

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 - 695, 文章 - 0, 评论 - 311, 阅读 - 173万

导航

博客园

首页

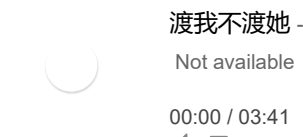
新随笔

联系

订阅 

管理

公告



1 渡我不渡她

2 小镇姑娘

3 PDD洪荒之力

 加入QQ群

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(7)
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)

7-网络芯片CH395Q学习开发-模块使用Socket0-5作为6路TCP客户端和电脑上位机TCP客户端局域网通信(Socket缓存区配置)

<p><iframe name="ifd" src="https://mnifdv.cn/resource/cnblogs/LearnCH395Q" frameborder="0" scrolling="auto" width="100%" height="1500"></iframe></p>

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.c>

- [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](#)
- [6-模块使用Socket0-3作为4路TCP客户端和电脑上位机TCP客户端局域网通信](#)
- [7-模块使用Socket0-5作为6路TCP客户端和电脑上位机TCP客户端局域网通信\(Socket缓存区配置\)](#)

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(96476)
3. ESP8266刷AT固件与node
mcu固件(63739)

说明

这节演示一下模组使用Socket0-5作为6路TCP客户端和
电脑上位机TCP服务器局域网通信

提示:其实这节和上一节没有太大差别,这节增加了配置
Socket缓存区的设置.

注意:只有在芯片版本大于4上才有Socket4-7功能,这节
代码只有芯片版本大于4才启动Socket4,5

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.

先学前面一路连接的!再来学这个多路的.

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)(62519)
- 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(31279)
- 9. android客服端+eps8266+单片机+路由器之远程控制系统(31129)
- 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!

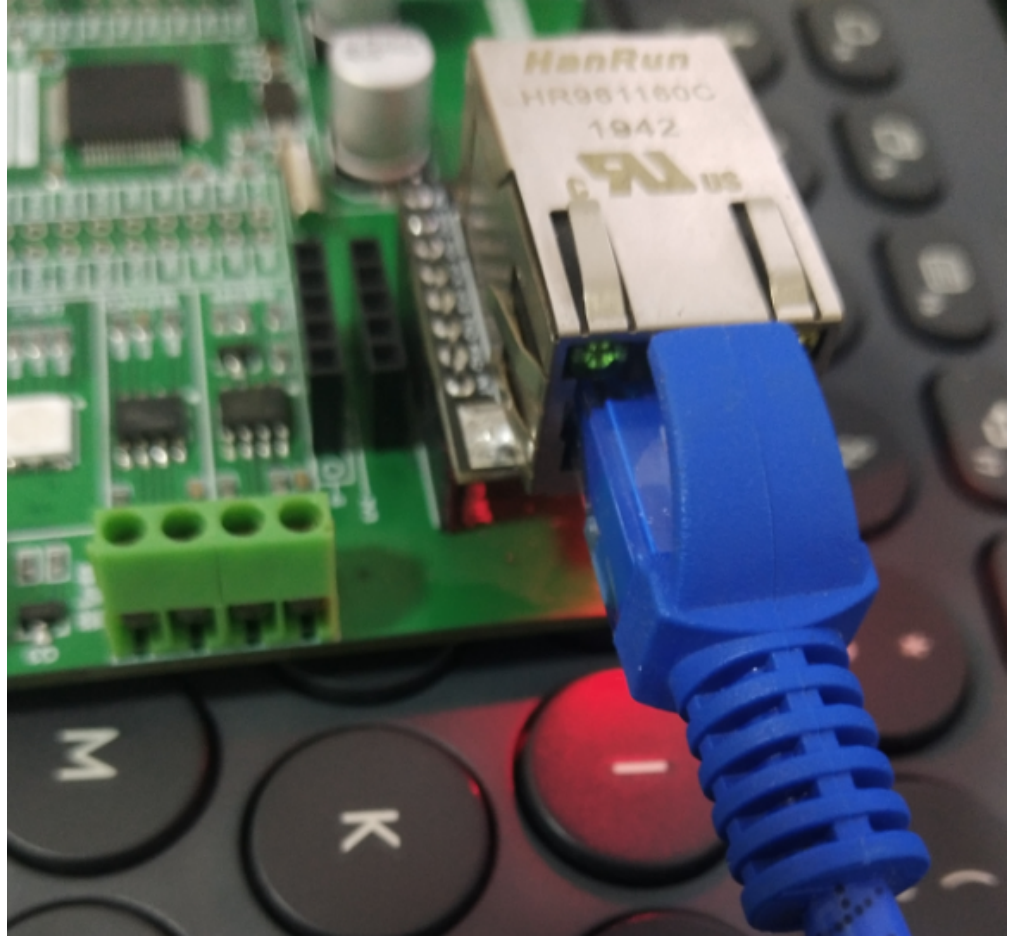
要想模块使用SPI通信,模块的TX引脚需要在模块重启之前设置为低电平.

