Compile Linux4.19 Firmware_RK3566

Content

Chapter I Set up the build environment

1.1 The build environment of Linux SDK

Note:

(1) It is recommended to develop in the Ubuntu 18.04 system environment. If other system versions are used, the build environment may need to be adjusted accordingly.

(2) Compile with normal user, do not compile with root user authority.

1.2 Download SDK

First prepare an empty folder to place SDK, better under home, here we use **~/proj** as example.

Attention: To avoid unnecessary errors, please do not place/unzip the SDK in VM shared folders or non-english directories.

Install a few software packages before getting SDK:

```
sudo apt update

sudo apt install -y repo git python
```

Using the command **repo** requires a higher network standard. Choose whether to use it.

Get full SDK options：

Please kindly note: contact sales to get the account and password of DWIN server, then submit the key.

```
mkdir ~/proj/rk356x_linux_release_v1.3.0b_20221213/

cd ~/proj/rk356x_linux_release_v1.3.0b_20221213/



## full SDK
```

```
    repo init -u ssh://dwin@192.168.10.107:/work/gitwork/platform/rk3566_linux/manifest.git --repo-url
ssh://dwin@192.168.10.107:/work/gitwork/platform/rk3566_linux/repo.git
```

1.3 Sync Code

Execute the following command to synchronize the code:

```
    # enter the SDK root directory
    cd ~/proj/rk356x_linux_release_v1.3.0b_20221213/


    # sync
    .repo/repo/repo sync -c
    .repo/repo/repo sync -c --no-tags
    .repo/repo/repo start firefly --all
```

You can use the following command to update the SDK later:

```
    .repo/repo/repo sync -c --no-tags
```

Probably due to the unstable network environment，**.repo/repo/repo sync -c --no-tags** may fail to update. You can execute it repeatedly.

1.4 Directory

```
    .
    ├── app
    ├── buildroot                                    # Buildroot root filesystem build directory
    ├── build.sh -> device/rockchip/common/build.sh                    # compile script
    ├── debian                                       # Debian root filesystem compilation directory
    ├── device                                       # Compile related configuration files
    ├── docs                                                                    # Documentation
    ├── envsetup.sh -> buildroot/build/envsetup.sh
```

```
├── external
├── kernel                                                                  # Kernel
├── Makefile -> buildroot/build/Makefile
├── mkfirmware.sh -> device/rockchip/common/mkfirmware.sh        # Link script
├── prebuilts                                                 # Cross compilation toolchain
├── rkbin
├── rkflash.sh -> device/rockchip/common/rkflash.sh                          # Flash script
├── tools                                                                   # Tools directory
└── u-boot                                    #U-Boot
```

1.5 Install Dependencies

Install directly on PC:

```
sudo apt-get install repo 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 \
unzip device-tree-compiler python-pip ncurses-dev python-pyelftools
```

Chapter II Compile Debian Firmware

This chapter introduces the compilation process of Debian firmware. It is recommended to develop under Ubuntu 18.04 system environment. If you use other system versions, you may need to adjust the compilation environment accordingly.

2.1 Compile SDK

2.1.1 Configuration before compilation

In the **device/rockchip/rk356x/** directory, there are configuration files of different board types.

Return to SDK root directory and execute build.sh to select the configuration file:

```
./build.sh BoardConfig-rk3566-dwin.mk
```

The configuration file will be linked to **device/rockchip/.BoardConfig.mk**，check the file to verify whether the configuration is successful.

2.1.2 Debian root filesystem

Change to the root filesystem directory:

```
cd debian

./build.sh
```

Create a link and link the filesystem to **linaro-rootfs.img：**

```
cd ..


ln -rsf debian/linaro-rootfs.img  rockdev/rootfs.img
```

2.1.3 Automatic compilation

Fully automatic compilation will perform all compilation and packaging operations to generate complete RK firmware.

```
./build.sh
```

2.1.4 Partial compilation

- Compile u-boot

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

- Compile kernel

```
./build.sh kernel
```

- Compile recovery

```
./build.sh recovery
```

2.1.5 Update link

Update each part of the mirror link to **rockdev/** directory:

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

2.1.6 Package the firmware

Pack the firmware, the generated complete firmware will be saved to the **rockdev/pack/** directory.

