

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.....	5
1.1 ALK8266WIFI® 高速 WIFI 模组方案介绍.....	5
1.2 术语约定.....	6
1.3 通过书签方便阅读和快速检索定位.....	7
1.4 参考文档.....	7
1.5 联系方式.....	7
2 主机接口模式.....	8
2.1 ALK8266WIFI®和主机接口的交互类型.....	8
2.2 ALK8266WIFI®主机接口模式.....	8
3 与主机的集成--硬件.....	9
3.1 模组的管脚定义和封装说明.....	9
3.1.1 模组的管脚定义.....	9
3.1.2 ALK8266WIFI®模组的封装尺寸和推荐的主板 PCB 封装说明.....	10
3.2 模组和主机的逻辑互连.....	11
3.2.1 主机接口模式 1-仅通过 UART 口对模组进行设置/查询/控制和数据传输.....	11
3.2.2 主机接口模式 2-仅通过 SPI 口对模组进行设置/查询/控制和数据传输.....	11
3.2.3 主机接口模式 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.....	12
3.3 模组的电气特性参数和供电设计.....	13
3.3.1 电气特性参数表.....	13
3.3.2 稳态供电电流与去耦.....	13
3.3.3 主板电路的电源完整性(PI)设计考虑.....	13
3.3.4 nRESET 管脚的去耦.....	14
3.4 模组管脚上的保护和匹配电路.....	15
3.4.1 管脚的接线保护.....	15
3.4.2 SPI 主机板接口布线建议.....	15
3.5 模组参考电路设计.....	16
3.5.1 最小电路设计.....	16
3.6 模组天线的使用及模组在主机板上的位置布局.....	17
3.6.1 板载 PCB 天线和 IPEX 天线接口.....	17
3.6.2 使用板载 PCB 天线的注意事项和模组的放置.....	17
3.7 其他注意事项.....	18
4 模组的一些接口与功能说明.....	19
4.1 模组的 SPI 接口.....	19
4.1.1 模组 SPI 的设置要求和字节读写时序.....	19
4.1.2 SPI 主机接口的固件初始化(跨平台移植时候需注意).....	19
4.2 模组的硬复位.....	20
4.2.1 模组的硬复位时序要求.....	20

4.2.2 模组的硬复位时序的固件实现	20
4.3 热点 AP 模式、工作站 STA 模式, 或 STA+AP 混合模式	21
4.4 自动连网和多种灵活的配网方式	21
4.4.1 自动连网	21
4.4.2 模组的配网	21
4.5 多种链接方式(UDP、TCP 客户端、TCP 服务器)	22
4.6 多链接通信	22
4.7 模组的收发端速度同步机制	22
4.7.1 速度同步机制的重要性	22
4.7.2 发送同步	22
4.7.3 接收同步	23
4.8 UDP 组播	23
4.9 接收中断	24
4.9.1 接收中断的管脚输出和时序图	24
4.9.2 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	24
4.10 低功耗	25
4.10.1 支持根据实际的传输要求, 调试发射功率, 避免功耗浪费	25
4.10.2 支持模组的深度休眠和自动唤醒	25
4.10.3 支持模组的断电使能 EN	25
4.10.4 单片机外部控制模组 VCC 供电断电	25
4.10.5 WIFI 模组的低功耗处理的相关问题	25
4.11 板载 LED 灯及其诊断显示	26
4.11.1 启动时两个 LED 灯先交替闪烁	26
4.11.2 工作时 LED1(绿灯)代表网络数据的接收状态	26
4.11.3 工作时 LED2(黄灯)代表网络接入状态	26
4.11.4 两个 LED 灯同时灭掉表示模组工作异常	26
5 和主机的集成--固件	28
5.1 模组在系统架构中的逻辑位置示意图	28
5.2 层次化结构设计和宏控制	28
5.2.1 层次化结构设计和硬件抽象	28
5.2.2 各层次的核心调用关系示意图	29
5.2.3 宏控制定义文件 brd_cfg.h	30
5.3 和单片机实现快速集成和验证的步骤	30
5.3.1 快速集成的步骤	30
5.3.2 参考例程包的流程示意图	31
5.4 底层的调试技巧—主机接口的硬件接线、初始化和匹配以及验证技巧(选阅)	32
5.4.1 目的	32
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	32
5.4.3 第二步, 验证 nCS 和 nRESET 对应的 GPIO 管脚的连接和初始化	33
5.4.4 验证主机 SPI 接口初始化逻辑是否正确的方法	34

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	34
5.4.5 提供验证过例程的单片机列表	35
6 单片机网络开发和测试策略	36
6.1 网络开发和测试辅助工具的使用技巧	36
6.1.1 网络数据包接收和发送软件工具的使用技巧	36
6.1.2 网卡数据抓包工具的使用技巧	37
6.1.3 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 ..	38
6.2 网络测试所用的实例主机平台	38
6.3 主机测试程序	38
6.4 实用性实测结果举例	38

1 前言

1.1 ALK8266WIFI® 高速 WIFI 模组方案介绍

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。它适用于单片机 WIFI 高速传输音视频、高速采集等应用场合。ALK8266WIFI®, 从模组硬件设计、模组固件设计、以及主机驱动优化算法等全面多角度, 来支持客户实现产品的功能与性能和稳定性的最优。

ALK8266WIFI® 模组包含有 (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。

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和方便的 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) 基于我们提供的基本的 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。

我们提供丰富 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, 确保稳定性和性能的同时最优。

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

包含乐鑫原有的智能配网 (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,极大的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,极大的提高了智能配网的灵活性可靠性以及和可诊断性,提高了用户使用传统的智能配网方式的操作体验。

ALK8266WIFI® 模组提供灵活使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,射频频率出厂时逐一校准并支持自动校准锁频,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, 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,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,对于上述这些系列的单片机型号,我们可以提供经过验证的单片机完整例程包,及测试结果。

