Rockchip RK3308 Linux5.10 SDK Quick Start

ID: RK-JC-YF-963

Release Version: V1.3.1

Release Date: 2023-12-20

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

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 the basic usage of Rockchip RK3308 Linux5.10 SDK, aiming to help developers get started with RK3308 Linux5.10 SDK faster.

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers

Software development engineers

Chipset and System Support

Chip Name	Uboot Version	Kernel Version	Buildroot Version
RK3308B/RK3308H/RK3308B-S/RK3308H-S	2017.9	5.10	2021.11

Revision History

Date	Version	Author	Revision History
2022-09-20	V1.0.0	LinJianHua	Initial version
2022-11-20	V1.0.1	Caesar Wang	Update Linux Upgrade Instruction
2023-09-20	V1.1.0	LinJianHua	Update SDK to V1.1.0
2023-09-20	V1.2.0	LinJianHua	Update SDK to V1.2.0
2023-09-20	V1.3.0	LinJianHua	Update SDK to V1.3.0
2023-12-20	V1.3.1	zack.huang	Added compilation instructions of smart home display control products

Contents

1. Set up an Development Environment

1.1 Preparing the development environment

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

1.2 Install libraries and toolsets

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

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

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

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

1.2.1 Setting up DNS to support for kgithub.com

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

1.2.2 Check and upgrade the `python` version of the host

The method of checking and upgrading the `python` version of the host is as follows:

- Check host `python` version

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

If you do not meet the requirements of python>=3.6 version, you can upgrade it in the following way:

- Upgrade `python 3.6.15` new version

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

1.2.3 Check and Upgrade the `make` Version on the Host

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

- Check the `make` version on the host

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

- Upgrade to the new version of `make 4.2`

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

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

1.2.4 Check and Upgrade the `lz4` Version on the Host

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

- Check the `lz4` version on the host

```
$ lz4 -v
*** LZ4 command line interface 64-bits v1.9.3, by Yann Collet ***
refusing to read from a console
```

- Upgrade to the new version of `lz4`

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

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

1.2.5 Check and Upgrade the `git` Version on the Host

- Check the `git` version on the host

```
$ /usr/bin/git -v
git version 2.38.0
```

- Upgrade to the new version of `git`

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

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

2. Software Development Guide

2.1 Development Guide

Aiming to help engineers get started with SDK development and debugging faster, “Rockchip_Developer_Guide_Linux_Software_CN.pdf” is released with the SDK, please refer to the documents under the project's docs/ directory, which will be continuously improved and updated.

2.2 Software Update History

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

```
.repo/manifests$ realpath rk3308_linux5.10_release.xml
# e.g.:the printed version is v1.3.0 and the update time is 20230920
#<SDK>/.repo/manifests/rk3308_linux/rk3308_linux5.10_release_v1.3.0_20230920.xml
```

3. SDK Configuration Framework Introduction

3.1 SDK Project Directory Introduction

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

- app: stores application APPs like qcamera/qfm/qplayer/qseting and other applications.
- buildroot: root file system based on Buildroot (2021.11).
- device/rockchip: stores 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.
- external: stores some third-party libraries, including audio, video, network, recovery and so on.
- kernel: stores kernel5.10 development code.
- output: stores the firmware information, compilation information, XML, host environment, etc. generated each time.
- 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. SDK Building Introduction

The SDK can configure and compile relevant features by using `make` or `./build.sh` with target parameters. Please refer to the `device/rockchip/common/README.md` compilation instructions for details.

4.1 SDK Compilation Command View

`make help`, for example:

