

# **Anylinkin® ALK8266WIFI®模组**

## **SPI 接口高速通信使用与集成**

# **主机驱动 API 函数**

# 目 录

1 前言.....	4
1.1 简介 .....	4
1.2 术语约定 .....	5
1.3 通过书签方便阅读和快速检索定位 .....	5
1.4 参考文档 .....	6
1.5 联系我们 .....	6
2 函数的友好性声明和注释 .....	7
2.1 友好性声明 .....	7
2.2 友好性注释 .....	7
3 M8266HostIf.c 主机接口源码函数说明 .....	9
3.1 M8266HostIf_GPIO_CS_RESET_Init .....	9
3.2 M8266HostIf_SPI_Init .....	9
3.3 M8266HostIf_USART_Init .....	10
3.4 M8266HostIf_Init .....	10
3.5 M8266HostIf_Set_nRESET_Pin .....	11
3.6 M8266HostIf_Set_SPI_nCS_Pin .....	12
3.7 M8266HostIf_SPI_ReadWriteByte .....	12
3.8 M8266HostIf_delay_us .....	13
4 M8266WIFIDrv 驱动库函数说(SPI 接口部分) .....	14
4.1 接口初始化和 SPI 连通性逻辑与性能测试 API .....	15
4.1.1 M8266HostIf_SPI_Select .....	15
4.1.2 M8266WIFI_SPI_Interface_Communication_OK .....	16
4.1.3 M8266WIFI_SPI_Interface_Communication_Stress_Test .....	17
4.2 基础 WIFI 操作 .....	18
4.2.1 M8266WIFI_SPI_Get_Opmode .....	18
4.2.2 M8266WIFI_SPI_Set_Opmode .....	19
4.2.3 M8266WIFI_SPI_STA_Connect_Ap .....	20
4.2.4 M8266WIFI_SPI_Config_STA_Static_IP_Addr .....	21
4.2.5 M8266WIFI_SPI_Get_STA_Connection_Status .....	22
4.2.6 M8266WIFI_SPI_Get_STA_IP_Addr .....	23
4.2.7 M8266WIFI_SPI_STA_Query_Current_SSID_And_RSSI .....	24
4.2.8 M8266WIFI_SPI_STA_Scan_Signals .....	25
4.2.9 M8266WIFI_SPI_Get_STA_Hostname .....	26
4.2.10 M8266WIFI_SPI_Set_STA_Hostname .....	27
4.2.11 M8266WIFI_SPI_Query_STA_Param .....	28
4.2.12 M8266WIFI_SPI_Config_AP .....	29
4.2.13 M8266WIFI_SPI_Query_AP_Param .....	31
4.2.14 M8266WIFI_SPI_Config_AP_Param .....	33
4.3 UDP 或 TCP 服务/链接的建立与查询控制 .....	35
4.3.1 M8266WIFI_SPI_Setup_Connection .....	35

4.3.2 M8266WIFI_SPI_Delete_Connection .....	37
4.3.3 M8266WIFI_SPI_Query_Connection .....	38
4.3.4 M8266WIFI_SPI_Op_Multicast_Group .....	40
4.3.5 M8266WIFI_SPI_Set_TcpServer_Auto_Discon_Timeout .....	41
4.3.6 M8266WIFI_SPI_Query_Last_Tcp_Disconnect_Cause .....	42
4.3.7 M8266WIFI_SPI_Query_Tcp_Retrans_Max .....	43
4.3.8 M8266WIFI_SPI_Config_Tcp_Retrans_Max .....	44
4.3.9 M8266WIFI_SPI_Query_Tcp_Mss .....	45
4.3.10 M8266WIFI_SPI_Query_Tcp_Window_num .....	46
4.3.11 M8266WIFI_SPI_Config_Tcp_Window_num .....	47
4.3.12 M8266WIFI_SPI_STA_Get_HostIP_by_HostName .....	48
4.4 UDP 或 TCP 数据包的收发 .....	49
4.4.1 M8266WIFI_SPI_Send_Data .....	49
4.4.2 M8266WIFI_SPI_Send_Data_Block .....	51
4.4.3 M8266WIFI_SPI_Send_Udp_Data .....	52
4.4.4 M8266WIFI_SPI_Has_DataReceived .....	54
4.4.5 M8266WIFI_SPI_RecvData .....	55
4.4.6 M8266WIFI_SPI_RecvData_Ex .....	57
4.5 智能配网 .....	59
4.5.1 M8266WIFI_SPI_Module_SmartConfig .....	59
4.5.1 M8266WIFI_SPI_StartSmartConfig .....	61
4.5.2 M8266WIFI_SPI_StartWpsConfig .....	62
4.6 模组上的 WEB 服务器 .....	64
4.6.1 SPI_Set_Web_Server .....	64
4.7 低功耗 .....	66
4.7.1 M8266WIFI_SPI_Set_Tx_Max_Power .....	66
4.7.2 M8266WIFI_SPI_Sleep_Module .....	67
4.9 模组信息查询 .....	69
4.9.1 M8266WIFI_SPI_Get_Module_Info .....	69
4.9.2 M8266WIFI_SPI_Get_Driver_Info .....	70
4.10 其他可通过 SPI 接口的设置、查询、和控制功能 .....	71

# 1 前言

## 1.1 简介

Anylinkin® ALK8266WIFI® 是一款价格低廉、灵活、功能强大、高性能、精简小尺寸、绿色环保、高性价比的 802.11 b/g/n 无线模组。它包含有(1)高性能且高度集成的无线片上系统芯片 ESP8266EX, 提供智能高效的无线接入;(2)标准 2.0mm 间距的排针全孔半孔(复合邮票孔), 提供高速通信 SPI 从机接口;(3)串口数据排针半孔(邮票孔)接口, 提供 UART 串行通信接口;(4)同时, 还带有一些 IO 外设接口和 LED 灯, 可用于用户扩展。

通过 ALK8266WIFI® 模组所提供的 SPI 半孔整孔复合(邮票孔)接口, MCU 系统(1)可以实现和远端 TCP/UDP 服务节点实现高速通信, 最大波特率可达 40Mbps, 实测有效吞吐量可以超过 M 字节每秒, 适用于高速采集、语音、图片以及视频传输等场合;(2)通过我们提供的基本的 SPI 控制协议, 直接通过 SPI 总线接口就可以对模组及其片上资源进行设置、查询和控制, 无需 UART 串口介入, 以便节约主机的串口线用作其他功能。

为了方便广大用户快速实现单片机主机和 ALK8266WIFI® 模组的集成, 实现更高效的 SPI 主机接口通信, 我们提供了经过测试验证的相关单片机主机的例程包和主机驱动 API。

这些驱动 API 函数包含在我们提供的驱动库里。除了提供功能性的 API 操作, 该驱动库还集成我司独有的 SPI 高速高效读写、均衡与容错等优化算法, 大大提高了单片机主机高速读写的速度和效率。采用我司的 SPI 高速高效读写算法, 在单片机主机上实现实测比 DMA 速度更高更高效的 SPI 读写速度和效率; 采用 SPI 访问均衡和容错算法, 分析和依据主机的本地环境参数及射频通信的无线环境因素, 自动进行读写均衡与容错分析处理, 确保稳定性和性能的同时最优。

本文是对主机驱动 API 函数的相关说明, 涵盖(1)通过 SPI 接口对 ALK8266WIFI® 模组的设置查询和控制, 包括但不限于: 联网配网、状态查询、模组板载资源的操控、建立/查询/切断 TCP/UDP 链接等;(2)通过 SPI 实现高速数据传输;(3)其他等操作。

本文作为《[ALK8266WIFI 模组 SPI 接口高速通信使用与集成--主机集成说明](#)》辅助参考资料, 随着所支持的 API 函数的增多, 本文也会不断添加和更新。

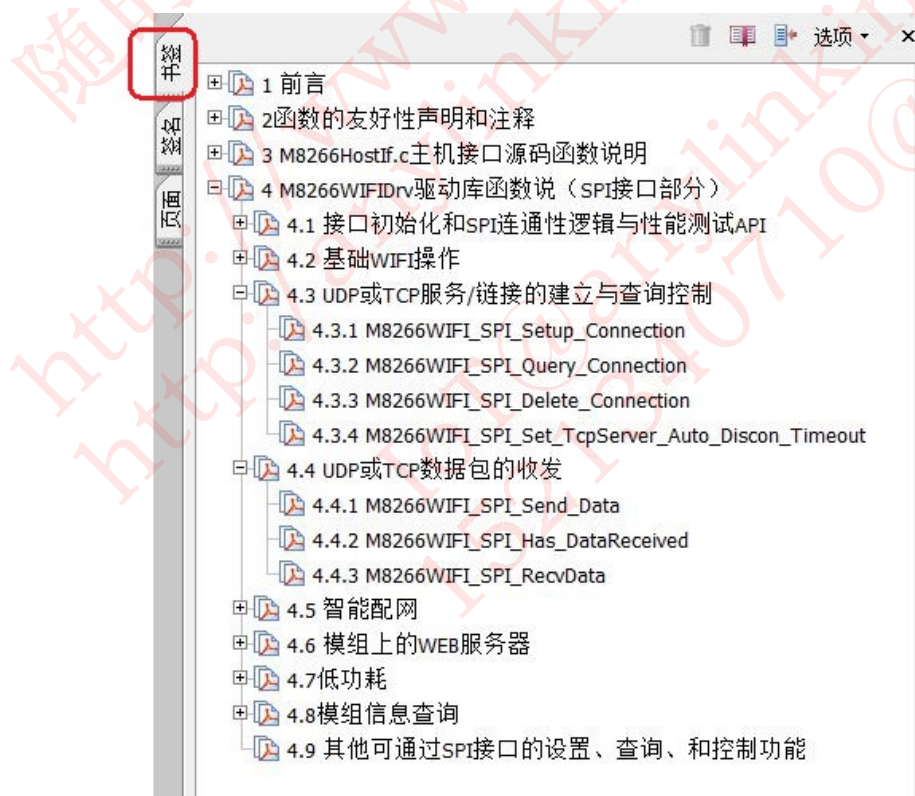
文中如有疏漏或认识不足之处, 欢迎大家积极指出、建议或批评斧正。谢谢!

## 1.2 术语约定

术语	解释
连接	一个广义的概念, 包括硬件电路上两个网表节点之间的连接、WIFI 工作站(STA)连接热点 AP 或路由器、网络层的 TCP 或 UDP 连接。在本文中, 为了避免混淆, 尽量使用接入和链接, 来区分后两种“连接”。如非特别声明, “连接”一般指硬件上的“逻辑互联”。
逻辑互联	特指硬件电路的两个网表节点之间, 用 PCB 布线、导线飞线等方式连接起来。
接入	特指 WIFI 工作站(STA)使用 SSID 和密码来连接热点 AP 或路由器, 对应英文单词 ACCESS。
链接	特指网络 TCP/IP 层的 TCP 或 UDP 套接字链路连接, 对应于模组上建立的一个服务 (Service)。
服务	一个网络节点开启的功能, 使用 IP 地址和端口标识, 通过套接字将两个节点的服务关联起来, 实现通信, 对应英文单词 Service。例如我们常说的 TCP 服务、UDP 服务等等, 英文单词是 service。

## 1.3 通过书签方便阅读和快速检索定位

本文档包含相关书签, 请以方便查阅和快速检索, 如下图所示



## 1.4 参考文档

- 1、ALK8266WIFI 模组数据手册
- 2、ALK8266WIFI 模组 SPI 接口高速通信使用与集成—主机集成说明
- 3、ALK8266WIFI 模组 SPI 接口高速通信使用与集成—常见问题
- 4、常见的 WIFI 模组配网方式简介与对比暨 ANYLINKIN 8266WIFI 模组配网方式和操作说明

## 1.5 联系我们

网址: <http://www.anylinkin.com>

淘宝: <http://anylinkin.taobao.com>  
<http://item.taobao.com/item.htm?id=576141575067>

电邮: [IoT@anylinkin.com](mailto:IoT@anylinkin.com), [1521340710@qq.com](mailto:1521340710@qq.com)

如有技术咨询、探讨、或疑惑, 欢迎和我们联系, 提出您的宝贵意见或建议。谢谢支持!



## 2 函数的友好性声明和注释

### 2.1 友好性声明

单片机主机例程包以及 API 头文件中, 在所有的函数体或函数声明前, 都有对该函数所实现的功能以及参数和调用说明。如果本文档中有说明不清楚之处, 可参考单片机例程包。如例所示:

```

/*****
 * M8266WIFI_SPI_Setup_Connection
 * .Description:
 *   To setup a UDP connection or an TCP client connection via SPI
 * .Parameter(s)
 *   1. tcp_udp      : connection type
 *                     =0, udp
 *                     =1, tcp client
 *                     =2, tcp server
 *   2. local_port   : local_port specified
 *                     =0, M8266WIFI module will generate a random local port
 *                     !=0, the local_port specified here will be used
 *   3. remote_ip    : string of ip address of remote connection
 *   4. remote_port  : port of remote connection
 *   5. link_no      : the number of link used for multiple links. Max 4
 *   6. timeout_in_s : the max timeout connecting to a remote, unit in seconds
 *   7. status       : pointer to return errcode(LSB) and status(MSB) upon error
 *                     Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Setup_Connection(u8 tcp_udp, u16 local_port,
                                   u8* remote_ip, u16 remote_port,
                                   u8 link_no, u8 timeout_in_s,
                                   u16* status);

```

### 2.2 友好性注释

在例程包中有详细的注释说明, 包括但不限于:

- (1) 在每个 API 函数的调用前, 都有对该函数的声明注释, 以方便阅读理解每个函数参数的具体意义, 如下例所示:

```

//u8 M8266WIFI_Reboot_And_Config_Connection(u8 tcp_udp, u16 local_port, u8* remote_ip, u16 remote_port, u8 link_no)
if(M8266WIFI_Config_Connection_via_SPI(TEST_CONNECTION_TYPE, TEST_LOCAL_PORT, TEST_REMOTE_IP_ADDR, TEST_REMOTE_PORT, link_no)==0)

```

- (2) 在必要处, 会有相应的详细解释。如下例所

```

/*****
 * M8266WIFI_Module_Hardware_Reset
 * Description
 * 1. To perform a hardware reset to M8266WIFI module via the nReset Pin
 * and bring M8266WIFI module to boot up from external SPI flash
 * 2. In order to make sure the M8266WIFI module bootup from external
 * SPI flash, nCS should be low during Reset out via nRESET pin
 * Parameter(s):
 * none
 * Return:
 * none
 *****/
void M8266WIFI_Module_Hardware_Reset(void) // total 800ms
{
    M8266HostIf_Set_SPI_nCS_Pin(0); // Module nCS==ESP8266 GPIO15 as well, should be low during reset in order for a normal reset
    M8266WIFI_Module_delay_ms(1); // delay 1ms, adequate for nCS stable

    M8266HostIf_Set_nRESET_Pin(0); // Pull low the nReset Pin to bring the module into reset state
    M8266WIFI_Module_delay_ms(1); // delay 1ms, adequate for nRESET stable

    M8266HostIf_Set_nRESET_Pin(1); // Pull high again the nReset Pin to bring the module exiting reset state
    M8266WIFI_Module_delay_ms(100); // at least 18ms required for reset-out-boot sampling bootstrap pin
    // Here, we use 50ms for adequate abundance

    M8266HostIf_Set_SPI_nCS_Pin(1); // delay 1ms, adequate for nCS stable
    //M8266WIFI_Module_delay_ms(1);

    M8266WIFI_Module_delay_ms(700-2); // Delay more than 500ms for M8266WIFI module bootup and initialization including bootup information print.
    // Wait time should be no less than 0.5s.
    // Here, we use wait 750ms
}

```



### 3 M8266HostIf.c 主机接口源码函数说明

这部分函数是单片机主机接口的底层实现, 需要根据实际的硬件连接来实现。

#### 3.1 M8266HostIf\_GPIO\_CS\_RESET\_Init

```
/* *****  
 * M8266HostIf_GPIO_SPInCS_nRESET_Pin_Init  
 * Description  
 * To initialise the GPIOs for SPI nCS and nRESET output for M8266WIFI module  
 * You may update the macros of GPIO PINs usages for nRESET from brd_cfg.h  
 * You are not recommended to modify codes below please  
 * Parameter(s):  
 * None  
 * Return:  
 * None  
 * ***** */  
void M8266HostIf_GPIO_CS_RESET_Init(void)
```

