3.9DDR Soldering Test Tool

Used to test DDR hardware connections and troubleshoot hardware problems such as soldering:

```
<SDK>/tools/windows/Rockchip_Platform_DDR_Test_Tool_V1.38_Release_Ann
oucement_CN.    7z
<SDK>/tools/windows/Rockchip_Platform_DDR_Test_Tool_V1.38_Release_Ann
oucement_EN.    7z
```


瑞芯微DDR用户测试工具 v1.38

测试文件 C:\Users\wxt\Desktop\Rockchip_Platform_DDR_Test_Tool_V1.38_Release_A

```
焊接测试:
通道0:
通道1:
测试结果:没发现问题
```

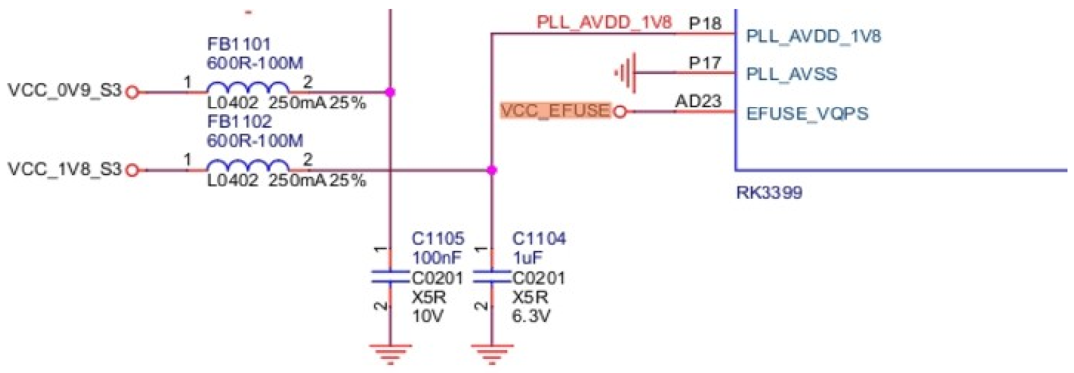
发现一台设备

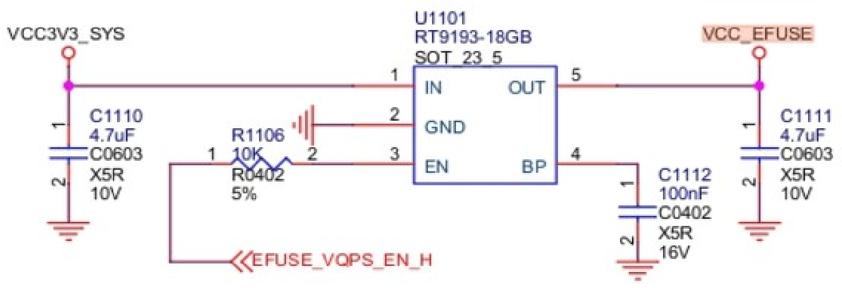
3.10 eFuse Burning Tool

The eFuse can be used to write to the RK3288/RK3368/RK3399/RK3399Pro platforms.

```
<SDK>/tools/windows/EfuseTool_v1.4.zip
```

If the SecureBoot function is enabled with eFuse, make sure that there is no problem with the hardware connection because the Kernel is not yet booted when eFuse is burned, so make sure that VCC_EFUSE is powered up in the MaskRom state before using it.





Use /Tools/windows/EfuseTool_v1.4.zip to put the board in MaskRom state.
Click "Firmware", select the signed update.img or Miniloader.bin, and click "Start" to begin burning the eFuse.

3.11 Mass Production Upgrade Tools

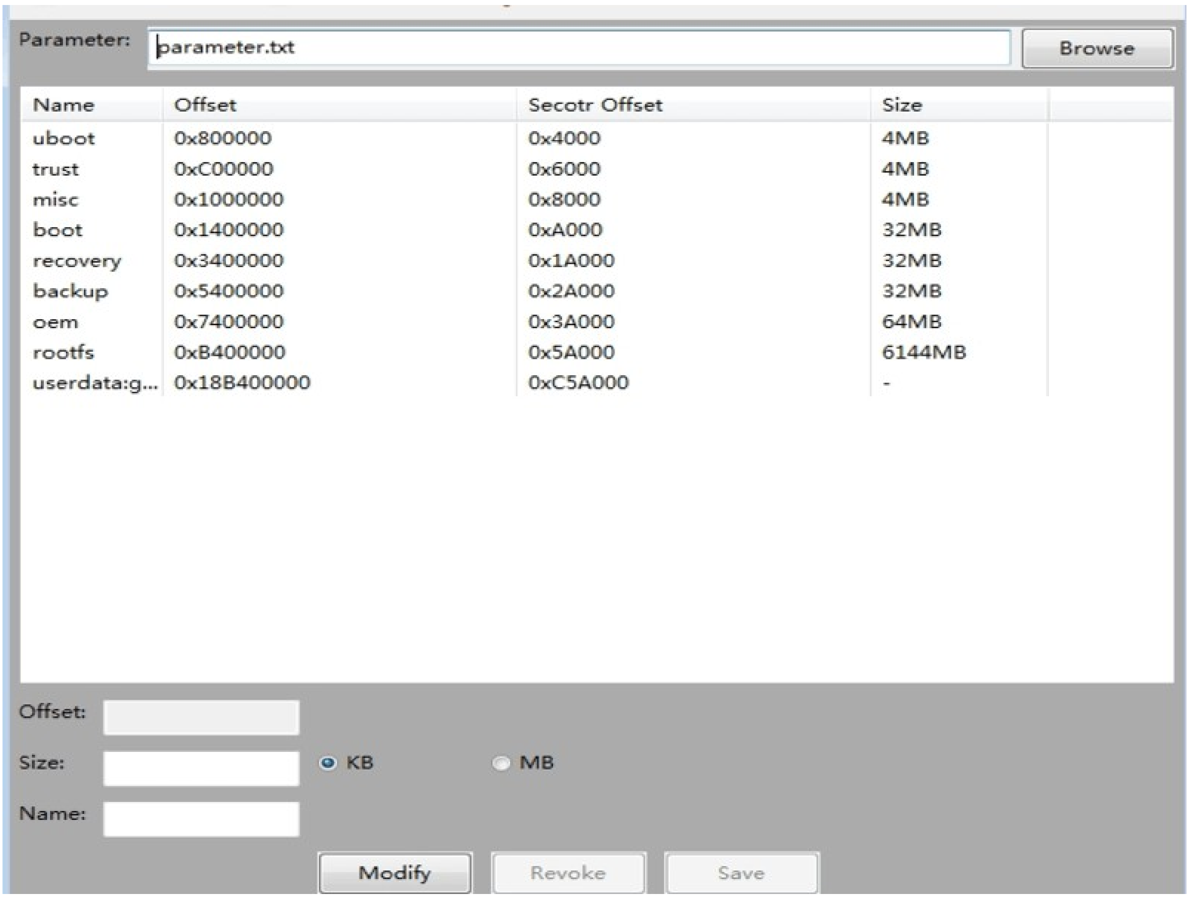
Use for batch burning of firmware for industrial use:

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

3.12 Partition Modifier

Use the partition modifier in Paramter.txt:

```
<SDK>/tools/windows/ParameterTool_v1.0.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:

3.13Camera Module Tester

The tool is used for camera module testing and is located in the same building as the camera module:

```
<SDK>/tools/windows/CameraFactoryTestTool-v2.0.5.1.zip
```

3.14EQTool Tools

EQ_DRC Tool (Equalizer & Dynamic Range Control Tool) is a voice equalizer and dynamic range planning tool.
--Use this tool to debug various audio parameters online for RK3308.
The tool is located:

```
<SDK>/tools/windows/eq_drc_tool-v1.23.zip
```

4.SDK Software Architecture

4.1Introduction to SDK Projects

A general-purpose Linux SDK workbook includes buildroot, debian, app, kernel, u-boot, device, docs, external, and so on.
The following are some of the characteristics of the RK3308/RV1108/RV1109/RV1126. Some of the feature cores片 such as RK3308/RV1108/RV1109/RV1126 will be different. Each project or sub-project will have a git task, and commits will need to be executed under the respective project.

- apps: Stores upper-level apps, mainly qcamera/qfm/qplayer/settings and other applications.
- buildroot: A root file system based on buildroot.
- debian: A debian-based root file system supporting some cores片 .
- device/rockchip: Stores board-level configuration and parameter files for each core, as well as scripts and preparatory files for compiling and packaging firmware.
- docs: Houses the片 module development guide files, platform support lists, platform-related files, and Linux development guides.
- IMAGE: Stores every生 compile time, XML, patch and firmware record.
- external: Storing third party related storage, including audio, video, web, recovery, etc.
- kernel: Code developed by kernel 4.4\4.19\5.10.
- prebuilts: Stores the cross-compilation toolchain.
- rkbin: store Rockchip related Binary and tools.
- rockdev: store compiled firmware.
- tools: Houses tools commonly used in Linux and Windows operating system environments.
- u-boot: Stores uboot code based on the v2017.09 version.
- yocto: A root file system based on yocto, supporting some cores片 .

4.2SDK Overview

4.2.1 Distribution

Distributions are the systems that we use Linux Host in Japan.

If you develop applications based on Distribution, things are relatively simple. The whole environment can be configured as long as the binaries are compiled and configured from the web source. 比For example, if you need weston or python, you don't need to be careful about what dependencies weston has, or what configurations you need to make for python compilation. 比For example, if you are developing a heavy web application with python or nodejs based backend, nginx, iptables, etc., and you want to build it based on Buildroot and Yocto, you will need to prepare a large team to do it.

The downside is also obvious: there is no way to customize it, which leads to redundancy and deficiencies. Redundancy is reflected in the larger size of比 ,

such as Xserver, the default Xserver will enable selinux, glx, and other features that are not available on the embedded system, and will take up more space than大 . For the same feature, Yocto may take up 500 Mb, while Debian needs 1 Gb. For the same functionality, Yocto may take up 500 Mb, while Debian requires 1 Gb. 比A disadvantage is that performance is not optimized. For example, QT does not have gles support the default version of QT, which uses gl by default, gles are used on the RIS, which means that when QT is run on the released version, the UI is done through the software's gl instead of the hardware's gles, which results in a performance loss.

4.2.2 Buildroot

The Rockchip Linux SDK is based on the Buildroot system, which contains all the source code, drivers, and other components needed for Linux-based development. 工

Buildroot is an open source autobuild framework for embedded Linux systems on the Linux platform. Buildroot consists of Makefile scripts and Kconfig configuration files. The Buildroot configuration allows you to build a complete Linux system that can be burned directly into a machine.

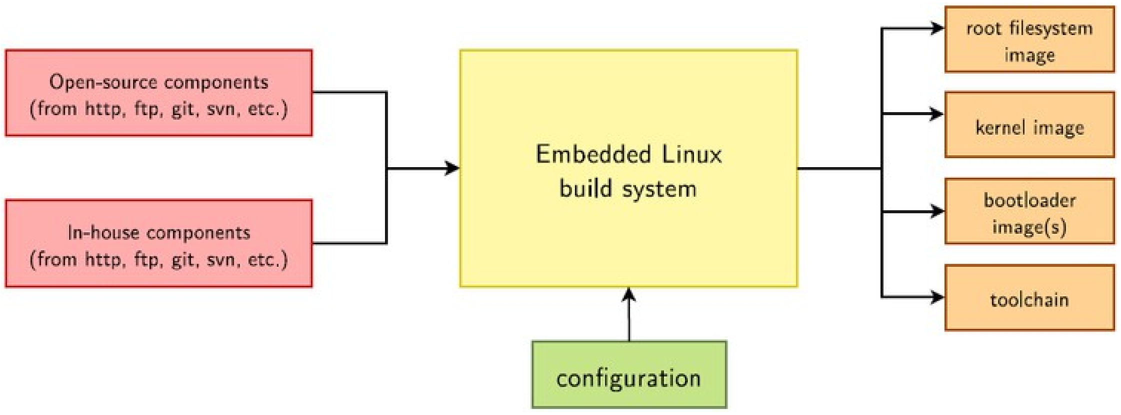

Open-source components (from http, ftp, git, svn, etc.)
In-house components (from http, ftp, git, svn, etc.)
Embedded Linux build system
configuration
root filesystem image
kernel image
bootloader image(s)
toolchain

Figure 4-1 Buildroot compilation block diagram

Buildroot has the following advantages:

- Built with great flexibility through source code;
- A convenient cross-compilation environment enables rapid building;• facilitates the configuration of system components and customized development.

The most famous project using Buildroot is Openwrt. As you can see, the image created by it can run on a router with 16 Mb SPI NOR, and the system is basically free of extras. This is due to the simplicity of the Buildroot location. The entire Buildroot project is maintained in a git.

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

Buildroot itself does not have the power to scale, and用戸 requires scripting to do the work itself. These listed features are the same as the Yocto Different方.

4.2.3 Yocto

Yocto, like Buildroot, is a tool for building reclosers, but they have completely different styles.
The Yocto project is maintained in individual packages (meta),比 , with some packages being responsible for the core and others for the periphery. 片Some packages are used to run Rockchip's core用 , some packages are used to install Qt, and some packages are used to run debian. The nodejs community, which also uses a similar mechanism, is very bloated and active, with people sharing their own low-quality garbage packages on github, which ensures that we can replicate the work of others over the Internet, which is quite valuable.

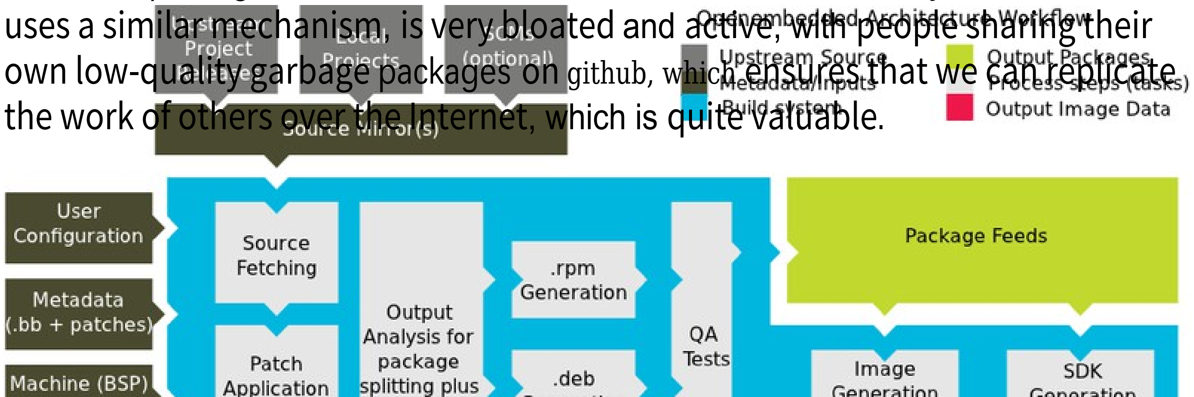


Figure 4-2 Block diagram of Yocto compilation

Yocto has a very flexible build system that allows the use of shell and python for a variety of special cases. More information about Yocto can be found below:

- Yocto
- Rockchip Yocto
- Yocto stable branch

4.3SDK Software Block Diagram

The SDK software, shown in Figure 4-3, is divided into four levels, from bottom to top: Bootloader, Linux Kernel, Libraries, and Applications. The contents of each level are as follows:

- The Bootloader layer mainly provides underlying system support packages, such as Bootloader, U-Boot, ATF related support.
- The Kernel layer provides a standard implementation of the Linux Kernel, which is an open operating system. The Linux kernel on the Rockchip platform is the standard Linux 4.4/4.19/5.10 kernel, which provides basic support for security, memory management, process management, and the grid stack. The main purpose is to manage the device hardware resources, such as CPU scheduling, cache, memory, I/O, etc., through the Linux kernel. The Linux kernel manages the hardware resources of the device, such as CPU scheduling, cache, memory, I/O, and so on.
- The Libraries layer corresponds to a general embedded system and is equivalent to the middleware layer. The Libraries layer contains various system libraries and third-party open-source libraries, and provides API interfaces to the application layer so that system customizers and application developers can develop new applications based on the APIs in the Libraries layer.
  用The
- The Applications layer is mainly for implementing specific product functions and interaction logic, and requires the support of some system base libraries and third-party libraries, so that developers can develop and implement their own applications to provide the system's various strengths to end-users.

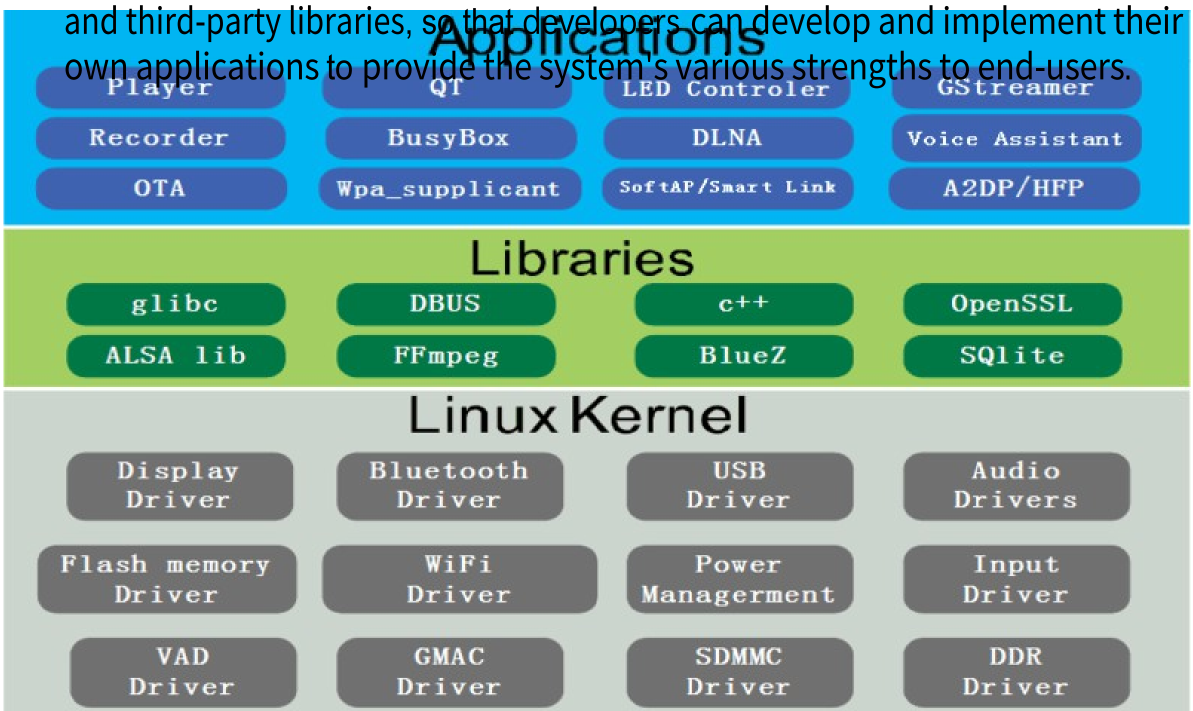


Figure 4-3 SDK Software Block Diagram

The SDK system startup flow

is shown in Figure 4-3.1.

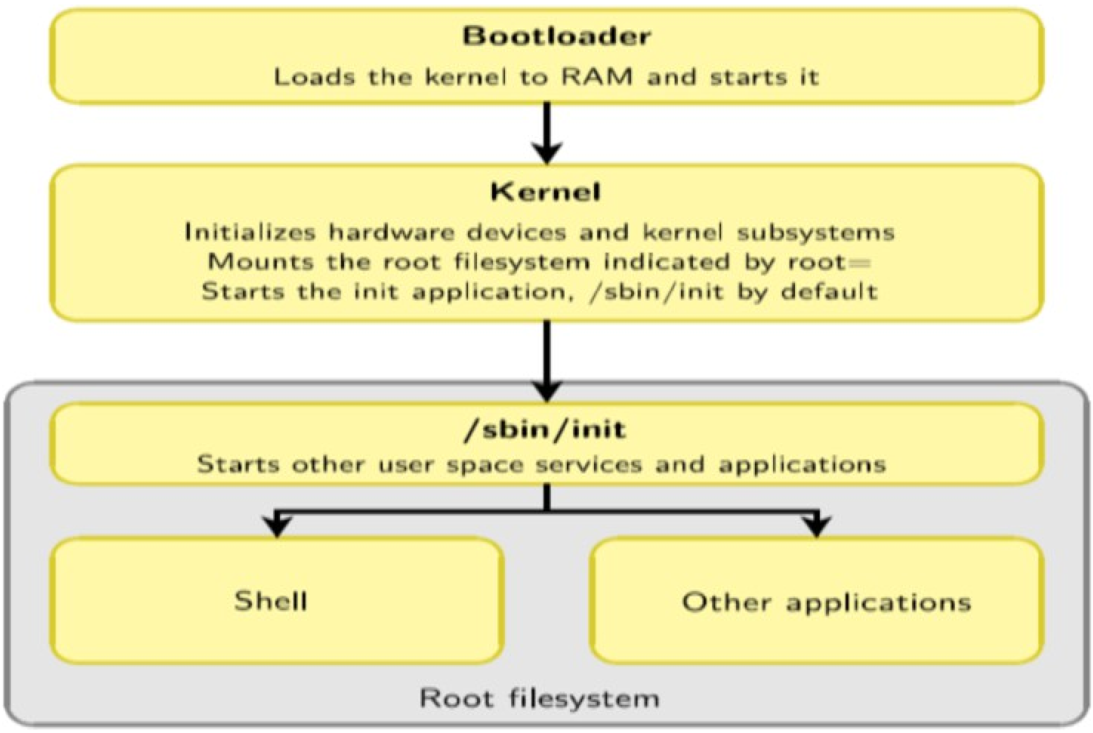

Bootloader
Loads the kernel to RAM and starts it
Kernel
Initializes hardware devices and kernel subsystems
Mounts the root filesystem indicated by root=
Starts the init application, /sbin/init by default
/sbin/init
Starts other user space services and applications
Shell
Other applications
Root filesystem

Figure 4-3.1 SDK startup flow

4.4SDK Development Process

The Rockchip Linux system is based on Buildroot/Yocto/Debian system, and the kernel is developed based on kernel 4.4/4.19/5.10, and the SDK is developed for various product forms, which can be used for system customization and application porting development.

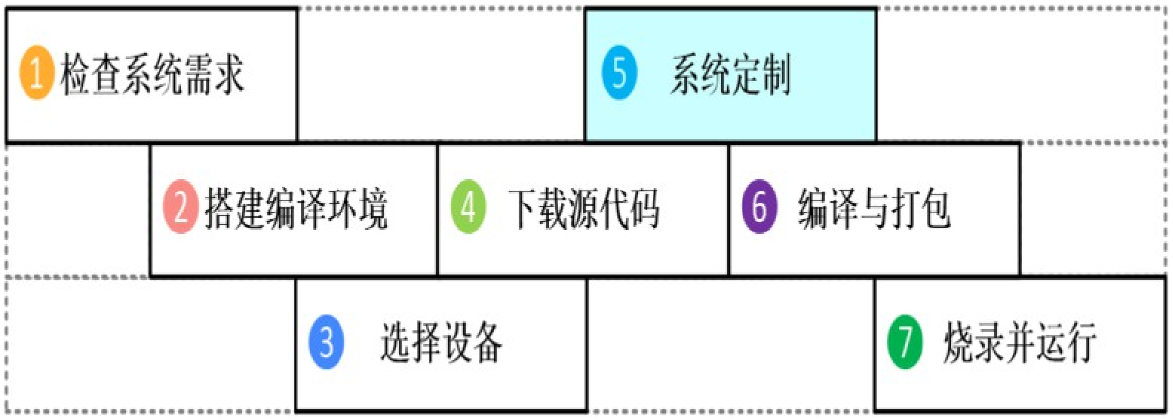


Figure 4-4 SDK Development Flow

As shown in Figure 4-4, developers can follow the development process described above to quickly build the Rockchip Linux system development environment and compile code locally. The following section briefly describes the process:

- Check system requirements: Before downloading code and compiling, make sure that the local development equipment is adequate for the requirements, including the hardware capabilities of the machine.
  力The SDK supports Linux operating systems, software systems, tool chains,

etc. The SDK currently supports compilation under Linux operating systems and tool chains under Linux only. So far the SDK supports compilation under Linux operating system and provides support for toolchain only under Linux environment, other systems such as MacOS, Windows, etc. are not supported at the moment.

- Build the compilation environment: Introduce the various software packages and tools that need to be installed on the development machine. For details, please refer to Chapter 5, Building the Development Environment, to know the verified version of the Linux operating system of Rockchip Linux, and the library files that are relied upon for compilation. Selection of Device: During the development process, developers are required to select the corresponding hardware board type according to their own needs. SDK Adaptation Hardware Full Auto Compilation Summary is available for more details. Downloading Source Code: After selecting the device type, you need to install the repo tool to download the source code in batch, please refer to Section 6.4 SDK Acquisition for details. System Customization: Developers can customize U-Boot (Section 9.1 U-Boot Development), Kernel (Section 9.2 Kernel Development), and Buildroot (Section 9.4 Buildroot Development) according to the hardware boards and product definitions used, please refer to the descriptions of the related development guides and configurations in the chapters.

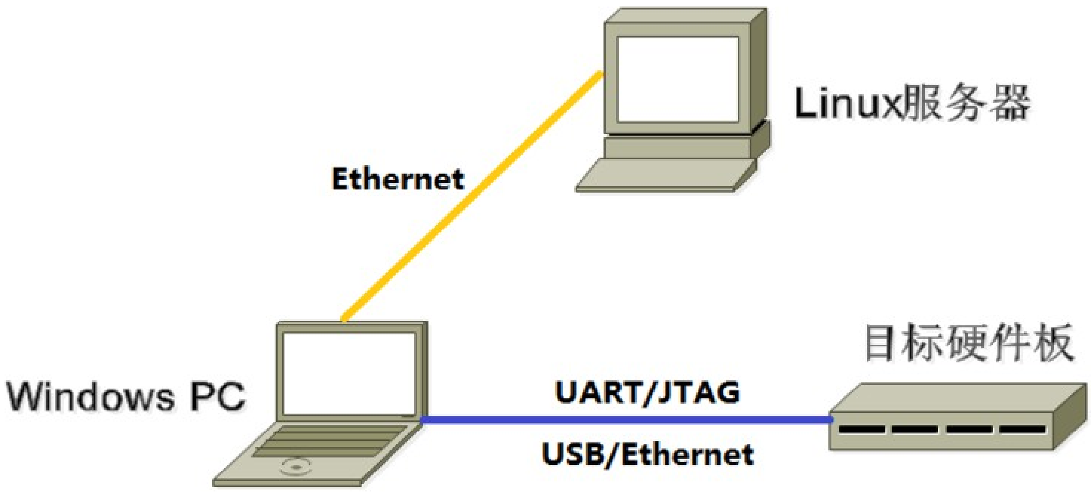
- Compiling and Packaging: This section describes how to select the product and initialize the compilation environment once the source code is available, and then execute compilation commands, including whole or module compilation and compilation cleanup, as described in Chapter 7, Compiling the SDK. Burning and Execution: Following the creation of an image file, we describe how to burn the image and execute it on a hardware device, as described in Section 8.1, Burning SDK Images.

5.SDK development environment setup

5.1summarize

This section describes how to build a local build environment to compile the Rockchip Buildroot Linux SDK source code. The current SDK only supports compilation under Linux environment and provides cross-compilation toolchain under Linux.
A typical embedded development environment usually consists of a Linux server, a Windows PC, and the target hardware version, as shown in the figure below.
示The

- Building a Cross-Compilation Environment on Linux Servers to Provide Code Update Download and Cross-Compilation Services for Software Development.
- You can share the program between a Windows PC and a Linux server, install Putty, remotely log in to the Linux server over the Internet, and perform cross-compilation and code development and debugging.
- A Windows PC connected to the target hardware board via serial port and USB can burn compiled image files to the target hardware board and debug the system or application.



[Note: A Windows PC is used in the development environment, but in fact many tasks can be done a Linux PC as well, such as using a minicom instead of a Linux PC.
You can choose from Putty, etc. for yourself.

5.2Linux server development environment setup

We recommend that Ubuntu 20.04 be used to compile the program. Other Linux versions may require adjustments to the software package. In addition to system requirements, there are other hardware and software requirements.
Hardware Requirements: 64-bit system with more than 40 gigabytes of hard disk space; if you are building multiple builds, you will need more hard disk space. Software requirements: Ubuntu 20.04:
The package installation commands for building the SDK environment are as follows:

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

Python dependency packages:

```
sudo pip3 install asciidoc
```

Ubuntu 20.04 or higher is recommended for development. If you encounter errors during compilation, you can install the appropriate packages depending on the error message.

Considering the time cost of setting up the development environment for customers, we also provide the cross-compiler docker image method for customers to verify and shorten the time-consuming compilation environment setup.

Refer to file `Docker/Rockchip_Developer_Guide_Linux_Docker_Deploy_CN.pdf`.

Docker compiled image system compatibility test results are referenced below:

Versions	**Docker** version	Mirror loading	Firmware Compilation
ubuntu 21.10	20.10.12	pass	pass
ubuntu 21.04	20.10.7	pass	pass
ubuntu 18.04	20.10.7	pass	pass
fedora35	20.10.12	pass	NR (not run)

5.2.1 The release package enables用 **Linux** server system versions

The following version of Linux is installed in the SDK development environment, and the SDK is compiled on this Linux system by default:

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

5.2.2 Introduction to the Cross-Compilation Toolchain

Since the Rockchip Linux SDK is so far only compiled for Linux PCs, we have only provided the cross-compilation toolchain for Linux. The compilation toolchain for U-Boot and Kernel is pre-built under prebuilt/gcc, and buildroot uses the toolchain compiled from this open source software.

- U-Boot and Kernel Compilation Toolchain:

```
Linux 4.4/4.19 SDK.
prebuilts/gcc/linux-x86/aarch64/gcc-linaro-6.3.1-2017.05-
x86_64_aarch64-linux- gnu/bin/aarch64-linux-gnu-

Linux 5.10 SDK.
prebuilts/gcc/linux-x86/aarch64/gcc-arm-10.3-2021.07-x86_64-aarch64-
none-linux- gnu/bin/aarch64-none-linux-gnu-
```

equivalent version

```
Linux 4.4/4.19 SDK.
gcc version 6.3.1 20170404 (Linaro GCC 6.3-2017.05)

Linux 5.10 SDK.
gcc version 10.3.1 20210621 (GNU Toolchain for the A-profile
Architecture 10.3- 2021.07 (arm-10.29))
```

- Buildroot's built-in cross-compilation

If you need to compile individual modules or third-party applications, you need to configure the cross-compilation environment. 比For example, for the RK3588, the cross-compilation tool is located in the
Under `buildroot/output/rockchip_rk3588/host/usr`, the bin/exports of the tool need to be changed to the same bin/exports as the
`aarch64-buildroot-linux-gnu/bin/` The target is set as an environment variable, and the script to configure the environment variable automatically is executed at the top-level target:

```
source envsetup.sh
```

Enter a command to view it:

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

The following message is printed:

```
aarch64-linux-gcc.br_real (Buildroot linux-5.10-gen-rkr3.3) 11.3.0
```

If toolchains for other platforms or versions are required, they need to be self-compiled.
After the above environment is ready, the Linux server development environment has been completed, you can download and compile the source code.

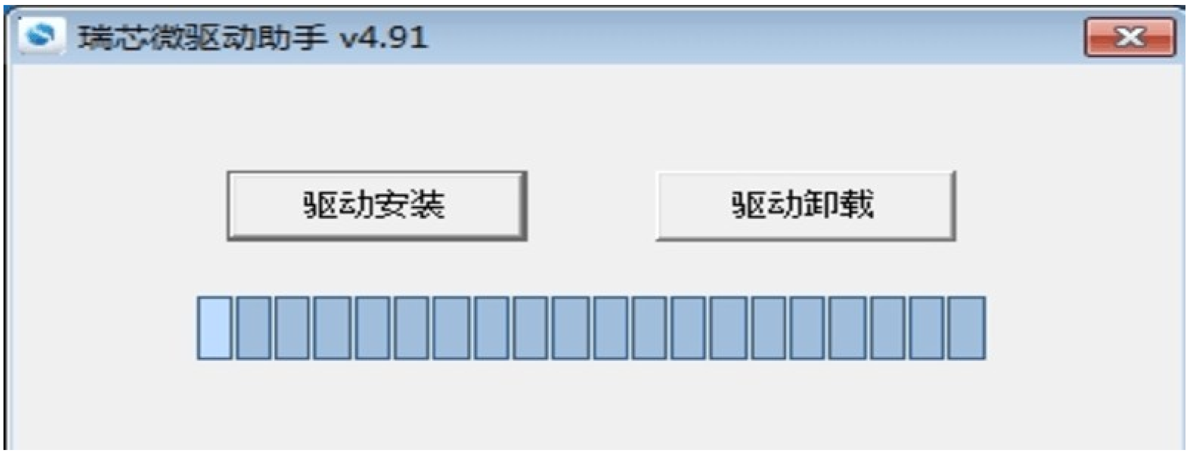
5.3Window PC development environment setup

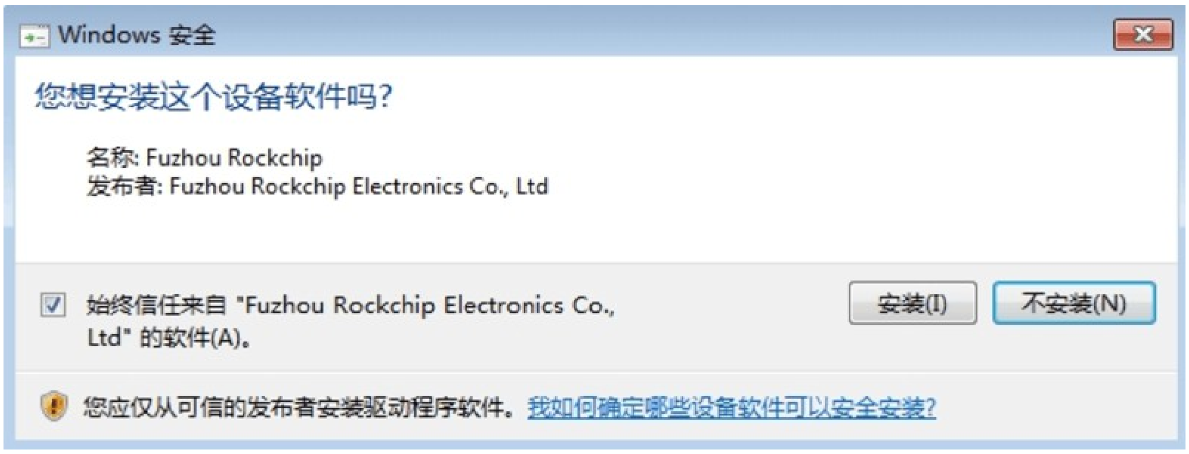
5.3.1 Installation of development work

- Please install editing software such as
- Vim and Notepad++ yourself.
  Download Virtual-Box (software
  that provides a virtual
  development environment).
- Download XShell6 (for establishing interaction with Linux systems)

5.3.2 Rockchip USB Driver Installation

During the development and debugging phase, the device needs to be switched to Loader mode or Maskrom mode, and the Rockusb driver needs to be installed to recognize the device properly.
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:




Windows 安全

您想安装这个设备软件吗?

名称: Fuzhou Rockchip
发布者: Fuzhou Rockchip Electronics Co., Ltd

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

安装(I) 不安装(N)

您应仅从可信的发布者安装驱动程序软件。我如何确定哪些设备软件可以安全安装?

5.3.3 Windows Burner用

Burning Tools for Windows Systems was released in tools/windows/RKDevTool/RKDevTool_Release can be used for development debugging and firmware writing in Windows environment. For more details, please refer to Section 12.3, RKDevTools.

5.3.4 Objective Hardware Board Preparation

Please refer to the list of hardware used in the SDK package to select the corresponding hardware board for subsequent development and debugging. The corresponding hardware user's guide file describes the hardware interface, user's guide, and burning operation.

6. Preparing for SDK Installation

6.1brief introduction

The Rockchip Linux SDK code and related files are versioned in separate干 git repositories, and developers can use repo
These git repositories can be downloaded, committed, and supported.

6.2Install the repo

Make sure that there is a bin/ entry under the master and that it is included in the path:

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

If you have access to google address, download the Repo utility and make sure it works:

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

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

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

In addition to the above two ways, you can also use the following command to obtain a repo

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

6.3Git Configuration

Please configure your own git information using the repo, otherwise you may encounter hook checking obstacles later on:

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

6.4SDK Acquisition

The SDK is distributed through the Rexchip code server. To apply for the SDK, customers must provide their SSH public key for server authentication and authorization, and can synchronize their code after obtaining the authorization. For more information about SSH public key authorization for Rexchip's code server, please refer to the following section
This chapter describes the SSH public key operation.

6.4.1 SDK Download Command

The Rockchip Linux SDK is designed to be adapted to a wide range of platforms such as RK3588, RK3566, RK3568, RK3308, RK3288, RK3326/PX30, RK3399, RK3399Pro, RK1808, etc., and the source code will vary to some extent for different platforms. When downloading the source code, the developer declares which platform he/she wants so that he/she does not have to download code that he/she does not need.

SDKs Use -m< *SDK* Version Release>.xml to declare which platform you want to download.

Please refer to the section of this file to download through the code server.

The code will start to download automatically, and careful waiting is required. The source code file will be located under the corresponding project name in the workbook. The initial synchronization will take an hour or more to complete.

6.4.2 SDK Code Zip

In order to facilitate customers to get the SDK source code quickly, Rexchip Technology Window usually provides the initial zip package of the corresponding version of the SDK, through which developers can get the initial zip package of the SDK code, which is the same as the source code downloaded through the repo.

Please refer to the chapter of this file to get it by unzipping the local zip file.

6.5Software update log

Software release upgrades are viewed via the workspace xml as follows:

```
.repo/manifests$ realpath
rk3588_linux_release.xml # Example: prints
version number v1.0.3 with update 20220920
# <SDK>/.repo/manifests/rk3588_linux_release_v1.0.3_20220920.xml
```

Software release version upgrade updates can be viewed through the workbook at 比 such as RK3588 can be viewed as follows:

```
.repo/manifests/rk3588_linux$ cat RK3588_Linux_SDK_Note.md
```

Or refer to the Workbook:

```
<SDK>/docs/RK3588/RK3588_Linux_SDK_Note.md
```

6.6SDK Updates

Subsequent developers can synchronize the update by command according to the update instructions published periodically in the FAE window.

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

6.7SDK Issue Feedback

The Rockchip bug system (Redmine) records the process and status of user issues in order to better assist customer development and facilitate simultaneous tracking by both parties, making issue handling more timely and efficient. Subsequent SDK issues, specific technical issues, and technical inquiries can be submitted to this bug system, and Rockchip technical services will distribute, handle, and track the issues in a timely manner. For more details, please refer to the following file /docs/Others/Rockchip_User_Guide_SDK_Application_And_Synchronization_CN.pdf.

7.SDK Compilation

This SDK was developed and tested on an Ubuntu system. It is recommended that the SDK be compiled on Ubuntu 22.04. Other
The Linux version may require adjustments to the software package. In addition to the system requirements, there are other hardware and software requirements. Hardware requirements: 64-bit system with more than 40 gigabytes of hard disk space; if you are doing multiple builds, you will need more hard disk space. Software requirements: Ubuntu 22.04.

To install the packages for building the SDK environment, refer to Package Installation.

7.1U-Boot Compilation

Enter the SDK. Perform the following commands to compile the SDK

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

For specific board-level compilation, refer to the compilation instructions in the SDK release file.

7.2Kernel Compilation

The kernel is compiled and packaged automatically executing the following commands at the root of the project.

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

Refer to the release notes or the compilation instructions in Quick Start for specific board-level compilation.

7.3Recovery Compilation

Execute the following commands at the root of the task to automatically complete the compilation and packaging of Recovery.

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

Compile and create recovery.img in the Buildroot destination output/rockchip core片 model recovery/images.

Note that recovery.img includes kernel.img, so every time the kernel is changed, Recovery will need to be repackaged to create a new version. For example, see below:

```
<SDK>$source envsetup.sh rockchip_core name
<SDK>$make recovery-rebuild
<SDK>$. /build.sh recovery
```

Please refer to the SDK release file for more compilation instructions.

7.4Buildroot Compilation

7.4.1 Rootfs compilation

The compilation and packaging of Rootfs is done automatically by executing the following commands at the root of the project:

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

Compile and create rootfs.ext4 in the Buildroot destination output/rockchip_models/images.

7.4.2 Module Compilation

比For example, for the rockchip-test module, the following compilation commands are commonly used:

- Compile rockchip-test

```
SDK$make rockchip-test
```

-
-

  Reprogramming rockchip-test

```
SDK$make rockchip-test-rebuild
```

  Delete rockchip-test

```
SDK$make rockchip-test-dirclean
or
SDK$rm -rf /buildroot/output/rockchip_rk3588/build/rockchip-test-master
```

7.5Debian compilation

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

or enter a debian/ entry:

```
cd debian/
```

For subsequent compilation and creation of Debian firmware, please refer to the current entry readme.md.

(1)Building base Debian system

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

Compiling 32-bit Debian.

```
RELEASE= bullseye TARGET= desktop ARCH= armhf . /mk-base-debian.sh
```

or compiling 64-bit Debian.

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

When the compilation is complete, the following will be created in the debian/ directory: linaro-bullseye-alip-xxxxx-1.tar.gz (xxxxxx indicates the time stamp of the creation).

FAQ:

- If the above compilation encounters the following problematic situation:

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

Solution:

```
mount -o remount,exec,dev xxx yyy
(where xxx      is the path to the workbase and yyyy is the mount, point and recompile)
```

If any other compilation exceptions are encountered, first rule out the use of the ext2/ext4 compilation system.

- Since compiling Base Debian requires access to a foreign web site, the download often fails when accessing a foreign web site from a domestic web site: Debian uses live build, and can be configured this way if you change the mirror source to a domestic one:

```
32-bit systems:
--- b/ubuntu-build-service/bullseye-desktop-armhf/configure
+++ b/ubuntu-build-service/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 "https://mirrors.tuna.tsinghua.edu.cn/debian" \
+ --mirror-chroot "https://mirrors.tuna.tsinghua.edu.cn/debian" \
+ ---mirror-chroot-security "https://mirrors.tuna.tsinghua.edu.cn/debian-
security"
\windshield
+ ---mirror-binary "https://mirrors.tuna.tsinghua.edu.cn/debian" \
+ ---mirror-binary-security "https://mirrors.tuna.tsinghua.edu.cn/debian-
security"
  --apt-indices false \
  --apt-recommends false \
  --apt-secure false \
```

64-bit systems:

```
--- a/ubuntu-build-service/bullseye-desktop-arm64/configure
+++ b/ubuntu-build-service/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 "https://mirrors.tuna.tsinghua.edu.cn/debian" \
+ --mirror-chroot "https://mirrors.tuna.tsinghua.edu.cn/debian" \
+ ---mirror-chroot-security "https://mirrors.tuna.tsinghua.edu.cn/debian-
security"
\windshield
+ ---mirror-binary "https://mirrors.tuna.tsinghua.edu.cn/debian" \
+ ---mirror-binary-security "https://mirrors.tuna.tsinghua.edu.cn/debian-
security"
  --apt-indices false \
  --apt-recommends false \
  --apt-secure false \
```

If you are unable to download the package for other reasons, you can share the pre-compiled package on your Baidu Cloud web disk and perform the next step directly from the current destination.

(2)Building rk-debian rootfs

Compile 32-bit Debian:

```
VERSION= debug ARCH= armhf . /mk-rootfs-bullseye.sh
```

or compile 64-bit Debian:

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

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

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

This will result in linaro-rootfs.img.

7.6Compiled by Yocto

The compilation and packaging of Rootfs is done automatically by executing the following commands at the root of the project:

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

Compile and create rootfs.img under

yocto/ build/lastest. FAQ:

If you encounter the following problem situation when compiling the logos:

```
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 the setup-locale-p y thon3 compilation for image creation at yocto/build/lastest/rootfs.img, with the default user name being root.
Yocto For more information, please refer to the Rockchip Wiki.

