Rockchip RK3588 Linux SDK Quick Start

ID: RK-JC-YF-915

Release Version: V1.4.0

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 RK3588 Linux SDK, aiming to help developers get started with RK3588 Linux 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	Debian Version	Buildroot Version
RK3588, RK3588S, RK3588M, RK3588J	2017.9	5.10	11	2021.11

Revision History

Date	Version	Author	Revision History
2022-01-15	V0.0.1	Caesar Wang	Initial version
2022-04-14	V0.1.0	Caesar Wang	Beta version
2022-04-21	V0.1.1	Caesar Wang	Beta version 0.1.1
2022-05-20	V1.0.0	Caesar Wang	Release version
2022-06-20	V1.0.1	Caesar Wang	Update SDK to v1.0.1
2022-08-20	V1.0.2	Caesar Wang	Update SDK to v1.0.2
2022-09-20	V1.0.3	Caesar Wang	Update SDK to v1.0.3
2022-11-20	V1.0.4	Caesar Wang	Update Linux Upgrade Instruction
2023-04-20	V1.1.0	Caesar Wang	Update SDK to v1.1.0
2023-05-20	V1.1.1	Caesar Wang	Update SDK to v1.1.1
2023-06-20	V1.2.0	Caesar Wang	Update SDK to v1.2.0
2023-09-20	V1.3.0	Caesar Wang	Update SDK to v1.3.0
2023-12-20	V1.4.0	Caesar Wang	Update SDK to v1.4.0

Contents

1. SDK Precompiled Image

By using the RK3588 Linux SDK precompiled image, developers can bypass the process of compiling the entire operating system from source code. Instead, they can directly flash the precompiled image into the RK3588 development board, enabling rapid start of development and related evaluations and comparisons. This can reduce development time and cost wastage caused by compilation issues.

The firmware can be downloaded from the public address Click here for SDK firmware download.

Firmware path: General Linux SDK Firmware -> Linux5.10 -> RK3588

If you need to modify SDK code or for a quick start, please refer to the following chapters.

2. Set up an Development Environment

2.1 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

2.2 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 libgmp-dev \
libmpc-dev bc python-is-python3 python2
```

> **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:
>
> - 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.

2.2.1 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"
```

2.2.2 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
```

2.2.3 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
```

2.2.4 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
```

3. Docker Environment Setup

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

Before using the Docker environment, you can refer to the following document for instructions: `<SDK>/docs/en/Linux/Docker/Rockchip_Developer_Guide_Linux_Docker_Deploy_EN.pdf`.

The following systems have been verified:

Distribution Version	Docker Version	Image Loading	Firmware Compilation
ubuntu 22.04	24.0.5	pass	pass
ubuntu 21.10	20.10.12	pass	pass
ubuntu 21.04	20.10.7	pass	pass
ubuntu 18.04	20.10.7	pass	pass
fedora35	20.10.12	pass	NR (not run)

The Docker image can be obtained from the website Docker Image.

4. Software Development Guide

4.1 Development Guide

Aiming to help engineers get started with SDK development and debugging faster, We have released “Rockchip_Developer_Guide_Linux_Software_EN.pdf” with the SDK, please refer to the documents under the project's `docs/en/RK3588` directory.

4.2 Chip Datasheet

Aiming to help engineers get started with RK3588/RK3588S development and debugging faster. We have released "Rockchip_RK3588_Datasheet_V1.5-20220802.pdf" and "Rockchip_RK3588S_Datasheet_V1.5-20230117.pdf", please refer to the documents under the project's `docs/en/RK3588/Datasheet` directory.

4.3 Buildroot Development Guide

To assist development engineers in quickly becoming familiar with the development and debugging of the Buildroot system, the “Rockchip_Developer_Guide_Buildroot_EN.pdf” development guide is released with the SDK. It can be found in the `docs/en/Linux/System` directory and will be continuously updated and improved.

4.4 Debian Development Guide

To assist development engineers in quickly becoming familiar with the development and debugging of the Debian system, the “Rockchip_Developer_Guide_Debian_EN.pdf” development guide is released with the SDK. It can be found in the `docs/en/Linux/System` directory and will be continuously updated and improved.

4.5 Third-Party OS Porting

To help development engineers quickly become familiar with porting and adapting third-party OS, the “Rockchip_Developer_Guide_Third_Party_System_Adaptation_EN.pdf” development guide is released with the SDK. It can be found in the `docs/en/Linux/System` directory and will be continuously updated and improved.

4.6 UEFI Development Guide

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

4.7 RKNPU Development Guide

The SDK provides RKNPU-related development tools, as follows:

RKNN-TOOLKIT2

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

The development kit is located in the `external/rknn-toolkit2` directory, primarily used for model conversion, optimization, quantization, inference, performance evaluation, and accuracy analysis, among other functions.

Basic functionalities include:

Function	Description
Model Conversion	Supports Pytorch / TensorFlow / TFLite / ONNX / Caffe / Darknet floating-point models Supports Pytorch / TensorFlow / TFLite Quantization Aware Models (QAT) Supports dynamic input models (Dynamicization/Native Dynamics) Supports large models
Model Optimization	Constant folding/ OP correction/ OP Fuse&Convert / Weight Sparsification/ Model Pruning
Model Quantization	Supports quantization types: Asymmetric i8/ fp16 Supports Layer / Channel quantization methods; Normal / KL/ MMSE quantization algorithms Supports mixed quantization for balancing performance and accuracy
Model Inference	Supports model inference on PC via emulator Supports transferring models to NPU hardware for inference (connected board inference) Supports inference in quantities and multi-input models
Model Evaluation	Supports performance and memory evaluation of models on NPU hardware
Accuracy Analysis	Supports quantization accuracy analysis (emulator/ NPU)
Additional Functions	Supports version/device query functions, etc.

For more detailed instructions, please refer to the current doc/ directory documents:

```
├── 01_Rockchip_RKNPU_Quick_Start_RKNN_SDK_V1.6.0_EN.pdf
├── 02_Rockchip_RKNPU_User_Guide_RKNN_SDK_V1.6.0_EN.pdf
...
├── RKNN-Toolkit2_OP_Support-1.6.0.md
```

RKNN API

RKNN API development materials are located in the project directory `external/rknpu2`, which are used for inferring rknn models generated by RKNN-Toolkit2.
For more detailed instructions, please refer to the current doc/ directory documents:

```
...
├── 03_Rockchip_RKNPU_API_Reference_RKNN_Toolkit2_V1.6.0_EN.pdf
├── 04_Rockchip_RKNPU_API_Reference_RKNNRT_V1.6.0_EN.pdf
├── 05_RKNN_Compiler_Support_Operator_List_v1.6.0.pdf
├── RKNN-Toolkit2_API_Difference_With_Toolkit1-1.6.0.md
```

4.8 DPDK Development Guide

To help the development and debugging of RK3588 DPDK faster,
"Rockchip_Developer_Guide_Linux_DPDK_EN.pdf" and
"Rockchip_Developer_Guide_Linux_GMAC_DPDK_EN.pdf" development documents are released with SDK.
These development guides can be found in the `<SDK>/external/dpdk/rk_docs` directory and will be

continuously improved and updated.

4.9 Real-Time Linux Development Instructions

With the increasing demands for real-time performance in products, standard Linux's real-time capabilities can no longer meet the needs of many products. It's necessary to optimize standard Linux to enhance its real-time performance, such as using PREEMPT_RT, Xenomai real-time systems.

Please refer to `docs/Patches/Real-Time-Performance` for specific development patch packages and instructions.

4.10 Software Update History

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

```
.repo/manifests$ realpath rk3588_linux_release.xml
# e.g.:the printed version is v1.4.0 and the update time is 20231220
# <SDK>/.repo/manifests/rk3588_linux/rk3588_linux_release_v1.4.0_20231220.xml
```

- The update content of software release version can be viewed through the engineering text, please refer to the engineering directory:

```
<SDK>/.repo/manifests/rk3588_linux/RK3588_Linux5.10_SDK_Note.md
or
<SDK>/docs/RK3588/RK3588_Linux5.10_SDK_Note.md
```

- The board version information can be obtained through the following methods:

```
buildroot:/# cat /etc/os-release
NAME=Buildroot
VERSION=linux-5.10-gen-rkr7
ID=buildroot
VERSION_ID=2021.11
PRETTY_NAME="Buildroot 2021.11"
OS="buildroot"
BUILD_INFO="xxx  Mon Dec 18 23:12:04 CST 2023 - rockchip_rk3588"
KERNEL="5.10 - rockchip_linux_defconfig"
```

5. Hardware Development Guide

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

RK3588 hardware design guide:

```
<SDK>/docs/en/RK3588/Hardware/Rockchip_RK3588_Hardware_Design_Guide_V1.3_EN.pdf
<SDK>/docs/en/RK3588/Hardware/Rockchip_RK3588S_Hardware_Design_Guide_V1.0_EN.pdf
```

RK3588 EVB hardware user guide:

```
<SDK>/docs/en/RK3588/Hardware/Rockchip_RK3588_EVB_User_Guide_V1.1_EN.pdf
<SDK>/docs/en/RK3588/Hardware/Rockchip_RK3588S_EVB_User_Guide_V1.1_EN.pdf
<SDK>/docs/en/RK3588/Hardware/Rockchip_RK3588_EVB7_User_Guide_V1.0_EN.pdf
```

6. SDK Configuration Framework Introduction

6.1 SDK Project Directory Introduction

There are buildroot, debian, recovery, app, kernel, u-boot, device, docs, external and other directories in the project directory. Repositories are managed using manifests, and the repo tool is used to manage each directory or its subdirectory corresponding to a git.

- buildroot: root file system based on Buildroot.
- debian: root file system based on Debian.
- 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.
- external: stores some third-party libraries, including audio, video, network, recovery and so on.
- kernel: stores kernel 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.
- uefi: store UEFI code developed based on edk2 v2.7 version.
- yocto: stores the root file system developed based on Yocto 4.0.

7. SDK Building Introduction

The SDK can be accessed through `make` or `./build.sh` , add target parameters to configure and compile related functions.

7.1 SDK Compilation Commands

`make help` , e.g:

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.

7.2 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.

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

Manifest: rk3588_linux_release_v1.4.0_20231220.xml

Log colors: message notice warning error fatal

Usage: build.sh [OPTIONS]
Available options:
chip[:<chip>[:<config>]]            choose chip
defconfig[:<config>]                choose defconfig
config                              modify SDK defconfig
...
updateimg                           build update image
otapackage                          build A/B OTA update image
all                                 build images
release                             release images and build info
save                                alias of release
all-release                         build and release images
allsave                             alias of all-release
shell                               setup a shell for developing
cleanall                            cleanup
clean[:module[:module]]...          cleanup modules
post-rootfs <rootfs dir>            trigger post-rootfs hook scripts
help                                usage

Default option is 'all'.
```

7.3 SDK board-level configuration

Enter the project `<SDK>/device/rockchip/rk3588` directory:

Board Configuration	Description
rockchip_rk3588_evb1_lp4_v10_defconfig	For RK3588 EVB1 with LPDDR4 development board
rockchip_rk3588_evb7_v11_defconfig	For RK3588 EVB7 with LPDDR4 development board
rockchip_rk3588s_evb1_lp4x_v10_defconfig	For RK3588S EVB1 with LPDDR4 development board
rockchip_defconfig	Default configuration

The first way:

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

Select the board configuration of the **RK3588 EVB1 with LPDDR4 development board**:

```
./build.sh device/rockchip/rk3588/rockchip_rk3588_evb1_lp4_v10_defconfig
```

Select the board configuration of the **RK3588 EVB7 with single pmic on development board**:

```
./build.sh device/rockchip/rk3588/rockchip_rk3588_evb7_v11_defconfig
```

Select the board configuration of the **RK3588S EVB1 with LPDDR4 development board**:

```
./build.sh device/rockchip/rk3588/rockchip_rk3588s_evb1_lp4x_v10_defconfig
```

You can use `make lunch` or `./build.sh lunch` to configure

```
$ ./build.sh lunch

Pick a defconfig:

1. rockchip_defconfig
2. rockchip_rk3588_evb1_lp4_v10_defconfig
3. rockchip_rk3588_evb7_v11_defconfig
4. rockchip_rk3588s_evb1_lp4x_v10_defconfig
Which would you like? [1]:
```

For the configuration of other functions, use `make menuconfig` to configure related attributes.

