

淘宝店铺

优秀不够，你是否无可替代

知识从未如此性感。烂程序员关心的是代码,好程序员关心的是数据结构和它们之间的关系 --QQ群: 607064330 --本人

QQ:946029359 --淘宝 <https://shop411638453.taobao.com/>

随笔 - 693, 文章 - 0, 评论 - 311, 阅读 - 173万

导航

博客园

首页

新随笔

联系

订阅 

管理

公告



渡我不渡她 -
Not available
00:00 / 00:00

- 渡我不渡她
- 小镇姑娘
- PDD洪荒之力



昵称：杨奉武
 园龄：5年8个月
 粉丝：607
 关注：1

搜索

我的标签

8266(88)
 MQTT(50)
 GPRS(33)
 SDK(29)
 Air202(28)
 云服务器(21)
 ESP8266(21)
 Lua(18)
 小程序(17)
 STM32(16)
 更多

随笔分类

Android(22)
 Android 开发(8)
 C# 开发(4)
 CH395Q学习开发(5)
 ESP32学习开发(8)
 ESP8266 AT指令开发(基于STC89C52单片机)(3)
 ESP8266 AT指令开发(基于STM32)(1)
 ESP8266 AT指令开发基础入门篇备份(12)
 ESP8266 LUA脚本语言开发(13)

5-网络芯片CH395Q学习开发-模块使用Socket0作为TCP客户端和电脑上位机TCP服务器局域网通信

网络芯片CH395Q学习开发

开发板链接:[开发板链接](#)

模组原理图:[模组原理图](#)

资料源码下载链

接:<https://github.com/yangfengwu45/CH395Q.g>

- [学习Android](#)
 教程中搭配的Android，C#等教程如上，各个教程正在整理。
- [1-硬件测试使用说明](#)
- [2-学习资料说明,测试通信,获取硬件版本,程序移植说明](#)
- [3-芯片初始化,网线连接检测实验](#)
- [4-关于中断检测和DHCP实验](#)
- [5-模块使用Socket0作为TCP客户端和电脑上位机TCP服务器局域网通信](#)
-
-
-
-
-
-

ESP8266 LUA开发基础入门篇
备份(22)
ESP8266 SDK开发(32)
ESP8266 SDK开发基础入门篇
备份(30)
GPRS Air202 LUA开发(11)
HC32F460(华大) +
BC260Y(NB-IOT) 物联网开发
(5)
NB-IOT Air302 AT指令和LUA
脚本语言开发(25)
PLC(三菱PLC)基础入门篇(2)
STM32+Air724UG(4G模组)
物联网开发(43)
STM32+BC26/260Y物联网开
发(37)
STM32+ESP8266(ZLESP8266/
物联网开发(1)
STM32+ESP8266+AIR202/30:
远程升级方案(16)
STM32+ESP8266+AIR202/30:
终端管理方案(6)
STM32+ESP8266+Air302物
联网开发(58)
STM32+W5500+AIR202/302
基本控制方案(25)
STM32+W5500+AIR202/302
远程升级方案(6)
UCOSii操作系统(1)
W5500 学习开发(8)
编程语言C#(11)
编程语言Lua脚本语言基础入
门篇(6)
编程语言Python(1)
单片机(LPC1778)LPC1778(2)
单片机(MSP430)开发基础入门
篇(4)
单片机(STC89C51)单片机开发
板学习入门篇(3)
单片机(STM32)基础入门篇(3)
单片机(STM32)综合应用系列
(16)
电路模块使用说明(10)
感想(6)
软件安装使用: MQTT(8)
软件安装使用: OpenResty(6)
数据处理思想和程序架构(24)
数据库学习开发(12)
更多

最新评论

1. Re:C#委托+回调详解
好文，撒也不说了，直接收
藏！
--杨咩咩plus
2. Re:2-STM32 替换说明-
CKS32, HK32, MM32,
APM32, CH32, GD32,
BLM32, AT32(推荐), N32,
HC华大系列
有用，谢谢！
--你跟游戏过吧

阅读排行榜

1. ESP8266使用详解(AT,LUA,
SDK)(172068)
2. 1-安装MQTT服务器(Windo
ws),并连接测试(96473)
3. ESP8266刷AT固件与node
mcu固件(63738)

说明

这节演示一下模组作为TCP客户端和电脑上位机TCP服
务器局域网通信

提醒:无论是SPI,USART,并口,程序操作步骤都是一样的!

只是不同的接口发指令发给模块,然后用不同的接收接收
数据而已.

