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

公告

渡我不渡她 -
Not available
00:00 / 03:41

1 渡我不渡她
2 小镇姑娘
3 PDD洪荒之力

搜索

找找看
谷歌搜索

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(10)
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)

10-网络芯片CH395Q学习开发-模块使用Socket0作为UDP广播通信

<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服务器局域网通信
- 7-模块使用Socket0-5作为6路TCP客户端和电脑上位机TCP服务器局域网通信(Socket缓存区配置)
- 8-模块使用Socket0作为TCP服务器和电脑上位机TCP客户端局域网通信(单连接和多连接)
- 9-模块使用Socket0作为UDP和电脑上位机UDP局域网通信
- 10-模块使用Socket0作为UDP广播通信
-

最新评论

阅读排行榜

-
-
-

说明

这节演示一下模块使用Socket0作为UDP广播通信

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(STM32F103xxxx)

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)(62545)
5. 有人WIFI模块使用详解(38095)
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)(35378)
7. 关于TCP和MQTT之间的转换(32229)
8. android 之TCP客户端编程(31284)
9. android客服端+eps8266+单片机+路由器之远程控制系统(31134)
10. C#中public与private与static(30942)

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!

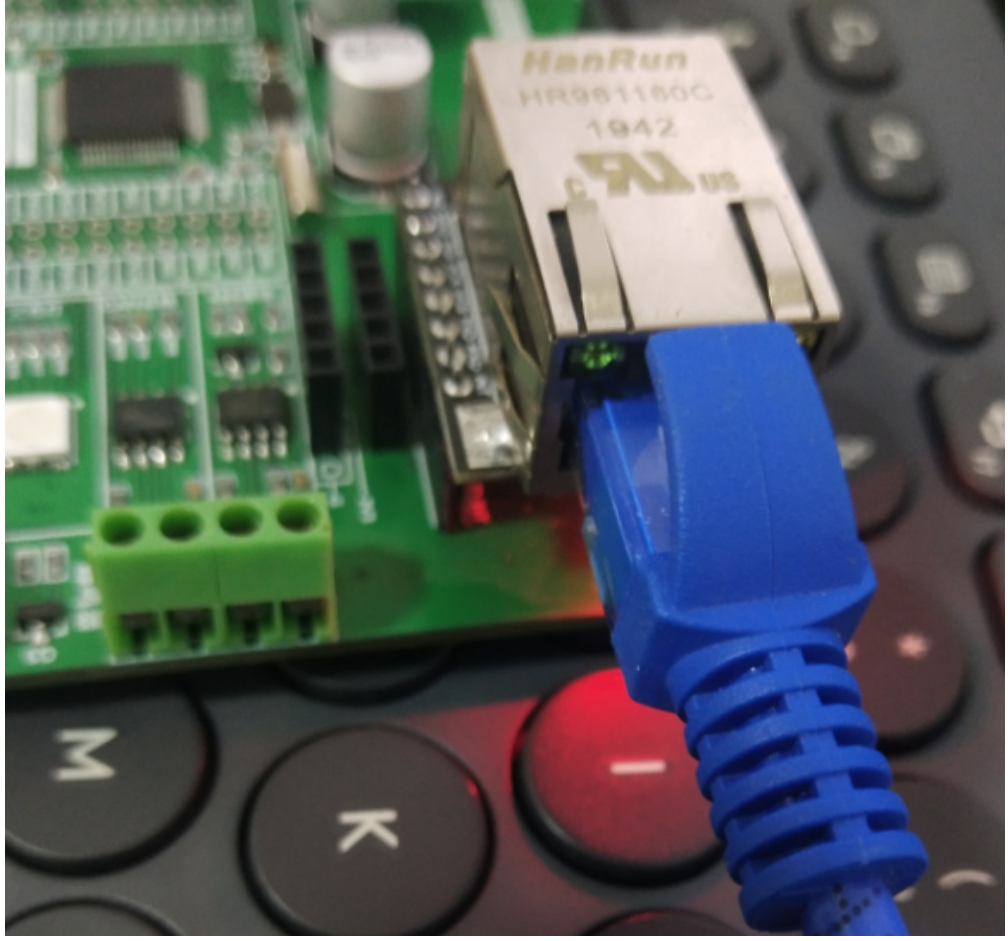
要想模块使用SPI通信,模块的TX引脚需要在模块重启之前设置为低电平.