7.7Automatic compilation

After completing the compilation of the Kernel/U-Boot/Recovery/Rootfs sections as described above, go to the root of the project and execute the following commands to automatically complete all the compilation:

```
. /build.sh all # Compile only module code (u-Boot, kernel,
Rootfs, Recovery)
                # To package the firmware, you need to run .
                /mkfirmware.sh to perform firmware packaging.
.               # In . /build.sh all
/build.s        # 1. add firmware package .
h               /mkfirmware.sh # 2.
                update.img packaging
                # 3. Copy the firmware from the rootdev directory to
                the IMAGE/***_RELEASE_TEST/IMAGES directory.
                # 4. Save patches for each module to the
                IMAGE/***_RELEASE_TEST/PATCHES directory
                # Note: . /build.sh is the same as . /build.sh
                allsave command.
```

The default is Buildroot and rootfs can be specified by setting the bad variable RK_ROOTFS_SYSTEM.
比If you want to buildroot, you can execute the following commands to create it:

```
<SDK>$export RK_ROOTFS_SYSTEM= buildroot
<SDK>$. /build.sh
```

For details on the use of the parameters, see比 :

```
<SDK>$ . /build.sh --help
```

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

```
rk3588$  ./build.sh
-h Usage: build.sh
[OPTIONS] Available
options:
BoardConfig*.     -switch to specified board config
mk-lunch          -list current SDK boards and switch to specified
                  board configuration
wifibt            -build wifibt
uboot             -build uboot
uefi         -build uefi
spl                -build spl
loader             -build loader
kernel             -build kernel
modules            -build kernel modules
toolchain          -build toolchain
rootfs             -build default rootfs, currently build buildroot
as default buildroot     -build buildroot rootfs
ramboot            -build ramboot image
multi-npu_boot     -build boot image for multi-
npu board yocto    -build yocto rootfs
debian             -build debian rootfs
pcba               -build pcba
Recovery -build    -build recovery
all                -build uboot, kernel, rootfs,
recovery image cleanall       -clean uboot, kernel,
rootfs, recovery firmware     -pack all the image we
need to boot up system updateimg    -pack update image
otapackage         -pack ab update otapackage image
(update_ota.img) sdpackage    -pack update sdcard package
image (update_sdcard.img)
save               -save images, patches, commands used to debug
```

```
allsave            -build all & firmware & updateimg &
                   save
check              -Check the environment of building
info               -See the current board building
                   information
app/<pk
g>                 -build packages in the dir of app/*
external/<pk
g>                 -build packages in the dir of
                   external/*
createkeys         -create secureboot root keys
security_rootf
s (just for
dm-v)              -build rootfs and some relevant images with
                   security paramter
security_boot      -build boot with security
                   paramter
security_uboot     -build uboot with security
                   paramter
security_recov
ery                -build recovery with security
                   paramter
security_check     -Check security paramter if
                   it's good.
Default option is
'allsave'.
```

8.SDK Firmware Upgrade

This section describes how to write and execute a complete image file on a hardware device.
Just a few of the tools available on the Rockchip platform are described below, so you can choose the appropriate method to burn the image. Before burning, you need to install the latest USB driver, see Rockchip USB Driver Installation.

Ɪutensil	Operations system	descriptive
RKDevTool	Windows (computer)	Raisin Micro Development Tools, Sub-Upgrade Firmware and the Entire Update Upgrade Firmware Tool
FactoryTool	Windows (computer)	Mass production upgraders supporting USB multi-programming
Linux_Upgrade_tool	Linux	Developed for Linux, supports firmware upgrades.

8.1Introduction to burn-in modes

The hardware of the Rockchip platform operates in the modes shown in the table, only when the device is in the Maskrom and Loader modes, and only when the device is in the Maskrom and Loader modes, and only when the device is in the Loader mode. Firmware can only be burned or updated in this mode.

paradigm	Work piece Burning	descriptive
Maskrom	支run (i.e. administer)	When no firmware is burned, the flash memory will boot into Maskrom mode, which allows the initial firmware write to be performed; If you encounter a situation where the loader cannot start normally during development and debugging, you can also go to Maskrom mode to burn firmware.
loaders	支run (i.e. administer)	Firmware can be burned and upgraded in Loader mode. A partition image file can be individually burned by the tool to facilitate debugging.
Recovery	No support	System boot recovery starts, mainly for upgrading and restoring boot settings.
Normal Boot	No support	The system boots rootfs, loads rootfs, and most development is debugged in this mode.

The following methods are used to enter the writing mode:

- No firmware burned, power up, enter Maskrom mode.
- After burning the firmware, press and hold the recovery button to power up or reset the system, the system will enter the Loader firmware writing mode.
- After burning the firmware, press and hold the Maskrom button to power up or reset, the system will enter the MaskRom firmware writing mode.
- After the firmware is burned and the board enters the system normally after power-on or reset, "An ADB device is found" or "An MSC device is found" is displayed on the Rxmicro development tool, and then click the button "Switch" on the tool to enter the Loader mode. Then click the button "Switch" on the tool to enter Loader mode.
- After burning the firmware, you can enter Loader mode by entering the reboot loader command in serial or ADB command mode.

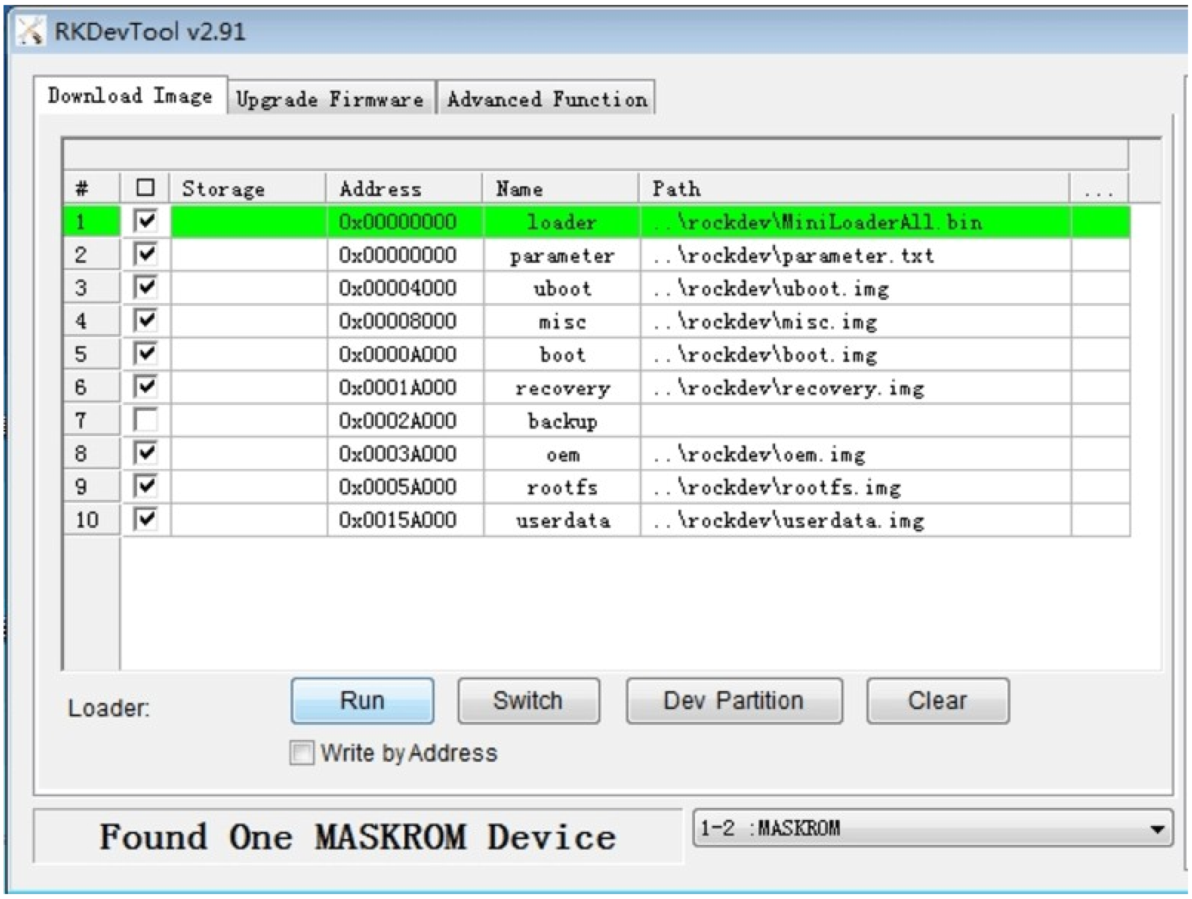
8.1.1 Windows Brush Instructions

The SDK is supplied with a Windows burner (version V2.84 or above is required),

which is located at the root of the workbench:

```
tools/
├── windows/RKDevTool
```

Once the firmware has been compiled, the device should be written in MASKROM or BootROM mode. After connecting the USB download cable, press and hold down the "MASKROM" button and press the reset button "RST", then release your hand to enter the MASKROM mode. Then, you can enter the
In MASKROM mode, after loading the path to the firmware that has been compiled and created, click "Execute" to burn the firmware, or press the "recovery" button and release the reset button "RST" to enter MASKROM mode. The following is the partition offset and the file to be written in MASKROM mode. (Note: Windows PCs are required to run the tool with administrator privileges to perform the task.)



#	☐	Storage	Address	Name	Path	...
1	✓		0x00000000	loader	..\rockdev\MiniLoaderAll.bin	
2	✓		0x00000000	parameter	..\rockdev\parameter.txt	
3	✓		0x00004000	uboot	..\rockdev\uboot.img	
4	✓		0x00008000	misc	..\rockdev\misc.img	
5	✓		0x0000A000	boot	..\rockdev\boot.img	
6	✓		0x0001A000	recovery	..\rockdev\recovery.img	
7	☐		0x0002A000	backup		
8	✓		0x0003A000	oem	..\rockdev\oem.img	
9	✓		0x0005A000	rootfs	..\rockdev\rootfs.img	
10	✓		0x0015A000	userdata	..\rockdev\userdata.img	

Found One MASKROM Device

Note: Before burning, you need to install the latest USB driver:

```
<SDK>/tools/windows/DriverAssitant_v5.11.zip
```

8.1.2 Linux Brush Instructions

The burning tool for Linux is located under tools/linux (Linux_Upgrade_Tool tool version needs to be V2.17 or above), make sure your board is connected to the MASKROM/loader rockusb.比 If the compiled firmware is located under rockdev, 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 sudo .
/upgrade_tool di -misc rockdev/misc.img
sudo . /upgrade_tool di -b
rockdev/boot.img
sudo . /upgrade_tool di -recovery
rockdev/recovery.img sudo . /upgrade_tool di
-oem rockdev/oem.img
sudo . /upgrade_tool di -rootfs
rocdev/rootfs.img sudo . /upgrade_tool di -
userdata rockdev/userdata.img sudo .
/upgrade_tool rd
```

or upgrade the packaged full firmware:

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

The following upgrades can be performed on the machine in the MASKROM state, either at the root destination or at the root directory:

```
. /rkflash.sh
```

8.1.3 System Partition Description

Default Partition Description (The following is the RK3568 EVB partition reference)

Number	Start (sector)	End (sector)	Size	Name
1	16384	24575	4096K	uboot
2	24576	32767	4096K	misc
3	32768	98303	32M	boot
4	98304	163839	32M	recovery

5	163840	229375	32M	bakcup
6	229376	12812287	6144M	rootfs
7	12812288	13074431	128M	oem
8	13074432	61071326	22.8G	userdata

- uboot partition: uboot.img for uboot compilation.• misc partition: misc.img for recovery. • boot partition: boot.img for kernel compilation.
- recovery partition: .img for recovery compilation.
- Backup partition: Reserved, not用 for now, and will be used as backup for recovery as in Android.
- rootfs partition: rootfs.img for buildroot, debian or yocto.

- oem partition: Used by the user to store the user's APP or data. Mounted at /oem destination.
- userdata partition: for APPs temporary creation of files or for end-users, mounted under the /userdata destination.

9.SDK development

9.1U-Boot Development

This section briefly introduces the basic concepts of U-Boot and compilation considerations to help customers understand the RK platform U-Boot framework, specific U-Boot development details can be found in the "Rockchip-Developer-Guide-UBoot-*.pdf" under the /docs/Common/U-Boot目錄.

9.1.1 Introduction to U-Boot

v2017 (next-dev) is a version of v2017.09 that was developed by RK from the official v2017.09 release of U-Boot, which already supports RK.
All major on-sale cores are available at片 . The main features supported are:

- Supports RK Android firmware boot;
- Supports Android AOSP firmware booting;• supports Linux Distro firmware booting;
- Supports Rockchip miniloader and SPL/TPL pre-loader boot;
- Supports LVDS, EDP, MIPI, HDMI, CVBS, RGB, and other display devices;
- Supports eMMC, Nand Flash, SPI Nand flash, SPI NOR flash, SD card, USB flash drive, etc.;• supports FAT, EXT2, EXT4 file system;
- Supports GPT, RK parameter partition tables;
- Supports power-on logo, charging animation, low battery management, and power management;
- Supports I2C, PMIC, CHARGE, FUEL GUAGE, USB, GPIO, PWM, GMAC, eMMC, NAND, Interrupt, etc;
- Supports Vendor storage for storing user data and configurations;
- Supports RockUSB and Google Fastboot USB gadgets to write eMMC;• supports USB devices such as Mass storage, ethernet, HID, etc;
- Supports dynamic selection of kernel DTB by hardware state;

9.1.2 releases

There are two versions of U-Boot RK: v2014 old and v2017 new, internally named

rkdevelop and next-dev respectively.用
戸There are two ways to confirm if the current U-Boot is v2017 version.

Formula 1: Verify that the version number of the root Makefile is 2017.

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

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

Option 2: Confirm that U-Boot 2017.09 is printed on the first page of the computer.

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

> Project Object Open Source: v2017 is open source and regularly updated to Github: https://github.com/rockchip-linux/u-boot内核版本: v2017 requires RK kernel version>= 4.4

9.1.3 preliminary

- Download rkbin

  This is a repository for RK's non-open source bin, scripts, and packages that U-Boot compiles from, indexing relevant files to create loaders, trusts, and uboot firmware. rkbin and U-Boot processes must be kept at the same level of the directory.

  > Warehouse downloads: please refer to the Appendix section.

- Download GCC

  The GCC compiler uses gcc-linaro-6.3.1 and is placed in the prebuilts directory. prebuilts and U-Boot are sibling directories. Prebuilts and U-Boot maintain a sibling relationship, as shown below:

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

  > GCC downloads: please refer to the appendix section

- Select defconfig

lamp pith片	**defconfig**	支Holding **kernel dtb**	clarification
rv1108	evb-rv1108_defconfig	N	\windshield
rk1808	rk1808_defconfig	Y	\windshield
rk1806	rk1806_defconfig	Y	\windshield
rk3036	rk3036_defconfig	Y	\windshield
rk3128x	rk3128x_defconfig	Y	\windshield
rk3128	evb-rk3128_defconfig	N	\windshield
rk3126	rk3126_defconfig	Y	\windshield
rk322x	rk322x_defconfig	Y	\windshield
rk3288	rk3288_defconfig	Y	\windshield
rk3368	rk3368_defconfig	Y	\windshield
rk3328	rk3328_defconfig	Y	\windshield
rk3399	rk3399_defconfig	Y	\windshield
rk3399pro	rk3399pro_defconfig	Y	\windshield
rk3399pro-npu	rknpu-lion_defconfig	Y	\windshield
rk3308	rk3308_defconfig	Y	\windshield
rk3308-aarch32	rk3308-aarch32_defconfig	Y	\windshield
px30	px30_defconfig	Y	\windshield
rk3326	rk3326_defconfig	Y	\windshield
rk3326-aarch32	rk3326-aarch32_defconfig	Y	\windshield
rv1126	rv1126_defconfig	Y	generic version
rv1126	rv1126-ab.config	Y	Common version+ supports A/B.
rv1126	rv1126-emmc-tb.config rv1126-lp3-emmc-tb.config rv1126-spi-nor-tb.config	Y	eMMC+DDR3 Fast Boot eMMC+LP3 Fast Boot Spi Nor+DDR3 Fast Boot

rv1126	rv1126-spi-nor-tiny_defconfig rv1126-ramboot.config rv1126-usbplug.config	Y	Spi Nor Small Capacity No memory device (memory startup) usbplug function
rk3566	rk3566.config rk3566-eink.config	Y	Common Version Electronic Book Version
lamp pith片	**defconfig**	支Holding **kernel dtb**	clarification
rk3568	rk3568_defconfig rk3568-dfu.config rk3568-nand.config rk3568-spl-spi-nand_defconfig rk3568-aarch32.config rk3568-usbplug.config	Y	generic version Supporting dfu Supports MLC/TLC/eMMC SPI-nand dedicated SPL. Supports aarch32 mode Supports usbplug mode
rk3588	rk3588_defconfig	Y	generic version

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

9.1.4 Startup process

The U-Boot boot process for the RK platform is as follows, listing only the important steps:

```
start.s
    // Assembly environment
    =>
    IRQ/FIQ/lowlevel/vbar/errata/cp15/g         // ARM architecture related
    ic                                           lowlevel initialization
                                                 // Prepare the stack
    => _main                                     needed for the C
       // [Stage 1] Initialize the C environment environment
       and start a series of function calls.
       => stack
       => board_init_f: init_sequence_f[]
           initf_malloc
           arch_cpu_init                         // [Low-level initialization of
                                                 SoC]
           serial_init                           // Serial port initialization
           dram_init                             // [Getting ddr capacity
                                                 information]
           reserve_mmu                           // Reserve memory from the
           reserve_video                         end of ddr to a lower address.
           reserve_uboot
           reserve_malloc
           reserve_global_da
           ta reserve_fdt
           reserve_stacks
           dram_init_banksi                      // Determine the address of
           ze sysmem_init                        the U-Boot自身 to relocated.
           setup_reloc
       // Assembly                               // Assembly implementation of
       environment                               U-Boot code relocation
       => relocate_code                          // Enable MMU and
       // [Stage 2] Initialize the C environment I/Dcache
       and launch a series of function calls.
       => board_init_r: init_sequence_r[]
           initr_caches
           initr_malloc
           bidram_initr
           sysmem_initr
           initr_of_live                         // Initialize of_live
           initr_dm                              // Initialize the dm
                                                 framework
           board_init                            // [Platform Initialization,
                                                 Core Awareness]
               board_debug_uart_                 // Serial port iomux, clk
               init                              configuration
               init_kernel_dtb                   // [Cut to kernel dtb]!
               clks_probe                        // Initialize system
                                                 frequency
```

```
        regulators_enable_boo          // Initialize system power
        t_on io_domain_init            // io-domain initialization
        set_armclk_rate                // weak, ARM boost
                                       (implemented on demand by
                                       platform)
        dvfs_init                      // FM voltage regulation for wide
        rk_board_init                  temperature cores
    console_init_r                     // [Platform late Initialization]
    board_late_init                    //__weak, implemented on a
                                       // Set the mac address
                                       platform-specific basis.
                                       // Set serialno
        rockchip_set_ethaddr
        rockchip_set_serialno          // Parsing the "reboot xxx"
                                       command,
        setup_boot_mode                // Recognize key and loader burn
        charge_display                 modes, recovery
        rockchip_show_l                // U-Boot charging
        ogo                            // Display the power-up logo
        soc_clk_dump                   // Print the clk tree
        rk_board_late_i                //__ weak, implemented on a
        nit                            platform-specific basis.
    run_main_loop                      // [Enter command mode, or
                                       execute startup command].
```

9.1.5 (computer) shortcut key

The RK platform provides serial port keystroke combinations to trigger events for debugging and burning (if not triggered, try a few more times; enable secure-boot). (No effect when the power is on). Press and hold while turning on the power:

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

9.2 Kernel Development

This section briefly describes some common kernel configuration changes, mainly dts configuration, to help customers make simple changes faster and more convenient.The kernel version is based on 4.4, and is described accordingly.

9.2.1 Introduction to DTS

9.2.1.1 DTS Overview

Earlier versions of the Linux Kernel configured board-related information directly in board-level configuration files, such as IOMUX, GPIOs that are high/low by default, and client device information under each I2C/SPI bus. In order to abandon this 'hard code' approach, Linux introduces the Device Tree (Device Tree) concept to describe different hardware structures.

Device Tree data is highly readable, follows the DTS specification, and is usually described in .dtsi and .dts source files. During kernel compilation, it is compiled into .dtb second file. During the boot phase, the dtb is loaded by the bootloader (e.g. U-Boot) into an address space in RAM and passed to the Kernel space as a parameter, the kernel parses the entire dtb file and extracts each device information for initialization.

The purpose of this file is to introduce how to add a new board dts configuration and some common dts syntax descriptions. For more -
detailed dts syntax descriptions, please refer to: devicetree-specifications and devicetree-bindings if you are interested.

9.2.1.2 A new product, DTS, has been added.

- Create dts file

Linux Kernel currently supports multi-platform use of dts, and dts files for the RK platform are stored in:

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

The general naming convention for dts files is "soc-board-name.dts", e.g. rk3399-evb-ind-lpddr4-linux.dts. soc refers to the model number of the core, and board_name is generally based on the silkscreen of the board.
If your board is a body board, you only need one dts file to describe it.
If hardware is designed with a core and backplane architecture, or if the product has multiple product forms, you can put the public hardware description in a dtsi file, which describes the different hardware modules, and include the public hardware description by including "xxx.dtsi".
├óΓé¼┼ôrk3399-evb-ind-lpddr4-linux.dts
| ├── rk3399-evb-ind.dtsi
| └── rk3399-linux.dtsi

- Modify the Makefile of 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 Kenrel, you can make dts-name.img (e.g. rk3399-evb-ind-lpddr4-linux.img) to create a corresponding boot.img (with dtb data).

- Just a few notes on dts syntax

The dts syntax can be like c/c++ to include other public use dts data by #include xxx.dtsi.
dts file
Inherits properties and values from all device nodes of the containing dtsi file. If property is defined in more than one dts/dtsi file, its value ends up being the definition of the dts. All core-related controller nodes are defined in soc.dtsi, and to enable the device, set its status to "okay" in the dts file. To disable the device, set its status to "disabled" in the dts file.

```
/dts-v1/.

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

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

9.2.2 Kernel Module Development Files

\docs\Common\ Objects are released in functional modules with corresponding development files, and this section focuses on
These development files are summarized in an index, and you can refer to the following table to read and study the corresponding development guides, which are available under docs/Common and will be continuously improved and updated, taking into account the problems encountered in actual development.

Module Functions	sub-projects	Corresponding File
音repetitious	Audio	Rockchip_Developer_Guide_Audio_CN.pdf
conspicuous示	DISPLAY	Rockchip_Developer_Guide_HDMI-CEC_CN.pdf Rockchip_Developer_Guide_HDMI_CN.pdf Rockchip_Developer_Guide_HDMI-PHY- PLL_Config_CN.pdf Rockchip_DRM_Display_Driver_Development_Guide_ V1.0.pdf rockchip_drm_integration_helper-zh.pdf Rockchip_DRM_Panel_Porting_Guide_V1.6_ 20190228.pdf Rockchip_DRM_RK628_Porting_Guide_CN.pdf
USB	USB	Rockchip_Developer_Guide_Linux_USB_Initialization_Log_Analysis _CN.pdf Rockchip_Developer_Guide_Linux_USB_Performance_Analysis_CN. Rockchip_Developer_Guide_USB2_Compliance_Test_CN.pdf Rockchip_Developer_Guide_Linux_USB_PHY_CN.pdf Rockchip_Developer_Guide_USB_PHY_CN. CN.pdf Rockchip_Developer_Guide_USB_EN.pdf Rockchip_Developer_Guide_USB_FFS_Test_Demo_CN.pdf Rockchip_Developer_Guide_USB_Gadget_UAC_CN.pdf Rockchip_Developer_Guide_USB_Gadget_UAC_CN.pdf Rockchip_Developer_Guide_Linux_USB_PHY. CN.pdf Rockchip_Developer_Guide_USB_SQ_Test_CN.pdf Rockchip_RK3399_Developer_Guide_USB_DTS_CN.pdf Rockchip_RK356x_Developer_Guide_USB _CN.pdf Rockchip_Trouble_Shooting_Linux4.19_USB_Gadget_UVC_CN.pdf
I2C	I2C	Rockchip_Developer_Guide_I2C_CN.pdf
IOMMU	IOMMU	Rockchip_Developer_Guide_Linux_IOMMU_CN.pdf
MMC	MMC	Rockchip_Developer_Guide_SD_Boot_CN.pdf Rockchip_Developer_Guide_SDMMC_SDIO_eMMC_CN.pdf
PCIe	PCIe	Rockchip_Developer_Guide_Linux4.4_PCIe_CN.pdf Rockchip_RK356X_Developer_Guide_PCIe_CN.pdf
GPIO	Pin-Ctrl	Rockchip-Developer-Guide-Linux-Pin-Ctrl-CN.pdf Rockchip_Problem_Shooting_Linux_GPIO_CN.pdf
Power supply, power consumption	Power	Rockchip_Developer_Guide_Power_Analysis_CN.pdf Rockchip_Developer_Guide_Power_Analysis_EN.pdf
SARADC	SARADC	Rockchip_Developer_Guide_Linux_SARADC_CN.pdf
temperature control	THERMAL	Rockchip_Developer_Guide_Thermal_CN.pdf Rockchip_Developer_Guide_Thermal_EN.pdf

TRUST	TRUST	Rockchip_Developer_Guide_Trust_CN.pdf Rockchip_Developer_Guide_Trust_EN.pdf Rockchip_RK3308_Developer_Guide_System_Suspend_CN.pdf Rockchip_RK3308_Developer_Guide_System_Suspend_EN.pdf Rockchip_RK3399_Developer_Guide_System_Suspend_CN.pdf
U-Boot	U-Boot	Rockchip_Developer_Guide_Linux_AB_System_CN.pdf Rockchip_Developer_Guide_UBoot_MMC_Device_Analysis_CN.pdf Rockchip_Developer_Guide_UBoot_MTD_Block_Device_Design_CN .pdf UBoot_MTD_Block_Device_Design_CN.pdf Rockchip_Developer_Guide_UBoot_Nextdev_CN.pdf Rockchip_Introduction_UBoot_rkdevelop_vs_nextdev_ CN.pdf
feed a dog	WATCHDOG	Rockchip_Developer_Guide_Linux_WDT_CN.pdf Rockchip_Developer_Guide_Linux_WDT_EN.pdf
Clock Configuration	CRU	Rockchip-Clock-Developer-Guide-RTOS-CN.pdf Rockchip_RK3399_Developer_Guide_Linux4.4_Clock_CN.pdf

Module Functions	sub-projects	Corresponding File
JTAG GDB etc. Debugging	DEBUG	Rockchip_Developer_Guide_DS5_CN.pdf Rockchip_Developer_Guide_GDB_Over_ADB_ CN.pdf Rockchip_Developer_Guide_OpenOCD_CN.pd f Rockchip_Developer_Guide_OpenOCD_CN. pdf Rockchip_Developer_Guide_J- Link_CN.pdf User_Guide_J-Link_CN.pdf
CPU/GPU etc. frequency voltage adjustm ent	DVFS	Rockchip_Developer_Guide_CPUFreq_CN.pdf Rockchip_Developer_Guide_CPUFreq_EN.pdf Rockchip_Developer_Guide_Devfreq_CN.pdf Rockchip_ Developer_Guide_Devfreq_EN.pdf
Ethernet Configuration	GMAC	Rockchip_Developer_Guide_Linux_GMAC_RGMII_Delayline_CN.pd f Rockchip_Developer_Guide_Linux_GMAC_RGMII_Delayline_EN.pd f Rockchip_Developer_Guide_Linux_GMAC_CN.pdf Rockchip_Developer_Guide_Linux_GMAC_Mode_Configuration_CN .pdf Developer_Guide_Linux_GMAC_CN.pdf Rockchip_Developer_Guide_Linux_GMAC_Mode_Configuration_CN .pdf Rockchip_Developer_Guide_Linux_MAC_TO_MAC_CN. TO_MAC_CN.pdf
GPIO Power Domain	IO-DOMAIN	Rockchip_Developer_Guide_Linux_IO_DOMAIN_CN.pdf

PMIC power meter, DCDC	PMIC	Rockchip_Developer_Guide_Power_Discrete_DCDC_EN.pdf<br/ Rockchip_RK805_Developer_Guide_CN.pdf Rockchip_RK808_Developer_Guide_CN. Rockchip_RK808_Developer_Guide_CN.pdf Rockchip_RK809_Developer_Guide_CN.pdf Rockchip_RK816_Developer_Guide_CN.pdf<br/ Rockchip_RK817_Developer_Guide_CN.pdf Rockchip_RK818_Developer_CN. RK818_Developer_Guide_CN.pdf Rockchip_RK818_RK816_Developer_Guide_Fuel_Gauge_CN.pdf Rockchip_RK818_RK816_Introduction_Fuel_Gauge_Log _CN.pdf
PWM	PWM	Rockchip_Developer_Guide_Linux_PWM_CN.pdf Rockchip_Developer_Guide_Linux_PWM_EN.pdf
SPI	SPI	Rockchip_Developer_Guide_Linux_SPI_CN.pdff Rockchip_Developer_Guide_Linux_SPI_EN.pdf
serial port communication	UART	Rockchip_Developer_Guide_UART_CN.pdff Rockchip_Developer_Guide_UART_EN.pdf

9.2.3 GPIO

比For example, the RK3399/RK3399Pro provides 122 GPIOs in 5 groups (GPIO0~GPIO4), all of which can be used as interrupts.
GPIO0/GPIO1 can be used as system wake-up pins, all GPIOs can be configured as pull-ups or pull-downs in software, and all GPIOs default outputs.
入For example, the GPIO4C0 in the schematic and the corresponding GPIO in the dts should be "gpio4 16". Regarding the correspondence between the GPIOs `<SDK>/docs/Common` /Pin-
`Ctrl/Rockchip_Developer_Guide_Linux_Pinctrl_CN.pdf` .
in the schematic and the GPIOs in the dts, for example, GPIO4C0, the corresponding GPIO in the dts should be "gpio4 16". Because GPIO4A has 8 pins, and GPIO4B also has 8 pins, corresponding GPIO in the dts should be "gpio4 16", because GPIO4A has 8 pins, and GPIO4B also has 8 pins, and so on.

