Rockchip RK312X Linux SDK Quick Start

ID: RK-JC-YF-940

Release Version: V1.1.0

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 the basic usage of Rockchip RK312X Linux SDK, aiming to help developers get started with RK312X Linux SDK 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	Kernel Version
RK3126C/RK3128	2018.02-rc3	10	N/A	4.4

Revision History

Date	Version	Author	Revision History
2022-06-08	V1.0.0	Hans Yang	Release Version
2022-06-20	V1.1.0	WJL	Update instructions

Contents

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.

Considering the time cost of setting up the customer's development environment, we also provide the image mode of cross compiler docker for customer verification, so as to shorten the time-consuming of setting up the compilation environment.

Reference documents `Docker/Rockchip_Developer_Guide_Linux_Docker_Deploy_EN.pdf`.

The compatibility test results of docker compilation image system are as follows:

OS Vesion	Docker Version	Doading	Image build
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）

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 Chip Datasheet

Aiming to help engineers get started with RK3126C、RK3128 development and debugging faster. We have released "Rockchip RK3126C Datasheet V1.1 20191107.pdf" and "Rockchip RK3128 Datasheet V1.2 20170720.pdf".

2.3 Debian Development Guide

Aiming to help engineers get started with RK3128 Debian development and debugging faster, “Rockchip_Developer_Guide_Debian_CN.pdf” is released with the SDK, please refer to the documents under the project's docs/ApplicationNote directory, which will be continuously improved and updated.

2.4 Software Update History

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

```
.repo/manifests$ realpath rk312x_linux_release.xml
# e.g.:the printed version is v1.4.0 and the update time is 20220620
# <SDK>/.repo/manifests/rk312x_linux_release_v1.4.0_20220620.xml
```

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

```
.repo/manifests$ cat RK312X_Linux_SDK_Note.md
```

Or refer to the project directory:

```
<SDK>/docs/RK312X/RK312X_Linux_SDK_Note.md
```

3. Hardware Development Guide

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

RK3126C/RK3128 EVB hardware design guide:

```
<SDK>/docs/RK312X/Hareware/Rockchip_RK3126C_Hardware_Design_Guide_V1.0_CN.pdf
<SDK>/docs/RK312X/Hareware/Rockchip_RK3128_Hardware_Design_Guide_V0.1_CN.pdf
```

RK3126C/RK3128 EVB hardware development guide:

```
<SDK>/docs/RK312X/Hardware/Rockchip_RK3126C_EVB_User_Guide_V1.0_20220512_CN.pdf
<SDK>/docs/RK312X/Hardware/Rockchip_RK3128_EVB_User_Guide_V1.0_20220512_CN.pdf
```

4. SDK Configuration Framework Introduction

4.1 SDK Project Directory Introduction

There are buildroot, debian, 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: store application APPs with Demo.
- buildroot: root file system based on Buildroot (2018.02-rc3).
- debian: root file system based on Debian10.
- 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.
- 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 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/rk312x` directory:

Board Configuration	Description
BoardConfig-rk3126c-evb-ddr3-v10-debian.mk	For RK3126 EVB with DDR3 development board, using Debian operating system
BoardConfig-rk3126c-evb-ddr3-v10-slc-nand.mk	For RK3126 EVB with DDR3 development board, SLC NAND, use ubifs filesystem
BoardConfig-rk3126c-evb-ddr3-v10.mk	For RK3126 EVB with DDR3 development board
BoardConfig-rk3128-evb-ddr3-v10-debian.mk	For RK3128 EVB with DDR3 development board, using Debian operating system
BoardConfig-rk3128-evb-ddr3-v10.mk	For RK3128 EVB with DDR3 development board
BoardConfig.mk	Default

The first way:

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

Select the board configuration of the **RK3126 EVB with DDR3 development board, using Debian operating system**:

```
./build.sh device/rockchip/rk312x/BoardConfig-rk3126c-evb-ddr3-v10-debian.mk
```

Select the board configuration of the **RK3126 EVB with DDR3 development board, SLC NAND, use ubifs filesystem**:

```
./build.sh device/rockchip/rk312x/BoardConfig-rk3126c-evb-ddr3-v10-slc-nand.mk
```

Select the board configuration of the **RK3126 EVB with DDR3 development board**:

```
./build.sh device/rockchip/rk312x/BoardConfig-rk3126c-evb-ddr3-v10.mk
```