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

The actual operation of make is `./build.sh`

You can also run `./build.sh <target>` to compile the relevant functions, which can be done through `./build.sh help` View the specific compilation commands.

```
$ ./build.sh -h

############### Rockchip Linux SDK ###############

Manifest: rockchip_linux5.10.xml

Usage: build.sh [OPTIONS]
Available options:
chip[:<chip>[:<config>]]                choose chip
defconfig[:<config>]                    choose defconfig
 *_defconfig                            switch to specified defconfig
    available defconfigs:
        rockchip_32bit_defconfig
        rockchip_defconfig
        rockchip_rk3308_rtos_amp_32bit_defconfig
        rockchip_rk3308_rtos_linux_64bit_defconfig
        rockchip_rk3308_rtos_smp_32bit_defconfig
        rockchip_rk3308b_32bit_defconfig
        rockchip_rk3308b_64bit_defconfig
        rockchip_rk3308bs_32bit_defconfig
        rockchip_rk3308bs_32bit_display_defconfig
        rockchip_rk3308bs_64bit_defconfig
        rockchip_rk3308bs_evb_v20_ia_32bit_defconfig
        rockchip_rk3308h_32bit_defconfig
        rockchip_rk3308hs_32bit_defconfig
 olddefconfig                           resolve any unresolved symbols in .config
 savedefconfig                          save current config to defconfig
 menuconfig                             interactive curses-based configurator
config                                  modify SDK defconfig
shell                                   setup a shell for developing
print-parts                             print partitions
mod-parts                               interactive partition table modify
edit-parts                              edit raw partitions
new-parts:<offset>:<name>:<size>...     re-create partitions
insert-part:<idx>:<name>[:<size>]       insert partition
del-part:(<idx>|<name>)                 delete partition
move-part:(<idx>|<name>):<idx>          move partition
rename-part:(<idx>|<name>):<name>       rename partition
resize-part:(<idx>|<name>):<size>       resize partition
kernel-4.19[:cmds]                      build kernel 4.19
kernel-4.4[:cmds]                       build kernel 4.4
kernel-5.10[:cmds]                      build kernel 5.10
kernel-5.10-rt[:cmds]                   build kernel 5.10-rt
kernel-6.1[:cmds]                       build kernel 6.1
kernel-rt[:cmds]                build kernel rt
kernel[:cmds]                           build kernel
modules[:cmds]                          build kernel modules
linux-headers[:cmds]                    build linux-headers
kernel-config[:cmds]                    modify kernel defconfig
kernel-make[:<arg1>:<arg2>]             run kernel make (alias kmake)
wifibt[:<dst dir>[:<chip>]]             build Wifi/BT
rtos                                    build and pack RTOS
buildroot-config[:<config>]             modify buildroot defconfig
buildroot-make[:<arg1>:<arg2>]          run buildroot make (alias bmake)
rootfs[:<rootfs type>]                  build default rootfs
buildroot                               build buildroot rootfs
yocto                                   build yocto rootfs
debian                                  build debian rootfs
```

```
recovery                                build recovery
pcba                                    build PCBA
security_check                          check contidions for security boot
createkeys                              build security boot keys
security_ramboot                        build security ramboot
security_uboot                          build uboot with security
security_boot                           build boot with security
security_recovery                       build recovery with security
security_rootfs                         build rootfs with security
loader[:cmds]                           build loader (uboot)
uboot[:cmds]                            build u-boot
uefi[:cmds]                             build uefi
firmware                                pack and check firmwares
edit-package-file                       edit package-file
edit-ota-package-file                   edit A/B OTA package-file
updateimg                               build update image
otapackage                              build A/B OTA update image
all                                     build all images
save                                    save images and build info
allsave                                 build all images and save them
cleanall                                cleanup
clean[:module[:module]]...              cleanup modules
    available modules:
        all
        config
        firmware
        kernel
        loader
        pcba
        recovery
        rootfs
        updateimg
post-rootfs <rootfs dir>                trigger post-rootfs hook scripts
help                                    usage

Default option is 'allsave'.
```

4.2 SDK Board Level Configuration

Enter the project `<SDK>/device/rockchip/rk3308` directory:

Board Configuration	Description
rockchip_defconfig	For RK3308BS EVB V20 development board with display run 64bits system
rockchip_32bit_defconfig	For RK3308BS EVB V20 development board with display run 64bits system
rockchip_rk3308bs_64bit_defconfig	For RK3308BS EVB V11\V20 development board run 64bits system
rockchip_rk3308bs_32bit_defconfig	For RK3308BS EVB V11\V20 development board run 32bits system
rockchip_rk3308hs_32bit_defconfig	For RK3308HS MODULE V10 development board run 32bits system
rockchip_rk3308b_64bit_defconfig	For RK3308B EVB V10 development board run 64bits system
rockchip_rk3308b_32bit_defconfig	For RK3308B EVB V10 development board run 32bits system
rockchip_rk3308h_32bit_defconfig	For RK3308H MODULE V10 development board run 32bits system
rockchip_rk3308bs_evb_v20_ia_32bit_defconfig	For RK3308 EVB V11/V20 development board run 32bits system display and control demo
rockchip_rk3308bs_32bit_display_defconfig	For RK3308 EVB V11/V20 development board run 32bits system lvgl demo
rockchip-rk3308-evb-audio-y10-64bit-defconfig	For RK3308 Audio EVB V10 development board run 64bits system

