Anylinkin® ALK8266WIFI®模组

SPI 接口高速通信使用与集成

主机集成说明

目　　录

1 前言

1.1 ALK8266WIFI® 高速 WIFI 模组方案介绍

Anylinkin® ALK8266WIFI® 是一款灵活、功能强大、高性能、精简小尺寸、绿色环保、在高性能前提下的高性价比的 802.11 b/g/n 无线模组。它适用于单片机 WIFI 高速传输音视频、高速采集、大块数据等应用场合。ALK8266WIFI® 解决方案，从模组硬件设计、模组固件设计、以及主机驱动优化算法等个多角度，来全面支持客户简单方便地实现对产品的功能与性能以及稳定性的高要求。

ALK8266WIFI® 模组包含有（1）高性能且高度集成的无线片上系统芯片，提供智能高效的无线接入和通信；（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。

和主机接口通信，可以单独使用 ALK8266WIFI® 模组所提供 UART 接口或 SPI 接口，也可以混合使用这两种接口。这两种接口均可以实现模组的控制查询与数据收发，既可以并行或交错存在，也可以各自独立使用。

通过 ALK8266WIFI® 模组所提供的 UART 串口邮票孔半孔接口，MCU 系统（1）基于兼容乐鑫标准的 AT 指令，可以和远端 TCP/UDP 服务节点实现串口转 UART 通信，最大速度可达 2Mbps 以上；（2）通过丰富的串口 AT 指令，便捷地实现对模组及其片上资源进行设置、查询和控制。它完全兼容乐鑫全部的 AT 指令，并在此基础上扩展了一些更加灵活方便和实用的 AT 指令。

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，实测有效吞吐速度可以超过 1 兆字节每秒，适用于高速采集、语音、图片以及视频传输等场合；（2）基于我司提供的基本 SPI 控制协议驱动，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，节约了主机的 UART 串口资源。

我司提供包含有丰富实用的 API 函数的驱动库，支持单片机主机灵活高效的对模组进行配置查询以及高速数据收发。此外，该驱动库还集成我司独有的 SPI 高速高效读写与均衡等优化算法，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了实测比 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，确保稳定性和性能同时最优。

ALK8266WIFI® 模组上的固件兼容乐鑫标准的 AT 版 SDK 固件，所以（1）支持了全部的乐鑫原版 AT 指令功能；同时，（2）在标准 AT 指令的基础上进行了 AT 指令扩展，以支持更多的开发功能和 SPI 接口配置及连网等功能。

同时，ALK8266WIFI® 模组优化设计了有关 SPI 接口和 TCP UDP 通信方面的实现，提供了底层协议握手，在收发缓存、主机访问效率、通信的效率/准确性/可靠性等方面做了深入优化，从而支持通过 SPI 接口进行模组的控制配置以及网络高速通信功能。在此，我们也提供单片机主机端的驱动和 API 例程。

ALK8266WIFI® 模组扩展了配网配置的功能实现，支持多种灵活的配置和配网方式：除了兼容乐鑫原有的智能配网（Esptouch SmartConfig、微信 Airkiss 等）、串口 AT 指令配置配网等方式之外，还提供了通过 SPI 接口的 API 直接配置配网，以及通过 WEB 网页配置配网，极大地方便了用户对模组的配置操作和场景适配性。同时，对于智能配网功能增加了额外的进展状态指示，大大地提高了智能配网的灵活性可靠性以及和可诊断性，优化了用户使用传统的智能配网方式的操作体验。

ALK8266WIFI® 模组提供灵活实用的无线通信解决方案。首先，它支持工作站 STA，热点 AP 以及 STA+AP 混合模式，因此无论是否存在第三方热点，都可实现通信；其次，模组硬件的设计充分考虑信号和电源的完整性及 EMC，板载 PCB 天线经过优化设计，且每个模组出厂时射频频率出厂时逐一校准并支持自动校准锁频；第三，支持 UDP，TCP 客户端，TCP 服务器，TCP 支持 SSL 通信，UDP 通信支持广播、组播以及单播，灵活高效，使用方便灵活；此外，它还只支持多链接、多客户端，每个链接独立随意配置，提供实用的多通道高速通信。

ALK8266WIFI® 模组与主机处理器的集成方便，占用主机处理器资源低。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，对于常见系列单片机型号，我们可以提供经过验证的单片机完整例程包，及测试结果。

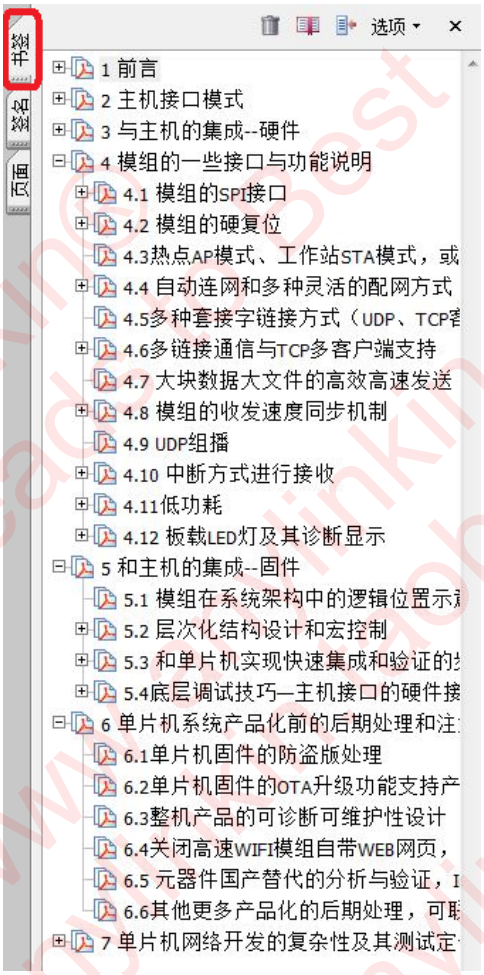
ALK8266WIFI® 模组已经成功地和多款主机处理器实现了集成与高速通信，包括但不限于：意法半导体 STM32 系列单片机（F0/F1/F2/F3/F4/F7/L1/L4/H7 等系列）、STM8 系列单片机、恩智浦 NXP 系列单片机（如 LPC11xx 系列、LPC1768/LPC1788 系列、LPC55 系列、LPC43XX 系列、K22/K27/K28、跨界处理器 i.MX RT105x / RT106x 系列、K60 和 KV58 等）、北欧半导体 Nordic nRF52 系列、TI MSP430 和 C2000 系列单片机/DSP、Holtek（台湾合泰）系列单片机（如 HT32F1656 等）、Nuvoton（华邦新唐）系列单片机（如 NUC123/NUM451 等）、C51 系列单片机、以及 Sunplus 凌阳等系列单片机，或基于这些单片机内核 IP 的 SOC，也越来越多的支持国产替代性质的单片机，包括但不限于兆易科技的 GD32 系列单片机、灵动科技的 MM32 系列单片机、华大半导体的 HC32 系列 等。在这些单片机上，通过杜邦飞线连接的实测速度基本都超过 500Kbytes/s(SPI 频率超过 8MHz)，绝大多数都在 800Kbytes/s 以上，那些 CPU 主频不低于 150MHz 的单片机则达到或超过 1MBytes/s。

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(Socket)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。

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—主机驱动 API 函数
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 联系方式

网址： http://www.anylinkin.com
淘宝： http://anylinkin.taobao.com
电邮： IoT@anylinkin.com, 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 主机接口

2.1 ALK8266WIFI®和主机接口的交互类型

ALK8266WIFI®模组主机接口，提供了 ALK8266WIFI®模组和单片机主机之间的交互功能，包括（1）查询控制，和（2）数据通信。查询控制接口支持主机对 ALK8266WIFI®模组进行查询和控制，数据通信接口支持主机通过 ALK8266WIFI® 模组和远端网络节点间的数据交互。

2.2 ALK8266WIFI®主机接口方式

根据数据通信和查询控制使用串口或 SPI 接口，ALK8266WIFI®模组可以工作在如下几种主机接口方式。我们推荐使用方式 2。

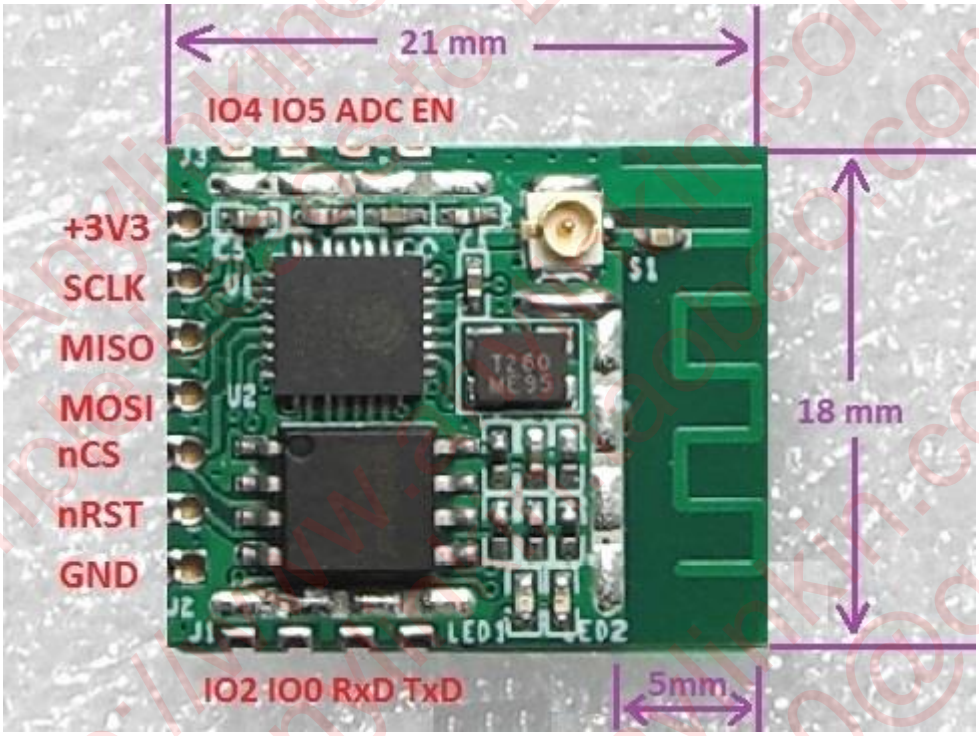
接口方式	主机接口		备注
	查询控制	数据通信	
接口方式 1 （仅通过 UART 串口进行设置、查询控制和数据通信）	UART 串口	UART 串口	1. 兼容乐鑫 SDK 提供的标准 AT 指令实现对模组的设置、查询和控制，指令功能丰富； 2. 兼容乐鑫 SDK 提供的标准 AT 指令实现网络通信，速度较慢，最大波特率 2Mbps； 3. 占用主机管脚资源较少，只需要 2 根串口线和一根 GPIO 实现硬复位。
接口方式 2 （仅通过 SPI 口进行设置、查询控制和数据通信）	SPI 口	SPI 口	1. 使用自定制 SPI 高速传输协议实现对模组的设置、查询和控制，接口功能足够丰富，操作也更高效； 2. 使用自定制 SPI 高速传输协议高速传输数据，速度快，最大波特率 40Mbps；瓶颈主要来自主机 SPI 读写速度，以及网络环境； 3. 占用主机管脚资源较少，需 4 根 SPI 线和以及一根 GPIO 实现硬复位。 **4. 我们推荐使用这种主机接口方式。**
接口方式 3 （通过 UART 串口进行设置、查询控制，通过 SPI 口进行数据通信）	UART 串口	SPI 口	1. 兼容乐鑫 SDK 提供的标准 AT 指令实现对模组的设置、查询和控制，指令功能丰富； 2. 使用自定制 SPI 高速传输协议高速传输数据，速度快，最大波特率 40Mbps；瓶颈主要来自主机 SPI 读写速度，以及网络环境； 3. 占用主机管脚资源较多，需要主机同时提供串口、SPI 接口、以及一根 GPIO 做硬复位； 4、考虑到模式 2 的接口功能足够丰富，操作也更高效，我们建议不必使用这种接线多占用主机接口相对较多的接口方式。

3 与主机的集成--硬件

3.1 模组的管脚定义和封装说明

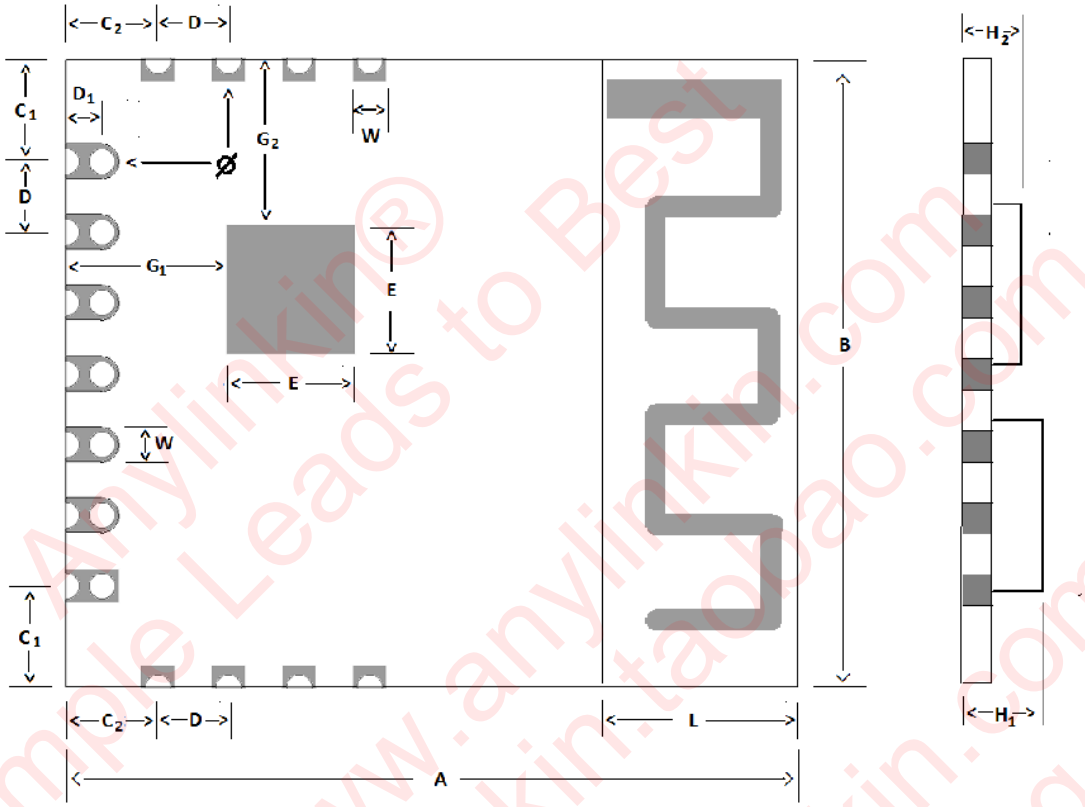
3.1.1 模组的管脚定义

ALK8266WIFI®模组的硬件管脚定义的详细说明，详情请参看文档《ALK8266WIFI 模组数据手册》中“管脚定义”章节。



3.1.2 ALK8266WIFI®模组的封装尺寸和推荐的主板 PCB 封装说明

ALK8266WIFI®模组采用标准 2.0mm 的排针插针，兼容邮票孔的贴片方式，如下图所示。



符号	符号含义	尺寸典数值 单位毫米	备注
A	模组长	21.00±0.1	
B	模组宽	18.00±0.1	
C_1	组合孔中心到模组边缘距离	2.90±0.1	
C_2	半孔中心到模组边缘距离	2.65±0.1	
D	邮票孔(半孔和组合孔)中心间距	2.03±0.1	80mil
D_1	组合孔中心间距	1.00±0.1	40mil
Φ	邮票孔(半孔和组合孔)内径直径	0.75±0.1	适配直径为 0.5mm 的排针
W	邮票孔焊盘宽	1.00±0.1	40mil
E	散热焊盘（正方形）的边长	3.75±0.1	
G_1	散热焊盘边缘到模组边缘 1 距离	4.56±0.1	
G_2	散热焊盘边缘到模组边缘 2 距离	4.71±0.1	
L	板载天线区域的长度	5.00±0.1	
H_1	模组主芯片高度	1.60±0.2	
H_2	模组最大总高度	2.60±0.2	