测试本节代码

1.用户可以使用杜邦线根据自己的情况设置和连接引脚

- 4. 用ESP8266+android,制作自己的WIFI小车(ESP8266篇)(62517)
- 5. 有人WIFI模块使用详解(38092)
- 6. (一)基于阿里云的MQTT远程控制(Android 连接MQTT服务器,ESP8266连接MQTT服务器实现远程通信控制----简单的连接通信)(35372)
- 7. 关于TCP和MQTT之间的转换(32223)
- 8. android 之TCP客户端编程(31277)
- 9. android客服端+eps8266+单片机+路由器之远程控制系统(31126)
- 10. C#中public与private与static(30925)

推荐排行榜

- 1. C#委托+回调详解(9)
- 2. 用ESP8266+android,制作自己的WIFI小车(ESP8266篇)(8)
- 3. 用ESP8266+android,制作自己的WIFI小车(Android 软件)(6)
- 4. ESP8266使用详解(AT,LUA,SDK)(6)
- 5. 关于TCP和MQTT之间的转换(5)

```
ch395cmd.h CH395INC.H CH395SPI.C usart.c delay.c timer.c main.c delay.h CH395SPI.H
2 #ifndef CH395SPI_H_
3 #define CH395SPI_H_
4
5 #include "CH395INC.H"
6
7 //*****配置GPIO (根据自己的修改)*****
8 //时钟
9 #define CH395_CONFIG_SPI_CLK() ( RCC_APB1PeriphClockCmd( RCC_APB1Periph_SPI2,ENABLE) )
10 #define CH395_CONFIG_GPIO_CLK() ( RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOA | RCC_APB2Peri
11 //设置使用的SPI
12 #define USE_SPI SPI2
13 //SPI_CS -- 连接模块SCS引脚
14 #define CH395_CS_PORT GPIOB
15 #define CH395_CS_PIN GPIO_Pin_12
16 //SPI_CLK -- 连接模块SCK引脚
17 #define CH395_CLK_PORT GPIOB
18 #define CH395_CLK_PIN GPIO_Pin_13
19 //SPI_MISO -- 连接模块SDO引脚
20 #define CH395_MISO_PORT GPIOB
21 #define CH395_MISO_PIN GPIO_Pin_14
22 //SPI_MOSI -- 连接模块SDI引脚
23 #define CH395_MOSI_PORT GPIOB
24 #define CH395_MOSI_PIN GPIO_Pin_15
25 //RST -- 连接模块RST引脚
26 #define CH395_RST_PORT GPIOA
27 #define CH395_RST_PIN GPIO_Pin_8
28 //TX -- 连接模块Tx引脚
29 #define CH395_TX_PORT GPIOA
30 #define CH395_TX_PIN GPIO_Pin_3
31 //INT -- 连接模块INT引脚 (检测到该引脚低电平信号之后再获取数据)
32 #define CH395_INT_PORT GPIOA
33 #define CH395_INT_PIN GPIO_Pin_0
34 /*****/
```

2,注意!

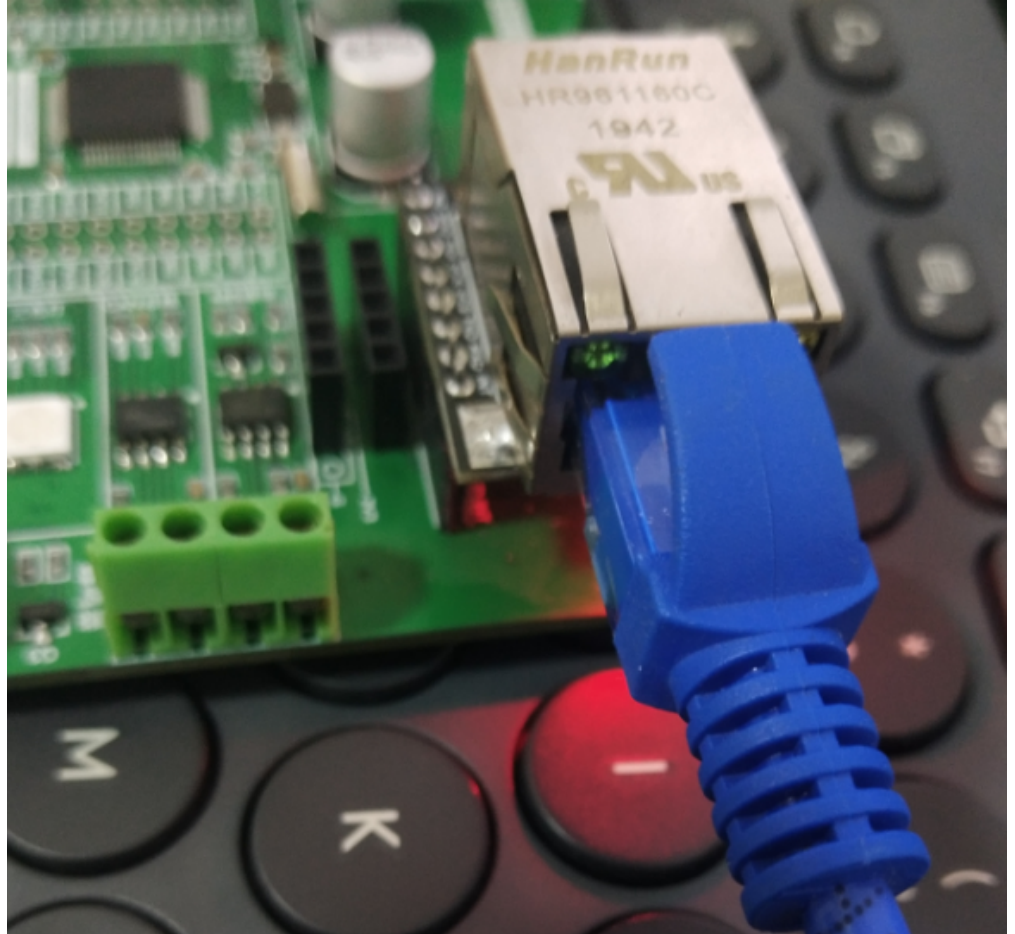
要想模块使用SPI通信,模块的TX引脚需要在模块重启之前设置为低电平.