The first way:

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

Select the board configuration of the **RK3308BS EVB V11\V20 development board**:

```
rk3308$./build.sh device/rockchip/rk3308/rockchip_rk3308bs_64bit_defconfig
```

The second way:

```
~/3308/5.10_sdk$ ./build.sh lunch

############### Rockchip Linux SDK ###############

Manifest: rockchip_linux5.10.xml

Log saved at /home/ljh/3308/5.10_sdk/output/sessions/2023-09-15_17-56-01

Pick a defconfig:

1. rockchip_defconfig
2. rockchip_32bit_defconfig
```

```
3. rockchip_rk3308_rtos_amp_32bit_defconfig
4. rockchip_rk3308_rtos_linux_64bit_defconfig
5. rockchip_rk3308_rtos_smp_32bit_defconfig
6. rockchip_rk3308b_32bit_defconfig
7. rockchip_rk3308b_64bit_defconfig
8. rockchip_rk3308bs_32bit_defconfig
9. rockchip_rk3308bs_32bit_display_defconfig
10. rockchip_rk3308bs_64bit_defconfig
11. rockchip_rk3308bs_evb_v20_ia_32bit_defconfig
12. rockchip_rk3308h_32bit_defconfig
13. rockchip_rk3308hs_32bit_defconfig
Which would you like? [1]: 10
```

4.3 Configuring Different Components of startup/kernel/system in the SDK

The SDK can be configured for different components using `make menuconfig`, and the currently available components are mainly as follows:

Note that after configuring menuconfig, you need to save the configuration with `make savedefconfig`.

```
(rk3308) SoC
    Rootfs  --->
    Loader (u-boot)  --->
    Kernel  --->
    Boot  --->
    Recovery (buildroot)  --->
    PCBA test (buildroot)  --->
    Security  --->
    Extra partitions  --->
    Firmware  --->
    Update (Update image, OTA and A/B)  --->
    Others configurations  --->
```

With the above configuration, different rootfs/loader/kernel configurations can be selected for various customize compilations. It also has a userful command line switching function.

> Note that after configuring menuconfig, you need to save the configuration with `make savedefconfig`

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. Save the patches of each module to the out 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
```

4.5.2 Kernel Build

- Method 1

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

- Method 2

```
cd kernel
export CROSS_COMPILE=../prebuilts/gcc/linux-x86/aarch64/gcc-arm-10.3-2021.07-x86_64-aarch64-none-linux-gnu/bin/aarch64-none-linux-gnu-
make ARCH=arm64 rk3308_linux_defconfig
make ARCH=arm64 rk3308bs-evb-amic-v11.img -j8
或
make ARCH=arm64 rk3308_linux_defconfig rk3308bs_mipi_display.config
make ARCH=arm64 rk3308bs-evb-mipi-display-v11.img -j8
```

4.5.3 Recovery Build

```
### Recovery build command
./build.sh 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 compilations, rootfs.squashfs is generated in Buildroot directory “ output/rockchip_rk3308/images”.

4.5.5 Cross-Compilation

4.5.5.1 SDK Directory Built-in Cross-Compilation

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

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

4.5.5.2 Buildroot Built-in Cross-compilation

The configuration of different target functions can be set through `source buildroot/envsetup.sh`

```
~/3308/5.10_sdk$ source buildroot/envsetup.sh
Top of tree: /home/ljh/3308/5.10_sdk

Pick a board:

...
11. rockchip_rk3308_32_release
12. rockchip_rk3308_b_32_release
13. rockchip_rk3308_b_release
14. rockchip_rk3308_bs_32_ia_release
15. rockchip_rk3308_bs_32_lvgl_release
16. rockchip_rk3308_bs_32_release
17. rockchip_rk3308_bs_lvgl_release
18. rockchip_rk3308_bs_release
19. rockchip_rk3308_h_32_release
20. rockchip_rk3308_recovery
21. rockchip_rk3308_release
...
Which would you like? [1]:
```

Default selection 18, `rockchip_rk3308_bs_release`.