注：
1. PCB建库时，邮票孔贴片焊盘建议向外延展1mm以方便焊接
2. PCB建库时，主板散热焊盘建议：焊盘内缩略小于模组散热焊盘实际尺寸，而隔离盘尺寸略大于模组散热焊盘建议焊盘3.2x3.2mm，隔离盘4.2x4.2mm

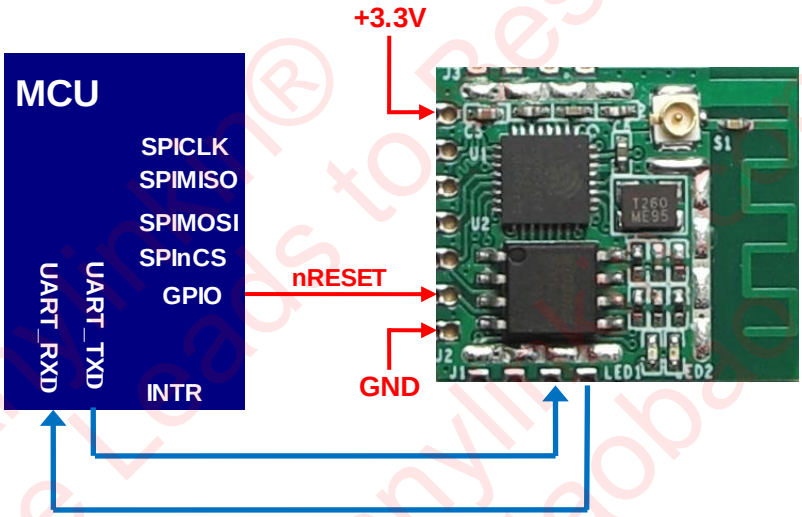
3.1.3 ALK8266WIFI®模组 贴装建议示意图（供参考）

对上图的补充说明：上图中“白浆”的作用是“电隔离”。如果模组区域主板表层没有走线或铺铜（而将对辅助屏蔽的铺铜放在了主板的中间层或底层），则不会存在模组底层走线和主板上表层在贴装后靠得太近的风险，“白浆”也可不要。

3.2 模组和主机的逻辑互连

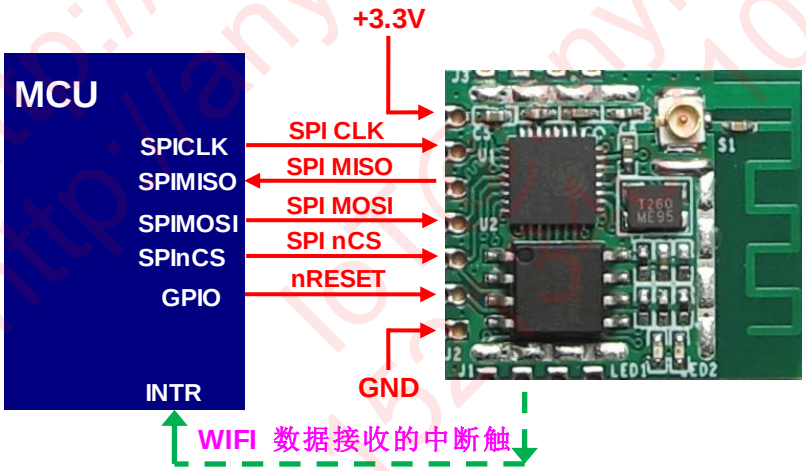
3.2.1 主机接口方式 1-仅通过 UART 口对模组进行设置/查询/控制和数据传输

这种方式，兼容乐鑫 SDK 提供的标准的 AT 指令，通过 UART 串口，对 ALK8266WIFI®模组进行设置、查询、控制和数据通信。适用于使用传统的串口 AT 指令方式通信对速度要求不高的场合。



3.2.2 主机接口方式 2-仅通过 SPI 口对模组进行设置/查询/控制和数据传输

这种方式，采用我们提供的 SPI API 接口对 ALK8266WIFI®模组进行设置、查询、控制和数据通信，不需要 UART 串口及 AT 指令。适用于可靠高速通信的场合。



上图中：

(1) ALK8266WIFI® 模组工作在 SPI 从机模式；

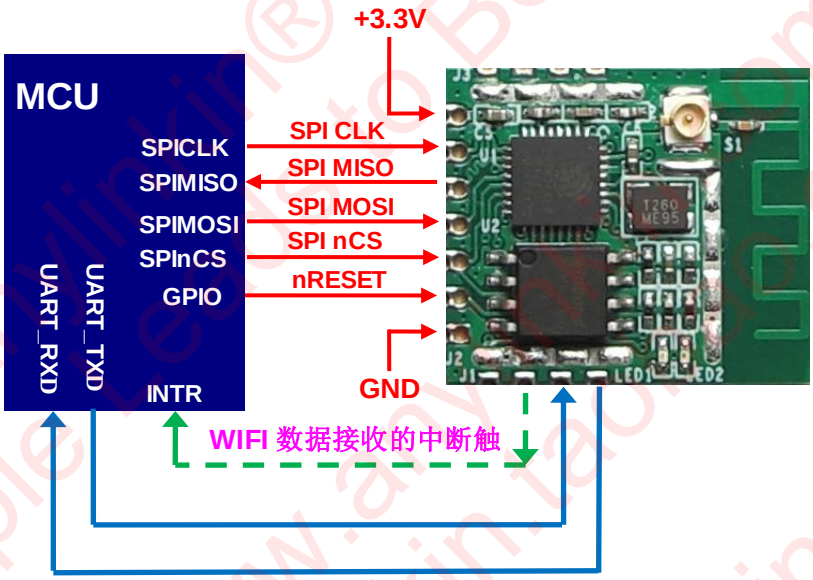
(2) 在接收到来自 WIFI 数据时，ALK8266WIFI 的 IO0 管脚会输出上升沿，可用于触发单片机的外部 IO 中断，来实现对模组接收数据的实时获取，如图示中虚线部分所示，详情可参看 4.10.1 中断方式接收的管脚输出和时序图。如果不使用

该特性，则可通过 SPI API 接口查询方式检查是否从 WIFI 上接收到了数据。

对于高速通信，我们推荐使用这种方式和单片机主机进行逻辑互联。

3.2.3 主机接口方式 3-通过 UART 口对模组进行设置/查询/控制，通过 SPI 口进行数据传输

这种方式，采用乐鑫 SDK 提供的标准的 AT 指令以及我们提供的扩展 AT 指令，对 ALK8266WIFI®模组进行设置、查询和控制，而采用我们所提供的 SPI API 接口对 ALK8266WIFI®模组进行高速数据传输。



上图中：

(1) ALK8266WIFI® 模组工作在 SPI 从机模式；

(2) 在接收到来自 WIFI 数据时，ALK8266WIFI 的 IO0 管脚会输出上升沿，可用于触发单片的外部 IO 中断，来实现对模组接收数据的实时获取，如图示中虚线部分所示，详情可参看 4.10.1 中断方式接收的管脚输出和时序图。如果不使用该特性，则可通过 SPI API 接口查询方式检查是否从 WIFI 上接收到了数据。

对于高速通信应用，我们认为没有必要占用串口使用这种方式和单片机主机进行逻辑互联。

3.3 模组的电气特性参数和供电设计

3.3.1 电气特性参数表

模组正常工作时的电气特性参数如下所示，详情可参看文档《ALK8266WIFI 模组数据手册》。

			最小	额定	最大	单位
供电电压	邮票孔管脚 VCC 供电电压		3.0	3.3	3.6	伏
IO 电压	邮票孔 IO 管脚电压	VIL，输入逻辑 0 电平	-0.3		0.25	伏
		VIH，输入逻辑 1 电平	0.75VCC		3.6	
		VOL，输出逻辑 0 电平			0.1VCC	
		VOH，输出逻辑 1 电平	0.8VCC			
IO 电流	邮票孔 IO 管脚的输出电流				12	mA

3.3.2 稳态供电电流与去耦

ALK8266WIFI® 模组的是射频收发模组，在射频收发时，射频模组一般都会需要较大的供电电流，以支持足够距离的射频辐射。模组的功耗参数请参看《ALK8266WIFI 模组数据手册》“功耗表”章节。

ALK8266WIFI® 模组在正常工作但没有 WIFI 收发时时，平均消耗电流典型值约为 30mA@3.3V，作为 STA 收发时的消耗电流典型值约为 110mA@3.3V，作为 AP 工作时消耗电流典型值约为 120mA。因此，建议模组的稳态供电电流不得小于 120mA@3.3V，最好确保在 150mA@3.3V 以上，以留有裕量。

ALK8266WIFI® 模组的设计充分考虑了电源的完整性设计。只需要在模组的 VCC 管脚附近接上一个 1.0~4.7uF 的电容，不需要再为模组并联额外的电容。建议最好选用 4.7uF，如果主板的电源完整性设计较好，也可以采用 1.0uF 的电容来节省成本。

3.3.3 主板电路的电源完整性（PI）设计考虑

射频模组，尤其是远距离发射模组，一般都具备一个特点：发射时的功耗较大（以便有更多的能量辐射出去从而传输较远的距离）；而不发射时的功耗则较小。例如 ALK8266WIFI®模组，发射时的功耗超过 110mA，而不发射时的功耗会迅速降低到 20mA。在模组的上电或断电阶段，则往往可能会在 0mA 和 100 多 mA 之间迅速切换。在设计主板电路时，需要考虑射频模组的这种特性对主板其他部分电源完整性的影响。

此外，在一些存在放大电路对噪声敏感的场合，比如音频放大电路的场合，这种电流噪声被耦合放大后，有可能会成为很显著的噪声。

所以，在这样一些场合，在使用射频发射模组时，都会注意一下几点：

（1）设计好电源分配网络拓扑结构，注意电源网表末端电路对中间电路供电的影响；

（2）确保地回路的完整性和最短性，避免多路信号回路的重叠和拥挤；

分析射频模组的电源路径和信号在的最小地回路，确保绕开敏感模拟电路区域；

（3）选取合适数值的电容值，放置电容时考虑电容的去耦半径。

3.3.4 nRESET 管脚的去耦

有些主板的电路设计不佳，或者处于较强的环境辐射干扰的场合，例如 nRESET 主板引线过长容易受到显著辐射耦合或串扰，被传导入模组后可能会导致模组意外复位。此时，可以通过在模组的相应管脚增加一个对应频率的去耦电容，一般为 1nF-0.1uF。对于这些信号，如果需要较长走线，建议布线两侧布地保护(Guarded Plane)，或者在在相应管脚出加上对应相关耦合噪声频率的容值的电容。

但是建议不要使用较大数值的电容，较大数值的电容可能会导致 nRESET 复位周期拉长，不利于模组的快速复位。

3.4 模组管脚上的保护和匹配电路

3.4.1 管脚的接线保护

基于精简小尺寸设计的需要，模组设计依传统方式，其管脚均从模组上的主芯片直接引出，并未增加相关的隔离保护。

对于常见的主板设计弱电场合，进行直接逻辑互联一般问题不大。

但是，若电路存在较大的阻抗失配，可能会导致出现较大的过冲或下冲，甚至超过模组管脚的最大高低电平，长期持续过冲或下冲可能会损伤模组。因此在产品开发阶段，建议可以用示波器观察相关信号的上下冲程度，必要时，应加上电阻，可起到匹配和限流的双重保护作用。特别地，对于一些特殊的场合，比如使用电机等存在强弱电共享的场合，建议模组管脚和主机板之间的串上几欧到几十欧的隔离电阻进行限流保护，必要时还需要在电源等管脚处加上稳压管或 TVS 管进行过电压保护。

3.4.2 SPI 主机板接口布线建议

为了尽可能地提高 SPI 可靠通信的频率，需要对 SPI 主机板上的走线兼顾高速电路布线的一些基本规则和技巧。为了减少布线不佳造成的反射或时序差过大，SPI 布线有如下建议：

1、MOSI 和 SCK 尽可能等长；

2、MISO 和 SCK 走线长度之和尽可能短（模组尽可能靠近单片机主机）。

如果模组距离单片机主机不得不较远，建议：

3、MISO、MOSI、SCK、nCS 尽量并排走线；

4、在模组管脚处留下电阻焊盘做串联匹配。

补充说明：

（1）串联匹配用的电阻一定要根据实际布板后阻抗进行调整匹配，如果随便串接一个电阻而不进行阻值的匹配调整，可能反而会起反作用，导致振荡或阻尼更大，更不利于 SPI 的高速通信。

（2）调整匹配的方法技巧也可以很简单，不必使用专业的仪器。比如，在单片机里写一小段代码，临时将 SCK 等初始化为 GPIO 输出模式，然后以相同的 IO 边沿方式（快速或慢速）配置好后，在这些管脚上输出一个上升沿或下降沿，用示波器观察对应模组管脚上信号波形。通过调整串联匹配电阻的阻值，直到观察到波形不出现较大的超调。

（3）建议保持一定比例的失配，使波形存在略微一些的超调（即上升之后还存在一定幅度的过冲，这个过冲所造成的振荡幅度控制高低电平的可靠识别范围和正常工作的电压的上下限之内）。保持一定超调有利于缩短信号的边沿时间有利于提高工作频率。

5、注意高速电路布线的一些基本规则，主要是反射和串扰等影响，必要时，可以在数据走线上，尤其是插座处，放置大约 10pF 的接地电容

有关主板电路设计上的信号完整性（SI）、电源完整性（PI）、电磁兼容（EMC）等方面的技巧或详情，可以联系我们具体探讨。

3.5 模组的散热

WIFI 模组通常属于高耗能器件，这是因为 WIFI 发射功率越大代表可以传输较远的距离，或者在同等距离下获得更高的抗干扰性能和较高的速度。当环境较为恶劣时，ALK8266WIFI® 模组也会自动调整发射功能直到最大，来提高传输的效率和可靠性。所以，WIFI 模组，尤其是长期处于高速通信状态下的 WIFI 模组，在电路设计时，需要考虑其散热处理。