##### 功能

初始化 ALK8266WIFI® 模组的片选信号 nCS 和复位信号 nRESET 对应的主机 GPIO。

##### 说明

1、详情可参看 M8266HostIf.c 中的参考源码实现。

##### 函数原型

**void M8266HostIf\_GPIO\_CS\_RESET\_Init(void)**

##### 参数

无。

##### 返回值

无。

#### 3.2 M8266HostIf\_SPI\_Init

```
/* *****  
 * M8266HostIf_SPI_Init  
 * Description  
 * To initialise the SPI Interface for M8266WIFI module  
 * You may update the macros of SPI usages for nRESET from brd_cfg.h  
 * You are not recommended to modify codes below please  
 * Parameter(s):  
 * None  
 * Return:  
 * None  
 * ***** */  
void M8266HostIf_SPI_Init(void)
```

##### 功能

初始化 ALK8266WIFI® 模组所连接的主机的 SPI 接口。

##### 说明

- 1、当工作在主机接口模式 2 或模式 3 (即: 需要使用 SPI 接口) 时, 必须实现本函数。
- 2、SPI 主机接口的初始化, 需要工作在“主机模式”、“模式 A (CPOL=0, CPHA=0)”、以及字节内高位先发送。详情参见《[ALK8266WIFI 模组 SPI 接口高速通信使用与集成—主机集成说明](#)》中章节“4.1.2 SPI 主机接口的固件初始化”。
- 3、详情可参看 M8266HostIf.c 中的参考源码实现。

## 函数原型

```
void M8266HostIf_SPI_Init(void)
```

## 参数

无。

## 返回值

无。

## 3.3 M8266HostIf\_USART\_Init

```

/*****
 * M8266HostIf_USART_Init
 * Description
 *   To initialise the Host USART interface for M8266WIFI module
 *   e.g. STM32F4 Host Interface -- USART
 *   - If uses USART1, USART1_TXD=PA9, USART1_RXD=PA10
 *   - If uses USART2, USART2_TXD=PA2, USART2_RXD=PA3
 * Parameter(s):
 *   baud: baud rate to set
 * Return:
 *   None
 *****/
void M8266HostIf_USART_Init(u32 baud)

```

## 功能

初始化 ALK8266WIFI® 模组所连接的主机的 UART 接口。

## 说明

- 1、当工作在主机接口模式 1 和模式 3（即需要使用 UART 接口）时，必须实现本函数。
- 2、详情可参看 M8266HostIf.c 中的参考源码实现。

## 函数原型

```
void M8266HostIf_UART_Init(void)
```

## 参数

无。

## 返回值

无。

## 3.4 M8266HostIf\_Init

```

/*****
 * M8266HostIf_Init
 * Description
 *   To initialise the Host interface for M8266WIFI module
 * Parameter(s):
 *   baud: baud rate to set
 * Return:
 *   None
 *****/
void M8266HostIf_Init(void)
{
    M8266HostIf_GPIO_CS_RESET_Init();
#ifdef M8266WIFI_CONFIG_VIA_USART
    M8266HostIf_USART_Init(115200);
#endif
    M8266HostIf_SPI_Init();
    M8266HostIf_SPI_SetSpeed(SPI_BaudRatePrescaler_8);
}

```

## 功能

初始化主机接口。

## 说明

- 1 初始化 ALK8266WIFI® 模组所连接的主机接口, 主要包括: (1) SPI\_nCS 和 nRESET 管脚对应的 GPIO 初始化, (2) 当使用 SPI 接口时, SPI 主机接口的初始化, (3) 当使用 UART 串口接口时, UART 主机接口的初始化。
- 2 main()函数的初始化部分中会调用该函数来初始化主机接口。

## 函数原型

**void M8266HostIf\_Init(void)**

## 参数

无。

## 返回值

无。

## 3.5 M8266HostIf\_Set\_nRESET\_Pin

```
/* *****  
 * M8266HostIf_Set_nRESET_Pin  
 * Description  
 * To Output HIGH or LOW onto the GPIO pin for M8266WIFI nRESET  
 * You may update the macros of GPIO PIN usages for nRESET from brd_cfg.h  
 * You are not recommended to modify codes below please  
 * Parameter(s):  
 * 1. level: LEVEL output to nRESET pin  
 *           0 = output LOW onto nRESET  
 *           1 = output HIGH onto nRESET  
 * Return:  
 * None  
 * ***** */  
void M8266HostIf_Set_nRESET_Pin(u8 level)
```

## 功能

主机在 nRESET 所对应的 GPIO 管脚上输出高低电平。

## 说明

无

## 函数原型

**void M8266HostIf\_Set\_nRESET\_Pin(u8 level)**

## 参数

**u8 level:** 主机在 GPIO 上的输出电平。

- 0 = 输出低电平, 即复位模组。  
其他 = 输出高电平, 即退出复位。

## 返回值

无。

### 3.6 M8266HostIf\_Set\_SPI\_nCS\_Pin

```

/*****
 * M8266HostIf_Set_SPI_nCS_PIN
 * Description
 *   To Output HIGH or LOW onto the GPIO pin for M8266WIFI SPI nCS
 *   You may update the macros of GPIO PIN usages for SPI nCS from brd_cfg.h
 *   You are not recommended to modify codes below please
 * Parameter(s):
 *   1. level: LEVEL output to SPI nCS pin
 *           0 = output LOW  onto SPI nCS
 *           1 = output HIGH onto SPI nCS
 * Return:
 *   None
 *****/
void M8266HostIf_Set_SPI_nCS_Pin(u8 level)

```

#### 功能

主机在 SPI\_nCS 所对应的 GPIO 管脚上输出高低电平, 实现对模组的 SPI 片选控制。

#### 说明

1. 它被 **M8266WIFIDrv.lib** 所用到, 必须实现。

#### 函数原型

```
void M8266HostIf_Set_SPI_nCS_Pin(u8 level)
```

#### 参数说明

**u8 level:** 主机在 GPIO 上的输出电平。

0 = 输出低电平, 即选中模组。

其他 = 输出高电平, 即不选中模组。

#### 返回值

无。

### 3.7 M8266HostIf\_SPI\_ReadWriteByte

```

/*****
 * M8266HostIf_SPI_ReadWriteByte
 * Description
 *   To write a byte onto SPI bus from MCU to M8266WIFI module and read back
 *   a byte from the SPI bus meanwhile
 *   You may update the macros of SPI usages from brd_cfg.h
 *   You are not recommended to modify codes below please
 * Parameter(s):
 *   1. TxdByte: the byte to be sent
 * Return:
 *   The byte read back from SPI bus
 *****/
u8 M8266HostIf_SPI_ReadWriteByte(u8 TxdByte)

```

#### 功能

- 1、这个函数向 SPI 总线接口 (MOSI) 写一个字节, 同时, 从 SPI 总线接口 (MISO) 读取回一个字节。

#### 说明

- 1、这个函数只是单纯的 SPI 总线操作, 不牵涉到片选信号的控制。
- 2、这个函数的实现步骤应该是: (1) 等待 SPI 发送寄存器为空; (2) 向 SPI 发送寄存器写如一个数据; (3) 等待 SPI 接收寄存器是否收到了数据; (4) 从接收寄存器读取数据。如下例所示 (这个函数是 STM32 上通过 SPI 读写的函数)

```

u8 M8266Host_SPI_ReadWriteByte(u8 TxData)
{
    while (SPI_I2S_GetFlagStatus(SPI1, SPI_I2S_FLAG_TXE) == RESET){} // Wait SPI TXD Buffer Empty
    SPI_I2S_SendData(SPI1, TxData); // Write the data to the TXD buffer and then shift out from MOSI
    while (SPI_I2S_GetFlagStatus(SPI1, SPI_I2S_FLAG_RXNE) == RESET){} // Wait the SPI RXD buffer has
    // received the data from MISO
    return SPI_I2S_ReceiveData(SPI1); // Return the data received from MISO
}

```

- 3、这种方式所实现的 SPI 读写效率很低, 在我们所提供的驱动库中, 已经废弃不用。  
所以, 对于这些平台, 可以不实现这个函数。

### 函数原型

**void M8266HostIf\_SPI\_ReadWriteByte(u8 TxData)**

### 参数说明

**u8 TxData:** 主机向 SPI 写的字节数值。

### 返回值

**u8。** SPI 总线向主机返回的字节数值

## 3.8 M8266HostIf\_delay\_us

```

/*****
 * M8266WIFIHostIf_delay_us
 * Description
 *   To loop delay some micro seconds.
 * Parameter(s):
 *   1. nus: the micro seconds to delay
 * Return:
 *   none
 *****/
void M8266HostIf_delay_us(u8 nus)
{
    delay_us(nus);
}

```

### 功能

- 1、这个函数用于延迟一定时间的微秒数。
- 2、这个函数的实现, 必须确保足够的精度。

### 说明

- 1、它被 **M8266WIFIDrv.lib** 所用到, 必须根据响应的平台精确实现。  
建议通过独立运行的定时器, 而非代码的执行延迟来实现。  
在存在任务调度的操作系统上, 注意处理好任务调度对延迟精度的影响。
- 2、上述截图举例中, m8266\_delay\_us 调用了 delay\_us(nus), 后者是 STM32 系统中通过定时器所实现的一个延迟函数。对于非 STM32 平台, 有其他对应的延迟函数。在我们所提供的单片机例程包中, 一般都提供了相应单片机平台的定时器实现的精度延迟的参考代码, 可以查阅例程包中的实现源码。

### 参数说明

**u8 nus:** 需要延迟的微秒数。

### 返回值

无。

## 4 M8266WIFIDrv 驱动库函数说 (SPI 接口部分)

以下库函数仅在使用 SPI 主机接口 (无论是用来设置模组还是用于数据通信) 时才需要使用。这些 API 函数的具体实现被封装在了 M8266WIFIDrv 库文件中, 在头文件 M8266WIFIDrv.h 中, 在每个 API 函数的函数声明部分, 有对应函数的详细的说明。

驱动库中的 API 函数, 包括如下几个部分:

- (1) 接口初始化的逻辑验证和性能测试
- (2) 基础 WIFI 操作
- (3) UDP/TCP 服务链接的建立、查询与控制
- (4) UDP/TCP 数据包的高速收发
- (5) 智能配网操作
- (6) 模组上的 WEB 服务器的配置和控制
- (7) 低功耗支持
- (8) 模组基本信息的查询
- (9) 其他

其中, 核心部分为 (2) (3) (4) 部分, 其他为辅助功能, 提供功能扩展。



## 4.1 接口初始化和 SPI 连通性逻辑与性能测试 API

影响 ALK8266WIFI® 模组高速通信性能的因素在于路由器网络环境(包括信道的干扰冲突和阻塞、所支持的最大无线速度等因素),以及主机 MCU 对于 ALK8266WIFI® 模组的 SPI 接口效率(包括 SPI 时钟以及读写效率,即每个 CLOCK 周期平均能读/写多少个有效数据字节,需要多少等待延迟等等)。

本驱动优化了主机 SPI 的读写效率,同时,对外提供一些 API 函数和方法,来协助检查主机的 SPI 接线或初始化是否正确(功能测试),以及找到当前接线方式下可靠通信的最大 SPI 频率,来确保充分发挥响 ALK8266WIFI® 模组的高速通信性能。

这一部分 API,是实现高速 WIFI 通信的第一步,即确保主机和模组之间在尽可能高的 SPI 频率下实现可靠的通信。

### 4.1.1 M8266HostIf\_SPI\_Select

```

/*****
 * M8266HostIf_SPI_Select
 * Description
 * 1.To specify the SPI used by providing the base address of SPI used
 * 2.Called when Initial M8266WIFI Module before perform SPI read/write
 * Parameter(s):
 * 1. spi_base_addr: the base address of used
 *    you may use M8266WIFI_INTERFACE_SPI defined in brd_cfg.h as example
 * 2. spi_clock: the spi_clock you set in HZ
 *    e.g. 10500000 for 10.5MHz
 * 3. status when error
 * Return:
 * 1. success
 * 0. failure
 *****/
u8 M8266HostIf_SPI_Select(uint32_t spi_base_addr, uint32_t spi_clock, u16* status);

```

#### 功能

- 1、这个函数向驱动传递所使用的 SPI 基地址,并告诉驱动所使用的 SPI 频率。

#### 说明

- 1、通过向驱动传递所使用的 SPI 基地址(指针),使得驱动可以兼容主机任意的 SPI 通道;通过向驱动传递目前主机所使用的 SPI 频率,驱动可以根据相应的 SPI 频率对时序进行微调优化,以最大化通信性能。
- 2、**必须在初始化完主机的 SPI 接口之后,并在开始使用 SPI 之前,调用本函数。**

#### 函数原型

**u8 M8266HostIf\_SPI\_Select(u32 spi\_base\_addr, u32 spi\_clock, u16\* status)**

#### 参数

**u32 spi\_base\_addr**

和模组连接的主机 SPI 的基地址。

**u32 spi\_clock**

主机 SPI 的频率,单位 Hz。

**u16 \*status**

异常时返回的状态字节。

这个函数在执行时, 会以极低的 SPI 频率和模组进行尝试通信。当无法正常通信时, 会在 **status** 的高字节返回从 SPI 总线上读取得到的字节数据。

因此, 当返回的 **status** 的高字节为 0xFF 即  $((\text{status}) \gg 8) == 0xFF$  时, 一般是因为 SPI 初始化不正确(例如将 nCS 的 GPIO 初始化成了输入、或者 SPI 的模式初始化出错了等原因), 或者主机接口的接线不正确(例如 MISO 和 MOSI 接反了、没有接牢固、或者片选信号的高低电平输出不到轨(rail-to-rail, 例如高电平的输出不足 3.0V 等等)等原因。而当返回的 **status** 的高字节为 0x00 即  $((\text{status}) \gg 8) == 0x00$  时, 一般可能是因为 MISO 线对地短路, 或者 MISO 和 MOSI 接反了, 或者该 SPI 总线上还有其他设备引入了读写冲突等原因。

#### 返回值

=1, 执行成功

=0, 出错了

#### 4.1.2 M8266WIFI\_SPI\_Interface\_Communication\_OK

```

/*****
 * M8266WIFI_SPI_Interface_Communication_OK
 * Description
 * 1. To write a byte and then read out from M8266WIFI module SPI registers
 *    to check whether the logical fundamental SPI communication (read/write)
 *    is stably OK under the clock and interconnection
 * 2. Called during the initialization of M8266WIFI Module
 *    DONOT CALL IT AFTER MODULE HAS STARTED THE NORMAL OPERATION
 * Parameter(s):
 * 1. byte: a pointer to the byte read out during test
 *    - normally the data should be 0x41 during test.
 *    - if it is 0x41, this function will return 1 for success as well
 *    - If readout other value, it may indicating the fundamental SPI
 *      communication is not OK. e.g. 0xFF may indicates that your spi
 *      port has problem such incoreect interconnection or initialization
 *    - user NULL if you don't expect this data
 * Return:
 * 1. success
 * 0. failure
 *****/
u8 M8266WIFI_SPI_Interface_Communication_OK(u8* byte);

```

#### 功能

- 1、这个函数用于开发调试阶段的 SPI 通信的逻辑功能测试。主机执行这个函数, 会向模组执行单次的基本字节读写, 并校验读写是否正确。

#### 说明

- 1、开发人员可以使用这个函数, 判断自己的接线和主机接口初始化是否正确。
- 2、这个函数仅在开发调试阶段用作功能测试, 在正式发布的单片机主机固件中, 建议不要使用此函数, 特别不要在开始使用 SPI 配置和数据通信后再调用此函数。