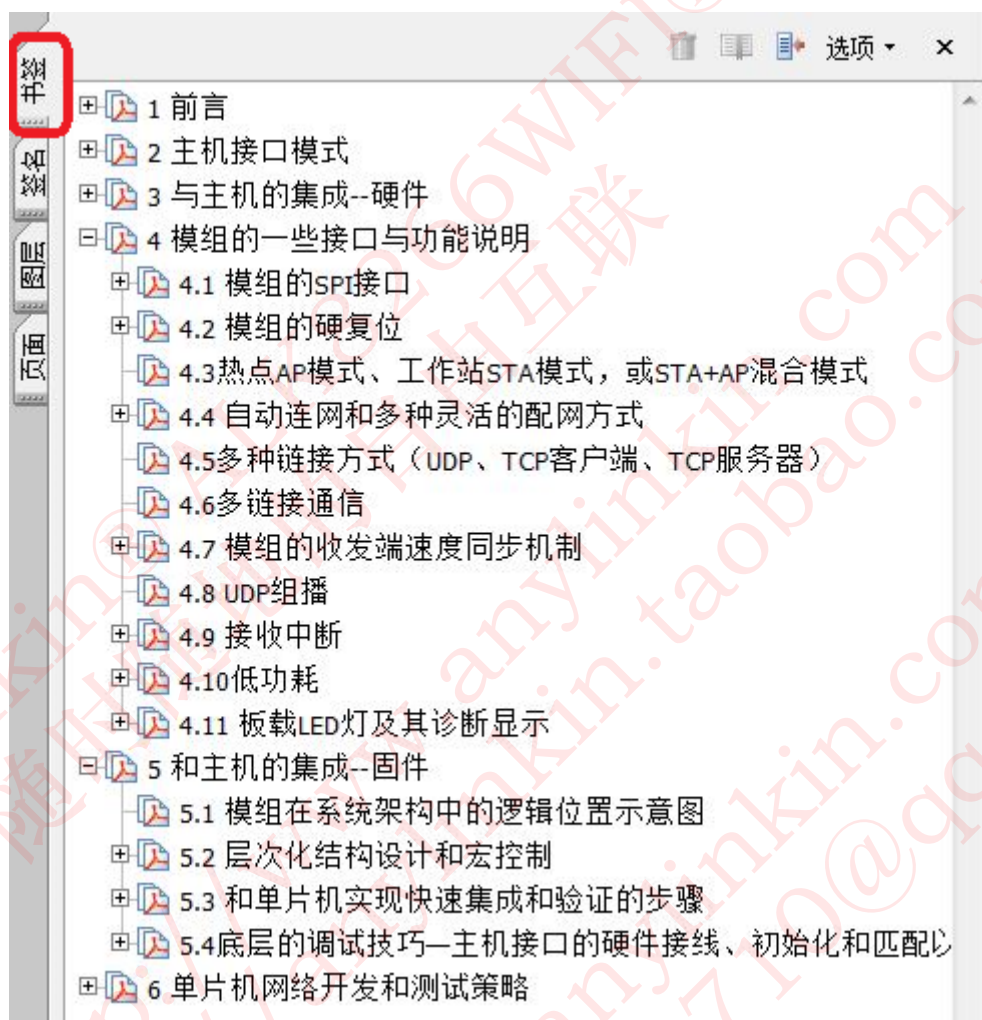
ALK8266WIFI® 模组已经成功地和多款主机处理器实现了集成与高速通信,包括但不限于: STM32 系列单片机 (F0/F1/F2/F3/F4/F7/L1/L4/H7 等系列)、STM8 系列单片机、NXP 系列单片机 (如 LPC1768 系列、K27/K28 系列等)、TI MSP430 和 C2000 系列单片机、Freescale 系列单片机 (如 K60 系列等)、Holtek (台湾合泰) 系列单片机 (如 HT32F1656 等)、Nuvoton (华邦新唐) 系列单片机 (如 NUC123/ NUM451 等)、C51 系列单片机、以及 Sunplus 凌阳等系列单片机,或基于这些单片机内核 IP 的 SOC。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),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 主机接口模式

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®主机接口模式

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模式。我们推荐使用模式 2。

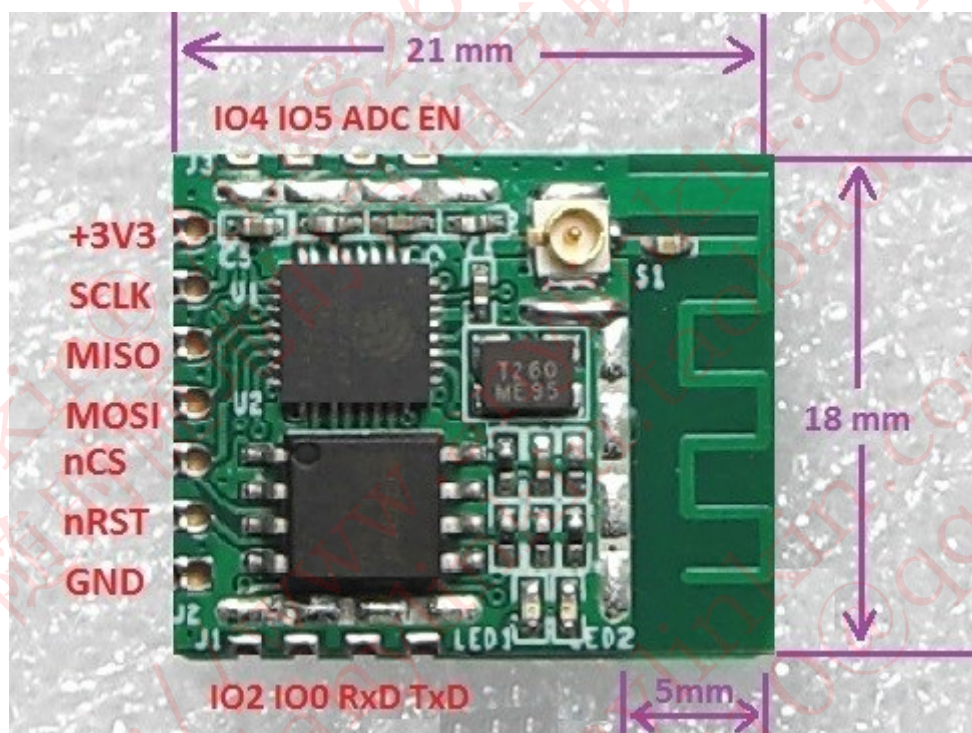
接口模式	主机接口		备注
	查询控制	数据通信	
接口模式 1 (仅通过 UART 串口进行 设置、查询控制和数据通信)	UART 串口	UART 串口	<ol style="list-style-type: none"> 1. 兼容乐鑫 SDK 提供的标准 AT 指令实现对模组的设置、查询和控制, 指令功能丰富; 2. 兼容乐鑫 SDK 提供的标准 AT 指令实现网络通信, 速度较慢, 最大波特率 2Mbps; 3. 占用主机管脚资源较少, 只需要 2 根串口线和一根 GPIO 实现硬复位。
接口模式 2 (仅通过 SPI 口进行 设置、查询控制和数据通信)	SPI 口	SPI 口	<ol style="list-style-type: none"> 1. 使用自定义 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. 使用自定义 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. 我们推荐大家使用这种主机接口模式。
接口模式 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	<ol style="list-style-type: none"> 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. 使用自定义 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模式。