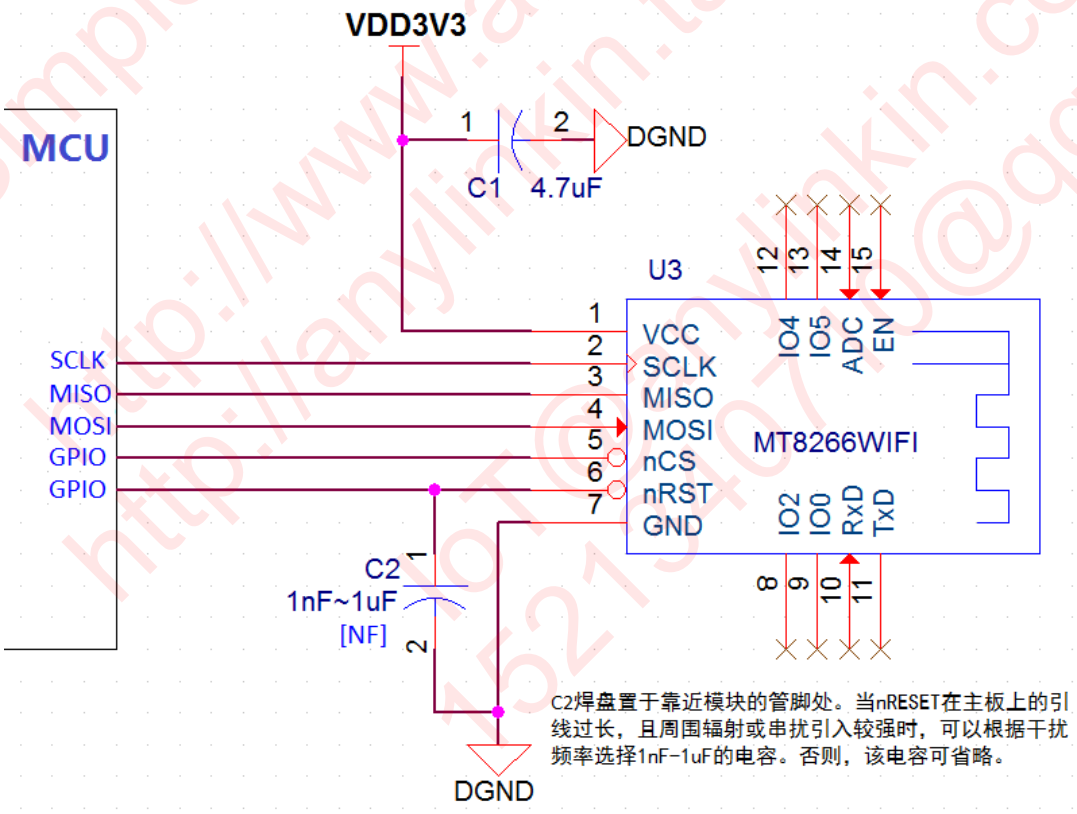
除了传统的加装散热片的做法之外，我们建议充分利用主板 PCB 辅助模组进行散热。

WIFI 模组背面有一个裸露的 GND 网表散热焊盘，在主板 PCB 设计时，（1）可以挖空 PCB 对应部分，以便该散热焊盘通过空气对流进行散热；（2）也可以在主板上对应部分留出 GND 裸露焊盘，装配时紧贴模组的扇热焊盘，同时将该 GND 裸露焊盘和主板上的大块铜皮宽相连，增大散热效果。我们推荐采用第（2）种方法。

另外，将模组放置在主板边缘容易空气对流的部分，也有利于模组的散热。

3.6 模组参考电路设计

3.6.1 最小电路设计



说明：SPI 主机的 MISO/MOSI 和 SPI 从机的 MISO/MOSI，**不要交错连接**。

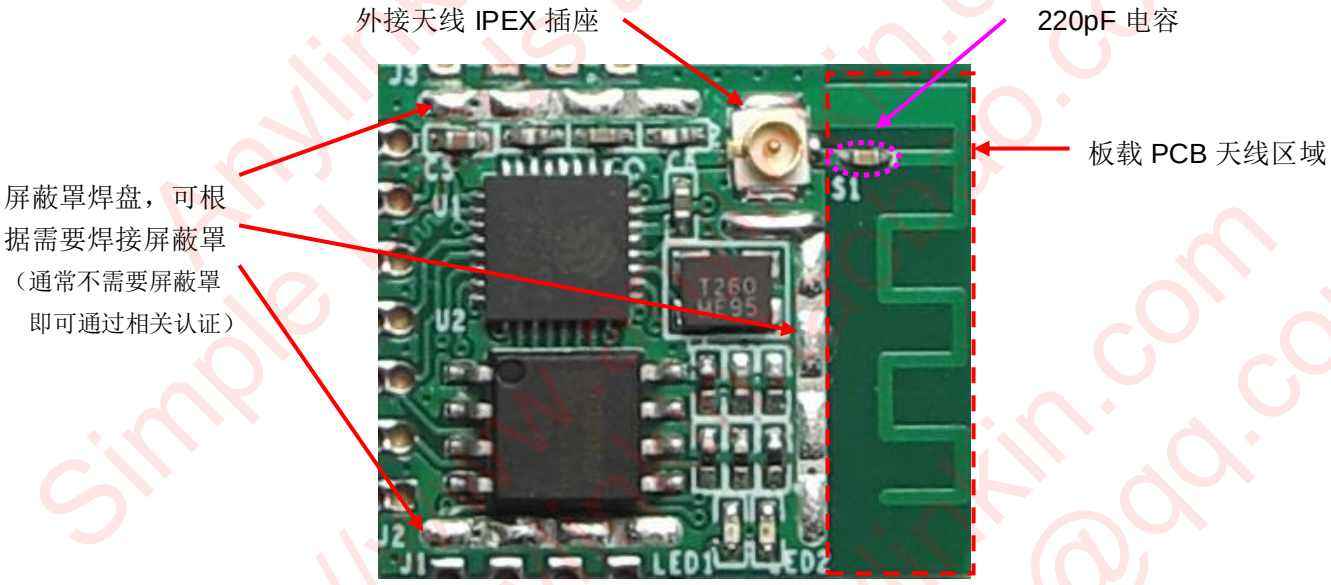
（理解为何不要交错，可参看 https://blog.csdn.net/kennychow/article/details/104807889）

3.7 模组天线的使用及模组在主机板上的位置布局

3.7.1 板载 PCB 天线和 IPEX 天线接口（一代 IPEX 天线座）

ALK8266WIFI® 模组上自带经过优化设计的高效的 PCB 天线，所以，如果对于传输方向性和距离没有特殊要求时，可以直接使用板载的 PCB 天线，而不用外接其他天线。PCB 天线的射频参数规范，请参看《ALK8266WIFI 模组数据手册》

如果对传输的方向性和距离有特殊要求，或者将 ALK8266WIFI® 模组放置在金属屏蔽箱体内而与和金属屏蔽箱体之外的设备进行通信，则需要考虑采用外接天线。模组上有一个一代 IPEX 天线座，提供对 50ohm 2.4GHz 天线的支持。当使用外接的 IPEX 天线时，可去掉 IPEX 旁边靠近 PCB 天线一侧的 220pF 电容。



3.7.2 使用板载 PCB 天线的注意事项和模组的放置

金属面/体靠近板载 PCB 天线区域，会对 PCB 天线的性能造成影响。因此，当采用 ALK8266WIFI® 模组的板载 PCB 天线时，PCB 天线区域和金属面/体的距离，需要超过 15mm。且在距离天线较近的位置的传输方向上，不能有大面积的金属面/体的屏蔽或遮挡。

PCB 区域的尺寸为 18x5mm，如 3.6.1 板载 PCB 天线和 IPEX 天线接口中图示。

金属面/体包括金属外壳或包装、PCB 天线区域下部的主板电源或地平面等。所以，在模组 PCB 天线区域下部以及边缘的主板 PCB 上，不得铺地平面或者有较密的走线（最好不要走线），包括表层、中间层、和底层，且边缘距离 PCB 天线区域的水平距离不小于 15mm。

说明：15mm 是一个保守的安全数据，具体数值还和模组 PCB 与底板的垂直距离以及通信的方向等因素有关，超过 15mm 后对于天线的影响基本可以忽略。例如，对于贴片模组的下部表层，在天线及其保守区域之外进行铺地平面，则有利于提高模组和底板之间的辐射隔离，可以联系我们探讨具体详情。

小技巧：**一种简单的检验“主板布局布线的好坏对 PCB 模组天线的影响”**，可以在相近的

距离，放置同款模组（但是不在主板上不受其影响），用其他设备简单扫描信号强度对比，如果发现两者信号强度的差别相差再 3-5dBm 之内，则表明主板布局布线对 PCB 模组天线的影响较小符合要求。。

如果需使用模组的 PCB 天线，**集成设计人员一定要评估主板是否布局布线合适，是否对 PCB 天线的性能造成了较大的负面影响**，如果发现主板对 PCB 模组天线的效果影响较大，一定要仔细分析原因并进行优化，或者联系我们协助。在技术协助用户的实战经验中，不少客户因为设计疏忽（包括但不限于：1、在 PCB 天线下面密集走线或铺铜；2、在 PCB 天线的外边缘铺地；3、在 PCB 天线的附近放置金属体；……）导致天线效果大打折扣（例如导致信号强度从-40dBm 降低到近-80dBm）。

3.8 其他注意事项

4 模组的一些接口与功能说明

4.1 模组的 SPI 接口

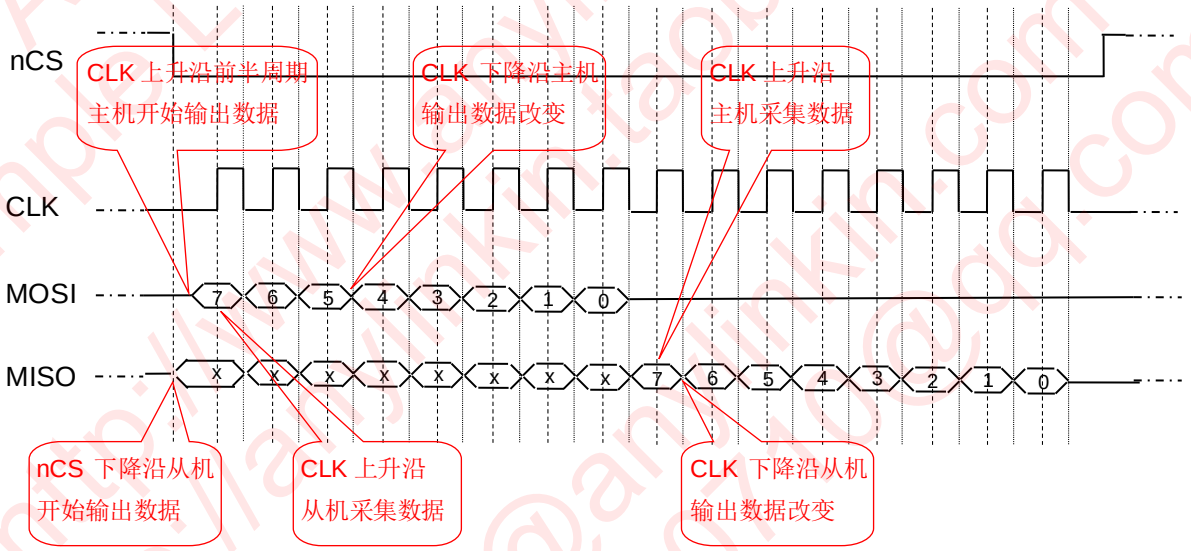
4.1.1 模组 SPI 的设置要求和字节读写时序

ALK8266WIFI® 模组的 SPI 接口工作在从机模式、模式 A 方式（即 CPOL=0, CPHA=0）、字节内高位优先串行发送、片选为软件控制模式。

所以，在对单片机主机的 SPI 接口进行初始化时，必须初始化为：SPI 主机模式、模式 A 方式（即 CPOL=0, CPHA=0）、字节内高位优先发送、片选为软件控制模式。

以下为标准 SPI 模式 A（或称为模式 0）的时序图示如下所示，供参考。

CPOL=LOW=0， 即：空闲时 CLK=0=低电平
CPHA=1st Edge = 0 即：在第一个边沿（上升沿）开始采集数据

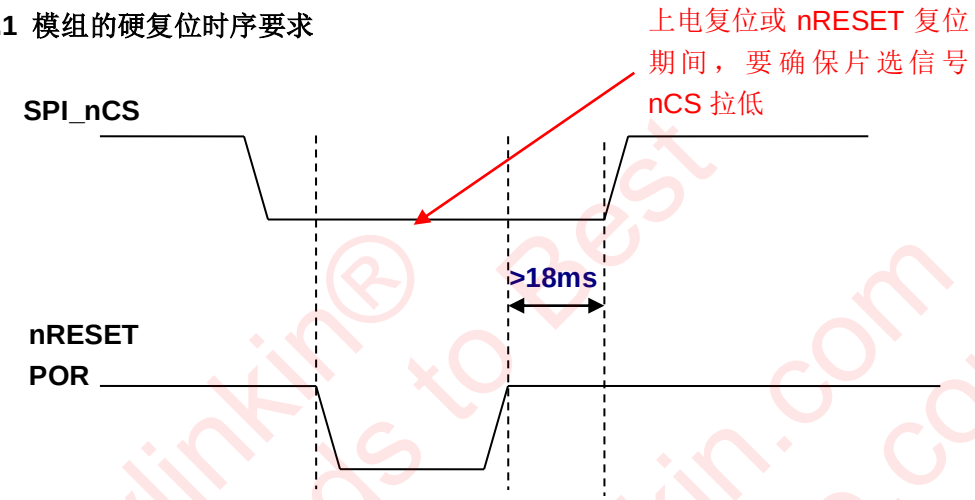


4.1.2 SPI 主机接口的固件初始化（跨平台移植时候需注意）

（1）SPI 模式 = 主模式 模组工作在 SPI 从模式，单片机工作在 SPI 主模式
（2）CPOL = LOW = 0 表示 SPI CLK 时钟在空闲时为低电平
（3）CPHA = 1st Edge = 0 表示在第一个边沿采集数据（结合 CPOL=0，上升沿）
（4）FIRST_BIT = MSB 表示一个字节内，高位先串行发出
（5）nCS 软件控制模式 即由单独的 GPIO 来控制模组的片选

4.2 模组的硬复位

4.2.1 模组的硬复位时序要求



说明：在模组退出硬复位时，需要确保 SPI_nCS 管脚为低电平，且在 nRESET 管脚稳定于高电平后，保持至少 18-20ms 以上。单片机编程控制时，建议使用 50ms 甚至 300ms 以上，以便有足够裕量，消除个体差异造成的竞争冒险。例如，在某些主板设计上，或许会因为寄生电容或长线失配振荡而导致 nRESET 稳定需要较长的时间。

4.2.2 模组的硬复位时序的固件实现

```
/*****************************************************************************************************************
 * 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, 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(5);          // delay 1ms, adequate for nRESET stable.
                                          // give more time especially for some board not good enough

   M8266HostIf_Set_nRESET_Pin(1);         // Pull high again the nReset Pin to bring the module exiting reset state
   M8266WIFI_Module_delay_ms(300);        // at least 18ms required for reset-out-boot sampling boottrap pin
                                          // Here, we use 300ms for adequate abundance, since some board GPIO,
                                          // needs more time for stable(especially for nRESET)
                                          // You may shorten the time or give more time here according your board v.s. effiency
   M8266HostIf_Set_SPI_nCS_Pin(1);
 //M8266WIFI_Module_delay_ms(1);          // delay 1ms, adequate for nCS stable

   M8266WIFI_Module_delay_ms(800-300-5-2); // Delay more than around 500ms for M8266WIFI module bootup and initialization
                                           // including bootup information print。
}
```

4.3 热点 AP 模式、工作站 STA 模式，或 STA+AP 混合模式

4.3.1 热点 AP 工作模式

ALK8266WIFI® 模组可以作为热点 AP 独立组网，其他网络节点作为 STA 加入 ALK8266WIFI® 模组建立的局域网来实现通信。当模组做 AP 时，在其所建立的局域网内，具备确定的静态 IP 地址，该地址缺省为 192.168.4.1（可通过 SPI API 接口或 WEB 网页修改该地址为其他值）。该特性适用于一些没有或者不想使用其他热点/路由器的场合，在此场合下，通过模组的热点 AP 一样可以实现组网通信。