上面的引脚分配把模块的TX引脚接到了单片机的PA3上,也就是串口2的RX上,如果用户使用了串口2,请注意!

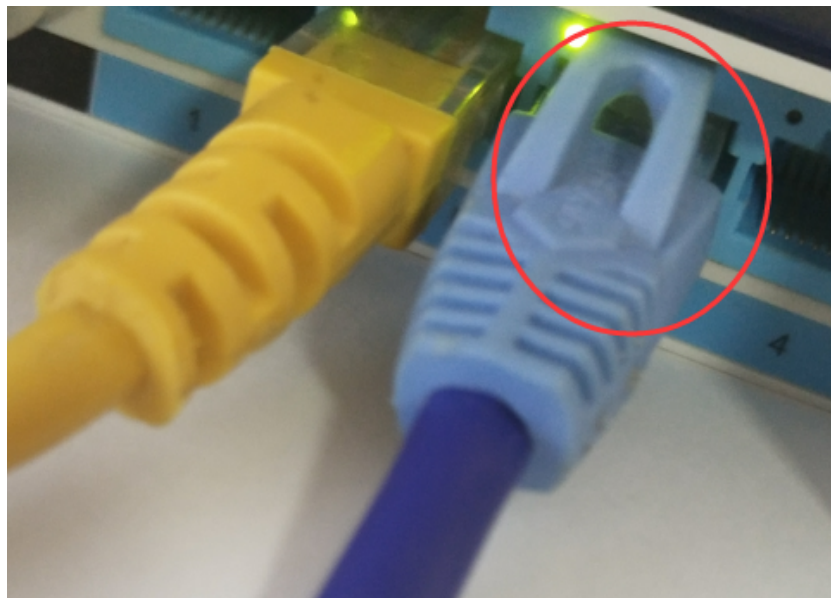
CH395 与单片机之间支持三种通讯接口: 8 位并行接口、SPI 同步串行接口、异步串口。在芯片上电复位时, CH395 将采样 SEL 和 TXD 引脚的状态, 根据这 2 个引脚状态的组合选择通讯接口, 参考下表 (表中 X 代表不关心此位, 0 代表低电平, 1 代表高电平或者悬空)。

SEL 引脚	TXD 引脚	选择通讯接口
1	1	异步串口
1	0	SPI 接口
0	1	8 位并口
0	0	错误接口

3.把模块用网线和路由器或者交换机(和上位机在同一个局域网下)



注意,连接路由器或者交换机的时候是连接其LAN口.