3 与主机的集成--硬件

3.1 模组的管脚定义和封装说明

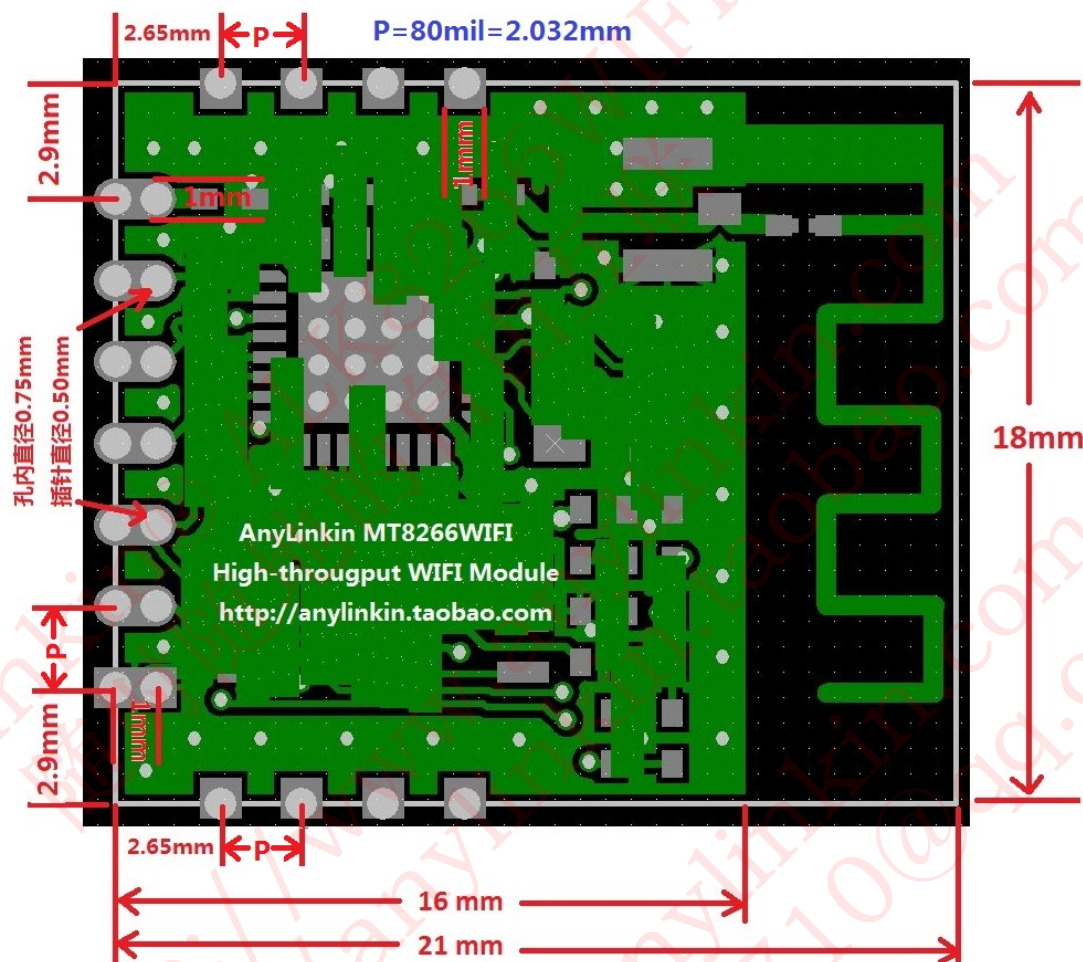
3.1.1 模组的管脚定义

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

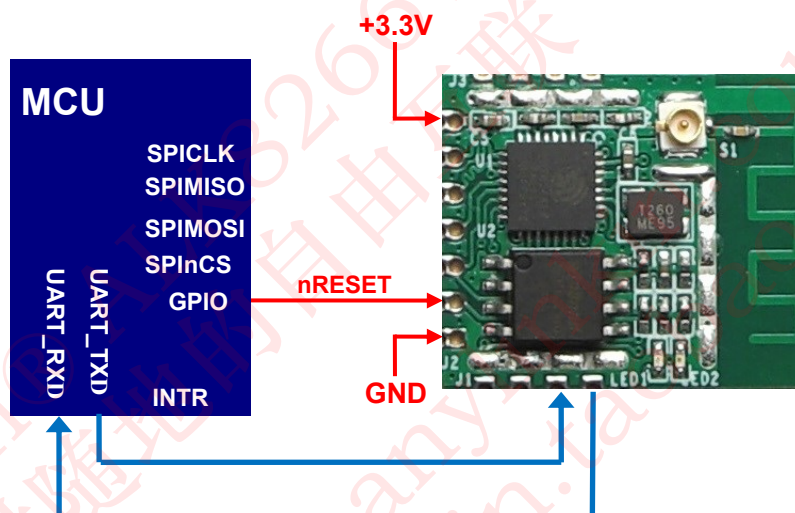
ALK8266WIFI®模组采用标准 2.0mm 的排针插针, 兼容邮票孔的贴片方式, 如下图所示。



3.2 模组和主机的逻辑互连

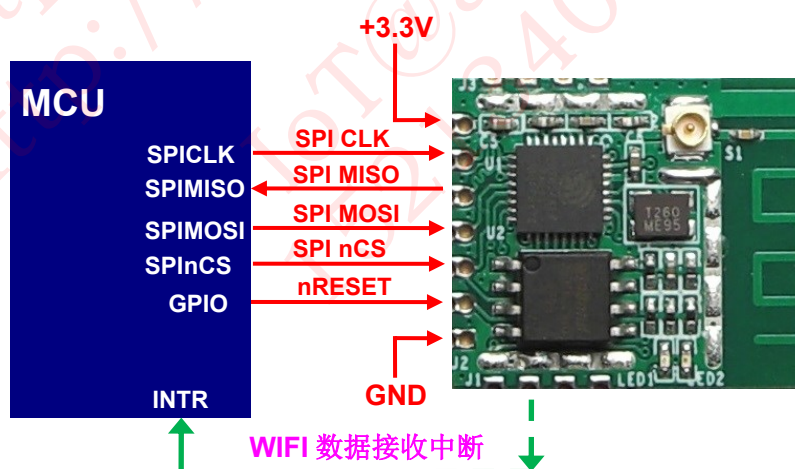
3.2.1 主机接口模式 1-仅通过 UART 口对模组进行设置/查询/控制和数据传输

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 主机接口模式 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。



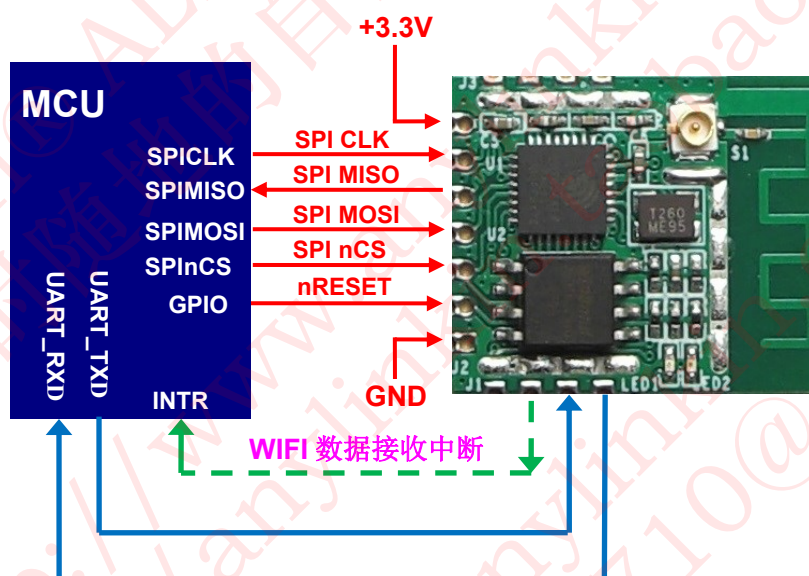
上图中:

- (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 [4.2.1 模组的硬复位时序要求](#)。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 and 单片机主机进行逻辑互联。

3.2.3 主机接口模式 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 [4.2.1 模组的硬复位时序要求](#)。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 and 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		0.25	伏
		VIH, 输入逻辑 1 电平		0.75VCC	
		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。比如 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所, 比如音频放大电路的场所, 这种电流噪声被耦合放大好, 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, 都会注意以下几点:

- (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;
- (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, 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, 在电源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

为了尽可能的提高 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比较远, 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