#### 函数原型

**u8 M8266HostIf\_SPI\_Interface\_Communication\_OK(u8\* byte)**

#### 参数

**u8 \*byte**

出错时, 主机从 SPI 总线上读回的字节数据。

#### 返回值

=1, 执行成功

=0, 出错了

### 4.1.3 M8266WIFI\_SPI\_Interface\_Communication\_Stress\_Test

```

/*****
 * M8266WIFI_SPI_Interface_Communication_Stress_Test
 * Description
 * 1. To perform a batch of byte write and then read out from M8266WIFI module
 *    SPI register to check whether the high-speed and bulk fundamental SPI
 *    communication (read/write) is stably OK under the clock
 * 2. Called during the initialization of M8266WIFI Module
 *    DONOT CALL IT AFTER MODULE HAS STARTED THE NORMAL OPERATION
 * 3. You may call it in your debug code for speed stability test
 *    during your stress performance test to your produc
 * Parameter(s):
 * 1. max_times: the max write-read times used for the stress test
 *    - And the test data to be written will be number byte of data
 *    from 0 to max_times
 * Return:
 * 1. success times of write-read-verify
 *****/
u32 M8266WIFI_SPI_Interface_Communication_Stress_Test(u32 max_times);

```

#### 功能

- 1、这个函数用于开发调试阶段的 SPI 通信的性能测试。主机执行这个函数, 会向模组执行多次的基本字节连续读写, 并校验读写是否正确和记录正确的次数。
- 2、根据传递给该函数的参数 **max\_times** 和该函数的返回数值 (读写校验正确的次数) 的差值, 可以判断, 在该频率下高速连续读写的可靠性。如果该差值为 0, 则表明在该频率下, SPI 的高速连续读写非常可靠, 可以适当进一步提高 SPI 进一步测试。而如果该数值的差别较大, 则可能因为布线或接线不够好, 而需要适当降低 SPI 的频率来保证通信的可靠性。

#### 说明

- 1、开发人员可以使用本函数, 判断当前的 SPI 频率下 SPI 的高速连续读写的可靠性。
- 2、本函数仅在开发调试阶段用作性能测试, 在正式发布的单片机主机固件中, 建议不要使用此函数, 特别不要在开始使用 SPI 配置和数据通信后再调用此函数。

#### 函数原型

**u32 M8266HostIf\_SPI\_Interface\_Communication\_Stress\_Test(u32 max\_times)**

#### 参数

**u32 max\_times**

指定压力测试时, 对模组连续读写的次数。一般设定该数值不小于 10 万。

#### 返回值

读写校验正确的次数

## 4.2 基础 WIFI 操作

要让单片机可以通过模组进行通信,首先要让模组作为 STA 加入 WIFI 网络,或作为 AP 建立局域网以便其他的节点加入这个局域网。这部分 API 提供了 WIFI 模组联网或组网、查询 IP、查询信号强度、配置 AP 等基础 WIFI 功能。

这一部分 API,是实现高速 WIFI 通信的基础,即支持主机配置模组联网或组网等功能。

### 4.2.1 M8266WIFI\_SPI\_Get\_Opmode

```

/*****
 * M8266WIFI_SPI_Get_Opmode
 * .Description:
 *   To get the op_mode(STA, AP, or STA+AP) of M8266WIFI module via SPI
 * .Parameter(s)
 *   1. op_mode : pointer to the op_mode returned
 *       =1,      STA mode
 *       =2,      AP mode
 *       =3,      STA+AP mode
 *   2. status  : pointer to return errcode(LSB) and status(MSB) upon error
 *               Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Get_Opmode(u8* op_mode, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 接口读取模组当前的 WIFI 操作模式。

#### 说明

- 1、适用于主机接口模式 2（只使用 SPI 接口不使用串口）下。

#### 函数原型

**u8 M8266WIFI\_SPI\_Get\_Opmode (u8\* op\_mode, u16\* status)**

#### 参数

**u8\* op\_mode**

执行正确时,读取得到的模组当前的操作模式。

=1, STA 模式

=2, AP 模式

=3, STA+AP 混合模式

**u16\* status**

执行出错时,返回的状态码的指针,方便故障诊断。

如果不需要返回状态码,可以使用 NULL。

#### 返回值

=1, 执行成功

=0, 出错了

## 4.2.2 M8266WIFI\_SPI\_Set\_Opmode

```

/*****
 * M8266WIFI_SPI_Set_Opmode
 * .Description:
 *   To set op_mode(STA, AP, or STA+AP) of M8266WIFI module via SPI
 * .Parameter(s)
 *   1. op_mode : the op_mode to set
 *       =1,      set to STA mode
 *       =2,      set to AP mode
 *       =3,      set to STA+AP mode
 *       =others, set to STA+AP mode
 *   2. saved    : to save into flash the opmode or not
 *       =0,      not saved, i.e. after reboot setting will restore to previous
 *       =others, saved, i.e. after reboot, the saved setting will be loaded
 *               PLEASE DO NOT CALL IT EACH TIME OF BOOTUP WITH SAVED != 0
 *               OR, THE FLASH ON MODULE MIGHT GO TO FAILURE DUE TO LIFE CYCLE
 *               OF WRITE
 *   3. status   : pointer to return errcode(LSB) and status(MSB) upon error
 *               Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Set_Opmode(u8 op_mode, u8 saved, u16* status);

```

### 功能

- 1、这个函数用于通过 SPI 接口设置模组当前的 WIFI 操作模式。

### 说明

- 1、适用于主机接口模式 2（只使用 SPI 接口不使用串口）下。

### 函数原型

**u8 M8266WIFI\_SPI\_Set\_Opmode(u8 op\_mode, u8 saved, u16\* status)**

### 参数

#### **u8\* op\_mode**

对模组设置的操作模式。

=1, STA 模式

=2, AP 模式

=3, STA+AP 混合模式

=其他, STA+AP 混合模式

#### **u8 saved**

是保存该配置

=0, 不保存到模组上的 FLASH 里, 下次启动时无效;

=1, 保存到模组上的 FLASH 里, 下次启动时依然有效。

#### **u16\* status**

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

### 返回值

=1, 执行成功

=0, 出错了



### 4.2.3 M8266WIFI\_SPI\_STA\_Connect\_Ap

```

/*****
 * M8266WIFI_SPI_STA_Connect_Ap
 * .Description:
 *   To connect the M8266WIFI STA to an AP or router via SPI
 * .Parameter(s)
 *   1. ssid      : the ssid of AP connected to. Max len = 32 Bytes
 *   2. password   : the password of AP connecting to. Max len = 64 Bytes
 *   3. saved      : to save the ssid and password into flash the opmode or not
 *                   =0,      not saved, i.e. after reboot setting will restore to previous
 *                   =others, saved, i.e. after reboot, the saved setting will be loaded
 *                   PLEASE DO NOT CALL IT EACH TIME OF BOOTUP WITH SAVED != 0
 *                   OR, THE FLASH ON MODULE MIGHT GO TO FAILURE DUE TO LIFT CYCLE
 *                   OF WRITE
 *   4. timeout_in_s: max time in seconds waiting for being connected
 *   5. status     : pointer to return errcode(LSB) and status(MSB) upon error
 *                   Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_STA_Connect_Ap(u8 ssid[32], u8 password[64], u8 saved, u8 timeout_in_s, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 接口让模组接入热点（包括路由器）。

#### 说明

- 1、适用于主机接口模式 2（只使用 SPI 接口不使用串口）下。

#### 函数原型

```

u8 M8266WIFI_SPI_STA_Connect_Ap(
    u8 ssid[32],
    u8 password[64],
    u8 saved,
    u8 timeout_in_s,
    u16* status);

```

#### 参数

**u8 ssid[32]**

需要连接到的热点的 SSID，最大支持 32 字节

**u8 password[64]**

需要连接到的热点的密码，最大支持 64 字节

**u8 saved**

是否保存该配置。如果保存，下次开机时会从 FLASH 里读取该参数并自动联网。

=0, 不保存到模组上的 FLASH 里，下次启动时无效；

=1, 保存到模组上的 FLASH 里，下次启动时依然有效。

**u8 time\_out\_in\_s**

连接超时时间，单位是秒。

如果在 timeout\_in\_s 秒的时间内，没有连接成功，则因超时而返回。

这个时间是“超时”时间，不是该函数的执行时间，即，如果在这个时间内连接成功了，该函数也会提前成功返回。

**u16\* status**

执行出错时，返回的状态码的指针，方便故障诊断。

如果不需要返回状态码，可以使用 NULL。



## 返回值

=1, 执行成功

=0, 出错了

## 4.2.4 M8266WIFI\_SPI\_Config\_STA\_Static\_IP\_Addr

```

/*****
 * M8266WIFI_SPI_Config_STA_Static_IP_Addr
 * .Description:
 *   To config static ip addresses of the module STA via SPI
 * .Parameter(s)
 *   1. ip_addr : the ip address to set
 *               - SHOULD be string like "192.168.1.103"
 *               - USE NULL if don't expect to set it
 *   2. gw_addr : the gateway address to set
 *               - SHOULD be string like "192.168.1.1"
 *               - USE NULL if don't expect to set it
 *   3. netmask : the netmask address to set
 *               - SHOULD be string like "255.255.255.0"
 *               - USE NULL if don't expect to set it
 *   4. status : pointer to return errcode(LSB) and status(MSB) upon error
 *               Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Config_STA_Static_IP_Addr(char* ip_addr, char* gw_addr, char* netmask, u16* status);

```

## 功能

1、这个函数用于通过 SPI 接口设置模组(STA 下)的静态 IP 地址。

## 说明

1、适用于主机接口模式 2（只使用 SPI 接口不使用串口）下。

## 函数原型

```

u8 M8266WIFI_SPI_Config_STA_Static_IP_Addr(
    char* ip_addr, char* gw_addr, char* netmask,
    u16* status)

```

## 参数

**char\* ip\_addr**

需要设置的 IP 地址, 字符串格式, 例如, "192.168.1.103"。

如果不想设置该参数, 可以使用 NULL。

**char\* gw\_addr**

需要设置的网关地址, 字符串格式, 例如, "192.168.1.1"。

如果不想设置该参数, 可以使用 NULL。

**char\* netmask**

需要设置的子网掩码地址地址, 字符串格式, 例如, "255.255.255.0"。

如果不想设置该参数, 可以使用 NULL。

**u16\* status**

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

## 返回值

=1, 执行成功

=0, 出错了

## 4.2.5 M8266WIFI\_SPI\_Get\_STA\_Connection\_Status

```

/* *****
 * M8266WIFI_SPI_Get_STA_Connection_Status
 * .Description:
 *   To get the connecting status of M8266WIFI STA to AP or routers via SPI
 * .Parameter(s)
 *   1. connection_status : the connection status returned
 *       =0, if station is in idle
 *       =1, if station is connecting to an AP
 *       =2, if station is to connect to an AP but with an wrong password
 *       =3, if station is to connect to an AP which is not found
 *       =4, if station failed to connect to the AP
 *       =5, if station is connected to the AP and has got the IP successfully
 *       =255, if the module is in AP-only mode
 *   2. status : pointer to return errcode(LSB) and status(MSB) upon error
 *               Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 * *****
 */
u8 M8266WIFI_SPI_Get_STA_Connection_Status(u8* connection_status, u16* status);

```

### 功能

- 2、这个函数用于通过 SPI 接口查询模组对热点 AP（包括路由器）的接入状态。  
这个函数常常用来判断 WIFI 是否掉线了。

### 说明

- 1、适用于主机接口模式 2（只使用 SPI 接口不使用串口）下。

### 函数原型

**u8 M8266WIFI\_SPI\_Get\_STA\_Connection\_Status (**  
**u8\* connection\_status, u16\* status)**

### 参数

**u8\* connection\_status**

返回连接状态的指针，该指针的内容为

- = 0, 没有连接
- =1, 正在连接 AP 的过程中
- =2, 连接失败，因为密码错了
- =3, 连接失败，因为没有搜索到指定的 SSID
- =4, 连接失败，因为其他原因
- =5, 连接成功

**u16\* status**

执行出错时，返回的状态码的指针，方便故障诊断。

如果不需要返回状态码，可以使用 NULL。

### 返回值

- =1, 执行成功
- =0, 出错了

#### 4.2.6 M8266WIFI\_SPI\_Get\_STA\_IP\_Addr

```

/*****
 * M8266WIFI_SPI_Get_STA_IP_Addr
 * .Description:
 *   To get ip address of M8266WIFI STA via SPI
 * .Parameter(s)
 *   1. sta_ip : the sta ip address returned if successful
 *               "0.0.0.0" returned if in AP-only mode or ip not achieved
 *   2. status : pointer to return errcode(LSB) and status(MSB) upon error
 *               Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Get_STA_IP_Addr(char* sta_ip , u16* status);

```

##### 功能

- 1、这个函数用于通过 SPI 接口获取模组的 STA 模式 IP 地址。

##### 说明

- 1、通过读取该数值,可以判断工作在 STA 或 STA+AP 模式的模组是否接入了路由器并获取到了 IP 地址。
- 2、如果模组工作在 STA 或 STA+AP 模式,并通过外部路由器链接远端服务节点,那么在建立链接之前,需要调用此函数先确保模组接入上了路由器并获取到了 IP 地址。
- 3、适用于主机接口模式 2 (只使用 SPI 接口不使用串口)下。

##### 函数原型

```

u8 M8266WIFI_SPI_Get_STA_IP_Addr (
    char* sta_ip,
    u16* status)

```

##### 参数

**char\* sta\_ip**

返回的 ip 地址

**u16\* status**

执行出错时,返回的状态码的指针,方便故障诊断。

如果不需要返回状态码,可以使用 NULL。

##### 返回值

=1, 执行成功

=0, 出错了

## 4.2.7 M8266WIFI\_SPI\_STA\_Query\_Current\_SSID\_And\_RSSI

```

/*****
 * M8266WIFI_SPI_STA_Query_Current_SSID_And_RSSI
 * .Description:
 *   To get current AP info (SSID and RSSI) M8266WIFI STA connected to
 * .Parameter(s)
 *   1. ssid      : the current SSID returned which m8266wifi sta connected to
 *   2. rssi      : the rssi of current connected ssid. 31 if error
 *   3. status    : pointer to return errcode(LSB) and status(MSB) upon error
 *                  Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_STA_Query_Current_SSID_And_RSSI(char* ssid, s8* rssi, u16* status);

```

### 功能

- 1、这个函数用于通过 SPI 接口获取模组当前所接入的热点的 SSID 以及信号强度。

### 说明

- 1、当模组处于 AP-Only 模式时, 该函数会返回错误。
- 2、适用于主机接口模式 2 (只使用 SPI 接口不使用串口) 下。

### 函数原型

```

u8 M8266WIFI_SPI_STA_Query_Current_SSID_And_RSSI (
    char* ssid,
    s8* rssi,
    u16* status)

```

### 参数

**char\* ssid**

返回当前连接的 SSID

**s8\* rssi**

返回当前连接的信号强度

**u16\* status**

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

### 返回值

=1, 执行成功

=0, 出错了

## 4.2.8 M8266WIFI\_SPI\_STA\_Scan\_Signals

```

/*****
 * M8266WIFI_SPI_STA_Scan_Signals
 * .Description:
 *   To perform a scanning procedure, the scanned signals sorted by rssi
 * .Parameter(s)
 *   1. scanned_signals : the return signals after scanning
 *   2. max_signals      : max counts of signals to scan
 *   3. channel          : channel to scan. USE 0xFF to scan all channels
 *   4. timeout_in_s     : time out in seconds
 *   5. status          : pointer to return errcode(LSB) and status(MSB) upon error
 *                       Use NULL if you don't expect them returned
 * .Return value:
 *   !=0, signal numbers scanned successfully
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_STA_Scan_Signals(struct ScannedSigs scanned_signals[], u8 max_signals,
                                   u8 channel, u8 timeout_in_s, u16* status);

```

### 功能

- 1、这个函数用于通过 SPI 接口扫描周边的热点信号。

### 说明

- 1、当模组处于 AP-Only 模式时，该函数会返回错误。
- 2、适用于主机接口模式 2（只使用 SPI 接口不使用串口）下。

### 函数原型

```

u8 M8266WIFI_SPI_STA_Scan_Signals(
    struct ScannedSigs scanned_signals[],
    u8 max_signals,
    u8 channel,
    u8 timeout_in_s,
    u16* status);

```

### 参数

**struct ScannedSigs scanned\_signals[]**

一个 ScannedSigs 结构类型的数组，用于返回扫描得到的热点信息，按照信号强度排序。

```

struct ScannedSigs{
    char ssid[32];
    u8   channel;
    u8   authmode;
    s8   rssi;
};

