

RV1126-RV1109 使用指导文档

目录

RV1126-RV1109 使用指导文档

目录

1 发布说明

1.1 版本说明

1.2 免责声明

1.3 版权声明

2 主板基本信息

2.1 pro-rv1126 基本信息

2.2 pro-rv1109 基本信息

2.3 rp-rv1126 基本信息

2.4 rp-rv1109 基本信息

3 编译环境配置

4 源码获取

4.1 百度网盘下载源码

4.2 源码解压

5 编译

5.1 buildroot (linux) 系统编译

5.1.1 版型选择

5.1.2 LCD配置 (可选)

5.1.3 摄像头配置 (可选)

5.1.4 编译所有镜像文件

5.1.5 单独编译 uboot

5.1.6 单独编译kernel

5.1.7 单独编译buildroot

5.1.8 单独编译 recovery

5.1.9 单独编译 buildroot 应用

5.2 ubuntu 系统编译

5.2.1 下载 SDK 和 ubuntu 文件系统

5.2.2 版型选择

5.2.3 LCD配置 (可选)

5.2.4 摄像头配置 (可选)

5.2.5 编译 ubuntu 镜像文件

常见编译问题

6 USB 烧写驱动安装

7 镜像升级工具安装

8 镜像升级

8.1 固件升级

8.2 擦除 flash 操作

8.3 单独升级指定分区镜像

9 外围设备

摄像头

GPIO 配置与控制

4G拨号

声卡

OTG/HOST模式切换

wifi 连接

wifi 热点

蓝牙

- LED 控制
- 10 调试开发
 - 内核配置
 - Buildroot 配置
 - 主设备树目录
 - 交叉编译工具
 - buildroot常用目录
 - rknpu
 - 文件上传和下载
 - 默认用户名和密码
 - 取消开机自启动应用
- 11 功能测试

1 发布说明

1.1 版本说明

日期	修订版本	修订内容	修改人	核定人	当前版本
2020-12-03	V1.0	初始版本	lixin	lixin	
2020-12-10	V1.1	原本的一个版型分为两个版型	lixin	lixin	
2021-02-03	V2.0	新增 ubuntu18.04 系统	lixin	lixin	
2021-03-25	V2.1	1.更新SDK版本为原厂v1.8.0 2.优化和修正指导文档内容 3.删除 ubuntu 系统及文档相关描述	lixin	lixn	

日期	修订版本	修订内容	修改人	核定人	当前版本
2021-06-07	V2.2	1.新增UVC Camera SDK编译说明 2.优化文档内容 3.适配了IPC版型	lixin	lixin	
2021-07-12	V2.3	1.新增ubuntu20.04系统编译说明 2.优化文档内容 3.适配了快速启动	lixin	lixin	
2021-08-17	V2.4	优化文档内容	lixin	lixin	
2022-03-23	V3.0	1.更新SDK版本为原厂v2.2 2.ubuntu20.04系统支持isp及双目摄像头 3.优化文档内容 4.合并rp和pro的指导文档，修改文档名称	lixin	lixin	√

1.2 免责声明

本文档由深圳荣品电子科技有限公司（本司，下同）提供，仅作参考、指导用。不对文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。

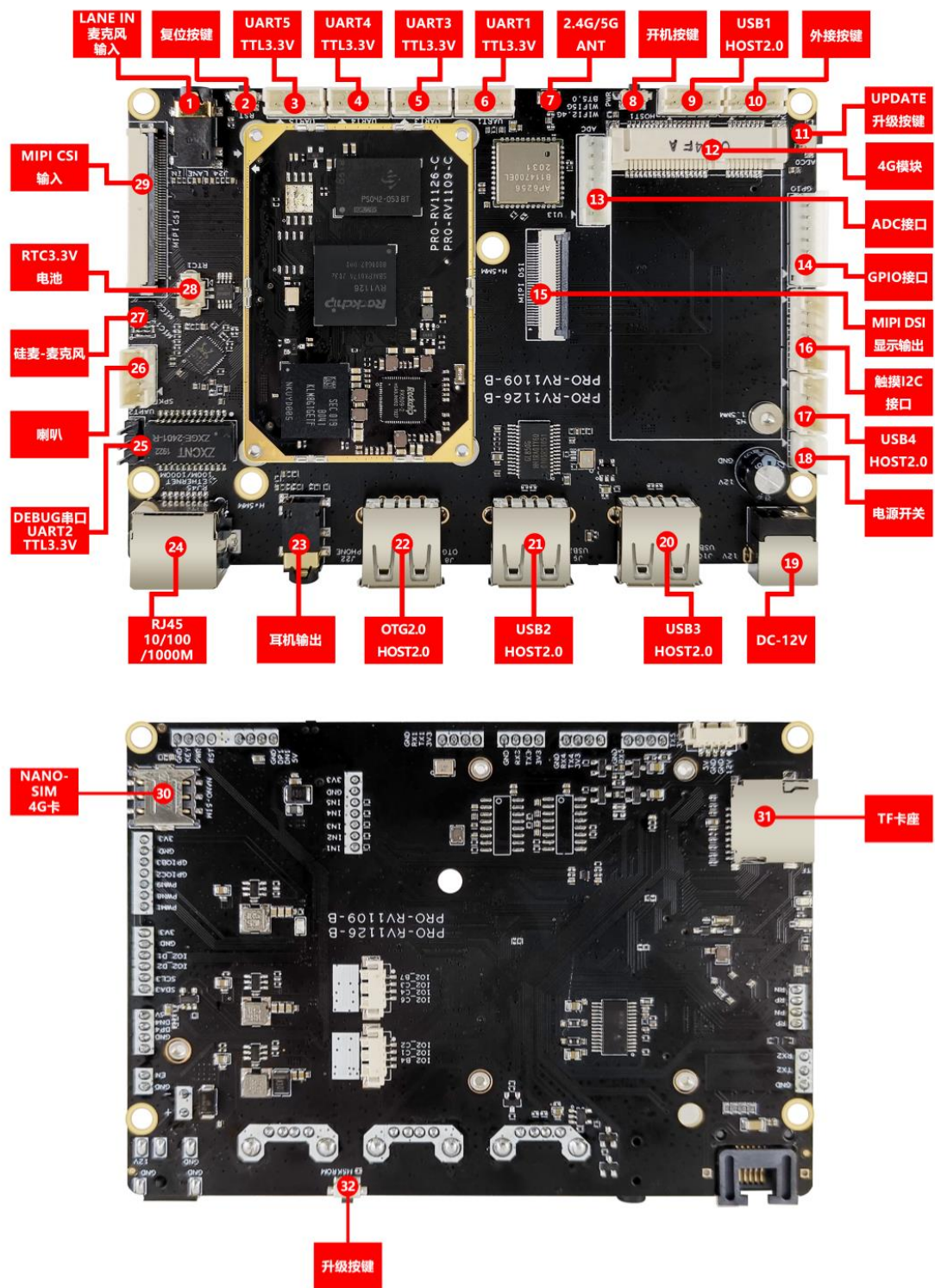
本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