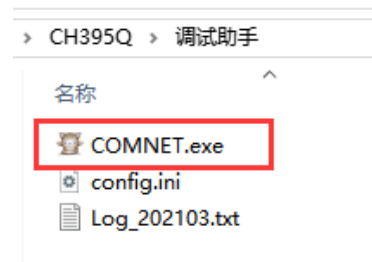




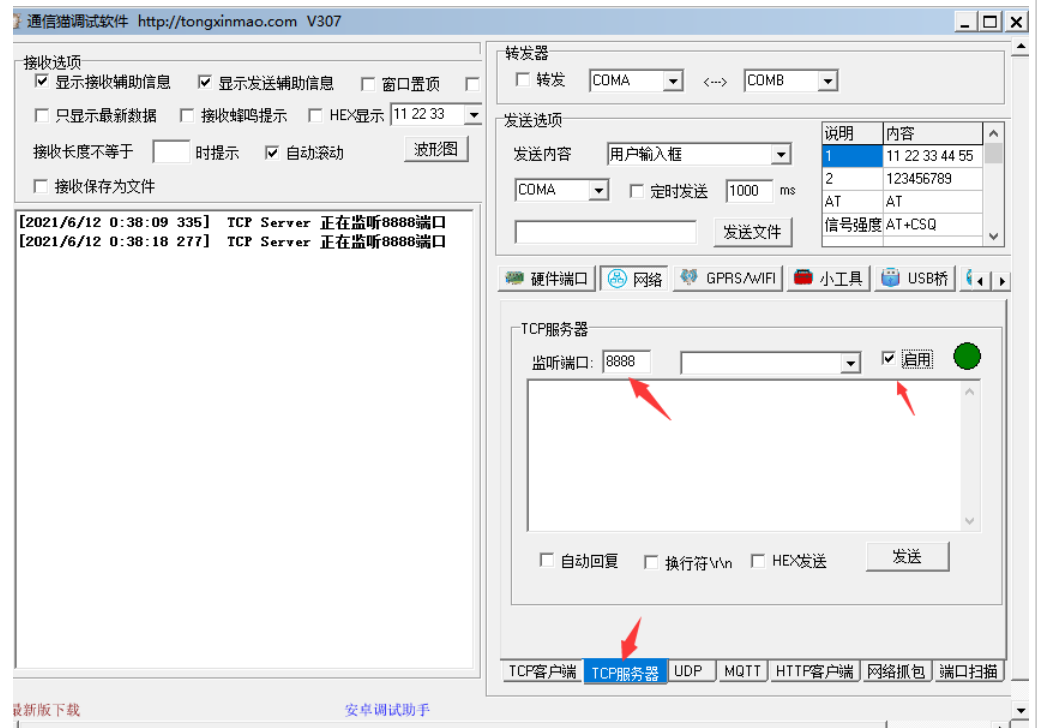
WAN端口：连接网线

LAN端口：连接电脑（任选一个端口就行）

4,在电脑上运行网络调试助手,开启TCP服务器



我设置监听的端口为8888



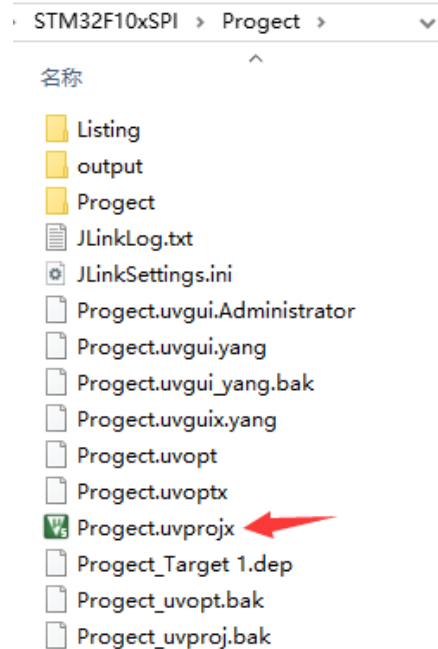
5.查看自己电脑的IP地址

我的为 192.168.0.103

所以我的TCP服务器地址为 192.168.0.103,端口号为8888



6,打开这节程序



7,根据自己的修改服务器IP地址和端口号

```
timer.c  usart.c  main.c  CH395CMD.H  CH395INC.H  CH395CMD.C  timer.h

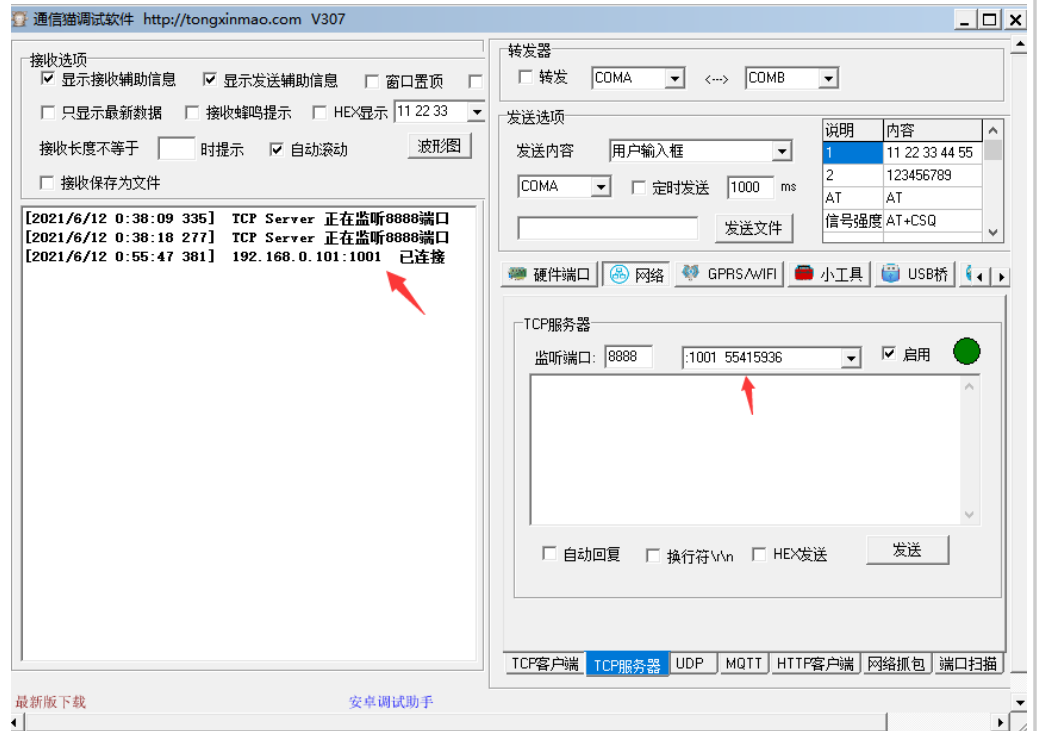
22 #include "CH395SPI.H"
23 #include "CH395INC.H"
24 #include "CH395CMD.H"
25 /*提示: (只是提示!这节使用的Socket0通信,并没有人为分配缓存区)
26 芯片共有48块缓存区,每个缓存区512字节
27 芯片共有8个Socket,默认把48块缓存区分给了Socket0,Socket1,Socket2,Socket3
28 这四个Socket,每个 Socket 使用8块缓存区作为接收,4块缓存区作为发送,
29 即Socket0,Socket1,Socket2,Socket3的接收区各为512*8 = 4KB
30 即Socket0,Socket1,Socket2,Socket3的发送区各为512*4 = 2KB
31 如果要使用Socket4,Socket5,Socket6,Socket7需要重新分配缓存区
32 */
33
34
35 /*存储网络接收的数据*/
36 #define recv_buff_len 1500
37 unsigned char recv_buff[recv_buff_len];
38
39 char ch395_version=0;//获取版本号
40
41 unsigned char buf[20];
42 int ch395_status=0;//获取中断事件
43
44 /* socket 相关定义*/
45 UINT8 SocketIndex = 0; /* Socket 索引(0,1,2,3,4,5,6,7) */
46 UINT8 SocketDesIP[4] = {192,168,0,103}; /* Socket 目的IP地址 */
47 UINT16 SocketDesPort = 8888; /* Socket 目的端口 */
48 UINT8 SocketStatus = 0; /*Socket状态 0:未连接服务器;1:连接上服务器 */
49
50 /* Socket 本地端口,初始化默认端口号 */
51 UINT16 SocketLocalPort = 1000;
52 /*动态获取本地端口号,每次获取端口号累加*/
53 UINT16 ch395_socket_tcp_client_port(void)
```