> Explanation:
> The RK3588 EVB obtained from Rockchip official before April 2023 defaults to EVB1 configuration, and will be updated from The RK3588 EVB officially obtained by Rockchip defaults to the EVB7 configuration.

7.4 SDK Customization Configuration

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

```
(rk3588) SoC
    Rootfs  --->
    Loader (u-boot)  --->
    RTOS  --->
    Kernel  --->
    Boot  --->
    Recovery (based on buildroot)  --->
    PCBA test (based on buildroot)  --->
    Security  --->
    Extra partitions  --->
    Firmware  --->
    Update (Rockchip update image)  --->
    Others configurations  --->
```

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

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

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

7.5 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
```

It is Buildroot by default, you can specify rootfs by setting the environment variable RK_ROOTFS_SYSTEM. There are two types of system for RK_ROOTFS_SYSTEM: buildroot and debian.

If you need debain, you can generate it with the following command:

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

> Note:
> Every time the SDK is updated, it is recommended to clean up the previous compiled products and run them directly `./build.sh cleanall`

7.6 Build and Package Each Module

7.6.1 U-boot Build

```
./build.sh uboot
```

7.6.2 Kernel Build

- Method 1

```
./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 rockchip_linux_defconfig rk3588_linux.config
make ARCH=arm64 rk3588-evb1-lp4-v10-linux.img -j
or
make ARCH=arm64 rk3588-evb7-v11-linux.img -j
```

- Method 3

```
cd kernel
export CROSS_COMPILE=aarch64-linux-gnu-
make ARCH=arm64 rockchip_linux_defconfig rk3588_linux.config
make ARCH=arm64 rk3588-evb1-lp4-v10-linux.img -j
or
make ARCH=arm64 rk3588-evb7-v11-linux.img -j
```

7.6.3 Recovery Compilation

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

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

After compilation, a recovery.img is generated in the Buildroot directory `output/rockchip_rk3588_recovery/images`.

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

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

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

7.6.4 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.ext4 is generated in Buildroot directory “ output/rockchip_rk3588/images”.

7.6.5 Debian Build

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

After compilation, generate linaro-rootfs. img in the Debian directory.

> Description：re-install the depend packages
> sudo apt-get install binfmt-support qemu-user-static live-build
> sudo dpkg -i ubuntu-build-service/packages/*
> sudo apt-get install -f

For specific details, please refer to Debian development documentation reference:

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

7.6.6 Yocto Build

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

EVB boards:

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

After compiling, rootfs.img is generated in yocto directory "/build/lastest".
The default login username is root.

Please refer to Rockchip Wiki for more detailed information of Yocto.

FAQ:

- If you encounter the following problem during above compiling:

```
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 tosetup-locale-python3.

7.6.7 Cross-Compilation

7.6.7.1 SDK Directory Built-in Cross-Compilation

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

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

You can download the toolchain from the following address:

Click here

7.6.7.2 Buildroot Built-in Cross-compilation

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

```
$ source buildroot/envsetup.sh
Top of tree: rk3588

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

46. rockchip_rk3588
47. rockchip_rk3588_base
48. rockchip_rk3588_ramboot
49. rockchip_rk3588_recovery

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

Default selection 35, `rockchip_rk3588`.

Then enter the Buildroot directory for RK3588 and start compiling the relevant modules.

> rockchip_rk3588_base is the compilation of the basic components of the Buildroot system, androckchip_rk3588_recovery is used to compile the Recovery module, rockchip_rk3588_ramboot is the configuration used when starting secureboot.

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

Enter the buildroot directory

```
SDK#cd buildroot
```

- Delete and recompile rockchip-test

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

- Recompile rockchip-test

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

- Delete rockchip-test