1.3 版权声明

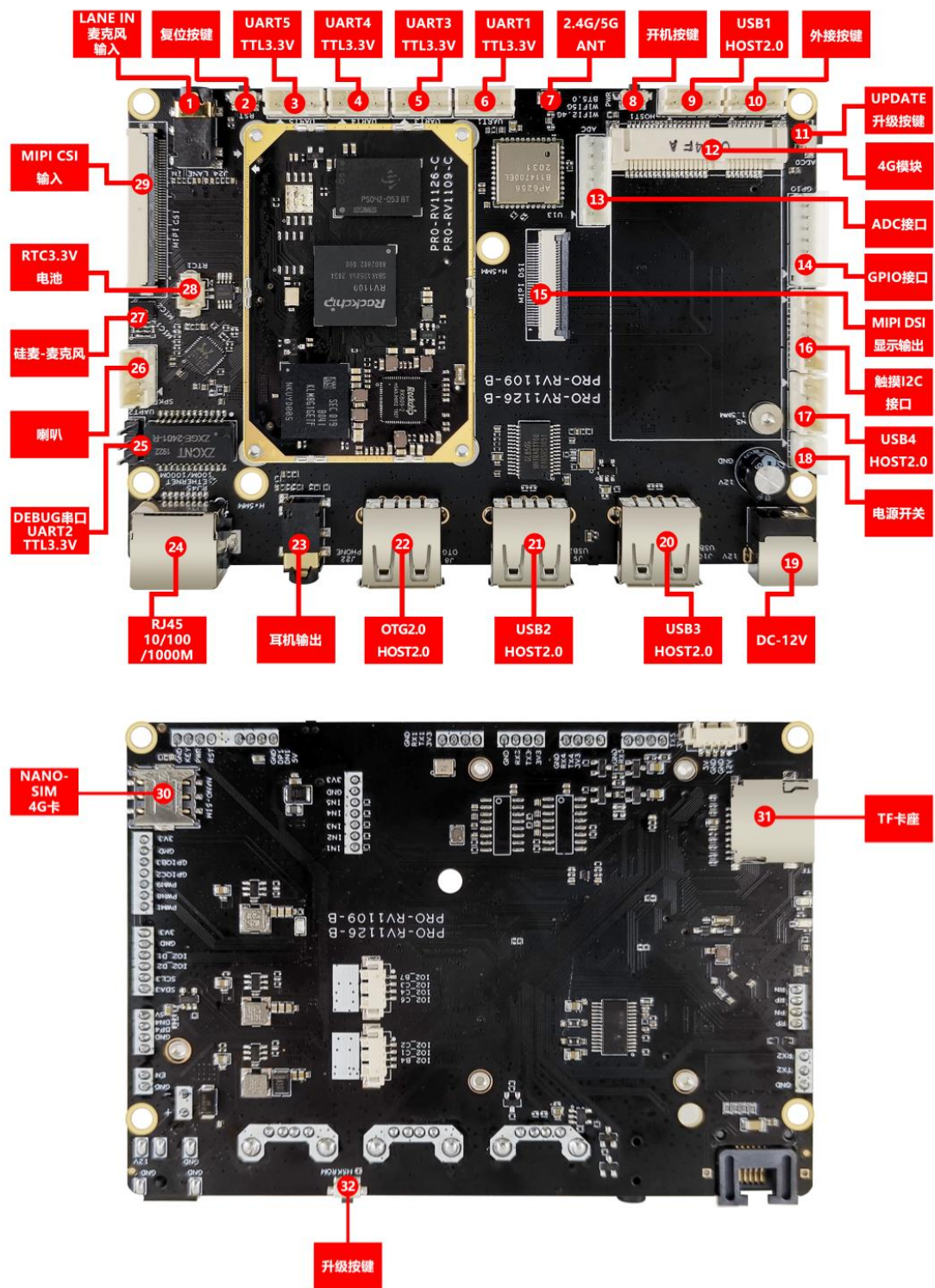
本文档版权归属于深圳荣品电子科技有限公司。非经本司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

2 主板基本信息

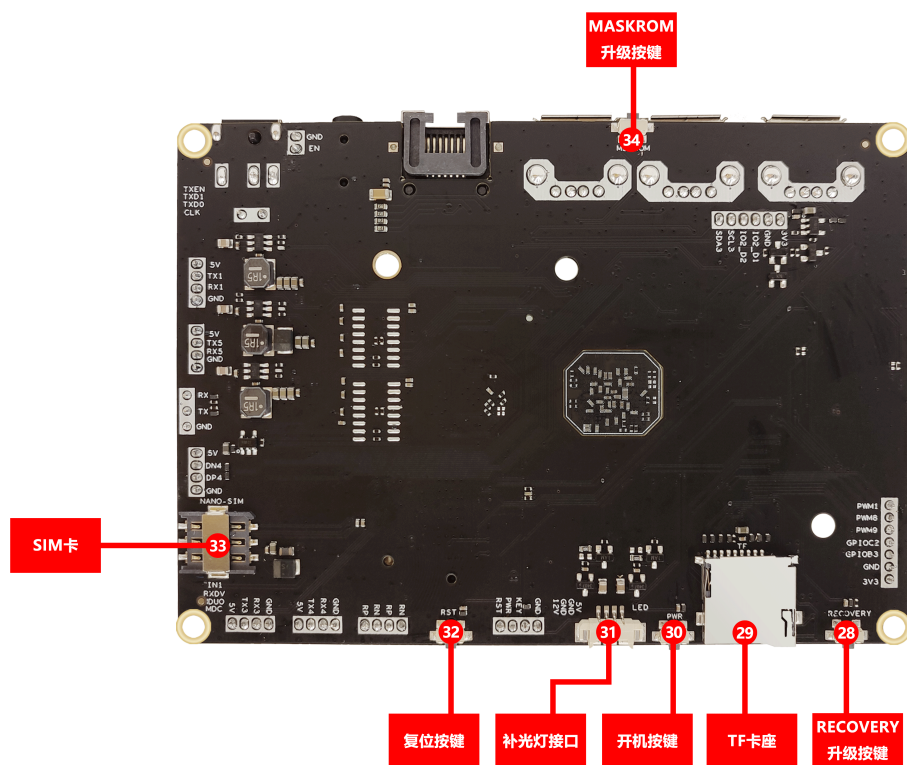
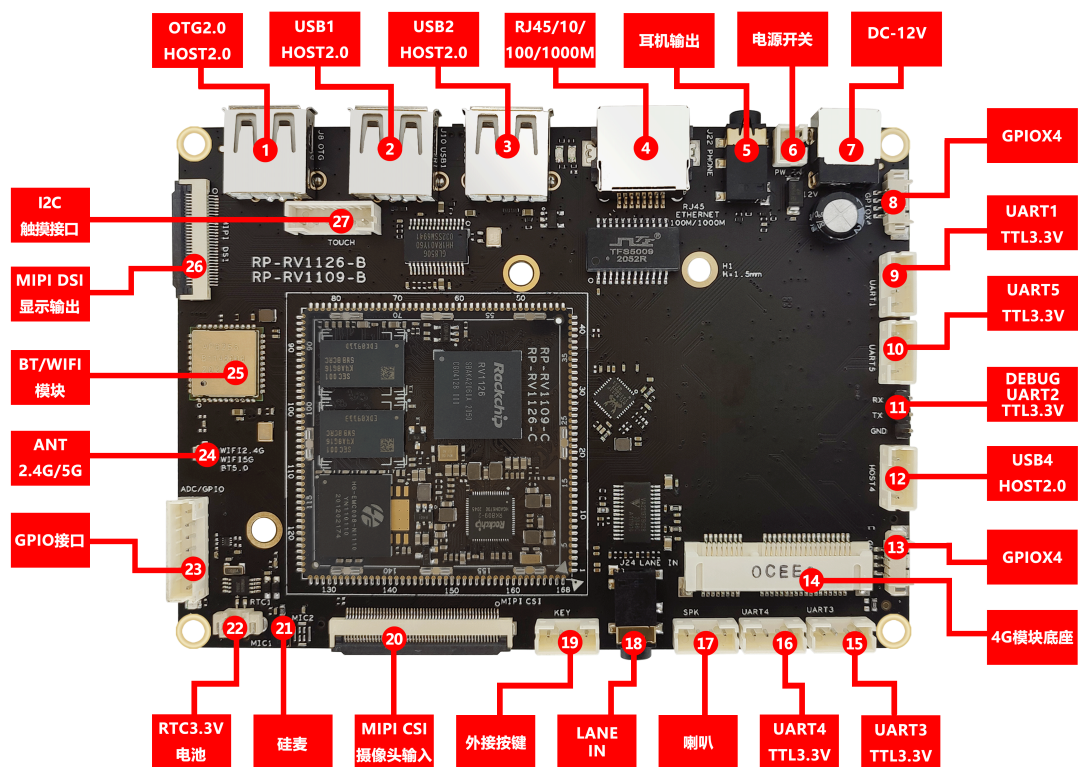
2.1 pro-rv1126 基本信息