Select the board configuration of the **RK3128 EVB with DDR3 development board, using Debian operating system**:

```
./build.sh device/rockchip/rk312x/BoardConfig-rk3128-evb-ddr3-v10-debian.mk
```

Select the board configuration of the **RK3128 EVB with DDR3 development board**:

```
./build.sh device/rockchip/rk312x/BoardConfig-rk3128-evb-ddr3-v10.mk
```

The second way:

```
rk312x$ ./build.sh lunch
processing option: lunch

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

0. default BoardConfig.mk
1. BoardConfig-rk3126c-evb-ddr3-v10-debian.mk
2. BoardConfig-rk3126c-evb-ddr3-v10-slc-nand.mk
3. BoardConfig-rk3126c-evb-ddr3-v10.mk
4. BoardConfig-rk3128-evb-ddr3-v10-debian.mk
5. BoardConfig-rk3128-evb-ddr3-v10.mk
6. BoardConfig.mk
Which would you like? [0]:
...
```

4.3 Compilation Commands

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

```
rk312x$ ./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
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`

```
rk312x$ ./build.sh -h kernel
###Current SDK Default [ kernel ] Build Command###
cd kernel
make ARCH=arm rockchip_linux_defconfig rk3126_linux.config
make ARCH=arm rk3126c-evb-ddr3-v10-linux.img -j12
```

[note]: The detailed compilation commands should depending on corresponding SDK version, and there may be some differences between configurations. But the build command of build.sh is fixed.

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
```

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
```

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 compilations, rootfs.ext4 is generated in Buildroot directory " output/rockchip_rk3126c/images".

4.5.5 Debian Building

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

Or enter debian/ directory:

```
cd debian/
```

Please refer to the readme.md in the directory for further building and Debian firmware generation.

(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
```

Build 32 bit Debian:

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

After building, linaro-bullseye-alip-xxxxx-1.tar.gz (xxxxx is timestamp generated) will be generated in "debian/":

FAQ:

- If you encounter the following problem during above building:

```
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/buster-desktop-arm64/chroot' mounted with noexec or nodev
```

Solution：

```
mount -o remount,exec,dev xxx (xxx is the project directory), and then rebuild
```

In addition, if there are other building issues, firstly, please check that the building system is not ext2/ext4.

- Because building Base Debian requires to access to foreign websites, and when domestic networks access foreign websites, download failures often occur:

The live build is used by Debian, you can configure like below to change the image source to domestic:

```
+++ b/ubuntu-build-service/buster-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"
\
+ --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 the package cannot be downloaded for other network reasons, there are pre-build packages shared on Baidu Cloud Disk, put it in the current directory, and then do the next step directly.

(2) Building rk-debian rootfs

Build 32bit Debian：

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

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

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

The linaro-rootfs.img will be generated.

4.5.6 Cross-Compilation

4.5.6.1 SDK Directory Built-in Cross-Compilation

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

Contents	Description
prebuilts/gcc/linux-x86/arm/gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf	gcc arm 6.3.1 32-bit toolchain

4.5.6.2 Buildroot Built-in Cross-compilation

If you need to compile a single module or a third-party application, you need to configure the cross-compilation environment. For example, RK3126C, its cross-compilation tool is located in the `buildroot/output/rockchip_rk3126c/host/usr` directory, you need to set the bin/ directory of the tool and the `arm-buildroot-linux-gnueabihf/bin/` directory as the environment variable, execute the script that automatically configures environment variables in the top-level directory:

```
source envsetup.sh
```

Enter the command to view:

```
cd buildroot/output/rockchip_rk3126c/host/usr/bin
./arm-linux-gcc --version
```

The following information will be printed:

```
arm-linux-gcc.br_real (Buildroot 2018.02-rc3-g20055997) 10.3.0
```

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

- To build busybox

```
SDK$make busybox
```

- Rebuild busybox

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

- Remove busybox