ALK8266WIFI® 模组的热点 AP 可对接入设备进行允许或禁止接入的控制，从而进一步增强局域网内访问的安全性。请参看《ALK8266WIFI 模组 SPI 接口高速通信使用与集成—主机驱动 API 函数》中章节“4.2.22 M8266WIFI_SPI_AP_Permit_Block_A_STA”和 “4.2.23 M8266WIFI_SPI_AP_Config_PermitBlock_List”部分。

4.3.2 工作站 STA 工作模式

ALK8266WIFI® 模组可以作为工作站 STA，去接入其他热点或路由器，实现和其他节点的通信。此时，既支持通过 DHCP 自动获取 IP 地址，也可以通过 SPI API 接口设置静态的 IP 地址。当处于工作站 STA 模式下时，支持灵活多样的配网方式，且支持自动连网。

ALK8266WIFI® 模组的工作站 STA 可以扫描信号，根据信号强度实现粗略的测距功能。支持针对 SSID 或 BSSID 对目标信号进行快速地、针对性的单一扫描，这极大地便利了根据信号强度进行测距离的实际应用场景。请参看《ALK8266WIFI 模组 SPI 接口高速通信使用与集成—主机驱动 API 函数》中章节“4.2.13 M8266WIFI_SPI_STA_ScanSignalsBySsid”和 “4.2.14 M8266WIFI_SPI_STA_ScanSignalsByBssid” 部分。

4.3.3 AP+STA 混合工作模式

ALK8266WIFI® 模组还可以工作在 STA+AP 混合模式下，同时发挥 STA 和 AP 各自特性功能，实现一些灵活便捷的功能。

4.3.4 三种工作模式之间的简单切换

ALK8266WIFI® 模组可以在 STA、AP、STA+AP 这三种模式之间自由切换。单片机主机可以通过 SPI API 接口或者 UART 串口 AT 指令，实现在这三种模式之间自由切换。在这三种模式之间切换时，不必重启模组。

4.4 自动连网和多种灵活的配网方式

4.4.1 自动连网

所谓“连网”，一般指的是模组使用 SSID 及密码来接入路由器或热点并获得 IP 地址的过程。模组在启动时或者掉线后，会自动（重新）接入路由器或热点，而不需其他介入，一般称为“自动连网”。

“自动连网”通常包括两种情形，一种启动时的“自动连网”，即，在开机阶段，模组会自动从模组上的存储空间里读取所保存的 SSID 及密码去连接对应的热点或路由器。因此，这种类型的“自动连网”，一般需要将配网阶段获得的 SSID 和密码保存在模组的 ROM 里。另外一种是掉线后的“自动连网”，又称“自动重连”，即，连接着热点或路由器的模组掉线后，会使用当前的 SSID 和密码自动重新连接热点或路由器。

ALK8266WIFI® 模组支持“自动连网”，包括启动时的“自动连网”和“自动重连”。

4.4.2 模组的配网

所谓“配网”，一般指的是通过某种方式，将路由器或热点的 SSID 及密码传递给模组，以便模组可以接入路由器或热点。配网，包括直接配网方式（通过 UART 串口 AT 指令或 SPI API 接口等方式，将 SSID 及密码显性地传递给模组）、智能配网（通过 Smart Config / Smart Link / Airkiss 或 WPS 等方式传递给模组）、WEB 网页直接配网（通过模组上内嵌的 WEB 服务器进行直接配网）。

详情请参考文档《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》。

4.5 多种套接字链接方式（UDP、TCP 客户端、TCP 服务器）

ALK8266WIFI® 模组可以作为 UDP 节点、TCP 客户端、TCP 服务器建立链接和通信。详情请参看《ALK8266WIFI 模组 SPI 接口高速通信使用与集成—主机驱动 API 函数》中章节“4.3.1 M8266WIFI_SPI_Setup_Connection” 部分。

当本模组作为 TCP 服务器，配合模组的 AP 模式，将非常方便地支持网络设备之间的高速传输，而不必依赖于第三方路由或热点。例如，在模组启动后，开启模组的 AP 模式支持，来给其他 WIFI 设备提供 WIFI 接入，同时建立一个作为 TCP 服务器，监控其他网路设备作为 TCP 客户端的套接字链接，简单地实现直连高速传输。

4.6 多链接通信与 TCP 多客户端支持

4.6.1 多链接通信

ALK8266WIFI® 模组支持多链接通信，即，在一个 ALK8266WIFI® 模组上，可同时建立多个 TCP 或 UDP 链路通道，每个通道用唯一一个通道链接号进行标识。目前支持同时建立最多 4 个链路通道，链接编号为 0~3。如果需要同时建立更多链路同道，可以和我们联系。

在创建一个链接时，会指定一个链接号 link_no，作为该链接的唯一标识。在发送数据时，通过通道链接号向对应的链接（目标节点的 IP 地址和端口号）发送数据。在接收数据时，也会返回当前数据接收的链接号。

4.6.2 TCP 多客户端支持

ALK8266WIFI® 模组支持建立 TCP 服务器，来监听多个 TCP 客户端。

每个 TCP 服务器所支持的客户端个数可以进行设置，单个 TCP 服务器最多可支持 15 个客户同时连接。*同时，模组上所有的 TCP 连接数（包括模组作为 TCP 客户端的通道链接、模组作为 TCP 服务器的通道链接上所有的客户端个数、以及模组的串口建立的 TCP 链接如果有的话、以及模组上的 TCP 服务器对应的客户端连接数等等）也不能超过 15 个。*

4.6.3 TCP 服务器对多客户端支持的代码实现

第一步：创建 TCP 服务器，指定服务器的本地端口

M8266WIFI_SPI_Setup_Connection

第二步：设置 TCP 服务器所支持的最大客户端个数

M8266WIFI_SPI_Config_Max_Clients_Allowed_To_A_Tcp_Server

说明，模组缺省的 TCP 服务器只允许接入 1 个客户端，如果需要允许接入多个客户端，需要调用上述函数来允许多客户端同时接入。

第三步：使用带远端地址和端口信息的收发函数

M8266WIFI_SPI_RecvData_ex()

– 这个函数可以同时返回所接收的数据来源客户端的地址和端口信息。

M8266WIFI_SPI_Send_Data_to_TcpClient

– 这个函数向多个客户端中所指定的那个客户端发送数据。

其他：

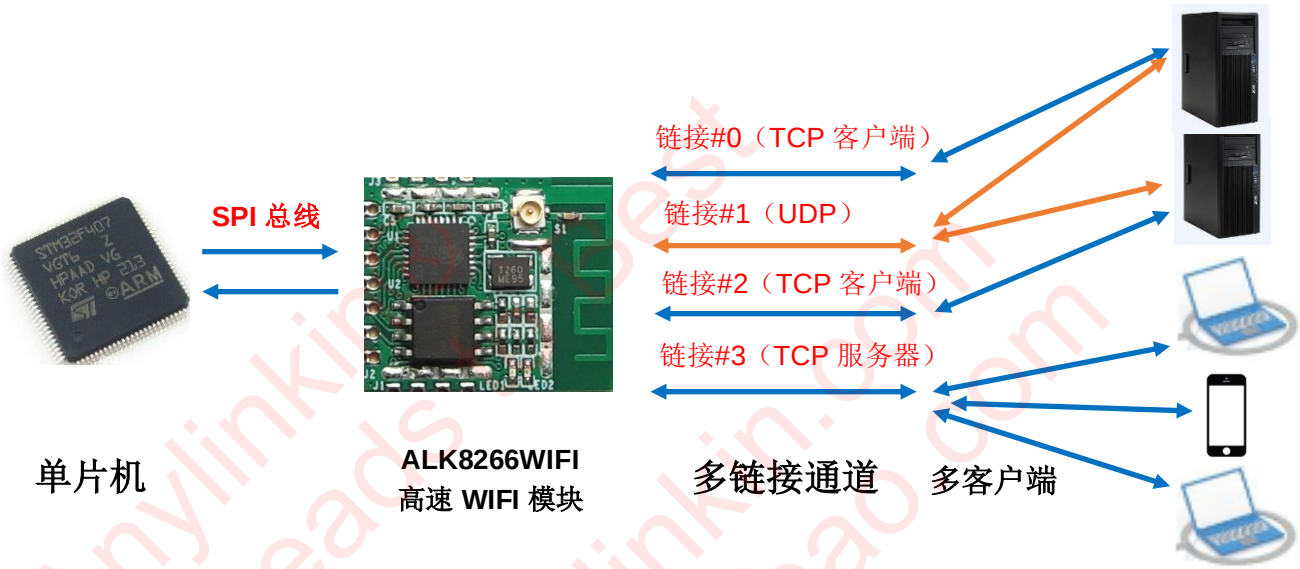
M8266WIFI_SPI_List_Clients_On_A_TCP_Server

- 这个函数可以列举连接着当前 TCP 服务器的所有客户端信息

M8266WIFI_SPI_Disconnect_TcpClient

- 这个函数可以断开当前连接的某个客户端

4.6.4 多链接通道和多客户端通信示意图举例



4.7 大块数据大文件的高效高速发送

本模组主要适用场合是进行高速高效的数据收发，所以大块数据的发送，是一个常见的应用场景。本模组支持大块数据的高速高效发送，数据块长度可以足够长，所以特别适合大文件等大块数据的发送。

第一步　在创建套接字之前，配置发送窗口数

M8266WIFI_SPI_Config_Tcp_Window_num(link_no, 4, &status)

第二步　调用块数据发送函数实现不限长度的大块数据的高效发送

M8266WIFI_SPI_Send_BlockData

4.8 模组的收发速度同步机制

4.8.1 速度同步机制的重要性

TCP/UDP 网络通信是一种流控制，而非实时传输，所以，我们看到的任何 TCP/UDP 通信，速度基本不会恒定，经常会受到网络条件的影响而出现明显地波动，甚至波动幅度偶尔还会很大。而单片机侧本地 SPI 总线，因为受到的环境影响较小，速度一般可以维持恒定。因此，确保一方的速度对另外一方保持同步，有助于提高通信的可靠性和效率。

ALK8266WIFI® 模组，和其他的模组比较，一个很大的优势是，启用了收发方的速度同步保持机制，从而大大提高了通信的可靠性和效率。

4.8.2 发送同步

在模组做 TCP 发送的过程中，如果接收方的速度偏慢（持续偏慢或偶尔偏慢），比如，因为路由带宽共享、周围环境干扰信号太多、接收方的处理太慢 ACK 响应延迟，导致单片机的写入 WIFI 模组请求发送的速度超过了 WIFI 链路上的实时速度，ALK8266WIFI® 模组会在

u16 M8266WIFI_SPI_Send_Data(u8 Data[], u16 Data_len, u8 link_no, u16* status)

函数的 status 指针的低字节返回 0x12 异常码，来通知单片机此时 SPI 接口写入的速度（相对接收方的接收速度）偏快了。此时，单片机可以立刻反复重新写入剩下的数据直到写入成功，也可以等待一些时间再来重写。重写或等待，就意味着单片机侧的速度在跟随网络断的速度进行自适应调整，从而实现了发送对接收环境的自适应同步。

4.8.3 接收同步

在模组做 TCP 接收的过程中，如果模组接收到数据后，单片机来不及从 SPI 接口读取走而导致累积过多，模组也会在 TCP 底层会自动通知发送方暂停发送，直到单片机取走数据释放缓存后，模组会在底层通知发送方可以继续发送，从而实现了接收处理对发送的速度同步。

接收同步的实现，不需要单片机侧做相关工作，模组独立完成对这个特性的支持。一般说来，符合 TCPIP 协议规范的上位机侧，也不需要做特殊的处理。

4.9 UDP 组播

UDP 支持广播和组播，极大的提高的某些场合下的通信效率。ALK8266WIFI® 模组，支持广播和组播。广播操作，只需要将目标地址填写为 255 广播地址，然后和普通目标节点一样发送。这里说一说，ALK8266WIFI® 模组对组播的支持。

组播的实施步骤：

1、确保模组在 STA 或 STA+AP 模式，不能是 AP-Only 模式

2、确保模组已经成功地连上了路由器（建立着这个组播组的路由器）

3、然后，将模组加入指定的组（如果这个组不存在，模组也会通知路由器建立这个组）详情请参看《ALK8266WIFI 模组 SPI 接口高速通信使用与集成—主机驱动 API 函数》中章节 4.3.5 M8266WIFI_SPI_Op_Multicuast_Group

4、建立 UDP 服务，目标地址指定为组播地址

5、开始组播

4.10 中断方式进行接收

4.10.1 中断方式接收的管脚输出和时序图

如 3.2 ALK8266WIFI®模组和主机的逻辑互联方式中的模式 2 和模式 3 中图示所示，WIFI 在接收到来自网络的数据时，会在 WIFI 模组的 IO0 管脚上输出相应的上跳沿。当主机取走数据后，会在 IO0 上输出低电平。因此 IO0 可以接单片机的某个 GPIO 来触发单片机的外部中断，以提高接收响应的实时性。对应的时序图如下所示：

小提示 1：这里所说的“中断方式接收”，指的是单片机的外部 IO 输入中断，由模组的 IO0 实现触发，及时地告知单片机主机，WIFI 模组收到了数据。不要误会成是 SPI 中断。单片机判断是否有数据可以接收的两种常见的方式，分别是 “查询方式接收”和“中断方式接收”。ALK8266WIFI® 模组通过调用 M8266WIFI_SPI_Has_DataReceived()来支持查询方式的接收，而通过 IO0 的输出波形实现对单片机主机的“中断方式接收”的支持。

小提示 2：如果主机从 WIFI 模组读取走一个网络数据包后，模组上还有缓存接收包在等待主机取走，那么，WIFI 模组会在主机读走这个数据包时，先拉低 IO0 输出，等待大约 50us 后，再次输出一个上升边沿，以便继续触发单片机外部 IO 中断来接收后续等待读取的数据包。因此，单片机的外部 IO 中断触发应配置为上升沿触发，且处理好可能的多次连续触发。

小提示 3：单片机主机在处理由外设来触发的外部 IO 中断时，应考虑外设复位时处于过渡态下的状态不定所造成的对外部 IO 中断的误触发。此外，模组在稳定退出复位开始启动的最初期，会在 IO0 上很短暂地输出一段振荡方波（大约 26MHz 频率的），这段振荡时间很短，一般不超过几个毫秒。因此，如果应用中会使用 IO0 来触发单片机的外部中断，为了避免该振荡造成多余的中断误触发，我们建议，单片机初始化阶段对该 IO 中断配置和/或使能，放置在对模组初始化阶段的复位（M8266WIFI_Module_Hardware_Reset()）完毕后，开始正式通信之前。如果在运行过程中单片机会对模组进行复位，那么在复位之前，也应临时先关闭单片机该 IO 中断使能，等复位完毕后再次开启。

小提示 4：通常单片机的中断处理，应考虑可能的共享冲突，可类比对常见的 SPI FLASH 的访问：如果单片机正处于 SPI FLASH 的读周期中，此时触发中断在中断里执行 SPI FLASH 的写周期，或者该 SPI 总线上的任何一个 SPI 外设的访问，那么之前正常进行的 SPI FLASH 读周期将被打乱而出现意外。这是因为中断里再次访问同一个 SPI 总线导致了共享冲突。进行中断接收处理时，访问 SPI WIFI 模组也有类似考虑。对于多线程系统也需有类似问题

