Rockchip RK3358 Linux SDK Quick Start

ID: RK-JC-YF-360

Release Version: V1.8.0

Release Date: 2022-06-20

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

DISCLAIMER

Trademark Statement

Rockchip Electronics Co., Ltd.

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

Website: www.rock-chips.com

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

Customer service Fax: +86-591-83951833

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

Preface

Overview

The document presents Rockchip RK3358 Linux SDK release notes, aiming to help engineers get started with RK3358 Linux SDK development and debugging faster.

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers

Software development engineers

Chipset and System Support

Chipset	Buildroot	Debian	Yocto
RK3358M/RK3358J	2018.02-rc3	N/A	N/A

Revision History

Date	Version	Author	Revision History
2021-12-30	V1.0.0	WJL	Initial version.
2022-06-20	V1.8.0	WJL	Update version to V1.8.0

Contents

Rockchip RK3358 Linux SDK Quick Start

1. Set up an Development Environment
2. Software Development Guide
   - 2.1 Development Guide
   - 2.2 Software Update History
3. Hardware Development Guide
4. SDK Building Introduction
   - 4.1 SDK Project Directory Introduction
   - 4.2 SDK Board Level Configuration
   - 4.3 Compilation Commands
   - 4.4 Automatic Build
   - 4.5 Build and package each module
     - 4.5.1 U-boot Build
     - 4.5.2 Kernel Build
     - 4.5.3 Recovery Build
     - 4.5.4 Buildroot Build
     - 4.5.5 Cross-compilation
       - 4.5.5.1 Cross-compilation inside SDK
       - 4.5.5.2 Cross-complilation of Buildroot
       - 4.5.5.3 Build Modules in Buildroot
     - 4.5.6 Firmware Package
5. Upgrade Introduciton
   - 5.1 Windows Upgrade Introduction
   - 5.2 Linux Upgrade Instruction
   - 5.3 System Partition Introduction
6. Fast start
7. OTP
8. RK3358 SDK Firmware

1. Set up an Development Environment

It is recommended to use Ubuntu 20.04 for compilation. Other Linux versions may need to adjust the software package accordingly. In addition to the system requirements, there are other hardware and software requirements. Hardware requirements: 64-bit system, hard disk space should be greater than 40G. If you do multiple builds, you will need more hard drive space

Software requirements: Ubuntu 20.04 system:

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

```
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
```

It is recommended to use Ubuntu 20.04 system or higher version for development. If you encounter an error during compilation, you can check the error message and install the corresponding software packages accordingly.

2. Software Development Guide

2.1 Development Guide

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

2.2 Software Update History

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

```
.repo/manifests$ realpath rk3358_linux_release.xml
# e.g.:printf version v1.8.0, update time on 20220620
# <SDK>/.repo/manifests/rk3358_linux_release_v1.8.0_20220620.xml
```

Software release version updated information can be checked through the project text file by the following command:

```
.repo/manifests$ cat RK3358_Linux_SDK_Release_Note.md
```

Or refer to the project directory:

```
<SDK>/docs/RK3358/RK3358_Linux_SDK_Release_Note.md
```

3. Hardware Development Guide

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

RK3358 hardware design guide:

```
<SDK>/docs/RK3358/Hardware/Rockchip_RK3358J_Hardware_Design_Guide_V1.0_EN.pdf
```

RK3358 EVB hardware development guide:

```
<SDK>/docs/RK3358/Hardware/Rockchip_RK3358J_User_Manual_EVB_V1.0_EN.pdf
```

4. SDK Building Introduction

4.1 SDK Project Directory Introduction

There are buildroot, , recovery, app, kernel, u-boot, device, docs, external and other directories in the project directory. Each directory or its sub-directories will correspond to a git project, and the commit should be done in the respective directory.