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!

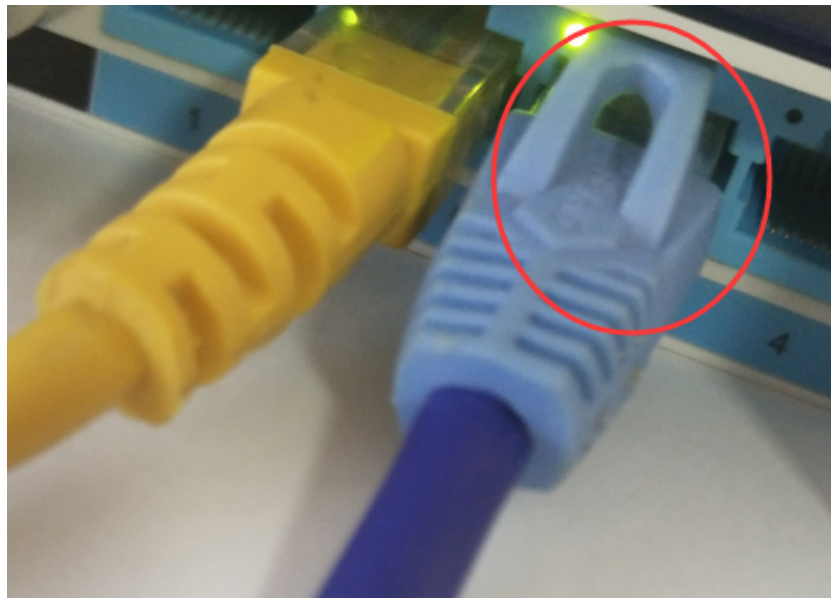
CH395 与单片机之间支持三种通讯接口: 8 位并行接口、SPI 同步串行接口、异步串口。在芯片上电复位时, CH395 将采样 SEL 和 TXD 引脚的状态, 根据这 2 个引脚状态的组合选择通讯接口, 参考下表 (表中 X 代表不关心此位, 0 代表低电平, 1 代表高电平或者悬空)。

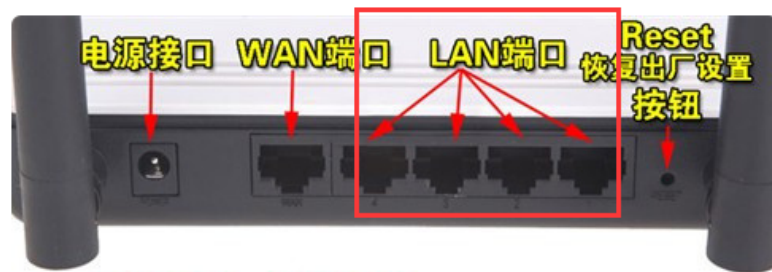
SEL 引脚	TXD 引脚	选择通讯接口
1	1	异步串口
1	0	SPI 接口
0	1	8 位并口
0	0	错误接口

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

换了个调试助手,上一个有点卡顿

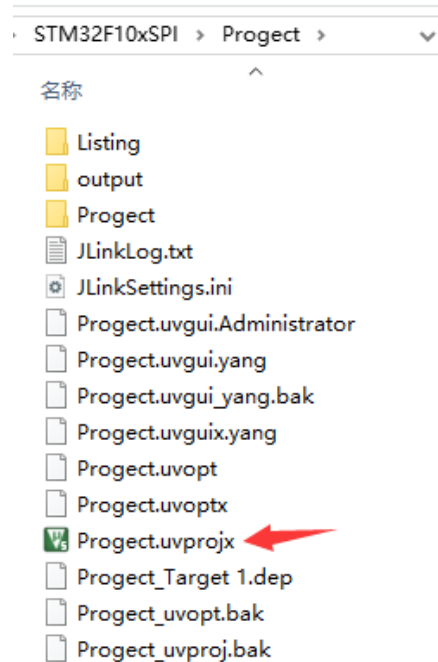
名称	修改日期
COMNET.exe	2021/6
config.ini	2021/6
Log_202103.txt	2021/6
TCPUDPDebug102_Setup.exe	2020/7