的考虑。一般采取的办法是临界区(critical area)的思路。单片机单线程开发也可简化处理，直接设置一个全局变量进行互斥处理。具体实现细节如需要可联系我们进行交流。

小提示 5：通常单片机的中断处理时间应尽量短。这有助于提高单片机系统主循环处理的及时性，降低单片机整机运行的不确定性，从而提高单片机整机系统的可靠性。

4.10.2 利用数据中断 IO 管脚来辅助诊断和优化单片机主程序的技巧

在 ALK8266WIFI® 模组上，IO0 管脚的输出，同时还通过 100Kohm 电阻连接 LED1 灯（远离 PCB 天线的绿灯）的阴极。因此， LED1 绿灯平时会处于常亮状态。当 ALK8266WIFI® 模组接收到网络数据后，LED1 绿灯会灭掉，当主机读走数据完毕后，会再次点亮。

在正常接收的情况下，因为单片机读取接收的速度很快，所以上图中高电平时间很短，肉眼可能看不到 LED1 灯的灭掉。但是在单步调试下，例如在下面的代码中 M8266WIFI_SPI_RecvData ()前设置一个断点，

```
if (M8266WIFI_SPI_Has_DataReceived () ==1 )
    M8266WIFI_SPI_RecvData (.....);
```

可以观察到，当跳到 ***M8266WIFI_SPI_Rcv(.....)***前的断点时，LED1 绿灯灭掉了（因为 ***If (M8266WIFI_SPI_Has_DataReceived() ==1)***为真，表明接收到了数据，模组的 IO0 管脚输出了高电平）；单步执行完毕 ***M8266WIFI_SPI_RecvData (.....)***后，可以看到 LED1（绿灯）再次点亮（因为数据接收数据被主机取走，IO0 返回低电平输出）。

如果在单片机全速执行的过程中，偶尔会看到 LED1 绿灯灭掉，则表明你的单片机主机没有及时地去取走模组所接收到的网络数据。例如，如果在单片机的大循环中，采查询接收方式 *"If (M8266WIFI_SPI_Has_DataReceived () ==1)"*判断是否接收到网络数据（有数据就会读走），与此同时，如果单片机程序的大循环中有其他任务执行时间太长，导致执行到这个判断处需要等待较长的时间，则可能出现肉眼可以观察到 LED1 绿灯灭掉或闪烁的情形。此时，建议改用中断接收方式，以便及时地取走所接收到的网络数据，或者优化大循环中其他部分的程序执行时间。

4.10.3 中断触发信号的连续触发间歇的调节，以及管脚配置更改

参见 4.10.1 中断方式接收的管脚输出和时序图小提示 2，这里有一个连续触发的间歇时间，缺省为 50 微秒，兼顾了大多数单片机系统的 IO 辨别处理性能。可以根据单片机的性能对这个时间进行调节优化。请参考《ALK8266WIFI 模组 SPI 接口高速通信使用与集成—主机驱动 API 函数》中章"4.8.11 M8266WIFI_SPI_Module_Config_Wifi_Rx_Interrupt_Trigger"。如果不确定该如何设置，可以不配置直接使用缺省的配置。

在缺省情况下，在收到数据包时，会在 IO0 管脚上输出上升沿来触发主机的外部中断，提高主机接收处理的实时性。这个信号也可以配置到模组的其他管脚 IO2、IO4 或 IO5 上来进行输出。如果只采用查询方式不需要这个中断触发输出功能，也可以直接禁止改管脚的输出。参看请参考《ALK8266WIFI 模组 SPI 接口高速通信使用与集成—主机驱动 API 函数》中章"4.8.11 M8266WIFI_SPI_Module_Config_Wifi_Rx_Interrupt_Trigger"。当禁止该管脚的中断输出触发功能后，该管脚可以作为普通的 IO 管脚功能供用户使用，此时 WIFI 模组会因为不再提供接续触发的延时，而接收性能会有略微提升。由于通信系统是一个多环节配合的

系统，如果对这里进行了配置，建议进行充分的测试评估。如果不确定该如何设置，可以不配置直接使用缺省的配置。

4.11 低功耗

值得澄清的是，“模组的低功耗”应该是在确保有用发射功率前提下的低功耗。有用发射功耗会影响发射的距离和性能。如果一味追求低功耗，可能会导致发射功率降低，从而带来通信发射性能大大降低，例如一些低功耗蓝牙设备的通信距离的限制。因此，有意义的低功耗，应该是在确保足够传输距离和性能前提下的低功耗，即，尽可能消除非通信周期的功耗、尽可能消除有用发射功率之外的热功耗。

本模组支持有实际使用价值的低功耗实现。

4.11.1 支持根据实际的传输要求，调试发射功率，避免功耗浪费

一般说来，发射功率越大，则传输距离和通信效果会越好，所以，大多说模组会追求满功率发射。但是在某些应用场合，可能并不需要太远的传输距离或者太高的传输性能，而是够用并保持一定的裕量就可以了，如果此时依然使用满功率发射，会造成功耗的极大浪费。

ALK8266WIFI® 模组支持发射功率的可调节，从而支持最大效率的使用发射功耗。具体请参考《ALK8266WIFI 模组 SPI 接口高速通信使用与集成—主机驱动 API 函数》中章节 4.7.1 M8266WIFI_SPI_Set_Tx_Max_Power”部分。

如果模组的工作模式包含有 AP 模式（AP Only 或 AP+STA 模式），就会周期性对外发射 Beacon，这个也会造成显著的周期性发射功耗。可以根据具体应用，适当拉长发射的周期，来节省功耗。具体请参考《ALK8266WIFI 模组 SPI 接口高速通信使用与集成—主机驱动 API 函数》中章节 4.2.17 M8266WIFI_SPI_Config_AP_Param 中设置参数 AP_PARAM_TYPE_BEACON_INTERVAL。如果不必要，可以彻底关闭 AP 模式。

4.11.2 支持模组的深度休眠和自动唤醒

具体请参考《ALK8266WIFI 模组 SPI 接口高速通信使用与集成—主机驱动 API 函数》中章节“4.7.2 M8266WIFI_SPI_Sleep_Moduler ”部分。

4.11.3 支持模组的断电使能 EN

模组上有一个 EN 使能管脚，内部上拉。可以通过拉低该管脚，将模组主芯片拉低到准断电状态，可以极大地降低模组的功耗到毫安级别以下。这一点对于那些不需要持续通信的应用场合，可以极大的降低系统的整体功耗。

4.11.4 单片机外部控制模组 VCC 供电断电

使用一个 MOSFET 开关，可以控制给模组的供电断电，最大化降低模组所消耗的功耗。

4.11.5 WIFI 模组的低功耗处理的相关问题

在做 WIFI 模组的超低功耗处理的时：

1、如果模组长期开启，则尽量不使用 AP 模式，可以减少不必要的热点信号周期性发射

2、如果模组作为 STA 模式工作，采用静态 IP 地址，可缩短退出休眠重新建立通信的过程。所有的 WIFI 模组从休眠状态退出重新连接通信的过程中，都会绕不开一个连接 WIFI 并获取 IP 地址的过程，包括密码认证以及 DHCP 获取 IP 地址。因此，启动退出休眠后模组重连开始通信的时间一般都会有几秒甚至十几秒（取决于路由器密码认证和 DHCP 的时间，尤其是后者，时间往往较长）。因此，采用固定 IP 地址的方式，有助于将退出休眠并再次建立链接开始通信的时间，缩短到秒级。

请参考《ALK8266WIFI 模组 SPI 接口高速通信使用与集成—主机驱动 API 函数》中章节“4.2.5 M8266WIFI_SPI_Config_STA_StaticIpAddr”部分，了解如何设置模组的静态 IP 地址。

可以参考《ALK8266WIFI 模组 SPI 接口高速通信使用与集成 常见问题》中章节“4.1.2 如何缩短 STA 连接 AP 或路由器的时间？“部分，了解联网时间长短的因素和加快连网速度的策略。

4.12 板载 LED 灯及其诊断显示

4.12.1 启动时两个 LED 灯先交替闪烁

无论是上电复位、硬件复位、还是软件复位带来的模组启动，在启动初始化完毕后，模组板载的两个 LED 会以 5Hz 的频率交替闪烁 10 次，然后两个灯都保持常亮，或 LED1 灯（绿灯）常亮而 LED2 灯（黄灯）闪烁。如果启动时看到了两个 LED 灯交替闪烁，表明模组正确和成功地启动了，可以开始正常通信。否则，则表明模组启动异常或者启动失败。

4.12.2 工作时 LED1 灯（绿灯）代表网络数据的接收状态

LED1 灯（绿灯）一般会常亮。

当使用 SPI 通信时，如果接收到来自网络的数据，该灯会灭掉；当主机读走数据后，该灯会重新点亮。

由于接收数据并取走的速度很快，所以正常工作时，一般很难看到该灯会灭掉，但是在单步调试时，可以看到该灯的灭掉和开启。

详情可参看 4.10.1 接收中断的管脚输出和时序图和4.10.2 利用数据中断 IO 管脚输出来辅助诊断和优化单片机主程序的技巧的相关说明。

另外，如果模组收到了数据，但是主机没有读取取走，该灯也会一直灭掉，这个特点可以用来对产品进行诊断。

4.12.3 工作时 LED2 灯（黄灯）代表网络接入状态

如果模组工作在 STA 模式下，当模组试图去接入热点或路由器时，LED2（黄灯）会一直快速闪速，直到连网成功后，再保持长亮。

因此： (1) 在启动初期，如果之前模组接入过热点或路由器而保存了 SSID 和密码，模组一般会自动去接入该热点或路由器，因为在 4.12.1 启动时两个 LED 灯先交替闪烁启动结束后，LED2（黄灯）也会一直快速闪烁，直到连网成功，并保持常亮。（2）如果在工作期间，突然发现 LED2（黄灯）开始闪烁，则表明模组联网掉线了，正在自动重新连网。

如果模组工作在 AP 或 STA+AP 模式下，LED2（黄灯）会一直会闪烁，闪烁的频率和模组 AP 的 beacon 周期相对应。因此，在启动后期，在两个 LED 灯交错闪烁 4.12.1 启动时两个 LED 灯先交替闪烁后，如果模组的工作模式包含有 AP 模式，则 LED2（黄灯）会一直闪烁。

4.12.4 两个 LED 灯同时灭掉表示模组工作异常

模组在正常工作时，两个 LED 灯一般不会同时长时间处于灭掉状态。当遇到这种情形时，一般表示模组处于异常工作状态。

遇到这种现象时，建议尽快检查模组上的 VCC 和 GND 等管脚的电压是否正常。如果电压异常，请尽快切断主板对模组的供电，以保护模组。

如果检查后发现电压正常，重新初始化后发现模组可恢复正常工作，则一般是因为模组在前面的工作中遇到了掉电复位或 nRESET 管脚复位，因为在这类情形导致的意外复位出现时，nCS 管脚往往处于高电平状态，而导致模组复位后进入其他启动模式，参看 4.2 模组的硬复位。此时建议用示波器观察模组正常工作中的 VCC 管脚的供电稳定性以及 nRESET 管脚输入信号是否存在毛刺。在一些电源完整性设计不佳的主机板上，尤其是在模组持续发射需要消耗较多电流的过程中，可能会因为主板稳态和动态供电不佳，而出现掉电复位。在某些强干扰的场合，也可能引入传导或辐射干扰，导致 VCC 供电不稳或者 nRESET 引入噪声造成振荡而导致复位。

5 和主机的集成--固件

5.1 模组在系统架构中的逻辑位置示意图

ALK8266WIFI®高速通信模组，在逻辑上实现了“**SPI 从机接口和 TCP/UDP 包之间的互转**”，如下图所示意：

对于这一点的理解，可以参照常见的 “UART 串口和 TCP/UDP 包之间互转”的普速通信模组来对照理解：两者在逻辑架构上基本相同，区别在于使用 SPI 接口可以得到更高的吞吐速率。

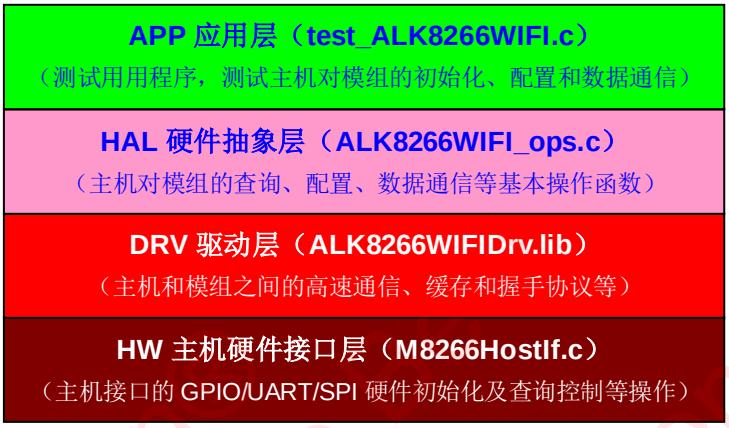
5.2 层次化结构设计和宏控制

5.2.1 层次化结构设计和硬件抽象

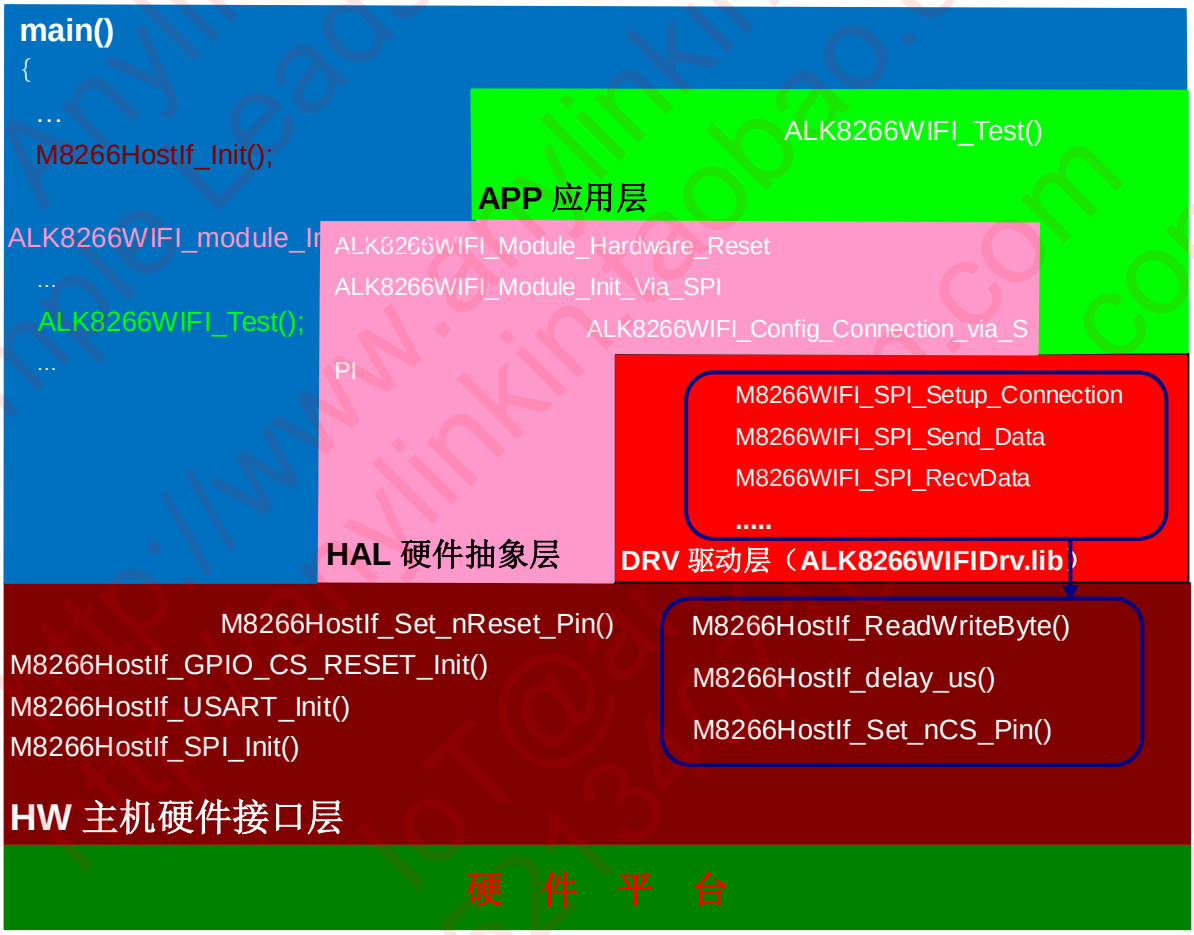
我们提供一套 ALK8266WIFI® 模组和主机集成的参考代码和例程，整个固件集成系统采用层次化结构设计，以方便跨平台移植。其示意图如下。

