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/

导航

博客园
首页
新随笔
联系
订阅 XML
管理

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)

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

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
- 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)
-
-
-
-

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(Windows),并连接测试(96476)
3. ESP8266刷AT固件与nodemcu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

```
#ifndef __CH395SPI_H_
#define __CH395SPI_H_

#include "CH395INC.H"

/*****************************配置GPIO (根据自己的修改)*********************************
//时钟
#define CH395_CONFIG_SPI_CLK()    ( RCC_APB1PeriphClockCmd( RCC_APB1Periph_SPI2,ENABLE) )
#define CH395_CONFIG_GPIO_CLK()   ( RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOA | RCC_APB2Peri
//设置使用的SPI
#define USE_SPI_SPI2
//SPI_CS  --  连接模块SCS引脚
#define CH395_CS_PORT    GPIOB
#define CH395_CS_PIN     GPIO_Pin_12
//SPI_CLK --  连接模块SCK引脚
#define CH395_CLK_PORT   GPIOB
#define CH395_CLK_PIN    GPIO_Pin_13
//SPI_MISO --  连接模块SDO引脚
#define CH395_MISO_PORT  GPIOB
#define CH395_MISO_PIN   GPIO_Pin_14
//SPI_MOSI --  连接模块SDI引脚
#define CH395_MOSI_PORT  GPIOB
#define CH395_MOSI_PIN   GPIO_Pin_15
//RST --  连接模块RST引脚
#define CH395_RST_PORT  GPIOA
#define CH395_RST_PIN   GPIO_Pin_8
//TX --  连接模块TX引脚
#define CH395_TX_PORT  GPIOA
#define CH395_TX_PIN   GPIO_Pin_3
//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)
#define CH395_INT_PORT  GPIOA
#define CH395_INT_PIN   GPIO_Pin_0
/*********************************************************************/
```

2,注意!

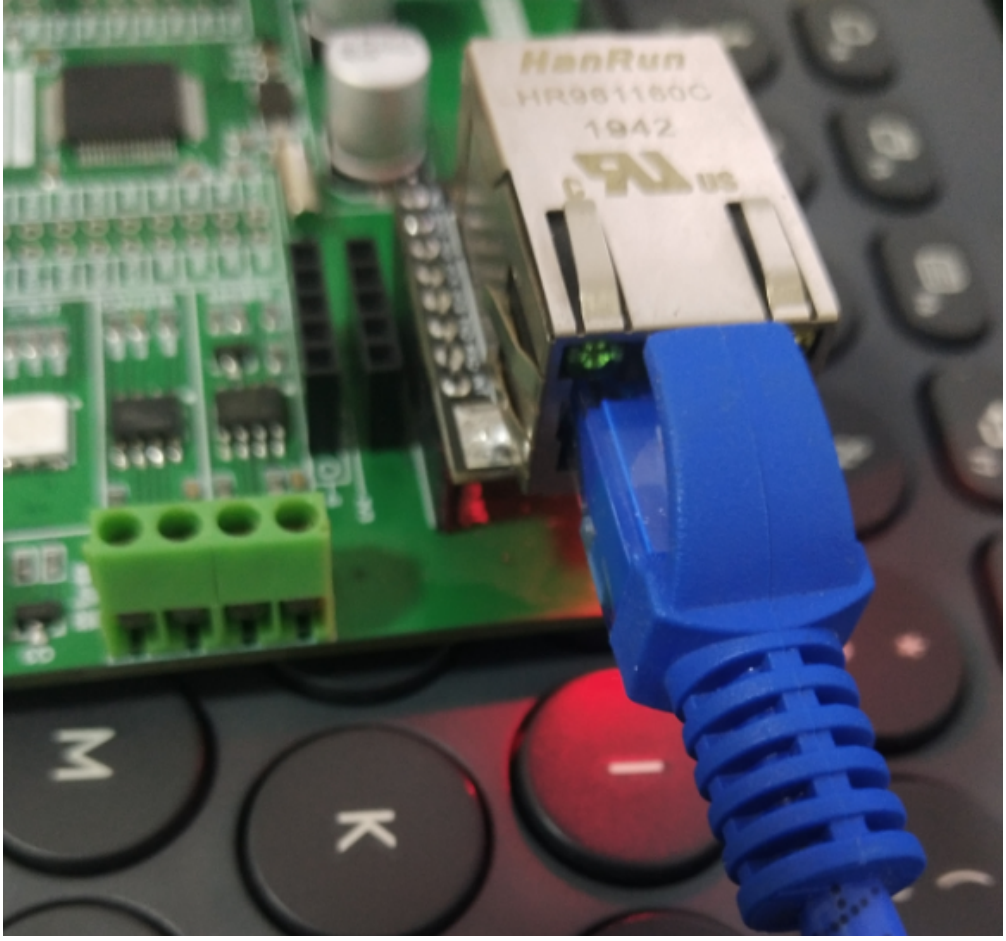
要想模块使用SPI通信,模块的TX引脚需要在模块重启之前设置为低电平.

