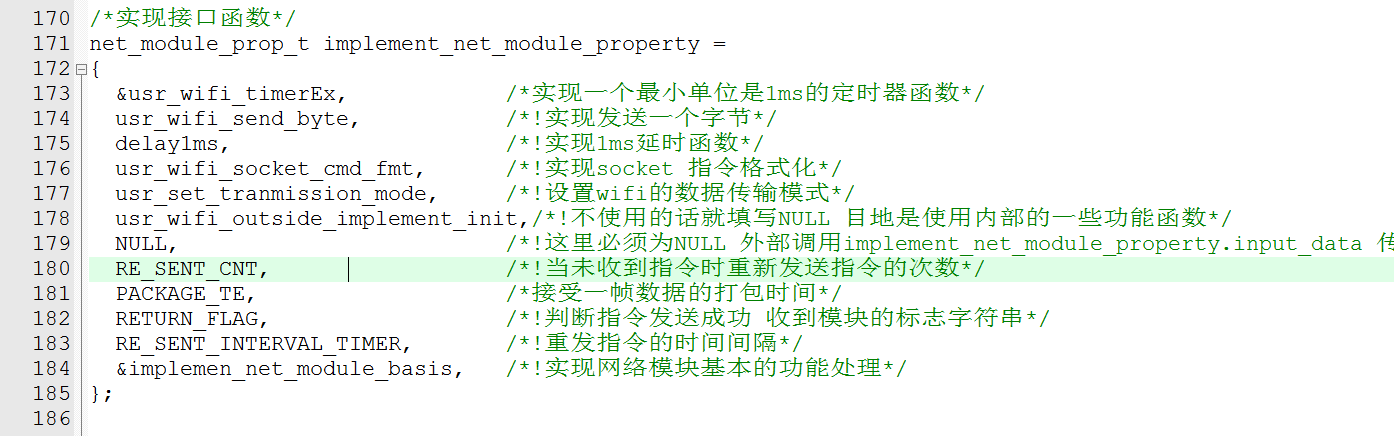
通用网络模块处理V18.0.0.0

前言:

当我们在使用不同的串口网络模块的时候,每个厂家的模块AT指令都用所不同,但在实际使用的过程中往往通信的模式,模块固有功能操作都基本相同,所以考虑有必要写一个抽象公用类来处理标准事件.实际使用的过程中我们只需要实现不同的AT指令,而不要需要改主体逻辑,大大提高程序的可维护性,提高迭代开发的速度.

第一步:

实现函数功能 主要实现这几个函数以及参数.



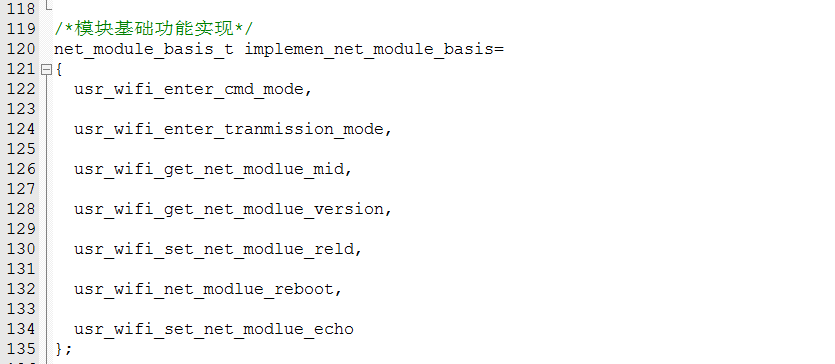
第二步:

要实现一个定时器函数 功能:自动重载,函数回调



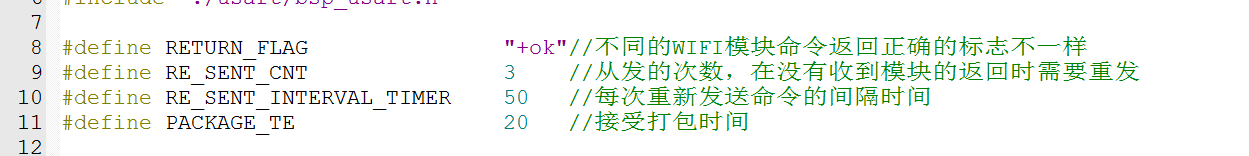
第三步:

实现模块基础功能 ,如果不用可以不实现



第四步:

宏定义说明



举例:

#include "string.h"

#include "user\_wifi.h"

void net\_module\_system\_cb(net\_module\_callback\_t\* p\_data)

{

Static u8 test\_buf[25];

/\*数据回调 注意指令数据 和透传数据都会从这里回调 注意区分一下不要处理延时太大的函数\*/

if(p\_data->receive\_data.data != NULL)

{

p\_data->receive\_data.data[p\_data->receive\_data.length] = 0x00;

memcpy(test\_buf,p\_data->receive\_data.data,p\_data->receive\_data.length);

}

}

Int main(void)

{

net\_module\_inside\_fun\_t \*wifi\_net\_module;

/\*初始化模块 指令缓存64byte 接收缓存128byte\*/

wifi\_net\_module = init\_net\_module(net\_module\_system\_cb,64,128);

