RV1126/RV1109 Linux SDK Quick Start

ID: RK-JC-YF-360

Release Version: V3.0.1

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 RV1126/RV1109 Linux SDK, aiming to help engineers get started with RV1126/RV1109 Linux SDK faster.

Product Version

Chipset	Kernel Version
RV1126/RV1109	Linux 4.19

Intended Audience

This document (this guide) is mainly intended for:

- Technical support engineers
- Software development engineers

Revision History

Version	Author	Date	Revision History
V0.0.1	CWW	2020-04-28	Initial version
V0.0.2	CWW	2020-05-09	Update the interface of RK IPCamera Tool
V0.0.3	CWW	2020-05-20	Add libssl-dev and expect for building environment
V1.0.0	CWW	2020-05-25	1. update chapter 3 & 4.4 & 4.5 2. add fast boot compile guide 3. add chapter 5.4
V1.1.0	CWW	2020-06-08	1. update company name 2. update document style 3. update chapter 2
V1.1.1	CWW	2020-06-16	fix company name
V1.2.0	HJC	2020-06-22	Add smart usb camera product section
V1.2.1	CWW	2020-06-29	1. update chapter 4.4 2. Add install fakeroot for building environment
V1.3.0	CWW	2020-07-09	1. Add module path and document introduction 2. Add Build different Boards
V1.3.1	CWW	2020-07-15	1. Remove unused board config introduction 2. fix same content3. fix eMMC typo
V1.4.0	CWW	2020-07-16	1. Add ISP Tool RKISP2.x_Tuner introduction 2. Add Develop Tool introduction 3. Add board config introduction
V1.4.1	CWW	2020-07-17	1. Update chapter of SDK Building Introduction 2. Update Tool's Description
V1.5.0	CWW	2020-08-07	1. Update SDK board configure and compile instruction 2. Add install cmake for development environment
V1.6.0	LJH	2020-08-22	1. Add facial gate product section 2. Update SDK compile instruction
V1.6.1	CWW	2020-09-07	1. Add new board RV1126_RV1109_IPC38_DEMO_V1.11_2020724LX
V1.7.0	CWW	2020-09-16	1. Add WiFi and Upgrade documents 2. Update compilation 3. update install bison and flex 4. Add print SDK version 5. Add two methods of compiling SDK in the chapter of "SDK Building Introduction"

Version	Author	Date	Revision History
V1.8.0	CWW	2020-09-25	1. Add install liblz4-tool, keychain and libtool for development environment 2. Update documents 3. Add ADB debug via network
V1.8.1	CWW	2020-10-29	update training video link
V1.8.2	LJH	2020-11-02	update facial gate product section
V1.9.0	CWW	2020-11-14	1. Update spi nand/slc nand BoardConfig and Document 2. Update requirement version for windows and Linux tool
V1.9.1	CWW	2020-11-22	Add spi nor BoardConfig
V1.9.2	CWW	2020-12-02	1. Add AB system boot-up board config reference 2. introduced to U-Boot's tftp
V1.9.3	CWW	2020-12-04	1. Add GPIO Precaution 2. Add introduction to starting face recognition function
V1.9.4	CWW	2020-12-17	1. Add AB system for SPI NAND with 38 board config 2. Add introduction to flash SPI NOR Firmware.img
V1.9.5	CWW	2020-12-29	1. Optimize typo 2. Remove some unused project
V1.9.6	CWW	2020-12-31	1. Add uboot use tftp to upgrage loader partition
V1.9.7	CWW	2021-01-08	1. Update the way to build UBI filesystem image
V1.9.8	XZY	2021-02-03	Add minigui_demo instruction
V2.0.0	CWW	2021-02-18	1. Add Linux Tool programmer_image_tool 2. Update introduction to Rootfs configuration 3. Update the project compilation method in the app and external directory 4. Add introduction to BSP compile
V2.0.1	CWW	2021-03-02	Add the QR code address
V2.1.0	CWW	2021-04-27	1. Update docs 2. Add cross-compile tool 3. Add application development suggestions
V2.2.0	CWW	2021-05-01	1. Update SDK Download Address 2. Fix typo Set up an Development Environment

Version	Author	Date	Revision History
V2.2.1	CWW	2021-05-14	1. Update Windows Tools programmer_image_tool
V2.2.2	CWW	2021-06-03	1. Update SDK development important documents
V2.2.3	CWW	2021-08-06	1. Add Booting from 128M DDR is supported 2. Add Support FIT image package 3. Add RGA development reference documents
V2.2.4	HJC	2021-08-16	1. Update smart usb camera product section
V2.2.5	CWW	2021-08-17	1. Add Filesystem type selection
V2.2.6	CWW	2022-08-09	1. Update ISP Tuning Tools and Document
V3.0.0	ZZJ GZC	2023-02-13	1. Remove smart usb camera product section 2. Remove facial gate section 3. Update Boardconfig, compilation and function introduction
V3.0.1	GZC	2023-12-20	Remove MiniGUI software just because the commercial license for the MiniGUI software from Beijing FMSoft Technologies Co., Ltd. ('FMSoft' for short) will expire from Dec 31, 2023. If you want to use, distribute MiniGui Software, you need to be licensed from FMSoft or be licensed under the GPL.

Contents

1. Set up an Development Environment

Ubuntu 16.04 system:
Please install software packages with below commands to set up a building environment:

```
sudo apt-get install repo device-tree-compiler \
git-core u-boot-tools mtools \
parted libudev-dev libusb-1.0-0-dev \
python-linaro-image-tools linaro-image-tools \
autoconf autotools-dev libsigsegv2 m4 \
intltool libdrm-dev curl sed make binutils \
build-essential gcc g++ bash patch gzip gawk \
bzip2 perl tar cpio python unzip rsync \
file bc wget libncurses5 libqt4-dev libglib2.0-dev \
libgtk2.0-dev libglade2-dev cvs git \
mercurial openssh-client subversion asciidoc w3m \
dblatex graphviz python-matplotlib \
libc6:i386 libssl-dev expect fakeroot cmake flex \
bison liblz4-tool libtool keychain
```

Ubuntu 17.04 or later version system:

In addition to the above software packages, the following dependencies is needed:

```
sudo apt-get install lib32gcc-7-dev  g++-7  libstdc++-7-dev
```

2. Important Documents that Must Be Read Before Developing SDK

Before SDK development, please read the following document "RV1126_RV1109_IO_Power_Domain_Configuration_Guide.pdf". This document mainly guides customers to correctly configure the IO power domain.

3. SDK Configuration Framework Introduction

3.1 SDK Project Directory Introduction

There are buildroot, 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.

- buildroot: customized root file system.
- app: store applications.

- external: related libraries, including audio and video.
- kernel: kernel code.
- device/rockchip: stores some scripts and prepared files for building and packaging firmware of each chip.
- docs: stores development guides, platform support lists, tool usage, Linux development guides, and so on.
- prebuilts: stores cross-compilation toolchain.
- rkbin: stores firmware and tools.
- rockdev: stores building output firmware.
- tools: stores some commonly used tools.
- u-boot: U-Boot code.

3.2 RV1109/RV1126 Modules Dirctory Introduction

some of modules directory path	modules introduction
external/linux-rga	Raster Graphic Acceleration (RGA2)
external/recovery	recovery and Rockchip upgrade code
external/rkwifibt	Wi-Fi and BT
external/rk_pcba_test	PCBA test code
external/isp2-ipc	Image signal processing server
external/mpp	encoding and decoding
external/rkmedia	Rockchip Multimedia interface
external/camera_engine_rkaiq	Image processing algorithm module
external/rknpu	NPU driver
external/rockface	Face recognition code
external/CallFunIpc	Application interprocess communication code
external/common_algorithm	Audio and video general algorithm library
external/rknn-toolkit	Development kit for model transformation, reasoning and performance evaluation
app/libIPCProtocol	Based on dbus, provides a functional interface for inter-process communication
app/mediaserver	Main application providing multimedia services (Used for IPC application development reference or simple function demonstration)
app/ipc-daemon	System guard service
app/dbserver	Database service
app/netserver	Network services
app/storage_manager	Storage management service
app/ipcweb-backend	web backend
app/librkdb	Database interface
app/ipcweb-ng	Web front end, using Angular 8 framework

