目 录



**S9188驱动RT-Thread操作系统平台使用指南**

[版本修改 1](#_Toc17792)

[前言 2](#_Toc20395)

[第1章 驱动简介 3](#_Toc14443)

[1.1 驱动接口 3](#_Toc12052)

[1.2 驱动功能 3](#_Toc9290)

[1.3 驱动目录结构 3](#_Toc13637)

[1.4 资源需求 3](#_Toc31729)

[第2章 驱动配置 5](#_Toc6345)

[2.1 组件配置 5](#_Toc5785)

[2.2 固件配置 10](#_Toc27470)

[2.2.1 Romfs使用 10](#_Toc24882)

[2.2.2 其它文件系统使用 12](#_Toc18481)

[2.3 驱动配置编译 15](#_Toc2850)

[2.4 调试信息打印等级 19](#_Toc9400)

[第3章 驱动加载、连接调试 21](#_Toc4647)

[3.1 STA模式 21](#_Toc14065)

[3.2 AP模式 25](#_Toc28225)

# 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。

# 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。

## 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。

## 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。

## 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。



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

## 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。

# 驱动配置

## 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。

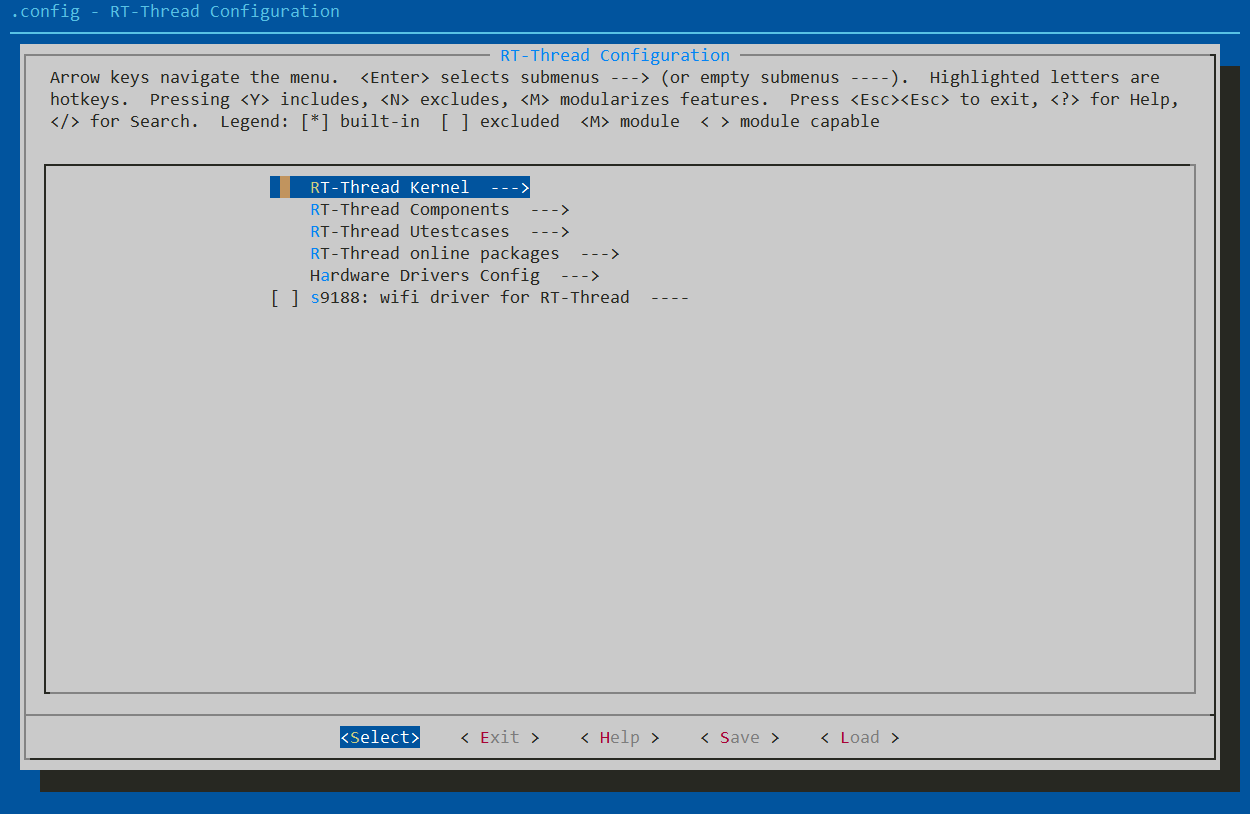


图 2.1 env运行界面

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。

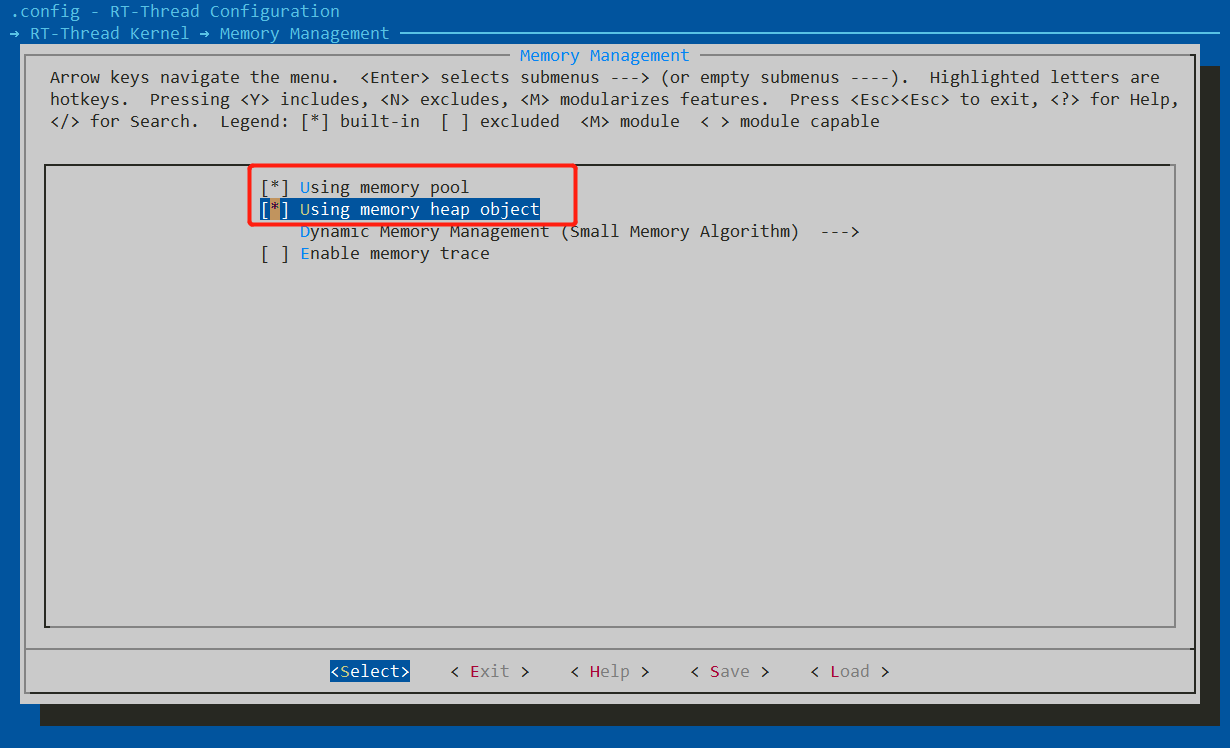


图 2.2 Heap配置

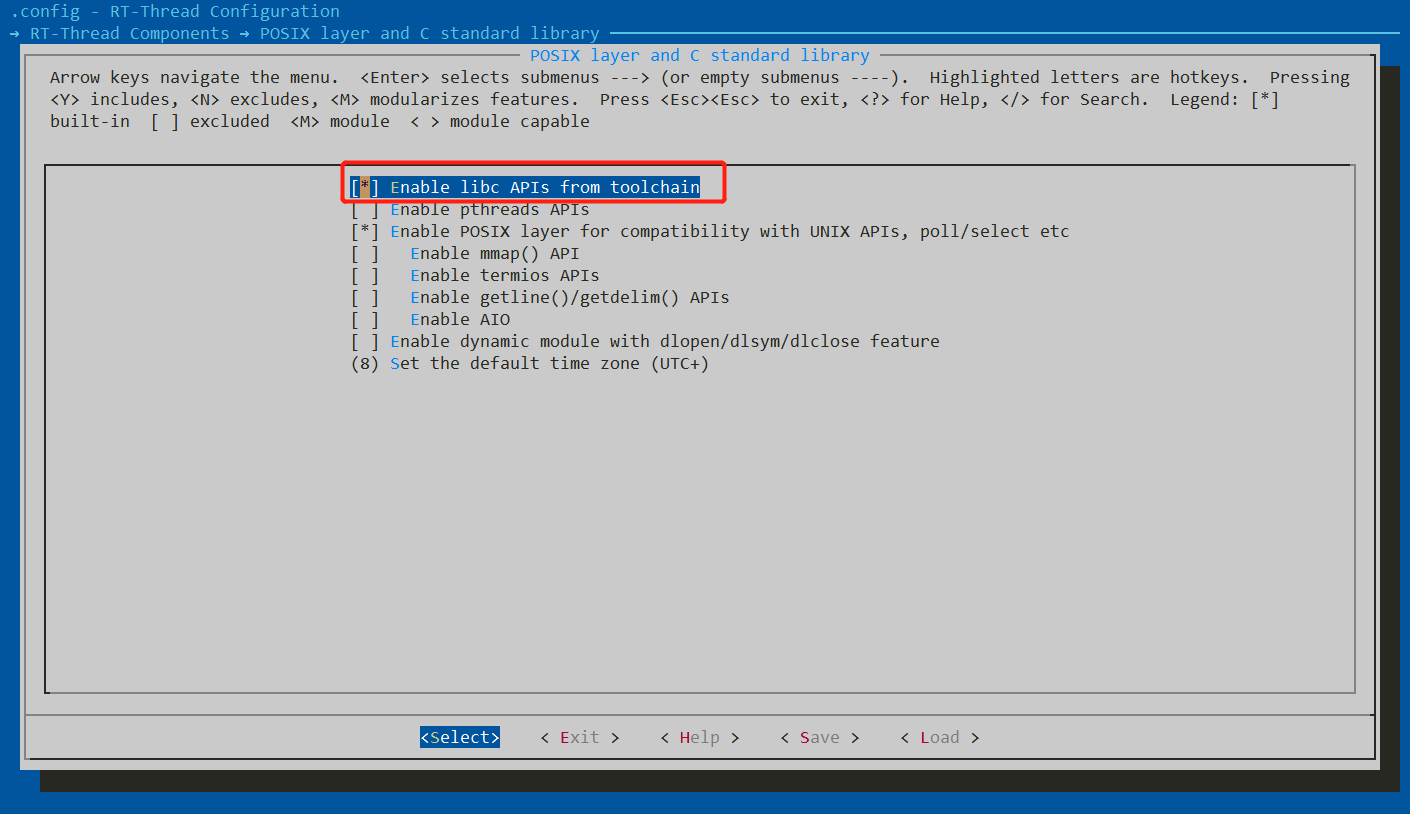


图 2.3 libc库api使能

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。

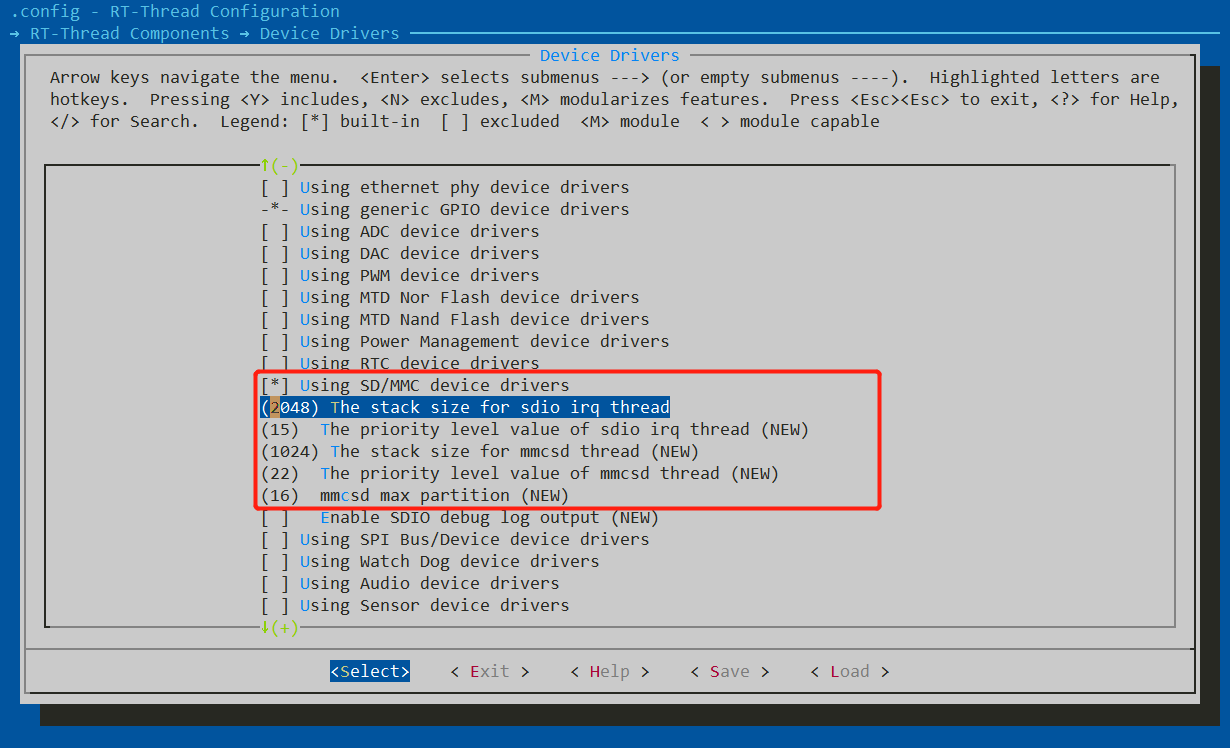
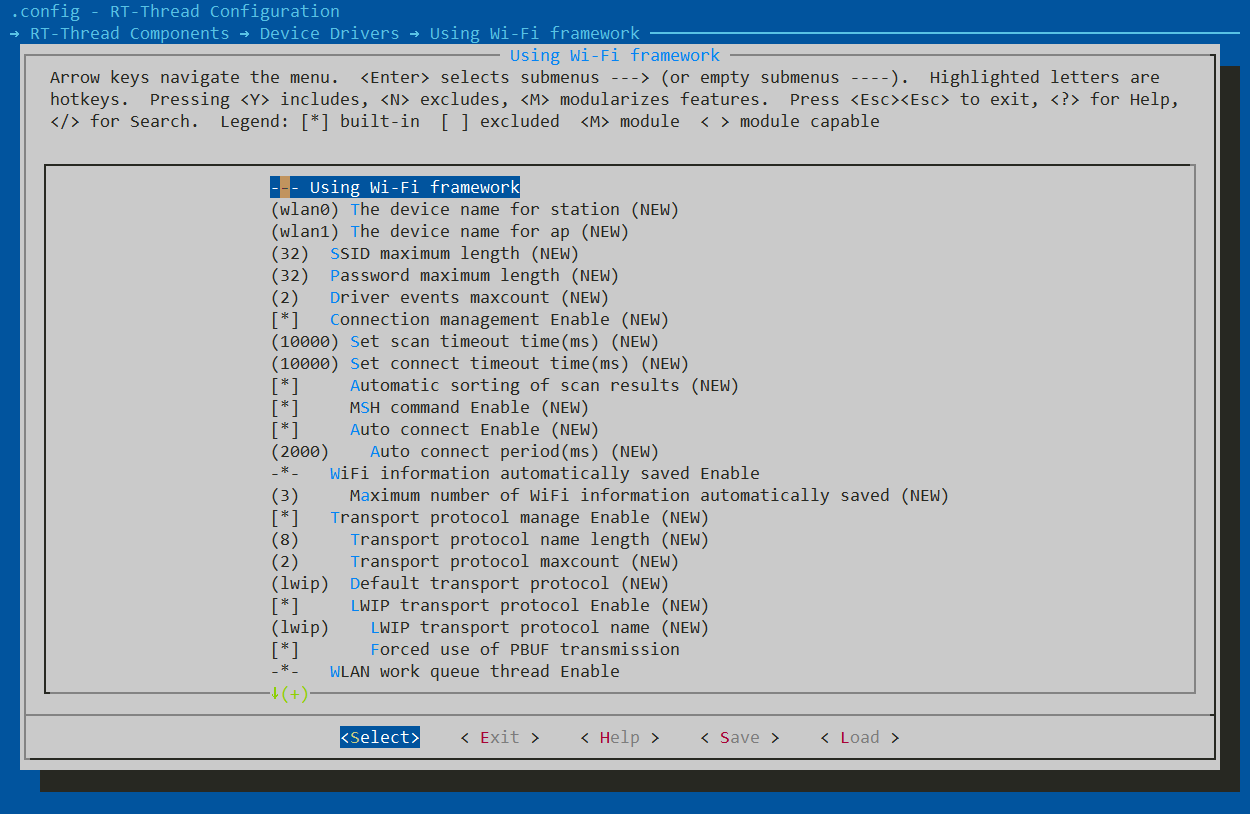