```

在调用本函数之前，将这个结构数组初始化为全零，这样，当扫描的个数较少于 max\_signals，可以通过结尾零的个数来判断实际返回的个数。

**u8 max\_signals**

指定所需要返回的信号最大个数。如果周边的可扫描的热点信号个数超过这里所指定的最大个数，那么将只返回信号最强的前 max\_signals 个热点信号。

**u8 channel**

指定所扫描的信道。如果使用 0xFF 来扫描全部的信道。

#### u8 timeout\_in\_s

执行扫描操作的最大超时时间, 单位是秒。

#### u16\* status

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

#### 返回值

=1, 执行成功

=0, 出错了

### 4.2.9 M8266WIFI\_SPI\_Get\_STA\_Hostname

```

/*****
 * M8266WIFI_SPI_Get_STA_Hostname
 * .Description:
 *   To get the hostname of the STA via SPI
 *   Note: Will return failure if module in AP-Only Mode
 * .Parameter(s)
 *   1. hostname   : the hostname to get
 *   2. status     : pointer to the status upon failure
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Get_STA_Hostname(char hostname[28+1], u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 接口获取模组 STA 模式下的 Hostname。

#### 说明

- 1、当模组处于 AP-Only 模式时, 该函数会返回错误。
- 2、适用于主机接口模式 2 (只使用 SPI 接口不使用串口) 下。

#### 函数原型

```

u8 M8266WIFI_SPI_Get_STA_Hostname (
    char hostname[28+1],
    u16* status)

```

#### 参数

**char hostname[28+1]**

返回 STA 模式下的 Hostname

**u16\* status**

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

#### 返回值

=1, 执行成功

=0, 出错了



#### 4.2.10 M8266WIFI\_SPI\_Set\_STA\_Hostname

```

/*****
 * M8266WIFI_SPI_Set_STA_Hostname
 * .Description:
 *   To set up the hostname of the STA via SPI
 *   Note: Will return failure if module in AP-Only Mode
 * .Parameter(s)
 *   1. hostbame      : the hostame to setup
 *   2. status        : pointer to the status upon failure
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Set_STA_Hostname(char hostname[28+1], u16* status);

```

##### 功能

- 1、这个函数用于通过 SPI 接口设置模组 STA 模式下的 Hostname。

##### 说明

- 1、当模组处于 AP-Only 模式时，该函数会返回错误。
- 2、适用于主机接口模式 2（只使用 SPI 接口不使用串口）下。

##### 函数原型

```

u8 M8266WIFI_SPI_Set_STA_Hostname (
    char hostname[28+1],
    u16* status)

```

##### 参数

**char hostname[28+1]**

需要设置的 STA 模式下的 Hostname

**u16\* status**

执行出错时，返回的状态码的指针，方便故障诊断。

如果不需要返回状态码，可以使用 NULL。

##### 返回值

- =1，执行成功
- =0，出错了

#### 4.2.11 M8266WIFI\_SPI\_Query\_STA\_Param

```

/*****
 * M8266WIFI_SPI_Query_STA_Param
 * .Description:
 *   To query a parameter of the module STA via SPI
 * .Parameter(s)
 *   1. param_type : the param type to set, enum of STA_PARAM_TYPE
 *       STA_PARAM_TYPE_SSID      = 0,
 *       STA_PARAM_TYPE_PASSWORD  = 1,
 *       STA_PARAM_TYPE_CHANNEL   = 2,
 *       STA_PARAM_TYPE_HOSTNAME  = 3,
 *       STA_PARAM_TYPE_USE_BSSID = 4,
 *       STA_PARAM_TYPE_BSSID     = 5,
 *       STA_PARAM_TYPE_RSSI      = 6,
 *       STA_PARAM_TYPE_IP_ADDR   = 7,
 *       STA_PARAM_TYPE_GATEWAY_ADDR = 8,
 *       STA_PARAM_TYPE_NETMASK_ADDR = 9,
 *   2. param      : pointer to the param value returned
 *   3. param_len  : pointer to length the param, unit in bytes
 *   4. status     : pointer to return errcode(LSB) and status(MSB) upon error
 *                   Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Query_STA_Param(STA_PARAM_TYPE param_type, u8* param, u8* param_len, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 接口获取模组 STA 模式下的单个参数。

#### 说明

- 1、当模组处于 AP-Only 模式时，该函数会返回错误。
- 2、适用于主机接口模式 2（只使用 SPI 接口不使用串口）下。

#### 函数原型

```

M8266WIFI_SPI_Query_STA_Param(
    STA_PARAM_TYPE param_type,
    u8* param,
    u8* param_len,
    u16* status)

```

#### 参数

**STA\_PARAM\_TYPE param\_type**

这个枚举类型，用于指定查询的参数类型

```

typedef enum{
    STA_PARAM_TYPE_SSID      = 0,
    STA_PARAM_TYPE_PASSWORD  = 1,
    STA_PARAM_TYPE_CHANNEL   = 2,
    STA_PARAM_TYPE_HOSTNAME  = 3,
    STA_PARAM_TYPE_USE_BSSID = 4,
    STA_PARAM_TYPE_BSSID     = 5,
    STA_PARAM_TYPE_RSSI      = 6,
    STA_PARAM_TYPE_IP_ADDR   = 7,
    STA_PARAM_TYPE_GATEWAY_ADDR = 8,
    STA_PARAM_TYPE_NETMASK_ADDR = 9,
}STA_PARAM_TYPE;

```

各个参数类型的说明如下

STA\_PARAM\_TYPE\_SSID: 返回当前连接的 SSID, 参数最长 32 字节

STA\_PARAM\_TYPE\_PASSWD: 返回当前连接的密码, 参数最长 64 字节

STA\_PARAM\_TYPE\_CHANNEL: 返回当前处于的频道

STA\_PARAM\_TYPE\_HOSTNAME: 返回当前在热点中显示的主机名

STA\_PARAM\_TYPE\_USE\_BSSID: 返回连接热点时需要校验热点的 BSSID

STA\_PARAM\_TYPE\_BSSID: 返回连接的热点的 BSSID (MAC 地址)

STA\_PARAM\_TYPE\_RSSI: 返回当前连接所连接热点的信号强度

STA\_PARAM\_TYPE\_IP\_ADDR: 返回当前的 IP 地址

STA\_PARAM\_TYPE\_GATEWAY\_ADDR: 返回当前的网关地址

STA\_PARAM\_TYPE\_NETMASK\_ADDR: 返回当前的子网掩码

### U8\* param

指针, 指向返回的参数

### U8\* param\_len

指针, 指向返回的参数长度, 单位字节

### u16\* status

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

### 返回值

=1, 执行成功

=0, 出错了

## 4.2.12 M8266WIFI\_SPI\_Config\_AP

```

/*****
 * M8266WIFI_SPI_Config_AP
 * .Description:
 *   To config the module AP via SPI
 * .Parameter(s)
 *   1. ssid : the ssid of AP, Max Size=13 Bytes
 *   2. password: the password of AP, Max Size=13 Bytes
 *   3. enc : authorisation mode, valid value is 0~4
 *           = 0, OPEN
 *           = 1, WEP
 *           = 2, WPA_PSK
 *           = 3, WPA2_PSK
 *           = 4, WPA_WPA2_PSK
 *           = others, OPEN
 *   4. channel: the channel of AP, valid value is 1~13.
 *       - Value other than 1~13 will set to channel 6
 *   5. saved : to save the ssid and password into flash the opmode or not
 *           =0, not saved, i.e. after reboot setting will restore to previous
 *           =others, saved, i.e. after reboot, the saved setting will be loaded
 *           PLEASE DO NOT CALL IT EACH TIME OF BOOTUP WITH SAVED != 0
 *           OR, THE FLASH ON MODULE MIGHT GO TO FAILURE DUE TO LIFT CYCLE
 *           OF WRITE
 *   6. status : pointer to return errcode(LSB) and status(MSB) upon error
 *           Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Config_AP(u8 ssid[13+1], u8 password[13+1], u8 enc, u8 channel, u8 saved, u16* status);

```

### 功能

1、这个函数用于通过 SPI 接口配置模组热点 AP 模式下的 AP 参数。

### 说明

1、适用于主机接口模式 2 (只使用 SPI 接口不使用串口) 下。

## 函数原型

```
u8 M8266WIFI_SPI_Config_AP (  
    u8 ssid[13+1],  
    u8 password[13+1],  
    u8 enc,  
    u8 channel,  
    u8 saved,  
    u16* status)
```

## 参数

**u8 ssid[14+1]**

需要设置的 AP 模式下的 SSID, 最多 13 个字符。

**u8 password[14+1]**

需要设置的 AP 模式下的密码, 最多 13 个字符。

**u8 enc**

需要设置的 AP 认证方式

=0, OPEN 开放式;

=1, WEP;

=2, WPA\_PSK;

=3, WPA2\_PSK;

=4, WPA\_WPA2\_PSK;

=其他, OPEN 开放式。

**u8 channel**

需要设置的 AP 模式下的信道。

**u8 saved**

是保存该配置

=0, 不保存到模组上的 FLASH 里, 下次启动时无效;

=1, 保存到模组上的 FLASH 里, 下次启动时依然有效。

说明: 调用本函数时, 如果选定了保存, 会对模块上的 FLASH 进行写操作。所以, 建议不要在单片机初始化阶段执行保存, 否则单片机每次启动都可能导致一次写 FLASH 操作, 这可能会影响 FLASH 的寿命。保存只需要执行一次就够了。

**u16\* status**

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

## 返回值

=1, 执行成功

=0, 出错了

#### 4.2.13 M8266WIFI\_SPI\_Query\_AP\_Param

```

/*****
 * M8266WIFI_SPI_Query_AP_Param
 * .Description:
 *   To query a parameter of the module AP via SPI
 * .Parameter(s)
 *   1. param_type : the param type to set, enum of AP_PARAM_TYPE
 *                   AP_PARAM_TYPE_SSID           = 0,
 *                   AP_PARAM_TYPE_PASSWORD        = 1,
 *                   AP_PARAM_TYPE_CHANNEL         = 2,
 *                   AP_PARAM_TYPE_AUTHMODE        = 3,
 *                   AP_PARAM_TYPE_SSID_HIDDEN     = 4,
 *                   AP_PARAM_TYPE_MAX_CONNECT     = 5,
 *                   AP_PARAM_TYPE_BEACON_INTERVAL = 6,
 *                   AP_PARAM_TYPE_IP_ADDR         = 7,
 *                   AP_PARAM_TYPE_GATEWAY_ADDR    = 8,
 *                   AP_PARAM_TYPE_NETMASK_ADDR    = 9,
 *   2. param       : pointer to the param value returned
 *   3. param_len   : pointer to length the param, unit in bytes
 *   4. status      : pointer to return errcode(LSB) and status(MSB) upon error
 *                   Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Query_AP_Param(AP_PARAM_TYPE param_type, u8* param, u8* param_len, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 接口查询模组热点 AP 模式下的 AP 参数。

#### 说明

- 1、适用于主机接口模式 2（只使用 SPI 接口不使用串口）下。

#### 函数原型

```

u8 M8266WIFI_SPI_Query_AP_Param(
    AP_PARAM_TYPE param_type,
    u8* param,
    u8* param_len,
    u16* status);

```

#### 参数

##### STA\_PARAM\_TYPE param\_type

这个枚举类型，用于指定查询的参数类型

```

typedef enum{
    AP_PARAM_TYPE_SSID           = 0,
    AP_PARAM_TYPE_PASSWORD        = 1,
    AP_PARAM_TYPE_CHANNEL         = 2,
    AP_PARAM_TYPE_AUTHMODE        = 3,
    AP_PARAM_TYPE_SSID_HIDDEN     = 4,
    AP_PARAM_TYPE_MAX_CONNECT     = 5,
    AP_PARAM_TYPE_BEACON_INTERVAL = 6,
    AP_PARAM_TYPE_IP_ADDR         = 7,
    AP_PARAM_TYPE_GATEWAY_ADDR    = 8,
    AP_PARAM_TYPE_NETMASK_ADDR    = 9,
}AP_PARAM_TYPE;

```

各个参数类型的说明如下

AP\_PARAM\_TYPE\_SSID: 返回当前热点的 SSID, 参数最长 27 字节

AP\_PARAM\_TYPE\_PASSWD: 返回当前热点的密码, 参数最长 27 字节

AP\_PARAM\_TYPE\_CHANNEL: 返回当前热点的频道

AP\_PARAM\_TYPE\_AUTHMODE: 返回当前热点的加密方式

=0, OPEN 开放式;

=1, WEP;

=2, WPA\_PSK;

=3, WPA2\_PSK;

=4, WPA\_WPA2\_PSK;

AP\_PARAM\_TYPE\_SSID\_HIDDEN: 返回当前热点是否隐藏

=0, 不隐藏

=1, 隐藏

AP\_PARAM\_TYPE\_MAX\_CONNECT: 返回当前热点支持的最大接入数

AP\_PARAM\_TYPE\_BEACON\_INTERVAL: 返回当前热点的 beacon 值

AP\_PARAM\_TYPE\_IP\_ADDR: 返回当前热点的 IP 地址

AP\_PARAM\_TYPE\_GATEWAY\_ADDR: 返回当前热点的网关地址

AP\_PARAM\_TYPE\_NETMASK\_ADDR: 返回当前热点的子网掩码

#### U8\* param

指针, 指向返回的参数

#### U8\* param\_len

指针, 指向返回的参数长度, 单位字节

#### u16\* status

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

#### 返回值

=1, 执行成功

=0, 出错了



#### 4.2.14 M8266WIFI\_SPI\_Config\_AP\_Param

```

/*****
 * M8266WIFI_SPI_Config_AP_Param
 * .Description:
 *   To config a parameter of the module AP via SPI
 * .Parameter(s)
 *   1. param_type : the param type to set, enum of AP_PARAM_TYPE
 *       AP_PARAM_TYPE_SSID           = 0,
 *       AP_PARAM_TYPE_PASSWORD       = 1,
 *       AP_PARAM_TYPE_CHANNEL        = 2,
 *       AP_PARAM_TYPE_AUTHMODE       = 3,
 *       AP_PARAM_TYPE_SSID_HIDDEN    = 4,
 *       AP_PARAM_TYPE_MAX_CONNECT    = 5,
 *       AP_PARAM_TYPE_BEACON_INTERVAL = 6,
 *       AP_PARAM_TYPE_IP_ADDR        = 7,
 *       AP_PARAM_TYPE_GATEWAY_ADDR   = 8,
 *       AP_PARAM_TYPE_NETMASK_ADDR   = 9,
 *   2. param      : pointer to the param value
 *   3. param_len   : length the param, unit in bytes
 *   4. saved       : to save the param into flash or not
 *       =0,        : not saved, i.e. after reboot setting will restore to previous
 *       =others,    : saved, i.e. after reboot, the saved setting will be loaded
 *                   : PLEASE DO NOT CALL IT EACH TIME OF BOOTUP WITH SAVED != 0
 *                   : OR, THE FLASH ON MODULE MIGHT GO TO FAILURE DUE TO LIFT CYCLE
 *                   : OF WRITE
 *   5. status      : pointer to return errcode(LSB) and status(MSB) upon error
 *                   : Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Config_AP_Param(AP_PARAM_TYPE param_type, u8* param, u8 param_len, u8 saved, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 接口配置模组热点 AP 模式下的单个 AP 参数。

#### 说明

- 1、适用于主机接口模式 2（只使用 SPI 接口不使用串口）下。

#### 函数原型

```

u8 M8266WIFI_SPI_Config_AP_Param((
    AP_PARAM_TYPE param_type,
    u8* param,
    u8 param_len,
    u8 saved,
    u16* status);

```

#### 参数

**STA\_PARAM\_TYPE param\_type**

这个枚举类型，用于指定配置的参数类型

```

typedef enum{
    AP_PARAM_TYPE_SSID           = 0,
    AP_PARAM_TYPE_PASSWORD       = 1,
    AP_PARAM_TYPE_CHANNEL        = 2,
    AP_PARAM_TYPE_AUTHMODE       = 3,
    AP_PARAM_TYPE_SSID_HIDDEN    = 4,
    AP_PARAM_TYPE_MAX_CONNECT    = 5,
    AP_PARAM_TYPE_BEACON_INTERVAL = 6,
    AP_PARAM_TYPE_IP_ADDR        = 7,
    AP_PARAM_TYPE_GATEWAY_ADDR   = 8,