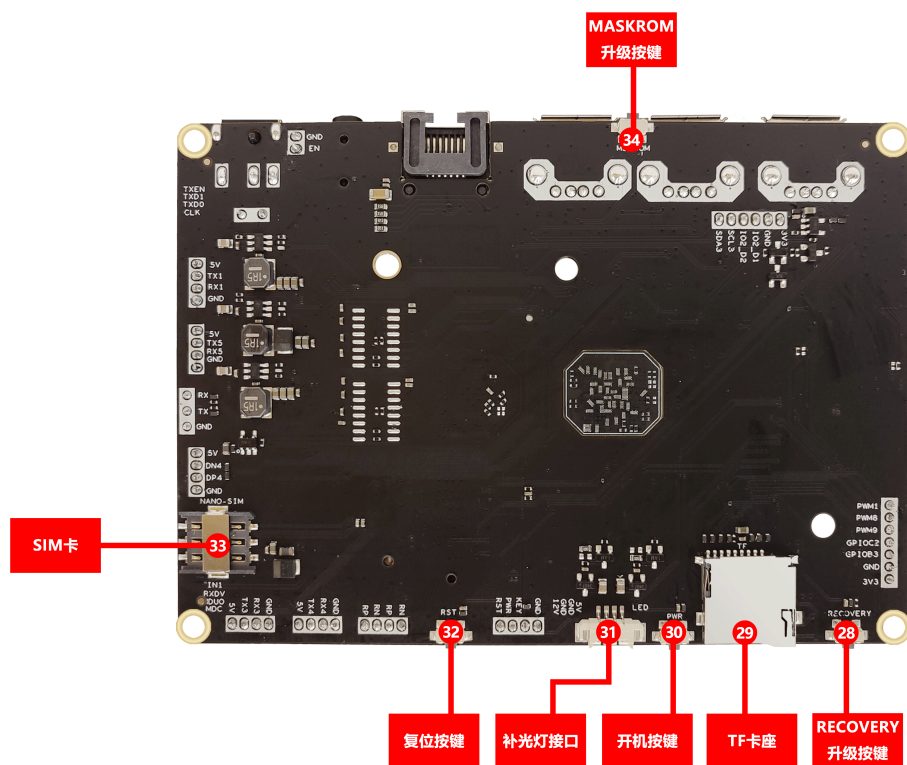
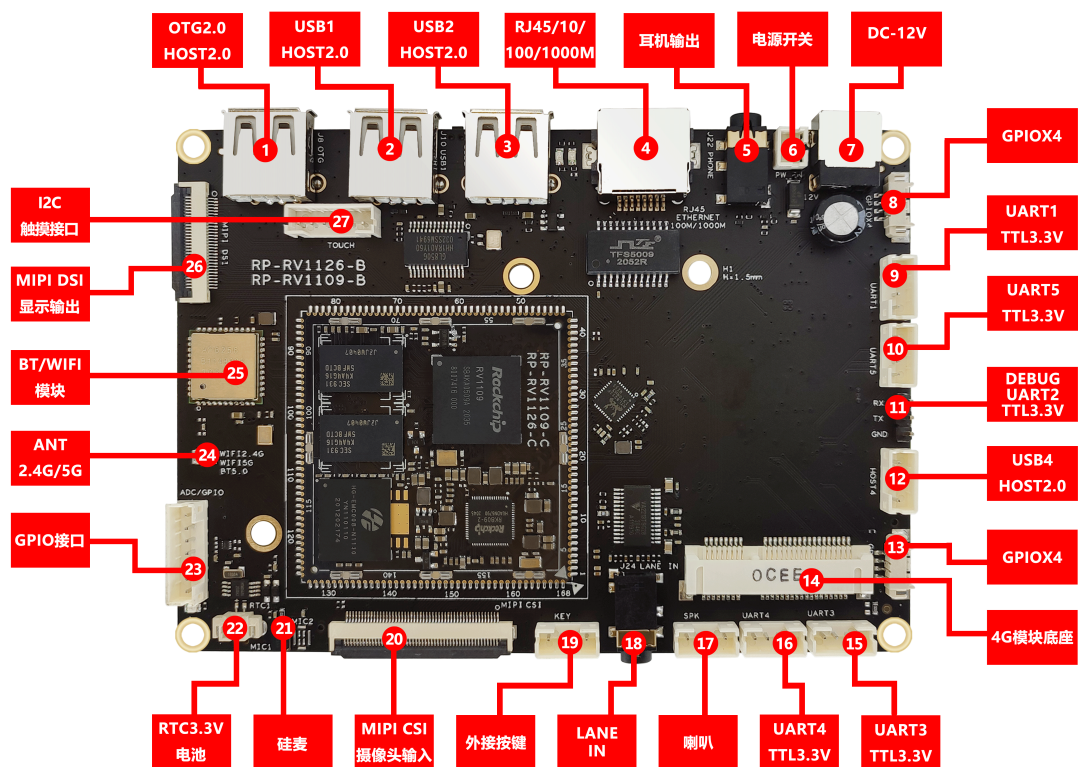
2.2 pro-rv1109 基本信息



2.3 rp-rv1126 基本信息



2.4 rp-rv1109 基本信息



3 编译环境配置

特别说明：不能放在虚拟机与 Windows 的共享文件夹中编译！！

ubuntu 16.04 需要安装的包如下

```
$ sudo apt-get install repo git-core gitk git-gui gcc-arm-linux-gnueabi \
u-boot-tools device-tree-compiler gcc-aarch64-linux-gnu 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 libgl2.0-dev libgtk2.0-dev \
libglade2-dev cvs git mercurial rsync openssh-client subversion \
asciidoc w3m dblatex graphviz python-matplotlib libc6:i386 \
libssl-dev expect fakeroot cmake bison
```

ubuntu 17.04 , 还需要安装

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

4 源码获取

源码发布在网盘，百度网盘发布的包包括源码、文档、镜像、工具等，网盘上的源码包，

特别说明： pro-rv1126 和 pro-rv1109 共用一套源码包。

4.1 百度网盘下载源码

网盘源码路径： pro-rv1126_rv1109/01SDK 源码/
源码包名： RV1126_1109_XXX.tgz
XXX 为 SDK 的发布日期

名称	修改日期	类型	大小
patch	2021/1/5 13:50	文件夹	
md5sum.txt	2021/2/22 20:42	文本文档	1 KB
RV1126_1109_20210203.tgz	2021/2/22 20:46	TGZ 文件	7,301,541 KB