3.3 RV1109/RV1126 Development Documents

3.3.1 Documents Index

```
├── docs
│   ├── Common (drivers documents of Linux kernel and
DDR/Flash/eMMC/Camera/WiFi/Bluetooth AVL)
│   │
│   ├── Linux (Rockchip Linux general documents, for RV1126/RV1109 reference
only)
│   │   ├── ApplicationNote
│   │   ├── Camera
│   │   ├── Graphics
│   │   ├── Multimedia (multimedia encoding and decoding interface develop
guide)
│   │   ├── Profile
│   │   ├── Recovery (upgrade document)
│   │   │   ├── Rockchip_Developer_Guide_Linux_Upgrade_CN.pdf
│   │   │   └── Rockchip_Developer_Guide_Linux_Upgrade_EN.pdf
│   │   │
│   │   ├── Security (security document, apply to RV1126/RV1109)
│   │   └── Wifibt   (WiFi and Bluetooth, apply to RV1126/RV1109)
│   │         ├── AP模组RF测试文档
│   │         ├── REALTEK模组RF测试文档
│   │         └── WIFI性能测试PC工具
│   │
│   ├── Others
│   │   ├── Rockchip_User_Guide_Bug_System_CN.pdf
│   │   └── Rockchip_User_Guide_SDK_Application_And_Synchronization_CN.pdf
│   │
│   │
│   └── RV1126_RV1109
│       ├── ApplicationNote (Rockchip Application Framework Develop
Introduction and Web Develop Guide)
│       │   ├── Rockchip_Developer_Guide_Linux_Application_Framework_CN.pdf
│       │   ├── Rockchip_Developer_Guide_Linux_Application_Framework_EN.pdf
│       │   ├── Rockchip_Instructions_Linux_CGI_API_CN.pdf
│       │   ├── Rockchip_Instructions_Linux_MediaServer_CN.pdf
│       │   ├── Rockchip_Instructions_Linux_MediaServer_EN.pdf
│       │   ├── Rockchip_Instructions_Linux_Web_Configuration_CN.pdf
│       │   └── Rockchip_Instructions_Linux_Web_Configuration_EN.pdf
│       │
│       ├── Camera (ISP develop guide)
│       │   ├── Camera_External_FAQ_v1.0.pdf
│       │   ├── Rockchip_Color_Optimization_Guide_ISP2x_V1.1.0.pdf
│       │   ├── Rockchip_Development_Guide_ISP2x_CN_v1.2.0.pdf
│       │   ├── Rockchip_Driver_Guide_ISP2x_CN_v0.1.0.pdf
│       │   ├── Rockchip_Instruction_Linux_Appliction_ISP20_CN.pdf
│       │   ├── Rockchip_IQ_Tools_Guide_ISP2x_CN_v1.0.0.pdf
│       │   ├── Rockchip_RV1109_RV1126_Developer_Guide_Linux_Ispserver_CN.pdf
│       │   └── Rockchip_Tuning_Guide_ISP2x_CN_v1.0.0.pdf
│       │
│       ├── Multimedia
│       │   └── Rockchip_Developer_Guide_Linux_RKMedia_CN.pdf (multimedia
interface develop guide)
│       │
│       ├── Rockchip_RV1126_RV1109_EVB_User_Guide_V1.0_CN.pdf (Hardware Develop
Guide)
```

```
│       ├── Rockchip_RV1126_RV1109_EVB_User_Guide_V1.0_EN.pdf
│       ├── Rockchip_RV1126_RV1109_Instruction_Linux_Separate_Building_EN.pdf
(Separate U-Boot/Kernel/Rootfs building from SDK)
│       ├── Rockchip_RV1126_RV1109_Linux_SDK_V1.1.1_20200711_CN.pdf (SDK
Release Note)
│       ├── Rockchip_RV1126_RV1109_Linux_SDK_V1.1.1_20200711_EN.pdf
│       ├── Rockchip_RV1126_RV1109_Quick_Start_Linux_CN.pdf  (Quick Start
Guide)
│       ├── Rockchip_RV1126_RV1109_Quick_Start_Linux_EN.pdf
│       ├── RV1109 Multimedia Codec Benchmark v1.2.pdf (Encoding&Decoding
Introduction)
│       ├── RV1126 Multimedia Codec Benchmark v1.1.pdf
│       └── RV1126_RV1109_Release_Note.txt
│
└── external
    │
    ├── linux-rga (RGA develop)
    │   ├── README.md
    │   └── docs
    │        ├── RGA_FAQ.md
    │        └── RGA_API_Instruction.md
    │
    ├── rknn-toolkit (Development kit for model transformation,
    │   │                  reasoning and performance evaluation)
    │   └── doc
    │       ├── Rockchip_Developer_Guide_RKNN_Toolkit_Custom_OP_V1.3.2_CN.pdf
    │       ├── Rockchip_Developer_Guide_RKNN_Toolkit_Custom_OP_V1.3.2_EN.pdf
    │       ├── Rockchip_Quick_Start_RKNN_Toolkit_V1.3.2_CN.pdf
    │       ├── Rockchip_Quick_Start_RKNN_Toolkit_V1.3.2_EN.pdf
    │       ├── Rockchip_Trouble_Shooting_RKNN_Toolkit_V1.3.2_CN.pdf
    │       ├── Rockchip_Trouble_Shooting_RKNN_Toolkit_V1.3.2_EN.pdf
    │       ├── Rockchip_User_Guide_RKNN_Toolkit_V1.3.2_CN.pdf
    │       ├── Rockchip_User_Guide_RKNN_Toolkit_V1.3.2_EN.pdf
    │       ├── Rockchip_User_Guide_RKNN_Toolkit_Visualization_V1.3.2_CN.pdf
    │       └── Rockchip_User_Guide_RKNN_Toolkit_Visualization_V1.3.2_EN.pdf
    └── rknpu
        └── rknn (Rockchip NPU Develop Guide)
            └── doc
                ├── Rockchip_User_Guide_RKNN_API_V1.3.3_CN.pdf
                └── Rockchip_User_Guide_RKNN_API_V1.3.3_EN.pdf
```

3.3.2 ISP Tuning Tools and Document

ISP developer guide documents and the support list of camera sensor can be got from the Redmine `https://redmine.rock-chips.com/documents/53`

3.3.3 SPI NAND/SLC NAND Document

Path of the document:
`docs/Linux/ApplicationNote/Rockchip_Developer_Guide_Linux_Nand_Flash_Open_Source_Solution_EN.pdf`

3.3.4 Some of Modules Video Training

- Instructions to RKMedia of RV1109 & RV1126

```
link: https://pan.baidu.com/s/1Z4o2v2KL6eCKXgI2fMEcSA
fetch code: vhk2
```

- Instructions to development kit of RK NPU and Q&A

```
link: https://pan.baidu.com/s/10w7R_q857uVEXq-88Pu-1g
fetch code: c66l
```

- Instructions to debug camera's sensor driver

```
link: https://pan.baidu.com/s/1rva6ZDj1x-T1rNcxV354KA
fetch code: z4uh
```

- RV1126&RV1109 Linux SDK Quick Start develop guide

```
link: https://pan.baidu.com/s/1iXwOdXH0jIR3iGQc0gluow
fetch code: t9o0
```

- Instructions to calibration process of RK ISP2

```
link: https://pan.baidu.com/s/1tZloen4B4jIIl2w1R2hWfg
fetch code: nrp3
```

- Instructions to calibration method of RK ISP2 base modules and tool usage

```
link: https://pan.baidu.com/s/1L0zSxInjqTyqDBesg4RL1w
fetch code: 8yc6
```

3.4 RV1109/RV1126 Development Tools

3.4.1 Windows Tools

Release Note: tools/windows/ToolsRelease.txt

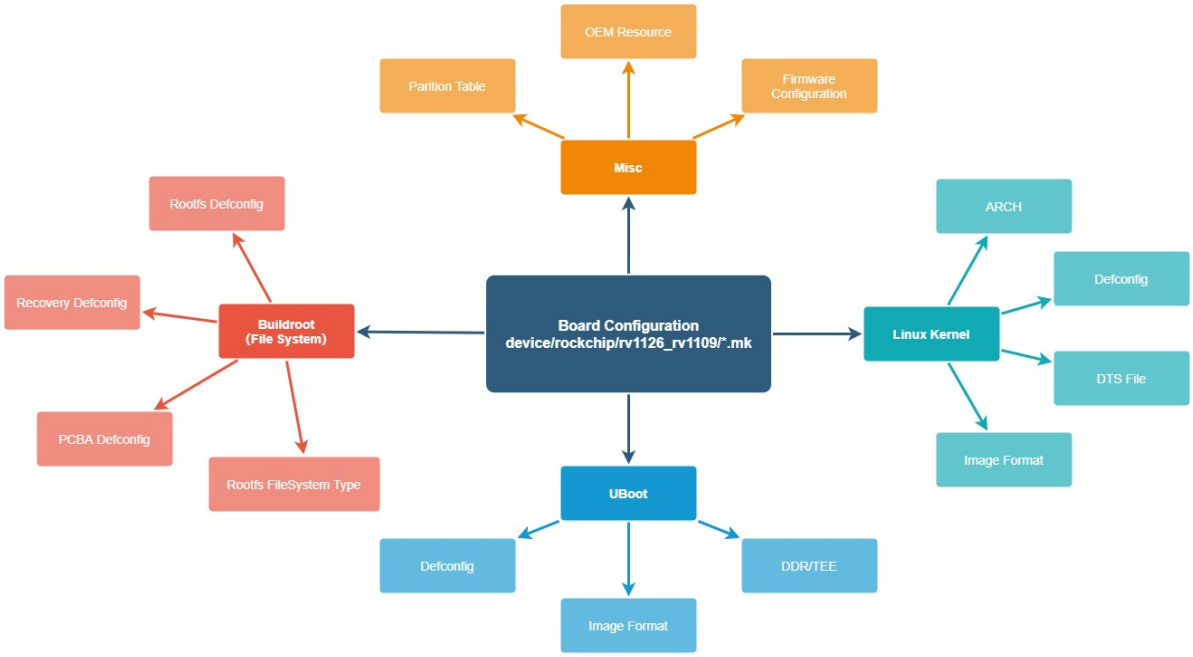
Tool's Name	Tool's Description
RKDevTool	discrete firmware upgrade and the entire update.img firmware upgrade tool
FactoryTool	factory production upgrade tool
SecureBootTool	firmware signing tool
efuseTool	efuse flash tool
RKDevInfoWriteTool	serial number burning tool
SDDiskTool	sd card firmware create tool
programmer_image_tool	build manufacture programmer firmware image for SPI NOR/SPI NAND/SLC NAND/eMMC
DriverAssitant	rockchip pc driver for upgrade tool
RKImageMaker	firmware package tool (generate update.img)
SpeakerPCBATool	soundbox PCBA test tool
RKDevTool_Release	rockchip firmware flash tool
ParameterTool	rockchip partition modify tool
RK_IPCamera_Tool	rockchip ipc camera search tool

3.4.2 Linux Tools

Release Note: tools/linux/ToolsRelease.txt

Tool's Name	Tool's Description
Linux_Pack_Firmware	firmware package tool (generate update.img)
Linux_Upgrade_Tool	rockchip firmware flash tool
Linux_SecureBoot	firmware signing tool
Firmware_Merger	SPI NOR firmware package tool (generate firmware.img for burner)
programmer_image_tool	build manufacture programmer firmware image for SPI NOR/SPI NAND/SLC NAND/eMMC

3.5 SDK Configuration Framework



4. SDK Building Introduction

Description of two methods of compiling SDK:

- One is to rely on the entire SDK environment to compile (the method introduced in this chapter)
- The other is to compile U-Boot, Linux Kernel, Rootfs and application libraries independently from the SDK (refer to the document for specific methods: **docs/RV1126_RV1109/Rockchip_RV1126_RV1109_Instruction_Linux_Separate_Building_EN.pdf**)

4.1 SDK Download Address

```
repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo \
    -u ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests \
    -b linux -m rv1126_rv1109_linux_release.xml

.repo/repo/repo sync -c -j4
```

4.1.1 SDK Sync and Log

4.1.1.1 SDK Sync

```
.repo/repo/repo sync -c -j4
repo: warning: Python 2 is no longer supported; Please upgrade to Python 3.6+.
repo: warning: Python 2 is no longer supported; Please upgrade to Python 3.6+.
Fetching projects: 100% (71/71), done.
info: A new version of repo is available

warning: project 'repo' branch 'stable' is not signed
warning: Skipped upgrade to unverified version
Checking out projects: 100% (71/71), done.
repo sync has finished successfully.
```

4.1.1.2 Get the Version of SDK

Run this command in the root directory of SDK:

```
realpath .repo/manifests/rv1126_rv1109_linux_release.xml

# e.g.  SDK version is v1.3.1
#       update time is 2020-09-21
# /home/rv1109-
SDK/.repo/manifests/rv1126_rv1109_linux/rv1126_rv1109_linux_v1.3.1_20200921.xml
```

4.1.1.3 Create the Branch Name Default for Every Project

```
.repo/repo/repo start default --all
repo: warning: Python 2 is no longer supported; Please upgrade to Python 3.6+.
repo: warning: Python 2 is no longer supported; Please upgrade to Python 3.6+.
Starting default: 100% (71/71), done.
```