/\*进入命令模式\*/

while(!wifi\_net\_module->net\_module\_basis\_handler()->enter\_cmd\_mode());

/\*参数设置\*/

wifi\_set\_bund("115200");

wifi\_set\_te("20");

set\_wifi\_work\_mode("STA");

/\*建立socket连接\*/

wifi\_net\_module->socket(UDP,CLIENT,"255.255.255.255","8080");

for(;;)

{

/\*开始发送数据\*/

wifi\_net\_module->hex\_data\_tranmission("hello world\r\n",sizeof("hello world\r\n"));

delay\_ms(25);

}

/\*主要用到的其他函数\*/

//char\* str = wifi\_net\_module->send\_cmd(“AT+E\r\n”);//str 是模块执行指令返回的结果

// wifi\_net\_module->send\_cmd\_check(AT+E\r\n,”+ok”); //判断收到数据的有效性可以用来单独发送指令

}

**函数说明:**

**1.网络模块初始化函数**

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名 | 参数1 | 参数2 | 参数3 |
| init\_net\_module | \*ptr | at\_buf\_size | receive\_buf\_size |
| 类型 | 函数指针 | Uint16\_t | Uint16\_t |
| 说明 | 传入函数回调异常事件,函数形参类型是net\_module\_callback\_t  不使用填写NULL | At命令的缓存大小 | 模块数据接受缓存大小 |
| 原型 | **void init\_net\_module(void (\*ptr)(net\_module\_callback\_t\*),uint8\_t at\_buf\_size,uint16\_t receive\_buf\_size)** | | |
| 返回值 | **net\_module\_inside\_fun\_t** | | |
| 说明 | 一级子类函数:   |  |  | | --- | --- | | 函数名 | 说明 | | Socket | 网络连接用 | | hex\_data\_tranmissi | 输出传输用 | | send\_cmd\_check | 发送数据用 | | send\_cmd | 发送数据用 |   二级子类函数:   |  |  | | --- | --- | | 函数名 | 说明 | | enter\_cmd\_mode | 进入命令模式 | | hex\_data\_tranmissi | 数据传输 | | enter\_tranmission\_mode | 进入传输模式 | | get\_net\_modlue\_mid | 得到模块ID | | get\_net\_modlue\_version | 得到模块版本 | | set\_net\_modlue\_reld | 恢复出厂设置 | | net\_modlue\_reboot | 重启模块 | | set\_net\_modlue\_echo | 回显开启关闭 | | | |

**一级子类函数说明**

**2.套接字函数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | 参数1 | 参数2 | 参数3 | 参数3 |
| **socket** | protocol | mode | ip | port |
| 类型 | SOCKET\_PROTOCOL\_U | CS\_MODE\_U | int8\* | int8\* |
| 说明 | UDP或TCP | SERVER或CLIENT | Ip,输入字符串 | 端口,输入字符串 |
| 原型 | **uint8\_t socket(SOCKET\_PROTOCOL\_U protocol,CS\_MODE\_U mode,int8\* ip,int8\* port)** | | | |
| 返回值 | **1:成功**  **0:失败** | | | |
| 函数说明 | Socket函数一旦使用,将会自动设置相关的参数,然后让模块进入透传模式,socket具有自动判断模块的传输模式,用户不用了解细节,socket帮用户处理了繁琐的逻辑.当模块不正常的时候Socket会发起20次重新连接,直到返回异常事件 | | | |

**3.检查指令函数**

|  |  |  |
| --- | --- | --- |
| 函数名 | 参数1 | 参数2 |
| **send\_cmd\_check** | cmd | return\_str |
| 类型 | int8\* | int8\* |
| 说明 | 输入的指令 | 判断模块返回的字符中是否有return\_str 字符串  填写为NULL的话,判断收到数据只要有数据就返回真 |
| 原型 | **uint8\_t send\_cmd\_check (int8\* cmd,int8\* return\_str)** | |
| 返回 | **1:成功**  **0失败** | |
| 函数说明 | 函数具有指令重发功能,当模块不正常或者出现数据延迟函数将会重新发送RE\_SENT\_CNT 次,直到收到返回数据为止, RE\_SENT\_CNT必须在使用之前设置好.每次间隔RE\_SENT\_INTERVAL\_TIMER次 ,也必须事先设置好 | |

**4. 指令发送函数**

|  |  |
| --- | --- |
| 函数名 | 参数1 |
| **send\_cmd** | Str |
| 类型 | int8\* |
| 说明 | AT指令字符串 |
| 原型 | **Int8\* send\_cmd (char\* str)** |
| 返回 | **Int8\* 返回执行指令的结果** |

**5.HEX发送函数**

|  |  |  |
| --- | --- | --- |
| 函数名 | 参数1 | 参数2 |
| **hex\_data\_tranmission** | data | length |
| 类型 | uint8\_t\* | uint16\_t |
| 说明 | 需要发送HEX数据 | 需要发送的字节数 |
| 原型 | void hex\_data\_tranmission(uint8\_t \*data,uint16\_t length) | |
| 函数说明 | 透明传输HEX发送函数 | |