4.2 源码解压

```
$ tar xzvf RV1126_1109-XXX.tgz
```

查看是否生成 .git

```
$ ls -al
drwxrwxr-x 8 rpdzkj rpdzkj 4096 May 9 14:33 .git
```

恢复完整的源码

```
$ git reset --hard
```

如果 git reset --hard 指令执行结束后看不到完整源码，请比较压缩包的 md5 校验码是否与我司提供的一致，不一致请重新下载源码包。

```

rv1126_1109
├─ app                                # 软件应用源码
├─ br.log                             # 编译日志
├─ buildroot                          # buildroot源码
├─ build.sh -> device/rockchip/common/build.sh # 编译脚本
├─ device                             # 存放版型配置
├─ docs                              # sdk相关文档
├─ envsetup.sh -> buildroot/build/envsetup.sh # 配置环境变量的脚本
├─ external                           # 存放rkmedia、rknn等
外设相关源码
├─ IMAGE                             # 存放所有编译过的镜像
├─ kernel                            # 内核源码
├─ Makefile -> buildroot/build/Makefile      # 顶层makefile
├─ mkfirmware.sh -> device/rockchip/common/mkfirmware.sh # 打包脚本
├─ prebuilts                         # 交叉编译工具
├─ rkbin                             # rk封装的一些bin文件
├─ rkflash.sh -> device/rockchip/common/rkflash.sh # 镜像制作脚本
├─ rockdev                           # 存放烧写镜像
├─ tools                             # 存放工具
└─ u-boot                            # uboot 源码

```

5 编译

以下操作，若无特殊说明，均在SDK根目录下进行。

5.1 buildroot (linux) 系统编译

5.1章节为编译 buildroot 系统，若需编译 ubuntu 系统，请直接前往[\[5.2\]](#)章节。如果SDK先编译了ubuntu系统，为了避免出现不必要的错误，请先清理内核再做操作。

```
$ cd kernel && make clean -j 20
```

5.1.1 版型选择

根据选项和当前主板的型号进行选择，主板型号可查看 CPU 确认，输入对应的序号选择，这里以 pro_rv1109 版型为例。

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

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

0. default BoardConfig.mk
1. pro-rv1109.mk
2. pro-rv1126.mk
3. rp-rv1109.mk
4. rp-rv1126.mk
which would you like? [0]: 1 # 比如选择 1 编译 pro-rv1109
```

说明：不同版本的 SDK 参数可能不一样，选择当前开发板对应的即可，其余的版型我司尚未匹配，内核管脚以及电源域的配置可能与我司开发板不符合，用户擅自编译烧写进去可能会导致 IO 损坏。

配置 buildroot 默认环境，pro_rv126 和 pro_rv1109 对应的版型为 rockchip_rv1126_rv1109_facial_gate

```
$ source envsetup.sh
Top of tree: /home/rpdzkj_debug/second/lixin/RV1126-
RV1109/rv1126_rv1109_201116/rv1126_1109
You're building on Linux
Lunch menu...pick a combo:
0. non-rockchip boards
1. rockchip_px30_32
2. rockchip_px30_64
.
.
.
78. rockchip_rv1126_rv1109
79. rockchip_rv1126_rv1109_ab
80. rockchip_rv1126_rv1109_facial_gate
81. rockchip_rv1126_rv1109_libs
82. rockchip_rv1126_rv1109_ramboot_uvcc
.
.
.
which would you like? [0]:80 # 选择 80
```

说明：不同版本的 SDK 参数可能不一样，选择 rockchip_rv1126_rv1109_facial_gate 对应的序号即可。

或者执行如下指令直接选择版型。

```
$ source envsetup.sh rockchip_rv1126_rv1109_facial_gate
```


5.1.2 LCD配置（可选）

若需点亮 LCD 屏，请针对 LCD 的尺寸、分辨率修改相应的配置。

配置文件： `kernel/arch/arm/boot/dts/{主dts}`，主dts文件选择[\[参考\]](#)

示例：配置 MIPI 5 寸 720X1280 的 LCD（只能选其一）

```
// -----lcd select -----
// #include "rongpin/rp-lcd-mipi2hdm-1t8912.dtsi"
// #include "rongpin/rp-lcd-mipi-10-1920-1200.dtsi"
// #include "rongpin/rp-lcd-mipi-10-800-1280-v2.dtsi"
// #include "rongpin/rp-lcd-mipi-10-800-1280.dtsi"
#include "rongpin/rp-lcd-mipi-5-720-1280-v2.dtsi"
// #include "rongpin/rp-lcd-mipi-5-720-1280.dtsi"
// #include "rongpin/rp-lcd-mipi-5.5-720-1280-v2.dtsi"
// #include "rongpin/rp-lcd-mipi-5.5-720-1280.dtsi"
// #include "rongpin/rp-lcd-mipi-7-1024-600.dtsi"
// #include "rongpin/rp-lcd-mipi-7-800-1280.dtsi"
// #include "rongpin/rp-lcd-mipi-8-800-1280.dtsi"
```

5.1.3 摄像头配置（可选）

配置文件： `kernel/arch/arm/boot/dts/{主dts}`，主dts文件选择[\[参考\]](#)

示例：配置为双目 GC2093 摄像头（只能选其一）

```
// -----camera select-----
// #include "rongpin/rp-camera-gc2053+ov2718.dtsi"
// #include "rongpin/rp-camera-imx307x1.dtsi"
#include "rongpin/rp-camera-gc2093x2.dtsi"
```

5.1.4 编译所有镜像文件

```
$ ./build.sh # 编译 uboot、kernel、rootfs、recovery
```

编译成功后将会在 `rockdev/` 目录下生成如下镜像文件

```
$ ls rockdev/
MiniLoaderAll.bin  #引导 uboot、检测启动方式
parameter.txt     #系统分区表、内核传参
uboot.img         #引导系统分区
trust.img         #ARM TrustZone 特性分区
misc.img          #记录 recovery 升级信息
boot.img          #内核、设备树、logo 分区
recovery.img      #恢复还原分区
rootfs.img        #根文件系统分区
oem.img           #存放 OEM 资源分区
update.img        #完整固件
```

5.1.5 单独编译 uboot

```
$ ./build.sh uboot && ./build.sh firmware
```

生成镜像路径: rockdev/

生成镜像: uboot.img

5.1.6 单独编译kernel

```
$ ./build.sh kernel && ./build.sh firmware
```

生成镜像路径: rockdev/

生成镜像: boot.img

5.1.7 单独编译buildroot

编译完 buildroot 完成后会自动编译 rootfs

```
$ ./build.sh buildroot && ./build.sh firmware
```

生成镜像路径: rockdev/

生成镜像: rootfs.img

5.1.8 单独编译 recovery

```
$ ./build.sh recovery && ./build.sh firmware
```

生成镜像路径: rockdev/

生成镜像: recovery.img

5.1.9 单独编译 buildroot 应用

sdk 在已经编译过文件系统的前提下 (执行过 ./build.sh 或者 ./build.sh buildroot) , 可以单独编译 buildroot 的单个应用甚至单个模块, 方便用于调试, 比如单独编译 QFacialGate 应用。

```
$ make QFacialGate-dirclean && make QFacialGate-rebuild -j16
```

单独编译 rkmedia

```
$ make rkmedia-dirclean && make rkmedia-rebuild -j16
```

编译成功后应用会被拷贝到 `buildroot/output/rockchip_rv1126_rv1109_facial_gate/target/` 目录下的文件系统中。

5.2 ubuntu 系统编译

5.2 章节为编译 ubuntu 系统，若需编译 buildroot 系统，请直接前往[\[5.1\]](#)章节，如果SDK先编译了 buildroot系统，为避免出现不必要的错误，请先清理内核再做操作。