RK firmware is the firmware packaged in Rockchip's proprietary format, and can be flashed to eMMC or SD card with the tools provided by Rockchip.

```
./build.sh updateimg
```

Chapter III Compile Buildroot firmware

This chapter introduces the compilation process of Buildroot firmware. It is recommended to develop in the Ubuntu 18.04 system environment. If you use other system versions, you may need to adjust the compilation environment accordingly.

3.1 Compile SDK

In the **device/rockchip/rk356x/** directory, there are configuration files of different board types.

Return to SDK root directory to select the configuration file:

```
./build.sh BoardConfig-rk3566-dwin.mk
```

The configuration file will be linked to **device/rockchip/.BoardConfig.mk**，check the file to verify whether the configuration is successful.

3.1.1 Partial compilation

- Compile u-boot

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

- Compile Kernel

```
./build.sh kernel
```

- Compile recovery

```
./build.sh recovery
```

- Compile Buildroot root filesystem

Compiling the Buildroot root filesystem will generate a compilation output directory in **buildroot/output** :

```
./build.sh rootfs

 # Note: Make sure to compile the Buildroot root filesystem as a normal user to avoid unnecessary errors.
```

3.1.2 Package the firmware

Update each part of the mirror link to the **rockdev**/ diectory：

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

Pack the firmware, the generated complete firmware will be saved to the rockdev/pack/ directory.

```
./build.sh updateimg
```

Chapter IV Upgrade the firmware

4.1 Upgrade the firmware via SD card

To upgrade the firmware using an SD card, you need to use a tool on a computer to write the unified firmware onto the SD card. Currently, this operation is only supported on the Windows operating system.

Operation steps:

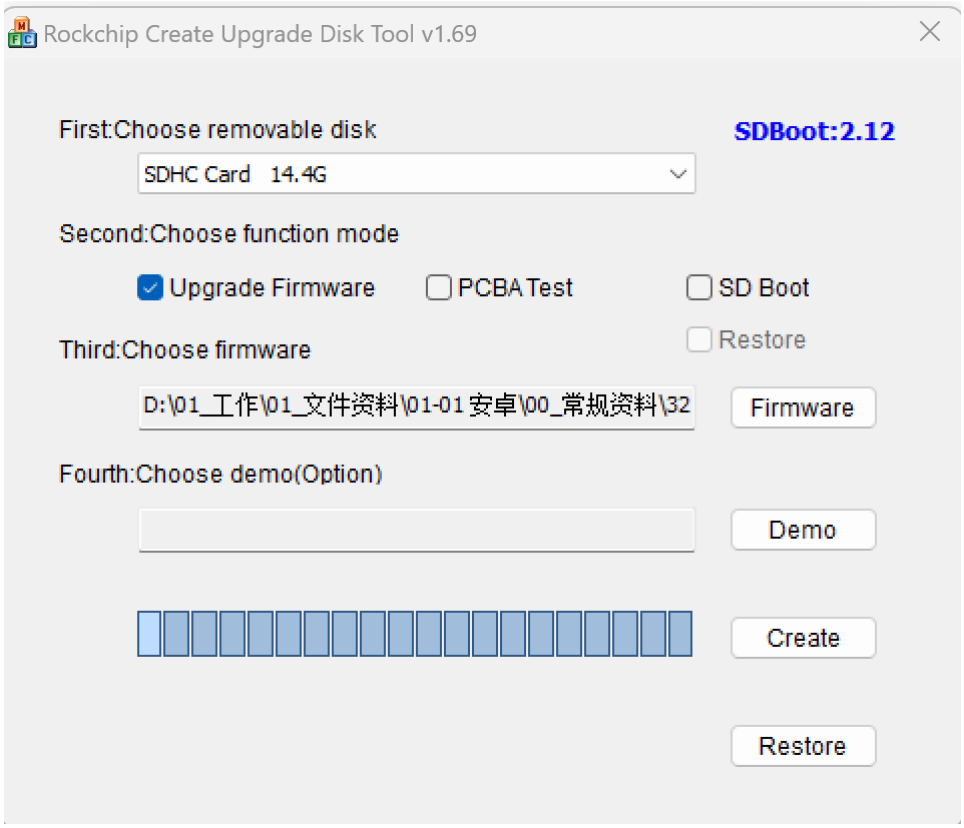
Run SDDiskTool_v1.69，check the “Upgrade Firmware” box and select the correct removable disk device.