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!

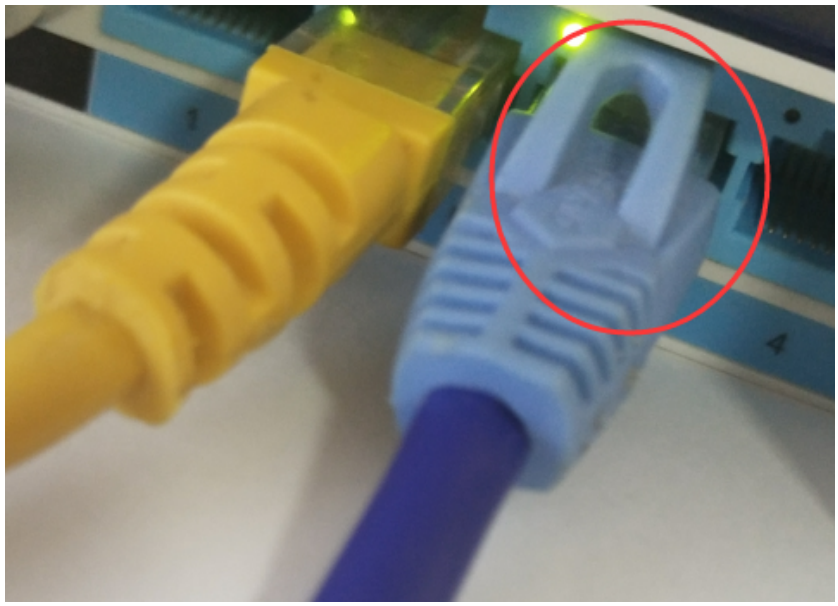
CH395 与单片机之间支持三种通讯接口：8 位并行接口、SPI 同步串行接口、异步串口。在芯片上电复位时，CH395 将采样 SEL 和 TXD 引脚的状态，根据这 2 个引脚状态的组合选择通讯接口，参考下表（表中 X 代表不关心此位，0 代表低电平，1 代表高电平或者悬空）。

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.模块往外发送数据的广播地址为255.255.255.255 端口号为6666

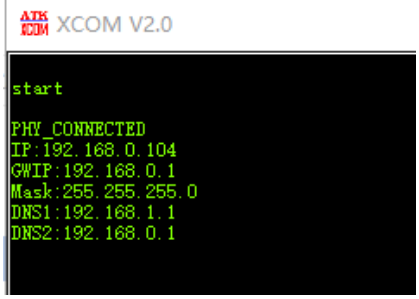
模块接收广播数据的端口号为 1000

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

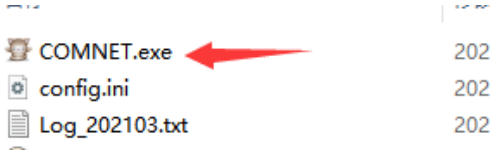
```
24  #include "CH395CMD.H"
25 /*提示:(只是提示!这节使用的Socket0通信,并没有人为分配缓存区)
26  芯片共有48块缓存区,每个缓存区512字节
27  芯片共有8个Socket,默认把48块缓存区分给了Socket0,Socket1,Socket2,Socket3
28  这四个Socket,每个 Socket 使用8块缓存区作为接收,4块缓存区作为发送,
29  即Socket0,Socket1,Socket2,Socket3的接收区各为512*8 = 4KB
30  即Socket0,Socket1,Socket2,Socket3的发送区各为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] = {255,255,255,255}; /* 广播的地址 */
47  UINT16 SocketDesPort   = 6666;             /* 广播的端口号 */
48  UINT16 SocketSourPort  = 1000;             /* Socket 本地端口 */
49
50 /**
51  * @brief   初始化socket
52  * @param   sockindex  Socket索引(0,1,2,3,4,5,6,7)
53  * @param   ipaddr  目的地址
```