```
$ cd kernel && make clean -j 20
```

5.2.1 下载 SDK 和 ubuntu 文件系统

编译之前请先下载我司提供的 ubuntu 文件系统，文件系统路径在网盘链接的 `pro-rv1126_rv1109/01SDK源码/ubuntu20.04/` 目录下，下载 SDK 源码和 ubuntu 文件系统，然后将 ubuntu20.04 文件系统的压缩包解压到 SDK 的 ubuntu 目录下，如果目录不存在，则可自行创建一个：

```
$ ls ubuntu/ -l
total 1638404
-rw-r--r-- 1 rpdzkj_debug rpdzkj_debug 1677721600 2月  3 09:32 rootfs.ext4
```

5.2.2 版型选择

根据选项和当前主板的型号进行选择，主板型号可查看 CPU 确认，输入对应的序号选择，这里以 `pro_rv1109` 的 ubuntu 系统为例。

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

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

0. default BoardConfig.mk
1. pro-rv1109-ubuntu.mk
2. pro-rv1109.mk
3. pro-rv1126-ubuntu.mk
4. pro-rv1126.mk
which would you like? [0]: 1  # 选择 1 编译 ubuntu 系统
```

说明：不同版本的 SDK 参数可能不一样，选择当前开发板对应的即可，其余的版型我司尚未匹配，内核管脚以及电源域的配置可能与我司开发板不符合，用户擅自编译烧写进去可能会导致 IO 损坏。

5.2.3 LCD配置（可选）

若需点亮 LCD 屏，请针对 LCD 的尺寸、分辨率修改相应的配置。

配置文件： `kernel/arch/arm/boot/dts/{主dts}`，主dts文件选择[\[参考\]](#)

示例：配置 MIPI 5 寸 720X1280 的 LCD（只能选其一）

```
// -----lcd select -----
//#include "rongpin/rp-lcd-mipi2hdm-1t8912.dtsi"
//#include "rongpin/rp-lcd-mipi-10-1920-1200.dtsi"
//#include "rongpin/rp-lcd-mipi-10-800-1280-v2.dtsi"
//#include "rongpin/rp-lcd-mipi-10-800-1280.dtsi"
#include "rongpin/rp-lcd-mipi-5-720-1280-v2.dtsi"
//#include "rongpin/rp-lcd-mipi-5-720-1280.dtsi"
//#include "rongpin/rp-lcd-mipi-5.5-720-1280-v2.dtsi"
//#include "rongpin/rp-lcd-mipi-5.5-720-1280.dtsi"
//#include "rongpin/rp-lcd-mipi-7-1024-600.dtsi"
//#include "rongpin/rp-lcd-mipi-7-800-1280.dtsi"
//#include "rongpin/rp-lcd-mipi-8-800-1280.dtsi"
```

5.2.4 摄像头配置（可选）

配置文件： `kernel/arch/arm/boot/dts/{主dts}`，主dts文件选择[\[参考\]](#)

示例：配置为 GC2093 摄像头（只能选其一）。

```
// -----camera select-----
//#include "rongpin/rp-camera-gc2053+ov2718.dtsi"
//#include "rongpin/rp-camera-imx307x1.dtsi"
#include "rongpin/rp-camera-gc2093x2.dtsi"
```

5.2.5 编译 ubuntu 镜像文件

```
$ ./build.sh
```

编译成功后将会在 rockdev/ 目录下生成如下镜像文件

```
$ ls rockdev/
MiniLoaderAll.bin  #引导 uboot、检测启动方式
parameter.txt      #系统分区表、内核传参
uboot.img          #引导系统分区
boot.img           #内核、设备树、logo 分区
recovery.img       #recovery分区
rootfs.img         #根文件系统分区
update.img         #完整固件
```

常见编译问题

1、编译 QFacialGate-1.0 失败的解决办法：执行指令：

```
make libIPCProtocol-rebuild;make QFacialGate-dirclean;make QFacialGate-rebuild;./build.sh buildroot
```

2、在较低版本的操作系统上可能会出现 ptherad:No such file or directory 找到编译报错的包生成的 makefile，然后去掉其编译参数 pthread

3、如果执行 build.sh 运行完成后没有在 `rockdev/` 目录下生成镜像文件，请手动执行

```
$ ./build.sh firmware;./build.sh updateimg
```

4、build.sh 执行完后提示：ln: failed to create symbolic link 'rockdev/rootfs.ext4': No such file or directory 执行一下

```
$ ./build.sh firmware;./build.sh
```

6 USB 烧写驱动安装

如果需要烧录固件，则该驱动必须安装。

USB 驱动路径：[网盘链接/06工具/烧写驱动](#)

USB 驱动文件：DriverAssitant_v4.91.zip（或更高版本）

安装方式：解压到本地目录，点击 DriverInstall.exe 安装即可

7 镜像升级工具安装

升级工具路径：[网盘链接/06工具/烧写工具](#)