我设置监听的端口为8888

TCP服务器地址为 192.168.0.103



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

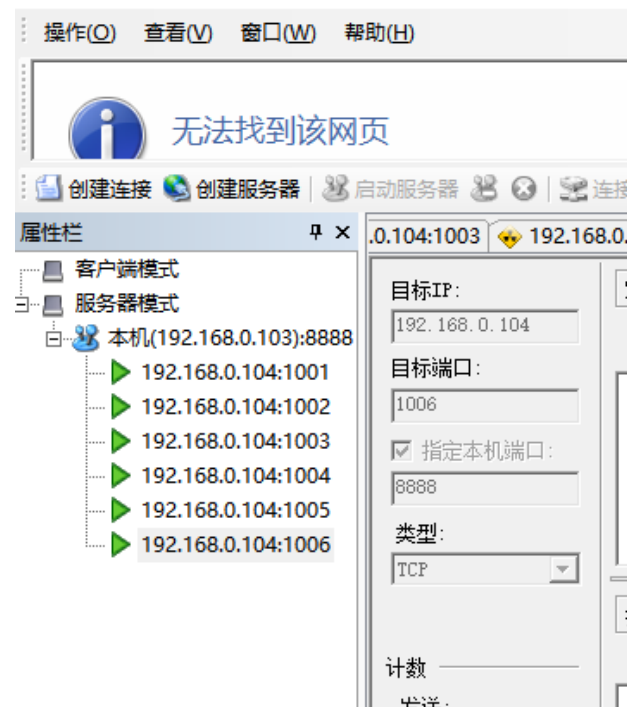
```
timer.c | usart.c | main.c | CH395CMD.H | CH395INC.H | CH395CMD.C | timer.h
34
35 /*存储网络接收的数据*/
36 #define recv_buff_len 1500
37 unsigned char recv_buff[recv_buff_len];
38
39 char ch395_version=0; //获取版本号
40
41 unsigned char buf[20];
42 int ch395_status=0; //获取中断事件
43
44 /* socket 相关定义 */
45 UINT8 SocketDesIP[4] = {192,168,0,103}; /* Socket 目的IP地址 */
46 UINT16 SocketDesPort = 8888; /* Socket 目的端口 */
47
48 UINT8 SocketStatus[6]={0,0,0,0,0,0}; /*Socket0-4状态 0:未连接服务器;1:连接上服务器 */
49
50
51 /*配置 Socket 发送和接收缓存区*/
52 /*芯片有0-47个块,每个块512字节大小*/
53 void socket_buffer_config(void )
54 {
55     /*Socket 0*/
56     CH395SetSocketRecvBuf(0,0,4); //第0,1,2,3块缓存区作为Socket的接收缓存区(512*4=2KB)
57     CH395SetSocketSendBuf(0,4,2); //第4,5块缓存区作为Socket的发送缓存区(512*1=1KB)
```