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!

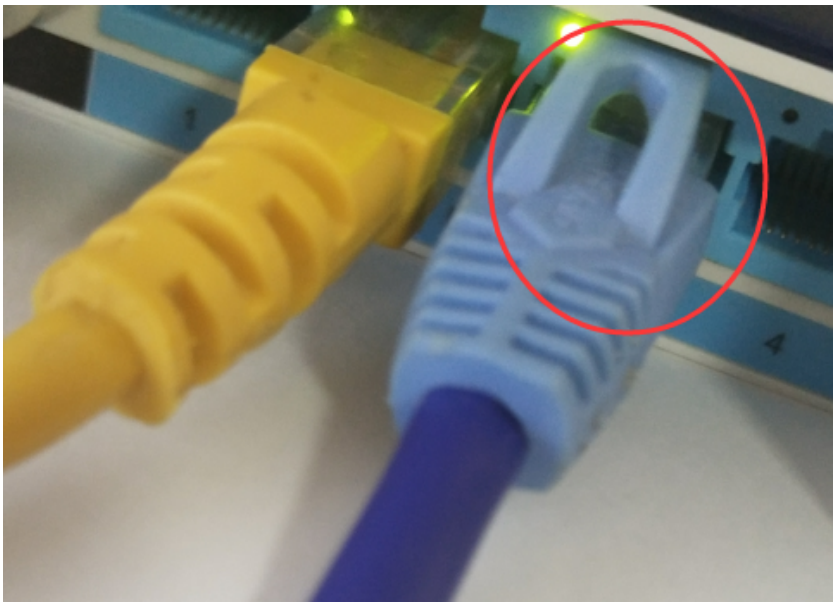
CH395 与单片机之间支持三种通讯接口：8 位并行接口、SPI 同步串行接口、异步串口。在芯片上电复位时，CH395 将采样 SEL 和 TXD 引脚的状态，根据这 2 个引脚状态的组合选择通讯接口，参考下表（表中 X 代表不关心此位，0 代表低电平，1 代表高电平或者悬空）。

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)

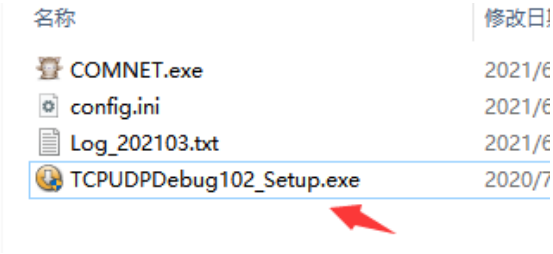


注意,连接路由器或者交换机的时候是连接其LAN口.