Then enter the Buildroot directory for RK3308 and start compiling the relevant modules.

Among them, `rockchip_rk3308_bs_32_release` is used to compile the 32-bit Buildroot system, and `rockchip_rk3308_recovery` is used to compile the Recovery module.

For example, to compile the rockchip_test module, the commonly used compilation commands are as follows:

SDK#cd buildroot

- Enter buildroot

```
SDK$cd buildroot
```

- To build rockchip-test

```
buildroot$make rockchip-test
```

- Rebuild rockchip-test

```
buildroot$make rockchip-test-rebuild
```

- Remove rockchip-test

```
buildroot$make rockchip-test-dirclean
or
buildroot$rm -rf /buildroot/output/rockchip_rk3568/build/rockchip-test-master/
```

If you need to compile a single module or a third-party application, you need to configure the cross-compilation environment. For example, RK3308, its cross-compilation tool is located in the `buildroot/output/rockchip_rk3308_bs_release/host/usr` directory, you need to set the bin/ directory of the tool and the `aarch64-buildroot-linux-gnu/bin/` directory as the environment variable, execute the script that automatically configures environment variables in the top-level directory::

```
source buildroot/envsetup.sh rockchip_rk3308_bs_release
```

Enter the command to view:

```
cd buildroot/output/rockchip_rk3308_bs_release/host/usr/bin
./aarch64-linux-gcc --version
```

The following information will be printed:

```
aarch64-linux-gcc.br_real (Buildroot -g900f5662) 11.3.0
```

4.5.6 Firmware Package

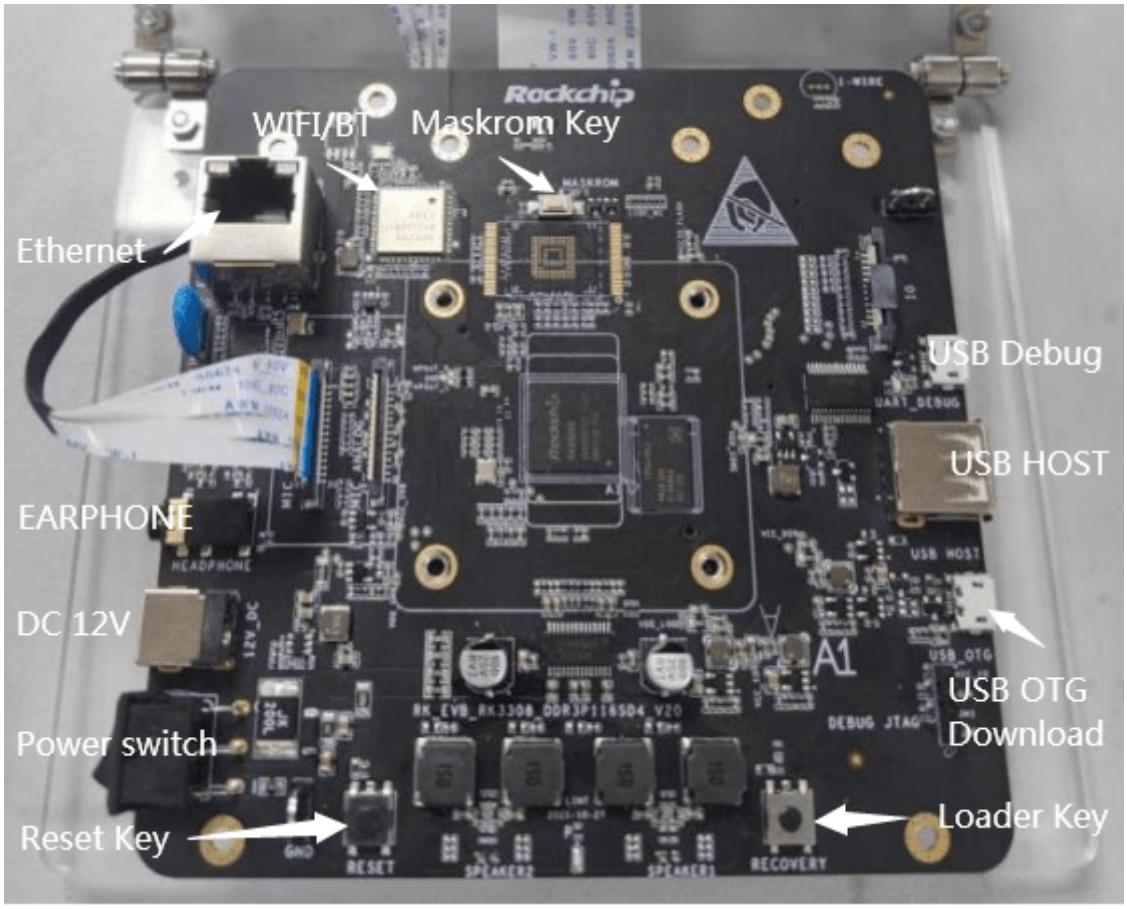
After compiling various parts of Kernel/U-Boot/Recovery/Rootfs above, enter root directory of project directory and run the following command to automatically complete all firmware packaged into rockdev directory:

Firmware generation:

```
./mkfirmware.sh
```

5. Upgrade Introduciton

The interface layout diagram of the top surface of RK3308B EVB V20 development board is as follows:

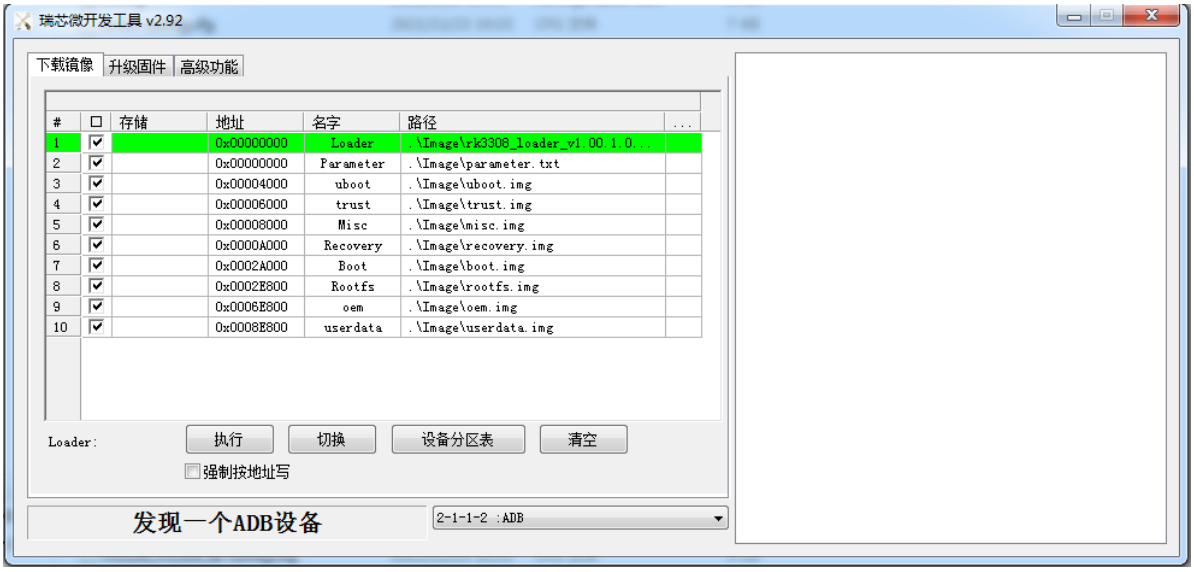


5.1 Windows Upgrade Introduction

SDK provides windows upgrade tool (this tool should be V291 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 rockdev/MiniLoaderAll.bin -noreset
sudo ./upgrade_tool di -p rockdev/parameter.txt
sudo ./upgrade_tool di -u rockdev/uboot.img
sudo ./upgrade_tool di -t 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 whole update.img in the firmware

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

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

```
./rkflash.sh
```

5.3 System Partition Introduction

Default partition introduction (below is RK3308 EVB reference partition):

- uboot partition: for uboot.img built from uboot.
- trust partition: for trust.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.
- rootfs partition: store rootfs.img built from buildroot or debian.
- oem partition: used by manufactor to store their APP or data, mounted in /oem directory
- userdata partition: store files temporarily generated by APP or for users, mounted in /userdata directory

6. RK3308 SDK Firmware

Link: https://console.zbox.filez.com/l/yJ42Ab