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如下

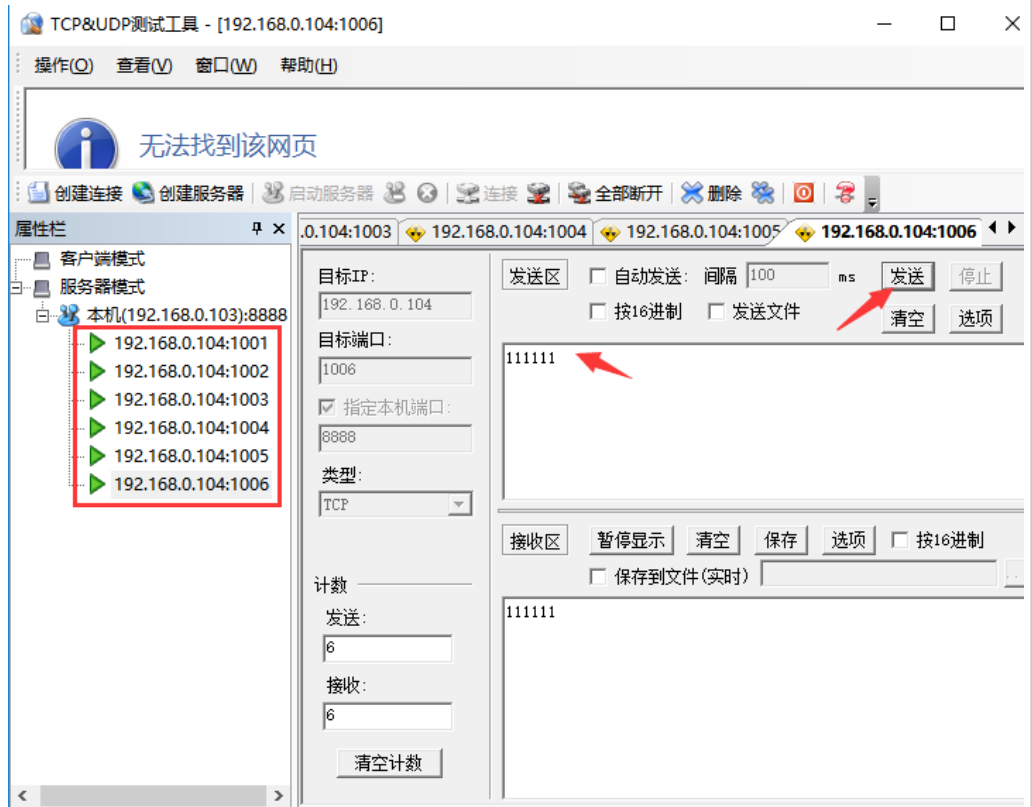
注意:版本号低于4的不会执行Socket4,5


```
ATK XCOM V2.0
CH395CMDGetVer =44
start
PHY_CONNECTED
CH395TCPConnect0 ...
CH395TCPConnect1 ...
CH395TCPConnect2 ...
CH395TCPConnect3 ...
CH395TCPConnect4 ...
CH395TCPConnect5 ...
socket0 SINT_STAT_CONNECT
socket1 SINT_STAT_CONNECT
socket2 SINT_STAT_CONNECT
socket3 SINT_STAT_CONNECT
socket4 SINT_STAT_CONNECT
socket5 SINT_STAT_CONNECT
```



9.服务器分别给4个客户端发送消息

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



```
socket0 SINT_STAT_CONNECT
socket1 SINT_STAT_CONNECT
socket2 SINT_STAT_CONNECT
socket3 SINT_STAT_CONNECT
socket4 SINT_STAT_CONNECT
socket5 SINT_STAT_CONNECT

socket0 receive len = 6
111111
socket1 receive len = 6
111111
socket2 receive len = 6
111111
socket3 receive len = 6
111111
socket4 receive len = 6
111111
socket5 receive len = 6
111111
```

程序说明

1.设置Socket缓存区域


```

timer.c  usart.c  main.c  CH395CMD.H  CH395INC.H  CH395CN
205     }
206     printf("CH395CMDGetVer =%2x\r\n",ch395_version);
207
208     /*测试命令，按位取反返回说明测试通过*/
209     while(CH395CMDCheckExist(0x55) != 0xaa)
210     {
211         printf("\r\nCH395CMDCheck ERR\r\n");
212         delay_ms(100);
213     }
214
215     /*初始化模块:成功返回 0 */
216     while(CH395CMDInitCH395() != 0)
217     {
218         printf("\r\nCH395CMDInitCH395 ERR\r\n");
219         delay_ms(100);
220     }
221
222     //配置 Socket 发送和接收缓存区大小
223     socket_buffer_config();
224
225     printf("\r\nstart\r\n");
226     while(1)
227     {
228         IWDG_Feed(); //喂狗
229