8.下载到单片机,单片机串口1作为日志打印口

连接上服务器会显示

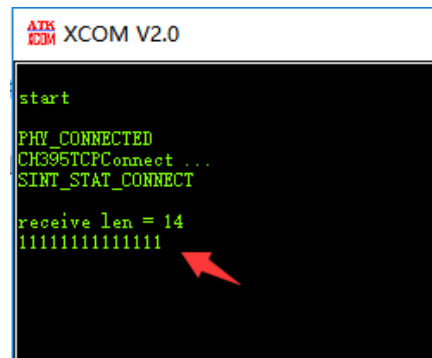
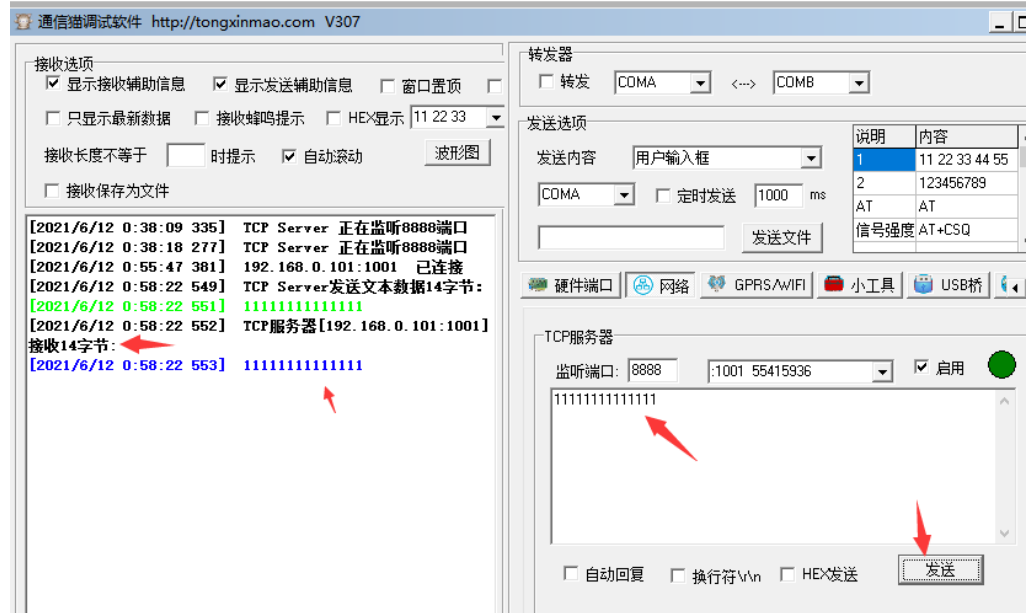

```
start
PHY_CONNECTED
CH395TCPConnect ...
SINT_STAT_CONNECT
```