5.编译下载到单片机

正常情况会打印模块的IP地址等信息



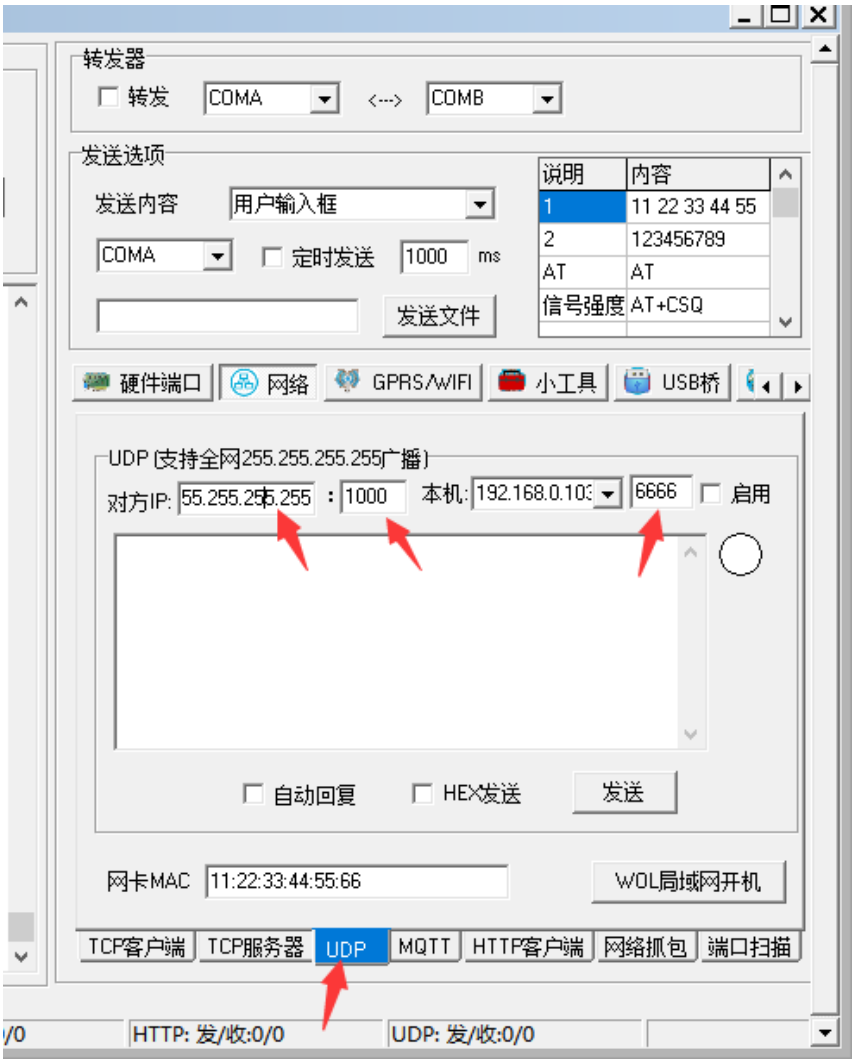
6.打开电脑端网络调试助手,并配置UDP



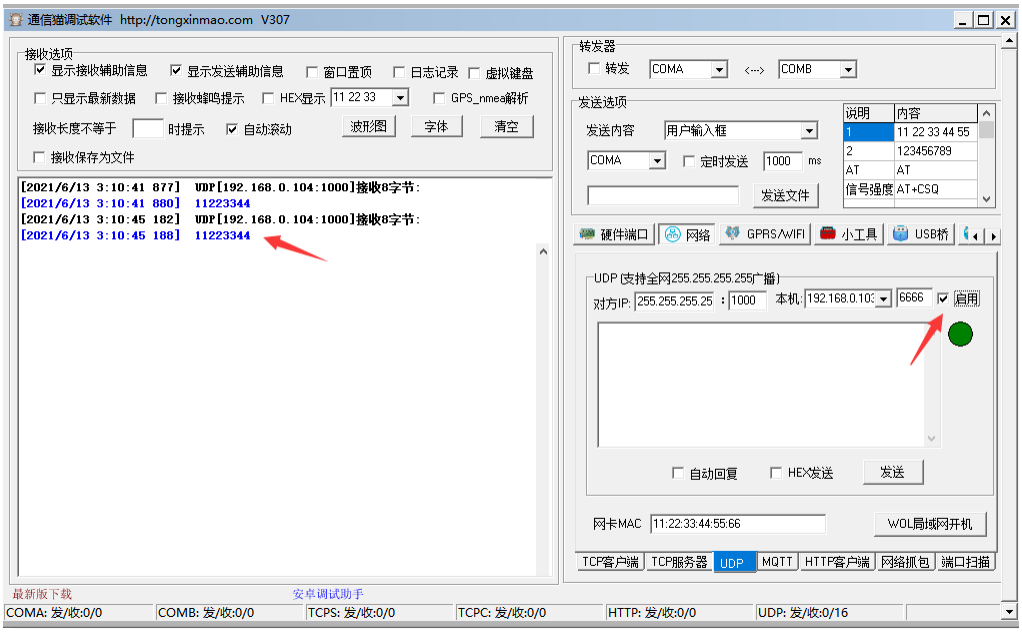
IP地址填写广播地址 255.255.255.255

对方地址填写 1000

本机地址填写6666

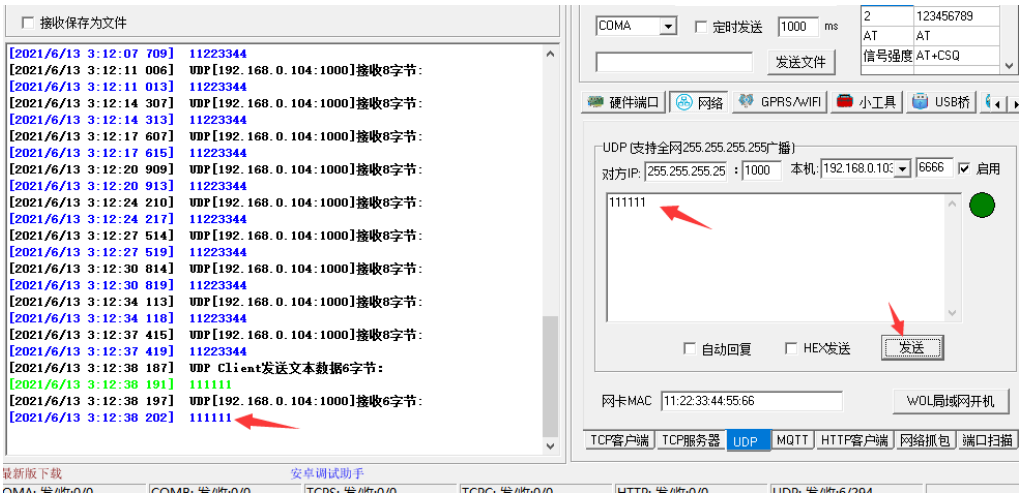


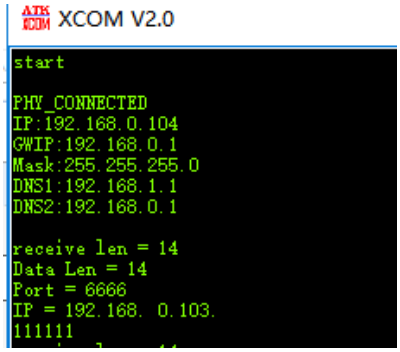
启动之后每隔3S会接收到模块广播的数据



7,网络调试助手,通过UDP广播发送数据给模块

单片机程序里面设置的接收什么数据就返回什么数据





程序说明

1.初始化UDP

UDP是面向无连接的,所以只需要配置一下.

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

```
144   IWDG_Init(IWDG_Prescaler_256,156*10);
145
146   /*获取芯片版本*/
147   while((ch395_version = CH395CMDGetVer()) < 0x40)
148   {
149     printf("CH395CMDGetVer ERR\r\n");
150     delay_ms(100);
151   }
152
153   /*测试命令，按位取反返回说明测试通过*/
154   while(CH395CMDCheckExist(0x55) != 0xaa)
155   {
156     printf("\r\nCH395CMDCheck ERR\r\n");
157     delay_ms(100);
158   }
159
160   /*初始化模块:成功返回 0 */
161   while(CH395CMDInitCH395() != 0)
162   {
163     printf("\r\nCH395CMDInitCH395 ERR\r\n");
164     delay_ms(100);
165   }
166
167   /*初始化UDP*/
168   while( ch395_socket_udp_init(SocketIndex,SocketDesIP,SocketDesPort,SocketSourPort) != 0)
169   {
170     printf("\r\nch395_socket_udp_init ERR\r\n");
171     delay_ms(100);
172   }
173
174   printf("\r\nstart\r\n");
175   while(1)
176   {
177     IWDG_Feed();//喂狗
```