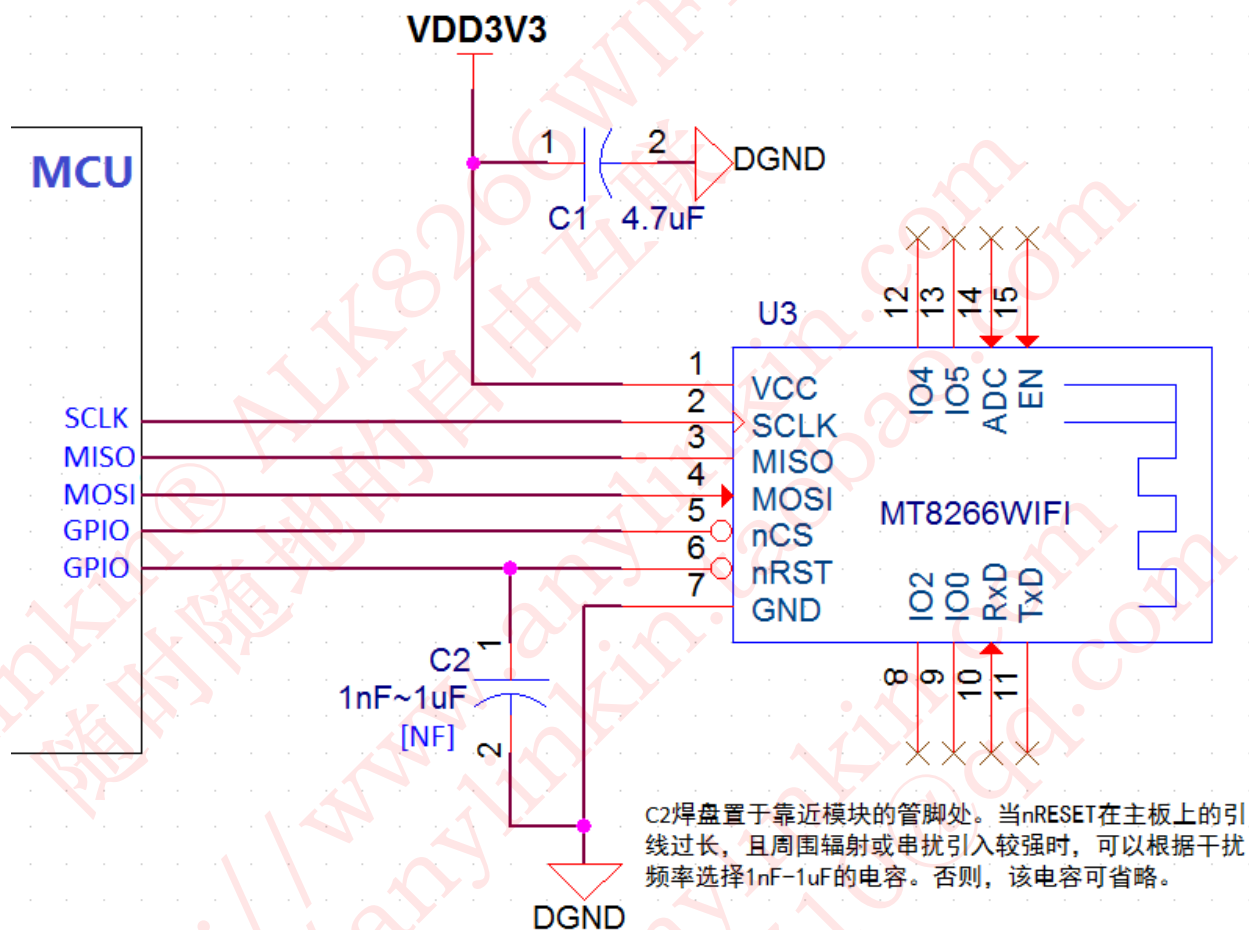
说明:

- (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 (快速或满足) 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, 知道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, 使波形存在一定的超调 (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 模组参考电路设计

3.5.1 最小电路设计

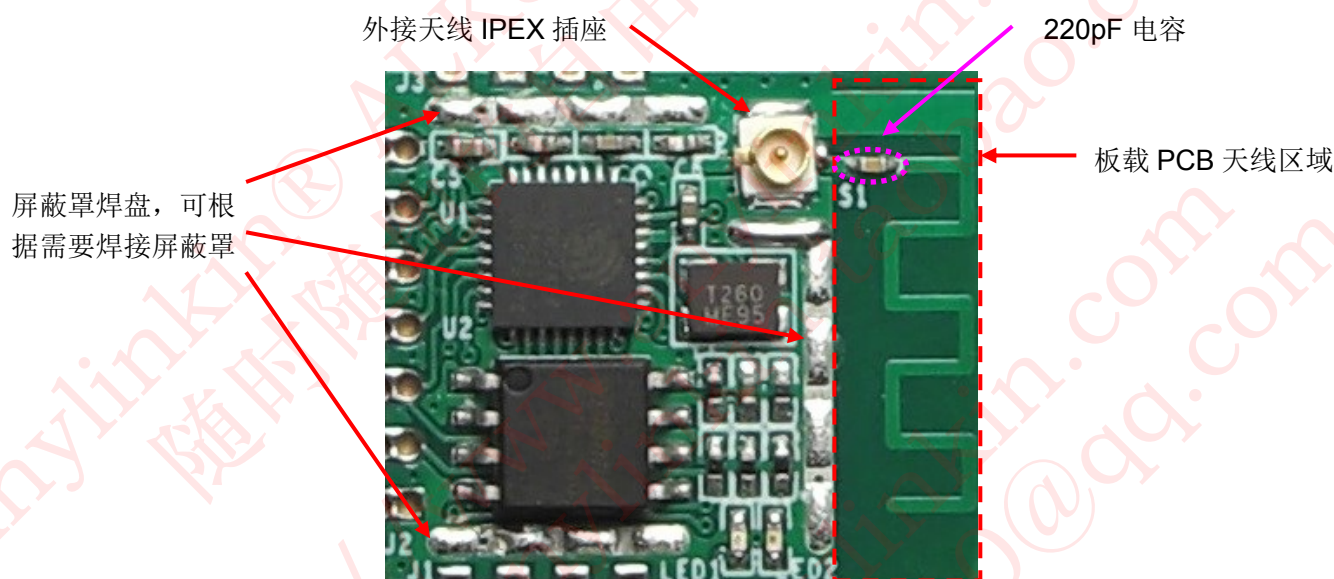


3.6 模组天线的使用及模组在主机板上的位置布局

3.6.1 板载 PCB 天线和 IPEX 天线接口

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部而与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 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, 建议去掉 IPEX 旁边靠近 PCB 天线一侧的 220pF 电容。