调试助手会显示连接



9.服务器给客户端发送消息

单片机程序里面写的是把接收的服务器返回给服务器,并使用串口打印接收的消息



程序说明

1.初始化

```
timer.c  usart.c  main.c  CH395CMD.H  CH395INC.H  CH395...
151
152 int main(void)
153 {
154     NVIC_Configuration();
155     uart_init(115200); //串口初始化为115200
156     delay_init();
157     timer2_config();
158
159     //初始化CH395使用的GPIO
160     CH395_PORT_INIT();
161     //复位 CH395
162     CH395_RST();
163
164     IWDG_Init(IWDG_Prescaler_256,156*10);
165
166     /*获取芯片版本*/
167     while((ch395_version = CH395CMDGetVer()) < 0x40)
168     {
169         printf("CH395CMDGetVer ERR\r\n");
170         delay_ms(100);
171     }
172
173     /*测试命令,按位取反返回说明测试通过*/
174     while(CH395CMDCheckExist(0x55) != 0xaa)
175     {
176         printf("\r\nCH395CMDCheck ERR\r\n");
177         delay_ms(100);
178     }
179
180     /*初始化模块:成功返回 0 */
181     while(CH395CMDInitCH395() != 0)
182     {
183         printf("\r\nCH395CMDInitCH395 ERR\r\n");
184         delay_ms(100);
185     }
186
187     printf("\r\nstart\r\n");
188     while(1)
189     {
190         IWDG_Feed();//喂狗
191     }
```

2.因为是局域网,连接了路由器,所以需要启用DHCP

```
timer.c  usart.c  main.c  CH395CMD.H  CH395INC.H  CH395CMD.C  timer.h
207
208 //INT引脚产生低电平中断以后进去判断
209 if(Query395Interrupt())
210 {
211     /*获取中断事件*/
212     if(ch395_version>=0x44)
213     {
214         ch395_status = CH395CMDGetGlobIntStatus_ALL();
215     }
216     else
217     {
218         ch395_status = CH395CMDGetGlobIntStatus();
219     }
220
221     /* 处理PHY改变中断*/
222     if(ch395_status & GINT_STAT_PHY_CHANGE)
223     {
224         if(CH395CMDGetPHYStatus() == PHY_DISCONN)//网线断开
225         {
226             printf("\r\nPHY_DISCONN\r\n");
227         }
228         else//网线连接
229         {
230             printf("\r\nPHY_CONNECTED\r\n");
231             CH395DHCPEnable(1);//启动DHCP
232         }
233     }
234
235     /* 处理DHCP/PPPOE中断 */
236     if(ch395_status & GINT_STAT_DHCP)
237     {
238         if(CH395GetDHCPStatus() == 0)//DHCP OK
239         {
240             //
241         }
242     }
243
244     /* 处理不可达中断，读取不可达信息 */
245     if(ch395_status & GINT_STAT_UNREACH){
246         CH395CMDGetUnreachIPPT(buf);
247     }
248
249     /* 处理IP冲突中断，建议重新修改CH395的 IP，并初始化CH395*/
250     if(ch395_status & GINT_STAT_IP_CONFLI){
251         //
252     }
253
254     /* 处理 SOCK0 中断 */
255     if(ch395_status & GINT_STAT_SOCKET0){
256         ch395_socket_tcp_client_interrupt(SocketIndex);
257     }
258 }
```

3.每隔8S判断,如果Socket没有连接,则初始化Socket和控制Socket连接服务器

```
timer.c  usart.c  main.c  CH395CMD.H  CH395INC.H  CH395CMD.C  timer.h
189 {
190     IWDG_Feed();//喂狗
191
192     /*每隔8S初始化Socket并执行连接函数*/
193     /*芯片内部连接超时时间为5S,必须大于此时间*/
194     if(Timer2Cnt>8000)
195     {
196         Timer2Cnt = 0;
197         if(!SocketStatus)
198         {
199             if(ch395_socket_tcp_client_init(SocketIndex,SocketDesIP,SocketDesPort,ch395_socket_tcp_client_port()) == 0)
200             {
201                 printf("CH395TCPConnect ... \r\n");
202                 CH395TCPConnect(SocketIndex);//连接服务器
203             }
204         }
205     }
206 }
```

```
timer.c  usart.c  main.c  CH395CMD.H  CH395INC.H  CH395CMD.C  timer.h

36 #define recv_buff_len 1500
37 unsigned char recv_buff[recv_buff_len];
38
39 char ch395_version=0;//获取版本号
40
41 unsigned char buf[20];
42 int ch395_status=0;//获取中断事件
43
44 /* socket 相关定义 */
45 UINT8 SocketIndex = 0; /* Socket 索引 (0,1,2,3,4,5,6,7) */
46 UINT8 SocketDesIP[4] = {192,168,0,103}; /* Socket 目的IP地址 */
47 UINT16 SocketDesPort = 8888; /* Socket 目的端口 */
48 UINT8 SocketStatus = 0; /*Socket状态 0:未连接服务器;1:连接上服务器 */
49
50 /* Socket 本地端口,初始化默认端口号 */
51 UINT16 SocketLocalPort = 1000;
52 /*动态获取本地端口号,每次获取端口号累加*/
53 UINT16 ch395_socket_tcp_client_port(void)
54 {
55     SocketLocalPort++;
56     if(SocketLocalPort>65535) SocketLocalPort = 1000;
57     return SocketLocalPort;
58 }
59
60 /**
61  * @brief 初始化socket
62  * @param sockindex Socket索引 (0,1,2,3,4,5,6,7)
63  * @param ipaddr 目的地址
64  * @param desprot 目的端口号
65  * @param surprot 本地端口号
66  * @retval 0:初始化成功; others:初始化失败
67  * @warning None
68  * @example
69  */
70 char ch395_socket_tcp_client_init(UINT8 sockindex,UINT8 *ipaddr,UINT16 desprot,UINT16 surprot)
71 {
72     UINT8 i;
73     CH395SetSocketDesIP(sockindex,ipaddr); /* 目的地址 */
74     CH395SetSocketProtType(sockindex,PROTO_TYPE_TCP); /* 协议类型 */
75     CH395SetSocketDesPort(sockindex,desprot); /* 目的端口号 */
76     CH395SetSocketSourPort(sockindex,surprot); /* 本地端口号 */
77     if (CH395OpenSocket(sockindex) !=0) /* 打开Socket */
78     {
79         return 1;
80     }
81     return 0;
82 }
83
```

