

# S9188 驱动 RT-Thread 操作系统平台使用指南



# 目 录

版本修改.....	1
前言.....	2
第 1 章 驱动简介.....	3
1.1 驱动接口.....	3
1.2 驱动功能.....	3
1.3 驱动目录结构.....	3
1.4 资源需求.....	3
第 2 章 驱动配置.....	5
2.1 组件配置.....	5
2.2 固件配置.....	10
2.2.1 Romfs 使用.....	10
2.2.2 其它文件系统使用.....	12
2.3 驱动配置编译.....	15
2.4 调试信息打印等级.....	19
第 3 章 驱动加载、连接调试.....	21
3.1 STA 模式.....	21
3.2 AP 模式.....	25

## 版本修改

版本号	修改记录	描述	修改人员	时间
V1.00	创建	创建文档	任海波	2021-09-18

## 前言

本文档主要描述本公司的 WIFI4 芯片 S9188 在 RT-Thread 操作系统下的使用指南，详细描述 S9188 驱动需要的硬件资源、驱动配置、编译及连接操作使用方法，为用户使用 S9188 进行项目开发提供指导。

本文档仅限于讲述 S9188 芯片驱动的配置使用方法，芯片的参数性能指标及 WIFI 协议支持相关请参考具体芯片手册。

阅读本文档前，用户需要先熟悉 RT-Thread 操作系统。RT-Thread 详细学习指南可以参考官方文档中心，连接地址：

<https://www.rt-thread.org/document/site/#/rt-thread-version/rt-thread-standard/README>。

S9188 的驱动和具体的硬件平台无关，本文档以 STM32H750VBT6 的开发板为例，详细介绍讲驱动配置、源码加入到工程编译、下载编译文件到目标板、WIFI 调试连接的方法。

## 第 1 章 驱动简介

S9188 是山东兆通微电子研发的 b/g/n 标准的 WIFI 芯片，分为 USB 接口的 S9188U 和 SDIO 接口的 S9188S。RT-Thread 操作系统下当前只支持 SDIO 接口的 S9188S，本文档主要描述 SDIO 接口的 S9188 驱动的配置、编译使用方法。

### 1.1 驱动接口

RT-Thread 操作系统下，当前驱动只支持 SDIO 接口，USB 接口未来会支持。

### 1.2 驱动功能

RT-Thread 操作系统下，S9188 WIFI 驱动只支持 STA 或 AP 模式，且主要支持 STA 模式，不支持 ADHOC、P2P 等模式。

STA 模式：安全模式支持 Open、WEP-SharedKey、WEP-OpenSystem、WPA-PSK、WPA2-PSK、WPA/WPA2-PSK；加密模式支持 AES、TKIP 及 AES/TKIP 方式。

AP 模式：安全模式只支持 Open 模式，即开放式网络。

由于 RT-Thread 的 WLAN 框架对网卡热插拔处理不支持，SDIO 框架对 SDIO 卡的热插拔支持也不是很好，所以本驱动暂时不支持热插拔。系统上的默认加载网卡驱动，不能卸载网卡。

### 1.3 驱动目录结构

S9188 驱动的目录结构如图 1.1 所示。下面详细介绍下各个目录。

名称	修改日期	类型	大小
fw	2021/9/13 15:43	文件夹	
nic	2021/9/13 16:06	文件夹	
os	2021/9/15 14:41	文件夹	
Kconfig	2021/9/15 14:42	文件	2 KB
Readme.md	2021/9/13 15:43	MD 文件	1 KB
SConscript	2021/9/14 10:53	文件	2 KB
wifi.cfg	2021/8/9 16:57	CFG 文件	1 KB

图 1.1 驱动目录结构

- fw: 发布的固件存放
- nic: 驱动核心部分
- os: 驱动操作系统相关移植及适配文件
- Kconfig: RT-Thread 的 menuconfig 配置需要的配置文件
- Readme.md: 驱动简要说明
- Sconscript: 驱动编译构建脚本
- wifi.cfg: wifi 工作配置文件

### 1.4 资源需求

当前驱动默认配置需要资源：

CODE SIZE: 185K 左右

全局 RAM 资源：12K 左右

Heap RAM 资源：356K 左右

以上资源占用在 IAR EWARM 8.40 版本环境测试，编译优化等级为 Low，其它环境或有差异。本资源使用情况不包括 wifi 本身固件的占用。如果将 wifi 固件用到 romfs 和系统编译到一起，则需要的 code size 会增加固件大小。

驱动设计网络收发的内存都采用预分配的方式，预分配分配的内存越大，网络性能越好。故在内存资源充足的情况下，建议收发缓冲区配置尽量大一些。

## 第 2 章 驱动配置

### 2.1 组件配置

S9188 驱动依赖组件有 RT\_USING\_HEAP、RT\_USING\_SDIO、RT\_USING\_WIFI、RT\_USING\_LWIP、RT\_USING\_DFS。下面描述各个组件的用途：

- RT\_USING\_HEAP: S9188 驱动实现使用很多动态内存分配，当然需要使能 HEAP。
- RT\_USING\_SDIO: S9188 SDIO 接口驱动需要使用 SDIO 框架。本 WIFI 驱动要求硬件平台支持基于 SDIO 框架的驱动，且支持 SDIO 卡的 SDIO 中断功能。
- RT\_USING\_WIFI: S9188 驱动符合操作系统的 WIFI 框架。
- RT\_USING\_LWIP: S9188 的 WIFI 网卡需要运行基于 LWIP 的 TCP/IP 协议栈。
- RT\_USING\_DFS: S9188 的固件存放在文件系统中。可以使用 romfs 或 elm fat，也可以使用 littlefs 等等，用户可以随意选择使用哪种文件系统。