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

```
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] = {255,255,255,255}; /* 广播的地址 */
47  UINT16 SocketDesPort   = 6666;             /* 广播的端口号 */
48  UINT16 SocketSourPort  = 1000;             /* Socket 本地端口 */
49
50  /**
51  * @brief   初始化socket
52  * @param   sockindex  Socket索引(0,1,2,3,4,5,6,7)
53  * @param   ipaddr  目的地址
54  * @param   desprot 目的端口号
55  * @param   surprot 本地端口号
56  * @retval  0:初始化成功; others:初始化失败
57  * @warning None
58  * @example
59  **/
60  char ch395_socket_udp_init(UINT8 sockindex,UINT8 *ipaddr,UINT16 desprot,UINT16 surprot)
61  {
62      CH395SetSocketDesIP(sockindex,ipaddr);            /* 目的地址 */
63      CH395SetSocketProtType(sockindex,PROTO_TYPE_UDP); /* 协议类型 */
64      CH395SetSocketDesPort(sockindex,desprot);         /* 目的端口号 */
65      CH395SetSocketSourPort(sockindex,surprot);        /* 本地端口号 */
66      if(CH395OpenSocket(sockindex) !=0)                /* 打开Socket */
67      {
68        return 1;
69      }
70      return 0;
71  }
72
```

2.模块连接路由器通信需要启用DHCP,并打印模块分得的地址信息

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

```
162   printf("\r\nstart\r\n");
163   while(1)
164   {
165     IWDG_Feed();//喂狗
166
167     //INT引脚产生低电平中断以后进去判断
168     if(Query395Interrupt())
169     {
170       /*获取中断事件*/
171       if(ch395_version>=0x44)
172       {
173         ch395_status = CH395CMDGetGlobIntStatus_ALL();
174       }
175       else
176       {
177         ch395_status = CH395CMDGetGlobIntStatus();
178       }
179
180       /* 处理PHY改变中断*/
181       if(ch395_status & GINT_STAT_PHY_CHANGE)
182       {
183         if(CH395CMDGetPHYStatus() == PHY_DISCONN)//网线断开
184         {
185           printf("\r\nPHY_DISCONN\r\n");
186         }
187         else//网线连接
188         {
189           printf("\r\nPHY_CONNECTED\r\n");
190           CH395DHCPEnable(1);//启动DHCP
191         }
192       }
193
194       /* 处理DHCP/PPPOE中断 */
195       if(ch395_status & GINT_STAT_DHCP)
196       {
197         if(CH395GetDHCPStatus() == 0)//DHCP OK
198         {
199           CH395GetIPInf(buf);//获取IP,网关和子网掩码
200           printf("IP:%d.%d.%d.%d\r\n",buf[0],buf[1],buf[2],buf[3]);
201           printf("GWIP:%d.%d.%d.%d\r\n",buf[4],buf[5],buf[6],buf[7]);
202           printf("Mask:%d.%d.%d.%d\r\n",buf[8],buf[9],buf[10],buf[11]);
203           printf("DNS1:%d.%d.%d.%d\r\n",buf[12],buf[13],buf[14],buf[15]);
204           printf("DNS2:%d.%d.%d.%d\r\n",buf[16],buf[17],buf[18],buf[19]);
205         }
206       }
207
```

3.每隔一段时间发送一条广播数据出去

```
165   }
166
167   /*初始化UDP*/
168   while( ch395_socket_udp_init(SocketIndex,SocketDesIP,SocketDesPort,SocketSc
169   {
170     printf("\r\nch395_socket_udp_init ERR\r\n");
171     delay_ms(100);
172   }
173
174   printf("\r\nstart\r\n");
175   while(1)
176   {
177     IWDG_Feed();//喂狗
178
179     /*每隔一段时间发送广播数据*/
180     if(Timer2Cnt>3000)
181     {
182       Timer2Cnt=0;
183 //      CH395SendData(SocketIndex,"11223344",8);
184       CH395UDPSendTo("11223344",8,SocketDesIP, SocketDesPort, SocketIndex);
185     }
186
187     //INT引脚产生低电平中断以后进去判断
188     if(Query395Interrupt())
189     {
190       /*获取中断事件*/
191       if(ch395_version>=0x44)
192       {
```

