ESP8266说明文档

- 1.目的
  - 本例程包的目的是演示单片机主机通过 `SPI` 接口对 `ALK8266WIFI` 高速模组进行查询、配置、以及通过该模组进行高速通信。
  - 写得比较详细，主要是有两点原因： （1）网络通信系统天然具备复杂性 （2）本例程面向的用户的技术背景各有不同。
  - 所以，阅读时，可根据需要酌情选阅，每段前都有大标题。
- 2. 只需要 `SPI` 口
  - 使用本高速WIFI模组，所有需要的操作，主机都可以通过 `SPI` 接口来完成，而不需要 `UART` 串口。
  - 这些操作包含设置模块的工作模式为STA或AP、对模块配网去链接热点或路由器、建立套接字链接、以及进行数据的高速收发， 以及更多的查询设置，包括IP地址的查询和设置、STA或AP参数的查询和设置、TCP套接字的查询和设置，等等，都可以通过 **SPI接口API函数**完成。
  - 完整的 `SPI` 接口 `API` 函数说明，可以查阅 ***《ALK8266WIFI模组SPI接口高速通信使用与集成_主机驱动API函数说明》***，可以通过 文档左侧的书签快速定位相关章节或API函数。
- 3. 如何接线
  - `ALK8266WIFI` 高速模组和单片机的连接非常简单，主要是 `SPI` 的3根线，以及2个 `GPIO` 分别做 `nCS` 和 `nRESET` 。
  - 3.1 `SPI` 接线示意图
    - 可以参看本例程包的M8266WIFI目录下的“**M8266WIFI模组主机接线示意图.JPG**”，按照模式2进行简单接线。
    - 这个接线图示，在文档 ***《ALK8266WIFI模组SPI接口高速通信使用与集成_主机集成说明》*** 中“硬件集成”章节有详细的说明解释，可以通过文档左侧的书签快速定位相关章节。
  - 3.2 具体的主机 `SPI#` 和 `IO管脚` 的用法 `brd_cfg.h` 文件
    - 具体单片机主机使用哪个 `SPI#` 来连接模组，以及哪2个 `GPIO` 做片选，可以在例程包中的 `brd_cfg.h` 文件找到宏定义
      - （1）打开 `brd_cfg.h` 文件，在里面搜索具体的单片机型号，比如，如果主机是 `STM32F1` ，就搜索“ `STM32F1` ”，可以看到 `nCS` 和 `nRST` 的宏定义，揭示了例程里缺省使用的是哪两个 `GPIO` 。
      - （2）打开 `brd_cfg.h` 文件，搜索关键字“ `M8266WIFI_SPI_INTERFACE_NO` ”，可以找到例程包里缺省使用的 `SPI#`
      - （3）需要保持这些宏定义和实际主板的接线一致，可以简单地按照这个头文件里的定义保持缺省不变，和主机板进行接线；也可以直接将头文件里的宏定义改成你所希

望的主板上的 `IO` 使用情形。总之，保证代码和实际的接线一致就成。

- 3.3 `SPI` 具体使用哪个 `GPIO` 来服用为 `SPI` 的信号线， `M8266HostIf_[主机型号].c` 文件（例如 `M8266HostIf_STM32xx.c` ）
  - 许多单片机对于一个 `SPI` 信号，会有很多种IO管脚来复用，需要对选定的IO管脚做 `GPIO初始化` 。
  - 具体使用哪个管脚，在 `M8266HostIf_[主机型号].c` 文件中，搜索对应的单片机类型，例如 `STM32F1` ，找到对应的“ `M8266WIFI_SPI_INTERFACE_NO` ”下的IO管脚初始化，通过 `#if1` 选择你所希望的IO管脚，或者 `#if 0` 注销你不希望使用的IO管脚
  - 当然，你也可以修改成你所希望的管脚组合。