3.6.2 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。

3.7 其他注意事项

4 模组的一些接口与功能说明

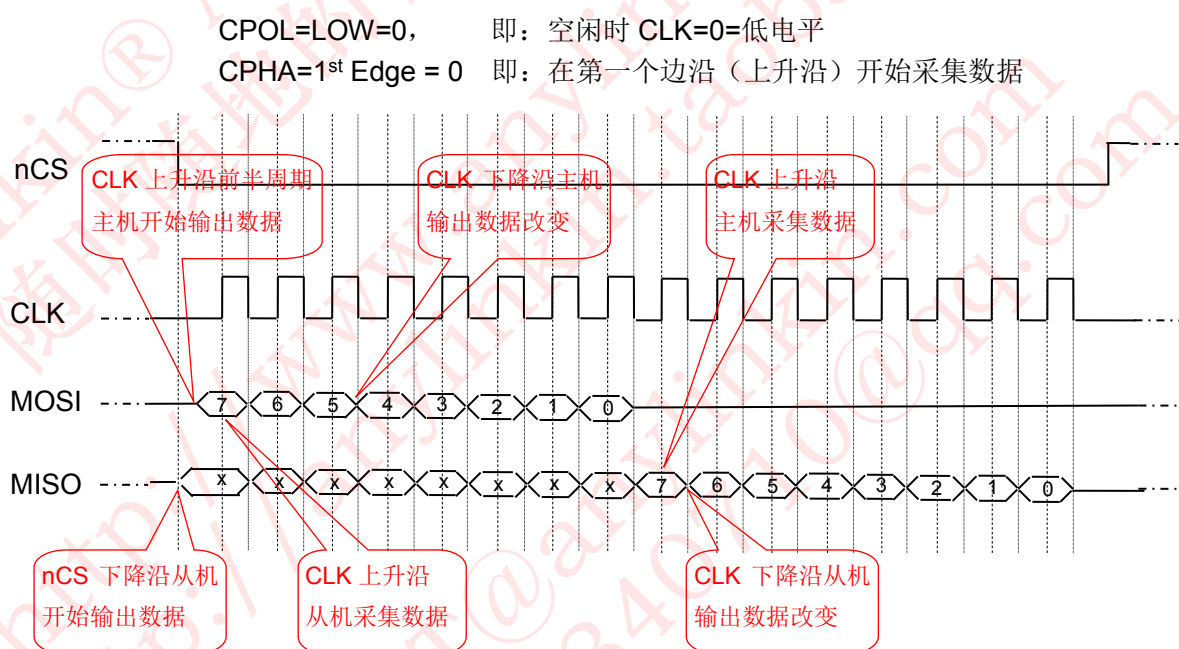
4.1 模组的 SPI 接口

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。

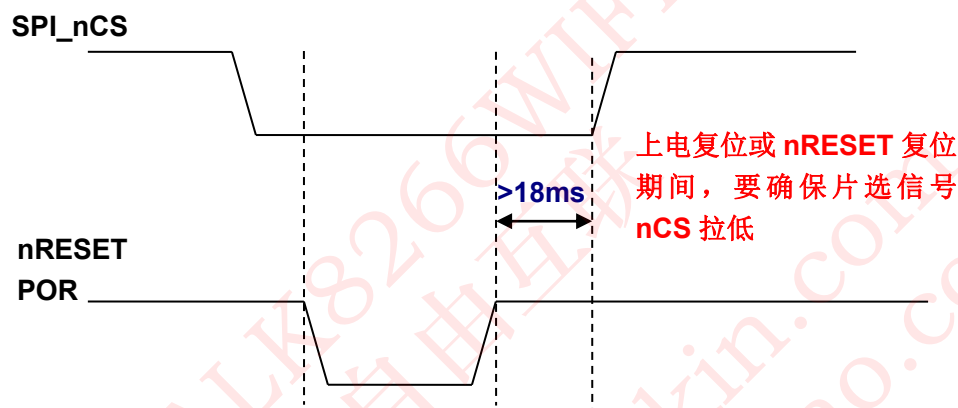


4.1.2 SPI 主机接口的固件初始化 (跨平台移植时候需注意)

- | | |
|-------------------------------------|-------------------------------|
| (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 = 1 st 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, 且保持至少 18-20ms 以上, 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输出模组上的 nRESET 稳定下来需要较长的时间。

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 bootstrap 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. efficiency  
  
    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

ALK8266WIFI® 模组可以作为热点 AP 独立组网, 其他网络节点加入 ALK8266WIFI® 模组模组建立的局域网来实现通信。当模组做 AP 时, 在其所建议的局域网内, 具备确定的静态 IP 地址, 该地址缺省为 192.168.4.1。可以通过 SPI API 接口或 WEB 网页配置该地址。当处于热点 AP 模式下, 在没有其他热点/路由器场合, 也可以实现通信

ALK8266WIFI® 模组可以作为工作站 STA, 接入其他热点或路由器, 实现和其他节点的通信。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。

ALK8266WIFI® 模组可以在 STA、AP、STA+AP 这三种模式之间自由切换。主机可以通过 UART 串口 AT 指令, 或者 SPI API 接口实现在这三种模式之间自由切换。

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 FLASH 里读取保存的 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。另外一种是在掉线后的“自动连网”, 简称“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着热点或路由器。

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 AP 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来提链接。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, 监控其他网络设备作为 TCP 客户端的链接, 简单地实现直连高速传输。

4.6 多链接通信

ALK8266WIFI® 模组支持多链接通信, 即, 在一个 ALK8266WIFI® 模组上, 可以同时建立多个 TCP 或 UDP 链接, 每个链接用唯一的一个链接号进行标识。目前支持同时建立最多 4 个链接, 链接编号为 0~3。如果需要同时建立更多链接, 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。

请参看扩展 AT 指令的说明(可通过串口 AT 指令“AT+SPI=?\r\n”查看), 以及通过 [1.3 参考文档 2](#) 中“4.3.1 M8266WIFI_SPI_Setup_Connection”、“4.4.1 M8266WIFI_SPI_Send_Data”、“4.4.3 M8266WIFI_SPI_RecvData”等章节查看相关 API 函数的介绍说明。

4.7 模组的收发端速度同步机制

4.7.1 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。

4.7.2 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。

4.7.3 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, 模组也会在 TCP 底层自动通知发送方暂停发送, 直到单片机取走数据后, 再通知发送方继续发送, 从而实现了接收处理对发送的速度同步。

接收同步, 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。

4.8 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_Multicast_Group
- 4、建立 UDP 服务, 目标地址指定为组播地址
- 5、开始组播

4.9 接收中断

4.9.1 接收中断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 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:



4.9.2 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 平时会处于常亮状态。当 ALK8266WIFI® 模组接收到网络数据后, LED1 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, 因为接收速度很快, 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 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 管脚输出了低电平); 单步执行完毕 M8266WIFI_SPI_RecvData (.....)后, 可以看到 LED1 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 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 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 低功耗

值得澄清的是,“模组的低功耗”应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。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、尽可能消除扣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。

4.10.1 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。

4.10.2 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。

4.10.3 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。

4.10.4 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,最大化模组的功耗。

4.10.5 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候:

- 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,时间往往较差)。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。

可以参考具体请参考《[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 4.2.4 M8266WIFI_SPI_Config_STA_Static_IP_Addr”部分,了解如何给模块设置静态 IP 地址。

4.11 板载 LED 灯及其诊断显示

4.11.1 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。

4.11.2 工作时 LED1 (绿灯) 代表网络数据的接收状态

LED1 (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 [4.9.1 接收中断的管脚输出和时序图](#)和 [4.9.2 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。

4.11.3 工作时 LED2 (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) 会一直快速闪烁,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,模组一般会去接入该热点或路由器,因为在 [4.11.1 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。因此,在启动后期,在两个 LED 灯交错闪烁 [4.11.1 启动时两个 LED 灯先交替闪烁](#)后,LED2 (黄灯) 会一直闪烁。

4.11.4 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,建议捡块检查模组上的 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

本高速通信模组, 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:



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

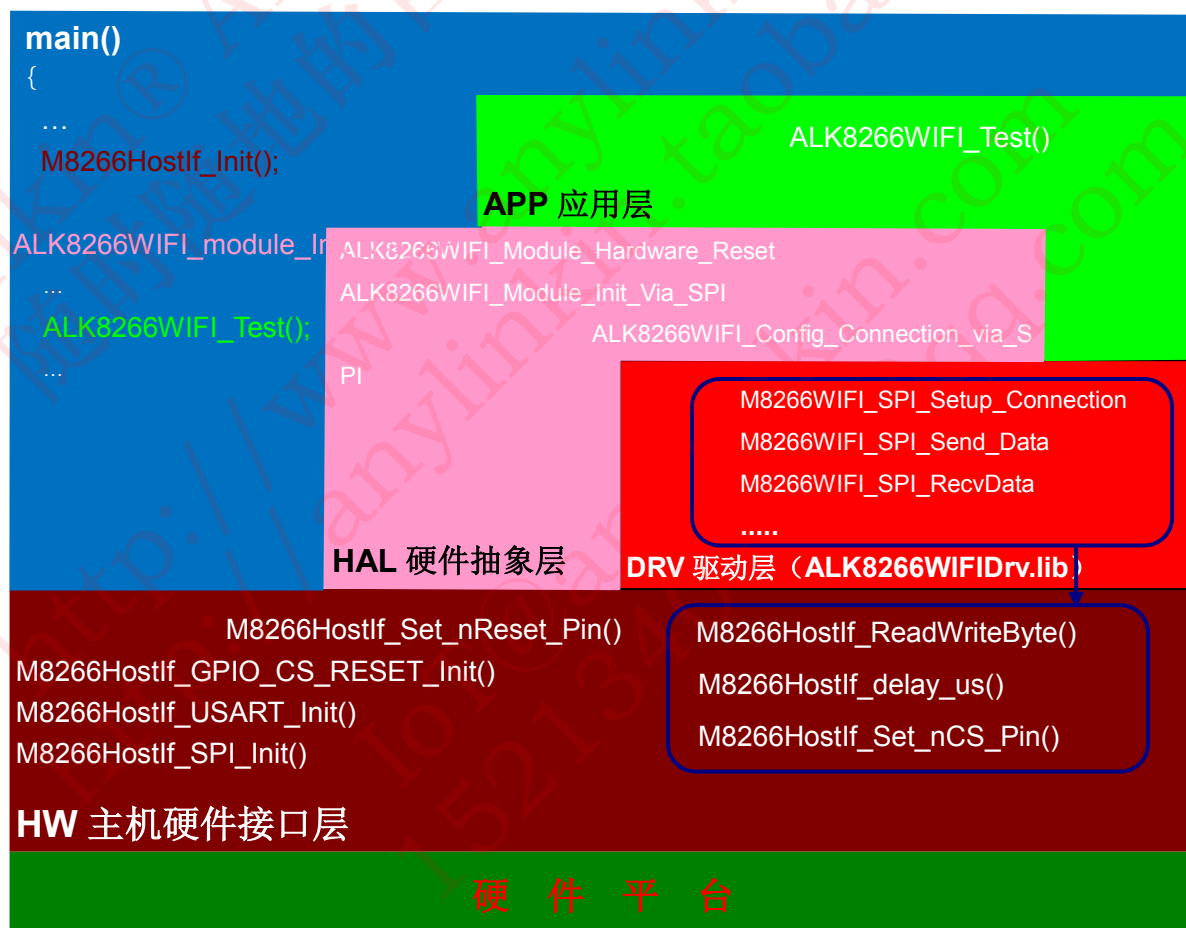
我们提供一套 ALK8266WIFI® 模组和主机集成的参考代码和例程, 整个固件集成系统采用层次化结构设计, 以方便跨平台移植。其示意图如下。

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

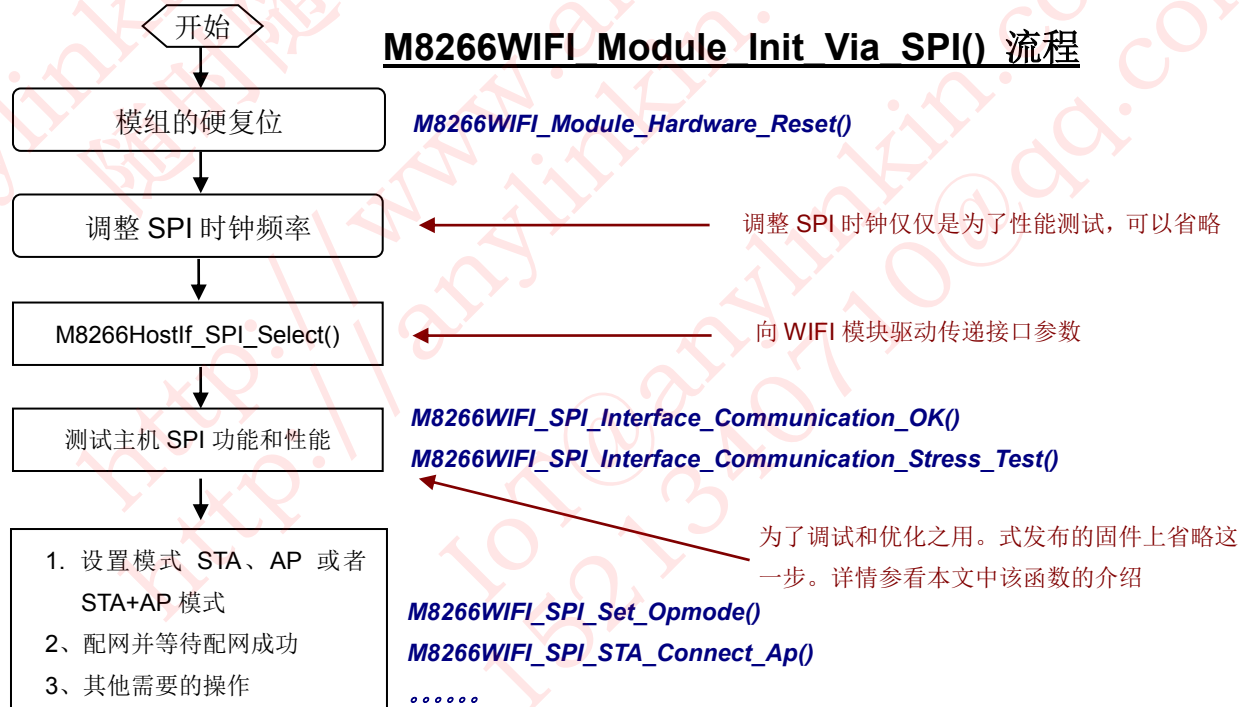
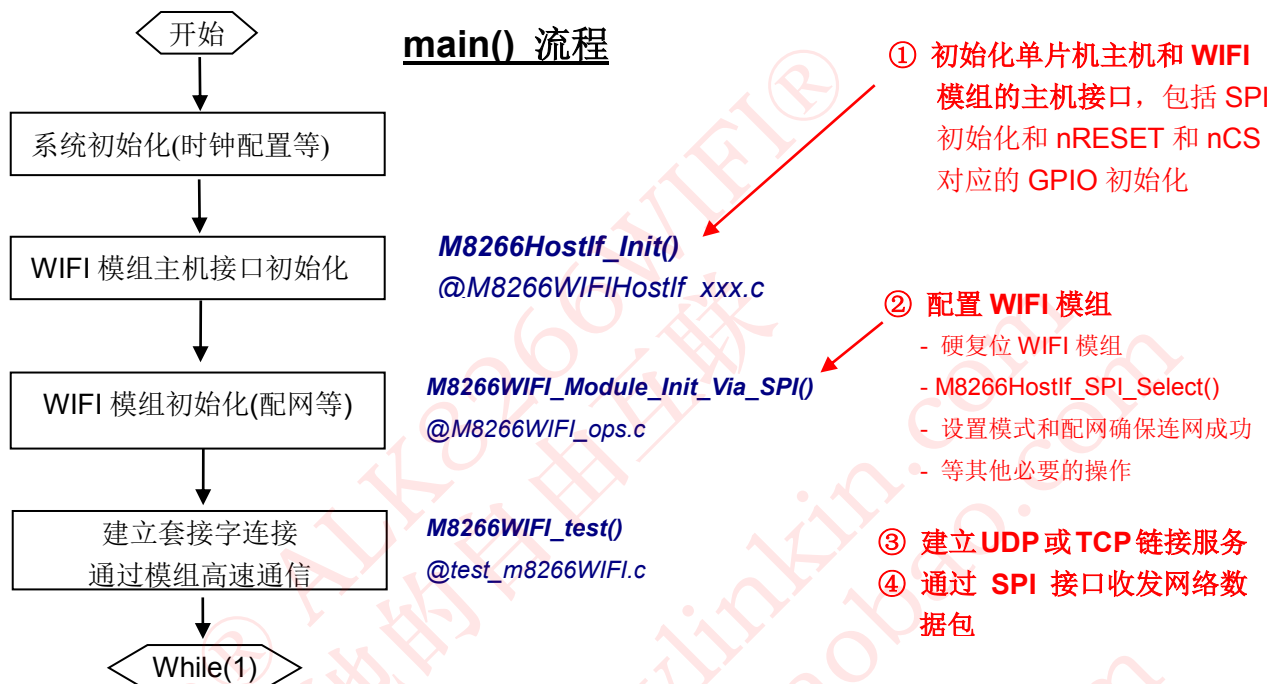
调用《[ALK8266WIFI 模组 SPI 接口高速通信使用与集成—主机驱动 API 函数](#)》章节“4.1.3 `M8266WIFI_SPI_Interface_Communication_Stress_Test`”函数,获取并验证主机板 SPI 接口可以稳定可靠通信的最大时钟频率。详情参看 [5.4.3 验证/获取主机板 SPI 接口稳定可靠通信的最大时钟频率](#)。使用可靠通信的最大 SPI 时钟频率,有利于在现有系统上最大化通信的吞吐率。