在使用的 bsp 目录打开 rt-thread 的 env 环境，如图 2.1 所示。下面逐个配置需要的组件。

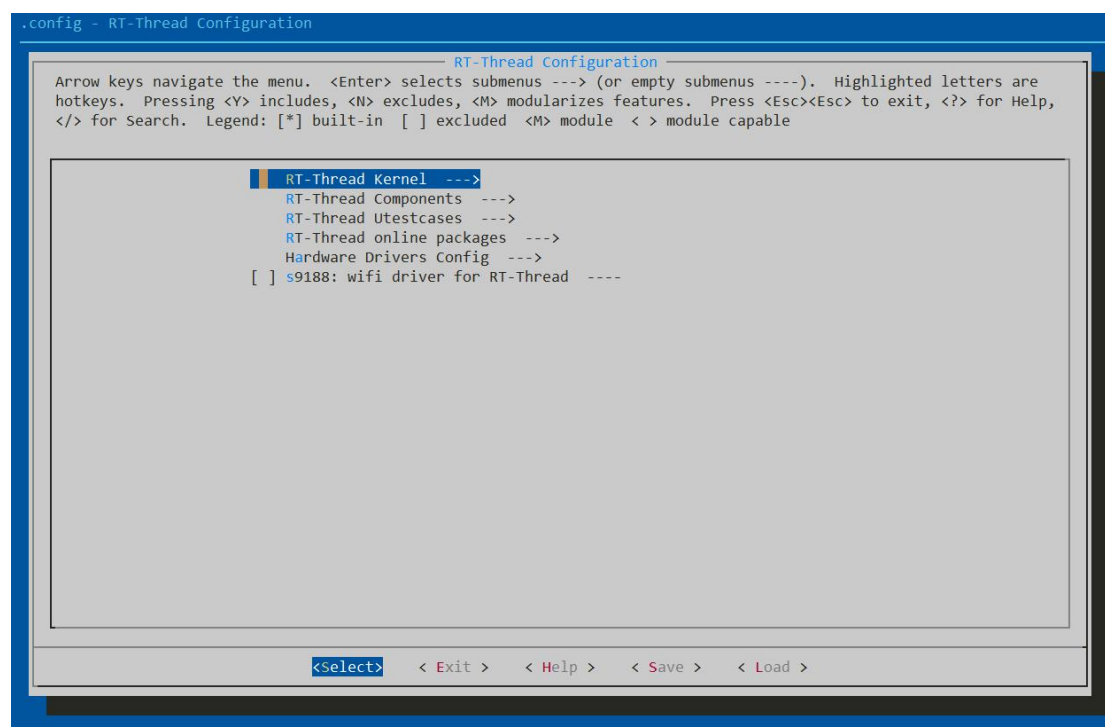


图 2.1 env 运行界面

Heap 配置如图 2.2 所示。S9188 驱动使用了标准 C 的文件读写接口，因此也需要使能 libc 的 api，配置如图 2.3 所示。

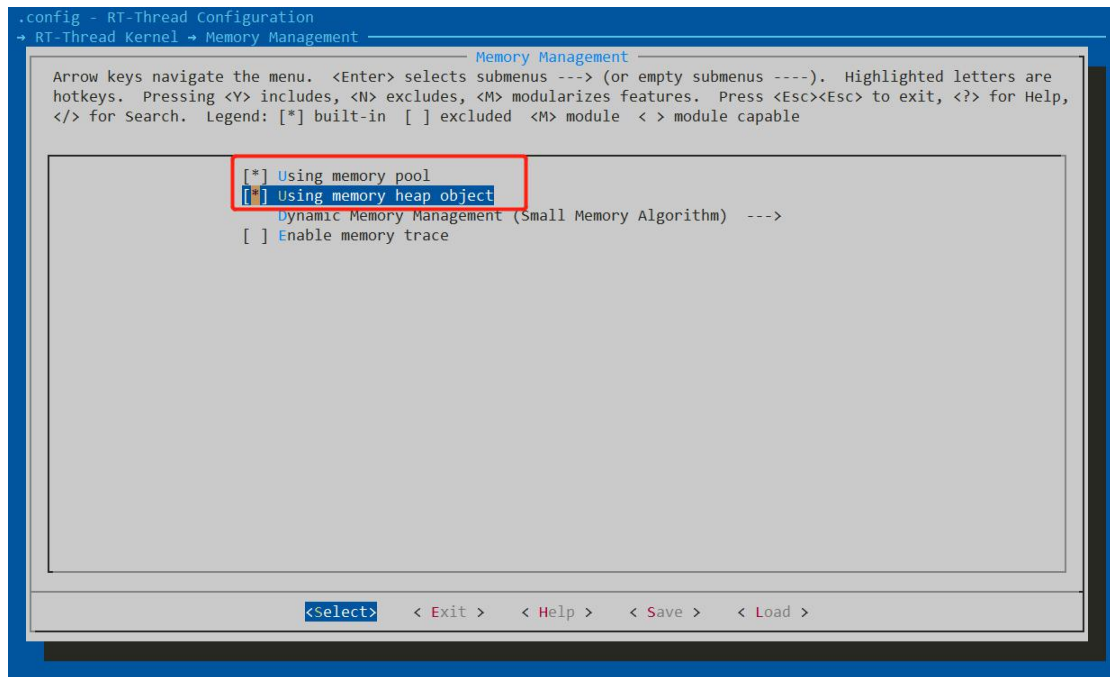


图 2.2 Heap 配置

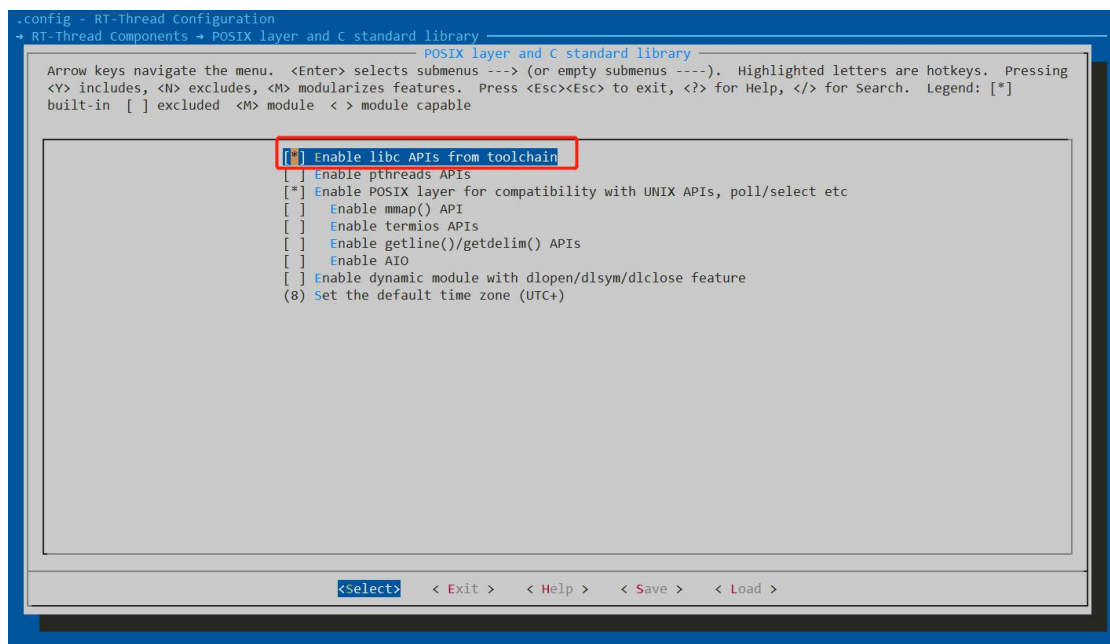


图 2.3 libc 库 api 使能

SDIO 框架配置如图 2.4 所示。用户可以适当调整 sdio 中断线程的优先级，以便更合理使用驱动。



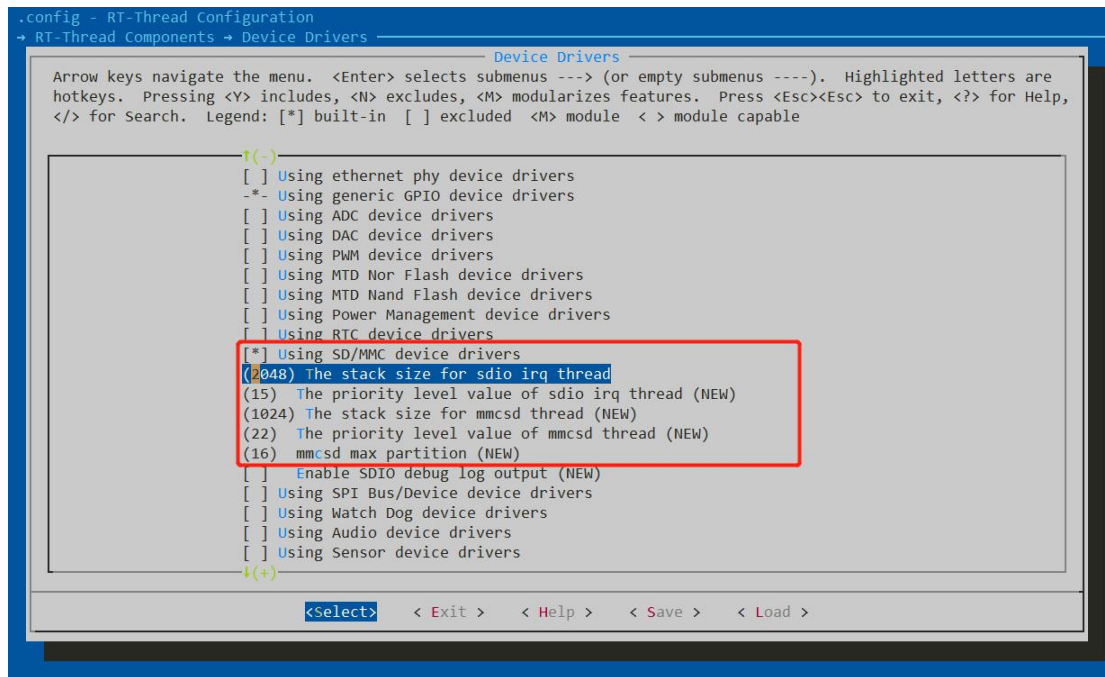
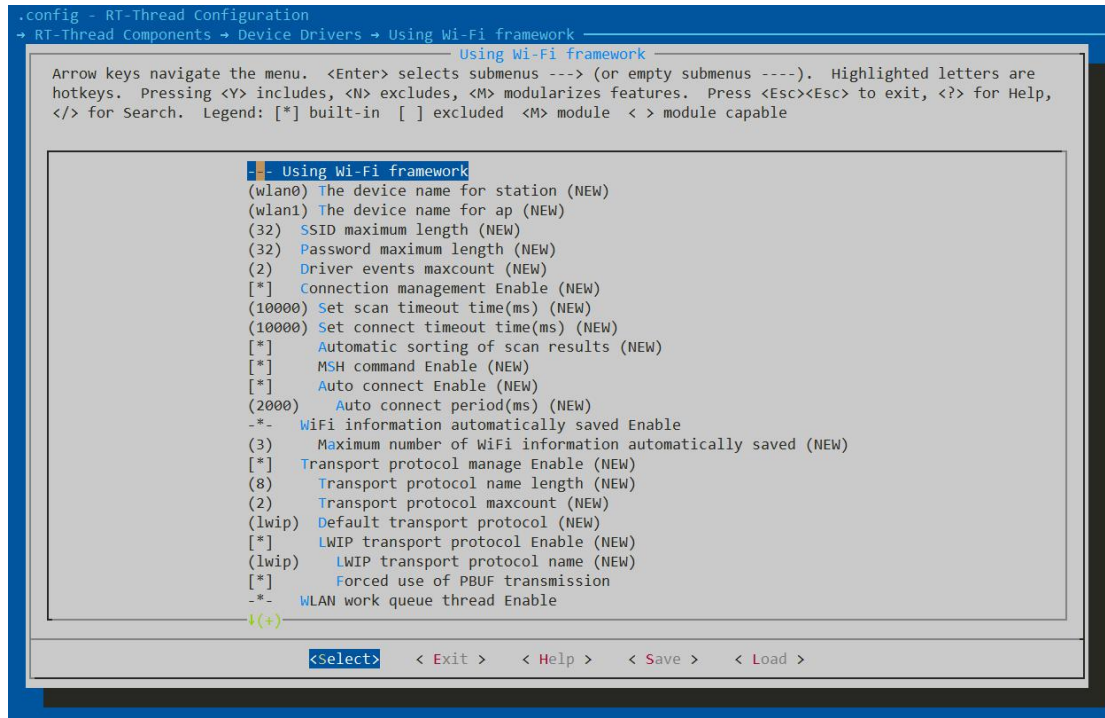


图 2.4 sdio 框架配置

WLAN 框架配置如图 2.5 所示。S9188 WIFI 使用基于 lwip 协议栈开发，数据收发缓冲结构要求使用 lwip 的 pbuf 结构，因此，Force use of pbuf transmission 必须使能。其它配置可以根据需求调整。



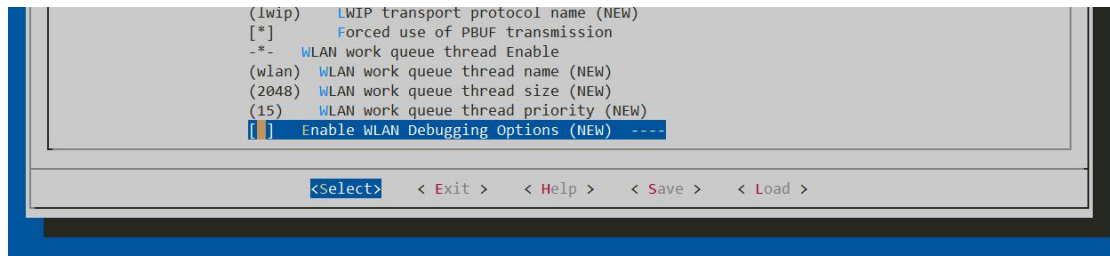


图 2.5 wlan 框架配置

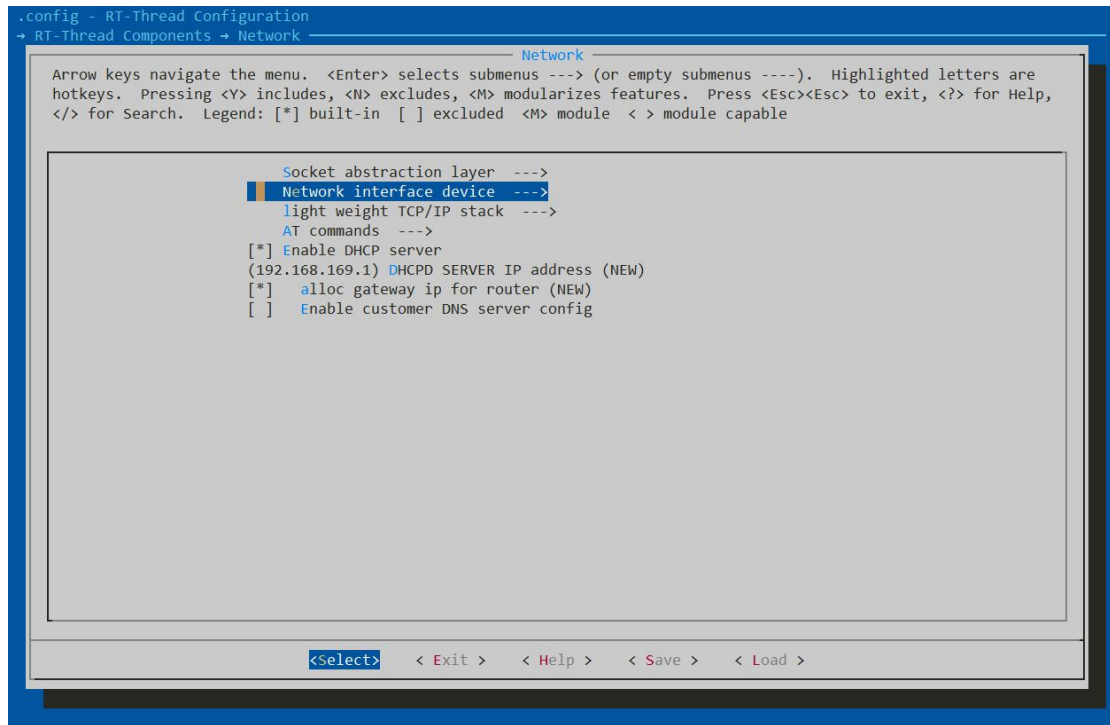


图 2.6 dhcp server 使能配置

网络协议栈 LwIP 配置。首先，AP 模式下需要使能 DHCP Server，以便为连接的 AP 提供 IP 地址，配置如图 2.6 所示；然后选择 light weight TCP/IP stack，进入进行配置，配置如图 2.7 所示，该配置根据项目应用需求可以适当调整，由于 dhcpd 的要求，lwip 尽量选择 2.1.2 版本；最后 S9188 驱动对 lwip 的 pbuf 的 buf\_size 有一个依赖，如程序清单 2.1 所示，由于该项配置 rt-thread 的 menuconfig 未提供配置界面接口，请在使用 bsp 的目录文件 rtconfig.h 增加定义如下：

```
#define RT_LWIP_PBUF_POOL_BUFSIZE 1580
```