```

```

        AP_PARAM_TYPE_NETMASK_ADDR          = 9
    }AP_PARAM_TYPE;

```

各个参数类型的说明如下

AP\_PARAM\_TYPE\_SSID: 设置当前热点的 SSID, 参数最长 27 字节

AP\_PARAM\_TYPE\_PASSWD: 设置当前热点的密码, 参数最长 27 字节

AP\_PARAM\_TYPE\_CHANNEL: 设置当前热点的频道

AP\_PARAM\_TYPE\_AUTHMODE: 设置当前热点的加密方式

=0, OPEN 开放式;

=1, WEB;

=2, WPA\_PSK;

=3, WPA2\_PSK;

=4, WPA\_WPA2\_PSK;

AP\_PARAM\_TYPE\_SSID\_HIDDEN: 设置当前热点是否掩藏

=0, 不隐藏

=1, 隐藏

AP\_PARAM\_TYPE\_MAX\_CONNECT: 设置当前热点支持的接入数, 最大 4

AP\_PARAM\_TYPE\_BEACON\_INTERVAL: 设置当前热点的 beacon 值

AP\_PARAM\_TYPE\_IP\_ADDR: 设置当前热点的 IP 地址

AP\_PARAM\_TYPE\_GATEWAY\_ADDR: 设置当前热点的网关地址

AP\_PARAM\_TYPE\_NETMASK\_ADDR: 设置当前热点的子网掩码

#### u8\* param

指针, 指向设置的参数内容

#### u8 param\_len

指针, 指向设置的参数长度, 单位字节

#### u8 saved

是否保存该配置

=0, 不保存到模组上的 FLASH 里, 下次启动时无效;

=1, 保存到模组上的 FLASH 里, 下次启动时依然有效。

说明:

(1) 调用本函数时, 如果选定了保存, 会对模块上的 FLASH 进行写操作。所以, 建议不要在单片机初始化阶段执行保存, 否则每次启动都可能导致一次写 FLASH 操作, 这可能会影响 FLASH 的寿命。保存只需要执行一次就够了。

(2) 如下参数时只能设置当前数值不能被保存, 每次模组复位后, 需重新设置:

AP\_PARAM\_TYPE\_MAX\_CONNECT、AP\_PARAM\_TYPE\_BEACON\_INTERVAL、

AP\_PARAM\_TYPE\_IP\_ADDR、AP\_PARAM\_TYPE\_GATEWAY\_ADDR、

AP\_PARAM\_TYPE\_NETMASK\_ADDR

#### u16\* status

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

返回值

=1, 执行成功

=0, 出错了

## 4.3 UDP 或 TCP 服务/链接的建立与查询控制

单片机网络通信, 一般都是建立在 UDP 或 TCP 包之上的通信。

这一部分 API, 支持主机在模组上建立、查询、以及控制 UDP 或 TCP 服务。

### 4.3.1 M8266WIFI\_SPI\_Setup\_Connection

```

/*****
 * M8266WIFI_SPI_Setup_Connection
 * .Description:
 *   To setup a UDP connection or an TCP client connection via SPI
 * .Parameter(s)
 *   1. tcp_udp      : connection type
 *                     =0, udp
 *                     =1, tcp client
 *                     =2, tcp server
 *   2. local_port    : local port specified
 *                     =0, M8266WIFI module will generate a random local port
 *                     !=0, the local_port specified here will be used
 *   3. remote_addr   : string of ip or dns address of the remote connection
 *   4. remote_port   : port of remote connection
 *   5. link_no       : the number of link used for multiple links. Max 4
 *   6. timeout_in_s : the max timeout connecting to a remote, unit in seconds
 *   7. status        : pointer to return errcode(LSB) and status(MSB) upon error
 *                     Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Setup_Connection(u8 tcp_udp, u16 local_port,
    char* remote_addr, u16 remote_port,
    u8 link_no, u8 timeout_in_s, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 接口在模组上建立一个 UDP 或 TCP 服务链接。

#### 说明

- 1、依照 TCP/IP 协议, 在建立一个 TCP 或 UDP 服务链接时, 由(本机地址, 本机端口)和(目标/远端地址、目标/远端端口)来唯一标识通信的双方。因此, 建立的每个链接时, 都有对应的 local\_addr, local\_port, remote\_addr, remote\_port。

模块可以获取自己的本机地址, 因此不需要向该函数传递这个参数, 本 API 函数也不包含该参数。当传递的参数 local\_port=0 时, 模块会产生一个随机的本机端口。

关于(本机地址, 本机端口)和(目标/远端地址、目标/远端端口)的使用规则和技巧, 可参看相关的 TCP IP 通信协议, 或参看常见问题列表中的相关解释。

- 2、目标地址 remote\_addr 可以是字符串形式传递的 IP 形式的地址, 也可以是字符串形式传递的域名地址。

例如可以是“192.168.1.100”或“202.89.233.100”, 也可以是“www.bing.com”。

如果是域名地址, 模块在建立链接之前, 会先自动做域名解析, 因此要求在模块的网段内有相应的域名服务器或路由, 否则模块将无法解析域名而在 status 指针的低字节返回 0x3C 错误, 提示域名解析超时。

- 2、如果建立的是 TCP 链接, 则模组可以是 TCP 客户端, 也可以是 TCP 服务器。
- 3、模组支持同时建立多个链接, 每个链接用 link\_no 来标识。
- 4、最多支持同时建立 4 个链接。在调用本函数时, 如果指定的 link\_no 下原来存在一个链接, 那么在建立新的链接之前, 会断开和删除该 link\_no 对应的当前链接。
- 5、适用于主机接口模式 2 (只使用 SPI 接口不使用串口) 下。

## 函数原型

**u8 M8266WIFI\_SPI\_Setup\_Connection (**

**u8 tcp\_udp,**  
**u16 local\_port,**  
**char\* remote\_addr,**  
**u16 remote\_port,**  
**u8 link\_no,**  
**u8 timeout\_in\_s,**  
**u16\* status)**

## 参数

**u8 tcp\_udp**

建立 tcp 或 udp 服务链接的类型

=0, UDP 服务

=1, TCP Client (客户端服务)

=2, TCP Server (服务器端服务)

**U16 local\_port**

本地端口。如果设定为 0, 则由模组会产生一个随机的本地端口。

**u8\* remote\_addr**

远端(目标)地址, 可以是 IP 形式, 也可以是域名形式。

**u16 remote\_port**

远端(目标)端口

**u8 link\_no**

建立服务链接所占用的链接通道, 数值为 0-3。

**u8 time\_out\_in\_s**

建立服务链接的超时时间, 单位是秒。

如果在 timeout\_in\_s 秒的时间内, 没有建立成功, 则因超时而返回。

这个时间是“超时”时间, 不是该函数的执行时间, 即, 如果在这个时间内连接成功了, 该函数也会提前成功返回。

这个超时, 一般只在作为 TCP 客户端连接时, 才有实际意义。

**u16\* status**

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

## 返回值

=1, 执行成功

=0, 出错了

### 4.3.2 M8266WIFI\_SPI\_Delete\_Connection

```
/* *****  
 * M8266WIFI_SPI_Delete_Connection  
 * .Description:  
 *   To delete/disconnect a client connection on M8266WIFI via SPI  
 * .Parameter(s)  
 *   1. link_no : the number of link to be deleted/disconnected  
 *   2. status  : pointer to the status upon failure  
 * .Return value:  
 *   =1, success  
 *   =0, has error(s)  
 * *****/  
u8 M8266WIFI_SPI_Delete_Connection(u8 link_no, u16* status);
```

#### 功能

- 1、这个函数用于通过 SPI 接口删除模组上的一个服务链接，删除的链接用链接号标识。如果该服务链接是一个 TCP 链接，同时也会断开该链接。

#### 说明

- 1、适用于主机接口模式 2（只使用 SPI 接口不使用串口）下。
- 2、直接对某个服务链接重新建立链接时，不必额外调用本函数，因为在重新建链前，如果该链接号存在一个实际的服务连接，建链函数会自动先删除该服务链接。

#### 函数原型

```
u8 M8266WIFI_SPI_Delete_Connection(  
    u8 link_no,  
    u16* status);
```

#### 参数

**u8 link\_no**

需要删除的服务链接通道的链接号，数值为 0-3。

**u16\* status**

执行出错时，返回的状态码的指针，方便故障诊断。

如果不需要返回状态码，可以使用 NULL。

#### 返回值

=1, 执行成功

=0, 出错了



### 4.3.3 M8266WIFI\_SPI\_Query\_Connection

```

/*****
 * M8266WIFI_SPI_Query_Connection
 * .Description:
 *   To Query the information about a connection on M8266WIFI via SPI
 *   including connection type, state, local port, remote_ip and port, etc
 * .Parameter(s)
 *   1. link_no      : the number of link to query
 *   2. connection_type : pointer to the connection type returned
 *                       = 0, udp
 *                       = 1, tcp
 *                       = others, invalid
 *                       Use NULL if you don't expect it returned
 *   3. connection_state: pointer to the connection state returned
 *                       = 0, invalid
 *                       = 1, wait
 *                       = 2, listen
 *                       = 3, connect
 *                       = 4, write
 *                       = 5, read
 *                       = 6, close
 *                       = others, invalid
 *                       Use NULL if you don't expect it returned
 *   4. local_port    : pointer to the local port returned
 *                       Use NULL if you don't expect it returned
 *   5. remote_ip     : pointer to the remote ip returned
 *                       Use NULL if you don't expect it returned
 *   6. remote_port   : pointer to the remote port returned
 *                       Use NULL if you don't expect it returned
 *   7. status        : pointer to return errcode(LSB) and status(MSB) upon error
 *                       Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Query_Connection(u8 link_no, u8* connection_type, u8* connection_state,
                                   u16* local_port, u8* remote_ip, u16* remote_port, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 查询某个服务链接的信息，包括是否依然链接等信息。

#### 说明

- 1、适用于主机接口模式 2（只使用 SPI 接口不使用串口）下。

#### 函数原型

```

u8 M8266WIFI_SPI_Query_Connection(
    u8 link_no,
    u8* connection_type,
    u8* connection_state,
    u16* local_port,
    u8* remote_ip,
    u16* remote_port,
    u16* status);

```

#### 参数

u8 link\_no



需要查询的服务链接通道的链接号, 数值为 0-3。

#### u8\* connection\_type

建立 tcp 或 udp 服务链接的类型, 返回值的意义如下:

=0, UDP 服务

=1, TCP 服务

如果不需要返回本信息, 可以使用 NULL。

#### u8\* connection\_state

建立 tcp 或 udp 服务链接当前的状态, 返回值的意义如下:

=0, 无效状态

=1, 等待状态

=2, TCP 服务器监控状态

=3, 连接状态

=4, 正在写发送缓冲区

=5, 正在读接收缓冲区

=6, 关闭状态

=其他, 无效

如果不需要返回本信息, 可以使用 NULL。

#### u16\* local\_port

本地端口。

如果不需要返回本信息, 可以使用 NULL。

#### u8\* remote\_ip

远端(目标) IP 地址。

如果不需要返回本信息, 可以使用 NULL。

#### u16 remote\_ip

远端(目标) 端口。

如果不需要返回本信息, 可以使用 NULL。

#### u16\* status

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

#### 返回值

=1, 执行成功

=0, 出错了

#### 4.3.4 M8266WIFI\_SPI\_Op\_Multicast\_Group

```

/*****
 * M8266WIFI_SPI_Op_Multicast_Group
 * .Description:
 *   To join or leave a multicast group AP via SPI
 *   The module should be in STA or STA+AP mode and have connected
 *   to a routers before call this API
 * .Parameter(s)
 *   1. join_not_leave : join or leave the multicast group
 *       =1, join
 *       =0, leave
 *   2. multicast_group_ip : the ip address of a multicast group
 *       e.g. "224.6.6.6".
 *   3. status : pointer to return errcode(LSB) and status(MSB) upon error
 *       Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Op_Multicast_Group(u8 join_not_leave, char multicast_group_ip[15+1], u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 接口来让模组加入/建立或离开一个组播群。

#### 说明

- 1、必须在模组处于 STA 或 STA+AP 模式，且连接上一个热点或路由之后，才可以调用本函数。
- 2、当加入一个组播群后，模块可以建立 UDP 连接进行组播发送或接收。组播或广播只适合 UDP 通信的场合，TCP 方式不存在组播或广播。
- 3、组播需要路由器支持，组播的地址必须是组播的保留地址，符合组播协议的规范。
- 4、适用于主机接口模式 2（只使用 SPI 接口不使用串口）下。

#### 函数原型

```

u8 M8266WIFI_SPI_Op_Multicast_Group (
    u8 join_not_leave,
    char multicast_group_ip[15+1],
    u16* status)

```

#### 参数

**u8 join\_not\_leave**

加入/建立 或 离开一个组播群。

= 1, 加入/建立 一个组播群

= 0, 离开 一个组播群

**char multicast\_group\_ip[15+1]**

组播的 ip 地址，必须符合组播地址规范。

**u16\* status**

执行出错时，返回的状态码的指针，方便故障诊断。

如果不需要返回状态码，可以使用 NULL。

#### 返回值

=1, 执行成功

=0, 出错了

### 4.3.5 M8266WIFI\_SPI\_Set\_TcpServer\_Auto\_Discon\_Timeout

```

/*****
 * M8266WIFI_SPI_Set_TcpServer_Auto_Discon_Timeout
 * .Description:
 *   To set up the tcp server's auto-disconnection timeout time
 *   when no communication from clients via SPI
 * .Parameter(s)
 *   1. link_no      : the number of link to setup
 *   2. timeout_in_s : the timeout in seconds the tcp server will auto
 *                       disconnect the connection when no communication
 *                       from clients
 *   2. status       : pointer to the status upon failure
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Set_TcpServer_Auto_Discon_Timeout(u8 link_no, u16 timeout_in_s, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 接口设置 TCP 服务器自动断开某个链接是超时时间。

#### 说明

- 1、一般地, 当 TCP 客户端正常断开一个 TCP 链接时, 会向 TCP 服务器发送一个通知, 因此 TCP 服务器同时会自动删除这个链接。但是当客户端异常断开链接时(例如突然重启), 如果没有向服务器端发送通知, 服务器端一般会通过监控到在一段时间内和客户端的通信情况, 如果在这段超时时间内没有收到来自 TCP 客户端的任何通信信息, TCP 服务器就会自动认为该 TCP 链接已经断开, 同时删除该套接字信息。
- 2、本函数仅限于在模组上建立的 TCP 服务器。
- 3、建立 TCP 服务器后, 如果不调用本函数设置 TCP 服务器超时时间, 则超时时间为初始缺省值 300 秒。
- 4、适用于主机接口模式 2(只使用 SPI 接口不使用串口)下。

#### 函数原型

```

u8 M8266WIFI_SPI_Set_TcpServer_Auto_Discon_Timeout(
    u8 link_no,
    u16 timeout_in_s,
    u16* status)

```

#### 参数

**u8 link\_no**

需要设置超时时间的 TCP 服务器链接通道的链接号, 数值为 0-3。

**u8 timeout\_in\_s**

执行本操作的超时时间, 单位秒。最大值 7200 秒。

**u16\* status**

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

#### 返回值

=1, 执行成功

=0, 出错了

### 4.3.6 M8266WIFI\_SPI\_Query\_Last\_Tcp\_Disconnect\_Cause

```

/*****
 * M8266WIFI_SPI_Query_Last_Tcp_Disconnect_Cause
 * .Description:
 *   To query the cause of last tcp disconnect
 * .Parameter(s)
 *   1. link_no       : the number of link to query
 *   2. discon_cause   : pointer to the last disconnect cause
 *                       -3, if sending timeout
 *                       -9, if reset request by remote peer reset
 *                       -20, if disconnected by remote peer reset
 *   3. status         : pointer to the status upon failure
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Query_Last_Tcp_Disconnect_Cause(u8 link_no, s8* disconnect_cause, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 查询 TCP 链接上次断开的原因。

#### 说明

- 1、适用于主机接口模式 2（只使用 SPI 接口不使用串口）下。

#### 函数原型