其中，只有“HW 主机硬件接口层”的代码实现依赖于硬件平台，DRV 驱动层、HAL 硬件抽象层、APP 应用层，在代码实现上，或者只通过调用“HW 主机硬件接口层”内部的函数，来实现其功能，或者本身与底层硬件无关，做到了与平台硬件的无关性操作。

层次化设计规范和硬件抽象层的存在，有利于程序的结构化实现、提高程序开发的速度和可靠性，也有利于程序的移植。例如，当调整 ALK8266WIFI® 模组和主机之间的连线方式时，只需对 HW 主机硬件接口层做相应的修改；当从一个平台迁移到另外一个平台时，上层代码可基本不用更改。



5.2.2 各层次的核心调用关系示意图



5.2.3 宏控制定义文件 brd_cfg.h

此外，如果配合宏定义控制的使用，能更加有利于实现在不同硬件连线的跨平台之间快速移植。在我们提供的例程包中，有一个文件 brd_cfg.h 定义了平台相关的宏定义，协助方便的实现跨平台移植。

5.3 和单片机实现快速集成和验证的步骤

5.3.1 快速集成的步骤

1、根据硬件调整主机接口初始化函数和基础读写实现函数

根据硬件互联方式、使用模式、及主机芯片，调整 brd_cfg.h 的宏定义或/和调整 M8266HostIf.c 中相应的主机底层函数实现和初始化。

主要是《ALK8266WIFI 模组 SPI 接口高速通信使用与集成—主机驱动 API 函数》章节“*3 M8266HostIf.c 主机接口源码函数说明*”的几个函数的相应调整或实现。

2、将 M8266WIFIDrv.lib 或.a 驱动库及其头文件添加到项目工程中。

3、SPI 主机接口通信的逻辑功能性验证

单次调用《ALK8266WIFI 模组 SPI 接口高速通信使用与集成—主机驱动 API 函数》章节“*4.1.2 M8266WIFI_SPI_Interface_Communication_OK*”函数，验证主机接口初始化以及基本字节的读写的逻辑。详情参看 5.4.2 验证主机 SPI 接口初始化逻辑是否正确的方法。

4、SPI 主机接口高速通信的压力测试和验证

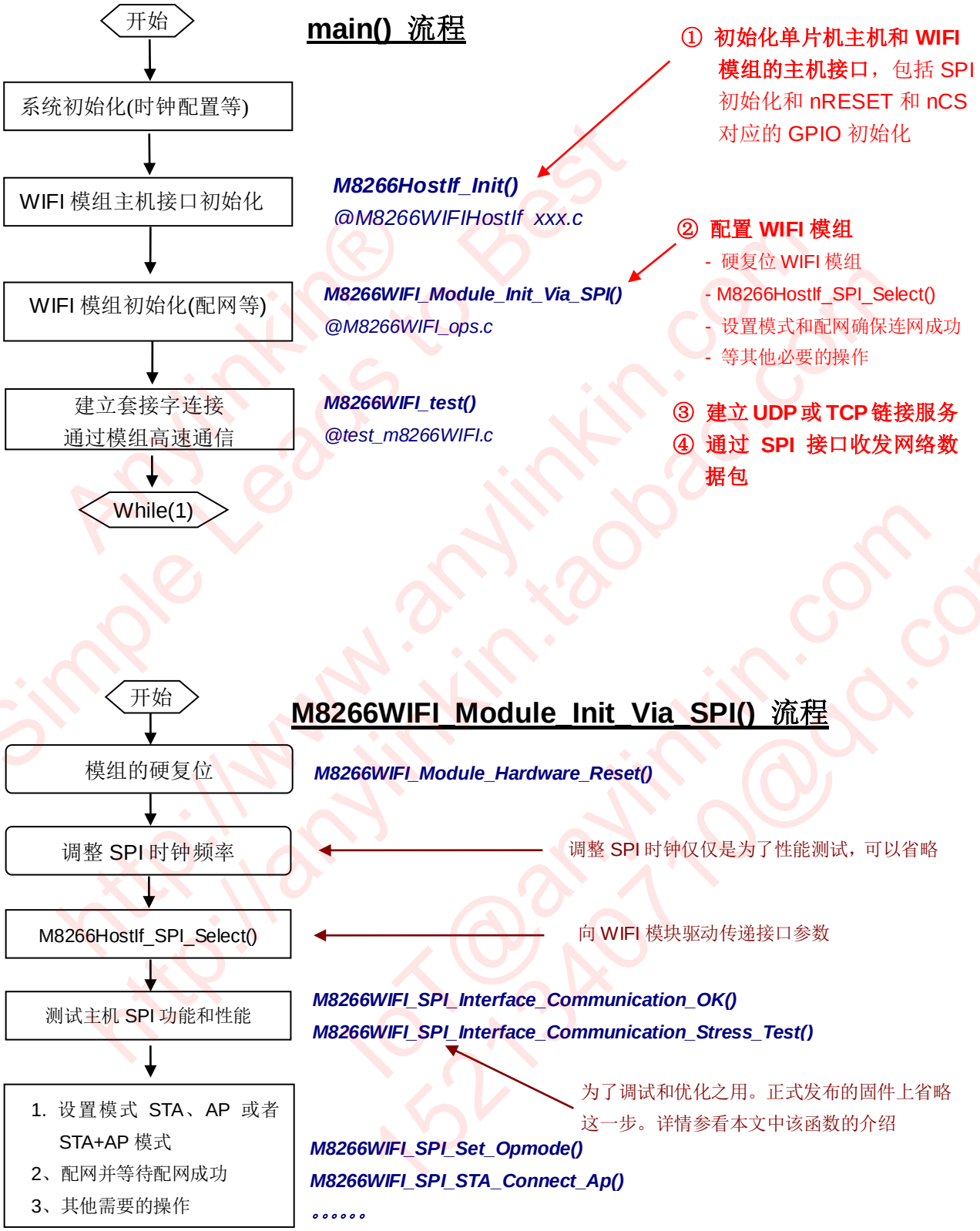
调用《ALK8266WIFI 模组 SPI 接口高速通信使用与集成—主机驱动 API 函数》章节“*4.1.3 M8266WIFI_SPI_Interface_Communication_Stress_Test*”函数，获取并验证主机板 SPI 接口可以稳定可靠通信的最大时钟频率。详情参看 5.4.3 验证/获取主机板 SPI 接口稳定可靠通信的最大时钟频率。使用可靠通信的最大 SPI 时钟频率，有利于在现有系统上最大化通信的吞吐率。

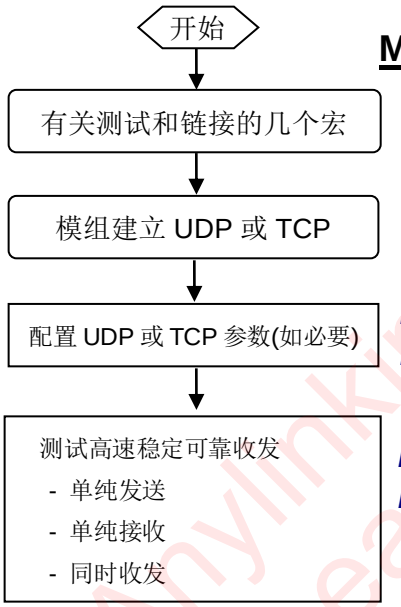
5、加入 M8266WIFI_ops.c 文件（这个文件里的函数代码做到了与平台无关），直接使用其中的操作函数，或者，你可以仿照其中的函数，扩展实现你所需要的功能,。

6、参考 main.c 和 test_M8266WIFI.c 这个文件里的函数代码基本做到了与平台无关）中的调用使用，完成项目的自定制。

在提供的 test_M8266WIFI.c 的测试函数体中，提供了四种测试机制：（1）模组只向远端节点连续发送数据包，来测试模组只发送的速率；（2）模组连续接收来自远端节点的数据，来测试接只接收的速率；（3）模组等待接收来自远端节点的数据包，在接收到将该数据包直接返回给发送方，来测试同时收发的各自速率；（4）多客户端测试。

5.3.2 参考例程包的流程示意图





M8266WIFI_Test() 流程

M8266WIFI_SPI_Setup_Connection ()

M8266WIFI_SPI_Op_Multicuast_Group ()
M8266WIFI_SPI_Set_TcpServer_Auto_Discon_Timeout ()

M8266WIFI_SPI_Send_BlockData ()
M8266WIFI_SPI_RecvData ()
。。。。。。

5.4 底层调试技巧—主机接口硬件接线、初始化匹配及验证技巧（选阅）

5.4.1 目的

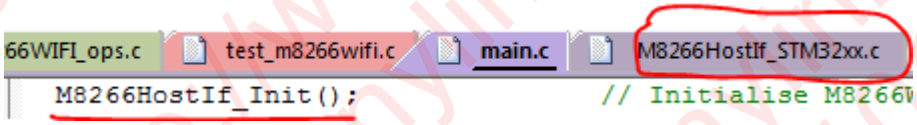
在集成的初期，需要根据硬件互联方式、使用模式、及主机芯片，调整 M8266HostIf.c 中相应的底层函数实现和初始化，包括 SPI 接口的初始化。确保主机 SPI 接口的初始化正确，且和 ALK8266WIFI 的 SPI 接口配置相匹配，这一点对于上层固件的通信非常重要。尤其是对于某些跨平台移植（我们没有提供相关的参考包），首先要确保底层主机接口和 SPI 字节基本读写通信是正确的，且可靠的。一旦与底层硬件有关的 SPI 通信，在功能和性能上通信调试验证可靠了，上层与硬件无关的应用实现就简单了。

所以，我们写了本章文字以供调试参考，主要是涉及硬件底层 SPI 接口的调试，总结了一些用户常见的问题的定位和解决。

5.4.2 第一步，检查初始化代码中，主机接口定义与初始化和实际接线是否一致正确

主机 SPI 接口的初始化，除了基本的 SPI 初始化寄存器配置之外，还必须满足 4.1.2 SPI 主机接口的固件初始化中的要求，即主机模式、模式 A、高位优先发送。

因为我们提供了层次化和验证好初始化代码，所以，这里一般不需要用户去做太多修改，大多数情况下，只需要用户接好线，并让接线和初始化代码里所使用的 GPIO 和 SPI 管脚对应上，就可以了。所以移植和调试相对比较简单，许多用户收到我们的模组和拿到我们的例程包之后，大约 2-3 个小时，就完成了全部的高速通信和测试体验。



根据我们的技术支持经验，我们也建议大家采用**“单步调试方法”**，来核对自己的接线和代码里是否保持一致，即在 main()的初始化阶段，单步执行进入这个函数，或者右键单击，跳入这个函数，一步步查看其中的宏定义配置是否和自己的接线一致。

5.4.3 第二步，验证片选和复位管脚对应 GPIO 的连接和初始化

可以使用相关的仪器例如万用表接触模组上的相应管脚和单片机主机对应的管脚，测试连通性。但我们这里也提供一种通过代码调试的方式，来检验此处的连接和初始化是否正确。

方法 1：通过单步执行 M8266WIFI_Module_Hardware_Reset()测量电压来检查 nCS 和 nRESET

在初始化阶段，会执行 M8266WIFI_Module_Init_Via_SPI()函数，在这个函数的最初，会调用一个函数 *M8266WIFI_Module_Hardware_Reset()*，该函数的具体实现如下：

M8266WIFI_ops.c

```
/*************************************************************************************************************************
 * 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, 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(5);          // delay 1ms, adequate for nRESET stable.
                                           // give more time especially for some board not good enough

    M8266HostIf_Set_nRESET_Pin(1);         // Pull high again the nReset Pin to bring the module exiting reset state
    M8266WIFI_Module_delay_ms(300);        // at least 18ms required for reset-out-boot sampling boottrap pin
                                           // Here, we use 300ms for adequate abundance, since some board GPIO,
                                           // needs more time for stable(especially for nRESET)
                                           // You may shorten the time or give more time here according your board v.s. effiency
    M8266HostIf_Set_SPI_nCS_Pin(1);
  //M8266WIFI_Module_delay_ms(1);          // delay 1ms, adequate for nCS stable

    M8266WIFI_Module_delay_ms(800-300-5-2); // Delay more than around 500ms for M8266WIFI module bootup and initialization
                                            // including bootup information print.
}
```

可以看到，该函数会依次对 nCS 和 nRESET 管脚输出高低电平，所以，我们可以在单步调试的模式下，测量执行每一步时对应于模组上相应管脚的电平是否为（到轨 to-rail），即能否输出到 0.3V 以下（对于输出 0）或 3.0V 以上（对应输出 1），来判断我们的接线和初始化是否正确。

方法 1：通过执行 M8266WIFI_Module_Hardware_Reset()观察模组上的 LED 等闪烁来验证 nCS 和 nRESET

如果 nCS 和 nRESET 的管脚连接和 GPIO 的初始化都是正确的，那么在执行完毕函数 *M8266WIFI_Module_Hardware_Reset()*后，可以观察到模组正常启动的现象，即：首先会看到模组上的两个灯先走马灯交替闪烁几下，然后绿灯维持长亮，黄灯或持续闪烁或长亮（参看 4.12.1 启动时两个 LED 灯先交替闪烁章节）。

所以，如果执行完毕函数 *M8266WIFI_Module_Hardware_Reset()*后，可以观察到模组正常启动的现象，那么表明 nRESET 工作正常。

但是此时，并不能表明 nCS 也一定是正常的（因为 nCS 在单片机主机侧也可能是悬空的，或者接上了另外一个处于输入状态悬浮的 IO 管脚，此时模组上的下拉电阻会起作用）。

考虑到让模组能正常复位，必须将 nCS 预先拉低再复位（参看 4.2 模组的硬复位时序），所以，在确保 nRESET 能正确复位前提下，可以采用反证法，来验证 nCS 也是正确，即：进入 *M8266WIFI_Module_Hardware_Reset()*，临时将起始的 nCS 设置改为输出 1，如下图红色圈住的 0 改成 1，编译后再执行该函数。如果此时模组不能正常复位，则表明 nCS 之前的接线和设置也都正常了。临时反证结束后，再将下面红色圈住的部分恢复正常（恢复为 0）。