升级工具：RKDevTool_Release_v2.74.zip（或更高版本）

安装方式：解压到本地目录，打开 RKDevTool.exe 即可

工具界面：

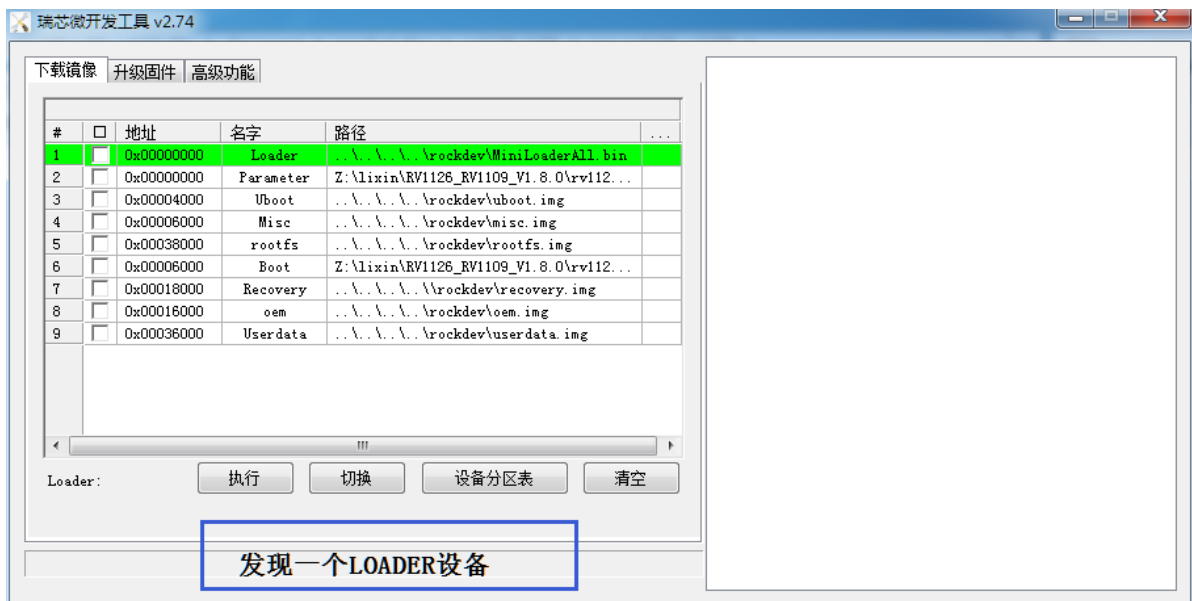


8 镜像升级

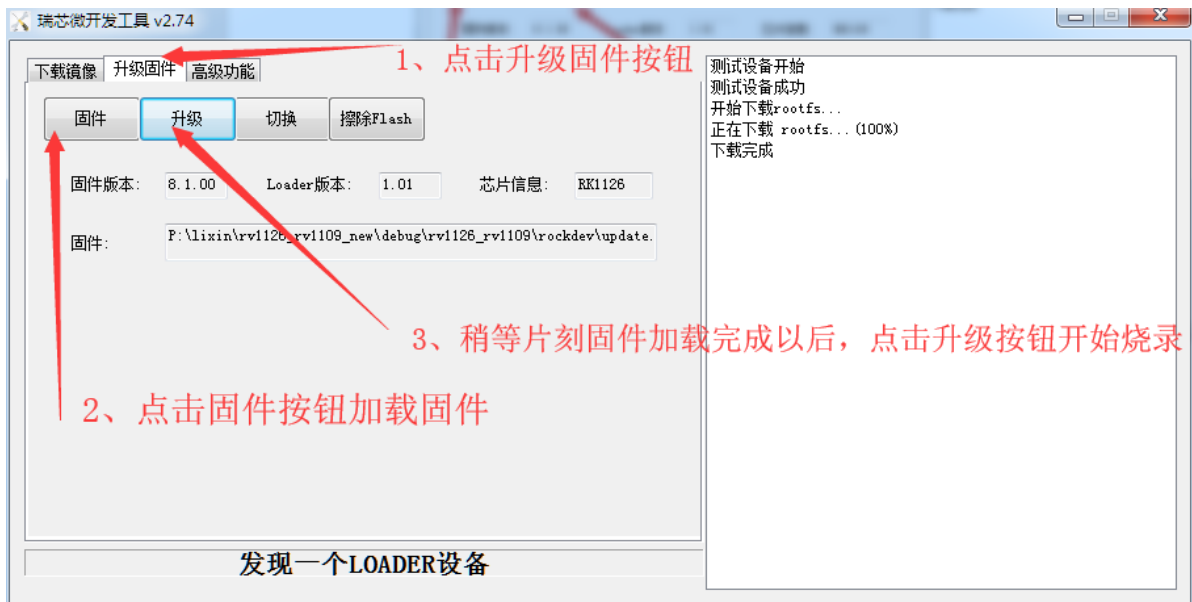
8.1 固件升级

在进行升级之前确保已经安装USB烧写驱动[\[参考\]](#)，固件升级是指烧写所有的分区，开发板上的所有分区都会被更新，加载的固件文件是将所有分区镜像都打包到一起的 update.img 文件，并非单个分区文件。操作流程如下：

1. 主板上电（12V）
2. 主板 USB（OTG 接口）与电脑连接
3. 按住主板 update 按键，按一下主板 reset 按键
4. 升级工具识别到 LOADER 或者 MASKROOM设备（注意不是ADB设备）后方可松开升级按键



5. 选择固件、开始升级固件

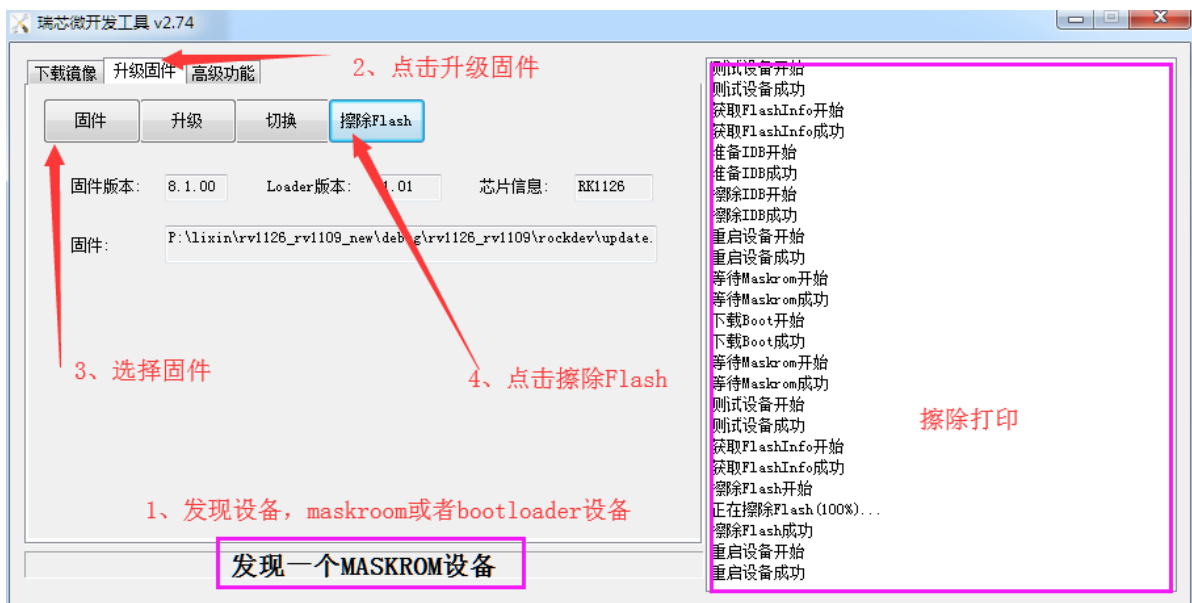


固件获取途径:

- 1、网盘提供的固件 `pro-rv1126_rv1109/04` 镜像文件/ 目录下
- 2、个人编译生成的固件: `<SDK 根目录>/rockdev/update.img`

8.2 擦除 flash 操作

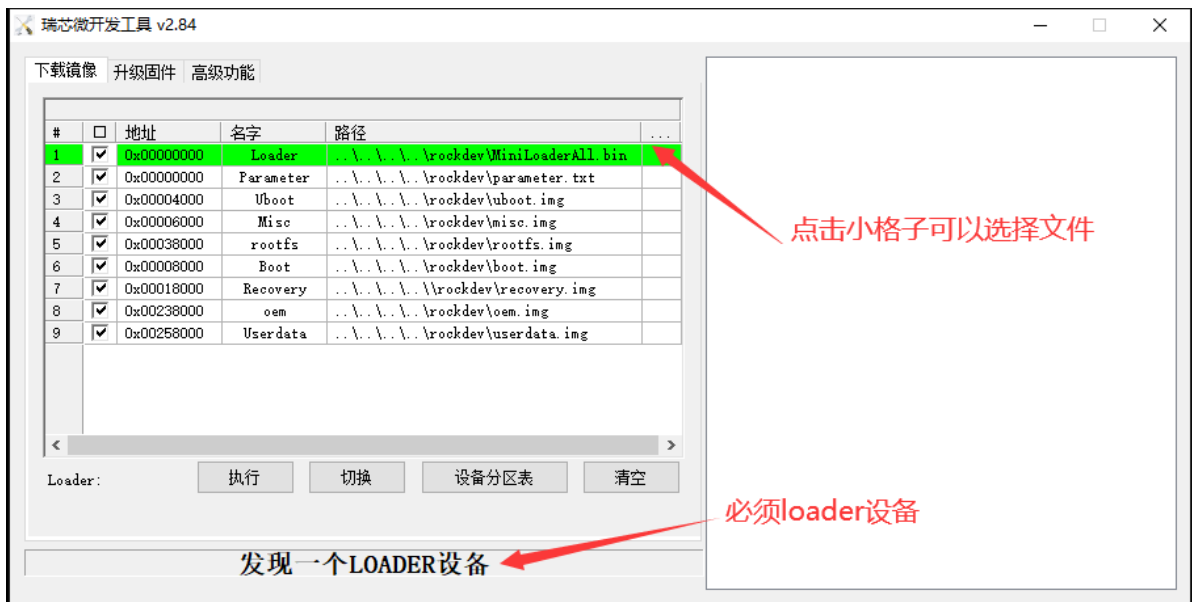
开发板切换到 loader 或 maskroom 模式，选择对应的升级固件，然后点击擦除 Flash