```

u8 M8266WIFI_SPI_Query_Last_Tcp_Disconnect_Cause(
    u8 link_no,
    s8* disconnect_cause,
    u16* status)

```

#### 参数

**u8 link\_no**

需要查询的服务链接通道的链接号，数值为 0-3。

**s8\* disconnect\_cause**

上次 TCP 链接关闭/断开的原因，返回值的意义如下：

- =-3, TCP 底层发送超时/TCP 底层重复发送次数超限
- =-9, 对等链接向本服务发送了 RST 请求
- =-20, 对等链接断开了，或者本服务主动断开了
- =其他值，请联系我司

**u16\* status**

执行出错时，返回的状态码的指针，方便故障诊断。

如果不需要返回状态码，可以使用 NULL。

#### 返回值

- =1, 执行成功
- =0, 出错了

### 4.3.7 M8266WIFI\_SPI\_Query\_Tcp\_Retrans\_Max

```

/*****
 * M8266WIFI_SPI_Query_Tcp_Retrans_Max
 * .Description:
 *   To query the tcp max retransmission
 * .Parameter(s)
 *   1. link_no      : the number of link to query
 *   2. max_retran    : pointer to max retransmission
 *   3. status        : pointer to the status upon failure
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Query_Tcp_Retrans_Max(u8 link_no, u8* max_retran, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 查询当前的 TCP 最大重发次数。

#### 说明

- 1、TCP 包底层发送时，当发出一个包后，如果在一段时间（本模块次参数为 500 毫秒）内没有收到接收方的 ACK 确认，则底层会自动重发。当重发次数超过某个数据时，TCP 链接将丢弃该包不再发送。这个数据称为 TCP 最大重发次数。
- 2、该值缺省为 3 次。
- 3、适用于主机接口模式 2（只使用 SPI 接口不使用串口）下。

#### 函数原型

```

u8 M8266WIFI_SPI_Query_Tcp_Retrans_Max(
    u8 link_no,
    u8* max_retran,
    u16* status)

```

#### 参数

**u8 link\_no**

需要查询的服务链接通道的链接号，数值为 0-3。

**u8\* max\_retran**

TCP 最大重发次数。

**u16\* status**

执行出错时，返回的状态码的指针，方便故障诊断。

如果不需要返回状态码，可以使用 NULL。

#### 返回值

=1，执行成功

=0，出错了

### 4.3.8 M8266WIFI\_SPI\_Config\_Tcp\_Retrans\_Max

```

/*****
 * M8266WIFI_SPI_Config_Tcp_Retrans_Max
 * .Description:
 *   To config the tcp max retransmission
 * .Parameter(s)
 *   1. link_no      : the number of link to query
 *   2. max_retran    : value of max retransmission, range 1-12
 *                     if <1, then use 1
 *                     if >12, then use 12
 *   3. status        : pointer to the status upon failure
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Config_Tcp_Retrans_Max(u8 link_no, u8 max_retran, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 设置当前的 TCP 最大重发次数。

#### 说明

- 1、TCP 包底层发送时,当发出一个包后,如果在一段时间(本模块次参数为 500 毫秒)内没有收到接收方的 ACK 确认,则底层会自动重发。当重发次数超过某个数据时, TCP 链接将丢弃该包不再发送。这个数据称为 TCP 最大重发次数。
- 2、该值设置范围为 1-12 次,初始缺省值为 3 次。
- 3、设置该数值需要专业综合分析。不恰当地设置该数值,可能导致通信系统不稳定或效率降低。因此建议不要轻易调用该 API 函数。
- 4、适用于主机接口模式 2(只使用 SPI 接口不使用串口)下。

#### 函数原型

```

u8 M8266WIFI_SPI_Config_Tcp_Retrans_Max(
    u8 link_no,
    u8 max_retran,
    u16* status)

```

#### 参数

**u8 link\_no**

需要查询的服务链接通道的链接号, 数值为 0-3。

**u8 max\_retran**

TCP 最大重发次数, 范围是 1- 12 次。

**u16\* status**

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

#### 返回值

=1, 执行成功

=0, 出错了



### 4.3.9 M8266WIFI\_SPI\_Query\_Tcp\_Mss

```

/*****
* M8266WIFI_SPI_Query_Tcp_Mss
* .Description:
*   To query the tcp mss value
* .Parameter(s)
*   1. link_no      : the number of link to query
*   2. tcp_mss      : pointer to max retransmission
*   3. status       : pointer to the status upon failure
* .Return value:
*   =1, success
*   =0, has error(s)
*****/
u8 M8266WIFI_SPI_Query_Tcp_Mss(u8 link_no, u16* tcp_mss, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 查询当前的 TCP 的 MSS 值。

#### 说明

- 1、该数值一般为 1460。
- 2、适用于主机接口模式 2（只使用 SPI 接口不使用串口）下。

#### 函数原型

```

u8 M8266WIFI_SPI_Query_Tcp_Mss(
    u8 link_no,
    u16* tcp_mss,
    u16* status)

```

#### 参数

**u8 link\_no**

需要查询的服务链接通道的链接号，数值为 0-3。

**U16\* tcp\_mss**

TCP MSS 值

**u16\* status**

执行出错时，返回的状态码的指针，方便故障诊断。

如果不需要返回状态码，可以使用 NULL。

#### 返回值

=1，执行成功

=0，出错了

#### 4.3.10 M8266WIFI\_SPI\_Query\_Tcp\_Window\_num

```
/*
 * M8266WIFI_SPI_Query_Tcp_Window_num
 * .Description:
 *   To query the tcp window num
 * .Parameter(s)
 *   1. link_no      : the number of link to query
 *   2. tcp_wnd_num  : number of tcp windows
 *   3. status       : pointer to the status upon failure
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 */
u8 M8266WIFI_SPI_Query_Tcp_Window_num(u8 link_no, u8* tcp_wnd_num, u16* status);
```

##### 功能

- 1、这个函数用于通过 SPI 查询当前的 TCP 的窗口数值。

##### 说明

- 1、适用于主机接口模式 2（只使用 SPI 接口不使用串口）下。

##### 函数原型

```
u8 M8266WIFI_SPI_Query_Tcp_Mss(
    u8 link_no,
    u16* tcp_mss,
    u16* status)
```

##### 参数

**u8 link\_no**

需要查询的服务链接通道的链接号，数值为 0-3。

**U16\* tcp\_mss**

TCP 窗口数值

**u16\* status**

执行出错时，返回的状态码的指针，方便故障诊断。

如果不需要返回状态码，可以使用 NULL。

##### 返回值

=1，执行成功

=0，出错了

### 4.3.11 M8266WIFI\_SPI\_Config\_Tcp\_Window\_num

```

/*****
* M8266WIFI_SPI_Config_Tcp_Window_num
* .Description:
*   To config the tcp max retransmission
* .Parameter(s)
*   1. link_no      : the number of link to query
*   2. tcp_wnd_num  : numbers of tcp windows, range 1-15
*                     if <1, then use 1
*                     if >15, then use 15
*   3. status       : pointer to the status upon failure
* .Return value:
*   =1, success
*   =0, has error(s)
*****/
u8 M8266WIFI_SPI_Config_Tcp_Window_num(u8 link_no, u8 tcp_wnd_num, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 设置当前的 TCP 窗口数。

#### 说明

- 1、该值设置范围为 1-15，初始缺省值为 2。
- 2、设置该数值需要专业综合分析。不恰当地设置该数值，可能导致通信系统不稳定或效率降低。因此建议不要轻易调用该 API 函数。
- 3、适用于主机接口模式 2（只使用 SPI 接口不使用串口）下。

#### 函数原型

```

u8 M8266WIFI_SPI_Config_Tcp_Window_num(
    u8 link_no,
    u8 tcp_wnd_num,
    u16* status)

```

#### 参数

**u8 link\_no**

需要查询的服务链接通道的链接号，数值为 0-3。

**u8 tcp\_wnd\_num**

TCP 窗口数，范围是 1-15，初始缺省值为 2。

**u16\* status**

执行出错时，返回的状态码的指针，方便故障诊断。

如果不需要返回状态码，可以使用 NULL。

#### 返回值

=1，执行成功

=0，出错了

### 4.3.12 M8266WIFI\_SPI\_STA\_Get\_HostIP\_by\_HostName

```

/*****
 * SPI_STA_Get_HostIP_by_HostName
 * .Description:
 *   To get the host ip_addr by hostName
 * .Parameter(s)
 *   1. hostIp      : the host ip_addr returned
 *   2. hostName     : the host name to get ip
 *   3. timeout_in_s: timeout in seconds
 *   3. status      : pointer to return errcode(LSB) and status(MSB) upon error
 *                   Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 SPI_STA_Get_HostIP_by_HostName(char* hostIp, char* hostName, u8 timeout_in_s, u16* status);

```

#### 功能

- 1、这个函数支持模块获取域名的 IP 地址，并通过 SPI 接口传递给单片机

#### 说明

- 1、需要模块可以访问网段内的域名服务。
- 2、适用于主机接口模式 2（只使用 SPI 接口不使用串口）下。

#### 函数原型

```

u8 M8266WIFI_SPI_STA_Get_HostIP_by_HostName(
    char* hostIp,
    char* hostName,
    u8 timeout_in_s,
    u16* status)

```

#### 参数

**char\* hostIp**

解析域名所得到的 IP 地址，例如“61.135.169.125”。

**char\* hostName**

需要解析的域名，例如“www.baidu.com”。

**u8 timeout\_in\_s**

执行本操作的超时时间，单位秒。

**u16\* status**

执行出错时，返回的状态码的指针，方便故障诊断。

如果不需要返回状态码，可以使用 NULL。

#### 返回值

- =1，执行成功
- =0，出错了

## 4.4 UDP 或 TCP 数据包的收发

在模组上成功地建立了 UDP 或 TCP 服务'链接'之后,接下来就是通过这些服务/链接,开始和远端(目标)节点进行数据通信交互了,单片机主机通过 SPI 口向模组写入的数据,会被转化为相应的 UDP 或 TCP 包发送给远端(目标)节点,而来自远端(目标)节点的 UDP 或 TCP 包数据,单片机主机可以通过 SPI 接口读取。

这一部分 API,支持主机通过模组,和远端(目标)节点之间进行数据交互。

### 4.4.1 M8266WIFI\_SPI\_Send\_Data

```

/*****
 * M8266WIFI_SPI_Send_Data
 * .Description:
 *   To send Data to WIFI via M8266 module SPI
 * .Parameters
 *   1. Data : the pointer to the Data buffer to be sent
 *   2. len : the length the Data buffer to be sent
 *   3. link_no: the wifi service link number sent to
 *   4. pointer to return errcode(LSB) and status(MSB) when error encountered
 *   use NULL if you don't expect errcode and status
 *   errcode:
 *     = 0x10: timeout when wait Module spi rxd Buffer ready
 *     = 0x11: timeout when wait wifi to send data
 *     = 0x12: Module Sending Buffer full
 *           Module Buffer full defined as
 *           - If total size of packets waiting to send via WIFI > 5*1024, or
 *           - If counts of packets waiting to send via WIFI > 8
 *     = 0x13: Wrong link_no used
 *     = 0x14: connection by link_no not present
 *     = 0x15: connection by link_no closed
 *     = 0x18: No clients connecting to this TCP server
 *     = 0x1F: Other errors
 * .Return value:
 *   Actually length that has been sent successfuully
 *****/
u16 M8266WIFI_SPI_Send_Data(u8 Data[], u16 Data_len, u8 link_no, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 向远端网络节点发送数据。单片机主机调用这个函数通过 SPI 接口向模组写入的数据,会被模组转化成为 TCP 或 UDP 包,发送到远端(目标)。

#### 说明

- 1、考虑到网络的实际情况 MTU 参数,以及单片机和模块实际的缓存大小等因素,建议一次发送字节数量不要超过 MSS=1460 个字节左右。对于需要发送更大块数据,建议分成多片发送。否则可能会出现拆包粘包等现象。尤其是在一些高速实时性通信的场合,拆包形成的小短包可能造成造成发送的效率降低;同时,因为拆包可能阵发包的数量连贯倍增,有可能造成接收方来处理不及出现 ACK 丢弃而造成发送方因为重发次数增多而降低通信效率,甚至可能因为重发次数出现越界而出现套接字状态复位。所以,对于连续高速通信,一般建议 TCP 包长度不要超过 1460 甚至更少以保持一定富裕量。
- 2、本函数为半阻塞式。即:调用该函数后会等待,直到主机将全部需要发送的数据通过 SPI 传给 ALK8266WIFI® 模组的缓存,然后返回。不会等到 WIFI 模组将该数据

发送完毕才返回。

- 3、该函数的返回值只是成功地将数据通过 SPI 接口从主机传递给 WIFI 模组的数值,不是通过 WIFI 实际发送的数值。但是因为模组处理的可靠性,实测结果发现,通过 SPI 写入模组的数据,会通过 WIFI 模组不丢包且及时地被发出。

**所以,调用该函数后,需要判断实际写入 WIFI 模块的字节数。**

- 4、还有一个类似的发送 API 函数 [4.4.2 M8266WIFI SPI Send Data Block](#),是对本 API 函数进行二次封装,发送更简单,但是效率和灵活性不如这个 API 函数。这个 API 函数可类比 linux 套接字编程中的 raw send (底层原始发送)。

## 函数原型

**u16 M8266WIFI\_SPI\_Send\_Data(**

**u8 Data[],**

**u16 Data\_len,**

**u8 link\_no,**

**u16\* status);**

## 参数

**u8 Data[]**

需要写入 WIFI 模组被发出的数据缓存。

**u16 Data\_len**

需要写入 WIFI 模组被发出的数据长度。

**u8 link\_no**

服务链接号,写入的数据需要通过该服务链接通道发送出去。

**u16\* status**

执行出错时,返回的状态码的指针,方便故障诊断。

如果不需要返回状态码,可以使用 NULL。

## 返回值

实际发送出去的数据长度。



#### 4.4.2 M8266WIFI\_SPI\_Send\_Data\_Block

```

/*****
 * M8266WIFI_SPI_Send_Data_Block
 * .Description:
 *   To send Data block to WIFI via M8266 module SPI
 * .Parameters
 *   1. Data           : the pointer to the Data buffer to be sent
 *   2. Data_len       : the length the Data buffer to be sent
 *   3. link_no        : the wifi service link number sent to
 *   4. tcp_packet_size : tcp packet size
 *                       smaller than 1460, i.e. 1024 recommended
 *                       not exceeding MSS (normally 1460) recommended
 *   5. pointer to return errcode (LSB) and status (MSB) when error encountered
 *       use NULL if you don't expect errcode and status
 *       errcode:
 *         = 0x13: Wrong link_no used
 *         = 0x14: connection by link_no not present
 *         = 0x15: connection by link_no closed
 *         = 0x18: No clients connecting to this TCP server
 *         = 0x1E: too many errors encountered during sending can not fixed
 *         = 0x1F: Other errors
 * .Return value:
 *   Actually length that has been sent successfully
 *****/
u32 M8266WIFI_SPI_Send_Data_Block(u8 Data[], u32 Data_len, u16 tcp_packet_size,
                                   u8 link_no, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 向远端网络节点发送数据，它会完整的发送一个数据块。单片机主机调用这个函数通过 SPI 接口向模组写入的数据，会被模组转化成为一个或多个 TCP 或 UDP 包，发送到远端（目标）。

#### 说明

- 1、这个函数是 [4.4.1 M8266WIFI\\_SPI\\_Send\\_Data](#) 函数的二次封装，方便简化调用，但是效率和灵活性不如 [4.4.1 M8266WIFI\\_SPI\\_Send\\_Data](#) 函数。
- 2、这个函数会将需要发送的数据块以 TCP 包的形式持续发下去，直到发送完毕为止，除非出现了断链、严重阻塞等异常。所以，调用起来会更简单。
- 3、该函数的返回值只是成功地将数据通过 SPI 接口从主机传递给 WIFI 模组的数值。
- 4、正确地制定底层每个 TCP 包的大小，有助于提高发送的效率，尤其是在高速连续通信的场合。强烈建议建议参数 tcp\_packet\_size 不要超过 TCP MSS 数值，如果不知道这个数值，建议使用 1024。

#### 函数原型