图 2.4 sdio框架配置

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。



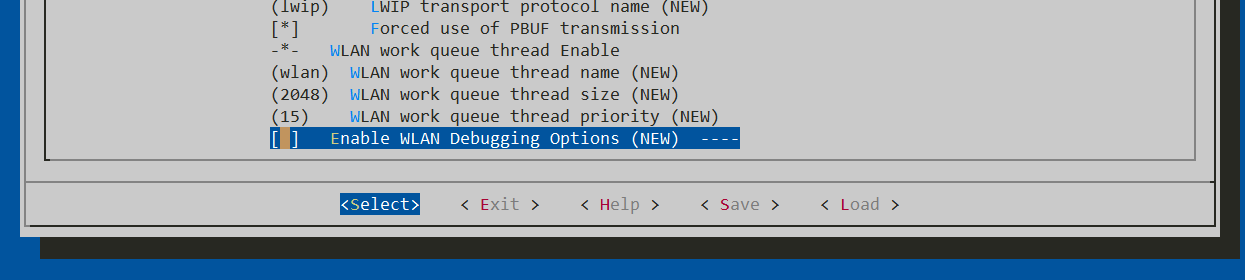


图 2.5 wlan框架配置

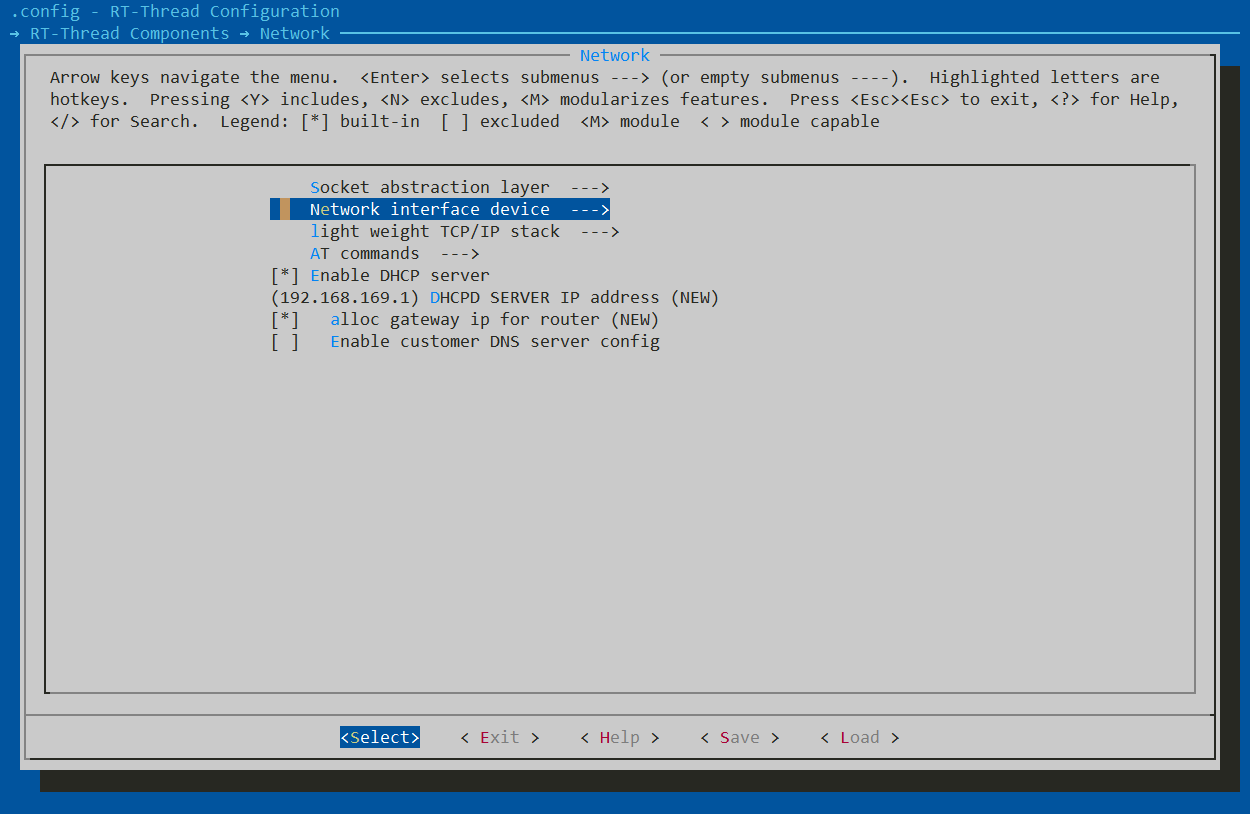


图 2.6 dhcp server使能配置

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。**

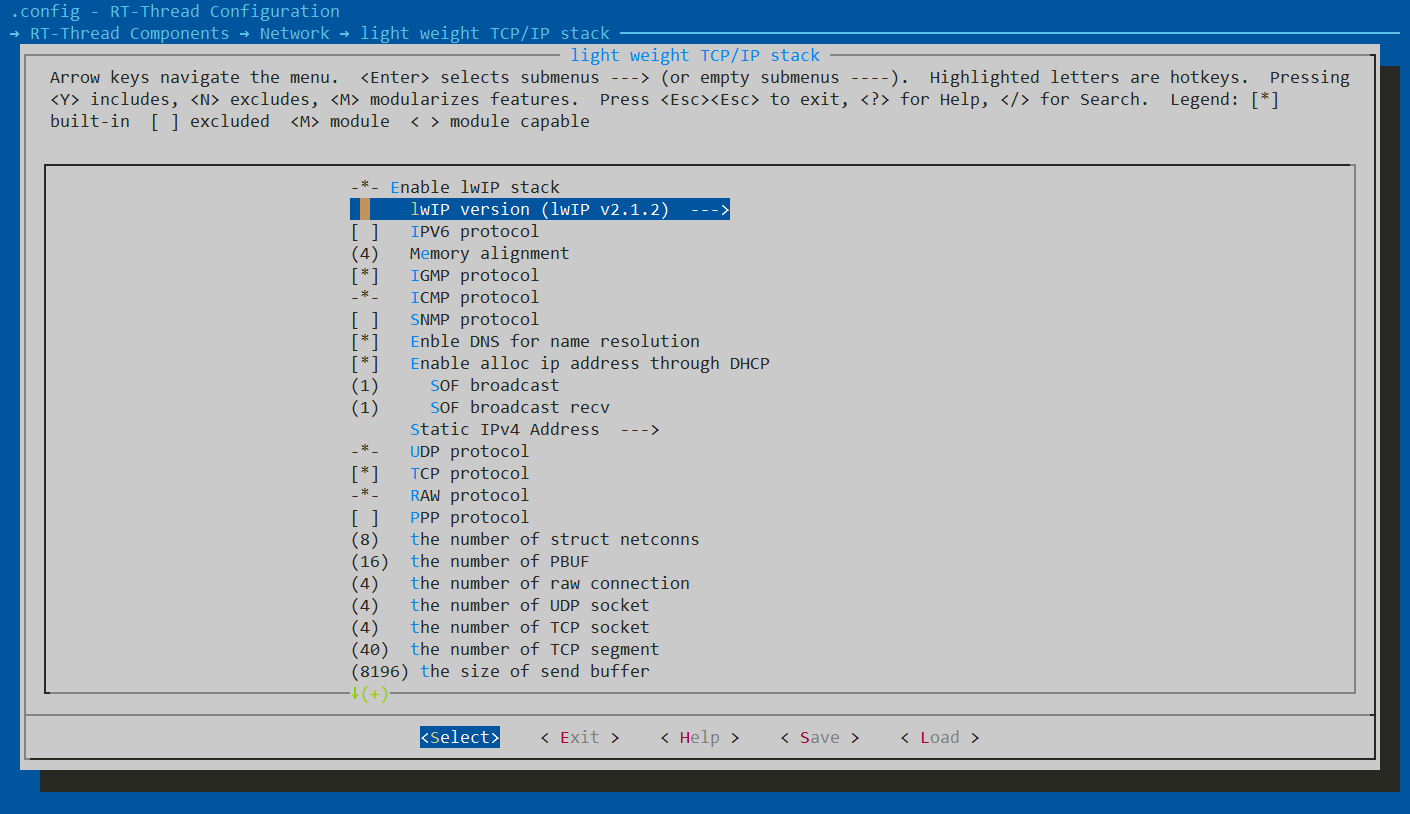


图 2.7 lwip协议栈基本配置

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。

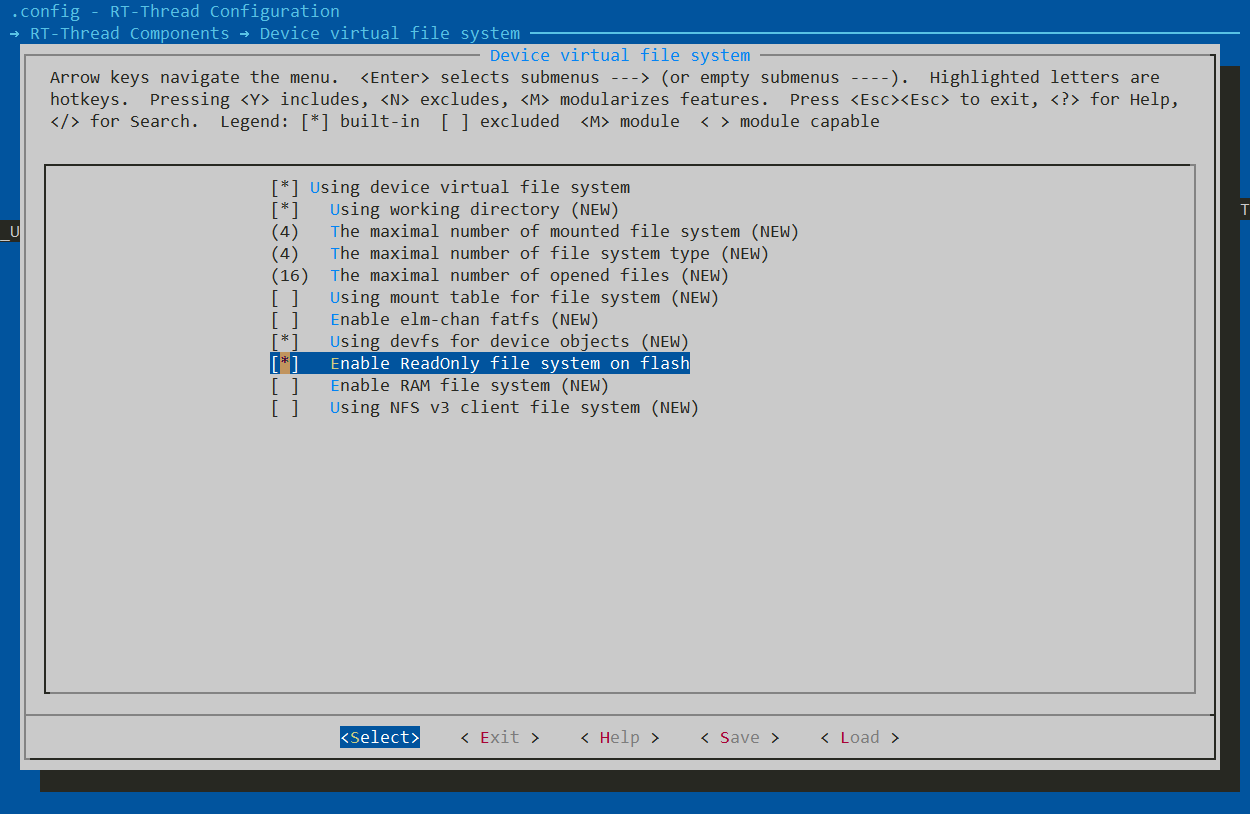


图 2.8 dfs使能配置