M8266WIFI_ops.c

```
/*************************************************************************************************************************
 * 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, 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(5);           // delay 1ms, adequate for nRESET stable.
                                            // give more time especially for some board not good enough

    M8266HostIf_Set_nRESET_Pin(1);          // Pull high again the nReset Pin to bring the module exiting reset state
    M8266WIFI_Module_delay_ms(300);         // at least 18ms required for reset-out-boot sampling boottrap pin
                                            // Here, we use 300ms for adequate abundance, since some board GPIO,
                                            // needs more time for stable(especially for nRESET)
                                            // You may shorten the time or give more time here according your board v.s. effiency
    M8266HostIf_Set_SPI_nCS_Pin(1);
  //M8266WIFI_Module_delay_ms(1);           // delay 1ms, adequate for nCS stable

    M8266WIFI_Module_delay_ms(800-300-5-2); // Delay more than around 500ms for M8266WIFI module bootup and initialization
                                            // including bootup information print.
}
```

5.4.4 验证主机 SPI 接口初始化逻辑是否正确的方法

在初始化 SPI 接口后，对模组进行复位（目的是确保模组也处于初始化后的初始状态），然后单次或单步调用函数 *M8266WIFI_SPI_Interface_Communication_OK()*，参见《ALK8266WIFI 模组 SPI 接口高速通信使用与集成—主机驱动 API 函数》章节“*4.1.2 M8266WIFI_SPI_Interface_Communication_OK*”。如果这个函数返回正确，那么，表明主机的 SPI 初始化是正确的，且和模组的 SPI 接口方式匹配，且基本字节读写的逻辑也都是正确的。

5.4.5 验证/获取主机板 SPI 接口稳定可靠通信的最大时钟频率

因为单片机主机板布线质量、接口引线或插槽等原因，SPI CLK 频率不一定为单片机理论所支持的最大 SPI CLK 频率，稳定可靠的连续高速读写的最高 SPI CLK，也不以定为单次读写成功的最高 SPI CLK。而高速 SPI 通信需要找到实际的逻辑互联（布线或连线）条件下，可靠通信的最大 SPI CLK 频率。

如果按照 5.4.2 验证主机 SPI 接口初始化是否正确的方法单次调用函数 *M8266WIFI_SPI_Interface_Communication_OK()* 返回成功，然后调用 *M8266WIFI_SPI_Interface_Communication_Stress_Test*()函数，参见《ALK8266WIFI 模组 SPI 接口高速通信使用与集成—主机驱动 API 函数》章节“*4.1.3 M8266WIFI_SPI_Interface_Communication_Stress_Test*”，观察返回成功次数。如果成功的接近 100%（最好是百分之百成功），则表明主机板的 SPI 接口在该频率下可以稳定可靠地通信，可以提高 SPI 时钟进一步测试。否则，主机板的 SPI 时钟可能需要适当降低。

5.4.5 提供验证过例程的单片机列表

STM32 等平台的主机驱动，已在基本直接读写方面进行了优化验证，如下列表所示。其他平台，请联系我们支持优化。

单片机供应商	单片机主机型号系列
意法半导体 (STMicroelectronics)	1、STM32 系列，如： STM32F0xx、STM32F1xx、 STM32F4xx、STM32F7xx、 STM32L1xx、STM32L4xx、 STM32H7xx 等系列 2、STM8 系列
恩智浦(NXP) 飞思卡尔(Freescale)	1、 LPC11xx/LPC17xx/ LPC55/LPC43XX 系列 2、 K27/K28 系列 3、 iMx RT105x/RT106x 系列 4、K60/KV58 系列
德州仪器(TI)	1、 MSP430 系列 2、 C2000 系列，例如 TMS320F28335
北欧半导体(Nordic)	1、nRF52 系列
台湾盛群(Holtek)	1、HT32 系列，如：HT32F16xx
台湾华邦新唐(Nuvoton)	1、NUC123 系列 2、NUM451 系列
珠海杰理科技(Jieli)	1、AD200
兆易科技(GigaDevice)	1、GD32 系列
灵动微电子(MindMotion)	1、MM32 系列
华大半导体(Huada)	1、HC32 系列
C8051 系列	1、C8051F120 系列 2、C8051F340 系列
更多值得期待……	

6 单片机系统产品化前的后期处理和注意事项（选阅）

在我们与客户技术交流（包括技术支持和协助客户使用好高速 WIFI 模组解决方案）做好产品的过程中，发现一些客户在产品的最后阶段，容易疏忽对产品化前的后期处理，并最终让其利益受损。以下几点和高速 WIFI 模组方案本身关系不大，但可能牵涉到客户本身利益的一些产品化后期处理，在此也列举出来，供大家参考。

6.1 单片机系统未使用管脚或功能的确定性设置、关闭或禁止

一些客户开发单片机嵌入式系统产品，习惯于简单基于芯片原厂的工程包参考样式，或者直接使用 CUBEX 等自动化生成代码的工具，这虽然极大地减少了开发新项目的工作量，但同时，这些参考样式或自动化生成工具，难以面面俱到或完全适配实际的应用特征，有些不适配具体产品特征的残留的初始化和配置，可能容易遇到一些运行上的稳定性问题，往往需要查好长时间才会被发现。

因为简单参考或直接自动化生产代码工程基础包时，常见的一些问题包括但不限于：

（1）一些参数（例如堆栈）的不合适设置
（2）不必要的 IO 的标准化初始化和使能，带来的管脚冲突。
（3）不必要的功能子模块的初始化和使能，带来的功能子模块之间的冲突。

举例说明：曾遇到一个公司，以原厂标准例程包为起点做新项目，开发阶段产品测试很稳定，但批量生产时约有 30%产品在出厂检验时出现稳定性问题，部署运行 1 周后，又有约 20%产品出现稳定性问题，这些稳定性问题的出现也无任何规律。开始以为是元器件或焊接组装不善的结果，但折腾了数月，最后定位到某个 IO 管脚的非适配性不正常的设置所致，而该设置来自原厂的缺省设计，而合作伙伴的主板硬件和原厂并非完全一致。

实际上，嵌入式系统很强调定制化最优。所以，如果可以不用自动化代码生成工具，就应尽量避免。至少在产品化处理阶段，需要对每个 IO 或模块等进行逐一梳理，以消除潜在风险。

6.2 单片机固件的防盗版处理

主板硬件本身的防盗抄板可能不太好控制，但是可以通过固件的防盗版措施，降低产品被盗版的概率。尤其是当产品面对的是准入门槛不高的大众化市场时，固件的防盗版处理，是产品化处理值得考虑的非常重要的一个环节。有以下几点建议供参考：

（1）做好生产文件的保密工作

外包工厂往往是泄露的主要来源，这些工厂的制度管理并非完善，人员流动性可能也会比较大，因此，即使签署过保密协议，也可能不完全可靠。所以不要轻易把“最终固件”显性地交给外包生产工厂。最好尽量将最终固件留给自己来烧录，而交给工厂生产的只是用于测试和加载最终固件的中间版临时固件；或者单独定制烧录工具，最终固件内嵌于定制烧录工具来交给工厂进行烧录，也便于控制生产的数量和不对外泄漏最终固件。

（2）产品出厂前需锁住单片机相关熔丝位

有一些单片机（例如 STM32）本身有熔丝位可以锁住单片上的固件，不让从外部读取，所以在烧录固件后的最后一步，可以锁住那个熔丝位。

（3）对固件进行加密或签名，与主板/关键芯片绑定

绝大多数单片机或者主板上的其他关键（加密）芯片，具备全球唯一序列号，且该序列号不可更改，因此可以使用这个序列号来对固件进行加密、签名、与主板/关键芯片进行绑定。这样处理后，即使单片机固件被读取出来了，但因为对固件进行加密或签名，与主板/主芯片进行了绑定，所以也无法用来进行批量生产。签名可以在生产烧录的最后阶段完成。

以上只是单片机固件最基本的防盗版措施，避免单片机“裸奔”，提高盗版的难度或成本。所谓“道高一尺，魔高一丈”，反之亦然。如有进一步防盗版需求，可以联系我们深入交流和探讨。

6.3 单片机固件的 OTA 升级功能支持产品迭代

一般说来，产品化后往往存在迭代的需求。例如，产品化初期往往可能依然会存在一些潜在的 BUG 或值得优化的地方，或者，一段时期后客户那边可能会出现一些新需求，这些可能可以通过固件优化来实现。于是，更新单片机固件显得非常必要。

对单片机固件提供 OTA（Over The Air）升级功能，可以支持对单片机固件的更新升级，同时大大降低产品的后期运行维护成本。既然单片机系统上已经加上了 WIFI 无线通信功能，我们建议考虑对单片机固件系统增加 OTA 升级功能。

单片机固件的 OTA 可以支持在本地升级，也可以通过远程实现 OTA 升级。如果单片机系统产品不方便连上互联网，那么可以通过手机 APP（内嵌单片机升级代码）和单片机 WIFI 系统，在本地通信进行固件升级。而用户可以通过更新手机 APP，来获取更新的单片机固件。如果单片机可以连上互联网，那么产品供应商可以自建升级服务器，单片机通过连接服务器来获取升级代码进行升级。

单片机固件的 OTA 升级功能，需要考虑升级通道的加密、代码校验、和升级过程中单片机新固件烧录过程中的可能出现意外的容错处理。升级通道加密的目的是为了防止被窃听截取，可根据具体应用场景酌情考虑。代码校验是为了确保下载下来的代码的合法有效性，确保烧录代码的完整性。容错处理，一般采用启动代码+双固件备份的方式，即启动代码本身不可以更改，由它完成初步的初始化和升级操作（包括是否需要升级以及完成升级），并决定启动后选择执行哪个固件。

如需要进一步的 OTA 升级方面的技术交流和探讨，可随时联系我们。

6.4 整机产品的可诊断可维护性设计

由于现场条件的多样性难以完全覆盖，以及某些整机产品本身的缺陷，所以，即使经过实验环境下充分测试的产品，在用户现场依然可能会表现出异常，而需要进行诊断和维护。用户的非专业性描述，给远程提取现象分析定位造成困难，而现场支持维护，往往会造成供应商

的成本升高，周期拉长，并可能给客户造成不佳的主观体验。因此，可诊断性可维护性设计，是产品设计和开发人员在产品全周期（主要是产品的开发和验证阶段）都会考虑的方面。

产品整机的可诊断性设计，首先需要构造一种简洁直观明确的的状态或异常规划。例如，我们常用的 LED 灯的不同的点亮组合或闪烁方式、LCD 或串口输出的自定义状态码，都可以用来告诉我们，是否或者具体是哪一步的初始化或者操作出现了何种异常。这样，当现场出现某种异常时，现场的维护人员或者用户，只需要简单地通过具体的闪烁模式或者状态码，就可以协助快速定位。

产品整机的可诊断性设计，其次需要合理选取状态和异常报告的方式和层级，以便更适合自己的产品形象和状态信息的提取。例如，状态灯的闪烁，可能只适合用于简单初略的提示，不适合复杂和频繁的使用，否则用户可能会感到诧异。又例如，有一些状态码，可以通过简单操作或者直接提示给用户然后由用户自行处理或转达，有一些状态码，则适宜现场维护人员或者通过网络后台提取。

在使用串口打印相关诊断信息输出时，应做好分级打印。可以参考下面的方式自定义一个简易的串口打印权限层级打印输出的封装：

```
#define my_printf1(right,format, ...)          do {          \
                                               if(right>=1)  \
                printf(format, ##__VA_ARGS__);               \
} while(0)
```

这样，只有当传递的 right 大于 1 时，才会打印输出。

或者，也可以类似 Linux 里的 DEBUG LEVEL 的打印方式，设置一个全局变量 current_print_level 来指示当前允许的打印输出的层级

```
#define my_printf1(format, ...)          do {          \
                                         if(current_print_level>=1)  \
                printf(format, ##__VA_ARGS__);         \
} while(0)
```

那么，只要将这个全局变量数值设置为大于等于 1，所有调用 my_printf1 的代码都会打印输出。依次类似可以定义 my_printf2，my_printf3，….，从而可以通过简单地控制 current_print_level 决定某个层级的信息是否值得通过串口输出。

ALK8266WIFI®模组作为一个产品，也践行着可诊断性的设计理念，提供了方便客户诊断的相关机制。具体表现在如下几个方面，客户可充分利用：

（1） 合理安排的 LED 等闪烁机制

例如，上电阶段会看到双灯先交替闪烁几下，然后看到绿灯长亮黄灯闪烁或长亮，这就表明 WIFI 模组正常启动了没有出现异常；如果看到模组的黄灯快速闪烁，表明 WIFI 模块可能在试图连接路由器也就是出现连接不上或者意外掉线；如果通信的过程中发现绿灯灭掉，则表明 WIFI 模组收到了数据但是单片机没有及时读走，等等。详情可参看 4.12 板载 LED 灯及其诊断显示。

（2） 配套的 WEB 网页协助查看和快速诊断

网络开发和应用的实战中，可能会出现一些通信故障但是表现现象却很浅显而需要

进一步深挖。例如一个问题“突然发不了数据”可能对应一系列底层的原因，其中包括 WIFI 断开或者套接字断开。这个时候，可以打开 WIFI 模组上自带的 WEB 网页，查看当前的 WIFI 状态、套接字信息、IP 地址和端口等等，查看哪里出现异常，以便进一步缩小定位。此外，WEB 网页里也提供了一个信号扫描的功能，可以用于验证或者分析通信信号的强度及可能相关联的一些问题。

但是 WEB 网页等友好性用户接口也可能会造成安全性问题，所以，我们建议采取需要使用时才临时开启的方式，详情参见 6.5 考虑产品的安全性，包括关闭高速 WIFI 模组自带 WEB 网页。

（3） 一整套的 API 函数异常返回的状态码和故障原因查询函数，协助客户缩小定位

我们提供的驱动 API 函数，基本都包含有 status 的状态码指针形参，用于在出现异常时返回一些状态码，来协助我们诊断。建议客户利用好该特性。

6.5 考虑产品的安全性，包括关闭高速 WIFI 模组自带 WEB 网页

高速 WIFI 模组上的 WEB 网页让用户操作变得更加简单方便，但也可能因简单方便会成为一个潜在的隐患。例如，若有好事者打开模组上的 WEB 网页后，修改模组所连接的路由器，则可能导致正在通信的套接字被断开，而让产品处于异常状态。尽管 WEB 网页有操作权限密码也可以修改，但是权限密码对于网页操作的限制保护能力也是有限的。

高速 WIFI 模组支持模组上的 WEB 网页的开启和关闭。WEB 网页的打开和关闭状态，可以当前不保存的，也可以是缺省保存的。保存后，每次启动时会根据保存的参数，来决定是否自动开启 WEB 网页。

因此，我们建议在产品化前做好相关处理。比如，平时都关闭模组上的 WEB 网页，只在需要使用 WEB 网页时通过 TCP 通信接口或单片机 SPI 接口来打开 WEB 网页。在使用完毕后，立刻关闭。

此外，批量采购我们的模组，可以要求我们将出厂配置为缺省状态为 WEB 网页处于开启还是关闭状态（即使出厂时缺省为关闭状态的，客户也可以通过 API 函数临时或永久打开或关闭 WEB 网页）。