8.3 单独升级指定分区镜像

单独升级指定分区必须是 **LOADER** 设备才能升级，以下任意一种方式均可使设备进入 **LOADER** 模式

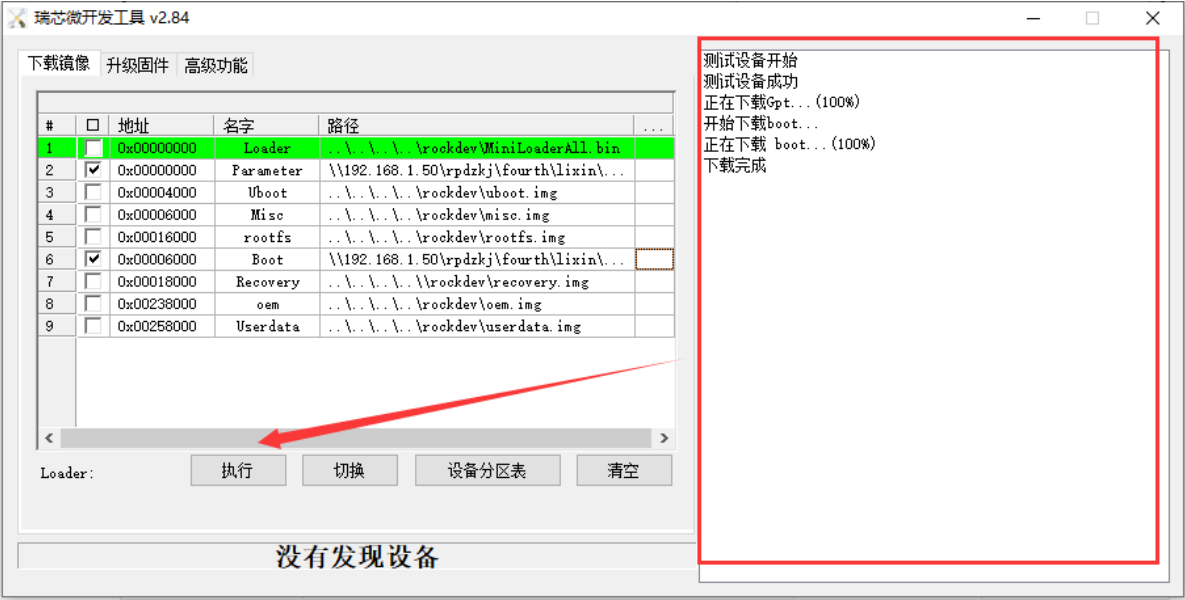
- 按住开发板 recovery 按键上电或者断电重启
- 串口执行 `reboot loader` 指令
- ADB 模式下点击工具的 `切换` 按钮



根据表格选择对应的镜像文件

名称	对应镜像文件	是否必选	说明
Loader	sdk/rockdev/MiniLoaderAll.bin		
Parmeter	sdk/rockdev/ parameter.txt	是	
Uboot	sdk/rockdev/ uboot.img		uboot分区
Misc	sdk/rockdev/ misc.img		
rootfs	sdk/rockdev/ rootfs.img		文件系统
Boot	sdk/rockdev/ boot.img		内核分区
Recovery	sdk/rockdev/ recovery.img		
oem	sdk/rockdev/ oem.img		
Userdata	sdk/rockdev/ userdata.img		

如果 rockdev 目录下没有镜像文件，请先 [参考](#) 编译章节编译镜像文件。勾选需要升级的分区，然后点击工具的 **执行** 按钮即可开始升级



9 外围设备

摄像头

摄像头节点

MIPI sensor0	MIPI sensor1	USB camera
video31 (rkispp_m_bypass)	video38 (rkispp_m_bypass)	video45
video32 (rkispp_scale0)	video39 (rkispp_scale0)	video46
video33 (rkispp_scale1)	video40 (rkispp_scale1)	
video34 (rkispp_scale2)	video41 (rkispp_scale2)	

可以使用 `media-ctl -p -d /dev/media3` 查看 sensor0 所有节点信息，`media-ctl -p -d /dev/media4` 查看 sensor1 所有节点信息。

对于没有屏幕的客户，可以使用如下指令通过rtsp推流预览摄像头画面。

预览 sensor0

```
$ /etc/init.d/S06_QFacialGate stop
$ rkmedia_vi_venc_rtsp_test -a /etc/iqfiles/ -I 0
```

预览 sensor1

```
$ rkmedia_vi_venc_rtsp_test -a /etc/iqfiles/ -I 1
```

使用 vlc 输入 rtsp://xxx.xxx.xxx.xxx/live/main_stream 查看画面。

有屏幕的客户可以使用 rkmedia 的其他 demo 查看和获取摄像头数据流，rkmedia 和人脸识别应用是冲突的，所以使用 rkmedia 之前需要停止掉人脸识别应用。

```
$ /etc/init.d/S06_QFacialGate stop
$ rkmedia_vo_scale_test -a /etc/iqfiles/ # 如运行单目缩放预览
$ rkmedia_vi_double_cameras_test -a /etc/iqfiles/ #双目同时显示预览，注意若运行失败需要指定宽高
```

屏幕预览USB摄像头，USB摄像头默认获取 YUV420 的数据，如果获取数据失败，确认当前摄像头是否支持 YUV 数据格式输出，节点根据实际情况修改，默认为 video45

```
$ /etc/init.d/S06_QFacialGate stop
$ rkmedia_usbcamera_test /dev/video45
```

USB摄像头推流，节点根据实际情况修改，默认为 video45

```
$ rkmedia_usbcamera_rtsp /dev/video45
```

使用 vlc 输入 rtsp://xxx.xxx.xxx.xxx/live/main_stream 查看画面。

GPIO 配置与控制

设备树文件：

kernel/arch/arm/boot/dts/{主dts}，主dts文件选择[\[参考\]](#)

示例：配置 GPIO1_B7 为输出功能

```
rp_gpio{
    status = "okay";
    compatible = "rp_gpio";

    gpio1b7 { // 节点名称与 /proc/rp_gpio/ 目录下的一致
        gpio_num = <&gpio1 RK_PB7 GPIO_ACTIVE_HIGH>;
        gpio_function = <0>;
    };
};
```

开发板上控制和读取

```
$ cat /proc/rp_gpio/gpio1b7 # 读取
$ echo 1 > /proc/rp_gpio/gpio1b7 # 拉高
$ echo 0 > /proc/rp_gpio/gpio1b7 # 拉低
```

4G拨号

后台有个 4G-demon 守护进程会自动拨号，自动检测断网重连，每24小时会消耗大约 304KB 的流量如不需要该进程可以重命名 `/etc/init.d/S52_4G-Daemon.sh` 为非 S 开头。

手动拨号操作参考：

```
$ sh /etc/ppp/peers/quectel-pppd.sh
```

声卡

音频录制，录制默认输出为 `/tmp/ai.pcm`，默认使用pdm数字硅麦，录音直接执行

```
$ rkmedia_ai_test
```

如果使用 rk809 声卡录音执行

```
$ rkmedia_ai_test -d default:CARD=rockchiprk809co
```

音频播放，需要指定声卡才能播放，**因为默认声卡为 pdm 数字硅麦声卡，不支持音频输出功能。**

```
$ rkmedia_ao_test -d sysdefault:CARD=rockchiprk809co -i /tmp/ai.pcm
```