- app: store application APPs with Demo.
- buildroot: root file system based on Buildroot (2018.02-rc3).
- device/rockchip: store board-level configuration for each chip and some scripts and prepared files for building and packaging firmware.
- docs: stores development guides, platform support lists, tool usage, Linux development guides, and so on.
- IMAGE: stores building time, XML, patch and firmware directory for each building.
- external: stores some third-party libraries, including audio, video, network, recovery and so on.
- kernel: stores kernel4.4/4.19 development code.
- prebuilts: stores cross-building toolchain.
- rkbin: stores Rockchip Binary and tools.
- rockdev: stores building output firmware.
- tools: stores some commonly used tools under Linux and Windows system.
- u-boot: store U-Boot code developed based on v2017.09 version.

4.2 SDK Board Level Configuration

Enter the project `<SDK>/device/rockchip/RK3358` directory:

Board level configuration	Note
BoardConfig-rk3358-evb-ddr3-v10.mk	Suitable for RK3358 EVB V10 development board with DDR3
BoardConfig-rk3358m-vehicle-ddr3.mk	Suitable for RK3358 Vehicle development board with DDR3
BoardConfig.mk	Default

The first way:

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

Select the board configuration of **RK3358 EVB V10 development board with DDR3**:

```
./build.sh device/rockchip/rk3358/BoardConfig-rk3358-evb-ddr3-v10.mk
```

Select the board configuration of **RK3358 Vehicle development board with DDR3**:

```
./build.sh device/rockchip/rk3358/BoardConfig-rk3358m-vehicle-ddr3.mk
```

The second way:

```
rk3358$ ./build.sh lunch
processing option: lunch

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

0. default BoardConfig.mk
1. BoardConfig-rk3358-evb-ddr3-v10.mk
2. BoardConfig-rk3358m-vehicle-ddr3.mk
3. BoardConfig.mk
Which would you like? [0]:
...
```

4.3 Compilation Commands

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

```
rk3358$ ./build.sh -h
Usage: build.sh [OPTIONS]
Available options:
BoardConfig*.mk    -switch to specified board config
lunch              -list current SDK boards and switch to specified board config
uboot              -build uboot
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 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/<pkg>          -build packages in the dir of app/*
external/<pkg>     -build packages in the dir of external/*

createkeys         -create secureboot root keys
security_rootfs    -build rootfs and some relevant images with security paramter
(just for dm-v)
security_boot      -build boot with security paramter
security_uboot     -build uboot with security paramter
security_recovery  -build recovery with security paramter
security_check     -check security paramter if it's good

Default option is 'allsave'.
```

View detailed build commands for some modules, for example: `./build.sh -h kernel`

```
rk3358$ ./build.sh -h kernel
###Current SDK Default [ kernel ] Build Command###
cd kernel
make ARCH=arm64 px30_linux_defconfig rk3358_linux.config
make ARCH=arm64 rk3358-evb-ddr3-v10-linux.img -j12
```

4.4 Automatic Build

Enter root directory of project directory and execute the following commands to automatically complete all build:

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

./build.sh     # Base on ./build.sh all
               # 1. Add firmware package ./mkfirmware.sh
               # 2. update.img package
               # 3. Copy the firmware in the rockdev directory to the
IMAGE/***_RELEASE_TEST/IMAGES directory
               # 4. Save the patches of each module to the
IMAGE/***_RELEASE_TEST/PATCHES directory
               # Note: ./build.sh  and  ./build.sh allsave command are the same
```

4.5 Build and package each module

4.5.1 U-boot Build

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

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

4.5.2 Kernel Build

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

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

4.5.3 Recovery Build

```
### Recovery build command
./build.sh recovery

### To view the detailed Recovery build command
./build.sh -h recovery
```

Note: Recovery is a unnecessary function, some board configuration will not be set

4.5.4 Buildroot Build

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

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

After build, rootfs.ext4 is generated in Buildroot directory “output/rockchip_chipset/images”.

4.5.5 Cross-compilation

4.5.5.1 Cross-compilation inside SDK

SDK prebuilts directory preset cross-compilation as follows:

Directory	Introdution
prebuilts/gcc/linux-x86/aarch64/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu	gcc arm 6.3.1 64bit toolchain
prebuilts/gcc/linux-x86/arm/gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf	gcc arm 6.3.1 32bit toolchain

4.5.5.2 Cross-complilation of Buildroot

If you need to compile individual modules or third-party applications, you need to configure the cross-compilation environment. For example, RK3358, whose cross-compilation tool is located in the `buildroot/output/rockchip_rk3358/host/usr` directory, needs to set the bin/ directory of the tool and the `aarch64-buildroot-linux-gnu/bin/` directory as environment variables, and execute the script of automatically configuring environment variables in the top-level directory:

```
source envsetup.sh
```

Enter the command to view:

```
cd buildroot/output/rockchip_rk3358/host/usr/bin
./aarch64-linux-gcc --version
```

Then the following logs are printed:

```
aarch64-linux-gcc.br_real (Buildroot 2018.02-rc3-XXXXXX) 10.3.0
# XXXXXX is the latest commit ID of Buildroot
```

4.5.5.3 Build Modules in Buildroot

For example, for the busybox module, commonly used build commands are as follows:

- Build busybox

```
SDK$make busybox
```

- Rebuild busybox

```
SDK$make busybox-rebuild
```

- delete busybox

```
SDK$make busybox-dirclean
or
SDK$rm -rf buildroot/output/rockchip_rk3358/build/busybox-1.34.1
```

4.5.6 Firmware Package

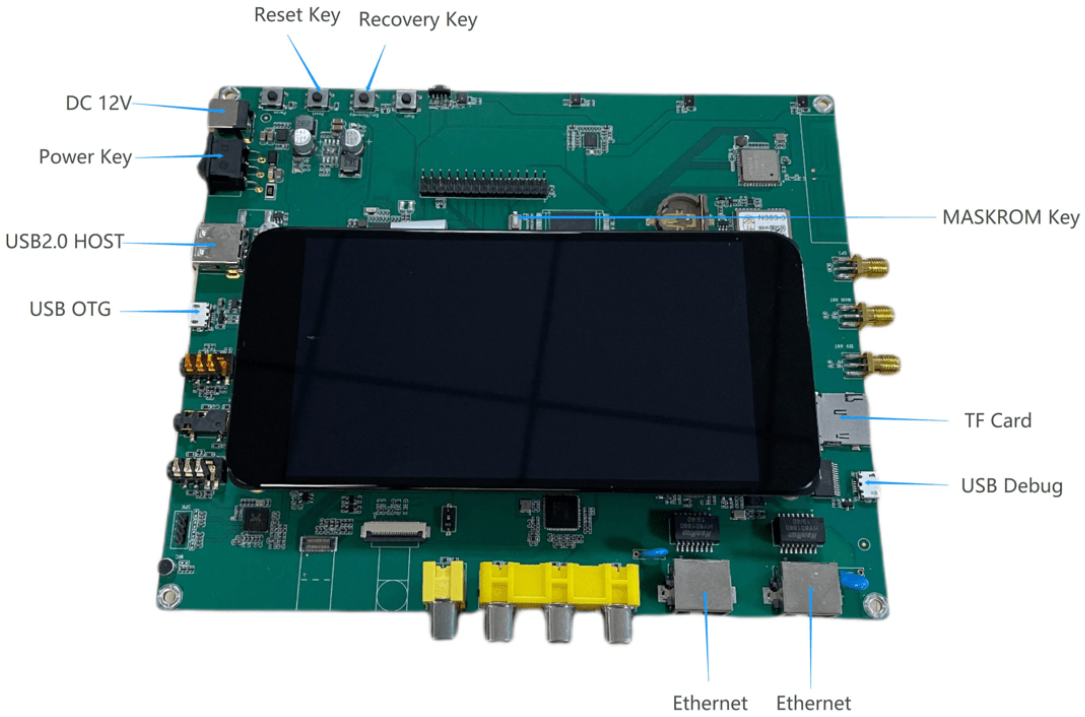
After compiling various parts of Kernel/U-Boot/Recovery/Rootfs above, enter root directory of project directory and run the following command to automatically complete all firmware packaged into rockdev directory:

Firmware generation:

```
./mkfirmware.sh
```

5. Upgrade Introduciton

Interfaces layout of RK3358 EVB board are showed as follows:

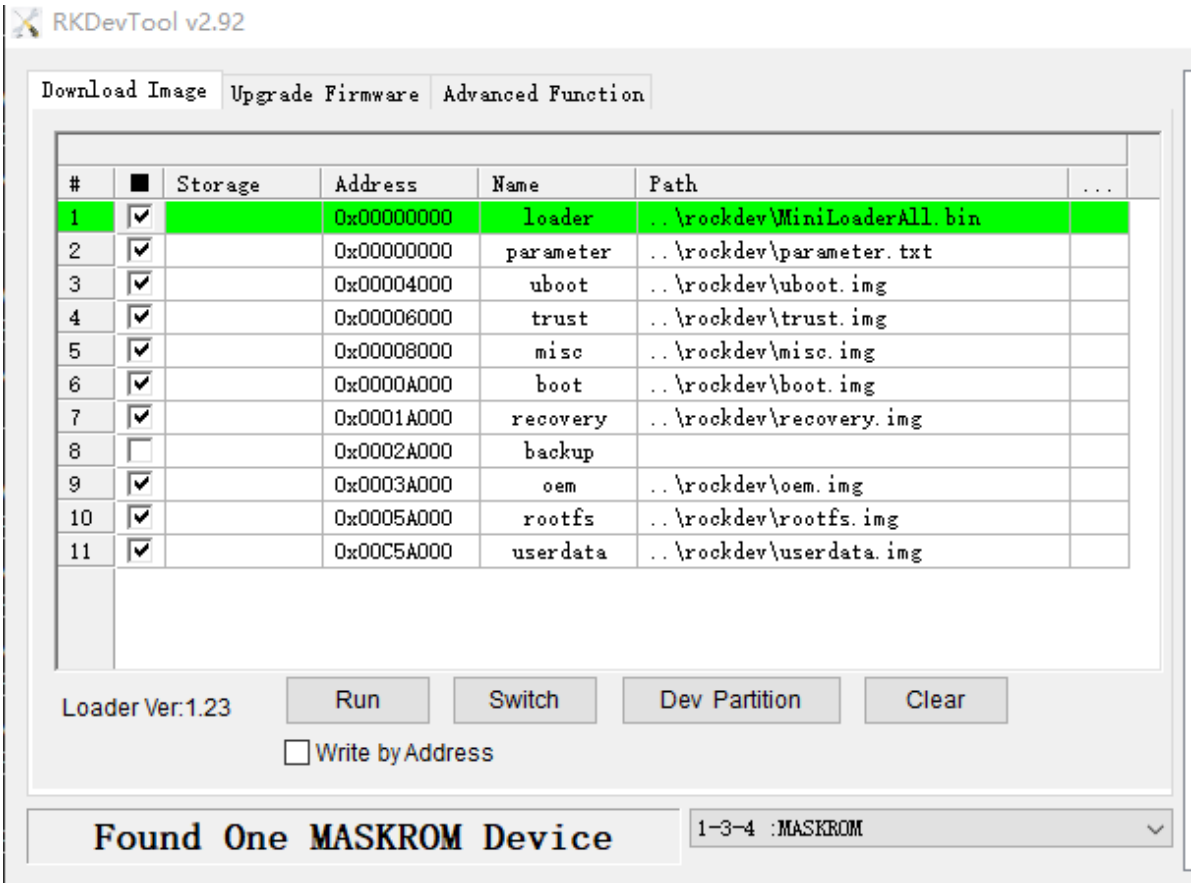


5.1 Windows Upgrade Introduction

SDK provides windows upgrade tool (this tool should be V2.92 or later version) which is located in project root directory:

```
tools/
├── windows/RKDevTool
```

As shown below, after compiling the corresponding firmware, device should enter MASKROM or BootROM mode for update. After connecting USB cable, long press the button “MASKROM” and press reset button “RST” at the same time and then release, device will enter MASKROM Mode. Then you should load the paths of the corresponding images and click “Run” to start upgrade. You can also press the “recovery” button and press reset button “RST”then release to enter loader mode to upgrade. Partition offset and flashing files of MASKROM Mode are shown as follows (Note: Window PC needs to run the tool as an administrator):



Note：Before upgrade, please install the latest USB driver, which is in the below directory:

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

5.2 Linux Upgrade Instruction

The Linux upgrade tool (Linux_Upgrade_Tool should be v2.1 or later versions) is located in “tools/linux” directory. Please make sure your board is connected to MASKROM/loader rockusb, if the compiled firmware is in rockdev directory, upgrade commands are as below:

```
sudo ./upgrade_tool ul -noreset rockdev/MiniLoaderAll.bin
sudo ./upgrade_tool di -p rockdev/parameter.txt
sudo ./upgrade_tool di -u rockdev/uboot.img
sudo ./upgrade_tool di -misc rockdev/misc.img
sudo ./upgrade_tool di -b rockdev/boot.img
sudo ./upgrade_tool di -recovery rockdev/recovery.img
sudo ./upgrade_tool di -oem rockdev/oem.img
sudo ./upgrade_tool di -rootfs rocdev/rootfs.img
sudo ./upgrade_tool di -userdata rockdev/userdata.img
sudo ./upgrade_tool rd
```

Or upgrade the whole update.img in the firmware

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

Or in root directory, run the following command on the machine to upgrade in MASKROM state:

```
./rkflash.sh
```

5.3 System Partition Introduction

Default partition introduction (below is RK3568 EVB reference partition):

Number	Start (sector)	End (sector)	Size	Name
1	16384	24575	4096K	uboot
2	24576	32767	4096K	trust
3	32768	40959	4096K	misc
4	40960	106495	32M	boot
5	106496	303104	32M	recovery
6	172032	237567	32M	bakcup
7	237568	368639	64M	oem
8	368640	12951551	6144M	rootfs
9	12951552	30535646	8585M	userdata

- uboot partition: for uboot.img built from uboot.
- misc partition: for misc.img built from recovery.
- boot partition: for boot.img built from kernel.
- recovery partition: for recovery.img built from recovery.
- backup partition: reserved, temporarily useless. Will be used for backup of recovery as in Android in future.
- oem partition: used by manufactor to store their APP or data, mounted in /oem directory

- rootfs partition: store rootfs.img built from buildroot.
- userdata partition: store files temporarily generated by APP or for users, mounted in /userdata directory

6. Fast start

The general startup process mostly goes through the following processes: miniloader -> uboot -> kernel -> rootfs -> app, so you only need to optimize the startup of each of the above processes to achieve a fast startup. This chapter describes tuning methods and considerations for RK3358 Fast Start.

Typical optimizations are as follows:

- Kernel and rootfs are loaded via SPL, and uboot is cut out；
- Kernel configuration for targeted trimming, reduce useless configuration, reduce kernel volume;
- Rootfs are tailored and reduced according to the needs of the application scenario;

For kernel, you can open init_debug, that is, CONFIG_PRINTK_TIME and CONFIG_KALLSYMS configuration, analyze the distribution of kernel startup time through `kernel/scripts/bootgraph .pl`, and optimize time-consuming modules in a targeted manner.

For rootfs, you need to build according to the application scenario, you can choose from toolchain, C library, GUI and other directions, such as:

- Later versions of gcc and binutils tend to have better optimization features;
- The standard and most comprehensive glibc can be replaced with configurable and small musl as needed;
- Rootfs' default integrated wayland weston can be replaced with other frameworks such as LVGL as needed;
- Trim the rootfs and remove useless startup and background services;

7. OTP

OTP (One Time Programmable), which is non-volatile storage that can be programmed only once. In contrast, flash storage can be erased and written multiple times. The OTP SIZE of the RK3358 is 64 bytes. Detailed instructions on OTP-related development can be found in the document /docs/Linux/Security/Rockchip_Developer_Guide_TEE_SDK_CN.pdf.

8. RK3358 SDK Firmware

- Baidu Cloud Disk

Buildroot

- Microsoft OneDriver

Buildroot