```
buildroot#make rockchip-test-dirclean
or
buildroot#rm -rf output/rockchip_rk3588/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, RK3588, its cross-compilation tool is located in the `buildroot/output/rockchip_rk3588/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_rk3588
```

Enter the command to view:

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

The following information will be printed:

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

> Save to rootfs configuration file
> buildroot$ make update-defconfig

7.6.8 Firmware Package

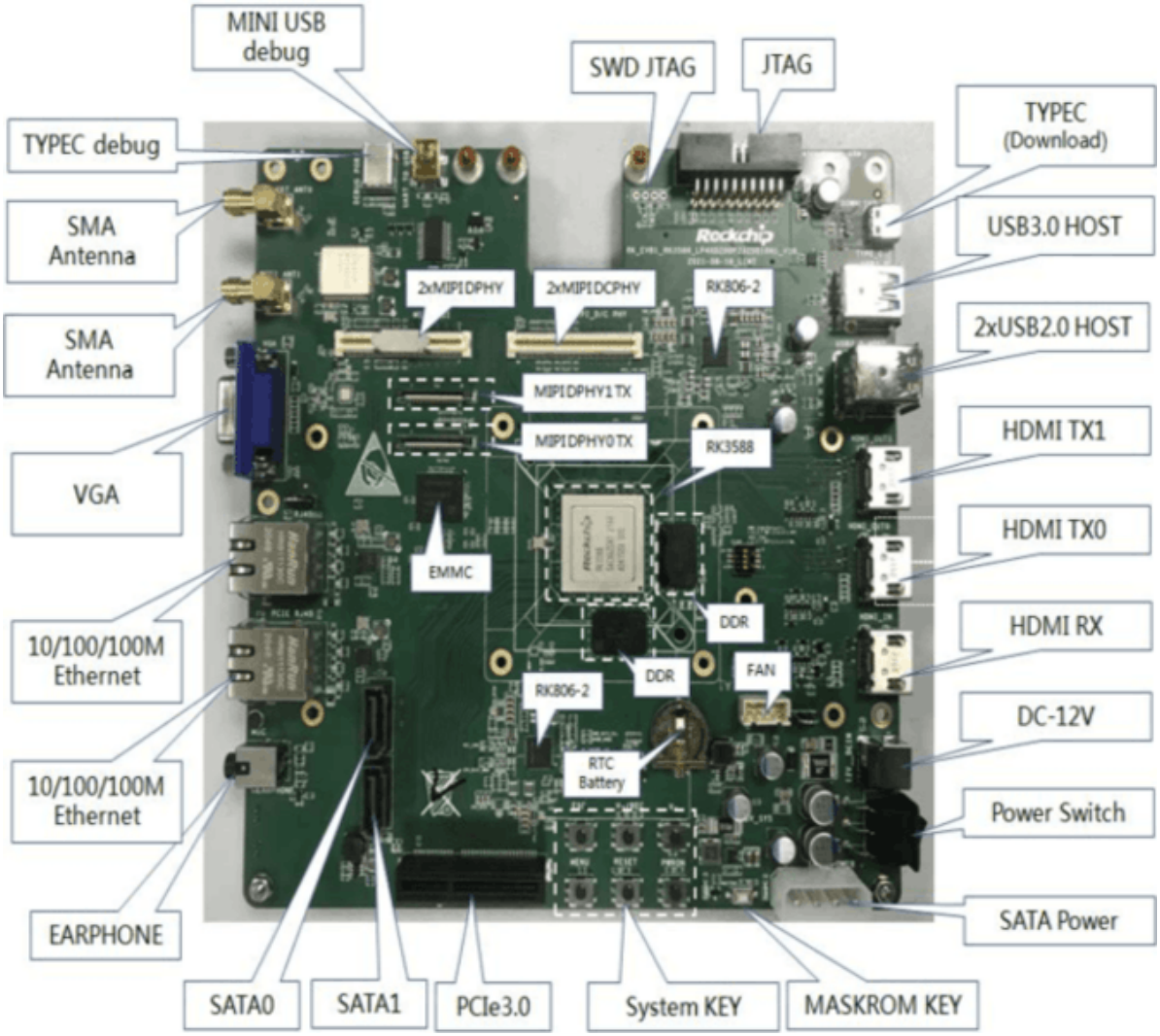
