IPC SDK 开发和使用指南

文档版本：V1.1

发布日期：2018-09-10

上海富瀚微电子股份有限公司

地址：上海市宜山路717号2号楼5楼-6楼　邮编：200233
网址：http://www.fullhan.com
传真：021 64066786
电话：021 61121558
邮箱：sales@fullhan.com

商标申明

Fulhan®、富瀚均为上海富瀚微电子股份有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

前言

本文描述了 SDK 各部分的使用方法。

修订记录

修订时间	修订版本	修订人	修订描述
2017-4-27	V1.0	Huxy	初始版本
2018-9-10	V1.1	Huxy	部分章节描述适配多个芯片
2018-10-17	V1.2	Tangyh	增加 ubifs 镜像相关说明

目录

1. SDK 概述和安装

1.1 SDK 模块组成

整个软件开发包有下列部分组成：

- 编译环境与工具链
- 预编译的镜像文件
- 板端调试开发工具
- U-Boot 源码
- Linux 内核源码
- 音视频示例程序
- 图像调试工具
- 相关文档手册

目录结构如下：

```
FHxxxx_IPC_Vx.x_xxxxxxxx 目录结构如下：
    |-- images                      # 可供FLASH烧写的映像文件，如内核、u-boot、Flash.img等
    |-- docs                        # 媒体开发参考手册、开发环境设置
    |-- board_support               # 存放操作系统及相关工具的目录
    |    |-- kernel                 # linux内核源代码
    |    |-- rootfs                 # rootfs源代码
    |    |-- toolchain              # 交叉编译器、打包的C运行时库
    |    |-- tools                  # 板级工具如gdbserver、mtd-utils，和PC工具如mkfs.jffs2、
    |    |                            mkfs.cramfs、mkimage和制作flash升级镜像的工具
    |    `-- uboot                  # uboot源代码
    |-- media_support               # 存放媒体处理平台的目录
        |-- include                 # 对外头文件
        |-- driver                  # 媒体处理内核模块和安装脚本
        |-- lib                     # 媒体库和sensor库，以及sensor配置文件
        |-- tools                   # LOGO图片转换工具、图像效果调试工具等
        `-- demo                    # 样例源代码
```

图 1 SDK 目录结构

在 docs 目录下有《SDK 安装以及升级使用说明.txt》，简要介绍了 SDK 的基本使用方法。同时在 SDK 的大多数子目录下有 readme.txt，描述了该目录下相关组件的使用方法。

1.2 toolchain 安装

目前 SDK 仅支持在 Linux 操作系统上安装、运行。如果当前系统运行的是 Windows，推荐在虚拟机中运行一个 Linux 发行版，目前测试过的发行版是 Ubuntu 12.04。

在 Linux 下安装开发环境步骤如下：

1. 解压 SDK 压缩包：

   $ unrar x FH8xxx_IPC_Vx.x_yyyymmdd.rar

   如果没有 unrar 工具，且使用的是虚拟机，可以先在 Windows 上解压，再通过虚拟机共享文件夹

的方式拷贝至虚拟机中。

2. 安装 toolchain:

   $ cd FH8xxx_IPC_Vx.x_yyyymmdd/board_support/toolchain

   $ ./ct_install.s

2. 启动过程

2.1 启动流程

芯片内部上电后按以下顺序启动

1. 运行芯片内部 ROMBOOT
2. 加载 Flash Bootloader 分区，运行 Bootloader（包含 Ramboot 和 u-boot）
3. 运行 U-Boot
4. 运行 Linux kernel uImage

2.2 U-Boot

可以通过网络、SD 卡、串口、FLASH 加载 Linux kernel，便于调试开发。

2.3 Flash 空间

分区	Start - End Addr	Size（Bytes）	Sector Name
Bootloader 分区	0x00000000 ~ 0x0003FFFF	256K	启动程序和参数
用户分区	0x00040000 ~ 0x0004FFFF	64K	uboot env
	0x00050000 ~ 0x0008FFFF	256K	uboot
	0x00090000 ~ 0x0048FFFF	4M	uImage
	0x00490000 ~ 剩余空间	-	自定义

表格 1 Flash 空间分配

典型 Flash 空间的分配如上表所示。用户可以根据实际使用情况调整 flash 空间分配，在制作 Flash.img 镜像文件时指定每一段的起始地址，具体方法请见第 6 章“Flash 镜像文件制作”。