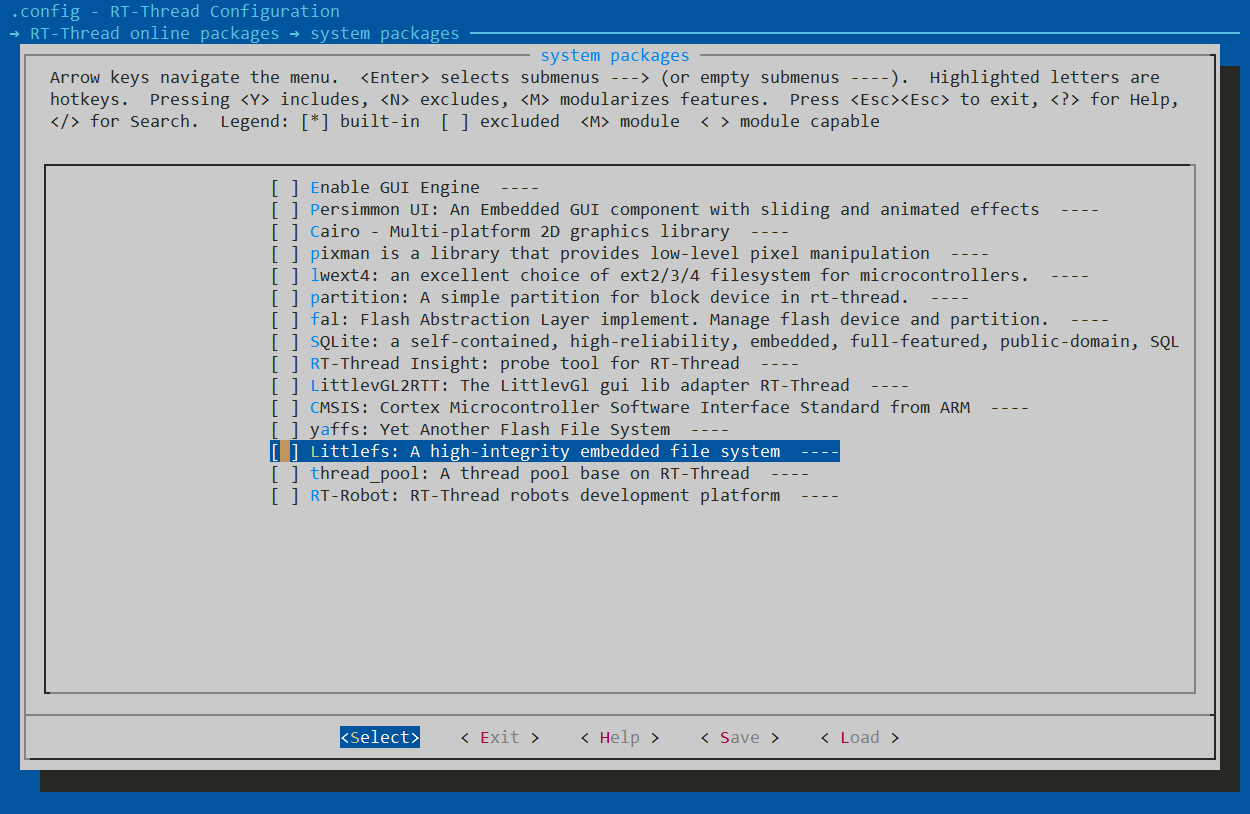


图 2.9 在线软件包littlefs选择

## 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。

### 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。

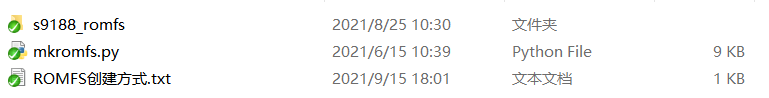


图 2.10 romfs使用说明目录

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 initializated!\n");

} else {

rt\_kprintf("ROM file system initializate 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。