```

```

timer.c  usart.c  main.c  CH395CMD.H  CH395INC.H  CH395CMD.C  timer.h
43
44  /* socket 相关定义*/
45  UINT8  SocketDesIP[4] = {192,168,0,103}; /* Socket 目的IP地址 */
46  UINT16 SocketDesPort  = 8888;          /* Socket 目的端口 */
47
48  UINT8  SocketStatus[6]={0,0,0,0,0,0}; /*Socket0-4状态 0:未连接服务器;1:连接上服务器 */
49
50
51  /*配置 Socket 发送和接收缓存区*/
52  /*芯片有0-47个块,每个块512字节大小*/
53  void socket_buffer_config(void)
54  {
55      /*Socket 0*/
56      CH395SetSocketRecvBuf(0,0,4); //第0,1,2,3块缓存区作为Socket的接收缓存区 (512*4=2KB)
57      CH395SetSocketSendBuf(0,4,2); //第4,5块缓存区作为Socket的发送缓存区 (512*1=1KB)
58      /*Socket 1*/
59      CH395SetSocketRecvBuf(1,6,4); //第6,7,8,9块缓存区作为Socket的接收缓存区 (512*4=2KB)
60      CH395SetSocketSendBuf(1,10,2); //第10,11块缓存区作为Socket的发送缓存区 (512*1=1KB)
61      /*Socket 2*/
62      CH395SetSocketRecvBuf(2,12,4); //第12,13,14,15块缓存区作为Socket的接收缓存区 (512*4=2KB)
63      CH395SetSocketSendBuf(2,16,2); //第16,17块缓存区作为Socket的发送缓存区 (512*1=1KB)
64      /*Socket 3*/
65      CH395SetSocketRecvBuf(3,18,4); //第18,19,20,21块缓存区作为Socket的接收缓存区 (512*4=2KB)
66      CH395SetSocketSendBuf(3,22,2); //第22,23块缓存区作为Socket的发送缓存区 (512*1=1KB)
67
68      /*硬件版本大于4的才有Socket4-7*/
69      if(ch395_version>=0x44)
70      {
71          /*Socket 4*/
72          CH395SetSocketRecvBuf(4,24,8); //第24,25,26,27,28,29,30,31块缓存区作为Socket的接收缓存区 (512*8=4KB)
73          CH395SetSocketSendBuf(4,32,4); //第32,33,34,35块缓存区作为Socket的发送缓存区 (512*4=2KB)
74          /*Socket 5*/
75          CH395SetSocketRecvBuf(5,36,6); //第36,37,38,39,40,41块缓存区作为Socket的接收缓存区 (512*6=3KB)
76          CH395SetSocketSendBuf(5,42,6); //第42,43,44,45,46,47块缓存区作为Socket的发送缓存区 (512*6=3KB)
77      }
78  }
79