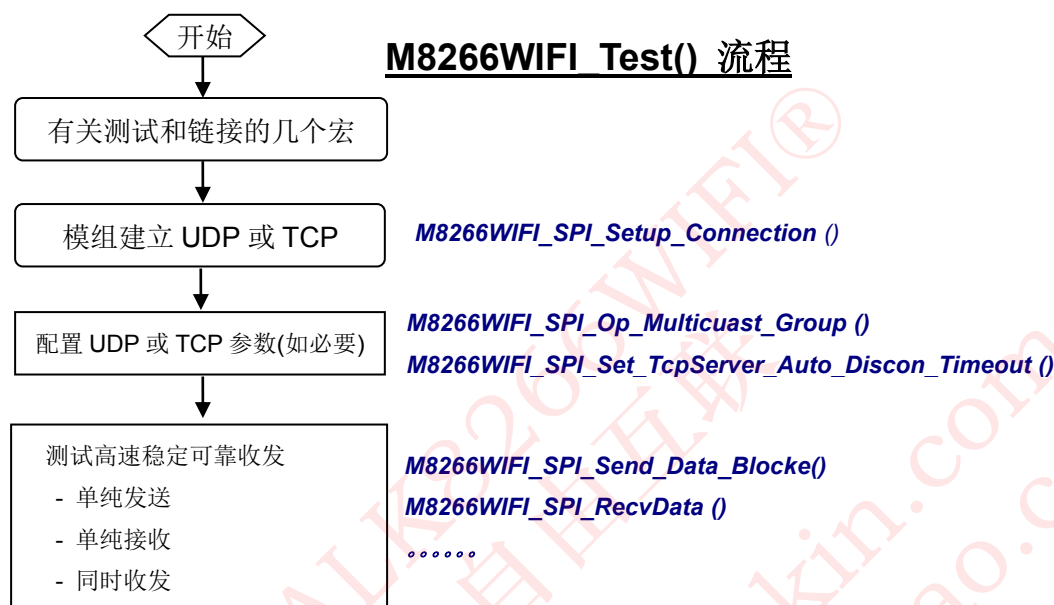
5、加入 M8266WIFI_ops.c 文件(这个文件里的函数代码做到了与平台无关),直接使用其中的操作函数,或者,你可以仿照其中的函数,扩展实现你所需要的功能。

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,提供了三种测试机制:(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,来测试接口接收的速率;(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。

5.3.2 参考例程包的流程示意图





5.4 底层的调试技巧—主机接口的硬件接线、初始化和匹配以及验证技巧 (选阅)

5.4.1 目的

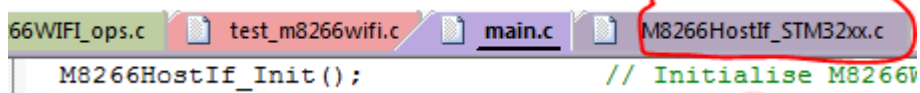
在集成的初期, 需要根据硬件互联方式、使用模式、及主机芯片, 调整 M8266HostIf.c 中相应的底层函数实现和初始化, 包括 SPI 接口的初始化。确保主机 SPI 接口的初始化正确, 且和 ALK8266WIFI 的 SPI 接口配置相匹配, 这一点对于上层固件的通信非常重要。尤其是对于某些跨平台移植 (我们没有提供相关的参考包), 首先要确保底层主机接口和 SPI 字节基本读写通信是正确的, 且可靠的。一旦与底层硬件有关的 SPI 通信, 在功能和性能上通信调试验证可靠了, 上层与硬件无关的应用实现就简单了。

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 nCS 和 nRESET 对应的 GPIO 管脚的连接和初始化