3. U-Boot 的开发和移植

3.1 编译

U-Boot 的源代码位于 SDK/board_support/uboot，编译方法如下：

1. 进入 SDK/board_support/uboot
2. 执行 make 命令，会自动解压 uboot.tar.gz，并进行编译

具体的编译命令请参见 SDK/board_support/uboot/Makefile 的内容。编译好的 uboot 二进制文件位于 SDK/board_support/uboot/uboot/_build 目录下，可用于制作 Flash.img。

3.2 移植

U-Boot 的板级配置文件是 include/configs/fhxxxx_devboard.h，其中 fhxxxx 根据使用的芯片名字来定，例如：fh8810_devboard.h、fh8830_devboard.h，可根据实际硬件设计和需要，修改其中的默认配置，U-Boot 的驱动移植和开发请参考官方文档。

4. kernel 的开发和移植

4.1 编译

Linux kernel 的源代码位于 SDK/board_support/kernel/linux-3.0.8.tar.gz，编译方法如下：

1. 进入 SDK/board_support/kernel
2. 执行 make 命令，会自动解压 linux-3.0.8.tar.gz，并进行编译

具体的编译命令请参见 SDK/board_support/kernel/Makefile 的内容。同时也可以手动进行解压编译：

1. tar xf linux-3.0.8.tar.gz
2. cd linux-3.0.8
3. make ARCH=arm CROSS_COMPILE=arm-fullhan-linux-uclibcgnueabi- fhxxxx_defconfig
4. make ARCH=arm CROSS_COMPILE=arm-fullhan-linux-uclibcgnueabi- uImage

编译好的 uImage 位于 linux-3.0.8/arch/arm/boot 目录下。

4.2 移植

4.2.1 板级配置

Kernel 的板级配置相关文件位于 linux-3.0.8/arch/arm/mach-fh 目录下，一般是 board-fhxxxx.c，根据芯片型号命名。

4.2.2 IOMUX

对于 FH8810、FH8812 芯片，IOMUX 的定义位于 linux-3.0.8/arch/arm/mach-fh/iomux.c；对于其他型号的芯片，IOMUX 是通过 pinctrl 功能实现的，默认配置位于:

linux-3.0.8/arch/arm/mach-fh/include/mach/board_config.h

在 linux-3.0.8/arch/arm/mach-fh/include/mach 目录下会有 board_config.xxx.appboard 或 board_config.xxx.testboard 这样的文件，是根据芯片型号和开发板类型来命名。根据当前使用的芯片和开发板类型，将对应后缀的 board_config 文件替换为 board_config.h，然后进入 linux-3.0.8 重新编译 uImage。

全部的 IOMUX 定义位于 fhxxxx_iopad_xxx.h，可以手动修改进行复用切换。同时也可以通过 /proc/driver/pinctrl 文件进行动态切换：

echo "dev,设备名称, 0, 0" > /proc/driver/pinctrl

echo "mux,设备名称, 引脚名称, 编号" > /proc/driver/pinctrl

cat /proc/driver/pinctrl

编号指从哪个 mux 出来，具体要看 iopad 头文件中的代码

举例：

echo "dev, GPIO61, 0, 0" > /proc/driver/pinctrl

echo "mux, GPIO61, GPIO61, 0" > /proc/driver/pinctrl

cat /proc/driver/pinctrl

4.2.3 驱动开发

驱动开发请参见文档《Linux 外设驱动开发指南》。

5. rootfs 制作

本章用到的工具都可以在 SDK/board_support/tools 目录下找到。

5.1 如何制作 rootfs

一个基本的嵌入式 rootfs 由 Linux 基本文件系统目录、busybox、udev、C 库组成。

在 SDK/board_support/rootfs 目录和 SDK/board_support/toolchain/runtime_lib 目录下已经提供了这 4 个部分，只需直接 make 即可：

$ cd SDK/board_support/rootfs

$ make

执行完成后，在当前目录下会有一个叫 rootfs_pub 的新目录生成，里面包含了编译好的 busybox 工具、udev 工具、配置文件、C 库。

您可以在此基础上修改或添加文件(如添加应用程序),然后请参照下面的方法制作 CPIO 镜像或 JFFS2、CRAMFS 镜像。

5.2 制作 CPIO 镜像

CPIO 镜像文件用于 initramfs，和 uImage 编译在一起，是一种完全运行于内存中的小型文件系统，一般用来初始化系统功能。

尽量使该镜像文件体积不要太大，以免导致 uImage 过大无法启动。

制作方法如下：

1. 假设 rootfs_pub 目录已经制作好，执行下面的命令：

   $ cd rootfs_pub

   $ sudo chown -R root:root .