用户也可以通过修改 Kconfig 增加 menuconfig 配置该项，如果直接添加代码，每次通过 menuconfig 修改都会被覆盖而导致配置丢失。

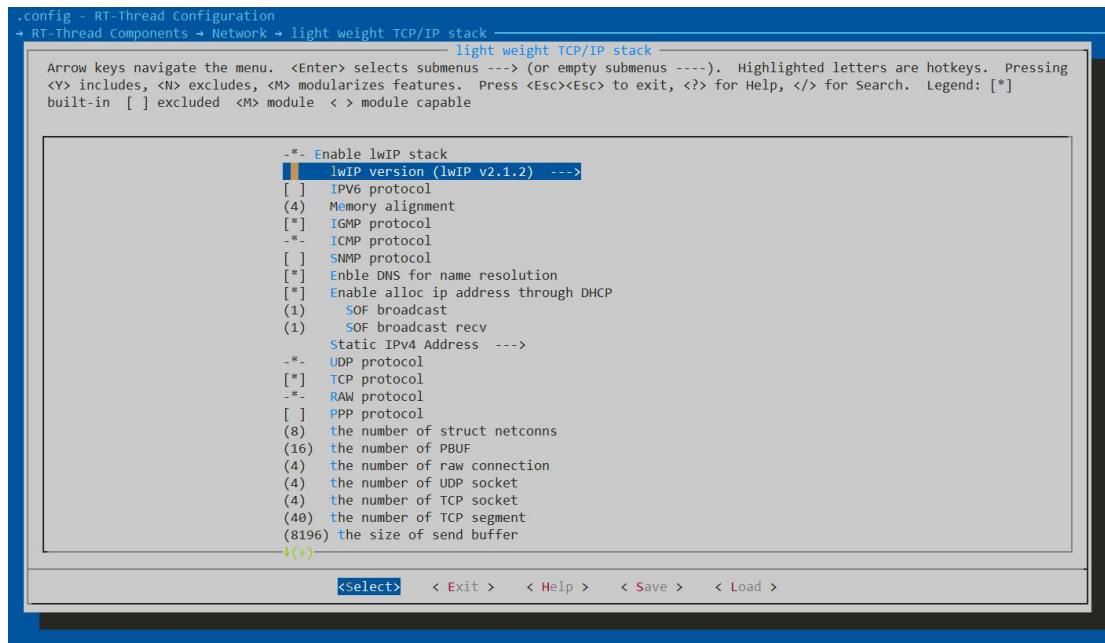


图 2.7 lwip 协议栈基本配置

## 程序清单 2.1 lwip 特殊配置需求

```

#if (PBUF_POOL_BUFSIZE < 1580)
#error "It must config PBUF_POOL_BUFSIZE >= 1580 in lwipopts.h"
#endif

```

DFS 配置如图 2.8 所示。这个是 dfs 的基本配置，这里使能了 ROMFS。如果用户要使用 littlefs，请通过 env 在 rt-thread 的在线软件包配置并下载，如图 2.9 所示。

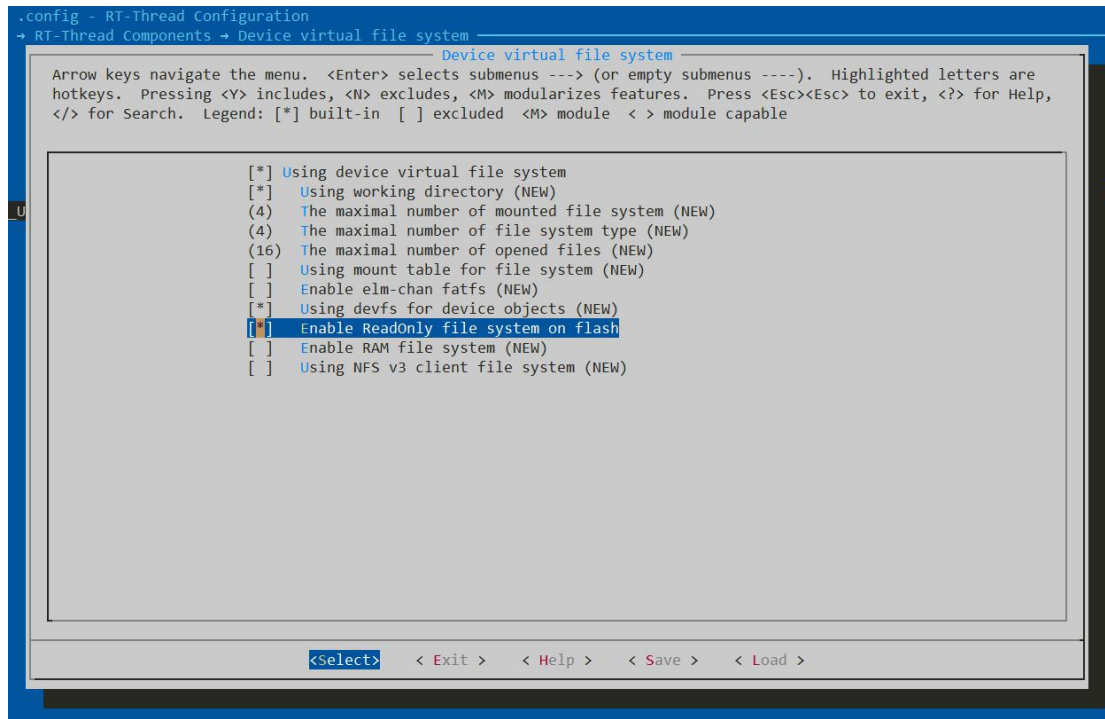


图 2.8 dfs 使能配置

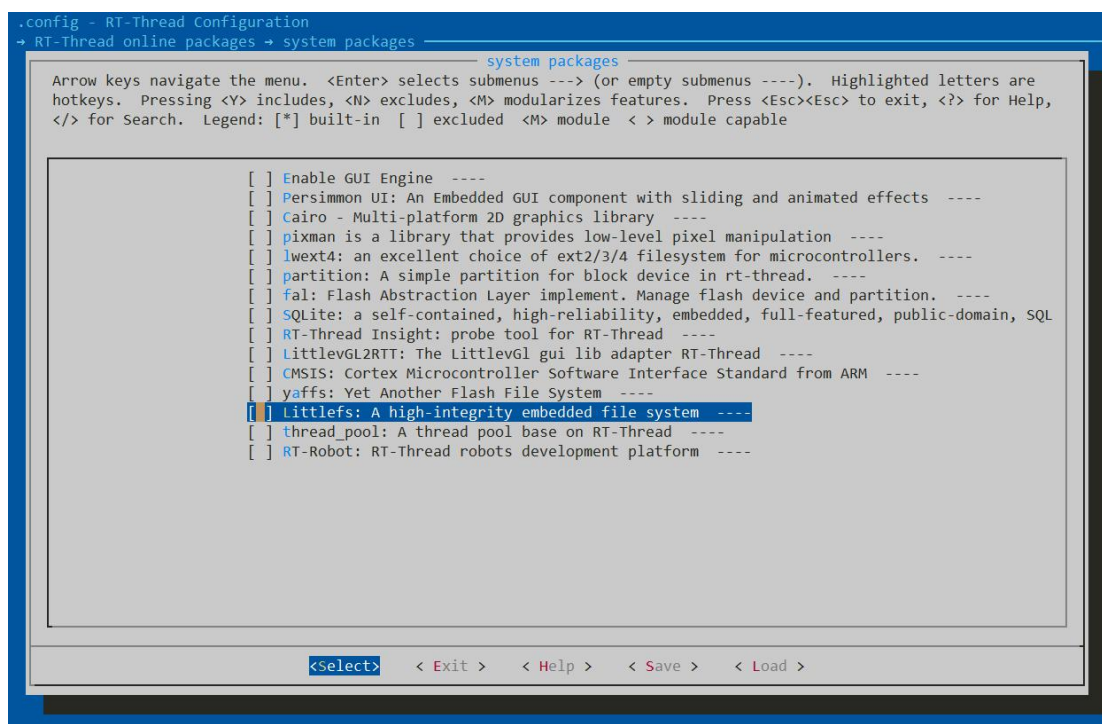


图 2.9 在线软件包 littlefs 选择

## 2.2 固件配置

S9188 的驱动设计固件存放在文件系统中，SDIO 卡枚举到的时候从文件系统读固件，下载到 WIFI 芯片中，使其运行起来，完成 WIFI 的管理工作。本节介绍固件使用 romfs 存储和其它可写文件系统存储的方法。

### 2.2.1 Romfs 使用

Romfs 的使用说明在 os/rt-thread/romfs 目录，目录结构如图 2.10 所示。




 s9188_romfs	2021/8/25 10:30	文件夹	
 mkromfs.py	2021/6/15 10:39	Python File	9 KB
 ROMFS创建方式.txt	2021/9/15 18:01	文本文档	1 KB

图 2.10 romfs 使用说明目录

目录结构内容描述：

- ROMFS 创建方式.txt：该文件是具体的使用说明。
- mkromfs.py：该文件是 romfs 构建 python 脚本，在 rt-thread 的 env 环境中运行该脚本。
- 目录 s9188\_romfs：该目录是 romfs 文件系统的目录结构。该目录创建 etc 目录，里面放入配置文件 wifi.cfg；创建 lib/firmware 目录，放入固件 fw\_9188\_r1751.bin。