实际上，对于操作简单但是牵涉到关键领域的操作，产品设计和开发人员尤其要慎重对待。

6.6 元器件国产替代的分析与验证，IDE/EDA 工具的免版权化

随着我国芯片工业和 EDA 产业等领域的国产化推进，以及基于大环境、成本、周期等综合因素的考虑，我们建议客户在产品化阶段，增加元器件的国产替代分析和充分验证，在确保产品质量的前提下，尽量选用国产元器件进行替代。考虑到国产化的阶段现状，也需要集成商进行严格地对照测试和优化规避---即使 A（芯片方案）可能会差一点，但是 B（集成）更努力做得更好一些，也可能让 A+B 的综合效果更好。

同时推荐大家可以考虑选用类似 *GCC 交叉编译* 开发环境或 *KiCAD* EDA 绘图工具进行替换。 这些工具基于 GPL 协议，一般不会牵涉到版权问题，使用也方便和简单，业界越来越多的工程师都改用这两款工具进行开发，体验效果都很满意，甚至有些地方的效果更好。

6.7 其他更多产品化的后期处理，可联系我们进一步交流探讨

7 单片机网络开发的复杂性及其测试定位与系统优化策略（选阅）

网络通信系统是一个相对比较复杂的系统，其开发往往会牵涉到比较多的方面，多因素的环境和测试条件中，如果有一个或多个因素或条件发生了改变，往往会带来系统整体效果和性能的显著改变。

因此开发单片机高速 WIFI 通信，往往需要开发人员具备足够的单片机开发和调试基础，熟练掌握单片机开发基本功和调试技巧，也需要熟悉网络通信的基础概念和常见经验。

对于相对复杂的系统，或者经验相对欠缺的开发人员来说，采用对照分析法来逐步缩小定位，则往往是一种高效和值得推荐的方法。对于经验相对丰富的开发人员，根据经验大胆假设，然后构造实验来对照测试和小心求证，也是值得灵活掌握的分析方法。

本章节主要源自我司技术人员多年的开发和定位技巧总结，以及对客户提供技术支持和协助的统计效果归纳，仅供参考，技术人员可酌情选阅读。

7.1 单片网络开发和问题定位的对照分析技巧

7.1.1 单片机高速 WIFI 通信系统的复杂性

和普通的 RS485、CAN、RF 等通信系统比较，网络通信系统是一个相对比较复杂的系统，所牵涉到的技术点会比较多，而“无线”、“单片机 WIFI”、“高速“等技术关键词，则更进一步增加了通信系统所涉及的复杂性。无线网络比有线网络更容易受到外界的干扰；较之于 TCP/IP 包装后的上层开发（比如在 Linux 应用层甚至 Java 应用层的开发），嵌入式单片机系统上的 TCP/IP 通信开发，往往牵涉到的底层也会更多；而在低性能的单片机上实现“高性能”的可靠高速通信，也会牵涉到可靠性、稳定性、以及整机系统中其他关联环节以及短板效应的分析与优化。

因此，做好单片机高速 WIFI 通信系统及其产品的开发，往往需要开发人员具备一定的技术功底。足够的单片机开发和调试技巧和功底，以及必要的网络通信概念基础，有助于开发人员将自己从一些单片机和网络通信基础性概念中解脱出来，重心放在高速稳定通信以及无线问题的定位和优化上；同时，具备足够的单片机通信基础，也有助于在分析问题时准确快递的捕获现象缩小定位。

磨刀不误砍材工，如果在这里基础或经验相对薄弱的开发人员，可通过必应或百度学习或通过单片机开发板的标准例程，打好相关基础，有助于做好单片机高速 WIFI 通信系统及其产品。

7.1.2 对照分析法和对照参考系统

为了方便客户用好高速 WIFI 模组和快速定位分析，在保持嵌入式系统开发的灵活性和高效性的前提下，并结合实用性考虑，我们的高速 WIFI 模组解决方案做了最大可能性的可控性封装，以及丰富的调试诊断策略支持。除了技术实现上的各种异常诊断提示之外，我们还提

供了一整套的测试方法，作为用户对照参考的起点，协助客户通过一步步调整因素来对照分析自己的系统上的各种问题或优化点。

对照分析法的基本思路是，如果有两套影响因素较多的系统，两者的差别只在于某个或某几个有限的因素，但是却造成了某个性能或功能的差别，那么基本上可以认为，这几个有限因素上的差别（因素本身或因素改变对其他环节的影响），就是这里性能和功能上差别的原因。于是，我们可以对这些因素进行分析来初步缩小定位分析和优化，而避免头发胡子一把抓或瞎猫撞死耗子的试凑方法，导致定位低效且结果容易偏颇或带来不确定。

这套对照参考系统，可由（1）用户自己的单片机最小系统，（2）从我司购买的高速 WIFI 模块+我司提供的例程包，（3）一台用户的电脑，所组成。这套系统的特点是，实现最简单、依赖性最小，起点最优。

我司强烈建议用户首先玩会我司所提供的这个对照参考系统，基于以下两个目的：

（1）通过单步执行例程，熟悉应用 API 和高速通信流程和要点，以方便后期移植到自己的工程里；

（2）构造一套稳定可靠高速的通信系统，作为对照参考系统，方便后期通过对照分析法来定位自己工程上的问题定位或优化入口。

7.1.3 对照分析法的具体实施

1、构造对照参考系统的起点

可以采用我司推荐的最小对照系统作为对照起点。这套系统对照参考系统的组成如下：

（1） 用户的单片机最小系统，占用 3 个 SPI 管脚和 2 个 GPIO 管脚资源

只需要单片机的 SPI 和两个 GPIO，所以对单片机资源的占用最小，从而先临时排除单片机的其他因素对通信系统的影响；

（2）缺省采用模组自带的 AP 做热点，电脑作为 STA 去连接模组

从而可以临时先排除路由器等因素对通信系统的可能影响；

（3）推荐使用测试过的 TCP 抓包测试工具

从而可以先临时排除上位机 APP 等通信对等节点应用对通信系统可能影响。

说明：

（1）上述对照参考系统的，目的是很容易地构造出一个可以实现稳定可靠高速通信的系统，作为对照分析的起点，为后面逐一向目标工程改变差别因素或条件，直到复现出较功能问题或较差的性能效果，来定位出目标工程上可能的原因（当然，并非说 WIFI 模组本身的性能需要依赖这些特殊的条件）。

（2）如果用户已经有了一套对照参考系统，既和目标系统存在所关注的功能/性能上的差别，又具备较少的测试条件因素上的差别，那么可以直接使用用户自己的对照参考系统，而不必使用这里所建议的。

2、向着目标系统，逐一改变参考系统的条件因素，直到复现出问题为止。

（1）我司提供的例程包+高速 WIFI 模组，经过我司以及上千客户所验证，可是实现稳定可靠高速的通信，有测试速度可供参考。用户一般很容易就实现高速可靠稳定的通信。万一如果用户在此处第一步都无法实现，功能上的问题，建议单片机执行找到原因，而性能上的问题，则通过更换电脑等方式，很容易做到定位。

（2）通信系统的短板梳理，从单片机、网络环境、对等节点应用等几个角度来逐一梳理。

可以画一个简单的系统图示，方便一一梳理。梳理时，可以参考优选法的思路，即简单地从某个分割点开始，前半段完全一样，后半段不一样。如果这样对照发现性能或功能上出现了差别，则问题定位到后半段。依此继续缩小定位。

（3）举例说明对照分析的定位实现

案例 1：曾有某用户，之前有上位机软件+单片机串口 WIFI 系统通信挺稳定不丢包的，但因度慢不理想，因此计划换用我司提供的高速 WIFI 模组。结果换用我司模组后，从上位机端观察，出现了严重的丢数据现象。于是客户来联系我们。

这里的因素差别在于模组，所以产生问题的原因在于模组或模组改变后，对其他环节造成的影响。而模组的差别就是速度，所以初步分析很容易想到，可能是速度提高后，对系统其他环节产生了影响。这属于大胆假设，具体分析定位步骤是这样的：

1、使用我们建议的对照参考系统测试（使用我们推荐的网络调试助手）速度快且不丢数据，但是移植到整个系统里后就开始出现问题。这里的差别在于单片机侧固件和上位机软件。第一步缩小定位。

2、先从相对简单的上位机软件入手，使用 wireshark 在底层观察收包情形，发现当客户的上位机程序表现出丢数据时，但是通过 wireshark 在底层并未发现出现丢数据的情形。

同时，简单调整工程包里的单片机程序，换成目标单片机固件后发现，使用网落调试助手依然不丢数据，但是换成客户的上位机软件后就丢数据了。

3、经过步骤 2 后，初步定位这里的问题可能是上位机软件的处理跟不上高速通信系统了，而且也初步认为不是单片机固件的主要原因。于是客户开始调试上位机软件，原先的上位机软件是基于串口通信书写的，缓存较小而处理延迟加大。现在速度提高了后，上位机应用层出现了缓冲区溢出而丢数据的情形。

4、最后客户通过优化上位机应用而完美的解决了问题。

案例 2：某客户在办公室环境测试器通信系统，速度很快也很稳定，但是到了用户

现场，速度只有办公室环境测试的一半。经过对照分析发现，差别因素包括路由器、周围环境的信号干扰等等。于是，简单地临时改用模组做 AP 进行通信，其他条件保持不变，结果发现速度很快就上来了。从而排除了其他可能的因素，将原因定位到了路由器上。进一步分析发现，该老款路由器在接入多个节点设备时，处理性能会显著降低。用户替换了一个新款路由器后，速度就上来了。最后，用户对于此类问题结合运营的解决策略是，如果在部署时路由器测试效果不错，就用路由器；而如果测试效果较差，就改用模组做热点 AP，完美地解决了这里的问题。

7.2 网络开发和测试辅助工具的使用技巧

配合使用网络测试工具，进行网络逻辑功能及性能的测试；辅助使用 360 卫士等测速工具进行通信速度的测试和优化指导。

7.2.1 网络数据包接收和发送软件工具的使用技巧

技巧 1、可在电脑上建立相关的 TCP 或 UDP 服务，作为一个节点，来和模组进行通信测试。

这些工具，可以在电脑上建立 TCP 或 UDP 服务，通过编辑框接收、发送或保存，并统计接收和发送的总字节数、成功发送和接收的实时速度，以及接收到的具体数据值。

技巧 2、在进行逻辑开发时，可以巧妙利用广播地址(255.255.255.255)，查看模组或 APP 发送的数据包情况。

例如，现在有单片机系统通过 ALK8266WIFI 模组向某手机 APP 发送一个 UDP 数据包，数据包里的数据是自定义的数据协议。但是单片机发出后，手机 APP 却没有反应。原因可能是：（1）单片机没有发出 UDP 包；（2）单片机发出的数据包内的数据协议不正确被手机 APP 给丢弃了；（3）手机没有接收到数据包，或者手机处理出错，等原因。

此时，可以网络测试工具进行辅助定位。（1）将单片机程序中发送 UDP 包的目标地址由原来的手机 IP 地址，改为 255.255.255.255（广播地址），目标端口保持为手机的应用对应的本地端口不变；（2）然后，在电脑上打开网络测试工具程序，开启 UDP 服务，并将电脑本地的端口设置为和手机应用相同的本地端口。如果此时电脑、手机、以及单片机所连接的 WIFI 模组处于同一个网段，单片机通过 WIFI 模组发送出来的 UDP 数据就同时可以被手机 APP 和电脑上的网络测试工具接收到。因此，可以通过查看网络测试工具上是否收到数据，以及所接收到的数据是否正确，来判断或判处是否属于单片机系统这边的问题。

技巧 3：在使用这些工具接收时，连续高速通信，不要显示接收到的数据，有助于得到真实的发送速度。例如，勾选上下图中红色圈标注的部分。

这是因为，这个在高速接收的前提下，这个软件需要以更高的速度去调用 Windows 资源来刷新显示，导致底层的接收处理速度变慢。

通过勾选与否下图中红色圈标注部分，来观察系统的测试速度，也可以体会接收方的处理性能，对系统的速度的影响，尤其是在 TCP 通信模式下，接收方较差的处理性能，对发送方速度的影响。

7.2.2 网卡数据抓包工具的使用技巧

ALK8266WIFI 模组提供的是 TCP 或 UDP 层的数据通信，有些应用场合需要传输更上层的 HTTP 协议数据或 FTP 协议数据。网上也可以搜索到许多网卡抓包软件，可以直接抓去网卡上所接收到的 UDP 或 TCP 包。因此，可以巧妙使用这个性能，协助单片机模仿填充 UDP 或 TCP 包来支持这些协议。

例如，需要通过单片机通过 HTTP 来访问某网页，发送 POST 或 GET 命令。此时，可以先在电脑上模拟这些操作，并用网卡抓包工具，抓去 TCP 或 UDP 包（这些工具往往还同时对这些包数据走出了格式解释），然后在单片机上模仿填充对应的 TCP 包或 UDP 包，通过 WIFI 模组发出或接收解析。

在单片机网络开发阶段，这种对照技巧也可以用来对单片机端程序问题的定位分析。

7.2.3 使用 XX 卫士等系列测速软件用来简单和准确测速的使用技巧

360 卫士、金山卫士、百度卫士等系列软件，一般都会有一个“流量监控”功能，可以对电脑上的每个软件的接收和发送速度进行监控，如下图所示，显示了每个软件当前的下载速度和上传速度。我们可以利用这个功能来测量模组和单片机集成的实时速度，并指导系统优化。

例如，当单片机通过模组向网络测试工具发送数据时，这里的“流量监控”功能会显示为“下载速度”， 当单片机通过 WIFI 模组从网络测试工具接收数据时，这里的“流量监控”功能会显示为“上传速度”。因此，可以用来观察模组在这种测试方式下的发送和接收实时速度。

此外，这个工具往往还带有一个浮标，显示电脑总的下载和上传速度，如下图所示。

如果此时没有其他的应用在分享网络速度，这个浮标显示的速度大致也可以用来判断当前测试的上传速度和接收速度。

7.3 网络测试所用的实例主机平台

由于大多数应用采用 STM32 主板，并且多以基于 STM32F4 的原子探索者居多，所以这里举例说明和该平台的测试说明。

由于大多数是初期测试时，会采用普通杜邦线将 ALK8266WIFI 模组和单片机的 SPI 接口进行连接，所以这里的测试方式，也采用杜邦线接线方式。

7.4 主机测试程序

在提供的参考项目工程包中，有一个测试程序 test_m8266wifi.c，里面有 3 个测试功能：
(1) 主机通过 ALK8266WIFI® 模组向远端连续发送数据；
(2) 主机通过 ALK8266WIFI® 模组连续接收来自远端发来的数据；
(3) 主机接收来自远端的数据后，立刻发送返还给远端。

7.5 实用性实测结果举例及演示视频

1、测试条件
（1）通过杜邦线飞线连接 ALK8266WIFI® 模组。

（2）调用 M8266WIFI_SPI_Interface_Communication_OK()和 M8266WIFI_SPI_Interface_Communication_Stress_Test()测试发现，可靠通信的时钟频率为 21MHz（主要受限于杜邦线的传输速度）。

2、在此实验条件下：

（1）测试单发送

通过电脑接收查看接收速度，吞吐率可以稳定在 1 兆字节/秒以上，且不丢不多不出错。

（2）测试单接收

通过检测单位时间内主机接收的正确数据量，吐吞率可稳定在 900K-1M 字节/秒以上，且不丢不多不出错。

（3）测试同时收发

主机接收来自远端的数据后，立刻发送返还给远端，并查看远端的速率，系统的双向吞吐率可以稳定在 450K-500K 字节/秒以上，且不丢不多不出错。

3、参考演示视频：

单片机通过 ALK8266WIFI 模组通信，实测有效吞吐速度超 1 兆字节每秒！

https://v.douyin.com/Ftj8vMD/

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。