- 4. 初步快速熟悉例程包 -- 单步执行
  - 按照3中接好线后，首先可以简单的对例程进行编译，然后下载到单片机里，开始单步执行，以便尽快熟悉。
  - 单步执行的好处是：
    - （1）有助于快速理解例程流程和功能
      - 通过单步执行，有助于快速理解代码（单片机带着我们来熟悉代码，而不需要自己看代码琢磨）
      - 比如，单步执行到某个地方，发现将 `WIFI模块的模式` 设置为了 `2` （AP Only模式），但是自己希望是将 `WIFI模块` 设置为 `STA` 模式去链接第三方路由器，而旁边的注释写着 `2` 是 `AP-Only` ， `1` 是 `STA Only` ，于是就可以将2改成1很快完成了自己的需求。
    - （2）有助于厘清具体的**IO管脚使用的宏定义**
      - 单步执行的过程中，可以看到代码中的各个宏定义或宏开关的具体情形，然后根据自己的需要进行该调整。
      - 比如，单步执行到 `nCS` 和 `nRST` 的 `GPIO初始化` 的地方时，看到对应的宏，右键单击该宏跳到其定义核对和实际使用的是否一致
    - （3）有助于对故障定位和分析优化
      - 我们尊从可测试性和可维护性的设计理念，目标是产品化最优。所以，在例程里，存在大量的测试性和诊断提示的状态码。
      - 单步执行的时候，可以看到哪一步出错，以及其状态提示，有助于快速定位问题或优化设计。

- 5、例程文件包的组成说明
  - 5.1 例程是一个完整的单片机平台工程
    - 其中包括有单片机对应的底层文件（如启动、时钟等等），这些和普通的单片机平台没有两样。
    - 你可以直接使用例程包的，也可以替换成你自己从其他地方验证过的。
  - 5.2 核心包 `M8266WIFI`

- （1）单片机主机接口文件 `M8266HostIf_[主机型号].c` 文件
  - 例如 `M8266HostIf_STM32xx.c`，核心函数 `M8266HostIf_Init()` 主要是单片机和WIFI模组的主机接口的初始化，是主机上的 `SPI` 、 `GPIO(nCS/nRST)` 的初始化和延迟函数等等的实现，这个文件里的一些函数，会被 `main()` 初始化阶段的 `M8266HostIf_Init()` 所调用，所以主要是服务于主机端接口的初始化
- （2）主机驱动库文件 `M8266WIFIDrv_[主机小型号].lib` （或.a文件）
  - 例如 `M8266WIFIDrv_STM32F1xx.a` ， 这个驱动库基于 `SPI底层` 的读写，实现了和模块的基础通信API函数，包括了上述2中提到的SPI操作的全部原子功能，例如配网、建立套接字等等。这些操作，都对应着一个个的API函数，保存在驱动库里。驱动库的具体实现代码不可见 :-)，但是每个API函数的功能和使用说明，都在**《ALK8266WIFI模组SPI接口高速通信，使用与集成_主机驱动API函数说明》**文档里有详细说明，对应的头文件 `M8266WIFIDrv.h` 里也有详细注释。
- （3）操作WIFI模块--配置/操作ALK8266WIFI模块 `M8266wifi_ops.c` 文件
  - 核心函数 `M8266WIFI_Module_Init_Via_SPI()`
  - 这个文件，可以理解成是使用（2）主机驱动API函数的程序部分样例，里面有一个核心函数 `M8266WIFI_Module_Init_Via_SPI()` 会被 `main()` 初始化阶段所调用，从名字上看也是对WIFI模块的初始化操作了，包括：
    - 1）模块的硬复位
    - 2）**主机 `SPI频率` 的设置、 `SPI号` 和频率传递给模组 `M8266HostIf_SPI_Select` 函数、以及 `SPI` 性能压力测试**
      - 这个主要是辅助评估接线的正确和优劣。正式产品中，除了 `M8266HostIf_SPI_Select()` 不可省略外，其他的都可以根据具体应用取舍。
    - 3) 工作模式（ `STA` / `AP` / `STA+AP`)设置，以及配网等等
    - 4）其他一些可有可无的操作，纯粹是为了演示一些API函数的使用，这些API函数，都用`#if 0`或`#if 1`包括，旁边都有注释，可以随意调整是否需要。
  - 所以，所谓的 `M8266WIFI_Module_Init_Via_SPI()` 对模块初始化的功能，实际是
    - 1）一个必须的模块的硬复位 和 `M8266HostIf_SPI_Select()` ，以及
    - 2）必要时对模组的工作模式的设置和配网-- 模组上之前保存的配置和需要一致，这一部分模组可以上电时完成，甚至也可以不要
    - 3）其他的都是功能扩展，根据实际情形取舍或增减
- （4）建立套接字，开始高速收发通信 `test_m8266wifi.c` 文件，
  - 核心函数 `M8266WIFI_Test()`
  - 在初始化完毕后， `main()` 会调用 `M8266WIFI_Test()` 对模组进行测试。
  - `M8266WIFI_Test()` 主要有2个功能： **（1）建立套接字链接 （2）开始数据的高速收发测试**
  - 在这个文件的开始，有一大段**宏定义**，控制测试的模式和方式