4,在电脑上运行网络调试助手,开启TCP服务器

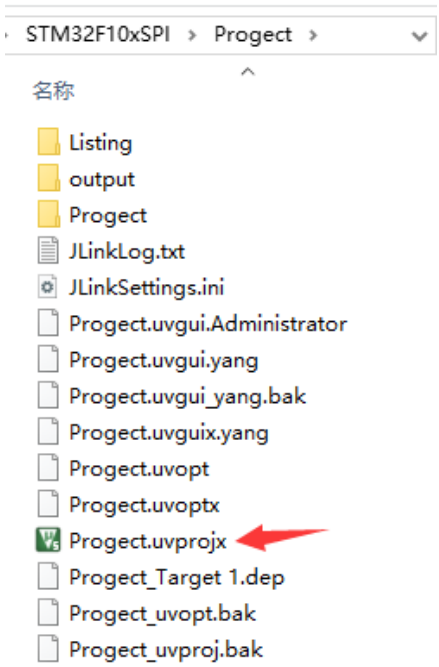
换了个调试助手,上一个有点卡顿



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

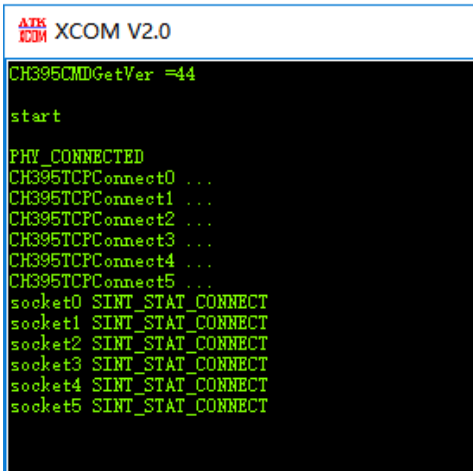


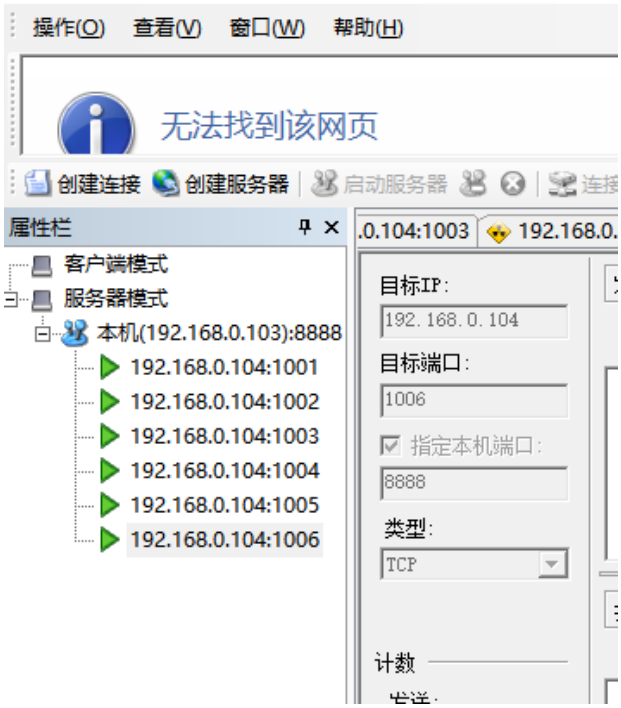
7,根据自己的修改服务器IP地址和端口号

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

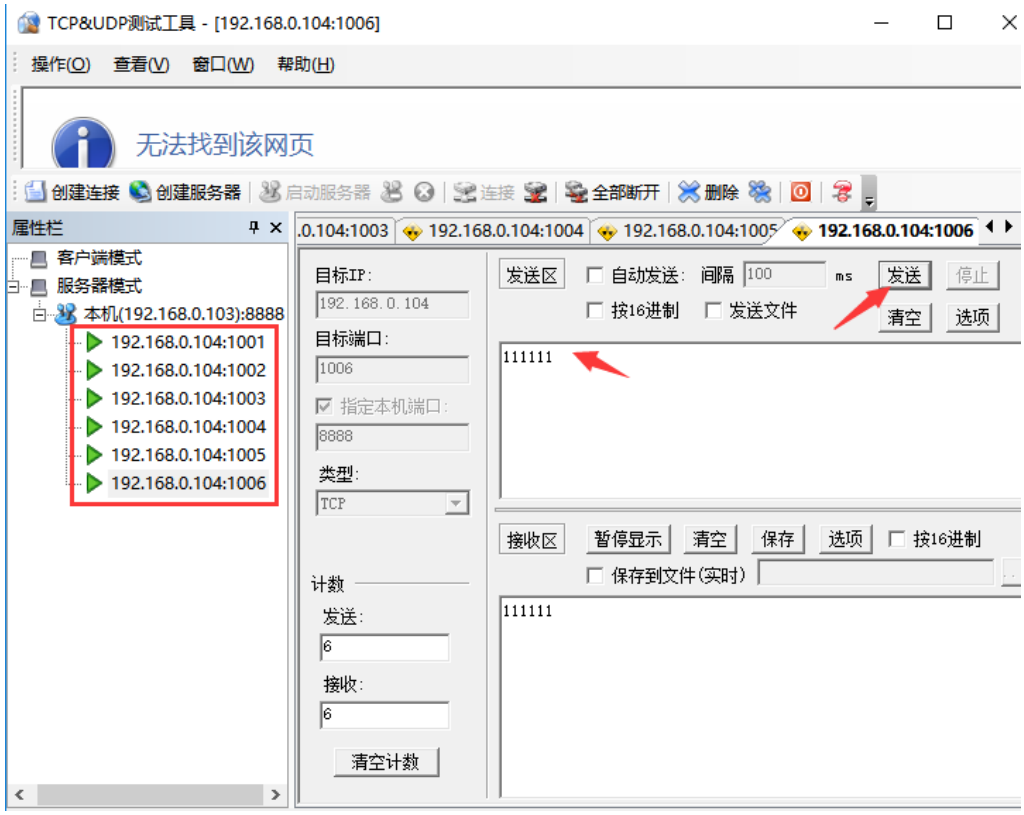
注意:版本号低于4的不会执行Socket4,5

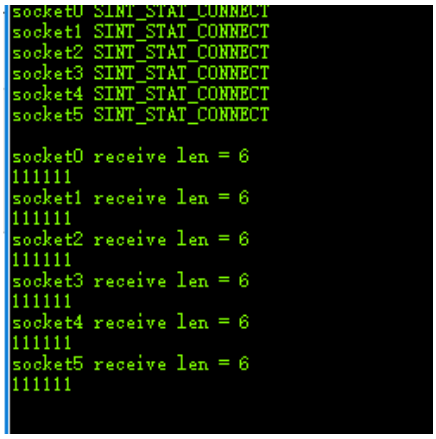




9.服务器分别给4个客户端发送消息

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



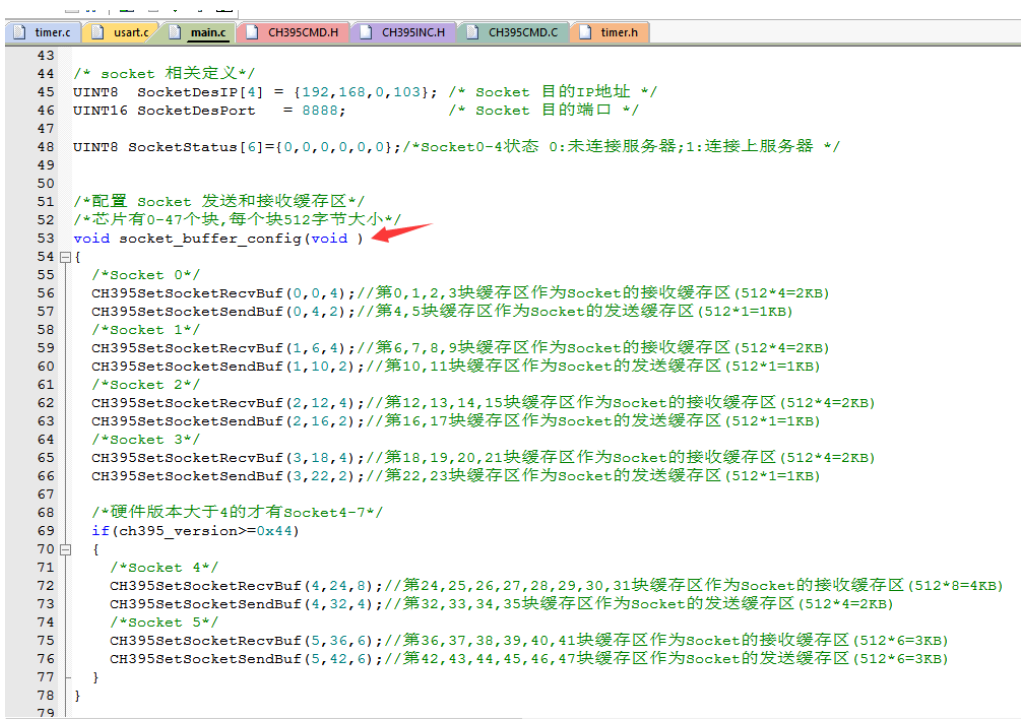


程序说明

1.设置Socket缓存区域

timer.c | usart.c | main.c | CH395CMD.H | CH395INC.H | CH395CM

```
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       }
```

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
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

```c
/**
* @brief   socket处理函数(把此函数放到全局socket中断里面)
* @param   sockindex  Socket索引(0,1,2,3,4,5,6,7)