```

u32 M8266WIFI_SPI_Send_Data_Block(
    u8 Data[],
    u32 Data_len,
    u16 tcp_packet_size,
    u8 link_no,
    u16* status);

```

#### 参数

**u8 Data[]**

需要写入 WIFI 模组被发出的数据块缓存。

**u32 Data\_len**

需要写入 WIFI 模组被发出的数据总长度。

#### u16 tcp\_packet\_size

底层单个 TCP 包的数据域长度, 强烈建议不要超过 TCP MSS 数值。如果不清楚, 建议使用 1024 或 1460。

#### u8 link\_no

服务链接号, 写入的数据需要通过该服务链接通道发送出去。

#### u16\* status

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

#### 返回值

实际发送出去的数据长度。

### 4.4.3 M8266WIFI\_SPI\_Send\_Udp\_Data

```

/*****
 * M8266WIFI_SPI_Send_Udp_Data
 * .Description:
 *   .To send Udp Data to WIFI via M8266 module SPI especially suitable for
 *   those UDP transmission requiring frequently changing destination
 *   .If the UDP transmission does not need update dest, please use
 *   M8266WIFI_SPI_Send_Data above for better efficiency
 * .Parameters
 *   1. Data : the pointer to the Data buffer to be sent
 *   2. len : the length the Data buffer to be sent
 *   3. link_no: the wifi service link number sent to
 *   4. udp_dest_addr: string of ip or dns address of the remote connection
 *   5. udp_dest_port: port of remote connection
 *   6. pointer to return errcode(LSB) and status(MSB) when error encountered
 *   use NULL if you don't expect errcode and status
 *   errcode:
 *   = 0x10: timeout when wait Module spi rxd Buffer ready
 *   = 0x11: timeout when wait wifi to send data
 *   = 0x12: Module Sending Buffer full
 *   Module Buffer full defined as
 *   - If total size of packets waiting to send via WIFI > 5*1024, or
 *   - If counts of packets waiting to send via WIFI > 8
 *   = 0x13: Wrong link_no used
 *   = 0x14: connection by link_no not present
 *   = 0x15: connection by link_no closed
 *   = 0x19: this link is not a UDP service
 *   = 0x1A: try to point to a new udp dest but failed
 *   = 0x1F: Other errors
 * .Return value:
 *   Actually length that has been sent successfully
 *****/
u16 M8266WIFI_SPI_Send_Udp_Data(u8 Data[], u16 Data_len, u8 link_no,
                                char* udp_dest_addr, u16 udp_dest_port, u16* status);

```

#### 功能

- 1、因为 UDP 链接时“假链接”, 可以随时更改发送的目标节点。为了适应某些场合实现在多个目标节点直接自由切换, 扩展了本 API 发送函数, 直接指定目标节点, 而不需要每次发送前都重新建立一次链接来切换目标节点。

#### 说明

- 1、此 API 函数仅限于 UDP 发送。

## 函数原型

```
u16 M8266WIFI_SPI_Send_Udp_Data(  
    u8 Data[],  
    u16 Data_len,  
    u8 link_no,  
    char* udp_dest_addr,  
    u16 udp_dest_port,  
    u16* status);
```

## 参数

### u8 Data[]

需要写入 WIFI 模组被发出的数据块缓存。

### U16 Data\_len

需要写入 WIFI 模组被发出的数据总长度。

### u8 link\_no

服务链接号, 写入的数据需要通过该服务链接通道发送出去。

### u8\* udp\_dest\_addr

远端(目标)地址, 可以是 IP 形式, 也可以是域名形式。

### u16 udp\_dest\_port

远端(目标)端口

### u16\* status

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

## 返回值

实际发送出去的数据长度。

#### 4.4.4 M8266WIFI\_SPI\_Has\_DataReceived

```
/* *****  
 * M8266WIFI_SPI_Has_DataReceived  
 * .Description:  
 *   To check whether the M8266WIFI module has received data awaiting master  
 *   to fetch away  
 * .Parameters  
 *   None  
 * .Return value:  
 *   1, if the M8266WIFI module has received data from wifi  
 *   0, if the M8266WIFI module has not received data from wifi  
 * *****  
u8 M8266WIFI_SPI_Has_DataReceived(void);
```

##### 功能

- 1、这个函数用于查询模式下，判断模组是否接收到来自网络的数据。

##### 说明

- 1、如果当前有一个链接上接收到了数据，该函数就会立刻返回 1；否则，会立刻返回 0。

##### 函数原型

**U8 M8266WIFI\_SPI\_Has\_DataReceived(void);**

##### 参数

无

##### 返回值

- =1, WIFI 收到了数据
- =0, WIFI 没有收到数据

#### 4.4.5 M8266WIFI\_SPI\_RecvData

```

/*****
 * M8266WIFI_SPI_RecvData
 * .Description:
 *   To receive the wifi data from M8266WIFI
 * .Parameters
 *   1. Data          - the buffer to contained the received Data
 *   2. max_len        - the max length of Data to fetech
 *   3. max_wait_in_ms - the max timeout to wait for the Data
 *   4. link_no        - pointer to return the link_no that current wifi Data
 *                       come from. use NULL if you don't expect it returned
 *   5. status         - pointer to return errcode(LSB) and status(MSB)
 *                       when error encountered. Use NULL if you don't expect
 *                       them returned
 *   errcode:
 *       = 0x20: timeout when wait module has received data via WIFI
 *       = 0x22: no date in Module wifi receive buffer
 *       = 0x23: Read data from the left of last packets in Module wifi
 *               receive buffer
 *       = 0x24: The packet in the Module wifi receive buffer is larger
 *               in size than the max_len specified here. Only part of
 *               the packet received
 *       = 0x2F: Other errors
 * .Return value:
 *   - the size of larger packet first in the Module wifi receive buffer
 *   if errcode = 0x24
 *   - the actual length of wifi packet received successfully
 *   if others
 *****/
u16 M8266WIFI_SPI_RecvData(u8 Data[], u16 max_len, uint16_t max_wait_in_ms,
                          u8* link_no, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 接口读取模组从网络上所接收到的 TCP/UDP 包。

#### 说明

- 1、WIFI 模组在收到 TCP/UDP 包后, 会先以 TCP/UDP 包为单元, 临时存储在模组上的缓冲区里, 等待主机读取。主机可以通过调用这个函数, 来读取模组所接收到的 TCP/UDP 包。
- 2、本函数是阻塞式函数。当成功地完成了一个 TCP/UDP 包的读取、或者读取的数据长度达到了 max\_len、或者等待 TCP/UDP 包的时间超过 max\_wait\_in\_ms 时, 这个函数就会返回。
- 3、所读取的 TCP/UDP 包的数据域, 会保存在接收缓冲区 Data[]中; 函数的返回值, 为实际所读取的数据长度; 在 link\_no 中会返回该 TCP/UDP 包所对应的链接号。
- 4、如果 WIFI 模组当前所收到的 TCP/UDP 包的数据长度小于等于 max\_len 时, 该函数在读取了完毕这个 TCP/UDP 包后会立刻返回, 此时在 status 里会返回 0x0000。
- 5、如果 WIFI 模组当前所收到的 TCP/UDP 包的数据长度大于 max\_len, 该函数在读取了这个 TCP/UDP 包数据域的前 max\_len 个字节后立刻返回, 此时, 在 status 的低字节里会返回 0x24, 表示当前这个 TCP/UDP 包还没有读取完毕, 需要继续调用这个函数来接收剩下的字节。在接下来调用这个函数时, 在 status 的低字节会返回 0x23, 表示此次读取的数据接续前面读取的数据, 组成一个完整的 TCP/UDP 包。所以, max\_len 一般用来指示接收数据缓冲区 data[]的大小, 以避免在接收过程中, 出现数组溢出。
- 6、当等待接收的时间超过 max\_wait\_in\_ms 后, 该函数会返回, 而不再继续等待接收,

此时 **status** 返回的低字节为 **0x20**, 表示等待超时。

## 函数原型

```
u16 M8266WIFI_SPI_RecvData(  
    u8 Data[],  
    u16 max_len,  
    uint16_t max_wait_in_ms,  
    u8* link_no,  
    u16* status);
```

## 参数

**u8 Data[]**

用于接收数据的接收缓存区。

**u16 max\_len**

接收数据的长度上限。

**u16 max\_wait\_in\_ms**

等待接收完毕的最大超时, 单位 ms。

**u8\* link\_no**

服务链接号, 返回当前读取数据所来源的服务链接号。

**u16\* status**

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 **NULL**。

## 返回值

通过 SPI 读取得到的数据长度。



#### 4.4.6 M8266WIFI\_SPI\_RecvData\_Ex

```

/*****
 * M8266WIFI_SPI_RecvData_ex
 * .Description:
 *   To receive the wifi data from M8266WIFI, extended
 *   Compared with M8266WIFI_SPI_RecvData(), this function also return the
 *   source remote_ip and remote_port meanwhile
 * .Parameters
 *   1. Data          - the buffer to contained the received Data
 *   2. max_len       - the max length of Data to fetech
 *   3. max_wait_in_ms - the max timeout to wait for the Data
 *   4. link_no       - pointer to return the link_no that current wifi Data
 *                     come from. use NULL if you don't expect it returned
 *   5. remote_ip     - array[4] to return the remote_ip that current wifi Data
 *                     come from. use NULL if you don't expect it returned
 *                     e.g. if remote ip is "192.168.4.2", then remote_ip will
 *                     return with remote_ip[0]=192, remote_ip[1]=168,
 *                     remote_ip[2]=4, and remote_ip[3]=2
 *   6. remote_port   - pointer to return the remote_port that current wifi Data
 *                     come from. use NULL if you don't expect it returned
 *   7. status        - pointer to return errcode(LSB) and status(MSB)
 *                     when error encountered. Use NULL if you don't expect
 *                     them returned
 *   errcode:
 *       = 0x20: timeout when wait module has received data via WIFI
 *       = 0x22: no date in Module wifi receive buffer
 *       = 0x23: Read data from the left of last packets in Module wifi
 *               receive buffer
 *       = 0x24: The packet in the Module wifi receive buffer is larger
 *               in size than the max_len specified here. Only part of
 *               the packet received
 *       = 0x2F: Other errors
 * .Return value:
 *   - the size of larger packet first in the Module wifi receive buffer
 *     if errcode = 0x24
 *   - the actual length of wifi packet received successfully
 *     if others
 *****/
u16 M8266WIFI_SPI_RecvData_ex(u8 Data[], u16 max_len, uint16_t max_wait_in_ms,
                              u8* link_no, u8 remote_ip[4], u16* remote_port, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 接口读取模组从网络上所接收到的 TCP/UDP 包。这个函数是 [4.4.5 M8266WIFI\\_SPI\\_RecvData](#) 的扩展函数，其功能和用法，与 [4.4.5 M8266WIFI\\_SPI\\_RecvData](#) 完全一样，只是同时显性地返回了发送当前包的远端地址和端口，特别适合 UDP 一对多通信的场合。

#### 说明

同于 [4.4.5 M8266WIFI\\_SPI\\_RecvData](#)。

#### 函数原型

```

u16 M8266WIFI_SPI_RecvData_ex(
    u8 Data[],
    u16 max_len,
    uint16_t max_wait_in_ms,
    u8* link_no,
    u8 remote_ip[4],
    u16* remote_port,
    u16* status);

```

#### 参数

**u8 Data[]**

用于接收数据的接收缓存区。

**u16 max\_len**

接收数据的长度上限。

**u16 max\_wait\_in\_ms**

等待接收完毕的最大超时, 单位 ms。

**u8\* link\_no**

服务链接号, 返回当前读取数据所来源的服务链接号。

**u8 remote\_ip[4]**

发送本数据包的远端节点的 IP 地址。

**u8\* link\_no**

发送本数据包的远端节点的端口。

**u16\* status**

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

**返回值**

通过 SPI 读取得到的数据长度。

## 4.5 智能配网

对 WIFI 模组的配网方式的支持,是模组功能完备性的一个基本属性。智能配网是(1)直接配网和(2)通过 WEB 网页配网中之外,也比较流行的配网方式。但是智能配网在其原理上有相应的局限性,详情可参看《常见的 WIFI 模组配网方式简介与对比暨 ANYLINKIN 8266WIFI 模组配网方式和操作说明》。当不适合通过直接配网的方式进行配网时,Anylinkin! 推荐通过简单明了且约束条件最少的 WEB 配网方式替代各种智能配网方式对模组进行配网。

这一部分 API,支持主机通过智能配网的方式对模组进行配网。

### 4.5.1 M8266WIFI\_SPI\_Module\_SmartConfig

```

/*****
 * u8 M8266WIFI_SPI_Module_SmartConfig
 * .Description:
 *   To perform an SmartConfig procedure to wifi module via SPI
 * .Parameter(s)
 *   1. timeout_in_s : timeout for an smartconfig procedure
 *   2. smartconfig_type : pointer to smartconfig type returned
 *       use NULL if you don't expect this param returned
 *       = 0, if type is ESPTOUCH
 *       = 1, if type is AIRKISS
 *       =-1, if invalid or unknown
 *   3. smartconfig_phone_ip :
 *       - the ip of the phone or smart device
 *       - that runs smartconfig app to broadcast ssid/password
 *       - e.g. "192.168.43.21"
 *       - use NULL if you don't expect this param returned
 *       - Airkiss does not response ip addresss of smart devices,
 *       - so smartconfig_phone_ip[0] will be 0 if airkiss
 *   4. status : pointer to return errcode upon error
 *       Use NULL if you don't expect them returned
 *       == 0x0000 success
 *       != 0x0000 failed
 *           (1) failed to start smartconfig if LSB(8) = 0x68
 *           (2) failed during smartconfig if LSB(8) = 0x6A
 *               - failed to find channel if MSB(8) = 0x00
 *               - failed to get ssid/password if MSB(8) = 0x01
 *               - failed to connect the ap/routers if MSB(8) = 0x02/03
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Module_SmartConfig(u8 timeout_in_s,
                                     u8* smartconfig_type, char smartconfig_phone_ip[15+1], u16* status)

```

#### 功能

- 1、这个函数用于通过 SPI 接口启动模组进入 SmartConfig 智能配网模式并监控模组的整个智能配网的全过程。

#### 说明

- 1、模块可自适应支持两种智能配网模式: 乐鑫 smartlink 和微信 airkiss。  
“自适应”指的是,调用了这个 API 函数将模块引入到智能配网的接收模式,无论此时 APP 采用的是乐鑫 smartlink 或微信 airkiss 格式,模块会自动适应完成智能配网。

- 2、调用本函数后, 通过支持乐鑫 smartlink 的 APP (乐鑫官网有可执行 APP 以及开发包), 或手机微信 Airkiss 可以向模组传递路由器的 SSID 和密码, 并自动接入路由器。一旦接入成功, SSID 和密码会被保存在模组的 FLASH 里, 并自动退出智能配网模式, 同时模组会恢复为之前的 STA 或 AP 或 STA+AP 模式。
- 3、适用于主机接口模式 2 (只使用 SPI 接口不使用串口) 下。
- 4、更多详情, 请参看《[常见的 WIFI 模组配网方式简介与对比暨 ANYLINKIN 8266WIFI 模组配网方式和操作说明](#)》。

## 原型