   $ find . | cpio -o -H newc > ../rootfs.cpio

   $ cd ..

   $ gzip rootfs.cpio

   此时当前目录下生成 rootfs.cpio.gz，将其拷贝到 SDK/board_support/kernel/linux-3.0.8/usr 目录下，重新编译 kernel 即可

2. 解压 rootfs.cpio.gz，执行下面的命令：

   $ gzip -d rootfs.cpio.gz

   $ mkdir -p rootfs

$ cd rootfs

$ sudo cpio -idv < ../rootfs.cpio

5.3 制作 JFFS2 镜像

使用 mkfs.jffs2 工具，位于 SDK/board_support/tools

方法：

1. 准备好一个文件夹，里面放入要使用的文件，例如 test
2. 执行以下命令：

   $ ./mkfs.jffs2 --pad=0x300000 -d test -e 0x10000 -l -o test.jffs2

   其中：

   --pad=0x300000 选项的值和分区大小保持一致;

   -d test 选项表示要制作镜像的目录名;

   -e 0x10000 选项指定 flash 的擦除大小

   -l 表示小端字节

   -o test.jffs2 指定输出文件名

5.4 烧写 JFFS2 镜像文件

1. 开发板进入 uboot 交互式 shell：

   U-BOOT>

2. 通过 tftp 下载镜像文件：

   U-BOOT> tftp a1000000 test.jffs2

2. 探测 flash：

   U-BOOT> sf probe 0

3. 擦除 flash 分区:

   U-BOOT> sf erase <flash offset> <大小>

4. 写入 flash：

   U-BOOT> sf write a1000000 <flash offset> <大小>

5.5 制作 cramfs 镜像

使用 mkfs.cramfs 工具，位于 SDK/board_support/tools，方法：

1. 准备好一个文件夹，里面放入要使用的文件，例如 test

2. 执行以下命令：

$ ./mkfs.cramfs -b 4096 -N little test test.cramfs

其中：

-b 4096 表示目标系统的分页大小，单位字节

-N little 表示小端字节

5.6 烧写 cramfs 镜像文件

方法和上面烧写 jffs2 镜像文件一样。

5.7 制作 ubifs 镜像

使用 mkfs.ubifs 工具，位于 SDK/board_support/tools，方法：

1. 准备好一个文件夹，里面放入要使用的文件，例如文件夹为 test

2. 执行以下命令：

$ ./mkfs.ubifs -F -m 2048 -e 124KiB -c 50 -r ./test ubifs.img

其中：

-F 必选，否则镜像无法重复加载

-m 读写单位，对于 nanflash 来说就是页大小，一般为 2048 字节

-e ubi 逻辑块大小，一般为物理块（128KiB）减去两页（4KiB），就是 124KiB

-c 最大逻辑块号，镜像大小不能超过它

-r 输入路径

ubifs.img 输出镜像的名字

3，生成 ubifs.img 后还要重新生成卷管理的镜像，使用 ubinize 工具，位于 SDK/board_support/tools，ubinize.config，也位于 SDK/board_support/tools。

方法：

执行下面命令:

$ ./ubinize -o ubi.img -p 128KiB -m 2048 ubinize.config

其中：

-o 输出卷管理镜像的名字

-p flash 物理块大小，参考器件手册一般为 128KiB

-m flash 页大小 ，参考器件手册 一般为 2048

ubinize.config 配置文件

4，ubinize.config 配置文件说明

[ubifs]

mode=ubi

image=ubifs.img 输入 Ubi 文件系统的镜像

vol_id=0 卷 ID 号

vol_size=30MiB 卷大小设置

vol_type=dynamic 卷类型设置为动态卷

vol_name=rootfs 卷名字

vol_alignment=1 卷表对齐大小

vol_flags=autoresize

5.8 烧写 ubifs 镜像文件

1. 开发板进入 uboot 交互式 shell：

   U-BOOT>

2. 通过 tftp 下载镜像文件：

   U-BOOT> tftp a1000000 ubi.img

3. 擦除 flash 分区:

   U-BOOT>nand erase <flash offset> <大小>

4. 写入 flash：

   U-BOOT> nand write a1000000 <flash offset> <大小>

5.9 Ubifs 文件系统挂载

1. 开发板进入到 kernel ,安装网络文件系统 nfs 目录

2. 拷贝 SDK/board_support/tools/ubiattach 工具到 nfs 目录下

3. 利用工具 ubiattach,执行下面命令

./ubiattach /dev/ubi_ctrl -m 5

-m MTD 设备号，即 ubi.img 所在的 FLASH 分区的 MTD 设备

执行完后会在/dev/下生成相应的 UBI 设备 ubi0 和卷设备 ubi0_0。

4.卷设备上创建文件系统

mkdir /mnt/ubi0

mount -t ubifs /dev/ubi0_0 /mnt/ubi0

5.10 设置 u-boot bootargs 使 kernel 挂载 flash 分区

1. rootfs 放在 490000 开始的地方，使用 cramfs：

   以 16M SPI Flash 为例。

   1） 地址空间说明，以下的操作均基于图示的地址空间分配，您也可以根据实际情况进行调整

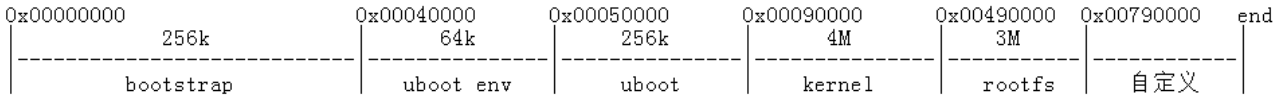

图 2 Flash 地址空间示例

   2） 设置 bootargs：

   set bootargs 'console=ttyS0,115200 ip=192.168.1.3 root=/dev/mtdblock4 rootfstype=cramfs mem=32M mtdparts=spi_flash:256k(bootstrap),64k(uboot-env),256k(uboot),4M(kernel),3M(rootfs)'

2. rootfs 放在 490000 开始的地方，使用 jffs2，flash 地址空间以上图为例，bootargs 设置为:

   set bootargs 'console=ttyS0,115200 ip=192.168.1.3 root=/dev/mtdblock4 rootfstype=jffs2 mem=32M mtdparts=spi_flash:256k(bootstrap),64k(uboot-env),256k(uboot),4M(kernel),3M(rootfs)'

5.11 busybox 配置

1. 解压 busybox-1.19.3.tar.bz2：

   $ tar xf busybox-1.19.3.tar.bz2

2. 在 SDK/board_support/rootfs/busybox 目录下有：

   - config_busybox_1_19_3_debug： 包含调试需要的工具
   - config_busybox_1_19_3_test： 包含很多测试需要的工具
   - config_busybox_1_19_3_release： 包含基本系统必需的工具

   将 config_busybox_1_19_3_release 拷贝到解压的 busybox-1.19.3 目录下

   将 Makefile_busybox_1_19_3 拷贝到解压的 busybox-1.19.3 目录下，重命名为 Makefile

3. 执行：

```
$ mv config_busybox_1_19_3_release .config
$ make menuconfig #然后根据需要选择，并保存
$ make CROSS_COMPILE=arm-fullhan-linux-uclibcgnueabi-
$ make install
```

编译完成，在解压的 busybox-1.19.3/_install 目录下，即是编译好的工具

6. Flash 镜像文件制作

6.1 简介

本章描述的方法制作出 Flash.img，可以直接烧写到 Flash 芯片上。Flash.img 的内容一般由几个二进制文件合成，按照第 2 章的描述，Flash.img 一般由 RamBoot、U-Boot、uImage 合成。

6.2 制作方法

1. 准备各个二进制文件：
   - RamBoot：SDK/images 目录下有编译好的二进制文件 RamBoot.bin
   - U-Boot：SDK/images 目录下有编译好的二进制文件 u-boot.bin，也可以通过 U-Boot 源码编译
   - uImage：SDK/images 目录下有编译好的二进制文件 uImage，也可以通过 kernel 源码编译
2. 将上述几个文件放入 SDK/board_support/tools/flash_upgrade 目录下
3. 进入 SDK/board_support/tools/flash_upgrade，执行：