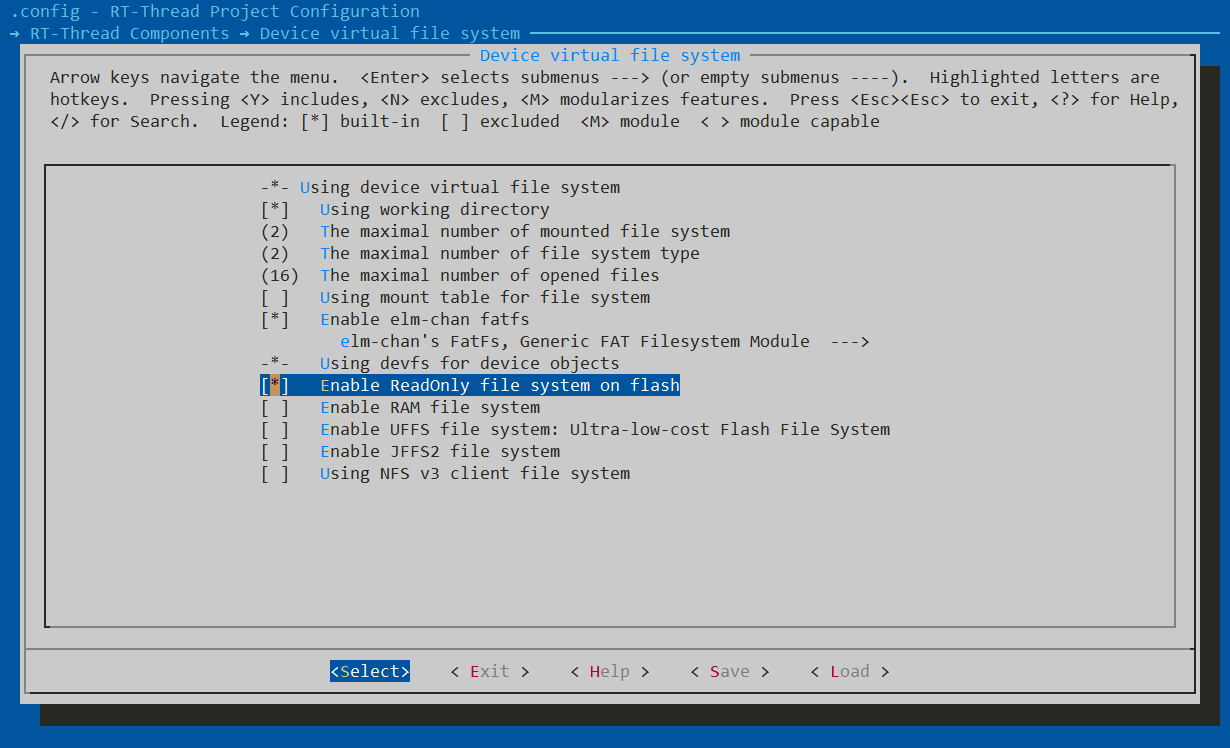


图 2.11 配置使能romfs文件系统

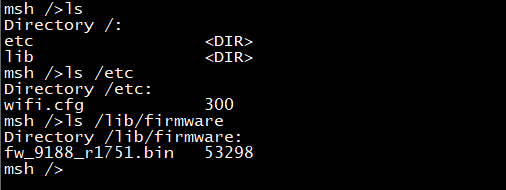


图 2.12 ROM文件系统查看

### 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。

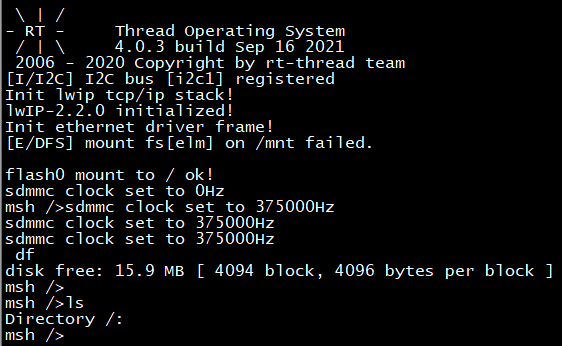


图 2.13 spi flash挂载littlefs文件系统查看



图 2.14 构建目录结构

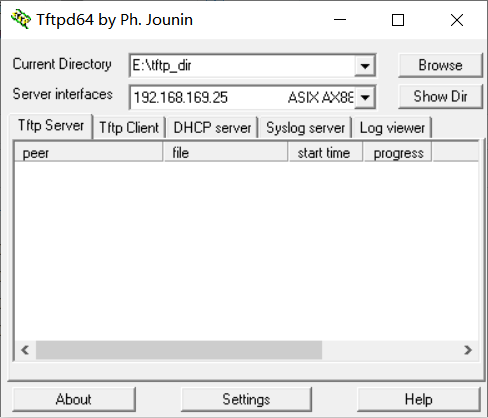


图 2.15 TFTP服务器运行界面

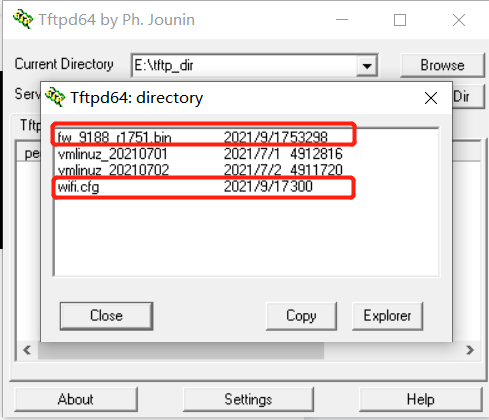


图 2.16 将固件和配置文件放入TFTP工作目录

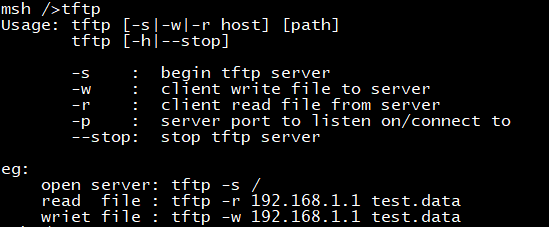


图 2.17 rt-thread tftp命令帮助

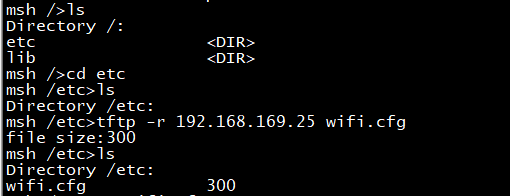


图 2.18 下载配置文件wifi.cfg到目录/etc中

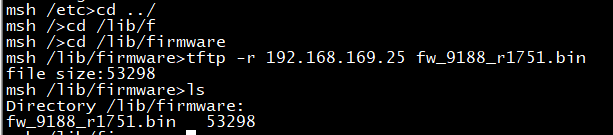


图 2.19 下载WIFI固件到目录/lib/firmware

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。

## 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。

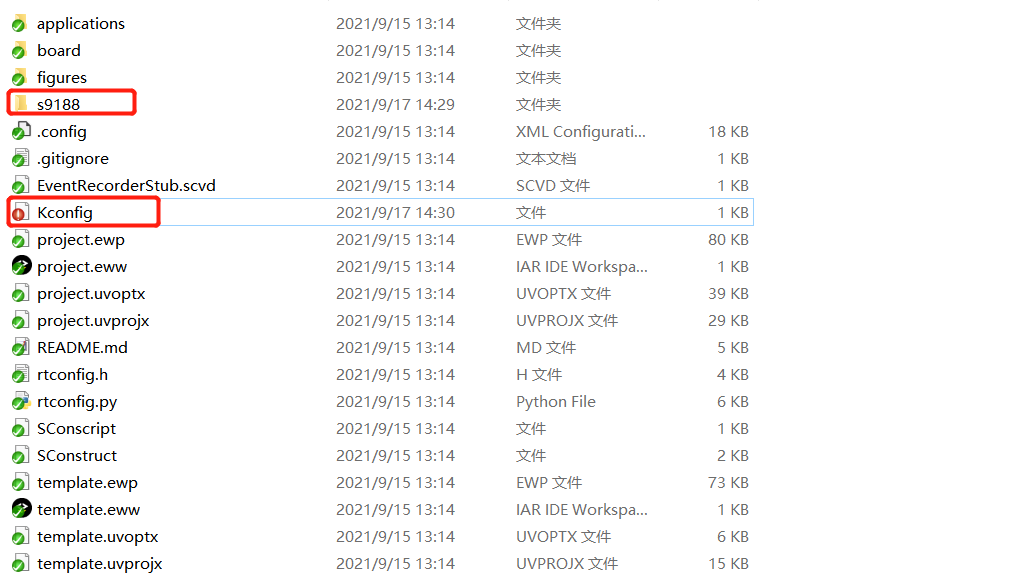


图 2.20 S9188驱动拷贝到bsp目录

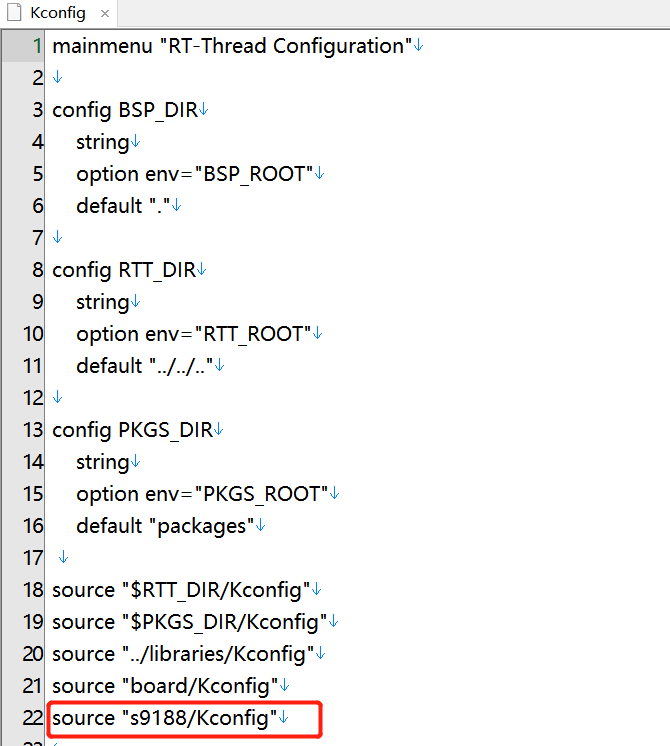