4.2 Cross-compile Tool Introduction

There are two cross-compile tools in the SDK, as follow:

Directory	Introduction
prebuilts/gcc/linux-x86/arm/gcc-arm-8.3-2019.03-x86_64-arm-linux-gnueabihf	used to build rootfs and app
prebuilts/gcc/linux-x86/arm/gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf	used to build U-Boot and Linux kernel

Network disk address:
Link: `https://eyun.baidu.com/s/3eTwRktG`
Password: yaCM

4.3 To Select Board Configuration

4.3.1 The Directory of SDK Board Config (device/rockchip/rv1126_rv1109)

Board Configuration	Product Use	Storage Medium	EVB Board name
rockchip_38x38_emmc_defconfig	General IPC (Discrete power supply)	eMMC	TBD
rockchip_38x38_spi_nand_defconfig	General IPC (Discrete power supply)	SPI NAND	RV1126_RV1109_38X38_SPI_DDR3P216DD6_V10_20200511LXF
rockchip_38x38_spi_nand_defconfig	General IPC (Discrete power supply)	SPI NAND	RV1126_RV1109_IPC38_DEMO_V1.11_2020724LX
rockchip_38x38_spi_nand_ab_defconfig	General IPC (Discrete power supply), Boot-up with AB system	SPI NAND	RV1126_RV1109_IPC38_DEMO_V1.11_2020724LX
rockchip_robot_defconfig	Robot Sweeper IPC	eMMC	RV1126_RV1109_EVB_DDR3P216SD6_V13_20200630LXF
rockchip_tb_v13_defconfig	Door lock or doorbell products with battery	eMMC	RV1126_RV1109_EVB_DDR3P216SD6_V13_20200630LXF
rockchip_spi_nand_defconfig	General IPC	SPI NAND	RV1126_RV1109_EVB_DDR3P216SD6_V12_20200515KYY
rockchip_defconfig	General IPC	eMMC	RV1126_RV1109_EVB_DDR3P216SD6_V13_20200630LXF
rockchip_ab_v13_defconfig	General IPC, Boot-up with AB system	eMMC	RV1126_RV1109_EVB_DDR3P216SD6_V13_20200630LXF
rockchip_v12_defconfig	General IPC	eMMC	RV1126_RV1109_EVB_DDR3P216SD6_V12_20200515KYY
rockchip_slc_nand_v12_defconfig	General IPC	SLC NAND	RV1126_RV1109_EVB_DDR3P216SD6_V12_20200515KYY
rockchip_v10_v11_defconfig	General IPC	eMMC	RV1126_RV1109_EVB_DDR3P216SD6_V11_20200312LXF
rockchip_spi_nor_v12_defconfig	Base system and only RTSP stream to preview	SPI NOR	RV1126_RV1109_EVB_DDR3P216SD6_V12_20200515KYY
++++++++++++++++++++++++	++++++++++++++++++++++++++++++++	++++++++++++	++

Command of select board configure:

Method 1
`./build.sh` "path to board config file", for example:

select **General IPC** board configuration

```
./build.sh device/rockchip/rv1126_rv1109/rockchip_defconfig
```

select **Door lock or doorbell products with battery** board configuration, apply to EVB Board RV1126_RV1109_EVB_DDR3P216SD6_V13_20200630LXF

```
./build.sh device/rockchip/rv1126_rv1109/rockchip_tb_v13_defconfig
```

Method 2

```
./build.sh lunch

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

1. rockchip_defconfig
2. rockchip_38x38_emmc_defconfig
3. rockchip_38x38_spi_nand_ab_defconfig
4. rockchip_38x38_spi_nand_defconfig
5. rockchip_ab_v13_defconfig
6. rockchip_battery_evb_defconfig
7. rockchip_robot_defconfig
8. rockchip_sl_defconfig
9. rockchip_slc_nand_v12_defconfig
10. rockchip_sllock_defconfig
11. rockchip_spi_nand_defconfig
12. rockchip_spi_nor_tb_v13_defconfig
13. rockchip_spi_nor_v12_defconfig
```

```
14. rockchip_tb_v13_defconfig
15. rockchip_trailcamera_defconfig
16. rockchip_v10_v11_defconfig
17. rockchip_v12_defconfig
Which would you like? [1]: 1
```

4.4 To View Building Commands

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

```
./build.sh help
Usage: build.sh [OPTIONS]
Available options:
app-clean           - clean buildroot app
app-rebuild         - rebuild buildroot app
app-sync            - sync buildroot app
lunch               - choose defconfig
*_defconfig         - switch to specified defconfig
    Available defconfigs:
        rockchip_38x38_emmc_defconfig
        rockchip_38x38_spi_nand_ab_defconfig
        rockchip_38x38_spi_nand_defconfig
        rockchip_ab_v13_defconfig
        rockchip_battery_evb_defconfig
        rockchip_defconfig
        rockchip_robot_defconfig
        rockchip_sl_defconfig
        rockchip_slc_nand_v12_defconfig
        rockchip_sllock_defconfig
        rockchip_spi_nand_defconfig
        rockchip_spi_nor_tb_v13_defconfig
        rockchip_spi_nor_v12_defconfig
        rockchip_tb_v13_defconfig
        rockchip_trailcamera_defconfig
        rockchip_v10_v11_defconfig
        rockchip_v12_defconfig
olddefconfig        - resolve any unresolved symbols in .config
savedefconfig       - save current config to defconfig
menuconfig          - interactive curses-based configurator
kernel              - build kernel
modules             - build kernel modules
linux-headers       - build linux-headers
loader              - build loader (uboot|spl)
uboot               - build u-boot
spl                 - build spl
uefi                - build uefi
wifibt              - build Wifi/BT
rootfs              - build rootfs (default is buildroot)
buildroot           - build buildroot rootfs
yocto               - build yocto rootfs
debian              - build debian rootfs
recovery            - build recovery
pcba                - build PCBA
security_check      - check contidions for security features
createkeys          - build secureboot root keys
```

```
security_uboot     - build uboot with security paramter
security_boot      - build boot with security paramter
security_recovery  - build recovery with security paramter
security_rootfs    - build rootfs and some relevant images with security
paramter (just for dm-v)
firmware           - generate and check firmwares
updateimg          - build update image
otapackage         - build OTA update image
sdpackage          - build SDcard update image
all                - build all basic image
save               - save images and build info
allsave            - build all & firmware & updateimg & save
cleanall           - cleanup
post-rootfs        - trigger post-rootfs hook scripts
shell              - setup a shell for developing
help               - usage

Default option is 'allsave'.
```

To view detailed building commands for some modules, for example: `./build.sh kernel:cmds`

```
./build.sh kernel:cmds
Commands of building kernel:
export CROSS_COMPILE=<SDK_root_path>/prebuilts/gcc/linux-x86/arm/gcc-arm-8.3-
2019.03-x86_64-arm-linux-gnueabihf/bin/arm-rockchip830-linux-gnueabihf-
make -C kernel/ ARCH=arm -j65 rv1126_defconfig
make -C kernel/ ARCH=arm -j65 rv1126-evb-ddr3-v13.img
<SDK_root_path>/device/rockchip/common/scripts/mk-fitimage.sh kernel/zboot.img
boot.its kernel/arch/arm/boot/zImage
```

4.5 U-Boot Building

```
### to view detailed U-Boot build command
./build.sh uboot:cmds

### U-Boot Building Command
./build.sh uboot
```

4.5.1 Instructions to U-Boot Config

```
### use menuconfig to configure U-Boot, select config, save and exit.
### rv1126_defconfig can be found in the directory of u-boot/configs
### command format: make "RK_UBOOT_DEFCONFIG"_defconfig
### RK_UBOOT_DEFCONFIG define in the rockchip*_defconfig file which ./build.sh
select
cd u-boot
make rv1126_defconfig
make menuconfig

### save config to rv1126_defconfig
make savedefconfig
cp defconfig configs/rv1126_defconfig
```

4.6 Kernel Building

```
### to view detailed Kernel build command
./build.sh kernel:cmds

### use menuconfig to configure Kernel, select config, save and exit.
### e.g. EVB Board
cd kernel
make ARCH=arm rv1126_defconfig
make ARCH=arm menuconfig

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

4.6.1 Instructions to kernel config

```
### e.g. device/rockchip/rv1126_rv1109/rockchip_defconfig
./build.sh device/rockchip/rv1126_rv1109/rockchip_defconfig
cd kernel

### command format: make ARCH=arm "RK_KERNEL_DEFCONFIG"
"RK_KERNEL_DEFCONFIG_FRAGMENT"
### RK_KERNEL_DEFCONFIG and RK_KERNEL_DEFCONFIG_FRAGMENT is define in the
rockchip*_defconfig file which ./build.sh select
### RK_KERNEL_DEFCONFIG_FRAGMENT is optional, refer to rockchip*_defconfig
make ARCH=arm rv1126_defconfig
make ARCH=arm menuconfig

make ARCH=arm savedefconfig
cp defconfig arch/arm/configs/rv1126_defconfig
```

4.7 Recovery Building

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

4.7.1 Instructions to Recovery config

```
### 1. source the defconfig of buildroot
source envsetup.sh rockchip_rv1126_rv1109_recovery

### 2. use menuconfig to configure Recovery, select config, save and exit.
### e.g. disable recovery ui config is BR2_PACKAGE_RECOVERY_NO_UI (see
buildroot/package/rockchip/recovery/Config.in)
make menuconfig  # enter menuconfig, input "/" and goto search mode, input
BR2_PACKAGE_RECOVERY_NO_UI

### 3. save to recovery config file
###    ./buildroot/configs/rockchip_rv1126_rv1109_recovery_defconfig
make update-defconfig
```

NOTE: Recovery is a non-essential function, some board configuration will not be set.

4.8 Rootfs Building

```
### Rootfs building command
./build.sh rootfs
```

4.8.1 Instructions to Rootfs Config

```
### 1. source the defconfig of buildroot
source envsetup.sh rockchip_rv1126_rv1109

### 2. use menuconfig to configure Rootfs, select config, save and exit.
### e.g. ipc-daemon's config
###      a. find the config file for app/ipc-daemon
###         grep -lr "app/ipc-daemon" buildroot/package
###         buildroot/package/rockchip/ipc-daemon/ipc-daemon.mk
###         Corresponding configuration file: buildroot/package/rockchip/ipc-
daemon/Config.in
###
###      b. Get configuration name: BR2_PACKAGE_IPC_DAEMON
###         see: buildroot/package/rockchip/ipc-daemon/Config.in
make menuconfig  # enter menuconfig, input "/" and goto search mode, input
BR2_PACKAGE_IPC_DAEMON

### 3. save to rootfs config file
###    ./buildroot/configs/rockchip_rv1126_rv1109_defconfig
make update-defconfig
```

4.8.2 The Compilation of Projects Which under the Directory of app and external

```
# It needs to be compiled with 'Makefile' in the
'<SDK>/buildroot/output/rockchip_*' directory. ('rockchip_*' is the
corresponding 'defconfig')
# e.g. Compile 'external/rkmedia' and 'app/rkipc' in
'rockchip_rv1126_rv1109_defconfig' configuration
cd ./buildroot/output/rockchip_rv1126_rv1109
make rkmedia-dirclean && make rkmedia-rebuild
make rkipc-dirclean && make rkipc-rebuild
```

[NOTE]: /app and /external are also the package of Buildroot.

4.9 Instructions to Build BSP Libraries

The SDK BSP package only contains audio and video codec libraries, NPU libraries, and header files.
Note: BSP package does not include file system.