After compiling various parts of Kernel/U-Boot/Recovery/Rootfs above, enter root directory of project directory and run the following command to automatically complete all firmware packaged into `output/firmware` directory:

Firmware generation:

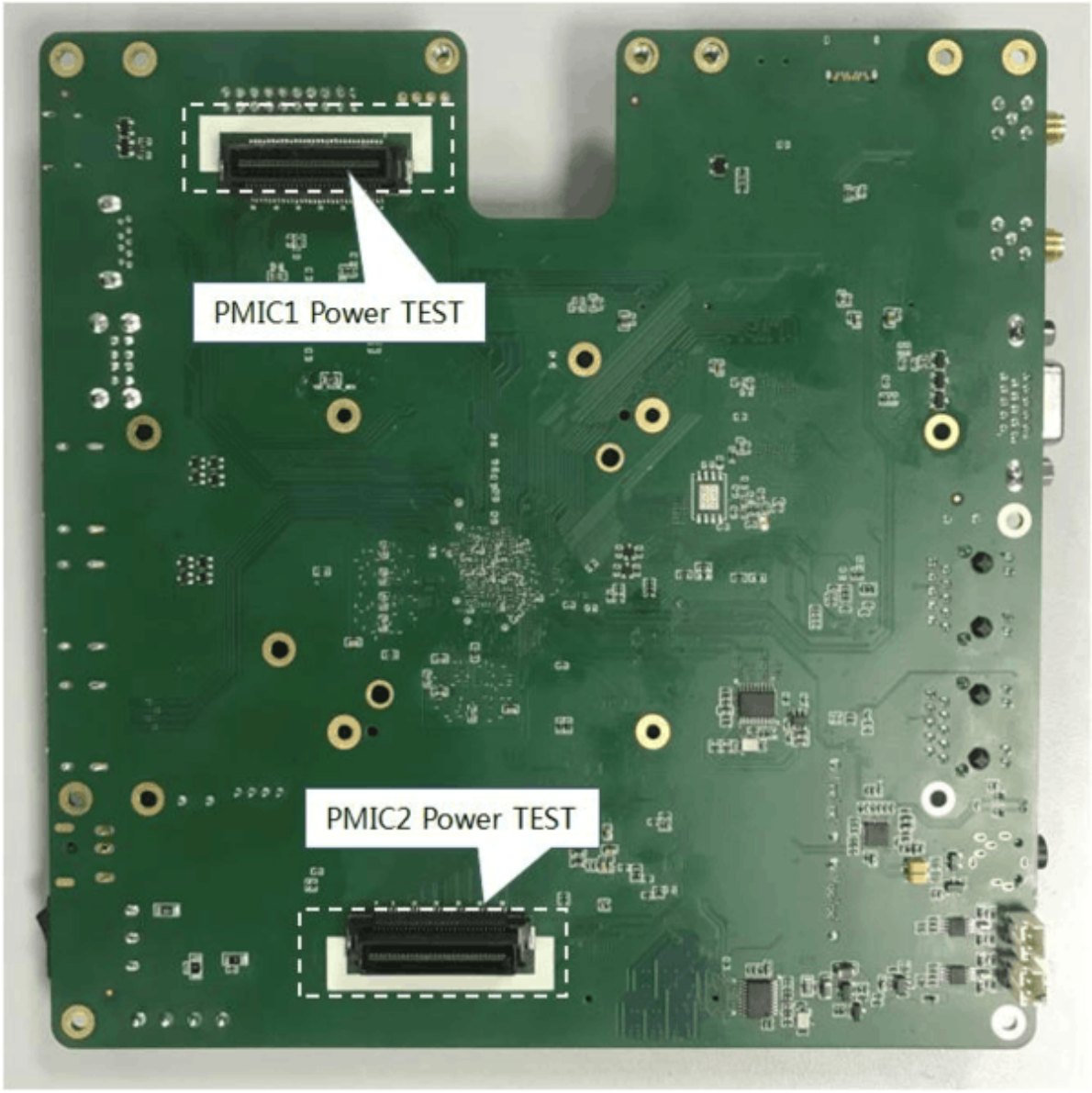
```
./build.sh firmware
```

8. Upgrade Introduciton

The interface layout diagram of the top surface of RK3588 EVB sdevelopment board is as follows:



The interface layout diagram of the bottom surface of RK3588 EVB development board is as follows:

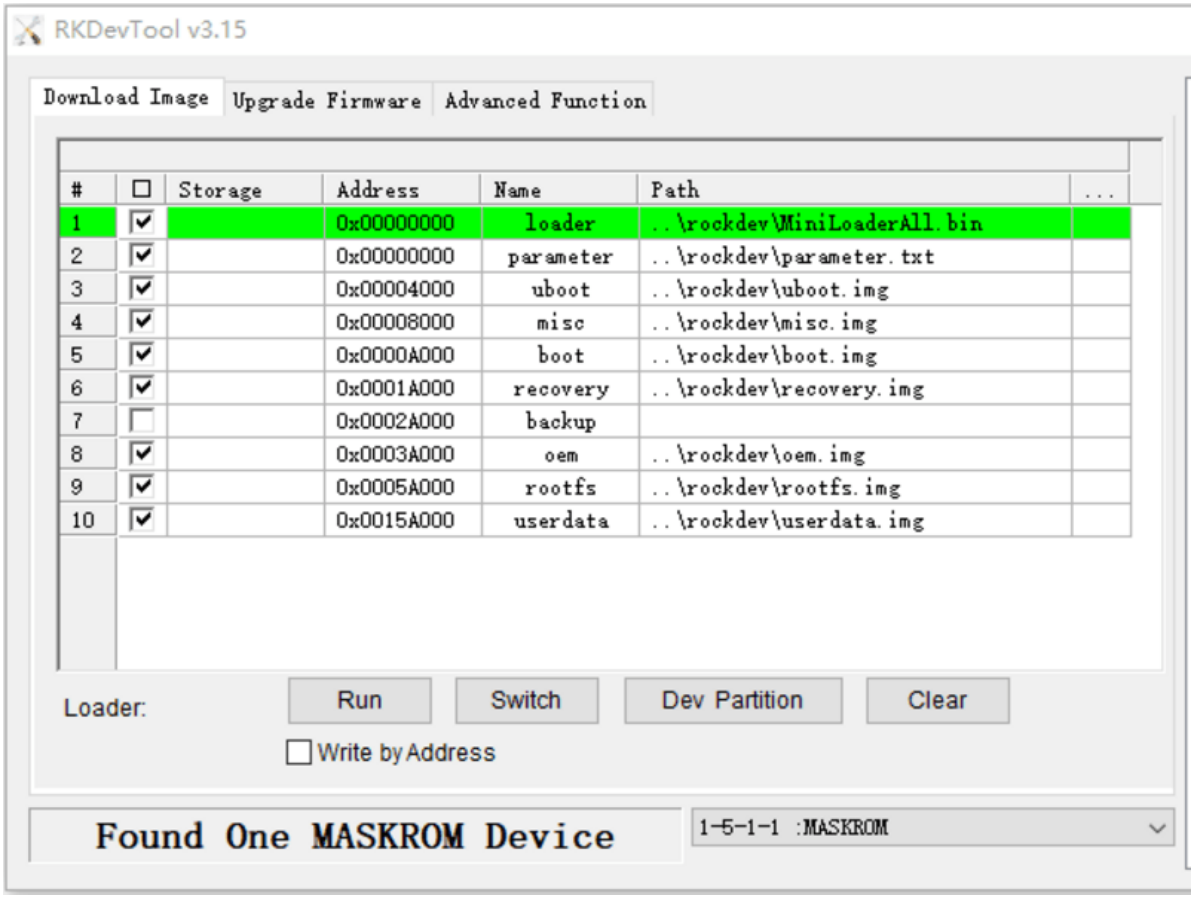


8.1 Windows Upgrade Introduction

SDK provides windows upgrade tool (this tool should be V2.17 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_v5.12.zip
```

8.2 Linux Upgrade Instruction

The Linux upgrade tool (Linux_Upgrade_Tool should be V2.17 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 -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
```

8.3 System Partition Introduction

Default partition introduction (below is RK3588 EVB reference partition):

Number	Start (sector)	End (sector)	Size	Name
1	8389kB	12.6MB	4194kB	uboot
2	12.6MB	16.8MB	4194kB	misc
3	16.8MB	83.9MB	67.1MB	boot
4	83.9MB	218MB	134MB	recovery
5	218MB	252MB	33.6MB	bakcup
6	252MB	15.3GB	15.0GB	rootfs
7	15.3GB	15.4GB	134MB	oem
8	15.6GB	31.3GB	15.6GB	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.
- 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

9. RK3588 SDK Firmware

Click here