图 2.21 bsp目录Kconfig修改增加s9188配置引用

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。

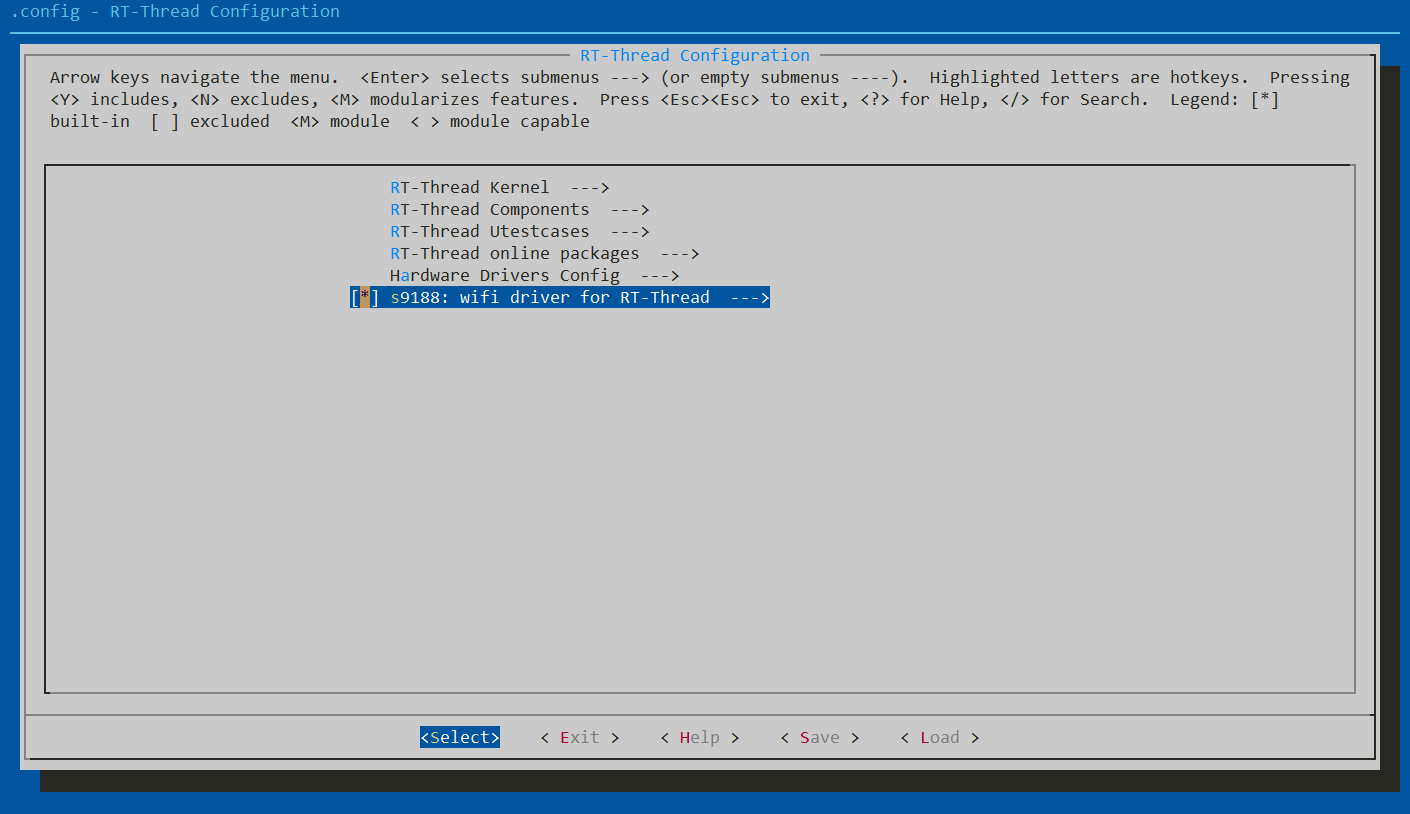


图 2.22 S9188驱动使能

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。

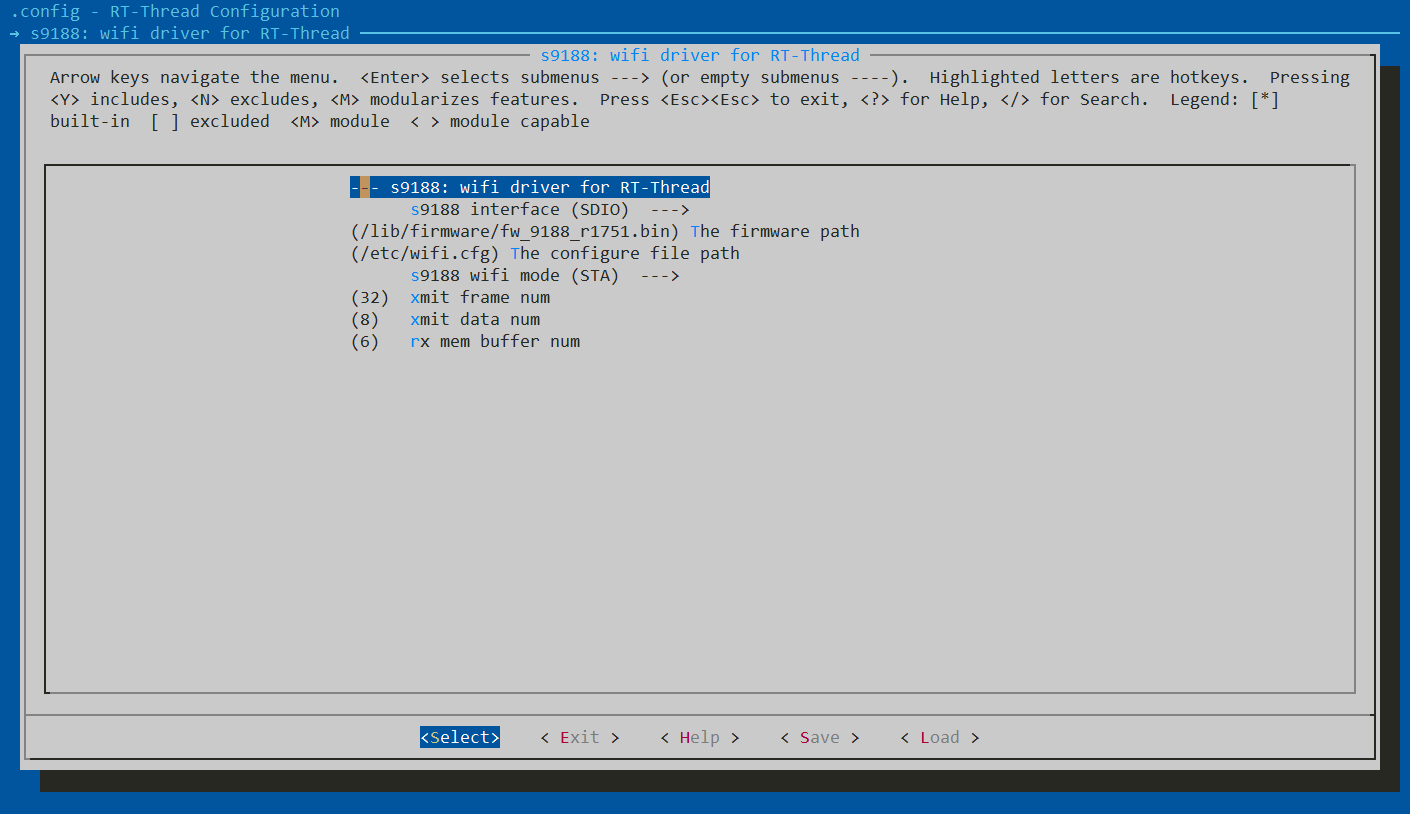


图 2.23 S9188驱动具体配置

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。

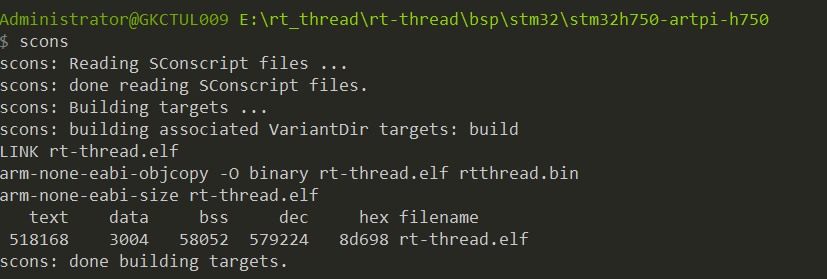


图 2.24 scons编译

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。

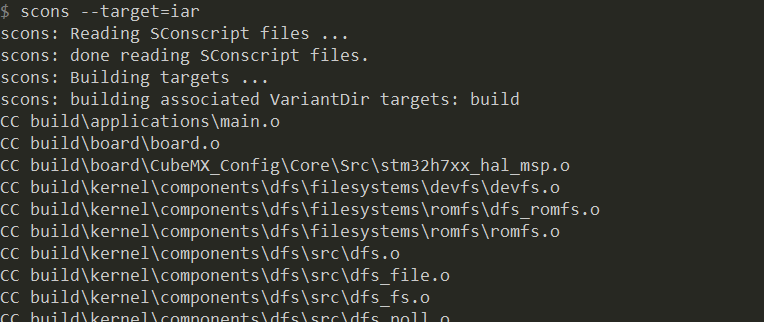


图 2.25 scons生产iar工程

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

## 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INGO、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。

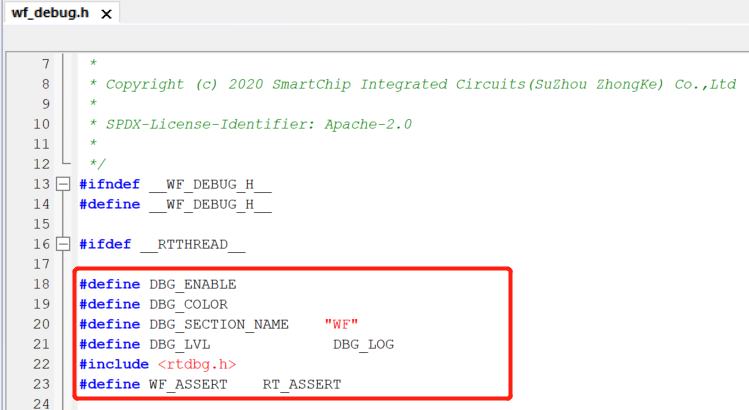


图 2.26 调试等级配置

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

# 驱动加载、连接调试

## 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。

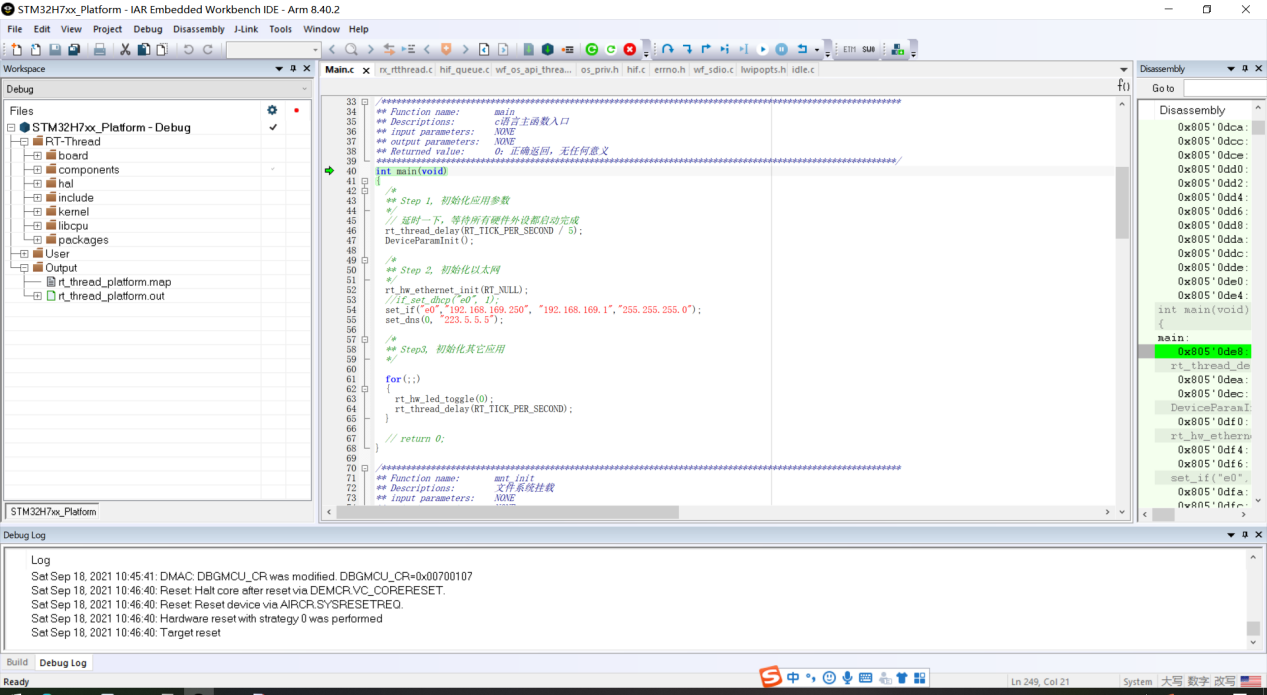