```
source envsetup.sh rockchip_rv1126_rv1109_libs

make -j12
```

Compile the directory generated by the BSP `buildroot/output/rockchip_rv1126_rv1109_libs/BSP`

```
tree buildroot/output/rockchip_rv1126_rv1109_libs/BSP/
buildroot/output/rockchip_rv1126_rv1109_libs/BSP/
├── example
│   ├── common
│   ├── iqfiles
│   ├── librtsp
│   ├── multi_audio_test
│   ├── rknn_model
│   ├── stressTest
│   └── vqefiles
├── include
│   ├── rga
│   ├── rkaiq
│   └── rkmedia
├── lib
└── npu
    ├── include
    ├── ko
    └── lib
```

4.10 Firmware Package

Firmware packaging command: `./mkfirmware.sh`

Firmware directory: rockdev

4.11 Full Automatic Building

Enter the project root directory and execute the following command to automatically complete all buildings:

```
./build.sh all # Only build with u-Boot, kernel, Rootfs and Recovery
               # then use ./mkfirmware.sh to package firmware.

./build.sh     # build with ./build.sh all, and then do these as follow:
               # 1. package firmware to directory "rockdev"
               # 2. package update.img
               # 3. copy rockdev directory to
'output/<dts_name>/BUILDROOT/<timestamp>/IMAGES'
               # 4. copy the patches of modules to
'output/<dts_name>/BUILDROOT/<timestamp>/PATCHES'
               # NOTE: ./build.sh allsave is the same as ./build.sh
```

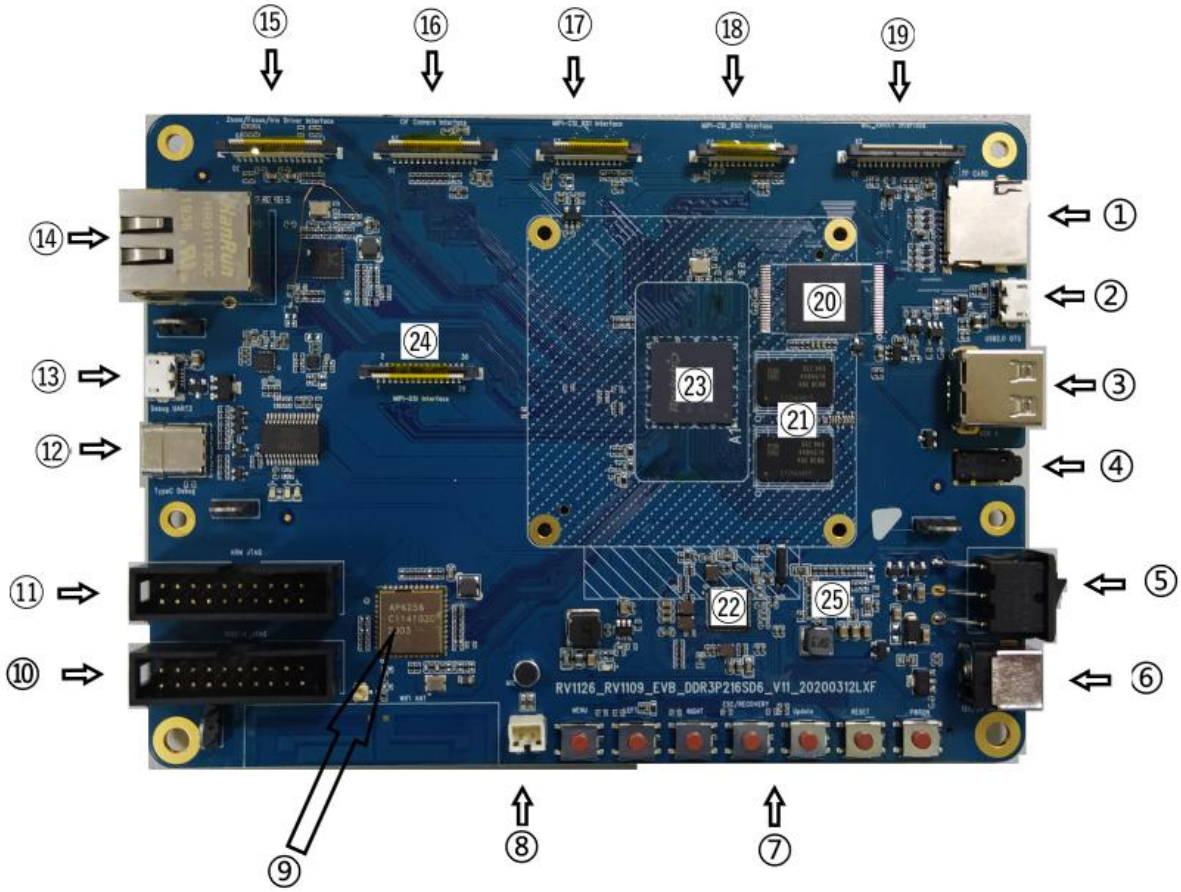
5. Application Development Suggestions

It is recommended to develop applications by reference to the `external/rkmedia/examples`

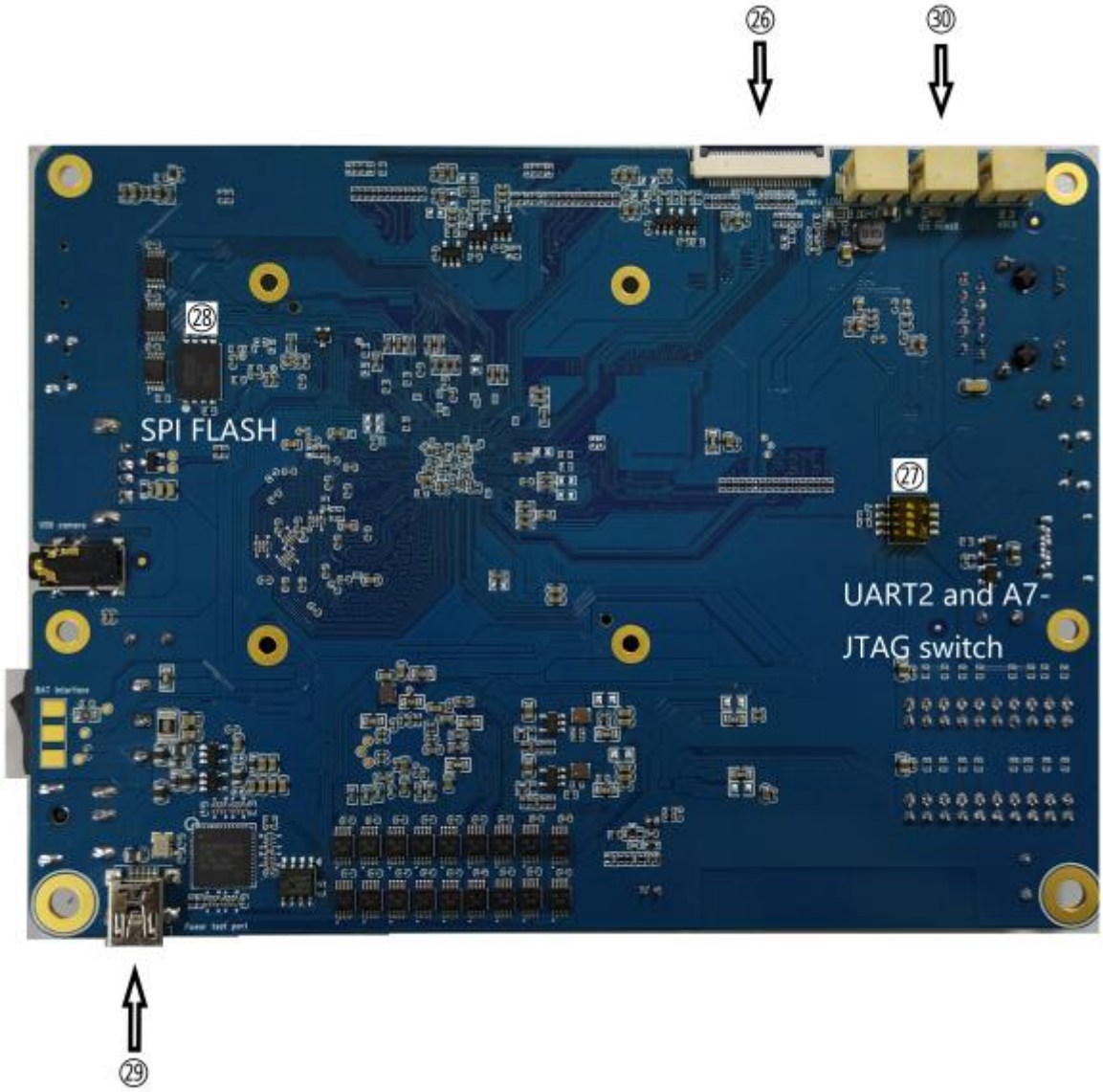
app/rkipc is just realized a simple IPC function, only for demonstration.

6. Upgrade Introduction

6.1 TOP Surface of the EVB



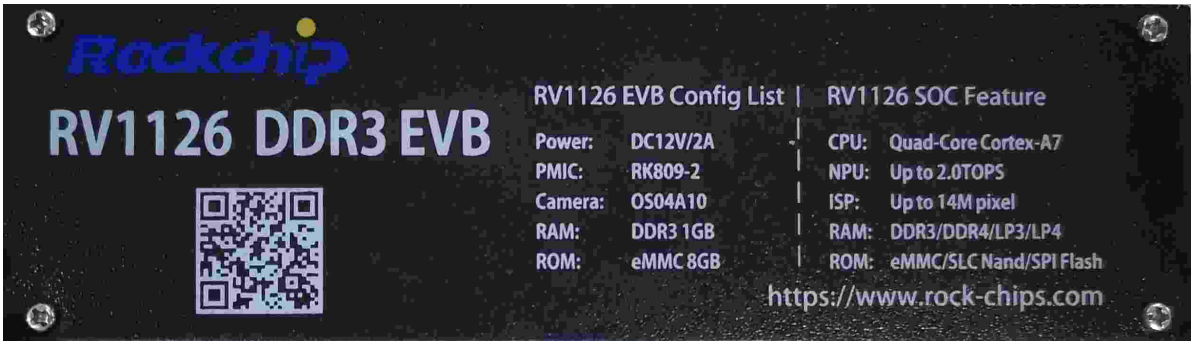
6.2 Bottom Surface of the EVB



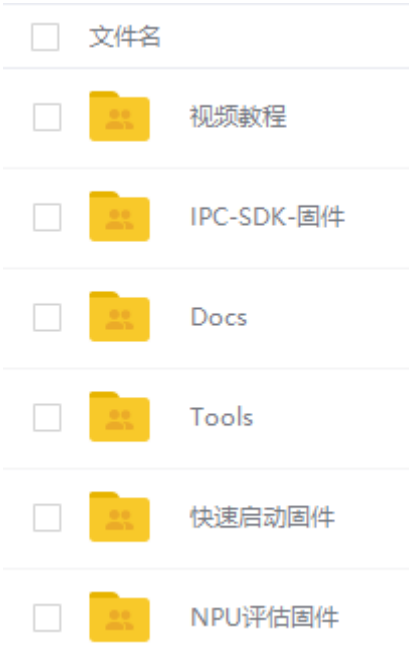
6.3 EVB Function Table

Item	Function Part	Requirement
1	TF Card	Normal recognition TF Card
2	USB Micro-B Port	Can recognize ADB device, can download images
3	USB Type-A Port	Can recognize device, and function normally
4	USB camera input	Normal recognition USB camera
5	Power swtich	The 12V power supply input by the DC adapter can be controlled ON/OFF by the boat switch.
6	12V power supply input	
7	KEY BAORD	All the buttons function normally
8	CLASS D output	speakers function normal
9	WI-FI/BT	AP6256 module functions normally
10	RISC-V JTAG/ A7-JTAG	Chip verification and debugging ， Tpyec is only used for chip verification
11	NC	
12	TPYEC	
13	USB Micro-B Port	Serial port input and output normally
14	Ethernet	Normal network connection
15	Zoom/ Iris Driver Interface	The EVB reserves Zoom/Focus/Iris/IRCUT connector as shown below, which is convenient for customers to debug CAMERA device.
16	CIF camera	Camera works normally, CIF camera input by default
17	MIPI Camera 1	Camera works normally, MIPI camera input by default
18	MIPI Camera 2	Camera works normally, MIPI camera input by default
19	MIC-ARRAY	Mic device input
20	eMMC Flash	Can normally recognize 16GByte
21	DDR DDR3	Can recognize total size 8Gbit
22	PMIC RK809-2	Output of each power supply is normal, accurate battery volume detection
23	CPU	RV1126_RV1109
24	MIPI panel	Screen image displays normally
25	BQ24171	2-cell battery normal charging and discharging
Bottom Surface		
26	BT1120 Camera	Camera works normally, MIPI camera input by default
27	Function switching	UART2 and a7-jtag function switching
28	SPI flash	Verify SPI flash function
29	USB Micro-B Port	For power consumption test
30	Camera_LED Drive output	Warm up lamp drive

6.4 Bottom Surface of EVB Sensor board



Scan the QR code which on sensor board or visit `https://eyun.baidu.com/s/3nwlh7iL` to get start with the EVB.

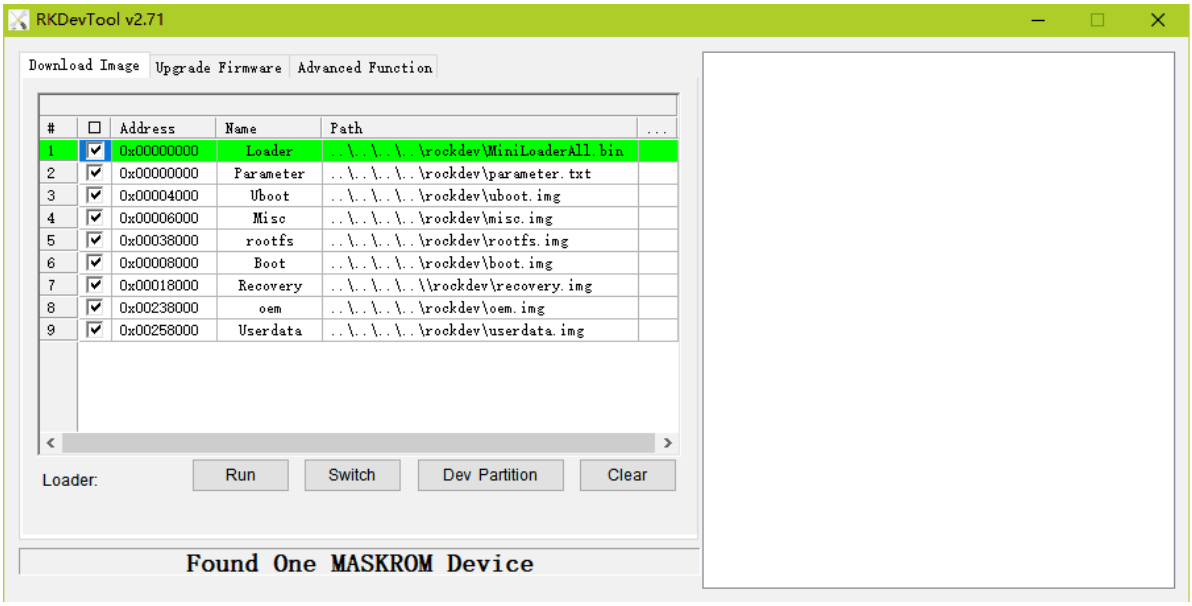


6.5 Windows Upgrade Introduction

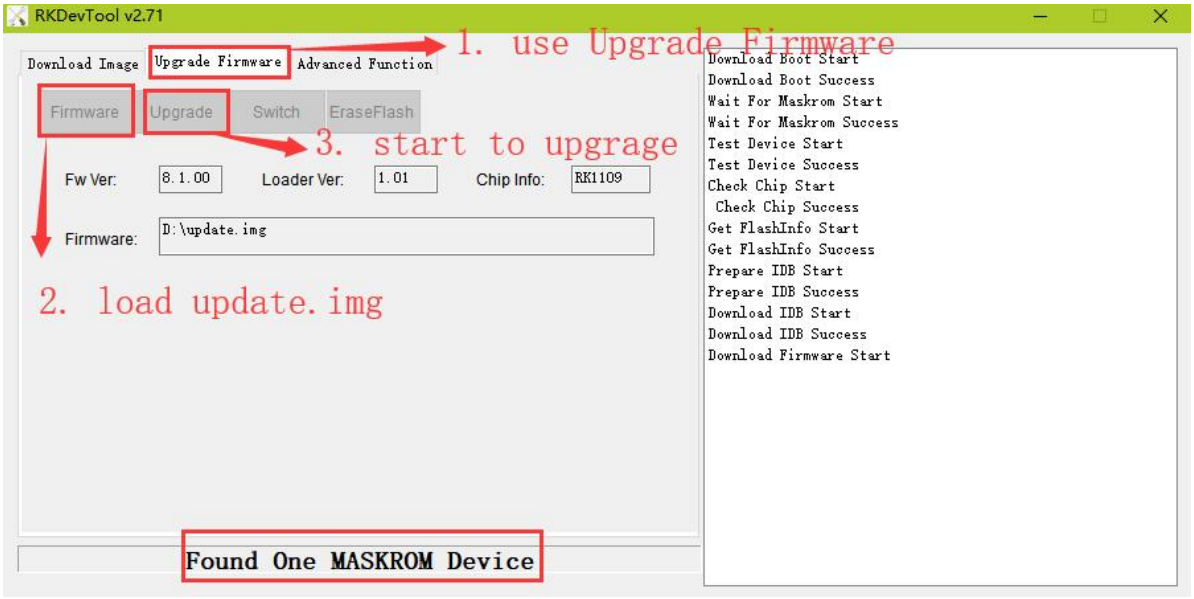
The SDK provides a windows flash tool (this tool should be V2.78 or later version) which is located in project root directory:

```
tools/
├── windows/RKDevTool
```

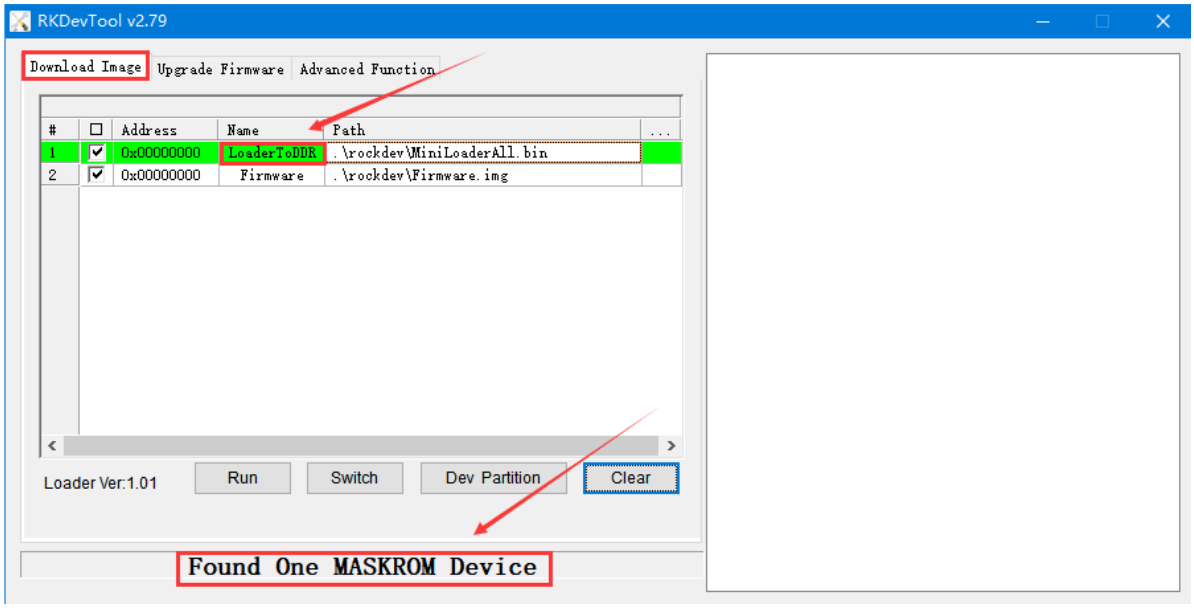
As shown below, after building and generating the firmware, device needs to enter MASKROM or BootROM mode for flashing. After connecting USB cable, long press the "Update" button and press "RESET" button at the same time and then release, device will enter MASKROM mode. Then you should load the paths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



The method to flash update.img:



The method to flash SPI NOR Firmware.img:



Note:

1. In addition to MiniLoader All.bin and parameter.txt, the actual partition to be burned is based on rockdev / parameter.txt configuration.
2. before upgrade, please install the latest USB driver, which is in the below directory:

```
<SDK>/tools/windows/DriverAssitant_v4.91.zip
```

6.6 Linux Upgrade Introduction

The Linux upgrade tool (Linux_Upgrade_Tool should be v1.57 or later versions) is located in "tools/linux" directory. Please make sure your board is connected to MASKROM/loader rockusb, if the generated firmware is in rockdev directory, upgrade commands are as below:

```
### In addition to MiniLoader All.bin and parameter.txt, the actual partition
to be burned is based on rockdev / parameter.txt configuration.
sudo ./upgrade_tool ul 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 firmware after packaging:

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

Upgrade the whole Firmware.img for SPI NOR:

```
sudo ./upgrade_tool db rockdev/MiniLoaderAll.bin
sudo ./upgrade_tool wl 0x0 rockdev/Firmware.img
sudo ./upgrade_tool rd
```

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

```
./rkflash.sh
```

7. EVB Function Introduction and Precaution

7.1 The precaution of GPIO Power Design

The IO level of the controller power domain must be consistent with the IO level of the connected peripheral chip, and the voltage configuration of software must be consistent with the voltage of hardware to avoid GPIO damage.