9.2.4 CPU, GPU, DDR Frequency Modification

DVFS (Dynamic Voltage and Frequency Scaling) is a real-time voltage and frequency scaling technique. The modules supporting DVFS in the current 4.4 kernel are CPU, GPU, DDR. CPUFreq is a set of modules defined by the kernel developers that support dynamic voltage and frequency scaling.
CPUFreq is a framework model to adjust the frequency and voltage of the CPU in a stateful manner, which can effectively reduce the power consumption of the CPU while taking into account the CPU performance. CPUFreq selects a suitable frequency for CPU usage by means of different inverting strategies, the current

version of the kernel provides the following strategies:

- interactive: Dynamically adjusts the frequency and voltage according to the CPU load;

- CONSERVATIVE: Conservative strategy, adjusting frequency and voltage step by step;
- ondemand: dynamically adjusts the frequency and voltage according to the CPU load.比 interactive strategy is slow to respond;
- Userspace: Voltage and frequency are set by yourself, the system will not adjust them automatically;• powersave: Power consumption is prioritized, always set the frequency at the lowest value;
- performance: Performance first, always set frequency to the highest value;

Please refer to `<SDK>/docs/Common/DVFS/`目 for detailed module functions and configurations. There are debugging interfaces for ARM/GPU/DDR, which can be operated by ADB commands, and the corresponding interface entries are listed below:

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

The following similar nodes are available under these books:

- available_frequencies: Displays supported frequencies.
- available_governors: Indicates supported frequency conversion strategies• cur_freq: Indicates current frequency
- governor: Displays the current inverter strategy.
- max_freq: Indicates the current maximum runtime frequency• min_freq: Indicates the current minimum runtime frequency

To perform frequency setting on an RK3399/RK3399pro GPU, proceed as follows:
Check which frequencies are supported:

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

Switching the inverter strategy:

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

Fixed Frequency:

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

9.2.5 Temperature Control Configuration

The ARM and GPU cores of the RK3399/RK3399Pro chipsets are equipped with temperature sensors to monitor the CPU and GPU temperatures in real time, and algorithms can be used to control CPU and GPU frequencies to control CPU and GPU temperatures. Since each product has a different hardware design and different molds, the temperature control parameters can be adjusted to suit the product by using the following configuration in dts.
Pins:

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

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

Set the temperature control upper limit temperature:

```
&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 the detailed description of temperature control, please refer to the relevant files under `<SDK>/docs/Common/THERMAL` project.

9.2.6 LPDDR4 Configuration

DDR configuration instructions can be found in the relevant files under `<SDK>/docs/Common/DDR` project.

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

```
&dfi {
    status= "okay";
};
&dmc {
    status= "okay";
    center-supply=  <&vdd_center>;//This in here needs to be configured by the
    customer according to the actual hardware circuit.
    upthreshold =< 40 ;>
    downdifferential =<
    20> ; system-status-
    freq = <
        /*system status freq(KHz)*/
        SYS_STATUS_NORMAL 856000
        SYS_STATUS_REBOOT 416000
        SYS_STATUS_SUSPEND 416000
        SYS_STATUS_VIDEO_1080P 416000
        sys_status_video_4k 856000
        sys_status_video_4k_10b 856000
        SYS_STATUS_PERFORMANCE 856000
        SYS_STATUS_BOOST 856000
        SYS_STATUS_DUALVIEW 856000
        SYS_STATUS_ISP 856000
    >;
    vop-pn-msch-readlatency =<
        /* plane_number readlatency */
        0  0
        4  0x20
    >;
    vop-bw-dmc-freq =<
        /* min_bw(MB/s) max_bw(MB/s)
        freq(KHz) */ 763 1893 416000
        3013 99999 856000 \
    >;
    auto-min-freq =< 0 ;>
};

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

```
    opp-hz= /bits/ 64 ;Â
    opp-microvolt =< 900000 ;>
};
opp-928000000 {
    opp-hz= /bits/ 64 ;Â
    opp-microvolt =<
    900000> ; status=
    "disabled" ;
};
opp-1056000000 {
    opp-hz= /bits/ 64 ;Â
    opp-microvolt =<
    900000> ; status=
    "disabled" ;
};
```

```
};
```

Here is what needs to be noted:

LPDDR4 only supports 416M and 856M, the other frequencies are disabled, so if the customer wants to use the same dts to support LPDDR4 and other types of DDR, other types of DDR will only have 416M and 856M, please make sure to configure the DDR frequency conversion function to be enabled by default. The LPDDR4 frequency conversion function limits the number of sound cards for the following reasons:

- If the LPDDR4 requires frequency conversion, the frequency buffer needs to be moved to the sram, and the RK3399Pro has limited sram space.
  用The current pre-allocated space for a single audio stream is 32k, so the maximum number of sound cards supported by the system is limited to two. (32k22, each soundcard contains both playback and recording), more soundcards can not be created successfully, except by reducing the pre-allocation of individual streams!大
  However, this reduces the maximum大 buffer size supported underneath, and there is a limit to setting a larger大 buffer when using the sound card with the lower layer. Note that USB sound cards are not supported because they do not use dma. In other words, it is possible to have two sound cards (including those with hdmi, spdif, i2s, etc.) and multiple usb sound cards.
- If LPDDR4 frequency conversion is required, the audio buffer needs to be moved to sram, and the system can only support up to 2 sound cards.

Configure as follows:

Adding a sram node to dts

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

iram nodes are used in the corresponding product dts.

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

- If LPDDR4 frequency conversion not needed, since LPDDR4 frequency conversion has a limit of 2 sound cards, if more than 3 sound cards are needed, you need to disable LPDDR4 frequency conversion, i.e. disable the dmc node in the dts of the corresponding product, as shown below:

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

Also, make sure that the following two configurations are removed from the kernel: Remove the following configurations from dts:

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

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

9.2.7 SD Card Configuration

For details of temperature control, please refer to the relevant files under

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

片比For some cores, such as the RK3326/RK3399PRO, the debug of the UART is used in conjunction with the sdcard, and the default configuration is to turn on debug.
用 The sdcard requires the following configuration:

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

9.3Recovery development

9.3.1 summary

The Recovery mechanism was developed similar to the Android Recovery function.

The main purpose is to erase user data and upgrade the system.

Recovery mode in Linux is a recovery partition on the device, which consists of kernel+resource+ramdisk and is mainly用 used for upgrading operations. u-boot determines whether the system to be booted is a Normal system or a Recovery system based on the fields stored in the misc partition.
Recovery Mode. Due to the unique nature of the system, Recovery Mode ensures the integrity of the upgrade, i.e. the upgrade can continue to be performed even if the upgrade process is interrupted, e.g. by an abnormal power loss.

9.3.2 adjust components during testing

The common debugging technique is to turn on debug.

buildroot/output/rockchip_xxx_recovery/target Create a hidden file.rkdebug, under the destination.

```
touch .rkdebug
```

The log of the upgrade in Recovery mode is printed out on the serial port. Another way is to check the userdata/recovery/Log file to see if there are any files be upgraded recovery mode.
For more Recovery development information, please refer to file /docs/Linux/Recovery/Rockchip_Developer_Guide_Linux_Recovery_CN.pdf.

9.4Buildroot Development

Rockchip has already configured bad variables, BSP configuration and development of each module for easy customer development and customization.

9.4.1 Setting of bad situation variables

```
source . /envsetup.sh (config_name)
```

config_name can source ./envsetup.sh is listed, choose a specific platform to compile.

9.4.2 Compile modules and systems

After choosing the compilation platform, you can compile each package, which consists of three parts: config, build, and install.

```
make <package>-
reconfigure make
<package>-rebuild make
<package>-reinstall
```

The cleanup package command is as follows:

```
make <package>-dirclean
```

Compile the Buildroot system, just make it!

```
make
```

9.4.3 Developing related modules

For package developme[illegible]/buildroot/package/* , where package/rockchip is the package developed by Rockchip.

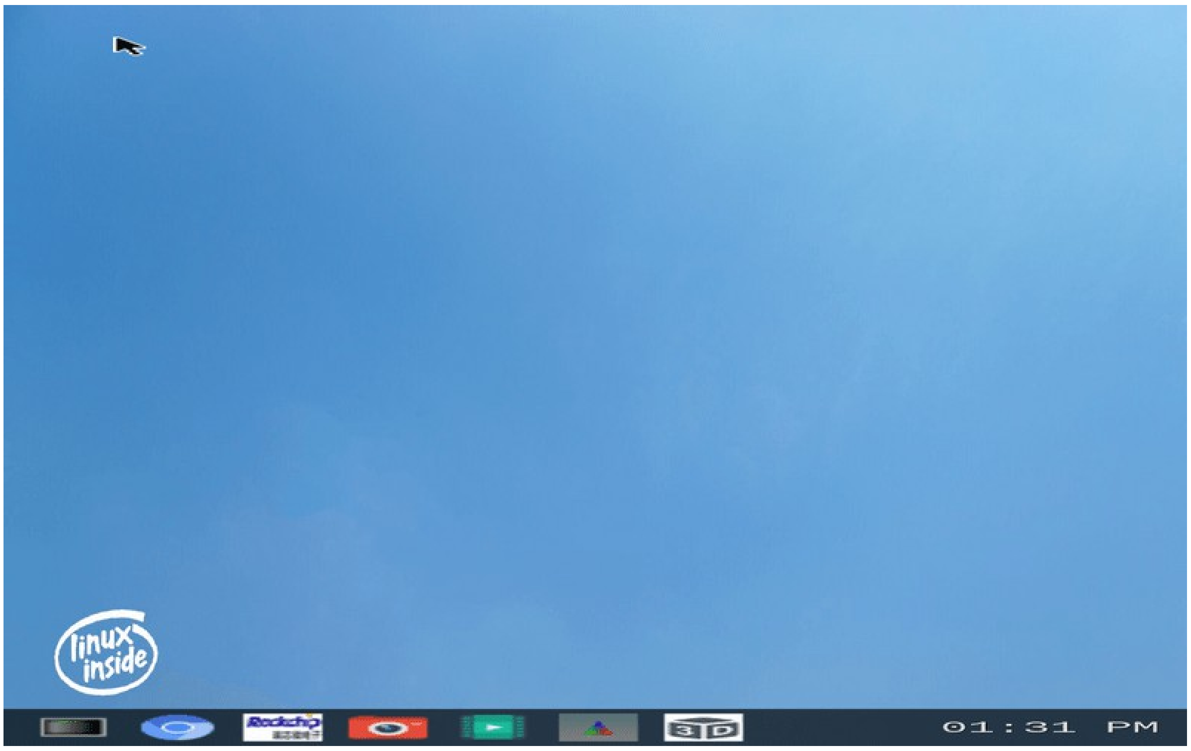
9.4.4 Customized Related Modules

Refer to /buildroot/configs/rockchip* for configuration switches, and each module can be customized.

```
buildroot$ tree configs/rockchip/
├── audio.config
├── audio_gst.config
├── base.config
├── base_extra.config
├── benchmark.config
├── bt.config
├── busybox.config
├── busybox_minimal.config
├── camera.config
├── camera_gst.config
├── chromium.config
├── debug.config
├── gdb.config
├── gpu.config
├── gpu_dummy.config
├── kernel.config
├── network.config
├── network_only_eth.config
├── npu2.config
├── ntfs.config
├── powermanager.config
├── recovery.config
├── recovery_base_nvr.config
├── rk3308.config
├── rk3308_aarch64.config
├── rk3308_arm.config
├── rk3326.config
├── rk3326_aarch64.config
├── rk3326_arm.config
├── rk3399.config
├── rk3399_aarch64.config
├── rk356x.config
├── rk356x_aarch64.config
├── rk356x_arm.config
├── rk3588.config
├── rk3588_aarch64.config
├── rknn_demo.conf
├── tee_aarch64_v2.config
├── test.config
├── tinyrootfs.config
├── video_gst.config
├── video_gst_rtsp.config
├── video_mpp.config
├── weston.config
└── yx.config
...
```

9.4.5 Tabletop Use

Buildroot supports the Weston desktop and some demos by default, as shown below:


linux inside
Rockchip
01:31 PM

These Weston applications provide some basic functionality, such as Termial, Launchers configuration, Chromium browser, camera preview, multiplexed video, GPU, 鼠 markers, and other demos. Additional demos can be added via the /etc/xdg/weston/weston.ini.d/ 02-desktop.ini configuration can be added.

> Note: Since third-party UI requires commercial license, there is a risk of infringement, if you want to enable Buildroot QT/Enlightenment/Minigui UI support, Rockchip official does not support it, and you need to get the corresponding third-party authorization.

9.4.6 Usage and Password

用戸: root
Password: rockchip

9.4.7 Weston Development

Weston is the official reference implementation of Wayland's open source display protocol, and the Rockchip Buildroot SDK display service uses Weston by default. 10.0.0 drm backend.

There are several ways to configure

Weston in the Buildroot SDK: a.