OTG/HOST模式切换

当 OTG USB 口接上 USB 数据线的时候，不能实现 OTG 模式与 HOST 模式切换，主板上电默认为 OTG 模式，切换之前可以先拔掉USB数据线再切换。

```
$ echo 1 > /proc/rp_gpio/otg_host      # 切换至 HOST 模式
$ echo 0 > /proc/rp_gpio/otg_host      # 切换至 OTG 模式
```

wifi 连接

默认可以使用如下几种方式连接 AP

浏览器连接，浏览器输入开发板 ip 后，输入用户名和密码，**默认用户和密码均为 admin**，然后在网络配置里面可以对 wifi 进行打开、关闭、扫描和连接等操作

终端使用connman，netserver 会在后监听网络，wifi 的状态会以数据库中状态为准，如果需要通过终端连接 wifi，需要先停止掉 netserver 应用，再进行连接操作

```
$ killall ipc-daemon netserver
```


然后执行 connmanctl 进入交互模式

```
$ connmanctl
connmanctl> enable wifi      #打开
connmanctl> scan wifi        #扫描
connmanctl> agent on
connmanctl> services         #显示扫描结果
*AO Wired                    ethernet_000000000000_cable
    RP405                    wifi_b00247d9f1cf_5250343035_managed_psk
    RP405-5G                 wifi_b00247d9f1cf_52503430352d3547_managed_psk
    ChinaNet-buUV
wifi_b00247d9f1cf_4368696e614e65742d62755556_managed_psk
    RP405a                   wifi_b00247d9f1cf_525034303561_managed_psk
    rpdzkj-test              wifi_b00247d9f1cf_7270647a6b6a2d74657374_managed_psk
```

```
connmanctl> connect wifi_b00247d9f1cf_5250343035_managed_psk
#连接的时候选择后面的psk,比如我连接RP405,则选中后面的wifi_xxx_psk

connmanctl> connect wifi_b00247d9f1cf_5250343035_managed_psk
Agent RequestInput wifi_b00247d9f1cf_5250343035_managed_psk
    Passphrase = [ Type=psk, Requirement=mandatory ]
Passphrase?          #如果需要密码则会提示你需要输入密码,然后回车确认
connmanctl> quit      #退出交互模式
```

执行 ifconfig 查看wifi信息

```
[root@RV1126 RV1109:/]# ifconfig wlan0
wlan0      Link encap:Ethernet  HWaddr B0:02:47:D9:F1:CF
           inet addr:169.254.149.65  Bcast:169.254.255.255  Mask:255.255.0.0
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:411 errors:0 dropped:0 overruns:0 frame:0
           TX packets:36 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1000
           RX bytes:32031 (31.2 KiB)  TX bytes:13946 (13.6 KiB)
```

如果使用 wpa_supplicant/wpa_cli 的方式,则进行如下操作

```
$ killall ipc-daemon netserver
$ wifi_start.sh RP405 rp778899aa  #参数分别是 wifi 密码
```

如果没有自动获取到 IP 地址,则执行

```
$ udhcpc -i wlan0
```

wifi 热点

创建加密热点,并且共享 eth0 的网络

```
$ create_ap wlan0 eth0 RV1126 00000000
```

创建无网络共享的加密热点

```
$ create_ap -n wlan0 RV1126 00000000
```

创建无加密无网络共享热点

```
$ create_ap -n wlan0 RV1126
```

更多使用方式可以使用 `create_ap -h` 查看

蓝牙

加载蓝牙固件

```
$ bt_init.sh
```

打开蓝牙设备

```
$ hciconfig hci0 up
```

扫描蓝牙设备

```
$ hcitool scan
```

配置开发板设备可被扫描到

```
$ hciconfig hci0 leadv 0  
$ hciconfig hci0 piscan
```

LED 控制

补光灯

```
$ echo 1 > /proc/rp_gpio/bl_led    #开  
$ echo 0 > /proc/rp_gpio/bl_led    #关
```

红外灯

```
$ echo 1 > /proc/rp_gpio/ir_led    #开  
$ echo 0 > /proc/rp_gpio/ir_led    #关
```

10 调试开发

ubuntu 系统相关的说明和注意事项请查看软件文档下的《RP-ubuntu20.04 FAQ.pdf》文档

内核配置

```
$ cd kernel
$ make ARCH=arm menuconfig
```

默认内核 config 文件: `kernel/arch/arm/configs/rv1126-facial-gate.config`, 如果想要保存默认配置, 执行 `make ARCH=arm savedefconfig` 会在 `kernel` 目录下生成 `defconfig`, 建议不要直接替换默认的配置文件, 和默认的配置对比, 在默认配置文件中增加新的配置即可。

Buildroot 配置

```
$ source envsetup.sh rockchip_rv1126_rv1109_facial_gate
$ make ARCH=arm menuconfig
```

建议修改后, 执行如下指令保存为默认配置, 否则重新编 buildroot 时配置会被默认配置覆盖掉。

```
$ make savedefconfig # 保存当前配置为默认配置
```

默认 buildroot config 文件路径:

`buildroot/configs/rockchip_rv1126_rv1109_facial_gate_defconfig`。

主设备树目录

主设备树路径:

版型	主设备树文件
pro-rv1126	kernel/arch/arm/boot/dts/pro-rv1126.dts
pro-rv1109	kernel/arch/arm/boot/dts/pro-rv1109.dts
rp-rv1126	kernel/arch/arm/boot/dts/rp-rv1126.dts
rp-rv1109	kernel/arch/arm/boot/dts/rp-rv1109.dts

交叉编译工具

交叉编译工具链路径:

`prebuilts/gcc/linux-x86/arm/ gcc-arm-8.3-2019.03-x86_64-arm-linux-gnueabihf`

buildroot常用目录

编译出来的文件系统路径:

```
buildroot/output/rockchip_rv1126_rv1109_facial_gate/target/
```

工具下载安装包目录

```
buildroot/dl/
```

rknpu

rknpu 的 demo 所在的路径是:

```
external/rknpu/rknn/rknn_api/examples/
```

版本查看

```
$ strings /usr/lib/librknn_runtime.so | grep build | grep version
librknn_runtime version 1.6.0 (6523e57 build: 2021-01-15 15:56:31 base: 1126)
```

文件上传和下载

使用 lrzsz 工具

```
$ rz          # 上传文件
$ sz filename # 下载文件
```

或者使用scp

```
$ scp rpdzkj@192.168.1.111:/home/test.txt /root # 上传文件
$ scp /root/test.txt rpdzkj@192.168.1.111:/home/ # 下载文件
```

或者使用nfs挂载

```
$ mount -t nfs -o nolock 192.168.1.50:/home/test/ /mnt # 需要先创建 mnt 文件夹
```

或者通过 adb 工具 (工具在网盘链接的工具目录下)

```
$ adb push D:/test/test.txt /root/ # 上传, 注意需要非连接模式下
$ adb pull /root/test.txt D:/test/ # 下载, 注意需要非连接模式下
```

注意图片中的红框:

```
D:\05 工具\adb>adb devices
D:\05 工具\adb>adb push D:/adb/test /userdata/
D:\05 工具\adb>adb pull /userdata/test D:/adb
D:\05 工具\adb>
D:\05 工具\adb>
```

默认用户名和密码

默认用户为 root，密码为 **rockchip**，需要自行通过串口或者 adb 重新设置。

取消开机自启动应用

开机默认启动人脸识别应用，如果需要去掉，可以重命名 `/etc/init.d/S06_qficalGate` 文件为非 S 开头然后重启即可。

11 功能测试

工具目录下有一个功能测试目录，将里面的内容拷贝到开发板，执行 `startTest.sh` 脚本后会运行基于 QT 编写的功能测试引用，可使用此应用测试相关的硬件功能是否正常。