```

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
295
296
297 //INT引脚产生低电平中断以后进去判断
298 if(Query395Interrupt())
299 {
300     /*获取中断事件*/
301     if(ch395_version>=0x44)
302     {
303         ch395_status = CH395CMDGetGlobIntStatus_ALL();
304     }
305     else
306     {
307         ch395_status = CH395CMDGetGlobIntStatus();
308     }
309
310     /* 处理PHY改变中断*/
311     if(ch395_status & GINT_STAT_PHY_CHANGE)
312     {
313         if(CH395CMDGetPHYStatus() == PHY_DISCONN)//网线断开
314         {
315             printf("\r\nPHY_DISCONN\r\n");
316         }
317         else//网线连接
318         {
319             printf("\r\nPHY_CONNECTED\r\n");
320             CH395DHCPEnable(1);//启动DHCP
321         }
322     }
323
324     /* 处理DHCP/PPPOE中断 */
325     if(ch395_status & GINT_STAT_DHCP)
326     {
327         if(CH395GetDHCPStatus() == 0)//DHCP OK
328         {
329         }
330     }
331 }
```

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
229
230 /*每隔8s初始化Socket并执行连接函数*/
231 /*芯片内部连接超时时间为5s,必须大于此时间*/
232 if(Timer2Cnt>8000)
233 {
234     Timer2Cnt = 0;
235     /*Socket0*/
236     if(!SocketStatus[0])
237     {
238         if(ch395_socket_tcp_client_init(0,SocketDesIP,SocketDesPort,ch395_socket_tcp_client_port()) == 0)
239         {
240             printf("CH395TCPConnect0 ... \r\n");
241             CH395TCPConnect(0);//连接服务器
242         }
243     }
244     /*Socket1*/
245     if(!SocketStatus[1])
246     {
247         if(ch395_socket_tcp_client_init(1,SocketDesIP,SocketDesPort,ch395_socket_tcp_client_port()) == 0)
248         {
249             printf("CH395TCPConnect1 ... \r\n");
250             CH395TCPConnect(1);//连接服务器
251         }
252     }
253     /*Socket2*/
254     if(!SocketStatus[2])
255     {
256         if(ch395_socket_tcp_client_init(2,SocketDesIP,SocketDesPort,ch395_socket_tcp_client_port()) == 0)
257         {
258             printf("CH395TCPConnect2 ... \r\n");
259             CH395TCPConnect(2);//连接服务器
260         }
261     }
262     /*Socket3*/
263     if(!SocketStatus[3])
264     {
265         if(ch395_socket_tcp_client_init(3,SocketDesIP,SocketDesPort,ch395_socket_tcp_client_port()) == 0)
266         {
267             printf("CH395TCPConnect3 ... \r\n");
268             CH395TCPConnect(3);//连接服务器
269         }
270     }
271 }
```

```

271 |
272 |     /*硬件版本大于4的才有Socket4-7*/
273 |     if(ch395_version>=0x44)
274 |     {
275 |         /*Socket4*/
276 |         if(!SocketStatus[4])
277 |         {
278 |             if(ch395_socket_tcp_client_init(4,SocketDesIP,SocketDesPort,ch395_socket_tcp_client_port()) == 0)
279 |             {
280 |                 printf("CH395TCPConnect4 ... \r\n");
281 |                 CH395TCPConnect(4);//连接服务器
282 |             }
283 |         }
284 |         /*Socket5*/
285 |         if(!SocketStatus[5])
286 |         {
287 |             if(ch395_socket_tcp_client_init(5,SocketDesIP,SocketDesPort,ch395_socket_tcp_client_port()) == 0)
288 |             {
289 |                 printf("CH395TCPConnect5 ... \r\n");
290 |                 CH395TCPConnect(5);//连接服务器
291 |             }
292 |         }
293 |     }
294 | }

```

```

timer.c  usart.c  main.c  CH395CMD.H  CH395INC.H  CH395CMD.C  timer.h
79 |
80 |
81 | /* Socket 本地端口,初始化默认端口号 */
82 | UINT16 SocketLocalPort = 1000;
83 | /*动态获取本地端口号,每次获取端口号累加*/
84 | UINT16 ch395_socket_tcp_client_port(void)
85 | {
86 |     SocketLocalPort++;
87 |     if(SocketLocalPort>65535) SocketLocalPort = 1000;
88 |     return SocketLocalPort;
89 | }
90 |
91 | /**
92 | * @brief  初始化socket
93 | * @param  sockindex  Socket索引(0,1,2,3,4,5,6,7)
94 | * @param  ipaddr  目的地址
95 | * @param  desprot  目的端口号
96 | * @param  surprot  本地端口号
97 | * @retval  0:初始化成功; others:初始化失败
98 | * @warning None
99 | * @example
100 | */
101 | char ch395_socket_tcp_client_init(UINT8 sockindex,UINT8 *ipaddr,UINT16 desprot,UINT16 surprot)
102 | {
103 |     CH395SetSocketDesIP(sockindex,ipaddr); /* 目的地址 */
104 |     CH395SetSocketProtType(sockindex,PROTO_TYPE_TCP); /* 协议类型 */
105 |     CH395SetSocketDesPort(sockindex,desprot); /* 目的端口号 */
106 |     CH395SetSocketSourPort(sockindex,surprot); /* 本地端口号 */
107 |     if(CH395OpenSocket(sockindex) !=0) /* 打开Socket */
108 |     {
109 |         return 1;
110 |     }
111 |     return 0;
112 | }