Startup Parameters

That is, the parameters of the command to start Weston, such as weston --tty=2,

located in /etc/init.d/S49weston, corresponding to the SDK code in the location: buildroot/package/weston/S49weston

b. weston.ini configuration file

The .ini files located under /etc/xdg/weston/weston.ini and /etc/xdg/weston/weston.ini.d/ are

located as follows in the corresponding SDK code: buildroot/board/rockchip/common/base/etc/xdg/weston/weston.ini

Reference:

https://fossies.org/linux/weston/man/weston.ini.ma

n c. Special Environment Variables

This environment variable is usually set in /etc/profile.d/weston.sh, which is located in the corresponding SDK code: buildroot/package/weston/west

on.sh. d. Dynamically

Configure Files

Weston in the Buildroot SDK provides some dynamic configuration support for the drm backend display function, with the default path being for /tmp/.weston_drm.conf, which can be specified via the environment variable WESTON_DRM_CONFIG.

e. udev rules

Some of the configurations of the input

devices in Weston require udev rules,

which can be found in the Weston

development files

```
<SDK>/docs/Linux/Graphics/Rockchip_Developer_Guide_Buildroot_Westo
n_CN.pdf
```

9.5Debian Development

Rockchip offers support for Debian 10/11, based on the X11 display architecture. The system is based on the Linaro version. Some support for graphics and video acceleration has been added. It includes packages such as libmali, xserver, gstreamer rockchip, etc. These packages are built with docker to compile the relevant deb packages, which are stored in /debian/packages/*.

```
<SDK>/docs/Linux/ApplicationNote/Rockchip_Developer_Guide_Debian_Dock
er_EN.pdf
```

Debian Development File Reference

```
<SDK>/docs/Linux/ApplicationNote/Rockchip_Developer_Guide_Debia
n_CN.pdf
```

9.6Yocto Development

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

9.7multimedia development

Multimedia development on the rockchip platform via gstreamer/rockit

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

vpu_service: driver
mpp: video codec middleware for rockchip
platform, please refer to mpp file
gstreamer/rockit: component to connect
with apps.
```

So far, the complete solution provided by rockchip linux with SDK is based on gstreamer, and the advantage of using gstreamer is that it is relatively convenient to use the complete player based on pipeline, encoder. For customized development based on rockit, please refer to the release of rockit.
File.

Specific information references:
/docs/Linux/Multimedia
├── Rockchip_Developer_Guide_Linux_RKADK_CN.pdf
├── Rockchip_Developer_Guide_MPP_CN.pdf
├── Rockchip_Developer_Guide_MPP_EN.pdf
├── Rockchip_Developer_Guide_RGA_CN.pdf
├── Rockchip_Developer_Guide_RGA_EN.pdf
├── Rockchip_FAQ_RGA_CN.pdf
├── Rockchip_FAQ_RGA_EN.pdf
├── Rockchip_Introduction_Linux_Audio_3A_Algorithm_CN.pdf
├── Rockchip_Introduction_Linux_Audio_3A_Algorithm_EN.pdf
├── Rockchip_User_Guide_Linux_Gstreamer_CN.pdf
├── Rockchip_User_Guide_Linux_Gstreamer_EN.pdf
└── Rockchip_User_Guide_Linux_Rockit_CN.pdf

9.8Grahpics Development

The Rockchip Linux platform Graphics is an ARM Linux platform that utilizes DRM and DMA-BUF. The advantage is that the common architecture makes it easier to customize development based on this architecture, and it is possible to use many existing components. Many basic open-source projects have started to be developed based on the Rockchip platform.

The platform is adapted to the ARM side. The disadvantage is that many people don't understand the contents of this document, and it takes a learning process to use it in practice. For more information, please refer to the Rockchip wiki and the following file.

```
<SDK>/docs/Linux/Graphics/
├── Rockchip_Developer_Guide_Buildroot_Weston_CN.pdf
├── Rockchip_Developer_Guide_Buildroot_Weston_EN.pdf
├── Rockchip_Developer_Guide_Linux_Graphics_CN.pdf
└── Rockchip_Developer_Guide_Linux_Graphics_EN.pdf
```

9.9Application Development

The SDK is commonly used for Weston, EFL, ROS, and other applications. Refer to the file under /docs/Linux/Graphics/ApplicationNote.

9.10Security mechanism development

Refer to `<SDK>/docs/Linux/Security` Objects file.

9.10.1 Secureboot

Secureboot is not enabled by default. To verify this, proceed as follows:

- BoardConfig adds security-related configuration

```
diff --git a/rk356x/BoardConfig-rk3568-evb1-ddr4-v10.mk
b/rk356x/BoardConfig- rk3568-evb1-ddr4-v10.mk
index 6282827...6d050d7 100644
--- a/rk356x/BoardConfig-rk3568-evb1-ddr4-v10.mk
+++ b/rk356x/BoardConfig-rk3568-evb1-ddr4-
v10.mk @@ -60,3 +60,6 @@ export
RK_MISC=wipe_all-misc.img
 export RK_DISTRO_MODULE=
 # Define pre-build script for this
 board export
 RK_BOARD_PRE_BUILD_SCRIPT=app-build.sh

+REALDIR=`dirname $(readlink -f ${BASH_SOURCE[0]})`
+source ${REALDIR}/BoardConfig-security-base.mk
```

- The above steps turn on secure encryption, and if encryption is
- enabled, then dmv checksums are executed (the encryption process uses dmv checksums). `/build.sh.` Follow the instructions for compilation errors step by step.

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

$ . /build.sh createkeys

$ . /build.sh

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

Write your `PC root`'s password to `` `u-boot/keys/root_passwd` ``.

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


    CONFIG_BLK_DEV_DM
    CONFIG_DM_CRYPT
    CONFIG_BLK_DEV_CRYPTOL
    OOP CONFIG_DM_VERITY
    CONFIG_TEE
    CONFIG_OPTEE
```

Kernel Add Related Configurations

```
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_defcon
fig @@ -590,3 +590,9 @@
CONFIG_RCU_CPU_STALL_TIMEOUT=60
 CONFIG_FUNCTION_TRACER=y
 CONFIG_BLK_DEV_IO_TRACE=y
 CONFIG_LKDTM=y
+CONFIG_BLK_DEV_DM=y
+CONFIG_DM_CRYPT=y
+CONFIG_BLK_DEV_CRYPTOLOOP=y
+CONFIG_DM_VERITY=y
+CONFIG_TEE=y
+CONFIG_OPTEE=y

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


    CONFIG_FIT_SIGNATURE
    CONFIG_SPL_FIT_SIGNATURE
```

`u-boot` adds the relevant configuration.

```
u-boot$ git diff
diff --git a/configs/rk3568_defconfig
b/configs/rk3568_defconfig index fbd9820acc..6efdaac1d6
100644
```

```
--- a/configs/rk3568_defconfig
+++ b/configs/rk3568_defconfig
@@ -220,3 +220,5 @@ CONFIG_RK_AVB_LIBAVB_USER=y
 CONFIG_OPTEE_CLIENT=y
 CONFIG_OPTEE_V2=y
 CONFIG_OPTEE_ALWAYS_USE_SECURITY_PARTITION=y
+CONFIG_FIT_SIGNATURE=y
+CONFIG_SPL_FIT_SIGNATURE=y
```

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

`Buildroot` adds the relevant configuration.

```
buildroot$ git diff
diff --git
a/configs/rockchip_rk3568_defconfig
b/configs/rockchip_rk3568_defconfig
index 6cdd795c64..5244694c7d 100644
--- a/configs/rockchip_rk3568_defconfig
+++
b/configs/rockchip_rk3568_defconfi
g @@ -23,3 +23,4 @@
 BR2_PACKAGE_RKWIFIBT_AP6398S=y
 BR2_PACKAGE_RKWIFIBT_BTUART="ttyS8"
 BR2_PACKAGE_RKNPU2=y
+BR2_ROOTFS_OVERLAY="board/rockchip/common/security-system-overlay"
```

Environment Installation.

If the compiler reports an error `No module named Crypto.Signature`, this is due to the fact that `the python` algorithm library is not installed on the development computer, and the following command should be executed:

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

2. If the following error occurs: `ModuleNotFoundError: No module named 'C[illegible]`

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

- For verification, look for a machine that has not burned rpmb. If the machine里面 has an encryption key, it will use the encryption key in the machine first.
- key, if the machine encryption key is different from our compiled encryption key, the startup will fail

The encryption key is placed in the rpmb area and can be erased repeatedly. If the system cannot be loaded because of a different key, you can set the `buildroot/board/rockchip/common/security-ramdisk-overlay/init`. in FORCE_KEY_WRITE=true turn on to force key update

Specific reference can be made to

- `<SDK>/docs/Linux/Security/Rockchip_Developer_Guide_Linux_Secure_Boot_CN.pdf`
- TEE Development Guide `<SDK>/docs/Linux/Security/Rockchip_Developer_Guide_TEE_SDK_CN`

  Linux secureboot Software Development Guide

9.11 WIFI/BT Development

References to files under /docs/Linux/Wifibt

10. SDK Testing

10.1 Integration with Rockchip's press-strength test scripts

rockchip_test integrates functionality, strength, and performance related testing

```
root@linaro-alip:~# /rockchip-
test/rockchip_test.sh
cpu                 1 (cpufreq stresstest)
test.
                    2 (memtester &
ddr                   stressapptest)
test.
```

```
gpu test.           3 (use glmark2)
npu test.           4 (npu2:rk3588)
auto reboot test.   5 (reboot
tests) suspend_resume test: 6
(suspend & resume) nand power lost
test: 7 (S5 stress tests) flash
stress test: 8 (flash tests) audio
test.               8 (flash tests)
audio test.         9 (audio tests)
recovery test: 10 (default wipe all)
10 (default wipe all) bluetooth test.
11 (bluetooth on&off test) wifi test.
12 (wifi on&off test) ethernet test.
13 (ethernet tests) camera test.  14
(use rkaiq_demo)
video test.         15 (use gstreamer-wayland and
app_demo) chromium test: 16 (chromium with video
hardware acceleration) hardware infomation: 17 (to get
the hardware infomation)    16 (chromium with video
hardware acceleration) hardware infomation: 17 (to get
the hardware infomation)

***********************************************
*** please input your test moudle.
```

10.2 Benchmark Test

Reference data for some commonly used benchmarks, which are located in the test file:

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

10.3 Rockchip Modules and Power Testing

This file provides some common module functions and methods of testing for force majeure:

```
<SDK>/docs/Linux/Profile/Rockchip_User_Guide_Linux_Software_Test_CN.pd
f
```

11. SDK Debugging

11.1 ADB Tools

11.1.1 summarize

- Execute the shell
- (command) of the device to manage the port mapping of the emulator or device.
- Uploading/Downloading
- Files Between Computers and Devices Installing Local Software至 Debian Devices
- ADB is a "client-server" program, where the client is mainly a PC and the server is a physical or virtual Debian device. Depending on how the PC connects to the Debian device, ADB can be divided into two categories: Network ADB: The host computer connects to the STB device over a wired/wireless network (same local area 网) USB ADB: The host computer connects to the STB device via a USB cable.

11.1.2 USB adb User Guide

The following restrictions apply to the use of USB adb:

- Supports USB OTG port only
- Simultaneous use of multiple clients is not supported.
- Only one device is supported to be connected to the host computer, and multiple devices are not supported:
  To test for a successful connection, run the "adb devices" command, and if the serial number of the machine is displayed, the connection is successful.

11.2 The system log information is obtained automatically.

The log messages are automatically fetched to the /info project, and the main log messages are as follows.

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

11.3 Copyright Testing Tools

11.3.1 Buildroot

```
make legal-info
```

11.3.2 Debian

Tests can be referenced to official tools

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

The copyright notice as `/usr/share/doc/*/copyright` package is located at

11.3.3 Yocto

The copyright notice as `build/tmp/deploy/licenses/*/recipeinfo` ocated at

12. expand one's financial resources

12.1 github

Rockchip code repository open source rockchip-github.

12.2 wiki

Rockchip information open source rockchip-wiki

12.3 upstream

- Rockchip upstream uboot.

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

Upstream U-Boot support Rockchip SoCs.

```
rk3036, rk3188, rk3288, rk3328, rk3399,
rk3566, rk3568
```

- Rockchip upstream kernel

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

Mainline kernel supports.

`rv1108, rk3036, rk3066, rk3188, rk3228, rk3288, rk3368, rk3399, rk3566, rk3568`

13. Frequently Asked Questions (FAQ)

13.1 How to confirm the current SDK version and system/kernel version?

The information can be obtained via /info/目錄 or ' /etc/os-release ',比 for example

```
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_defco
nfig" KERNEL="5.10 - rockchip_linux_defconfig"
```

13.2 SDK compilation related issues

13.2.1 Repo causes synchronization problems

`repo sync -c` Mention示`No module named formatter` when updating this because you need to use a newer version of python on your host.
比If python 3.8+ completely removes the formatter, and the repo version of the external SDK is too old, you can only update the repo version to比 as shown below.
Solution by Formula

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

This supports repo version 2022, the default master support is 2020. Alternatively, add `--repo- rev=stable` to the repo init task to switch to the new repo.

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

13.2.2 Buildroot compilation issues

If the build of buildroot fails due to some web reasons, the following methods can be used to solve the problem. The following method can be used to solve the problem:

Pre-built dl entries, dl is a pre-compiled package that can be integrated into the buildroot in advance. This reduces download time and increases compilation efficiency.

比For example, rk3588 linux sdk can be modified this way to increase the dl destination:

```
$ cd .repo/manifests
$ Make the following changes to the rk3588_linux_release.xml file:

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.xm
l @@ -14,6 +14,7 @@
```

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

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

13.3Debian related issues

13.3.1 Encountering "noexec or nodev" problem

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

Solution:

```
mount -o remount,exec,dev xxx
(where xxx is the path to the workbook, then recompile)
```

If any other compilation exceptions are encountered, first rule out the use of the ext2/ext4 compilation system.

13.3.2 Failed to download "Base Debian".

• Since compiling Base Debian requires access to a foreign web site, the download often fails when accessing a foreign web site from a domestic web site: Debian uses live build, and can be configured this way if you change the mirror source to a domestic one:

32-bit systems:

```
+++ 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 systems:

```
  --- 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 you are unable to download the package for other reasons, you can share the pre-compiled package on your Baidu Cloud web disk and perform the next step directly from the current destination.