        - （1）我们的测试方式，包括模组单纯高速发送、模组单纯高速接收、模组一边接收一边发送、以及多客户端通信演示，4个测试，所以，在开始有个宏 `TEST_M8266WIFI_TYPE` ，1表示只发，2表示只收，3表示一边发一边收，4表示多客户端演示。
        - （2）我们测试模式，包括 `UDP` 、 `TCP客户端` 、 `TCP服务器` 。所以这里有个宏 `TEST_CONNECTION_TYPE` ，0表示UDP，1表示模组做TCP客户端，2表示模组做TCP服务器。
    - 用户可随意控制调节这里的宏，完成不同的测试组合。
- 5.3 例程缺省配置下的测试类型
    - （1）在 `M8266wifi_ops.c` 文件的核心函数 `M8266WIFI_Module_Init_Via_SPI()` 里，模组缺省设置为AP模式
    - （2）在 `test_m8266wifi.c` 文件的核心函数 `M8266WIFI_Test()` 里，模组缺省设置为TCP服务器超时时间为120秒，单纯发
    - 所以，如果拿到例程后，调整好了步骤“3.接线”后，没有修改这里的配置的话，那么模组就建立起了一个AP和TCP服务器，等着别人来连接入网和链接TCP服务器。一旦被连接和链接上了，模组就会开始高速给这个接入的STA和TCP客户端高速发数据了。
    - 例如，直接找一套电脑，扫描模组建立的 `热点ALK_xxx` 这样的，首先连上这个热点，获取到IP地址。然后打开TCP测试工具，去连接一个 地址为192.168.4.1端口为4321的TCP服务器，一旦连上，就会看到被高速发送数据了。
    - **缺省测试**的好处是，不需要配网而依赖路由器、不需要去指定TCP服务器而依赖服务器等等，所以适合第一步快速简化测试。

6、性能测试

- 性能测试包括速度测试 和丢包测试。使用我们的例程，辅助其他**TCP/UDP测试工具**，或者**两个模组之间对传**，来测试模组的通信性能。
- 我们的测试结果是：
    - 1、30天持续运行不掉线不卡死持续正常运行
    - 2、TCP高速通信速度维持在兆字节（取决于单片机）每秒左右，持续运行5小时，不丢包不多包，不丢字节不多字节
- 直接使用我们的例程就可以测试得到这个结果。如果你的测试结果没有达到这个效果，且你需要这个效果，可以联系我们，一起来梳理通信系统（ALK8266WIFI模组只是通信系统中的一个环节）中的短板。

7、测试时的常见问题

- （1）全速跑缺省配置的例程，然后用互联网共享免费软件 `USR-TCP-232` ，却发现连不上，如何诊断?
    - 1）单步执行例程，查看程序能否成功地执行到了 `M8266WIFI_Test()` 函数里的 `M8266WIFI_SPI_Setup_Connection()` 代码处，且这个函数执行成功。这个检查，是为了确保模组必要的初始化和套接字的建立都是成功的。只有成功了，才会创建起TCP服务器供其他客户端来连接。