Romfs 文件系统构建流程：

- 1、通过 menuconfig 或者其它方式使能 romfs 文件系统，如图 2.11 所示。
- 2、在 os/rt-thread/romfs 目录打开 rt-thread 的 env 环境，执行命令 `python mkromfs.py s9188_romfs romfs.c`，即可生成 romfs.c 的文件
- 3、将 romfs.c 加入工程

4、将程序清单 2.2 中的代码加入工程，切记如果同时存在多种文件系统，挂载目录和其它文件系统不能冲突。

程序清单 2.2 romfs 文件系统挂载代码实现

```
int romfs_mnt(void)
{
    if (dfs_mount(RT_NULL, "/", "rom", 0, &(romfs_root)) == 0) {
        rt_kprintf("ROM file system initialized!\n");
    } else {
        rt_kprintf("ROM file system initialize failed!\n");
        return 0;
    }
}

INIT_ENV_EXPORT(romfs_mnt);
```

此时，文件系统即挂载到系统中，可以通过命令查看下目录结构，如图 2.12 所示。

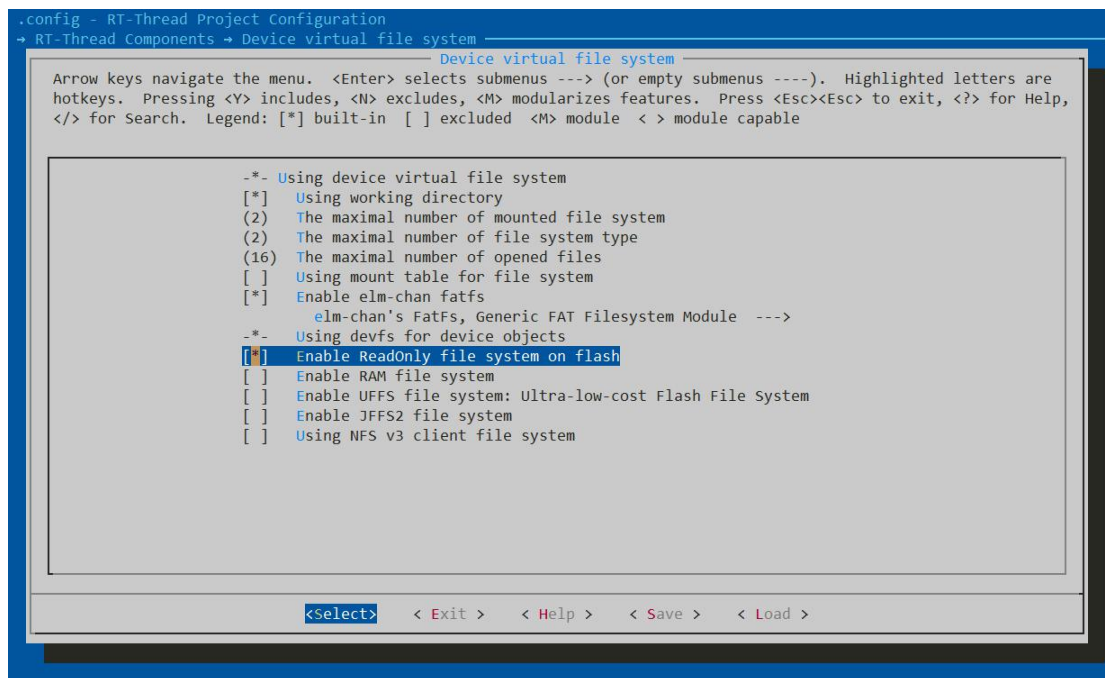


图 2.11 配置使能 romfs 文件系统

```
msh />ls
Directory /:
etc          <DIR>
lib          <DIR>
msh />ls /etc
Directory /etc:
wifi.cfg     300
msh />ls /lib/firmware
Directory /lib/firmware:
fw_9188_r1751.bin 53298
msh />
```

图 2.12 ROM 文件系统查看

## 2.2.2 其它文件系统使用

S9188 驱动基于 STM32H750 的硬件平台测试，硬件平台上板载了 16MB 的 SPI Flash W25Q128。本例子将使用运行在 SPI Flash 上的 littlefs 文件系统，首先需要编写 SPI Flash 的 MTD 设备驱动，SPI Flash 的 MTD 设备驱动不属于本文讲述的内容。SPI Flash 的 MTD 驱动注册后，就可以挂载 littlefs 文件系统了，挂载代码如程序清单 2.3 所示。运行效果如图 2.13 所示。

首先，通过命令构建 littlefs 的文件系统目录结构，操作命令如图 2.14 所示。

程序清单 2.3 littlefs 文件系统挂载

```
int mnt_init(void)
{
    if (dfs_mount("flash0", "/", "lfs", 0, 0) == 0)
    {
        rt_kprintf("flash0 mount to / ok!\r\n");
    }
    else
    {
        rt_kprintf("flash0 mount to / failed!, fatmat and try again!\n");

        /* fatmat filesystem. */
        dfs_mkfs("lfs", "flash0");

        /* re-try mount. */
        if (dfs_mount("flash0", "/", "lfs", 0, 0) == 0)
        {
            rt_kprintf("flash0 mount to / ok!\r\n");
        } else {
            rt_kprintf("flash0 can't mount to /, and can't format\r\n");
            for(;;)
            {
                rt_hw_led_toggle(0);
                rt_thread_delay(RT_TICK_PER_SECOND * 2);
            }
        }
    }

    return 0;
}

INIT_ENV_EXPORT(mnt_init);
```

然后，通过 HTTP 或者 TFTP 下载文件到文件系统。STM32H750 的开发板具有以太网接口，因此可以通过网络接口下载驱动。下面介绍通过 TFTP 下载 S9188 配置文件和固件的方法。打开 TFTP 服务器软件，如图 2.15 所示；将 S9188 的配置文件和固件放入 TFTP 服务器的工作目录，如图 2.16 所示；在 rt-thread 的 shell 终端输入 tftp 命令，可以查看 tftp 命令的帮助，如图 2.17 所示；通过 tftp 下载 S9188 配置文件 wifi.cfg 到/etc 目录，操作步骤如



图 2.18 所示；同理，通过 tftp 下载 S9188 固件文件 fw\_9188\_r1751.bin 到目录/lib/firmware 中，操作步骤如图 2.19 所示。其它方法，在电脑运行 HTTP 服务器，通过 wget 也可以下载固件和配置文件。如果使用的硬件没有网口，建议使用 romfs，如果一定想使用可写文件系统，可以使用 rt-thread 的 y-modem 组件进行下载。

```

\ | /
- RT -      Thread Operating System
/ | \      4.0.3 build Sep 16 2021
2006 - 2020 Copyright by rt-thread team
[I2C] I2C bus [i2c1] registered
Init lwip tcp/ip stack!
lwIP-2.2.0 initialized!
Init ethernet driver frame!
[E/DFS] mount fs[elm] on /mnt failed.

flash0 mount to / ok!
sdmmc clock set to 0Hz
msh />sdmmc clock set to 375000Hz
sdmmc clock set to 375000Hz
sdmmc clock set to 375000Hz
df
disk free: 15.9 MB [ 4094 block, 4096 bytes per block ]
msh />
msh />ls
Directory /:
msh />

```

图 2.13 spi flash 挂载 littlefs 文件系统查看

```

msh />mkdir /etc
msh />mkdir /lib
msh />mkdir /lib/firmware
msh />ls /etc
Directory /etc:
msh />ls /lib
Directory /lib:
firmware          <DIR>
msh />ls /lib/firmware
Directory /lib/firmware:
msh />

```

图 2.14 构建目录结构

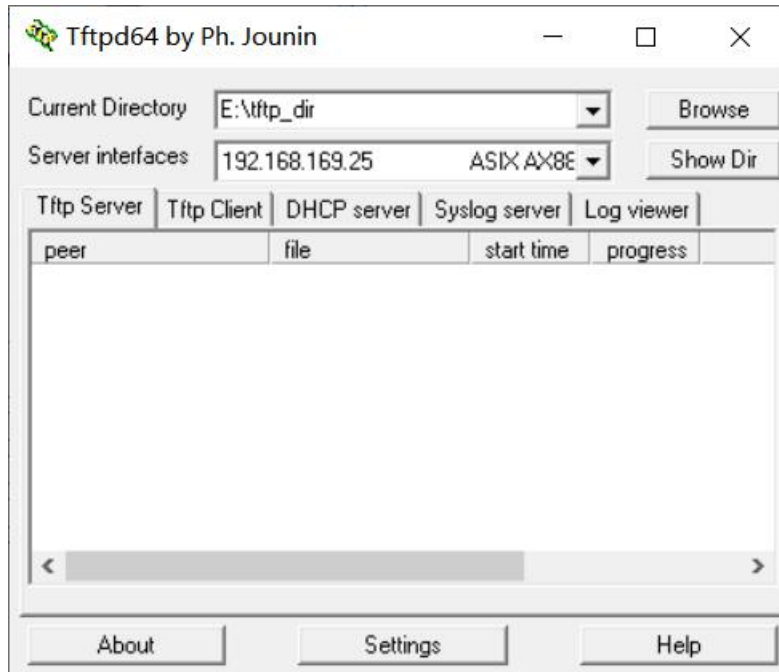


图 2.15 TFTP 服务器运行界面

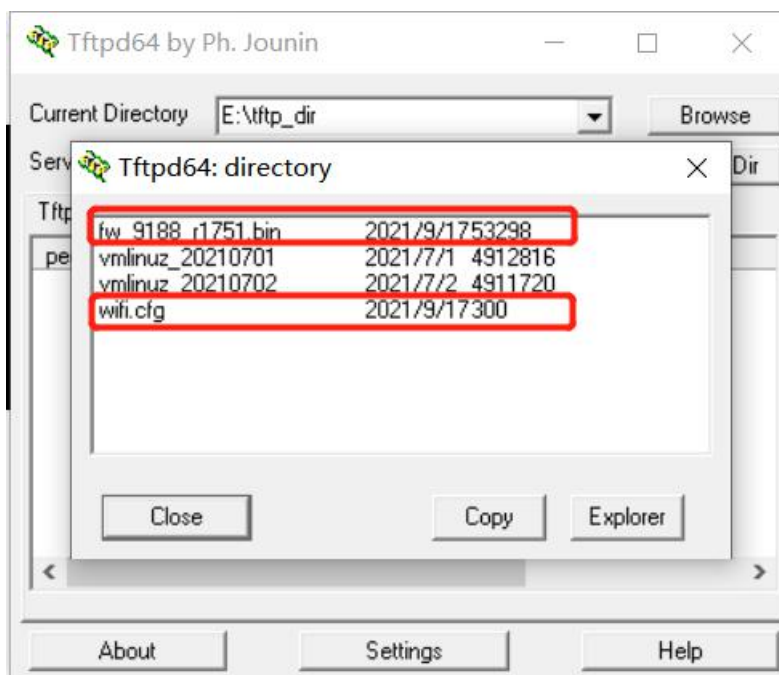


图 2.16 将固件和配置文件放入 TFTP 工作目录



```

msh />tftp
Usage: tftp [-s|-w|-r host] [path]
       tftp [-h|--stop]

       -s      : begin tftp server
       -w      : client write file to server
       -r      : client read file from server
       -p      : server port to listen on/connect to
       --stop  : stop tftp server

eg:
open server: tftp -s /
read file  : tftp -r 192.168.1.1 test.data
wriet file : tftp -w 192.168.1.1 test.data