   ./mkflashimg　　ini 文件名（用于 FH8830、FH8630D、FH8630M）

或　./mkflashimgv4　ini 文件名（用于 FH8632、FH8856、FH8852）

7. APP 开发

7.1 demo 简介

音视频相关的文件都放在 SDK/media_support 目录下：

- demo：示例代码
- driver：音视频驱动
- driverv2：FH8812 平台音视频驱动
- include：音视频相关头文件
- lib：音视频相关静态库、动态库
- libv2：FH8812 平台音视频相关静态库、动态库
- tools：音视频辅助开发工具

下面是 demo 目录下的示例代码功能说明：

- common：公共目录，存放通用代码，通常是将各个示例代码都会用到的代码片段封装到这里，目前是将 ISP 相关的代码放在这里
- audio：音频 API 示例
- audio_convert：演示音频编解码功能
- cover_detect：遮挡侦测 API 示例
- crypto： AES 加密 API 示例
- face_detect：人脸检测示例（仅部分 SDK 支持）
- fisheye：鱼眼正方形分辨率示例（仅部分 SDK 支持）
- gpio：GPIO 示例，演示控制 LED 灯闪烁和读取 GPIO 中断
- mjpeg：通过 HTTP 发送 MJPEG 码流到客户端，客户端可以通过 VLC 或浏览器（Firefox、Chrome）观看，输入 http://<开发板 IP>:8080/?action=stream
- motion_detect：移动侦测 API 示例
- overlay：OSD、LOGO、MASK API 示例
- smart_enc：智能编码示例（仅部分 SDK 支持）
- venc：多通道视频编码示例，包括获取 JPEG、YUV 数据的示例、图像旋转、分辨率切换、帧率切换的示例
- vlcview：通过 UDP 发送码流到 VLC 客户端，可以实时观看图像

7.2 编译 demo

可以在 demo 目录这一层执行 make，也可以单独进入一个 demo 子目录执行 make，同时需要指定 sensor 名字：

make SENSOR=名字

例如：

make SENSOR=ov9732

Sensor 名字需要使用小写，和 lib 目录下的 hex 文件名_attr.hex 前半部分一致。

对于 FH881X SDK，编译时需要指定芯片类型，例如：

make CHIP=FH8810

make CHIP=FH8812

7.3 运行 demo

在开发期间，一般是通过 NFS 将 PC 主机上的目录挂载到开发板的系统上，然后在开发板上运行测试程序。这里以 NFS 挂载为例描述如何搭建运行 demo 的环境。

1. 将 media_support/driver 目录下的所有文件拷贝到 NFS 目录下
2. 将 media_support/lib 目录下对于 sensor 名字的 hex 文件拷贝到 NFS 目录下
3. 将编译好的 demo 程序，例如 sample_vlcview 拷贝到 NFS 目录下
4. 在开发板上进入 NFS 挂载目录，执行：

   ./load_modules.sh

   ./sample_vlcview　电脑 IP 地址　端口号
5. 在电脑上，打开 VLC 软件，按 CTRL+N 快捷键，输入：

   udp://@:端口号

8. SDK 工具使用说明

SDK 工具分为板级开发工具和视频开发工具，分别位于:

SDK/board_support/tools

SDK/media_support/tools

8.1 编译 cramfs 工具

1. 进入 SDK/board_support/tools/cramfs_tool
2. 执行 make

8.2 编译 mtd ubi yaffs2 工具

1. 进入 SDK/board_support/tools/mtd-utils
2. 执行 make board_tools，编译出的工具是运行在开发板上的，包括：

   mtdinfo

   ubicrc32

   ubiformat

   ubinfo

   ubirename

   ubirsvol

   ubiattach

   ubidetach

   ubimkvol

   ubirmvol

   ubiupdatevol

   flashcp

   flash

   erase

   mtd_debug

3. 执行 make pc_tools，编译出的工具是运行在 PC 上的，包括：

   mkfs.jffs2、mkfs.ubifs、ubinize、mkyaffs2image

8.3 使用 mkimage

主要用于编译 kernel 时将 zImage 转换为 uImage，只需放到系统环境变量 PATH 中的任意一个路径下面即可被 kernel 中的 Makefile 使用。一般可以放到/usr/local/bin 目录下。