图 3.1 IDE下载程序到MCU

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。

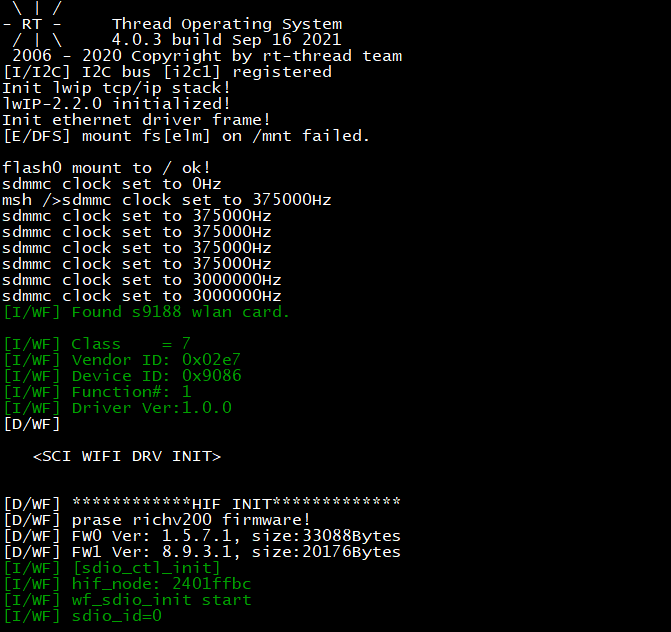


图 3.2 程序运行

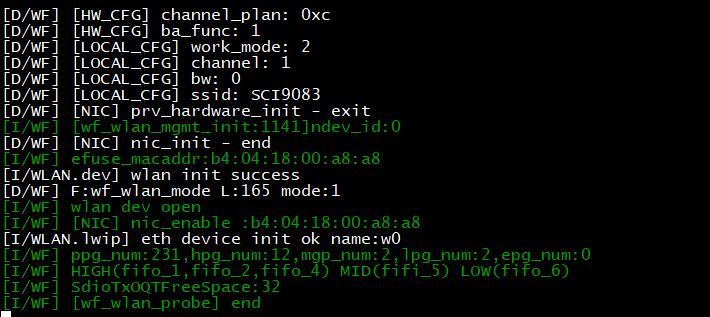


图 3.3 S9188驱动正确加载

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。

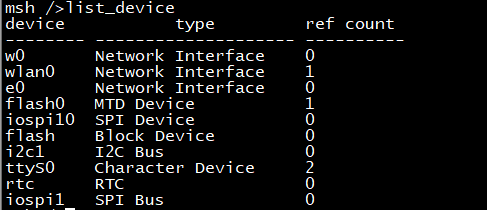


图 3.4 查看通用设备信息

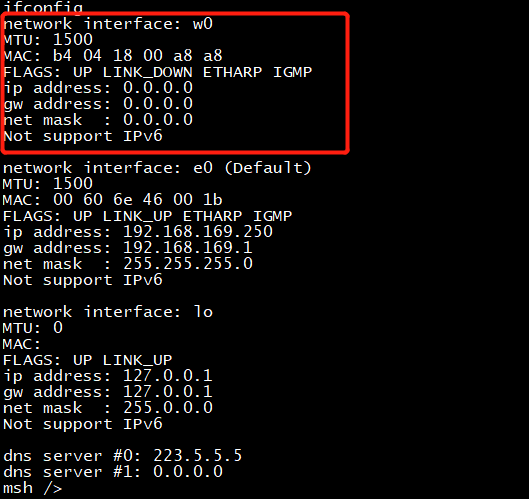


图 3.5 查看网卡信息

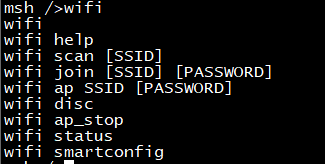


图 3.6 wifi命令帮助

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。

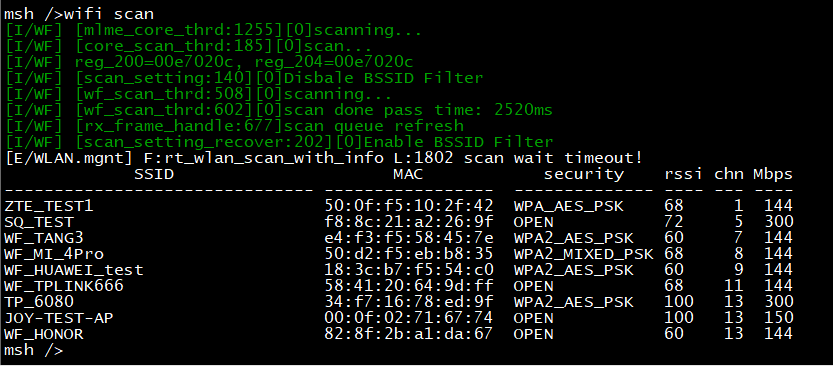


图 3.7 wifi扫描

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。

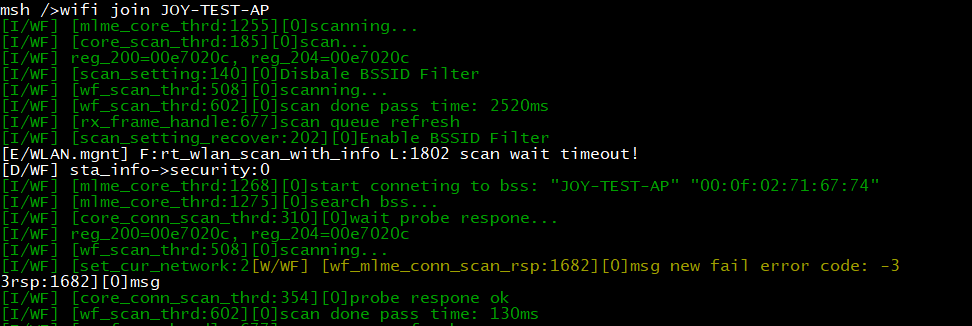


图 3.8 wifi连接

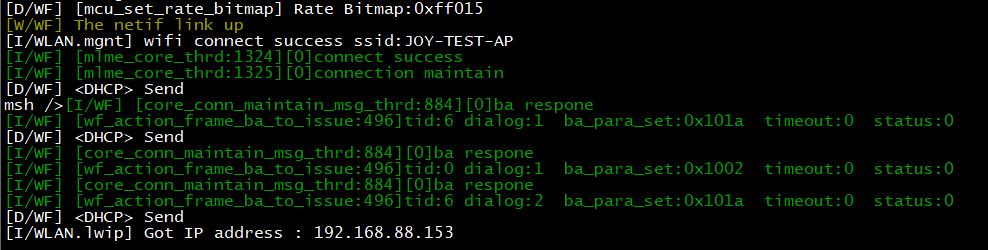


图 3.9 wifi连接成功

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。

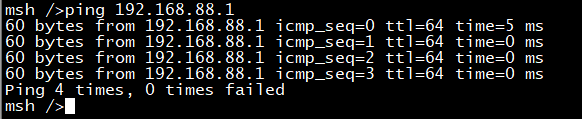


图 3.10 测试网络