```

图 2.17 rt-thread tftp 命令帮助

```

msh />ls
Directory /:
etc          <DIR>
lib          <DIR>
msh />cd etc
msh /etc>ls
Directory /etc:
msh /etc>tftp -r 192.168.169.25 wifi.cfg
file size:300
msh /etc>ls
Directory /etc:
wifi.cfg    300

```

图 2.18 下载配置文件 wifi.cfg 到目录/etc 中

```

msh /etc>cd ../
msh />cd /lib/f
msh />cd /lib/firmware
msh /lib/firmware>tftp -r 192.168.169.25 fw_9188_r1751.bin
file size:53298
msh /lib/firmware>ls
Directory /lib/firmware:
fw_9188_r1751.bin  53298

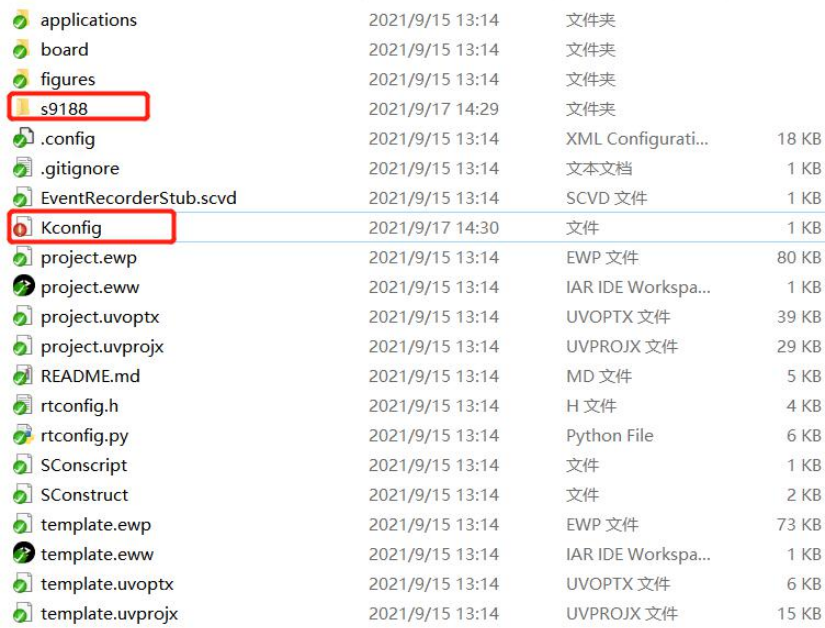
```

图 2.19 下载 WIFI 固件到目录/lib/firmware

固件和配置文件下载完成，下一步就可以配置驱动并进行编译下载了。

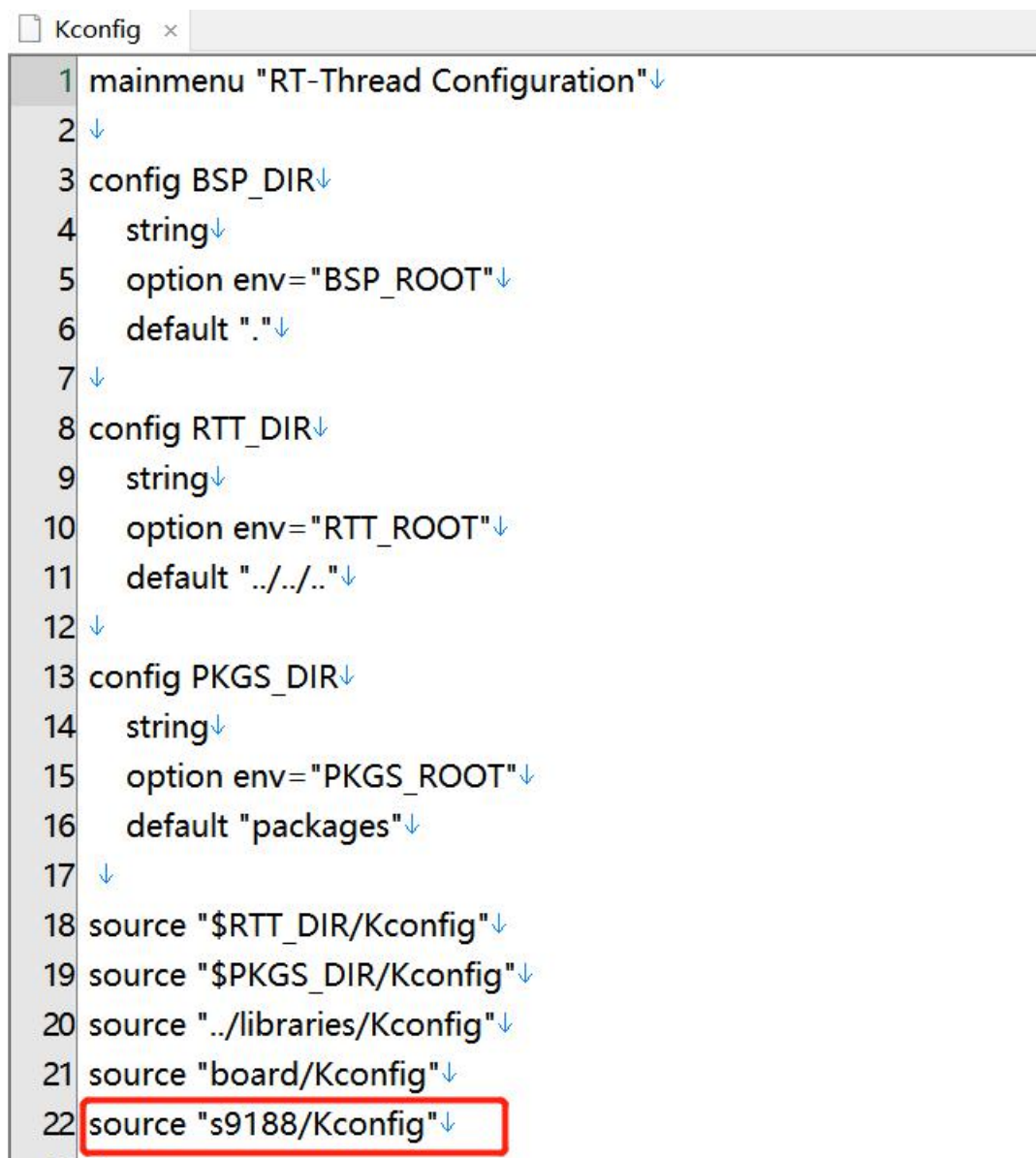
## 2.3 驱动配置编译

用户拿到 s9188 的驱动文件源码，如图 1.1 所示。将其完整目录拷贝到使用的 bsp 目录，如图 2.20 所示。修改 bsp 的 Kconfig 文件，增加 s9188 目录 Kconfig 的引用，如图 2.21 所示。



applications	2021/9/15 13:14	文件夹	
board	2021/9/15 13:14	文件夹	
figures	2021/9/15 13:14	文件夹	
s9188	2021/9/17 14:29	文件夹	
.config	2021/9/15 13:14	XML Configurati...	18 KB
.gitignore	2021/9/15 13:14	文本文档	1 KB
EventRecorderStub.scvd	2021/9/15 13:14	SCVD 文件	1 KB
Kconfig	2021/9/17 14:30	文件	1 KB
project.ewp	2021/9/15 13:14	EWP 文件	80 KB
project.eww	2021/9/15 13:14	IAR IDE Workspa...	1 KB
project.uvoptx	2021/9/15 13:14	UVOPTX 文件	39 KB
project.uvprojx	2021/9/15 13:14	UVPROJX 文件	29 KB
README.md	2021/9/15 13:14	MD 文件	5 KB
rtconfig.h	2021/9/15 13:14	H 文件	4 KB
rtconfig.py	2021/9/15 13:14	Python File	6 KB
SConscript	2021/9/15 13:14	文件	1 KB
SConstruct	2021/9/15 13:14	文件	2 KB
template.ewp	2021/9/15 13:14	EWP 文件	73 KB
template.eww	2021/9/15 13:14	IAR IDE Workspa...	1 KB
template.uvoptx	2021/9/15 13:14	UVOPTX 文件	6 KB
template.uvprojx	2021/9/15 13:14	UVPROJX 文件	15 KB

图 2.20 S9188 驱动拷贝到 bsp 目录



```
1 mainmenu "RT-Thread Configuration"↓
2 ↓
3 config BSP_DIR↓
4     string↓
5     option env="BSP_ROOT"↓
6     default "."↓
7     ↓
8 config RTT_DIR↓
9     string↓
10    option env="RTT_ROOT"↓
11    default "../.."↓
12    ↓
13 config PKGS_DIR↓
14    string↓
15    option env="PKGS_ROOT"↓
16    default "packages"↓
17    ↓
18 source "$RTT_DIR/Kconfig"↓
19 source "$PKGS_DIR/Kconfig"↓
20 source "../libraries/Kconfig"↓
21 source "board/Kconfig"↓
22 source "s9188/Kconfig"↓
```