```
SDK$make busybox-dirclean
or
SDK$rm -rf /buildroot/output/rockchip_rk3126c/build/busybox-1.34.1
```

4.5.7 Firmware Package

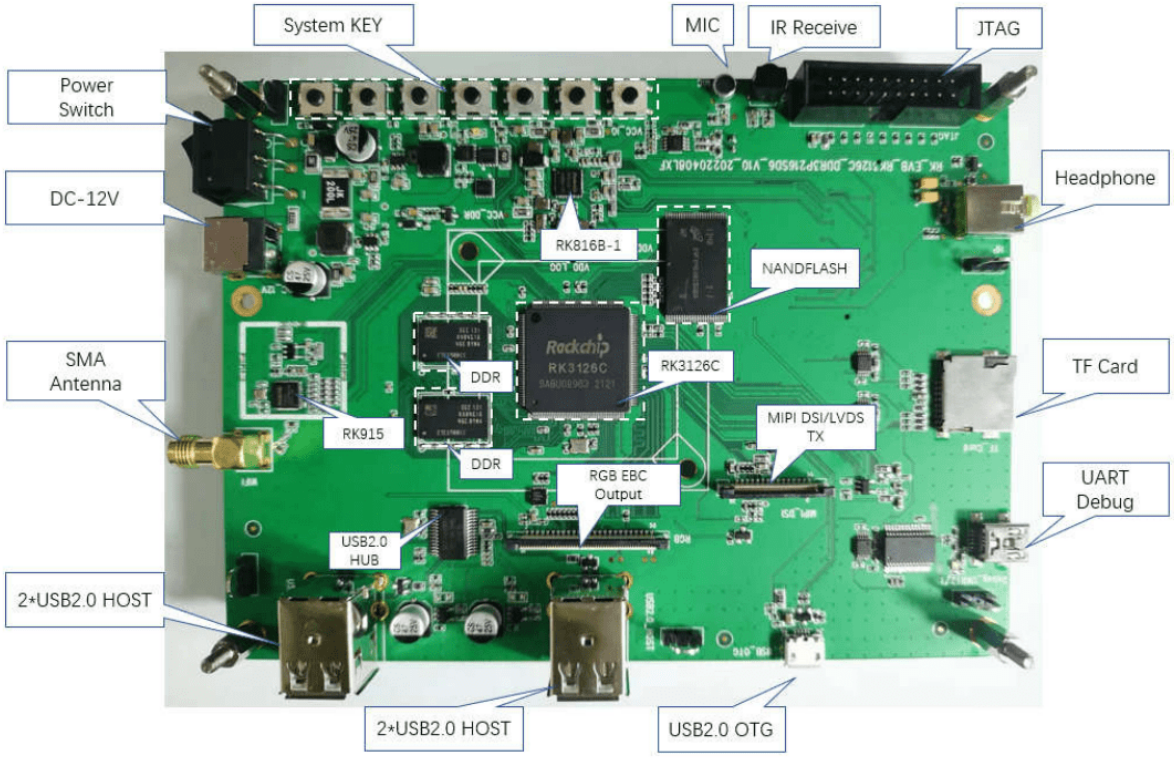
After compiling various parts of Kernel/U-Boot/Recovery/Rootfs above, enter root directory of project directory and run the following command to automatically complete all firmware packaged into rockdev directory:

Firmware generation:

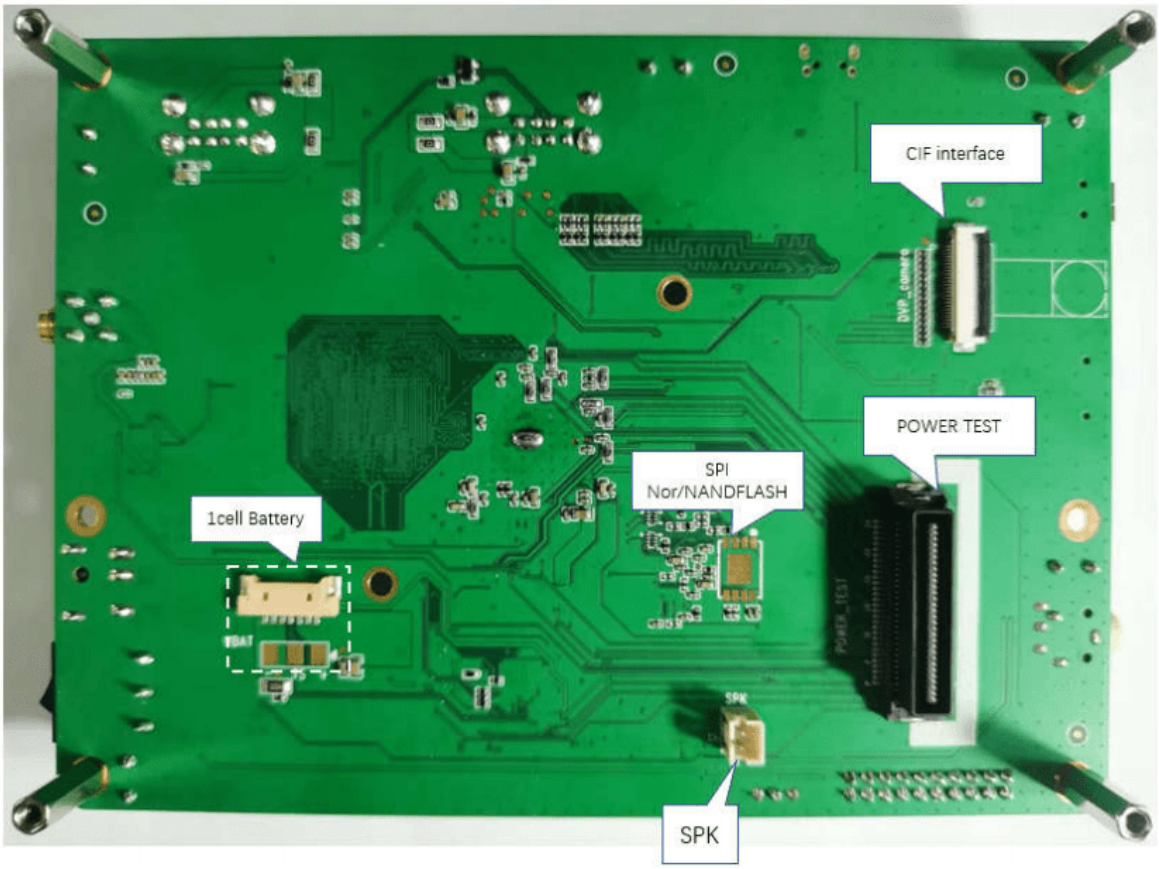
```
./mkfirmware.sh
```

5. Upgrade Introduciton

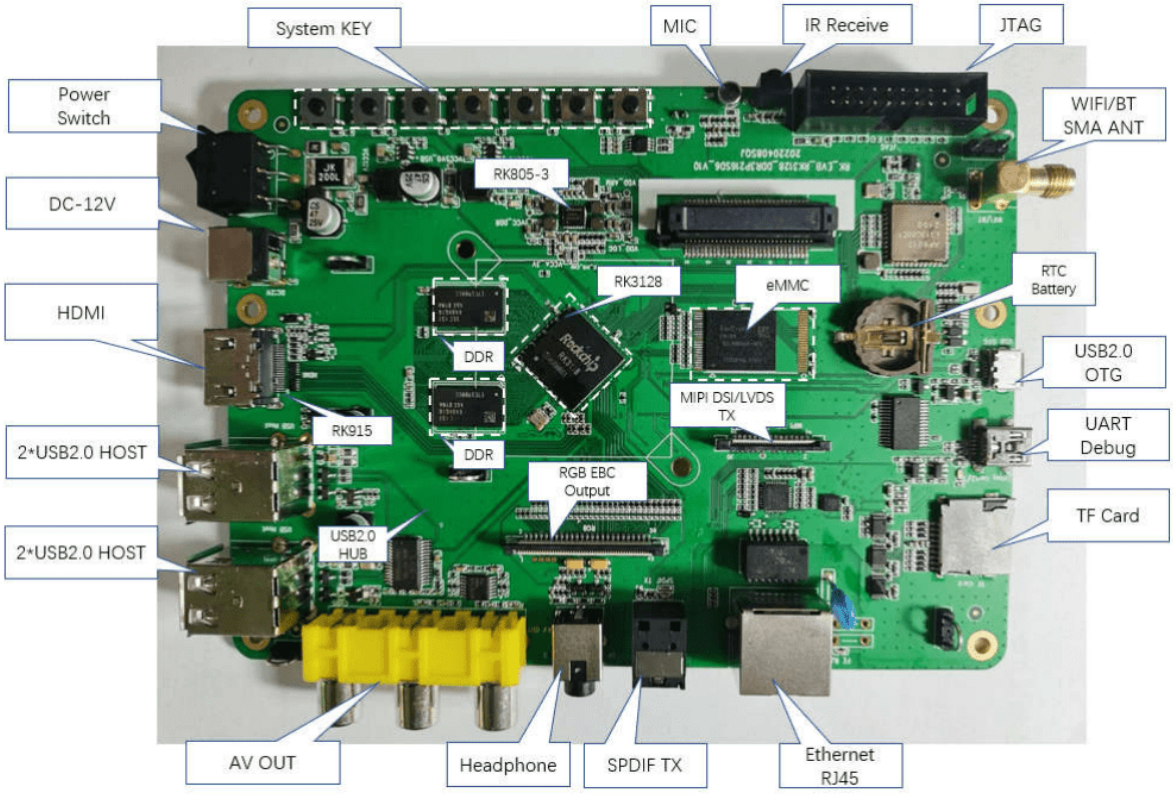
The interface layout diagram of the top surface of RK3126C EVB development board is as follows:



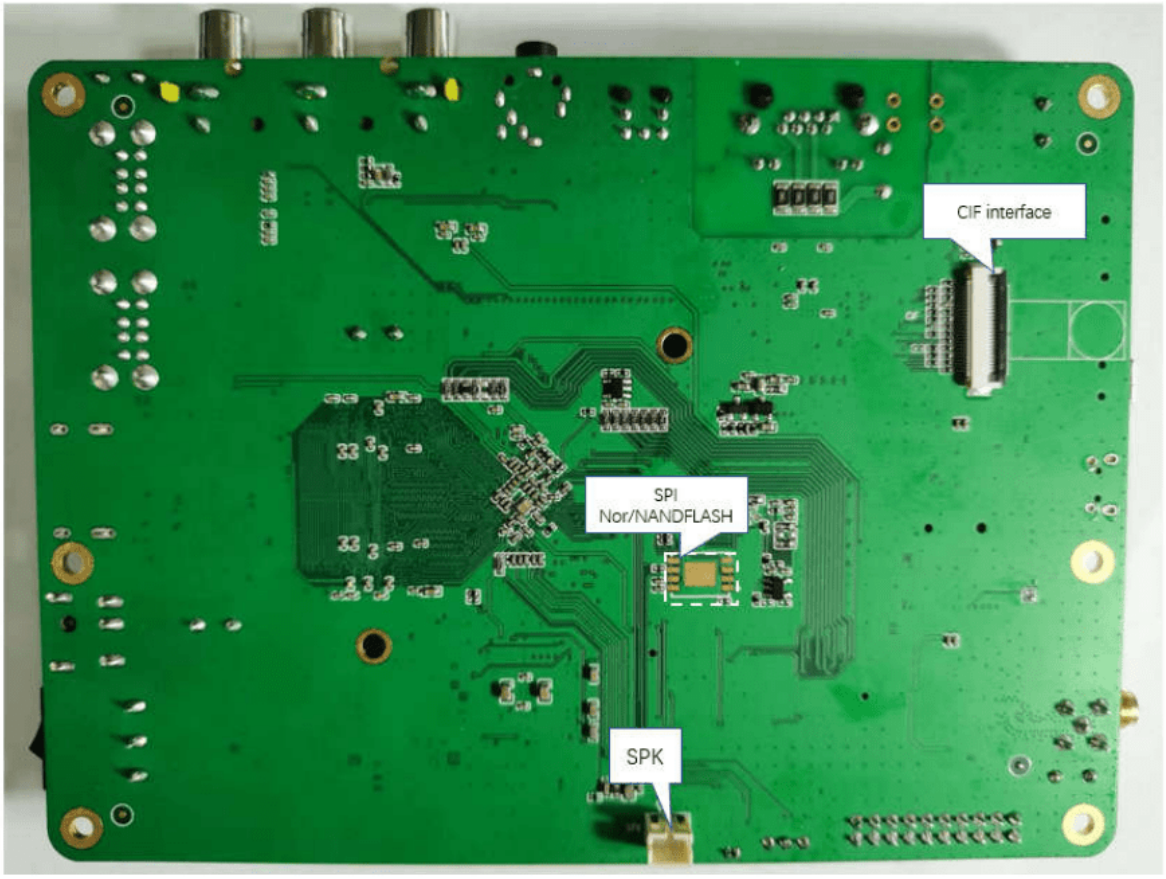
The interface layout diagram of the bottom surface of RK3126C EVB development board is as follows:



The interface layout diagram of the top surface of RK3128 EVB development board is as follows:



The interface layout diagram of the bottom surface of RK3128 EVB development board is as follows:

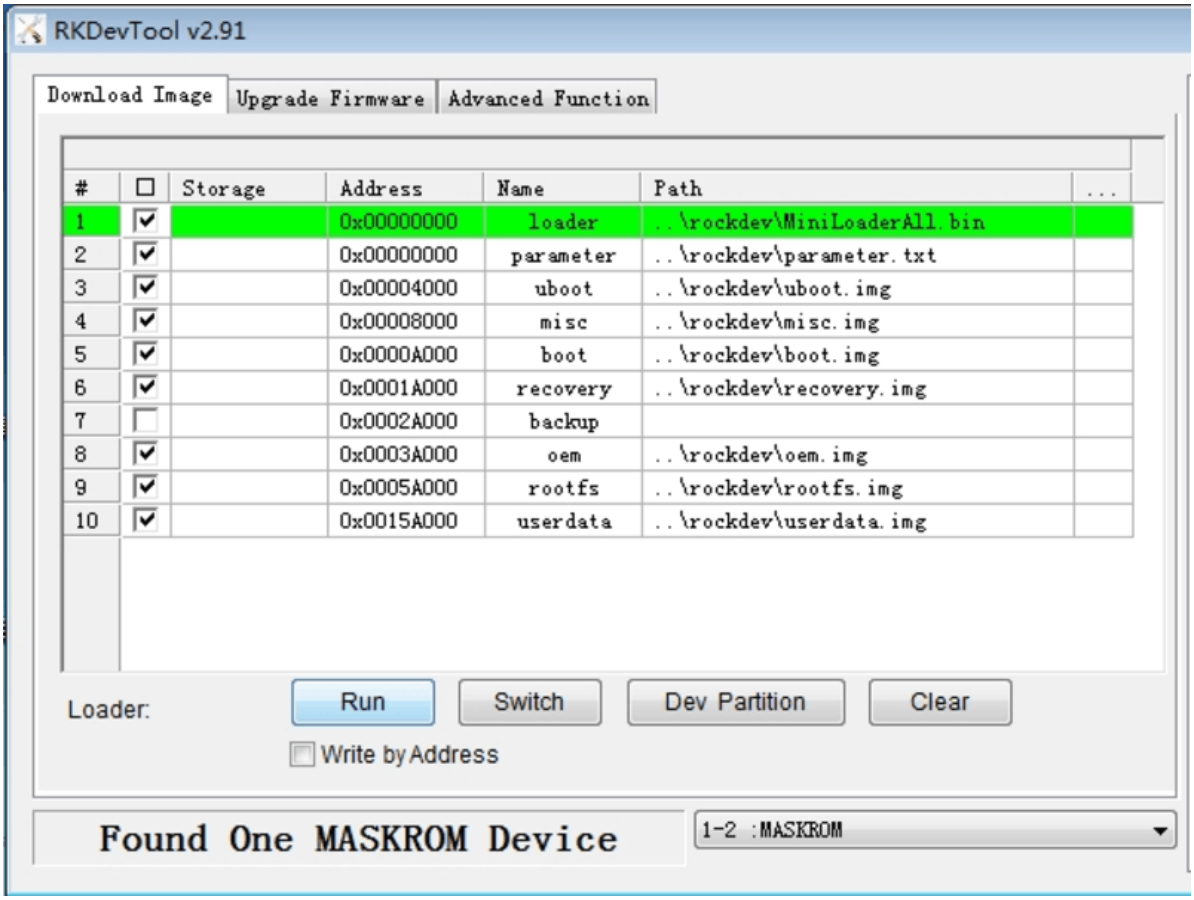


5.1 Windows Upgrade Introduction

SDK provides windows upgrade tool (this tool should be V2.91 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 device to upgrade in MASKROM state:

```
./rkflash.sh
```

5.3 System Partition Introduction

Default partition introduction (below is RK312X EVB reference partition):

Number	Name
1	uboot
2	misc
3	boot
4	recovery
5	bakcup
6	rootfs
7	oem
8	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