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。

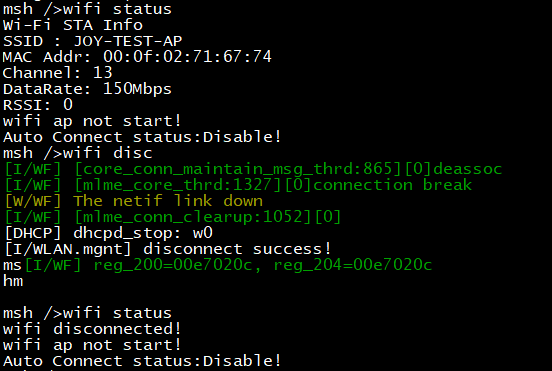


图 3.11 WIFI连接状态查看

## 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。

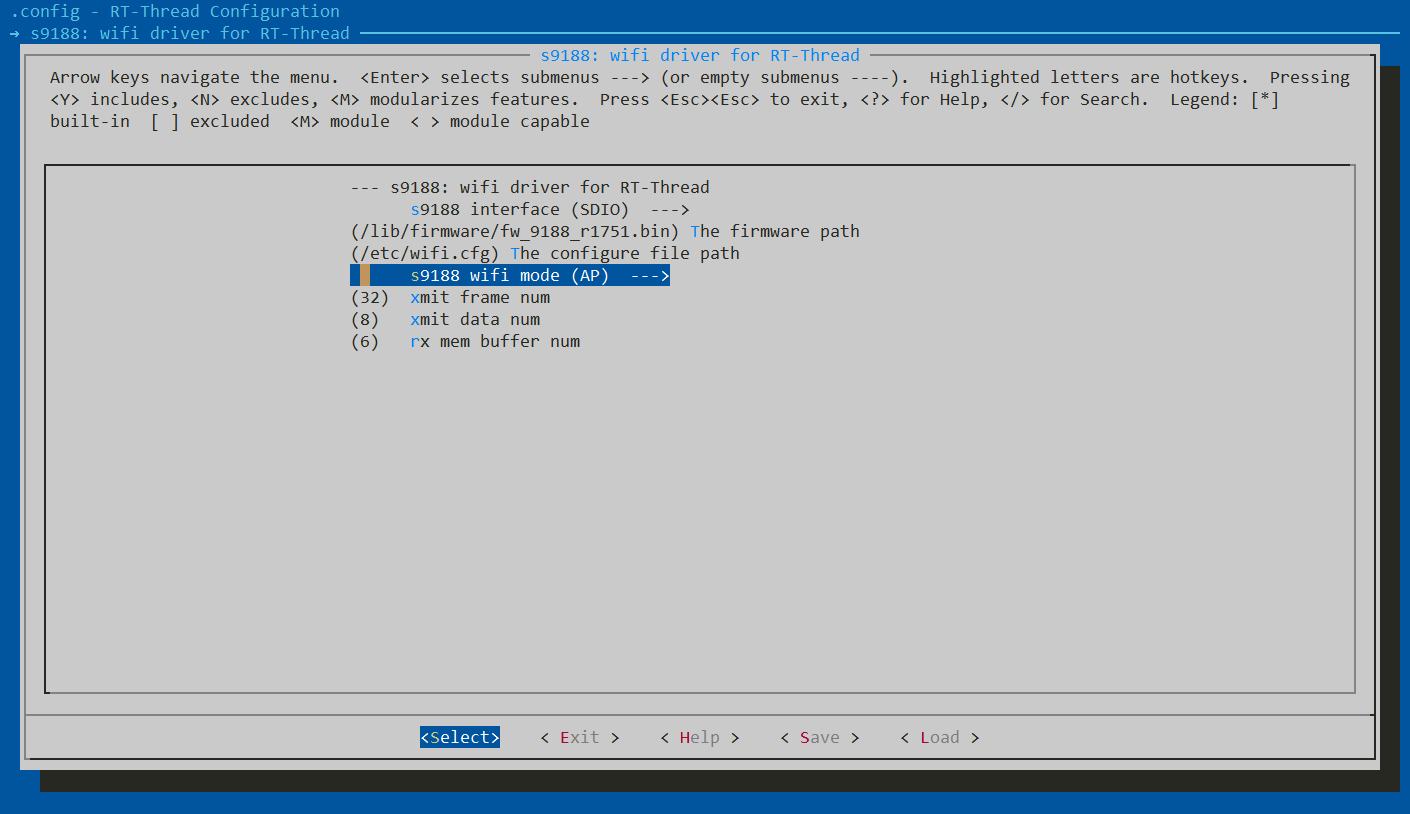


图 3.12 配置驱动运行在AP模式

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。

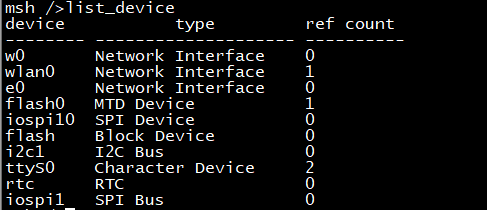


图 3.13 查看通用设备信息

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。

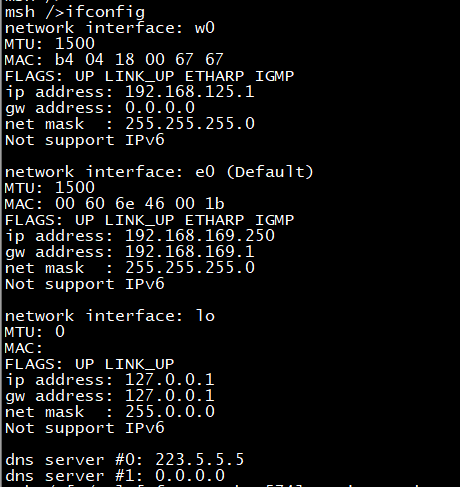


图 3.14 查看网卡信息

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。

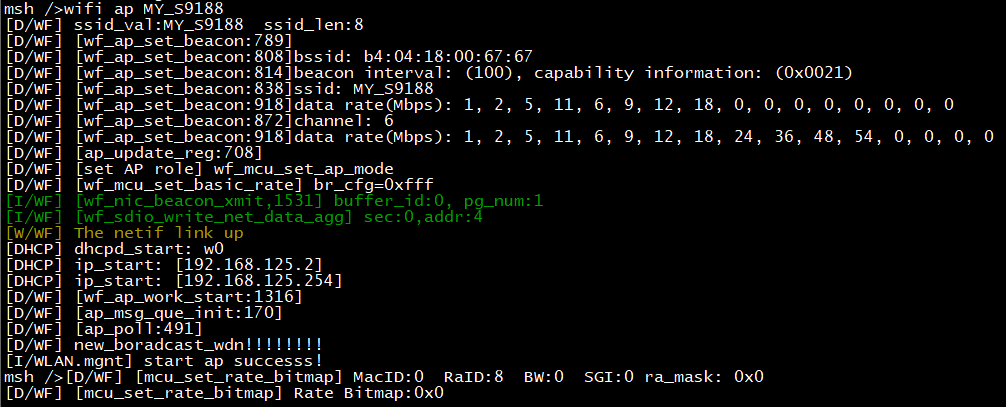


图 3.15 启动AP

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。

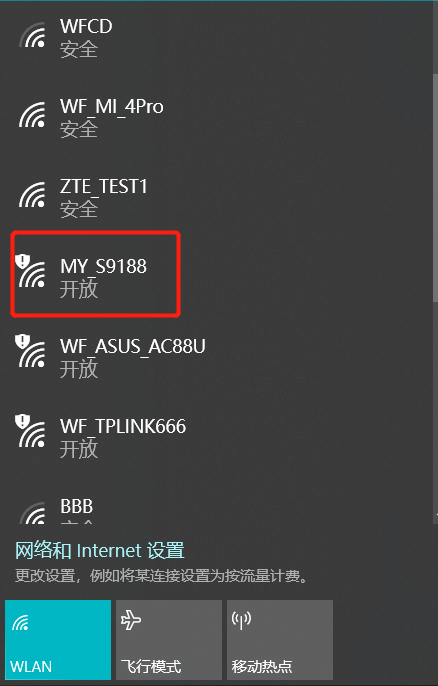


图 3.16 电脑连接AP

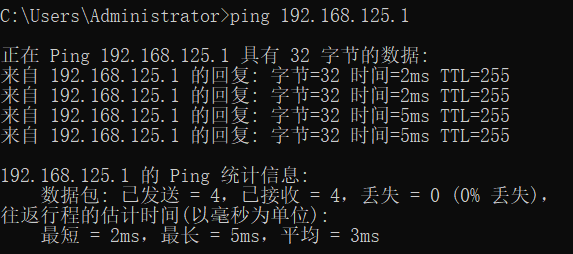


图 3.17 测试网络连接

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。

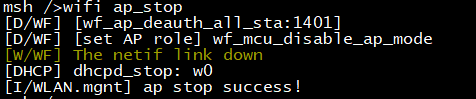


图 3.18 停止AP