图 2.21 bsp 目录 Kconfig 修改增加 s9188 配置引用

rt-thread 的 env 环境输入 menuconfig, 进入配置环境。使能 S9188 的驱动配置, 如图 2.22 所示。

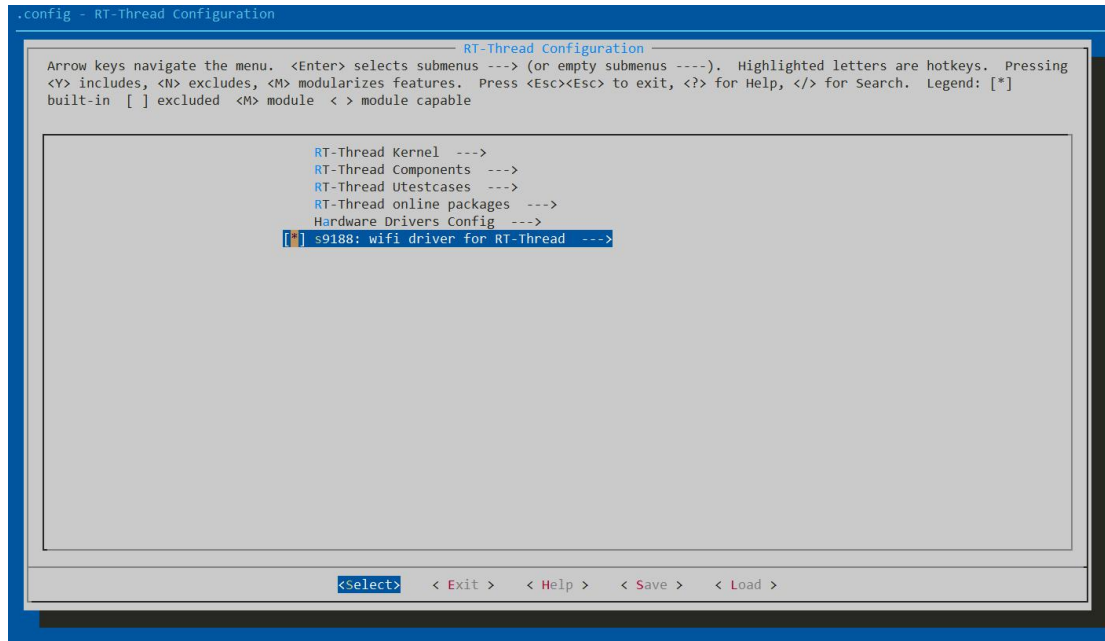


图 2.22 S9188 驱动使能

进入 S9188 驱动配置，选择 s9188 接口为 sdio；选择固件路径和配置文件 wifi.cfg 的路径；配置工作模式为 STA；然后配置收发缓冲区大小，如图 2.23 所示。

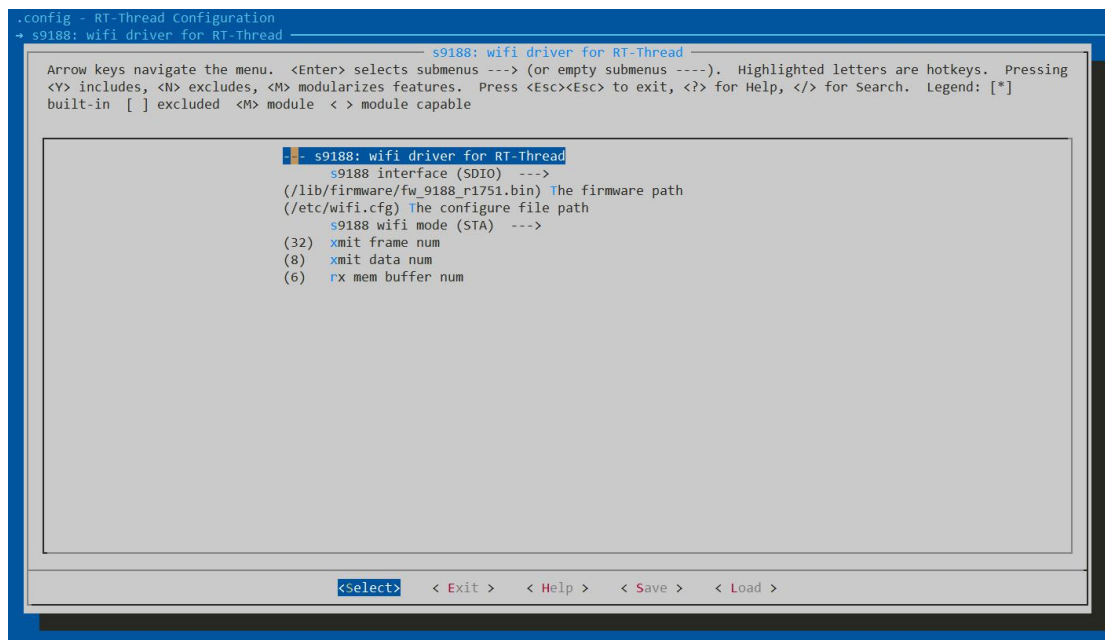


图 2.23 S9188 驱动具体配置

rt-thread 环境集成了 arm mcu 的 gcc 编译环境，直接在 env 下执行 scons 命令，开始编译，经过漫长的编译过程，编译完成，如图 2.24 所示。编译完成生成 bin 文件，可以下载到 MCU 中进行测试。

```
Administrator@GKCTUL009 E:\rt_thread\rt-thread\bsp\stm32\stm32h750-artpi-h750
$ scons
scons: Reading SConscript files ...
scons: done reading SConscript files.
scons: Building targets ...
scons: building associated VariantDir targets: build
LINK rt-thread.elf
arm-none-eabi-objcopy -O binary rt-thread.elf rtthread.bin
arm-none-eabi-size rt-thread.elf
   text    data     bss     dec     hex filename
 518168   3004   58052  579224  8d698 rt-thread.elf
scons: done building targets.
```

图 2.24 scons 编译

rt-thread 的 env 环境输入 `scons --target=iar`, 如图 2.25 所示, 生成 IAR IDE 的工程。如果命令报错, 请在 `rtconfig.py` 中正确设置 IAR 的安装路径。生成 IAR IDE 工程后, 可以用 IDE 打开工程, 进行编译、下载、调试。

```
$ scons --target=iar
scons: Reading SConscript files ...
scons: done reading SConscript files.
scons: Building targets ...
scons: building associated VariantDir targets: build
CC build\applications\main.o
CC build\board\board.o
CC build\board\CubeMX_Config\Core\Src\stm32h7xx_hal_msp.o
CC build\kernel\components\dfs\filesystems\devfs\devfs.o
CC build\kernel\components\dfs\filesystems\romfs\dfs_romfs.o
CC build\kernel\components\dfs\filesystems\romfs\romfs.o
CC build\kernel\components\dfs\src\dfs.o
CC build\kernel\components\dfs\src\dfs_file.o
CC build\kernel\components\dfs\src\dfs_fs.o
CC build\kernel\components\dfs\src\dfs_poll.o
```

图 2.25 scons 生产 iar 工程

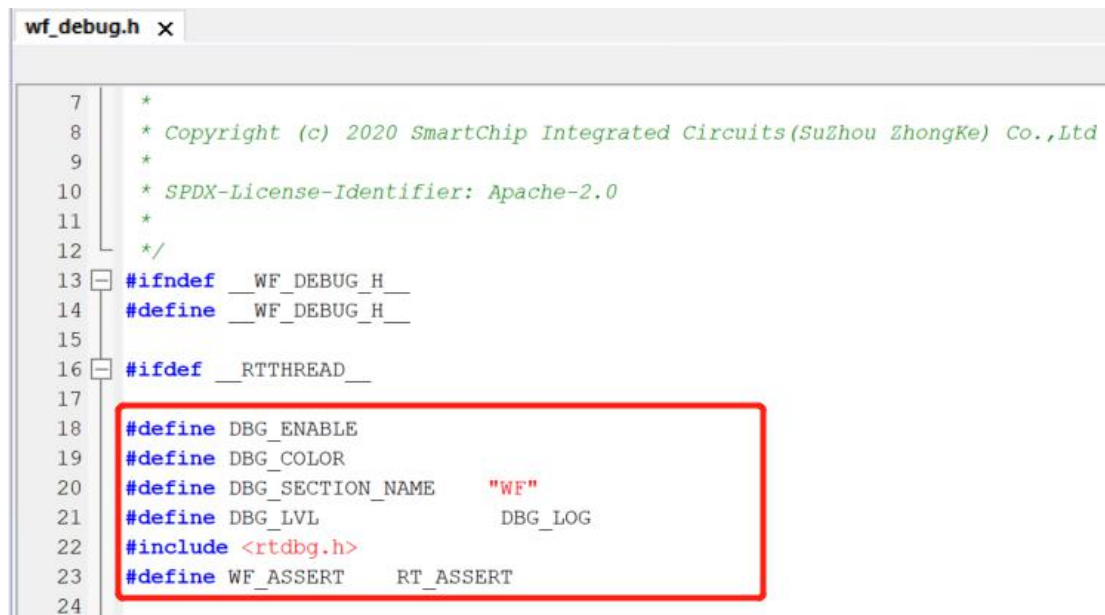
同理, rt-thread 的 env 环境输入 `scons --target=mdk`, 可以生成 mdk 的工程。

## 2.4 调试信息打印等级

调试信息打印等级的配置在文件 `os/wf_debug.h` 中, 如图 2.26 所示。本平台的调试信息打印完全符合 rt-thread 操作系统的打印信息标准。打印等级划分如程序清单 2.4 所示, 数值越大, 打印等级越高, 限制越小。

- 宏定义 `DBG_ENABLE`: 用于表征是否使能打印模块。不定义该宏, 则无任何打印。
- 宏定义 `DBG_COLOR`: 用于表征是否打印字体颜色。不定义该宏, 则打印全部黑色字体。
- `DBG_SECTION_NAME`: 本驱动打印的前缀, 用户可以修改。
- `DB_LVL`: 打印等级配置。例如设置打印等级为 `DBG_LOG`, 则打印所有的调试信息; 设置打印等级为 `DBG_INFO`, 则打印 INFO、WARNING、ERROR 的调试信息; 设置打印等级为 `DBG_ERROR`, 则只打印 ERROR 的调试信息。

关于打印配置的更详细的介绍, 可以参考 RT-Thread 文档中心相关介绍。



```
7  *
8  * Copyright (c) 2020 SmartChip Integrated Circuits (SuZhou ZhongKe) Co.,Ltd
9  *
10 * SPDX-License-Identifier: Apache-2.0
11 *
12 */
13 #ifndef __WF_DEBUG_H__
14 #define __WF_DEBUG_H__
15
16 #ifdef __RTTHREAD__
17
18 #define DBG_ENABLE
19 #define DBG_COLOR
20 #define DBG_SECTION_NAME    "WF"
21 #define DBG_LVL             DBG_LOG
22 #include <rtdbg.h>
23 #define WF_ASSERT          RT_ASSERT
24
```

图 2.26 调试等级配置

## 程序清单 2.4 调试等级

```
/* DEBUG level */
#define DBG_ERROR        0
#define DBG_WARNING      1
#define DBG_INFO         2
#define DBG_LOG          3
```



## 第3章 驱动加载、连接调试

### 3.1 STA 模式

用 IAR 或其它 IDE 打开工程,本文以 IAR EWARM 为例。连接仿真器,下载整个 rt-thread 编译的程序到 MCU,如图 3.1 所示。全速运行程序,即可看到程序运行起来。