13.3.3 Abnormal operation causes **mount/dev** error problem

比If this askpass command or cannot use one

The cause may be the frequent abnormal operation (CTRL+C) during the compilation process, which leads to the error on the upper screen, and can be repaired by the following methods:

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

13.3.4 Multiple mounts cause **/dev** error problem

比If this occ

```
sudo: unable to allocate pty: No
such device
```

The reason may be that the compilation process mounts multiple times, which leads to the error on the upper screen, and can be fixed by the following way:

```
ssh < 用户名>@<IP地址> -T sudo -S umount /dev -l
```

13.3.5 How to view system related information

13.3.5.1 How do I check the **Debian** version of my system?

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

13.3.5.2 How to check whether

Debian is displayed with **X11** or **Wayland**

on an X11 system:

```
$ echo
$XDG_SESSION_TYPE
x11
```

on the Wayland system:

```
$ echo
$XDG_SESSION_TYPE
wayland
```

13.3.5.3 How to view the system partition

```
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        Name     Flag
                            system               s
 1      8389kB       4194k              uboot
 12.6MB
                     B                  misc
 2      12.6MB
                     4194k              boot
 16.8MB
                     B
                                        recover
 3      16.8MB
                     67.1M
 83.9MB                     ext         y
                     B
 4      83.9MB              4           backup
                     134MB
 218MB
                            ext         rootfs
                     33.6M
 5      218MB
                     B      2           oem
 252MB
                     15.0G  ext         userda
 6      252MB
                     B      2           ta
 15.3GB
                     134MB
 7      15.3GB
                     15.6G
 15.4GB
                     B
 8      15.6GB
 31.3GB
```

13.3.5.4 System with ssh.service service exception

This is a problem that existed with Debian 10 or earlier /etc/rc.local Add the following:

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

exit 0
```

13.3.6 Debian11 base package does not compile

An error like the following is encountered

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

The main requirement is that the PC's kernel version is 5.10+, which is a bug that exists in older QEMUs. There are two main solutions:

- The kernel version that comes with the PC should be sufficient for 5.10+. Checking the PC kernel version

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

- Update the system qemu reference qemu.

13.3.7 How to Unpack, Modify, and Repackage Debian deb Packages

If you want to modify and repackage the original deb, do it as follows.

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

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

#Modify File xxx

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

13.3.8 How to increase swap partition in Debian

When the physical memory of the system is insufficient, Debian can add a swap virtual memory partition for use by the currently running program.
比For example, to create a virtual memory of 2G.

- Creating a swap file

```
cd /opt
mkdir
swap
dd if=/dev/zero of=swapfile bs=1024
count=2000000 # count
```

stands for size, in this case 2G.

- Convert file to swap file

```
sudo mkswap swapfile
```

- Activate swap file

```
swapon /opt/swapfile
```

Uninstallation.

```
swapoff /opt/swapfile
```

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

- Verification of Vitality

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

13.3.9 Debian will restart the service on the一 update.

比 Debian is used to install different packages for different chipsets (片 , /etc/init.d/rockchip.sh) at the first startup, such as libmali isp and other packages. After installing these packages, the service is restarted. If the project is a proprietary one, you can handle the differences when you make the image.

13.3.10 Error in Debian with libGL related dri.so call.

The explanations are mainly as follows.

- EGL is an extension of OpenGL for the x window system on the ARM platform, and is equivalent to the glx library on the x86 platform.
- Since Xorg uses Driver modesettings that load libglx.so by default (disabling glx can cause some applications to fail by detecting the glx environment), libglx.so searches for dri libraries in the system. However, Xorg 2D acceleration is directly based on the DRM implementation and does not implement the dri library, so libglx.so reports the following error during startup.

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

This has no effect on system operation and does not need to be handled.

For the same reason, the following error may be reported during the startup of some applications, which should not be handled and will not affect the operation of the application.

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

13.3.11 How to confirm that the hardware icon layer is used in Debian?

- The kernel dts is configured to look like 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 tests the layer for uploading

```
10     0      0      0,           0,     0           0x000000
                     0            0                  02
2 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 the summary to see if the hard icon layer is used.

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

If you have both steps 1 and 2, and still have problems, then check /var/log/drm-cursor.log for any exceptions.

13.4 Linux Video Related Questions

13.4.1 Playing video stutters and Jiji gets a frame loss error, how do I fix it?

The high frame rate can be confirmed using fpsdisplaysink, and if the high frame rate is close to the desired frame rate, the high frame rate can be confirmed by specifying Sync=false to solve the problem, some platforms can turn on AFBC. lagging can also be solved by looking at the hardware overhead time, echo . > /sys/module/rk_vcodec/parameters/mpp_dev_debug

13.4.2 gst-launch-1.0 Follow the camera video preview command.

The v4l2src plug-in can be used, e.g. gst-launch-1.0 v4l2src ! autovideosink

13.4.3 The screen flickers after turning on AFBC.

Frame flickering is usually due to insufficient ddr bandwidth, so try the fixed performance mode. In addition, because of the way the display hardware is implemented, vertical scaling requires relatively high performance and bandwidth, and may be insufficient, so you need to use other scaling methods (e.g., GPIO, GPIO, GPIO, GPIO, GPIO). rga/gpu)

13.4.4 Is the Gstreamer framework buffer zero-copy?

Zero-copy data processing using dmabuf-related interface

13.4.5 gst-launch-1.0 How can I test the performance of decoding the most high?

To set the performance mode, check the frame rate using the official fpsdisplaysink, and check the driver frame rate using the debugging interface of the driver.

13.4.6 How to solve the problem of screen flickering and water ripples during playback?

Jitter is usually indicative of underperforming hardware, mostly due to insufficient ddr bandwidth, and can be detected by fixing the highest frequency in ddr performance mode.

13.4.7 How can Gstreamer quickly put in opengles?

Generally, the synthesis is supported by the gl plug-in, and the delivery is supported by a third-party display service, which can be performed internally.
opengles Synthesis (e.g. weston)

13.5 Third-best OS Porting Issues

13.5.1 Is there any introduction to Kirin system porting, that is, downloading the standard iso image and extracting rootfs.squafs for porting?

You can refer to the porting file in the SDK, which is also applicable to Kirin Os, but Kirin Os is relatively closed, so for better results, you can find Kirin's mirror of the RK platform.

13.5.2 Adapted to which domestic OS

Unicorn and Kirin have been adapted. In addition, Hongmeng community is currently working on the adaptation of RK3588, RK356X and RK3399. You can also consult our business or follow the open source community of Hongmeng.

13.5.3 Whether or not it supports UEFI boot.

RK3588 supports UEFI as a priority.

13.6 Display Related Issues

13.6.1 How to make the video send to the video layer

drm has an interface to query the type of the plane. see gstreamer's kmssink method for details.

13.6.2 wayland How to configure the position of each screen in Multi-screen Heterodyne mode,比 e.g. left/right or up/down position.

The weston iso-display only supports left-right alignment, which is the order in which the screen is loaded.
/docs/Linux/*/Rockchip_Developer_Guide_Buildroot_Weston_CN.pdf 2.9 Multi-screen configuration

13.6.3 What is the Debian xserver version?

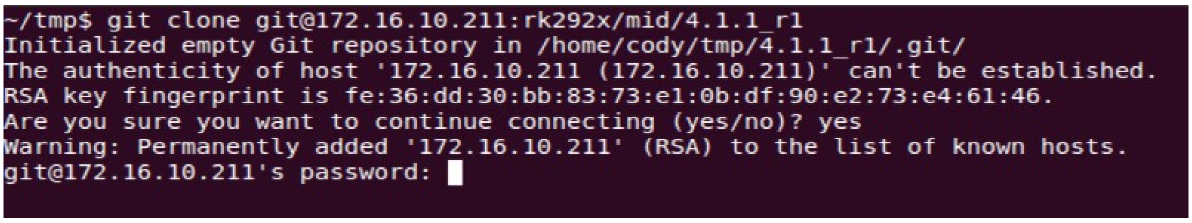
Debian10 uses xserver1.20.4, Debian11 uses xserver1.20.11.

14. SSH Public Key Operating Instructions

Please follow the instructions in the file /docs/Others/Rockchip_User_Guide_SDK_Application_And_Synchronization_CN.pdf. Create an SSH public key and send an email tofae@rock-chips.com to request SDK code. The file will be released to the customer during the license request process.

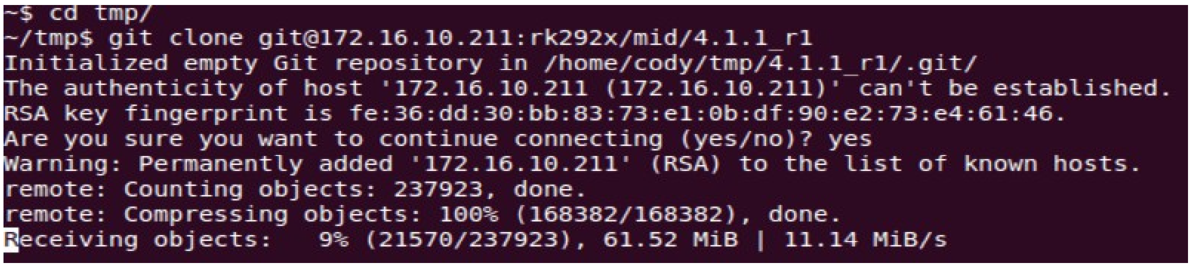
14.1 Multiple Machines Using the Same SSH Public Key

To use it on a different machine, copy your SSH private key file id_rsa to "~/.ssh/id_rsa" on the machine you want to use.
When using the wrong private key, the following message will be displayed, so 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 the 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
```

The following error may be displayed when adding an SSH private key.

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

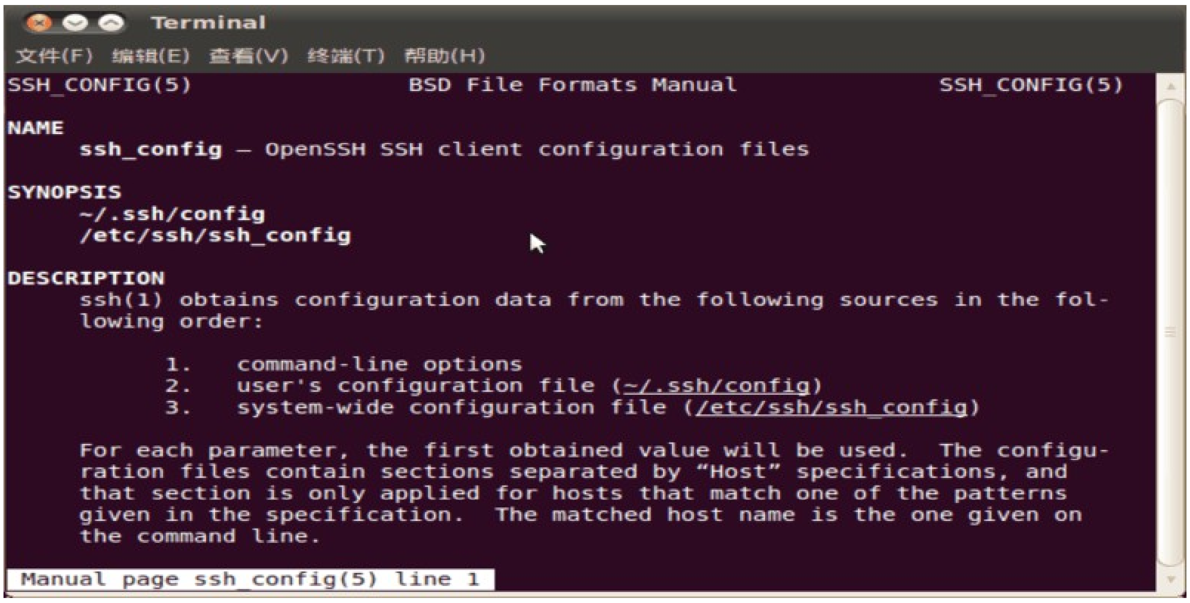
This can be solved by entering the following command in console.

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

14.2 Switching Different SSH Public Keys on One Machine

You can refer to ssh_config file to configure SSH.

```
~$ man ssh_config
```

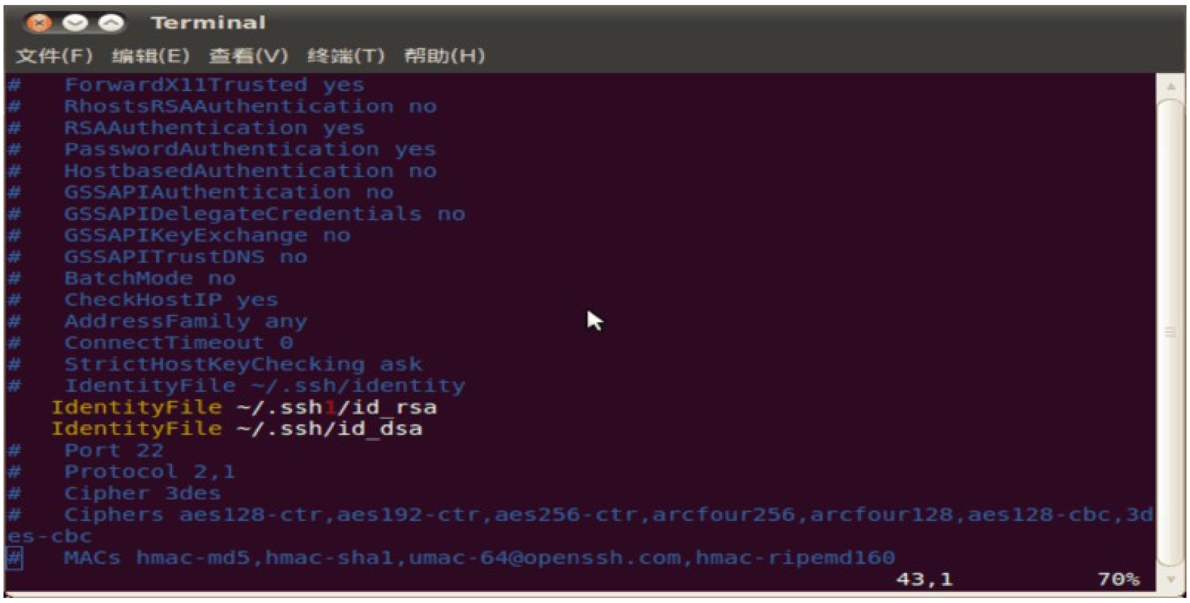


```
SSH_CONFIG(5)               BSD File Formats Manual               SSH_CONFIG(5)

NAME
     ssh_config — OpenSSH SSH client configuration files

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

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

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

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

 Manual page ssh_config(5) line 1
```

Configure the current SSH configuration with the following command.

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

As shown in the figure, use another destination file "~/.ssh1/id_rsa" for SSH as the authentication private key. In this way, you can switch between different keys.



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

14.3 Key authority management

The server can monitor the number of downloads and IP address of a key in real time, and disable the download privileges of the corresponding key if any abnormality is detected.
Keep the private key file in a safe place. Do not authorize二 to be used by a third party.

14.4reference file

For more detailed instructions, refer to文
File/docs/Others/Rockchip_User_Guide_SDK_Application_And_Synchronization_CN.pdf.