 Note:

About matching of GPIO power domain and IO level:

PMUIO0_VDD, PMUIO1_VDD, VCCIO1_VDD, VCCIO2_VDD, VCCIO3_VDD, VCCIO4_VDD, VCCIO5_VDD, VCCIO6_VDD, VCCIO7_VDD, voltage of these GPIO power domain must be consistent with the IO level voltage of the connected peripheral to avoid GPIO damage.

Also need to note that the voltage configuration of software should be consistent with the voltage of hardware: For example, if hardware IO level is connected to 1.8V, the voltage configuration of software should be configured to 1.8V accordingly; if hardware IO level should be connected to 3.3V, and the voltage configuration of software should also be configured to 3.3V to avoid GPIO damage.

7.2 How to Access 2 RTSP and 1 RTMP Network Stream

The EVB supports the following functions:

- Support 2 RTSP and 1 RTMP network stream
- Support 1280x720 local screen display
- Support to save the main stream to the device
- Support access device from web

Connect a network cable to the network port of the EVB, power on and start. It will obtain the IP address automatically by default.

7.2.1 Get Device IP Address by Serial Port or ADB of the EVB

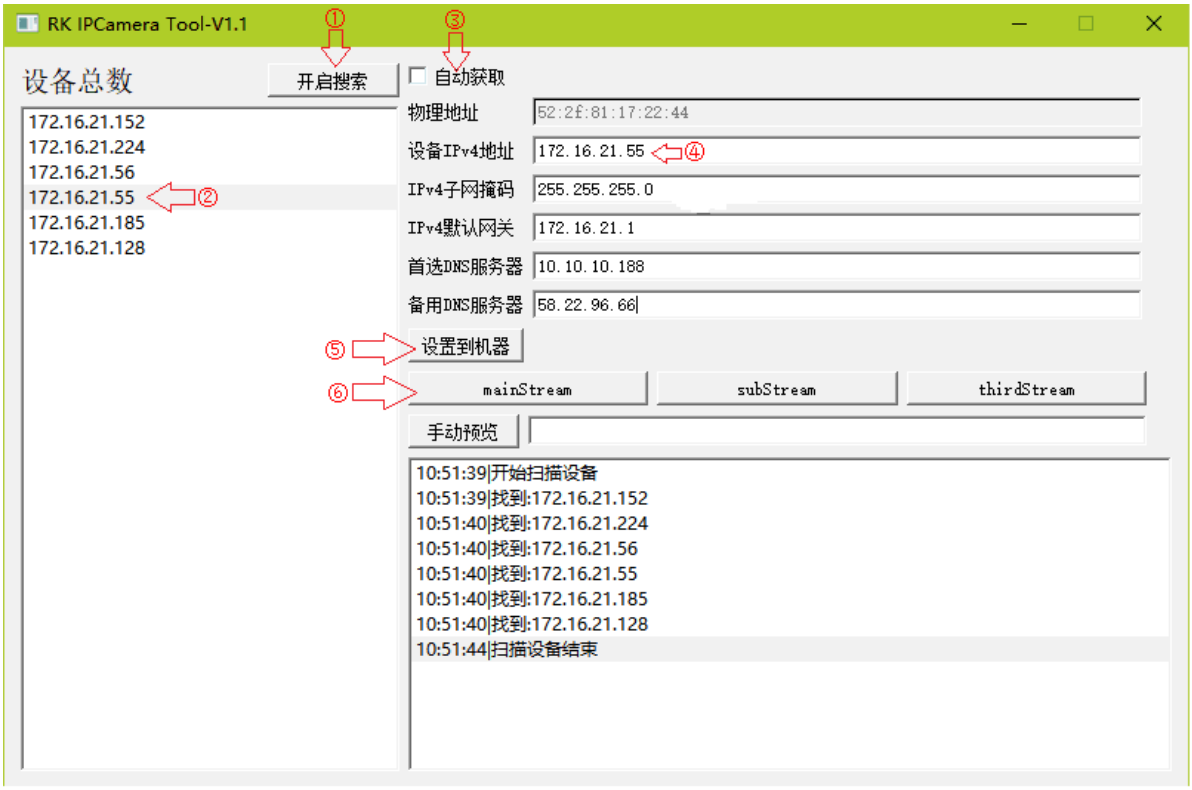
```
ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 02:E0:F9:16:7E:E9
          inet addr:172.16.21.218  Bcast:172.16.21.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:199225 errors:0 dropped:2231 overruns:0 frame:0
          TX packets:372371 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:20874811 (19.9 MiB)  TX bytes:522220899 (498.0 MiB)
          Interrupt:56
```

Connect to the EVB through the serial port, you have to configure as follows:

```
Baud rate: 1500000
Data bits: 8
Stop bit: 1
Parity: none
Flow control: none
```

7.2.2 Get Device IP Address by RK IPCamera Tool

Install the tool in the SDK directory `tools/windows/RK_IPCamera_Tool-V1.1.zip`. Open the tool and connect the EVB board to the computer through the network port. In the local area network, check the RK IPCamera Tool device list to obtain the device IP address.



Note:

- Step 1: click "开启搜索" to search devices
- Step 2: select a device
- Step 3: cancel "自动获取" and change to static IP
- Step 4: set a static IP
- Step 5: set the IP to device
- Step 6: open to preview

7.2.3 Access Network Stream

Use a player that supports RTSP or RTMP to access, for example (VLC player).

RTSP access address:

- rtsp://**IP address of the device**/live/0
- rtsp://**IP address of the device**/live/1

RTMP access preview via web.

7.3 How to Access Device Information via Web

Open a web browser (Chrome browser is recommended) to access the address:

```
http://IP address of the device
```

For detailed operation instructions on the web, please refer to the documents under the SDK docs directory which named Rockchip_Instructions_Linux_Web_Configuration_CN.pdf.

7.4 How to Test Face Recognition Function

Note: The rkipc of rkmedia version does not support this function.

Use a player to access RTSP main stream: rtsp://**IP address of the device**/live/0

Face Recognition is closed by default. We can visit on `http://IP address of the device` and "Config" --> "INTELLIGENCE ANALYSIS" to enable face recognition.
The default authorization test time of the SDK's face recognition function is 30 ~ 60 minutes. When the authorization is invalid, the main stream preview will prompt "gace algorithm software is not authorized", and you have to restart to test again.

7.5 How to Debug With EVB via Network

7.5.1 Debug With SSH

Connect EVB with network, get EVB board's IP address with the Chapter 5.1.2 Get Device IP Address by RK IPCamera Tool. Ensure that the PC can ping the EVB board.

```
### Clean last login message (EVB IP address: 192.168.1.159)
ssh-keygen -f "$HOME/.ssh/known_hosts" -R 192.168.1.159
### Command of SSH
ssh root@192.168.1.159
### input the default passwd: rockchip
```

7.5.2 Debug With SCP

```
### Upload the test-file from PC to EVB board dirctory /userdata
scp test-file root@192.168.1.159:/userdata/
root@192.168.1.159's password:
### input the default passwd: rockchip

### Download the EVB file (/userdata/test-file) to PC
scp root@192.168.1.159:/userdata/test-file test-file
root@192.168.1.159's password:
### input the default passwd: rockchip
```

7.5.3 Debug with ADB via TCP/IP

```
### e.g. EVB IP address: 192.168.1.159
adb connect 192.168.1.159

adb devices
List of devices attached
192.168.1.159:5555      device

### login on EVB via ADB
adb -s 192.168.1.159:5555 shell
```

```
### Upload the test-file from PC to EVB board dirctory /userdata
adb -s 192.168.1.159:5555 push test-file /userdata/

### Download the EVB file (/userdata/test-file) to PC
adb -s 192.168.1.159:5555 pull /userdata/test-file test-file
```

7.6 SPI NAND/SLC NAND ubi filesystem introduction

7.6.1 Introduction to ubi filesystem of rootfs

The rootfs of Nand Flash is ubifs, SDK default configure Nand Flash information is Page Size 2KB and Block Size 128KB.
To modify board config: `device/rockchip/rv1126_rv1109/rockchip*_defconfig`.

```
# Set ubifs page size, 2048(2KB) or 4096(4KB)
# Option.
RK_UBI_PAGE_SIZE=2048

# Set ubifs block size, 0x20000(128KB) or 0x40000(256KB)
# Option.
RK_UBI_BLOCK_SIZE=0x20000

# Set 2 extra partition
RK_EXTRA_PARTITION_NUM=2

# Set userdata partition size (byte) if define RK_USERDATA_DIR
# MUST, if userdata partition is grow partition.
RK_EXTRA_PARTITION_1_DEV="userdata"
RK_EXTRA_PARTITION_1_FSTYPE="ubi"
RK_EXTRA_PARTITION_1_SIZE="60M"

# Set oem partition size (byte)
# Option. if not set, it will get from parameter auto.
RK_EXTRA_PARTITION_2_DEV="oem"
RK_EXTRA_PARTITION_2_FSTYPE="ubi"
RK_EXTRA_PARTITION_2_SIZE="30M"
```

The rootfs is ubifs default, if want to use squashfs, steps are as follows:

- Configure buildroot/configs/rockchip_rv1126_rv1109_spi_nand_defconfig

```
diff --git a/configs/rockchip_rv1126_rv1109_spi_nand_defconfig
b/configs/rockchip_rv1126_rv1109_spi_nand_defconfig
index 5da9b25935..8af9226920 100644
--- a/configs/rockchip_rv1126_rv1109_spi_nand_defconfig
+++ b/configs/rockchip_rv1126_rv1109_spi_nand_defconfig
@@ -41,6 +41,8 @@ BR2_PACKAGE_RK_OEM=y
 BR2_PACKAGE_RK_OEM_RESOURCE_DIR="$(TOPDIR)/../device/rockchip/oem/oem_ipc"
 BR2_PACKAGE_RK_OEM_IMAGE_FILESYSTEM_TYPE="ubi"
 BR2_PACKAGE_RK_OEM_IMAGE_PARTITION_SIZE=0x6400000
+BR2_PACKAGE_ROOTFS_UBI_USE_CUSTOM_FILESYSTEM=y
+BR2_PACKAGE_ROOTFS_UBI_CUSTOM_FILESYSTEM="squashfs"
```

```
 BR2_PACKAGE_CAMERA_ENGINE_RKAIQ=y
 BR2_PACKAGE_CAMERA_ENGINE_RKAIQ_IQFILE="os04a10_CMK-OT1607-FV1_M12-40IRC-4MP-
F16.xml"
 BR2_PACKAGE_IPC_DAEMON=y
@@ -79,4 +81,5 @@ BR2_PACKAGE_NGINX=y
 BR2_PACKAGE_NGINX_HTTP_SSL_MODULE=y
 BR2_PACKAGE_NGINX_DEBUG=y
 BR2_PACKAGE_NGINX_RTMP=y
+BR2_TARGET_ROOTFS_SQUASHFS4_XZ=y
 BR2_TARGET_ROOTFS_UBIFS_MAXLEBCNT=4096
```

- Config the bootargs paramerter in kernel dts as follows:

```
ubi.mtd=3 ubi.block=0,rootfs root=/dev/ubiblock0_0 rootfstype=squashfs    /*
mount SquashFS on UBI block */
```

NOTICE: ubi.mtd=3 (3 is the number of rootfs in the partition and the first partition number is 0)

7.6.2 Introduction to oem and userdata Partition Used in ubifs

The SDK default OEM is packaged in Buildroot as a UBI image.
The userdata partition default is not packaged, when system boot up userdata will fotmat to ubifs auto.

If configure RK_OEM_DIR (RK_OEM_BUILDIN_BUILDROOT is not defined) or RK_USERDATA_DIR in the rockchip*_defconfig, then use `./mkfirmware.sh` which in the root directory of SDK.
RK_OEM_DIR define as the directory which in the device/rockchip/oem/.
RK_USERDATA_DIR define as the directory which in the device/rockchip/userdata/.

The detailed UBI filesystem image document:
/docs/Linux/ApplicationNote/Rockchip_Developer_Guide_Linux_Nand_Flash_Open_Source_Solution_EN.pdf

7.7 Introduced to the Usage of U-Boot's tftp

7.7.1 Configure U-Boot's ethernet phy

The ethernet phy which U-Boot support is RTL8211F, default. The initialization of U-Boot's ethernet will read dtb from kernel, if fail to read, U-Boot will read from U-Boot's dtb. So we will configure the node gmac of U-Boot's dtb if the phy is not RTL8211F.

The follow is the reference:
Note: the quote node in the U-Boot's dts (e.g. gpio2/rmiim1_pins/gmac_clk_m1_pins) MUST add "u-boot,dm-pre-reloc;"

```
diff --git a/arch/arm/dts/rv1126-pinctrl.dtsi b/arch/arm/dts/rv1126-
pinctrl.dtsi
index 67f7c742b8..967598b4fb 100644
--- a/arch/arm/dts/rv1126-pinctrl.dtsi
+++ b/arch/arm/dts/rv1126-pinctrl.dtsi
@@ -1122,6 +1122,11 @@
                                /* clk_out_ethernet_m1 */
                                <2 RK_PC5 2 &pcfg_pull_none>;
                };
+               gmac_clk_m1_pins: gmac-clk-m1-pins {
```

```
+                       rockchip,pins =
+                               /* rgmii_clk_m1 */
+                               <2 RK_PB7 2 &pcfg_pull_none>;
+               };
        };
        sdmmc0 {
                sdmmc0_bus4: sdmmc0-bus4 {
diff --git a/arch/arm/dts/rv1126-u-boot.dtsi b/arch/arm/dts/rv1126-u-boot.dtsi
index 01547feff6..baf8509946 100644
--- a/arch/arm/dts/rv1126-u-boot.dtsi
+++ b/arch/arm/dts/rv1126-u-boot.dtsi
@@ -166,26 +166,37 @@
        status = "okay";
 };

+&gpio2 {
+       u-boot,dm-pre-reloc;
+       status = "okay";
+};
+
+&rmiim1_pins {
+       u-boot,dm-pre-reloc;
+       status = "okay";
+};
+
+&gmac_clk_m1_pins{
+       u-boot,dm-pre-reloc;
+       status = "okay";
+};
+
 &gmac {
        u-boot,dm-pre-reloc;

-       phy-mode = "rgmii";
-       clock_in_out = "input";
+       phy-mode = "rmii";
+       clock_in_out = "output";

-       snps,reset-gpio = <&gpio3 RK_PA0 GPIO_ACTIVE_LOW>;
+       snps,reset-gpio = <&gpio2 RK_PA5 GPIO_ACTIVE_LOW>;
        snps,reset-active-low;
-       /* Reset time is 20ms, 100ms for rtl8211f */
-       snps,reset-delays-us = <0 20000 100000>;
+       snps,reset-delays-us = <0 50000 50000>;

-       assigned-clocks = <&cru CLK_GMAC_SRC>, <&cru CLK_GMAC_TX_RX>, <&cru
CLK_GMAC_ETHERNET_OUT>;
-       assigned-clock-parents = <&cru CLK_GMAC_SRC_M1>, <&cru RGMII_MODE_CLK>;
-       assigned-clock-rates = <125000000>, <0>, <25000000>;
+       assigned-clocks = <&cru CLK_GMAC_SRC>, <&cru CLK_GMAC_TX_RX>;
+       assigned-clock-rates = <50000000>;
+       assigned-clock-parents = <&cru CLK_GMAC_SRC_M1>, <&cru RMII_MODE_CLK>;

        pinctrl-names = "default";
-       pinctrl-0 = <&rgmiim1_pins &clk_out_ethernetm1_pins>;
-
-       tx_delay = <0x2a>;
-       rx_delay = <0x1a>;
```

```
+       pinctrl-0 = <&rmiim1_pins &gmac_clk_m1_pins>;

        phy-handle = <&phy>;
        status = "okay";
```

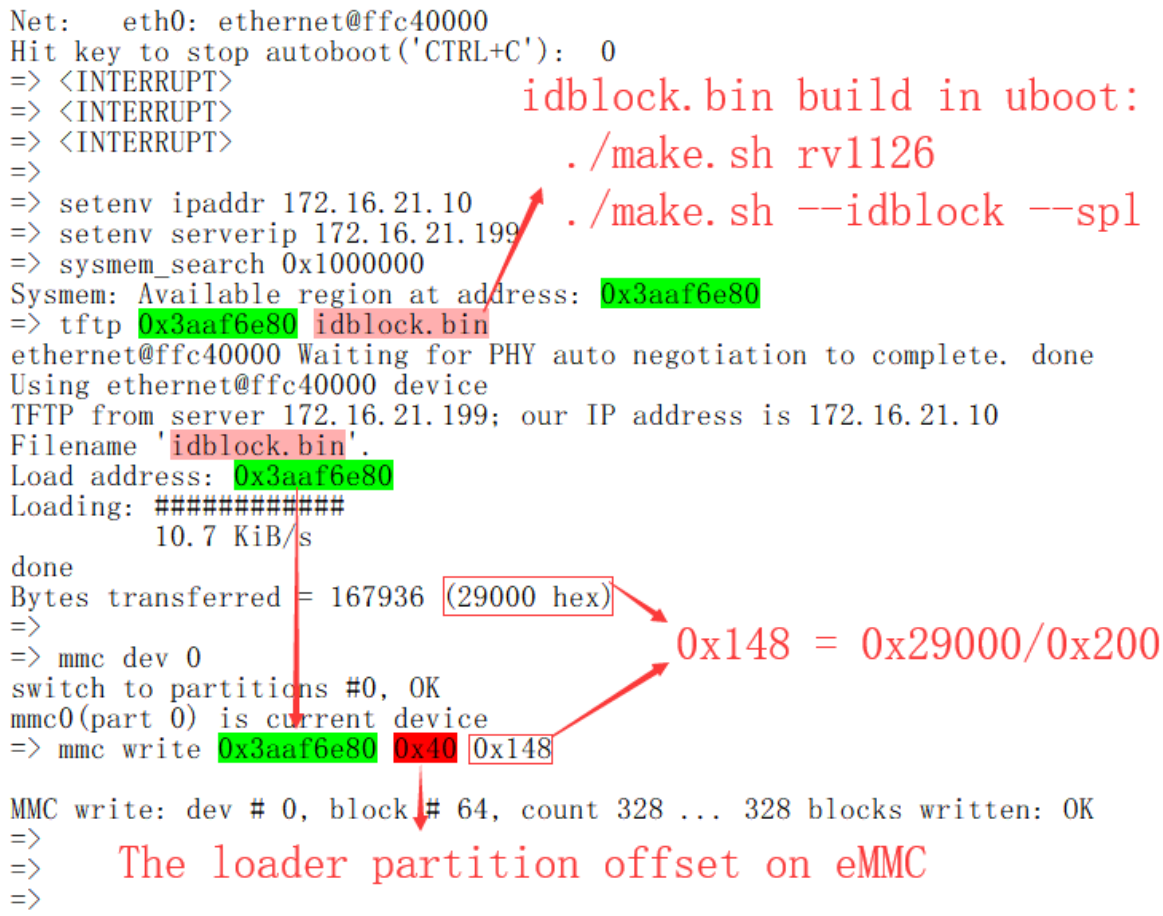
7.7.2 The Usage of U-Boot's tftp Download

Use the command `sysmem_search` to get a free memory address and then set IP address, finally use tftp download file.