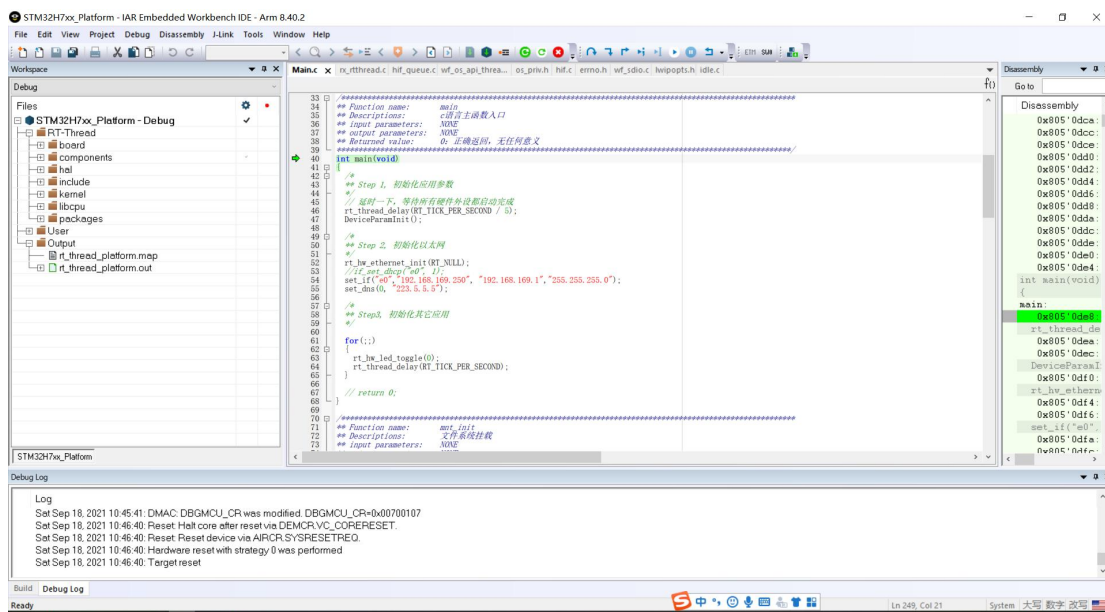


图 3.1 IDE 下载程序到 MCU

打开 SecureCRT,打开串口的终端,连接开发板串口和电脑,可以看到 rt-thread 运行 shell 输出信息,如图 3.2 所示。程序运行即已经加载 S9188 驱动,看到图 3.3 所示信息,表示 S9188 驱动已经正确加载并运行。

```

\ \ / /
- RT -      Thread Operating System
/ \ \      4.0.3 build Sep 16 2021
2006 - 2020 Copyright by rt-thread team
[I/I2C] I2C bus [i2c1] registered
Init lwip tcp/ip stack!
lwIP-2.2.0 initialized!
Init ethernet driver frame!
[E/DFS] mount fs[elm] on /mnt failed.

flash0 mount to / ok!
sdmmc clock set to 0Hz
msh />sdmmc clock set to 375000Hz
sdmmc clock set to 375000Hz
sdmmc clock set to 375000Hz
sdmmc clock set to 375000Hz
sdmmc clock set to 375000Hz
sdmmc clock set to 3000000Hz
sdmmc clock set to 3000000Hz
[I/WF] Found s9188 wlan card.

[I/WF] Class      = 7
[I/WF] Vendor ID: 0x02e7
[I/WF] Device ID: 0x9086
[I/WF] Function#: 1
[I/WF] Driver Ver:1.0.0
[D/WF]

<SCI WIFI DRV INIT>

[D/WF] *****HIF INIT*****
[D/WF] prase richv200 firmware!
[D/WF] FW0 Ver: 1.5.7.1, size:33088Bytes
[D/WF] FW1 Ver: 8.9.3.1, size:20176Bytes
[I/WF] [sdio_ctl_init]
[I/WF] hif_node: 2401ffbc
[I/WF] wf_sdio_init start
[I/WF] sdio_id=0

```

图 3.2 程序运行

```

[D/WF] [HW_CFG] channel_plan: 0xc
[D/WF] [HW_CFG] ba_func: 1
[D/WF] [LOCAL_CFG] work_mode: 2
[D/WF] [LOCAL_CFG] channel: 1
[D/WF] [LOCAL_CFG] bw: 0
[D/WF] [LOCAL_CFG] ssid: SCI9083
[D/WF] [NIC] prv_hardware_init - exit
[I/WF] [wf_wlan_mgmt_init:1141]ndev_id:0
[D/WF] [NIC] nic_init - end
[I/WF] efuse_macaddr:b4:04:18:00:a8:a8
[I/WLAN.dev] wlan init success
[D/WF] F:wf_wlan_mode L:165 mode:1
[I/WF] wlan dev open
[I/WF] [NIC] nic_enable :b4:04:18:00:a8:a8
[I/WLAN.lwip] eth device init ok name:w0
[I/WF] ppg_num:231,hpg_num:12,mgp_num:2,lpq_num:2,epg_num:0
[I/WF] HIGH(fifo_1,fifo_2,fifo_4) MID(fifi_5) LOW(fifo_6)
[I/WF] SdioTxOQTFreeSpace:32
[I/WF] [wf_wlan_probe] end

```

图 3.3 S9188 驱动正确加载

执行命令：list\_device，可以看到注册到 rt-thread 设备驱动框架的所有设备，如图 3.4 所示。设备 wlan0 是 wifi 管理的接口；设备 w0 是注册到 lwip 的网卡设备。设备 wlan0 和 w0 是一一映射的对应关系，两者之间可以互相索引。



执行命令:ifconfig, 可以查看所有的网络设备, 设备 w0 就是 S9188 的网卡, 如图 3.5 所示。

```
msh />list_device
device          type          ref count
-----
w0              Network Interface 0
wlan0          Network Interface 1
e0              Network Interface 0
flash0         MTD Device       1
iospi10        SPI Device       0
flash          Block Device     0
i2c1           I2C Bus          0
ttyS0          Character Device 2
rtc            RTC              0
iospi1         SPI Bus          0
```

图 3.4 查看通用设备信息

```
ifconfig
network interface: w0
MTU: 1500
MAC: b4 04 18 00 a8 a8
FLAGS: UP LINK_DOWN ETHARP IGMP
ip address: 0.0.0.0
gw address: 0.0.0.0
net mask : 0.0.0.0
Not support IPv6

network interface: e0 (Default)
MTU: 1500
MAC: 00 60 6e 46 00 1b
FLAGS: UP LINK_UP ETHARP IGMP
ip address: 192.168.169.250
gw address: 192.168.169.1
net mask : 255.255.255.0
Not support IPv6

network interface: lo
MTU: 0
MAC:
FLAGS: UP LINK_UP
ip address: 127.0.0.1
gw address: 127.0.0.1
net mask : 255.0.0.0
Not support IPv6

dns server #0: 223.5.5.5
dns server #1: 0.0.0.0
msh />
```

图 3.5 查看网卡信息

```
msh />wifi
wifi
wifi help
wifi scan [SSID]
wifi join [SSID] [PASSWORD]
wifi ap SSID [PASSWORD]
wifi disc
wifi ap_stop
wifi status
wifi smartconfig
```

图 3.6 wifi 命令帮助

执行命令：wifi 或 wifi help，可以查看 wifi 管理操作 shell 命令的帮助信息，执行结果如图 3.6 所示。

执行命令：wifi scan，启动 wifi 扫描，执行结果如图 3.7 所示。

```
msh />wifi scan
[I/WF] [m1me_core_thrd:1255][0]scanning...
[I/WF] [core_scan_thrd:185][0]scan...
[I/WF] reg_200=00e7020c, reg_204=00e7020c
[I/WF] [scan_setting:140][0]Disbale BSSID Filter
[I/WF] [wf_scan_thrd:508][0]scanning...
[I/WF] [wf_scan_thrd:602][0]scan done pass time: 2520ms
[I/WF] [rx_frame_handle:677]scan queue refresh
[I/WF] [scan_setting_recover:202][0]Enable BSSID Filter
[E/WLAN.mgmt] F:rt_wlan_scan_with_info L:1802 scan wait timeout!
      SSID                MAC                security      rssi  chn  Mbps
-----
ZTE_TEST1                50:0f:f5:10:2f:42  WPA_AES_PSK   68    1   144
SQ_TEST                  f8:8c:21:a2:26:9f  OPEN          72    5   300
WF_TANG3                  e4:f3:f5:58:45:7e  WPA2_AES_PSK  60    7   144
WF_MI_4Pro                50:d2:f5:eb:b8:35  WPA2_MIXED_PSK 68    8   144
WF_HUAWEI_test            18:3c:b7:f5:54:c0  WPA2_AES_PSK  60    9   144
WF_TPLINK666              58:41:20:64:9d:ff  OPEN          68   11   144
TP_6080                   34:f7:16:78:ed:9f  WPA2_AES_PSK  100   13   300
JOY-TEST-AP              00:0f:02:71:67:74  OPEN          100   13   150
WF_HONOR                  82:8f:2b:a1:da:67  OPEN          60   13   144
msh />
```

图 3.7 wifi 扫描

执行命令：wifi join SSID(开放式网络)或命令 wifi join SSID PASSWORD(加密网络)，启动 wifi 网络连接，执行结果如图 3.8。只要通过 menuconfig 配置使能了 dhcp 客户端功能，则连接通过就通过 DHCP 自动获取 IP 地址。WIFI 连接成功如图 3.9 所示。

```
msh />wifi join JOY-TEST-AP
[I/WF] [m1me_core_thrd:1255][0]scanning...
[I/WF] [core_scan_thrd:185][0]scan...
[I/WF] reg_200=00e7020c, reg_204=00e7020c
[I/WF] [scan_setting:140][0]Disbale BSSID Filter
[I/WF] [wf_scan_thrd:508][0]scanning...
[I/WF] [wf_scan_thrd:602][0]scan done pass time: 2520ms
[I/WF] [rx_frame_handle:677]scan queue refresh
[I/WF] [scan_setting_recover:202][0]Enable BSSID Filter
[E/WLAN.mgmt] F:rt_wlan_scan_with_info L:1802 scan wait timeout!
[D/WF] sta_info->security:0
[I/WF] [m1me_core_thrd:1268][0]start conneting to bss: "JOY-TEST-AP" "00:0f:02:71:67:74"
[I/WF] [m1me_core_thrd:1275][0]search bss...
[I/WF] [core_conn_scan_thrd:310][0]wait probe response...
[I/WF] reg_200=00e7020c, reg_204=00e7020c
[I/WF] [wf_scan_thrd:508][0]scanning...
[I/WF] [set_cur_network:2][W/WF] [wf_m1me_conn_scan_rsp:1682][0]msg new fail error code: -3
3rsp:1682][0]msg
[I/WF] [core_conn_scan_thrd:354][0]probe response ok
[I/WF] [wf_scan_thrd:602][0]scan done pass time: 130ms
```