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
34  /*
35  * M8266WIFI_Module_Hardware_Reset
36  * Description
37  * 1. To perform a hardware reset to M8266WIFI module via the nReset Pin
38  * and bring M8266WIFI module to boot up from external SPI flash
39  * 2. In order to make sure the M8266WIFI module bootup from external
40  * SPI flash, nCS should be low during Reset out via nRESET pin
41  * Parameter(s):
42  * none
43  * Return:
44  * none
45  */
46 void M8266WIFI_Module_Hardware_Reset(void) // total 800ms
47 {
48     M8266HostIf_Set_SPI_nCS_Pin(0); // Module nCS==ESP8266 GPIO15 as well, Low during reset in order for a normal reset
49     M8266WIFI_Module_delay_ms(1); // delay 1ms, adequate for nCS stable
50
51     M8266HostIf_Set_nRESET_Pin(0); // Pull low the nReset Pin to bring the module into reset state
52     M8266WIFI_Module_delay_ms(5); // delay 1ms, adequate for nRESET stable.
53     // give more time especially for some board not good enough
54
55     M8266HostIf_Set_nRESET_Pin(1); // Pull high again the nReset Pin to bring the module exiting reset state
56     M8266WIFI_Module_delay_ms(300); // at least 18ms required for reset-out-boot sampling bootstrap pin
57     // Here, we use 300ms for adequate abundance, since some board GPIO,
58     // needs more time for stable(especially for nRESET)
59     // You may shorten the time or give more time here according your board v.s. efficiency
60
61     M8266HostIf_Set_SPI_nCS_Pin(1);
62     //M8266WIFI_Module_delay_ms(1); // delay 1ms, adequate for nCS stable
63
64     M8266WIFI_Module_delay_ms(800-300-5-2); // Delay more than around 500ms for M8266WIFI module bootup and initialization
65     // including bootup information print.
66 }

```

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 [4.11.1 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
34
35 * M8266WIFI_Module_Hardware_Reset
36 * Description
37 * 1. To perform a hardware reset to M8266WIFI module via the nReset Pin
38 * and bring M8266WIFI module to boot up from external SPI flash
39 * 2. In order to make sure the M8266WIFI module bootup from external
40 * SPI flash, nCS should be low during Reset out via nRESET pin
41 * Parameter(s):
42 * none
43 * Return:
44 * none
45
46 void M8266WIFI_Module_Hardware_Reset(void) // total 800ms
47 {
48     M8266HostIf_Set_SPI_nCS_Pin(0); // Module nCS==ESP8266 GPIO15 as well, Low during reset in order for a normal reset
49     M8266WIFI_Module_delay_ms(1); // delay 1ms, adequate for nCS stable
50
51     M8266HostIf_Set_nRESET_Pin(0); // Pull low the nReset Pin to bring the module into reset state
52     M8266WIFI_Module_delay_ms(5); // delay 1ms, adequate for nRESET stable.
53     // give more time especially for some board not good enough
54
55     M8266HostIf_Set_nRESET_Pin(1); // Pull high again the nReset Pin to bring the module exiting reset state
56     M8266WIFI_Module_delay_ms(300); // at least 18ms required for reset-out-boot sampling bootstrap pin
57     // Here, we use 300ms for adequate abundance, since some board GPIO,
58     // needs more time for stable(especially for nRESET)
59     // You may shorten the time or give more time here according your board v.s. efficiency
60
61     M8266HostIf_Set_SPI_nCS_Pin(1); // delay 1ms, adequate for nCS stable
62     //M8266WIFI_Module_delay_ms(1);
63
64     M8266WIFI_Module_delay_ms(800-300-5-2); // Delay more than around 500ms for M8266WIFI module bootup and initialization
65     // including bootup information print.
66 }
```

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 接口模式匹配,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)	1、LPC17xx 系列 2、K27/K28 系列
飞思卡尔(Freescale)	1、K60 系列
德州仪器(TI)	1、MSP430 系列 2、C2000 系列, 例如 TMS320F28335
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
C8051 系列	1、C8051F120 系列 2、C8051F340 系列
更多值得期待.....	

6 单片机网络开发和测试策略

6.1 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。

6.1.1 网络数据包接收和发送软件工具的使用技巧

技巧 1、可以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, 通过编辑框接收和发送, 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。

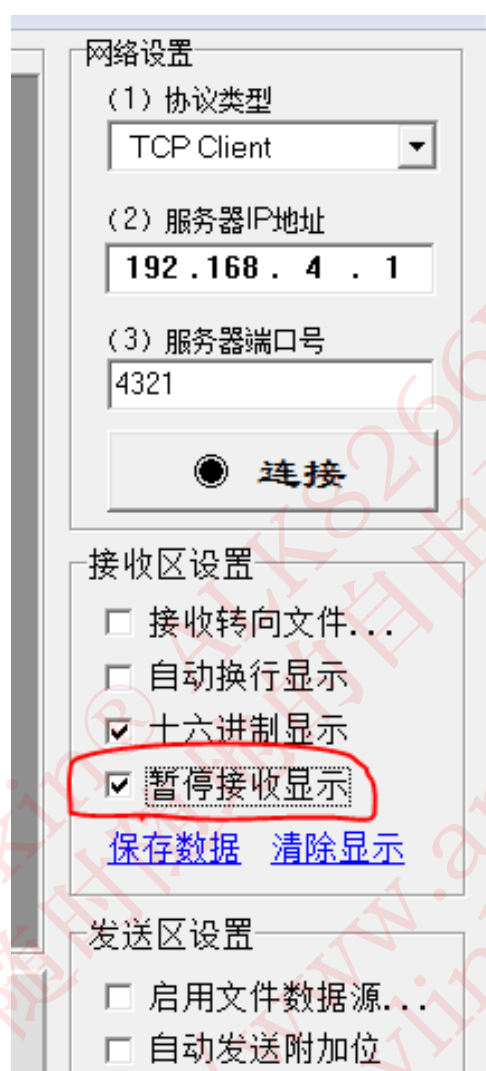
例如, 现在有单片机系统通过 ALK8266WIFI 模组向某手机 APP 发送一个 UDP 数据包, 数据包里的数据是自定义的数据协议。但是单片机发出后, 手机 APP 却没有反应。原因可能是: (1) 单片机没有发出 UDP 包; (2) 单片机发出的数据包内的数据协议不正确被手机 APP 给丢弃了; (3) 手机没有接收到数据包, 或者手机处理出错, 等原因。

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, 来判断或判断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。



6.1.2 网卡数据抓包工具的使用技巧

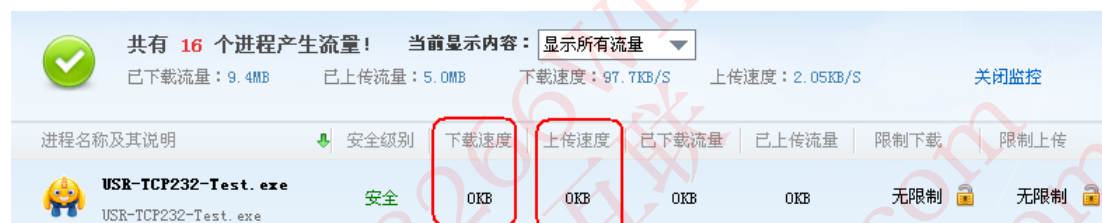
ALK8266WIFI 模组提供的是 TCP 或 UDP 层的数据通信,有些应用场合需要传输更上层的 HTTP 协议数据或 FTP 协议数据。网上也可以搜索到许多网卡抓包软件,可以直接抓去网卡上所接收到的 UDP 或 TCP 包。因此,可以巧妙使用这个性能,协助单片机模仿填充 UDP 或 TCP 包来支持这些协议。

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。

6.1.3 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。

6.2 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。

6.3 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_ALK8266WIFI.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。

6.4 实用性实测结果举例

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 900K-1M 字节/秒以上

(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,吐吞率可以稳定在 900K-1M 字节/秒以上

(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。

3、瓶颈分析

这里的有效吞吐率并不高,主要的瓶颈在于 STM32F4 单片机系统(主要受限于普通杜邦线的连接)的时钟频率只能可靠的工作在 21MHz 左右,所以 SPI 写的有效吞吐率大约只有 13-14Mbps 左右。

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。