        - 如果卡在了之前的代码里，请检查接线或单片机硬件系统的稳定性，可根据所出错的函数里返回的status进行诊断。也可以将出错时所在的API函数截图和status数值发给我们协助分析。
        - 判断“模组必要的初始化和套接字的建立”是否成功，也可以打开访问模组上的WEB服务器，点击TCPUDP标签，查看是否存在一个TCP服务器。如果不存在，则表明并未成功建立，需要按照上面的单步调试方法，排查前面代码关联的（接线等）问题。
    - 2）如果在(a)已经确证TCP服务器已经成功建立起来了，接下来进一步检查：
        - (i) 确保测试用的电脑和WIFI模组在网络上是联通的。
            - 简单的方法是，查看电脑当前的IP地址是否为192.168.4.* (这是缺省的网段），并在电脑控制台cmd里ping一下模组AP缺省所对用的IP地址(ping 192.168.4.1)
        - (ii) 确保测试用的电脑上的防火墙没有限制测试用的缺省端口4321
            - 有些时候，也可以临时再找一台电脑或手机对照测试。
        - (iii) 对互联网免费测试软件**USR-TCP-232**的特别处理
            - 互联网共享免费软件**USR-TCP-232**可能存在一个BUG，就是当电脑上存在多网卡（含虚拟机网卡）时，这个软件可能会不知改选择哪个网卡去连出的意外。此时，如果临时停止其他网卡只留下当前去连接WIFI模组的那个网卡，就不会有问题了。
            - 互联网共享免费软件**周立功的UDP&TCPDebug**不存在这个多网卡的问题。
- （2）跑缺省配置的例程，开始为何无法达到各个平台的极限速度
    - 通信系统的速度，由通信系统中的短板所决定，模组只是其中一个环节。尽管缺省的例程配置，采用了模组做AP消除了路由器等潜在影响，但是测试软件的如下配置，可能会造成对系统性能的短板：
        - 1） 互联网共享免费软件**USR-TCP-232**和**周立功UDP&TCPDebug**都有一个“**暂停显示**"/"**暂停接收显示**"/"**Receive Pause**"的选项，可以按下该按钮或选中该标签，对比查看对速度的影响，体会”接收方的处理“对通信系统整体性能的影响这一个特征。
        - 2）对照测试发现，使用**UDP&TCPDebug**测试的速度很难超过500K字节每秒，但是使用**USR-TCP-232**到1兆字节每秒，体会接收软件处理的好坏，对通信系统整体性能的影响这一个特征。
        - 3）其他原因，比如接线不可靠、供电不可靠等等。但是一般只要(a)和(b)厘清了，速度很容易就上去了。
        - 要求使用例程缺省配置测试的速度，达到该平台下极限速度的至少80%以上，否则，请略花时间仔细梳理，熟悉通信系统的规则和经验，为后期做产品打好基础。

8、移植到自己的工程包中

- 1、将驱动库 `M8266WIFIDrv_[主机小型号].lib` （或.a文件） 加入到你的工程包
- 2、参考 `M8266HostIf_[主机型号].c` 文件写 `M8266HostIf_Init()` ，包括主机端的SPI初始化、 `nCS/nRESET` 对应 `GPIO` 的初始化，以及us延迟函数的实现
- 3、参考 `M8266wifi_ops.c` 文件的核心函数 `M8266WIFI_Module_Init_Via_SPI()` ，实现对模组的初始化和配置

    - 主要是硬件复位、调用 `M8266HostIf_SPI_Select()` 向模组传递SPI号和频率参数、必要时的模组工作模式设置和配网操作，以及其他一些具体应用所需要的配置查询操作等等
- 4、参考 `test_m8266wifi.c` 文件的核心函数 `M8266WIFI_Test()` 建立套接字服务和数据的高速收发。
- **【小提示】**一些客户在移植后，会遇到程序跑飞出现 ***hardware fault***的单片机异常，这个错误往往是因为系统堆栈设置小了所致。通常，变量编译后，会放置在堆栈里。通信时如果使用了较大的数组变量做缓存，就可能需要调整系统堆栈避免溢出。

9、非公开声明

- 本参考例程及其相关资料（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。凡是经过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其相关资料的授权。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。 谢谢理解和配合！
- 如有其他问题，可查阅集成说明文档或随时联系我们。也欢迎大家支持我们、积极和我们沟通，对我们提出宝贵意见或建议。谢谢！
- AnyLinkin!
- Link anytime, anywhere, and anyhow!
- <IoT@anylinkin.com>
- http://www.anylinkin.com
- htpp://anylinkin.taobao.com