图 3.8 wifi 连接

```
[D/WF] [mcu_set_rate_bitmap] Rate Bitmap:0xff015
[W/WF] The netif link up
[I/WLAN.mgmt] wifi connect success ssid:JOY-TEST-AP
[I/WF] [m1me_core_thrd:1324][0]connect success
[I/WF] [m1me_core_thrd:1325][0]connection maintain
[D/WF] <DHCP> Send
msh />[I/WF] [core_conn_maintain_msg_thrd:884][0]ba response
[I/WF] [wf_action_frame_ba_to_issue:496]tid:6 dialog:1 ba_para_set:0x101a timeout:0 status:0
[D/WF] <DHCP> Send
[I/WF] [core_conn_maintain_msg_thrd:884][0]ba response
[I/WF] [wf_action_frame_ba_to_issue:496]tid:0 dialog:1 ba_para_set:0x1002 timeout:0 status:0
[I/WF] [core_conn_maintain_msg_thrd:884][0]ba response
[I/WF] [wf_action_frame_ba_to_issue:496]tid:6 dialog:2 ba_para_set:0x101a timeout:0 status:0
[D/WF] <DHCP> Send
[I/WLAN.lwip] Got IP address : 192.168.88.153
```

图 3.9 wifi 连接成功

图 3.9 可以看到 DHCP 获取到的 IP 地址是 192.168.88.153，执行命令：192.168.88.1，测试下和路由器的网络连接，执行结果如图 3.10 所示。

```
msh />ping 192.168.88.1
60 bytes from 192.168.88.1 icmp_seq=0 ttl=64 time=5 ms
60 bytes from 192.168.88.1 icmp_seq=1 ttl=64 time=0 ms
60 bytes from 192.168.88.1 icmp_seq=2 ttl=64 time=0 ms
60 bytes from 192.168.88.1 icmp_seq=3 ttl=64 time=0 ms
Ping 4 times, 0 times failed
msh />
```

图 3.10 测试网络

执行命令：wifi status，可以查看 wifi 网络连接情况，如图 3.11 所示。

执行命令：wifi disc，可以断开当前 WIFI 网络连接，如图 3.11 所示。

```
msh />wifi status
Wi-Fi STA Info
SSID : JOY-TEST-AP
MAC Addr: 00:0f:02:71:67:74
Channel: 13
DataRate: 150Mbps
RSSI: 0
wifi ap not start!
Auto Connect status:Disable!
msh />wifi disc
[I/WF] [core_conn_maintain_msg_thrd:865][0]deassoc
[I/WF] [mlme_core_thrd:1327][0]connection break
[W/WF] The netif link down
[I/WF] [mlme_conn_cleanup:1052][0]
[DHCP] dhcpcd_stop: w0
[I/WLAN.mgmt] disconnect success!
ms[I/WF] reg_200=00e7020c, reg_204=00e7020c
hm

msh />wifi status
wifi disconnected!
wifi ap not start!
Auto Connect status:Disable!
```

图 3.11 WIFI 连接状态查看

## 3.2 AP 模式

通过 rt-thread 的 env 环境执行命令 menuconfig 重新配置 s9188 驱动为 AP 模式，配置如图 3.12 所示。

执行命令 scons --target=iar 或者 scons --target=mdk，生成 IDE 工程并重新编译、链接、下载到 MCU 中运行。

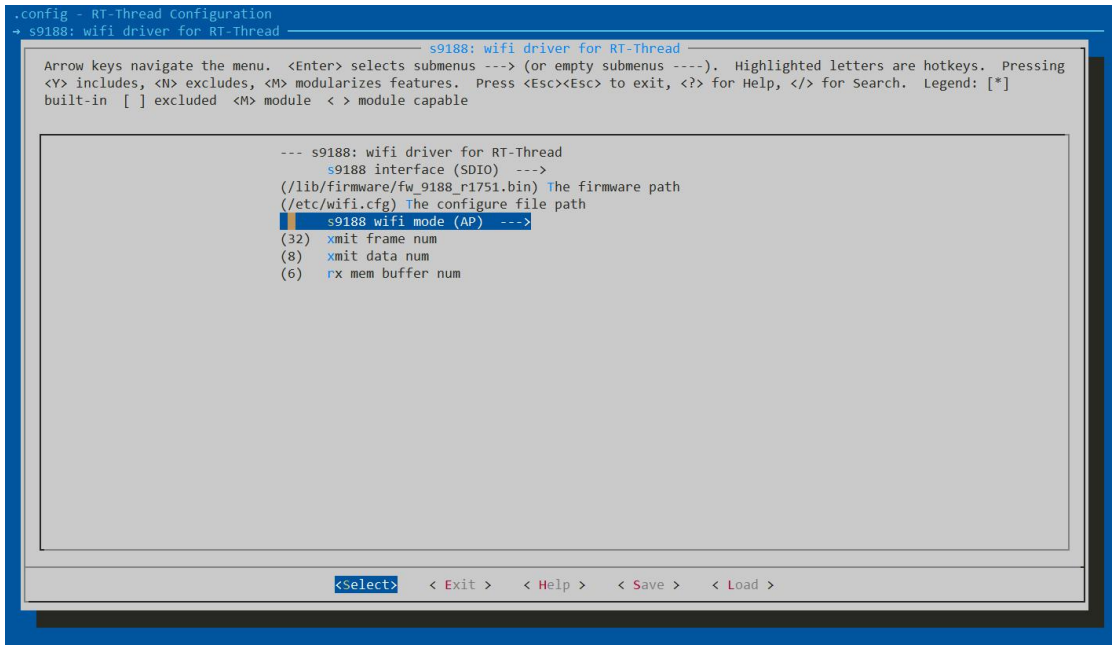


图 3.12 配置驱动运行在 AP 模式

执行命令：list\_device，可以看到注册到 rt-thread 设备驱动框架的所有设备，如图 3.13 所示。设备 wlan0 是 wifi 管理的接口；设备 w0 是注册到 lwip 的网卡设备。设备 wlan0 和 w0 是一一映射的对应关系，两者之间可以互相索引。

```
msh />list_device
```

device	type	ref count
w0	Network Interface	0
wlan0	Network Interface	1
e0	Network Interface	0
flash0	MTD Device	1
iospi0	SPI Device	0
flash	Block Device	0
i2c1	I2C Bus	0
ttyS0	Character Device	2
rtc	RTC	0
iospi1	SPI Bus	0

图 3.13 查看通用设备信息

执行命令:ifconfig，可以查看所有的网络设备，设备 w0 就是 S9188 的网卡，如图 3.14 所示。



```

msh />ifconfig
network interface: w0
MTU: 1500
MAC: b4 04 18 00 67 67
FLAGS: UP LINK_UP ETHARP IGMP
ip address: 192.168.125.1
gw address: 0.0.0.0
net mask : 255.255.255.0
Not support IPv6

network interface: e0 (Default)
MTU: 1500
MAC: 00 60 6e 46 00 1b
FLAGS: UP LINK_UP ETHARP IGMP
ip address: 192.168.169.250
gw address: 192.168.169.1
net mask : 255.255.255.0
Not support IPv6

network interface: lo
MTU: 0
MAC:
FLAGS: UP LINK_UP
ip address: 127.0.0.1
gw address: 127.0.0.1
net mask : 255.0.0.0
Not support IPv6

dns server #0: 223.5.5.5
dns server #1: 0.0.0.0

```

图 3.14 查看网卡信息

S9188 驱动工作在 AP 模式时，当前版本驱动只支持开放式网络。

执行命令：wifi ap SSID，启动 ap 模式，运行结果如图 3.15 所示。

```

msh />wifi ap MY_S9188
[D/WF] ssid_val:MY_S9188 ssid_len:8
[D/WF] [wf_ap_set_beacon:789]
[D/WF] [wf_ap_set_beacon:808]bssid: b4:04:18:00:67:67
[D/WF] [wf_ap_set_beacon:814]beacon interval: (100), capability information: (0x0021)
[D/WF] [wf_ap_set_beacon:838]ssid: MY_S9188
[D/WF] [wf_ap_set_beacon:918]data rate(Mbps): 1, 2, 5, 11, 6, 9, 12, 18, 0, 0, 0, 0, 0, 0, 0, 0
[D/WF] [wf_ap_set_beacon:872]channel: 6
[D/WF] [wf_ap_set_beacon:918]data rate(Mbps): 1, 2, 5, 11, 6, 9, 12, 18, 24, 36, 48, 54, 0, 0, 0, 0
[D/WF] [ap_update_reg:708]
[D/WF] [set AP role] wf_mcu_set_ap_mode
[D/WF] [wf_mcu_set_basic_rate] br_cfg=0xffff
[I/WF] [wf_nic_beacon_xmit,1531] buffer_id:0, pg_num:1
[I/WF] [wf_sdio_write_net_data_agg] sec:0,addr:4
[W/WF] The netif link up
[DHCP] dhcpd_start: w0
[DHCP] ip_start: [192.168.125.2]
[DHCP] ip_start: [192.168.125.254]
[D/WF] [wf_ap_work_start:1316]
[D/WF] [ap_msg_queue_init:170]
[D/WF] [ap_poll:491]
[D/WF] new broadcast wdn!!!!!!!
[I/WLAN.mgmt] start ap success!
msh />[D/WF] [mcu_set_rate_bitmap] MacID:0 RaID:8 BW:0 SGI:0 ra_mask: 0x0
[D/WF] [mcu_set_rate_bitmap] Rate Bitmap:0x0

```

图 3.15 启动 AP

在 windows 或其它桌面操作系统环境运行 WIFI 扫描，可以看到运行起来的 S9188 AP，如图 3.16 所示。

在 windows 或其它桌面操作系统环境中选中热点，可以启动连接。连接成功后，可以自动获取到 IP 地址。

S9188 AP 的 IP 地址为 192.168.125.1。在 windows 环境打开 cmd 窗口，执行命令：ping 192.168.125.1，测试网络连接，如图 3.17 所示。



图 3.16 电脑连接 AP

```
C:\Users\Administrator>ping 192.168.125.1

正在 Ping 192.168.125.1 具有 32 字节的数据:
来自 192.168.125.1 的回复: 字节=32 时间=2ms TTL=255
来自 192.168.125.1 的回复: 字节=32 时间=2ms TTL=255
来自 192.168.125.1 的回复: 字节=32 时间=5ms TTL=255
来自 192.168.125.1 的回复: 字节=32 时间=5ms TTL=255

192.168.125.1 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 2ms, 最长 = 5ms, 平均 = 3ms
```

图 3.17 测试网络连接

执行命令:wifi ap\_stop,停止 AP。所有连接到 AP 的 STA 都断开连接。运行结果如图 3.18 所示。

```
msh />wifi ap_stop
[D/WF] [wf_ap_deauth_all_sta:1401]
[D/WF] [set AP role] wf_mcu_disable_ap_mode
[W/WF] The netif link down
[DHCP] dhcpd_stop: w0
[I/WLAN.mgmt] ap stop success!
```

图 3.18 停止 AP