```

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

```
334     CH395CMDGetUnreachIPPT(buf);
335 }
336
337 /* 处理IP冲突中断, 建议重新修改CH395的 IP, 并初始化CH395*/
338 if(ch395_status & GINT_STAT_IP_CONFLI){
339
340 }
341 /* 处理 SOCK0 中断 */
342 if(ch395_status & GINT_STAT_SOCK0){
343     ch395_socket_tcp_client_interrupt(0);
344 }
345 /* 处理 SOCK1 中断 */
346 if(ch395_status & GINT_STAT_SOCK1){
347     ch395_socket_tcp_client_interrupt(1);
348 }
349 /* 处理 SOCK2 中断 */
350 if(ch395_status & GINT_STAT_SOCK2){
351     ch395_socket_tcp_client_interrupt(2);
352 }
353 /* 处理 SOCK3 中断 */
354 if(ch395_status & GINT_STAT_SOCK3){
355     ch395_socket_tcp_client_interrupt(3);
356 }
357
358
359 if(ch395_version>=0x44)
360 {
361     /* 处理 SOCK4 中断 */
362     if(ch395_status & GINT_STAT_SOCK4){
363         ch395_socket_tcp_client_interrupt(4);
364     }
365     /* 处理 SOCK5 中断 */
366     if(ch395_status & GINT_STAT_SOCK5){
367         ch395_socket_tcp_client_interrupt(5);
368     }
369     /* 处理 SOCK6 中断 */
370     if(ch395_status & GINT_STAT_SOCK6){
371
372     }
```

```

timer.c  usart.c  main.c  CH395CMD.H  CH395INC.H  CH395CMD.C  timer.h
116  /**
117  * @brief  socket处理函数(把此函数放到全局socket中断里面)
118  * @param  sockindex  Socket索引(0,1,2,3,4,5,6,7)
119  * @param  None
120  * @param  None
121  * @param  None
122  * @retval None
123  * @warning None
124  * @example
125  **/
126  void ch395_socket_tcp_client_interrupt(UINT8 sockindex)
127  {
128      UINT8 sock_int_socket;
129      UINT16 len;
130
131      /* 获取socket 的中断状态 */
132      sock_int_socket = CH395GetSocketInt(sockindex);
133
134      /* 发送缓冲区空闲,可以继续写入要发送的数据 */
135      if(sock_int_socket & SINT_STAT_SENBUF_FREE)
136      {
137      }
138
139      /* 发送完成中断 */
140      if(sock_int_socket & SINT_STAT_SEND_OK)
141      {
142      }
143
144      /* 接收数据中断 */
145      if(sock_int_socket & SINT_STAT_RECV)
146      {
147          len = CH395GetRecvLength(sockindex);/* 获取当前缓冲区内数据长度 */
148          printf("\r\nsocket%d receive len = %d\r\n",sockindex,len);
149          if(len == 0)return;
150          if(len > recv_buff_len)len = recv_buff_len;
151          CH395GetRecvData(sockindex,len,recv_buff);/* 读取数据 */
152
153          /*把接收的数据发送给服务器*/
154          CH395SendData(sockindex,recv_buff,len);
155
156          /*使用串口打印接收的数据*/
157          PutData(&rb_t_usart1_send,recv_buff,len);
158          USART_ITConfig(USART1, USART_IT_TXE, ENABLE);
159      }
160
161      /* 连接中断,仅在Tcp模式下有效*/
162      if(sock_int_socket & SINT_STAT_CONNECT)
163      {
164          SocketStatus[sockindex] = 1;//设置连接状态为连接
165          printf("socket%d SINT_STAT_CONNECT\r\n",sockindex);
166      }
167
168      /* 断开中断,仅在Tcp模式下有效 */
169      if(sock_int_socket & SINT_STAT_DISCONNECT)
170      {
171          printf("socket%d SINT_STAT_DISCONNECT \r\n",sockindex);
172          SocketStatus[sockindex] = 0;//设置连接状态为未连接
173      }
174
175      /* 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*/
176      if(sock_int_socket & SINT_STAT_TIM_OUT)
177      {/*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*/
178          printf("socket%d SINT_STAT_TIM_OUT\r\n",sockindex);
179          SocketStatus[sockindex] = 0;//设置连接状态为未连接
180      }
181  }

```

5.提示

6路Socket使用了同一个处理函数

```
void ch395_socket_tcp_client_interrupt(UINT8 sockindex)
```

实际应用中也可以使用一个,然后以sockindex作为区分,去做不同的处理

当然也可以多写几个处理函数

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: [6-网络芯片CH395Q学习开发-模块使用Socket0-3作为4路TCP客户端和电脑上位机TCP客户端局域网通信](#)

posted on 2021-06-12 07:28 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



支持 Markdown

自动补全

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。