Insert SD card into USB card reader and then into USB port of host computer.

Click button “Create” to make it and wait until it is finished.

Remove the SD card, insert it into the SD card slot of the motherboard, power on the board, it will start upgrading automatically.

After the upgrade, remove the SD card and restart the motherboard automatically to complete the whole process of firmware update.



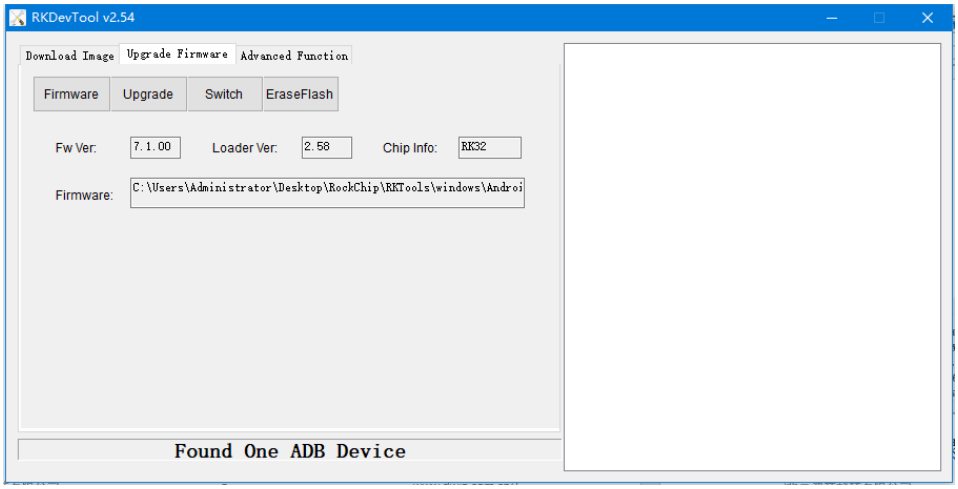
4.2 Upgrade the firmware via Micro USB

If the computer is being used for the first time to perform the burn-in process, you need to install the driver. Please refer to the "USB Driver Installation Instructions" in the USB driver directory.

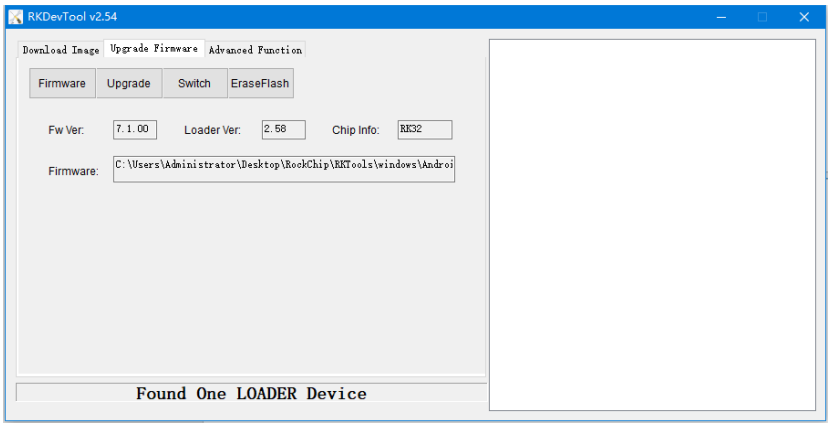
Operation steps:

Download the firmware you need to upgrade to the screen.

Open RKDevTool, select "Upgrade Firmware," and click on "Firmware" to choose the **.img** file to be burned.



While the device is powered off, press and hold the Recover button on the Android screen. First, connect the PC using a USB cable, and then connect the power supply (DC-12V). The following interface will appear. Click "Upgrade" to start the burn-in process. Once the burn-in is complete, the device will automatically restart.



Chapter V Revision Records

Rev	Revise Date	Content	Editor
00	July 24, 2024	First Edition	Zhong Liu