* @param   None
* @param   None
* @param   None
* @retval  None
* @warning None
* @example
**/
void ch395_socket_tcp_client_interrupt(UINT8 sockindex)
{
  UINT8  sock_int_socket;
  UINT16 len;

  /* 获取socket 的中断状态 */
  sock_int_socket = CH395GetSocketInt(sockindex);

  /* 发送缓冲区空闲，可以继续写入要发送的数据 */
  if(sock_int_socket & SINT_STAT_SENBUF_FREE)
  {
  }

  /* 发送完成中断 */
  if(sock_int_socket & SINT_STAT_SEND_OK)
  {
  }

  /* 接收数据中断 */
  if(sock_int_socket & SINT_STAT_RECV)
  {
    len = CH395GetRecvLength(sockindex);/* 获取当前缓冲区内数据长度 */
    printf("\r\nsocket%d receive len = %d\r\n",sockindex,len);
    if(len == 0)return;
    if(len > recv_buff_len)len = recv_buff_len;
    CH395GetRecvData(sockindex,len,recv_buff);/* 读取数据 */

    /*把接收的数据发送给服务器*/
    CH395SendData(sockindex,recv_buff,len);

    /*使用串口打印接收的数据*/
    PutData(&rb_t_usart1_send,recv_buff,len);
    USART_ITConfig(USART1, USART_IT_TXE, ENABLE);
  }

  /* 连接中断，仅在TCP模式下有效*/
  if(sock_int_socket & SINT_STAT_CONNECT)
  {
    SocketStatus[sockindex] = 1;//设置连接状态为连接
    printf("socket%d SINT_STAT_CONNECT\r\n",sockindex);
  }

  /* 断开中断，仅在TCP模式下有效 */
  if(sock_int_socket & SINT_STAT_DISCONNECT)
  {
    printf("socket%d SINT_STAT_DISCONNECT \r\n",sockindex);
    SocketStatus[sockindex] = 0;//设置连接状态为未连接
  }

  /* 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*/
  if(sock_int_socket & SINT_STAT_TIM_OUT)
  {/*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*/
    printf("socket%d SINT_STAT_TIM_OUT\n",sockindex);
    SocketStatus[sockindex] = 0;//设置连接状态为未连接
  }
}
```

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

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

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...

单片机,物联网,上位机,…

扫一扫二维码，加入群聊。