4.在中断检测事件里面处理Socket相关事件

```
timer.c  usart.c  main.c  CH395CMD.H  CH395INC.H  CH395CMD.C  timer.h

234
235      /* 处理DHCP/PPPOE中断 */
236      if(ch395_status & GINT_STAT_DHCP)
237      {
238          if(CH395GetDHCPStatus() == 0) //DHCP OK
239          {
240              }
241          }
242
243      /* 处理不可达中断, 读取不可达信息 */
244      if(ch395_status & GINT_STAT_UNREACH){
245          CH395CMDGetUnreachIPPT(buf);
246      }
247
248      /* 处理IP冲突中断, 建议重新修改CH395的 IP, 并初始化CH395*/
249      if(ch395_status & GINT_STAT_IP_CONFLI){
250
251      }
252      /* 处理 SOCK0 中断 */
253      if(ch395_status & GINT_STAT_SOCK0){
254          ch395_socket_tcp_client_interrupt(SocketIndex);
255      }
256      /* 处理 SOCK1 中断 */
257      if(ch395_status & GINT_STAT_SOCK1){
258
259      }
260      /* 处理 SOCK2 中断 */
261      if(ch395_status & GINT_STAT_SOCK2){
262
263      }
264      /* 处理 SOCK3 中断 */
265      if(ch395_status & GINT_STAT_SOCK3){
266
267      }
```

```

timer.c  usart.c  main.c  CH395CMD.H  CH395INC.H  CH395CMD.C  timer.h
83
84
85 /**
86  * @brief   socket处理函数(把此函数放到全局socket中断里面)
87  * @param   sockindex   Socket索引(0,1,2,3,4,5,6,7)
88  * @param   None
89  * @param   None
90  * @param   None
91  * @retval   None
92  * @warning  None
93  * @example
94  */
95 void ch395_socket_tcp_client_interrupt(UINT8 sockindex)
96 {
97     UINT8  sock_int_socket;
98     UINT16 len;
99
100    /* 获取socket 的中断状态 */
101    sock_int_socket = CH395GetSocketInt(sockindex);
102
103    /* 发送缓冲区空闲, 可以继续写入要发送的数据 */
104    if(sock_int_socket & SINT_STAT_SENBUF_FREE)
105    {
106    }
107
108    /* 发送完成中断 */
109    if(sock_int_socket & SINT_STAT_SEND_OK)
110    {
111    }
112
113    /* 接收数据中断 */
114    if(sock_int_socket & SINT_STAT_RECV)
115    {
116        len = CH395GetRecvLength(sockindex);/* 获取当前缓冲区内数据长度 */
117        printf("\r\nreceive len = %d\r\n",len);
118        if(len == 0)return;
119        if(len > recv_buff_len)len = recv_buff_len;
120        CH395GetRecvData(sockindex,len,recv_buff);/* 读取数据 */
121
122        /*把接收的数据发送给服务器*/
123        CH395SendData(sockindex,recv_buff,len);
124
125        /*使用串口打印接收的数据*/

```

```

124
125    /*使用串口打印接收的数据*/
126    PutData(&rb_t_usart1_send,recv_buff,len);
127    USART_ITConfig(USART1, USART_IT_TXE, ENABLE);
128    }
129
130    /* 连接中断, 仅在Tcp模式下有效*/
131    if(sock_int_socket & SINT_STAT_CONNECT)
132    {
133        SocketStatus = 1;//设置连接状态为连接
134        printf("SINT_STAT_CONNECT\n");
135    }
136
137    /* 断开中断, 仅在Tcp模式下有效 */
138    if(sock_int_socket & SINT_STAT_DISCONNECT)
139    {
140        printf("SINT_STAT_DISCONNECT \n");
141        SocketStatus = 0;//设置连接状态为未连接
142    }
143
144    /* 超时中断, 仅在Tcp模式下有效 ,TCP CLIENT无法顺利连接服务器端会进入此中断*/
145    if(sock_int_socket & SINT_STAT_TIM_OUT)
146    {/*此时可以把Socket源端口号进行自加处理, 以新的端口去连接服务器*/
147        printf("SINT_STAT_TIM_OUT\n");
148        SocketStatus = 0;//设置连接状态为未连接
149    }
150    }
151
152