8.4 使用 gdbserver

运行在开发板上，PC 端通过运行 arm-fullhan-linux-uclibcgnueabi-gdb 进行远程调试。使用方法如下：

1. 按照 7.3 节的方法搭建 demo 运行环境
2. 将 gdbserver 工具拷贝到 NFS 目录
3. 执行:

   ./gdbserver　PC 机 IP 地址:调试端口号 ./sample_vlcview　　PC 机 IP 地址

4. 在 PC 机上执行：

   arm-fullhan-linux-uclibcgnueabi-gdb

5. 进入(gdb)交互式界面后，执行：

   target remote　开发板 IP 地址:调试端口号

   如果连接正常，开发板会打印：Remote debugging from host

6. 此时在 PC 机上可以执行 gdb 命令进行调试

8.5 使用 iperf

iperf 用于测量 TCP、UDP 带宽性能。Iperf 运行在开发板上，iperf_x86 运行在 PC 机上。用法如下：

在 PC 上运行 iperf 作为 server：

TCP Server:

```
./iperf　-s
```

UDP Server:

```
./iperf　-u -s
```

在开发板上运行：

TCP：

```
./iperf -c 192.168.1.110 -m -M 88 -f M -d -t 60 -w 8K
```

UDP：

./iperf -c 192.168.1.110 -w 128k -l 1500 -f M -u -t 60 -b 1M

8.6 使用 iwlist

iwlist 用于扫描无线热点，使用方法：

iwlist wlan0 scanning

8.7 使用 iwconfig

iwconfig 可以查看无线网卡的状态，例如是否加入了热点；同时还可以用来加入无密码的热点：

iwconfig wlan0 essid "TPLINK"

8.8 使用 wpa_passphrase

用于生成无线热点配置文件，为后续使用 WPA 加密方法加入热点做准备：

wpa_passphrase SSID PASSWORD > wpa_supplicant.conf

8.9 使用 wpa_supplicant

使用 WPA 加密方法加入热点：

wpa_supplicant -Dwext -iwlan0 -c./wpa_supplicant.conf –B

参数含义：

- -D: 使用 wext 接口和 wifi 驱动通信
- -i: 指定 wifi 接口名字
- -c: 指定配置文件
- -B: 以后台进程方式运行

8.10 使用 regrw

regrw 可以直接读写芯片的寄存器。不加参数运行，可以打印出帮助信息和使用示例。

读寄存器：

regrw -r 寄存器地址

写寄存器：

regrw -w 寄存器地址 寄存器值

修改寄存器某个 bit 值：

regrw -m 寄存器地址 第几个 bit bit 值

8.11 使用 PIC2ARGB

将常用的图片如 JPG、PNG 转换为 ARGB 格式。使用方法：

1.双击 PIC2ARGB.exe

2.将源图片拖拽到窗口上部的图片预览区域

3.程序下方选项中，"data"选择 "ARGB15555"

4.使用鼠标将程序窗口上部的图片预览区域拖拽到程序窗口外面任意地方，会生成转换好的文件

选项说明：

file

- Hex: 生成的文件为二进制格式
- Array：生成的文件为一个头文件，将图片数据保存为 C 语言数组

8.12 使用 CoolView

将 rar 文件解压，双击 CoolViewPro.exe 即可运行。

连接开发板：

1. 点击工具栏齿轮图标，或者按键盘 F10
2. 在打开的对话框中，可以选择“串口”或“网络”，注意勾选上方的启用选框。一般选择“网络”
3. 修改“本机 IP 地址”、“远程 IP 地址”（即开发板的 IP）
4. 修改“远程端口号”和开发板上应用程序运行 debug interface 的端口号保持一致
5. 如果开发板上的 debug interface 线程使用 TCP，需要勾选“启用 TCP”
7. 点击“保存”按钮，成功后，窗口右下角的状态图标变为绿色

修改 ISP 配置：

连接成功后，在左边“详细列表”中选择功能模块，在右侧区域进行具体的配置。

9. WiFi 驱动编译与使用

请参见文档《Linux 外设驱动开发指南》中的说明。