```
u8 M8266WIFI_SPI_Module_SmartConfig(  
    u8 timeout_in_s,  
    u8* smartconfig_type,  
    char smartconfig_phone_ip[15+1],  
    u16* status)
```

## 参数

**u8 timeout\_in\_s**

执行智能配网过程的超时时间。如果超过了这个时间智能配网尚未完成, 这个函数将返回错误, 并在 status 里返回错误原因。

**u8\* smartconfig\_type**

执行智能配网的过程中, 模组可以判断智能配网的方式, 并在这个参数中返回当前判断的智能配网的方式。这个指针返回的数值有如下含义:

= 0, ESPTOUCH

= 1, AIRKISS

= 其他值, 无效

**char smartconfig\_phone\_ip[15+1]**

智能配网成功后, 这个字符串里会返回用来配网的智能设备如手机的 IP 地址, 例如“192.168.43.21”。但是值得注意的是, 微信 AIRKISS 方式一般不会返回智能设备的 IP 地址, 所以当配网方式为 AIRKISS 时这里的返回值会是 smartconfig\_phone\_ip[0]=0

**u16\* status**

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

这个指针的返回值具备以下含义:

= 0x0068, 启动 smartconfig 失败

= 0x006A, 启动 smartconfig 成功, 但是搜索信道失败。

= 0x016A, 启动 smartconfig 成功, 但是没有捕获到 SSID 和密码, 例如并未启动智能设备 (如手机) 上的智能配网 APP 来发送 SSID 和密码。

= 0x026A, 启动 smartconfig 成功, 捕获到了 SSID 和密码, 但是配网失败, 例如密码错误等。

= 0x036A, 同 0x026A, 例如路由器热点等不分配 IP 地址等。

= 其他值, SPI 接口通信异常, 可联系我们进一步诊断。

## 返回值

=1, 执行成功

=0, 出错了

#### 4.5.1 M8266WIFI\_SPI\_StartSmartConfig

```

/*****
 * M8266WIFI_SPI_StartSmartConfig
 * .Description:
 *   To start/stop smart config of M8266WIFI via SPI
 *   After perform a start operation, the M8266WIFI will be set in STA mode
 * .Parameter(s)
 *   1. en      : to start or stop
 *                =0, to stop
 *                others, to start
 *   2. status  : pointer to return errcode(LSB) and status(MSB) upon error
 *                Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_StartSmartConfig(u8 en, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 接口启动模组进入 SmartConfig 智能配网模式。

#### 说明

- 1、调用本 API 函数后, 会将模块引入到智能配网的接收模式。
- 2、模块可自适应支持两种智能配网模式: 乐鑫 smartlink 和微信 airkiss。

“自适应”的意思是, 只要调用了这个 API 函数将模块引入到智能配网的接收模式, 无论此时 APP 采用的是乐鑫 smartlink 或微信 airkiss 格式, 模块会自动适应完成智能配网。

- 3、调用本函数后, 通过支持乐鑫 smartlink 的 APP (乐鑫官网有可执行 APP 以及开发包), 或手机微信 Airkiss 可以向模组传递路由器的 SSID 和密码, 并自动接入路由器。一旦接入成功, SSID 和密码会被保存在模组的 FLASH 里, 并自动退出智能配网模式, 同时模组会被设置成 STA 模式。

- 3、适用于主机接口模式 2 (只使用 SPI 接口不使用串口) 下。

- 4、更多详情, 请参看 [《常见的 WIFI 模组配网方式简介与对比暨 ANYLINKIN 8266WIFI 模组配网方式和操作说明》](#)。

#### 原型

```

u8 M8266WIFI_SPI_StartSmartConfig(
    u8 en,
    u16* status)

```

#### 参数

**u8 en**

启动/退出模组 SmartConfig 智能配网模式。

=1, 启动模组 SmartConfig 智能配网模式。



=0, 退出模组 SmartConfig 智能配网模式。

#### u16\* status

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

#### 返回值

=1, 执行成功

=0, 出错了

### 4.5.2 M8266WIFI\_SPI\_StartWpsConfig

```

/*****
 * M8266WIFI_SPI_StartWpsConfig
 * .Description:
 *   To start/stop wps config of M8266WIFI via SPI
 *   After perform a start operation, the M8266WIFI will be set in STA mode
 * .Parameter(s)
 *   1. en      : to start or stop
 *               =0, to stop
 *               others, to start
 *   2. status  : pointer to return errcode(LSB) and status(MSB) upon error
 *               Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_StartWpsConfig(u8 en, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 接口启动模组进入 WPS 配网模式。

#### 说明

- 1、调用本函数后, 模组会被设置成 STA 模式。此时, 通过路由器的 WPS 功能向模组传递路由器的 SSID 和密码, 并自动接入路由器。一旦接入成功, SSID 和密码会被保存在模组的 FLASH 里, 并自动退出 WPS 配网模式。
- 2、适用于主机接口模式 2 (只使用 SPI 接口不使用串口) 下。
- 3、Anylinkin!建议大家不要再使用这种不安全也不再被市场所支持的配网方式。
- 4、更多详情, 请参看 [《常见的 WIFI 模组配网方式简介与对比暨 ANYLINKIN 8266WIFI 模组配网方式和操作说明》](#)。

#### 原型

```

u8 M8266WIFI_SPI_StartWpsConfig(
    u8 en,
    u16* status)

```

#### 参数

##### u8 en

启动/退出模组 WPS 配网模式。

=1, 启动模组 WPS 配网模式。

=0, 退出模组 WPS 配网模式。

##### u16\* status

执行出错时, 返回的状态码的指针, 方便故障诊断。



如果不需要返回状态码, 可以使用 NULL。

#### 返回值

=1, 执行成功

=0, 出错了

## 4.6 模组上的 WEB 服务器

Anylinkin®系列 WIFI 模组支持板载的内嵌 WEB 服务器功能, 来支持 WEB 网页配网或对模组资源的配置和查询。

这一部分 API, 支持单片机主机对内嵌 WEB 进行开启、关闭或设置。

### 4.6.1 SPI\_Set\_Web\_Server

```

/*****
 * SPI_Set_Web_Server
 * .Description:
 *   To set the web server via SPI
 * .Parameter(s)
 *   1. open_not_shutdown : to start or shutdown the local web server
 *       =0,               to shutdown the webserver if it is running
 *       =others,         to (re-)start the webserver
 *                       the previous running web server if will be shutdown
 *                       before re-open. Therefore, you don't have to shutdown
 *                       it if previously running
 *   2. server_port : the port used for web_server
 *       - Ignored when open_not_shutdown==0.
 *       - When open_not_shutdown !=0, and server_port=0, then default
 *         port = 80 will be used
 *   3. status : pointer to return errcode(LSB) and status(MSB) upon error
 *       Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 SPI_Set_Web_Server(u8 open_not_shutdown, u16 server_port, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 接口配置模组内嵌的 WEB 服务器, 包括关闭内嵌 WEB 服务器、打开内嵌 WEB 服务器、以及设置内嵌 WEB 服务器的端口号。

#### 说明

- 1、可以通过关闭或修改 WEB 服务器的端口, 增强 WEB 访问的安全性。
- 2、适用于主机接口模式 2 (只使用 SPI 接口不使用串口) 下。

#### 原型

```

u8 M8266WIFI_SPI_Set_Web_Server(
    u8 open_not_shutdown,
    u16 server_port,
    u16* status)

```

#### 参数

##### u8 open\_not\_shutdown

启动/关闭模组上的内嵌 WEB 服务器。

=1, 启动模组上的内嵌 WEB 服务器。

=0, 关闭模组上的内嵌 WEB 服务器。

##### u8 server\_port

启动模组上的内嵌 WEB 服务器时, 指定的服务器端口号。

关闭模组上的内嵌 WEB 服务器时, 该参数无效, 可随便填充。

#### **u16\* status**

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

#### **返回值**

=1, 执行成功

=0, 出错了

## 4.7 低功耗

这一部分 API, 支持主机通过模组的发射功率进行调节和设置模组进入休眠模式, 来支持模组的低功耗应用。

### 4.7.1 M8266WIFI\_SPI\_Set\_Tx\_Max\_Power

```

/*****
 * M8266WIFI_SPI_Set_Tx_Max_Power
 * .Description:
 *   To set the RF Tx Max Power via SPI comannd
 * .Parameter(s)
 *   1. tx_max_power : the max tx power in 0.25 dBm.
 *                      range = 0~82, i.e. 0~20.5dBm, or 1mW~112mW
 *                      tx_max_power  dBm      P/mW
 *                      0              0        1.0
 *                      4              1        1.3
 *                      12             3        2.0
 *                      28             7        5.0
 *                      40             10       10.0
 *                      52             13       20.0
 *                      68             17       50.0
 *                      80             20      100.0
 *                      82             20.5    112.20
 *   upon bootup default, tx_max_power = 82, i.e. 20.5dBm or 112.20mW
 *   2. status : pointer to return errcode(LSB) and status(MSB) upon error
 *              Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Set_Tx_Max_Power(u8 tx_max_power, u16 *status);
    
```

#### 功能

- 1、这个函数用于通过 SPI 接口配置模组 WIFI 发射时的最大功率。

#### 说明

- 1、当传输距离不远时, 可以通过调用这个函数调整发射的最大功率, 以节省发射功耗。
- 2、提醒: 降低最大发射功率, 会降低接收方所感应到的信号强度, 降低最远通信距离。
- 3、适用于主机接口模式 2 (只使用 SPI 接口不使用串口) 下。

#### 原型

```

u8 M8266WIFI_SPI_Set_Tx_Max_Power (
    u8 tx_max_power,
    u16* status)
    
```

#### 参数

**u8 tx\_max\_power**

需要设置的最大发射功率, 单位是 dBm, 取值范围是 0~82, i.e. 0~20.5dBm, 对应 1mW~112mW

tx_max_power	发射功率 dBm	发射功率 mW
0	0	1.0

4	1	1.3
12	3	2.0
28	7	5.0
40	10	10.0
52	13	20.0
68	17	50.0
80	20	100.0
82	20.5	112.20

启动后的初始缺省值为: tx\_max\_power = 82, 即对应 20.5dBm 或 112.20mW。

#### u16\* status

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

#### 返回值

=1, 执行成功

=0, 出错了

### 4.7.2 M8266WIFI\_SPI\_Sleep\_Module

```

/*****
 * M8266WIFI_SPI_Sleep_Module
 * .Description:
 *   To bring the M8266WIFI module into sleep mode via SPI comannnd
 * .Parameter(s)
 *   1. sleep_type : the type of sleep
 *     = 0       : reserved
 *     = 1       : reserved
 *     = 2       : reserved
 *     = 3       : deep sleep
 *     = others  : reserved
 *   2. time_to_wakeup_in_ms : time to wakeup from sleep, unit in ms
 *     - max value 4294967 ms (about 1.19hour).
 *     - if a value exceed this provided, then 4294967ms will be used
 *     - use 0 if expect to sleep for ever without automatic wakeup
 *   2. status : pointer to return errcode(LSB) and status(MSB) upon error
 *     Use NULL if you don't expect them returned
 * Note:
 *   1. The nCS should be pulled low before exit sleep and release after bootup
 *     from sleep if reset_type=3 in order for a normal bootup from extern flash
 *   2. after a call of this function, normally the module should be re init
 *     via functions such as M8266WIFI_Module_Init_Via_SPI(), and the connection
 *     should be re-established as well.
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Sleep_Module(u8 sleep_type, u32 time_to_wakeup_in_ms, u16 *status);

```

#### 功能

- 1、这个函数用于通过 SPI 接口配置模组进入深度休眠模式。

#### 说明

- 1、在退出休眠时, 片选信号 nCS 必须设置为低电平, 在正常启动后, 再释放拉高。
- 2、在退出休眠时或为了退出休眠, 通常可以直接调用模组的初始化函数来完成, 例如 M8266WIFI\_Module\_Init\_Via\_SPI() 以及其他的配置服务链接的函数。
- 3、适用于主机接口模式 2 (只使用 SPI 接口不使用串口) 下。

## 原型

```
u8 M8266WIFI_SPI_Sleep_Module (  
    u8 sleep_type,  
    u32 time_to_wakeup_in_ms,  
    u16* status)
```

## 参数

### u8 sleep\_type

休眠类型。

=3, 深度休眠。

=其他, 保留值。

### u8 time\_to\_wakeup\_in\_ms

休眠多久后退出休眠, 单位是毫秒。

如果设置为 0, 则模组将永久休眠, 直到外部唤醒。

### u16\* status

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

## 返回值

=1, 执行成功

=0, 出错了



## 4.9 模组信息查询

这一部分 API, 支持主机查询模组的相关信息和驱动的相关信息。

### 4.9.1 M8266WIFI\_SPI\_Get\_Module\_Info

```

/*****
 * M8266WIFI_SPI_Get_Module_Info
 * .Description:
 *   To get the M8266WIFI module information, such as ESP8266_ID,
 *   Flash Size, Firmware Version
 * .Parameter(s)
 *   1. esp8266_id: the pointer to the returned esp8266_id
 *                   Use NULL if you don't expect it returned
 *   2. flash_size: the pointer to the returned flash_size
 *                   returned value
 *                   = 0, if 512KB
 *                   = 1, if 1MB
 *                   = 2, if 2MB
 *                   = 3, if 4MB
 *                   = others, invalid
 *                   Use NULL if you don't expect it returned
 *   3. fw_ver      : the pointer to the returned firmware version
 *                   buffer size should be no less than 16 bytes
 *                   returned value e.g. "1.1.4-4"
 *                   Use NULL if you don't expect it returned
 *   4. status      : pointer to return errcode(LSB) and status(MSB) upon error
 *                   Use NULL if you don't expect them returned
 * .Return value:
 *   =1, success
 *   =0, has error(s)
 *****/
u8 M8266WIFI_SPI_Get_Module_Info(u32* esp8266_id, u8* flash_size, char* fw_ver, u16* status);

```

#### 功能

- 1、这个函数用于通过 SPI 接口获取模组的相关信息,, 包括模组的 ID、板载 FLASH 大小、模组固件的版本信息等。

#### 说明

- 1、适用于主机接口模式 2 (只使用 SPI 接口不使用串口) 下。

#### 原型

```

u8 M8266WIFI_SPI_Get_Module_Info(
    u32* esp8266_id,
    u8* flash_size,
    char* fw_ver,
    u16* status)

```

#### 参数

**u8 esp8266\_id**

模组的 ID。每个模组具备不同的 ID。

**u8 flash\_size**

模组上的 FLASH 的实际大小。

= 0, 512KBytes

= 1, 1MBytes

= 2, 2Mbytes

= 3, 4MBytes

=其他, 无效值

#### u8 fw\_ver

模组上固件的版本信息

#### u16\* status

执行出错时, 返回的状态码的指针, 方便故障诊断。

如果不需要返回状态码, 可以使用 NULL。

#### 返回值

=1, 执行成功

=0, 出错了

### 4.9.2 M8266WIFI\_SPI\_Get\_Driver\_Info

```

/*****
 * M8266WIFI_SPI_Get_Driver_Info
 * .Description:
 *   To get the M8266WIFI driver information
 * .Parameter(s)
 *   1. drv_info: the pointer to the returned driver information buffer.
 *               buffer size should be no less than 64 Bytes
 *               e.g. "ANYLINKIN M8266WIFI SPI DRIVER V1.4, 20170316"
 * .Return value
 *   driver_info: the pointer to the returned driver information
 *****/
char* M8266WIFI_SPI_Get_Driver_Info(char* drv_info);

```

#### 功能

1、这个函数返回模组驱动的相关信息。

#### 说明

无

#### 原型

**u8 M8266WIFI\_SPI\_Get\_Driver\_Info(char\*drv\_ver)**

#### 参数

**char\* driver\_info**

返回驱动信息的字符串数组, 长度不能小于 64 字节。

#### 返回值

指向驱动信息字符串数组的指针。

## 4.10 其他可通过 SPI 接口的设置、查询、和控制功能

如果需要其他的可通过 SPI 接口的设置、查询、控制等功能, 或对某些功能做出调整, 可以和我们联系探讨添加。

ALK8266WIFI®模组支持 OTA 远程固件升级, 在需要更新固件来支持新增功能时, 可以通过 OTA 来是对固件的更新。升级过程也很简单, 只需要将模块接上单片机的 SPI 接口, 让模块接入互联网, 并通过一个简单 SPI API 函数就可以启动升级。升级过程一般不超过 10 秒。

## 非公开声明

本文档及相关资料(包括相关文档和参考例程等),仅授权购买我司高速 WIFI 模组 ALK8266WIFI® 的客户(公司)使用参考,其他人员不得使用或参考。

凡经我司正常渠道接收本文档及其相关资料的用户,即获得使用本文档及相关资料的授权。

未经我司同意,授权客户不得对外公开、分享,或转让本文档及其相关资料的部分或全部。