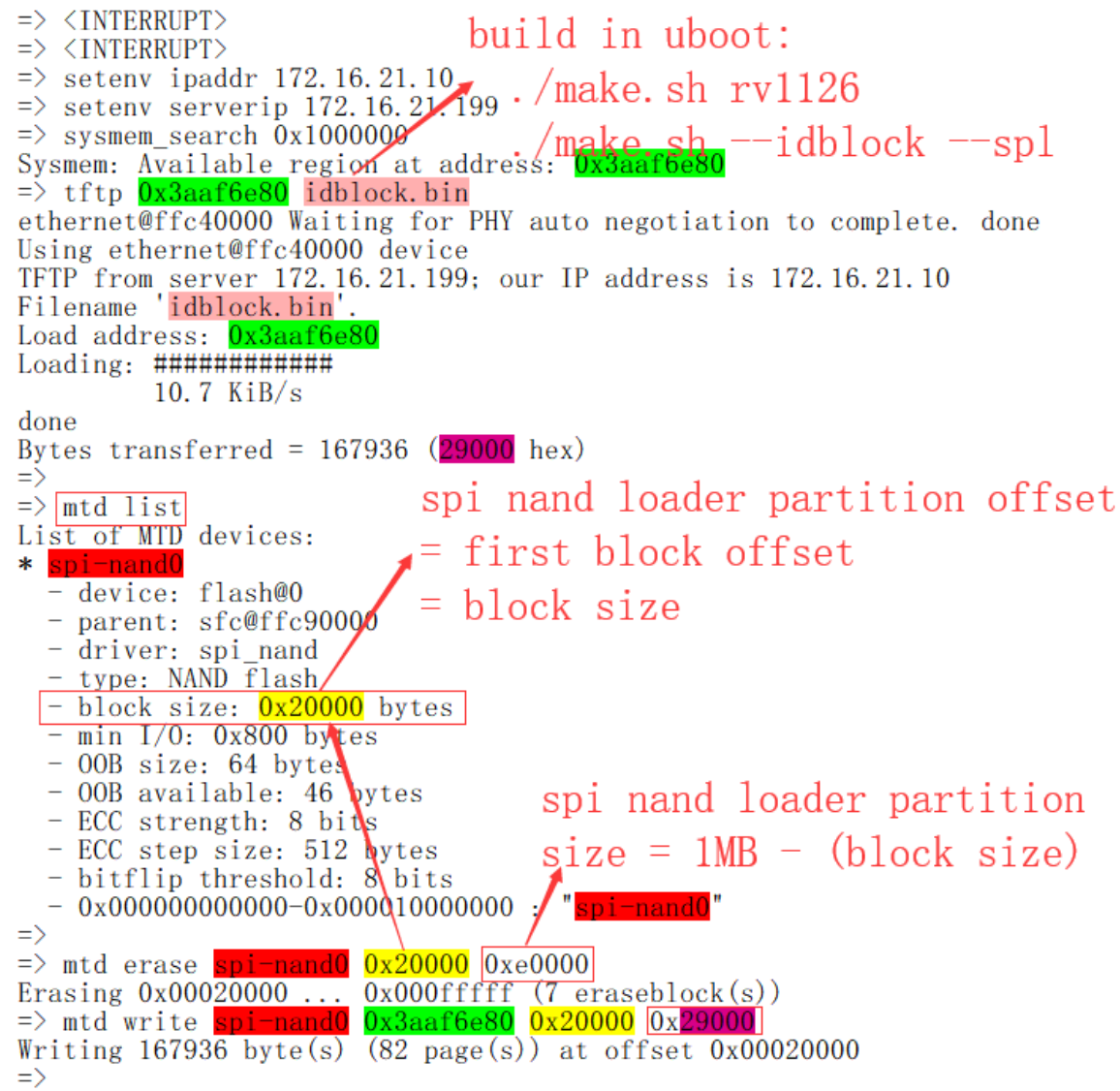
```
Hit key to stop autoboot('CTRL+C'):  0
=> <INTERRUPT>
=> <INTERRUPT>
=>
=> sysmem_search
sysmem_search - Search a available sysmem region

Usage:
sysmem_search <size in hex>
=> sysmem_search 0x6400000
Sysmem: Available region at address: 0x356f6cc0
=> setenv ipaddr 172.16.21.47
=> setenv serverip 172.16.21.199
=> tftp 0x356f6cc0 uboot.img
ethernet@ffc40000 Waiting for PHY auto negotiation to complete. done
Using ethernet@ffc40000 device
TFTP from server 172.16.21.199; our IP address is 172.16.21.47
Filename 'uboot.img'.
Load address: 0x356f6cc0
Loading: ##################################################################
         ##################################################################
         ######################
         139.6 KiB/s
done
Bytes transferred = 2228224 (220000 hex)
=>
```

7.7.2.1 Upgrede Loader with tftp for eMMC Storage



7.7.2.2 Upgrede Loader with tftp for SPI NAND Storage




```
=> <INTERRUPT>
=> <INTERRUPT>
=> setenv ipaddr 172.16.21.10
=> setenv serverip 172.16.21.199
=> sysmem_search 0x1000000
Sysmem: Available region at address: 0x3aaf6e80
=> tftp 0x3aaf6e80 idblock.bin
ethernet@ffc40000 Waiting for PHY auto negotiation to complete. done
Using ethernet@ffc40000 device
TFTP from server 172.16.21.199; our IP address is 172.16.21.10
Filename 'idblock.bin'.
Load address: 0x3aaf6e80
Loading: #############
         10.7 KiB/s
done
Bytes transferred = 167936 (29000 hex)
=>
=> mtd list
List of MTD devices:
* spi-nand0
  - device: flash@0
  - parent: sfc@ffc90000
  - driver: spi_nand
  - type: NAND flash
  - block size: 0x20000 bytes
  - min I/O: 0x800 bytes
  - OOB size: 64 bytes
  - OOB available: 46 bytes
  - ECC strength: 8 bits
  - ECC step size: 512 bytes
  - bitflip threshold: 8 bits
  - 0x000000000000-0x000010000000 : "spi-nand0"
=>
=> mtd erase spi-nand0 0x20000 0xe0000
Erasing 0x00020000 ... 0x000fffff (7 eraseblock(s))
=> mtd write spi-nand0 0x3aaf6e80 0x20000 0x29000
Writing 167936 byte(s) (82 page(s)) at offset 0x00020000
=>
```

NOTICE: SLC NAND is not support tftp download idblock.bin to upgrade loader temporarily.

7.8 Battery EVB DDR Startup Failure Solution

- Locate the config file in the U-Boot according to RK_UBOOT_CFG in the boardconfig. The config file path is `<SDK>/u-boot/configs/${RK_UBOOT_CFG}.config`
- Change the CONFIG_LOADER_INI parameter in the config file based on the actual DDR type. The ini file in the parameter must exist in `<SDK>/rkbin/RKBOOT/`

7.9 Booting from 128M DDR Is Supported

- Update SDK version to V2.2.0
- U-Boot configure `CONFIG_TRUST_INI="RV1126TOS_MINI_MEM.ini"`

The reference of SPI NOR configuration is as follows:

```
diff --git a/configs/rv1126-spi-nor-tiny_defconfig b/configs/rv1126-spi-nor-
tiny_defconfig
index 9b46da7c5c..b84d3a6411 100644
--- a/configs/rv1126-spi-nor-tiny_defconfig
+++ b/configs/rv1126-spi-nor-tiny_defconfig
@@ -14,7 +14,7 @@ CONFIG_ROCKCHIP_FIT_IMAGE_PACK=y
 CONFIG_ROCKCHIP_UART_MUX_SEL_M=2
 # CONFIG_GICV2 is not set
 CONFIG_LOADER_INI="RV1126MINIALL_SPI_NOR_TINY.ini"
-CONFIG_TRUST_INI="RV1126TOS_SPI_NOR_TINY.ini"
+CONFIG_TRUST_INI="RV1126TOS_MINI_MEM.ini"
 CONFIG_SPL_SERIAL_SUPPORT=y
 CONFIG_SPL_DRIVERS_MISC_SUPPORT=y
 CONFIG_TARGET_EVB_RV1126=y
```

7.10 Support FIT Image Package

```
# support package recovery image as FIT image format , Configure
rockchip*_defconfig:
export RK_RECOVERY_FIT_ITS=boot4recovery.its
```

```
# support package kernel image as FIT image format , Configure
rockchip*_defconfig:
export RK_KERNEL_FIT_ITS=boot.its
```

7.11 Filesystem Type Selection

Storage Medium	Filesystem Type
eMMC	Readable and Writable: ext2/ext4 Read-only: squashfs
SPI NOR	Readable and Writable: jffs2 Read-only: squashfs
SPI NAND	Readable and Writable: ubifs Read-only: squashfs
SLC NAND	Readable and Writable: ubifs Read-only: squashfs

Note: Readable and writable file system images are large and require modifications to the partition size.

7.11.1 SPI NOR and eMMC Filesystem Configurations

The default filesystem of SDK is squashfs, Here are the steps to modify:

1. Enter the directory of device/rockchip, configure rockchip*_defconfig of RK_ROOTFS_TYPE

```
diff --git a/rv1126_rv1109/rockchip_defconfig
b/rv1126_rv1109/rockchip_defconfig
index 14e9c67..f9b894f 100644
--- a/rv1126_rv1109/rockchip_defconfig
+++ b/rv1126_rv1109/rockchip_defconfig
@@ -1,4 +1,4 @@
-RK_ROOTFS_TYPE="squashfs"
+RK_ROOTFS_TYPE="ext4"
 RK_KERNEL_DTS_NAME="rv1126-evb-ddr3-v13"
 RK_UBOOT_OPTS="--spl-new"
 RK_BOOT_IMG="zboot.img"
```

2. Run the command at the root directory of SDK: `./mkfirmware.sh`
3. Flash the image of rockdev/rootfs.img

7.11.2 Configuration Filesystem For SPI NAND And SLC NAND

1. Select and add the appropriate code in dts bootargs：

Filesystem type	The bootargs configuration of dts
squashfs	ubi.mtd=3 ubi.block=0,rootfs root=/dev/ubiblock0_0 rootfstype=squashfs
ubifs	ubi.mtd=3 root=ubi0:rootfs rootfstype=ubifs

NOTICE: ubi.mtd=**3**, where the serial number **3** corresponds to the partition in the parameter.txt, counting from mtd0 or mtdblock0.

2. Configurations of Buildroot

Get ${RK_CFG_BUILDROOT} from the command of `./build.sh info`

Modify buildroot/configs/${RK_CFG_BUILDROOT}_defconfig
BR2_PACKAGE_ROOTFS_UBI_CUSTOM_FILESYSTEM=”squashfs“ or "ubifs"

If squashfs, optional configuration of the squashfs file system is as follows:

squashfs file system compression type	buildroot/configs/${RK_CFG_BUILDROOT}_defconfig
lz4	BR2_TARGET_ROOTFS_SQUASHFS4_LZ4=y
lzo	BR2_TARGET_ROOTFS_SQUASHFS4_LZO=y
lzma	BR2_TARGET_ROOTFS_SQUASHFS4_LZMA=y
xz	BR2_TARGET_ROOTFS_SQUASHFS4_XZ=y
zstd	BR2_TARGET_ROOTFS_SQUASHFS4_ZSTD=y
gzip (default)	BR2_TARGET_ROOTFS_SQUASHFS4_GZIP=y

3. `./build.sh rootfs` and `./build.sh kernel`
4. Flash the images of rockdev/rootfs.img and rockdev/boot.img