```

5.提示

在执行完CH395TCPConnect(SocketIndex);//连接服务器

以后,如果连接成功,会进入Socket连接成功回调,然后就是正常通信

```
195
196     Timer2Cnt = 0;
197     if(!SocketStatus)
198     {
199         if(ch395_socket_tcp_client_init(SocketIndex, Sock
200         {
201             printf("CH395TCPConnect ... \r\n");
202             CH395TCPConnect(SocketIndex); //连接服务器
203         }
204     }
205 }
206
```



```
122 /*把接收的数据发送给服务器*/
123 CH395SendData(sockindex, recv_buff, len);
124
125 /*使用串口打印接收的数据*/
126 PutData(&rb_t_usart1_send, recv_buff, len);
127 USART_ITConfig(USART1, USART_IT_TXE, ENABLE);
128 }
129
130 /* 连接中断, 仅在TCP模式下有效*/
131 if(sock_int_socket & SINT_STAT_CONNECT)
132 {
133     SocketStatus = 1; //设置连接状态为连接
134     printf("SINT_STAT_CONNECT\n");
135 }
136
137 /* 断开中断, 仅在TCP模式下有效 */
138 if(sock_int_socket & SINT_STAT_DISCONNECT)
139 {
140     printf("SINT_STAT_DISCONNECT \n");
141     SocketStatus = 0; //设置连接状态为未连接
142 }
143
144 /* 超时中断, 仅在TCP模式下有效, TCP CLIENT无法顺利连接服务器端会进入此中断*/
145 if(sock_int_socket & SINT_STAT_TIM_OUT)
146 { /*此时可以把Socket源端口号进行自加处理, 以新的端口去连接服务器*/
147     printf("SINT_STAT_TIM_OUT\n");
148     SocketStatus = 0; //设置连接状态为未连接
149 }
150 }
151
```

在执行完CH395TCPConnect(SocketIndex);//连接服务器

以后,如果连接超时(默认5S)会进入Socket连接超时回调

然后程序到了8S之后便会再次尝试连接.


```
timer.c  usart.c  main.c  CH395CMD.H  CH395INC.H  CH395CMD.C  timer.h
128      }
129
130      /* 连接中断, 仅在TCP模式下有效 */
131      if(sock_int_socket & SINT_STAT_CONNECT)
132      {
133          SocketStatus = 1; //设置连接状态为连接
134          printf("SINT_STAT_CONNECT\n");
135      }
136
137      /* 断开中断, 仅在TCP模式下有效 */
138      if(sock_int_socket & SINT_STAT_DISCONNECT)
139      {
140          printf("SINT_STAT_DISCONNECT \n");
141          SocketStatus = 0; //设置连接状态为未连接
142      }
143
144      /* 超时中断, 仅在TCP模式下有效, TCP CLIENT无法顺利连接服务器端会进入此中断 */
145      if(sock_int_socket & SINT_STAT_TIM_OUT)
146      { /*此时可以把Socket源端口号进行自加处理, 以新的端口去连接服务器 */
147          printf("SINT_STAT_TIM_OUT\n");
148          SocketStatus = 0; //设置连接状态为未连接
149      }
150  }
151
```

6.如果在其它位置发送数据,推荐的方式

```
/*Socket连接*/
if(SocketStatus)
{
    CH395SendData(SocketIndex, "111111", 6); //发送数据
}
```

分类: [CH395Q学习开发](#)

好文要顶

关注我

收藏该文



杨奉武

关注 - 1

粉丝 - 607

0

0

« 上一篇: [4-网络芯片CH395Q学习开发-关于中断检测和DHCP实验](#)

posted on 2021-06-12 01:16 杨奉武 阅读(0) 评论(0) 编辑 收藏 举报

[刷新评论](#) [刷新页面](#) [返回顶部](#)

发表评论

编辑

预览

B



提交评论

退出

[Ctrl+Enter快捷键提交]

【推荐】百度智能云618年中大促，限时抢购，新老用户同享超值折扣

【推荐】大型组态、工控、仿真、CAD\GIS 50万行VC++源码免费下载!

【推荐】阿里云爆品销量榜单出炉，精选爆款产品低至0.55折

【推荐】限时秒杀！国云大数据魔镜，企业级云分析平台

【推荐】华为应用软件专题日 | 生态市场企业特惠GO

园子动态：

- 致园友们的一封检讨书：都是我们的错
- 数据库实例 CPU 100% 引发全站故障
- 发起一个开源项目：博客引擎 fluss

最新新闻：

- 2021 苹果设计奖出炉！《原神》《英雄联盟》获奖，还有 10 个年度 App
 - K 歌、主播、B 站 UP 主..... 这一届「银发族」如何开启第二人生
 - 黑客可向邮件服务器发送数据以干扰HTTPS连接
 - AI产品正逐渐走向成熟，行业竞争已经出现两极分化
 - 解读滴滴招股书：提供“移动”价值的全球共享经济企业潜力几何？
- » 更多新闻...

Powered by:

博客园

Copyright © 2021 杨奉武

Powered by .NET 5.0 on Kubernetes



单片机,物联网,上位机,...

扫一扫二维码，入群聊。