4.在中断检测事件里面处理Socket相关事件(本例中使用的Socket 0)

```
201           printf("GWIP:%d.%d.%d.%d\r\n",buf[4],buf[5],buf[6],buf[7]);
202           printf("Mask:%d.%d.%d.%d\r\n",buf[8],buf[9],buf[10],buf[11]);
203           printf("DNS1:%d.%d.%d.%d\r\n",buf[12],buf[13],buf[14],buf[15]);
204           printf("DNS2:%d.%d.%d.%d\r\n",buf[16],buf[17],buf[18],buf[19]);
205         }
206       }
207
208       /* 处理不可达中断，读取不可达信息 */
209       if(ch395_status & GINT_STAT_UNREACH){
210         CH395CMDGetUnreachIPPT(buf);
211       }
212
213       /* 处理IP冲突中断，建议重新修改CH395的 IP，并初始化CH395*/
214       if(ch395_status & GINT_STAT_IP_CONFLI){
215
216       }
217       /* 处理 SOCK0 中断 */
218       if(ch395_status & GINT_STAT_SOCK0){
219         ch395_socket_udp_interrupt(SocketIndex);
220       }
221       /* 处理 SOCK1 中断 */
222       if(ch395_status & GINT_STAT_SOCK1){
223
224       }
225       /* 处理 SOCK2 中断 */
226       if(ch395_status & GINT_STAT_SOCK2){
227
228       }
229       /* 处理 SOCK3 中断 */
```

```c
* @example
**/
void ch395_socket_udp_interrupt(UINT8 sockindex)
{
  UINT8  sock_int_socket;
  UINT16 len;
  UINT16 tmp,port;

  /* 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\nreceive len = %d\r\n",len);
    if(len == 0)return;
    if(len > recv_buff_len)len = recv_buff_len;
    CH395GetRecvData(sockindex,len,recv_buff);/* 读取数据 */

    /*打印Socket信息*/
    //0,1存储接收的数据个数
    tmp = ((UINT16)recv_buff[1] << 8) + recv_buff[0];
    //2,3存储Socket端口号
    port = ((UINT16)recv_buff[3] << 8) + recv_buff[2];
    printf("Data Len = %d\r\n",tmp);
    printf("Port = %d\r\n",port);
    //4,5,6,7存储Socket IP地址
    printf("IP = %2d.%2d.%2d.%2d\r\n",(UINT16)recv_buff[4],(UINT16)recv_buff[5],(UINT16)recv_buff[6],(UINT16)recv_buff[7]);

    //从8开始存储真实数据
    CH395UDPSendTo(&recv_buff[8], (len -8),&recv_buff[4], port, sockindex);//返回数据

    /*使用串口打印接收的数据*/
    PutData(&rb_t_usart1_send,&recv_buff[8],len);
    USART_ITConfig(USART1, USART_IT_TXE, ENABLE);
  }
}
```

注意:
对方UDP的信息存储的位置

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： 9-网络芯片CH395Q学习开发-模块使用Socket0作为UDP和电脑上位机UDP局域网通信

posted on 2021-06-13 03:21 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

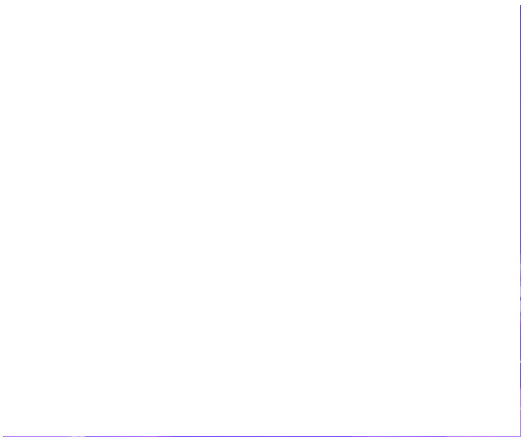
编辑 预览

支持 Markdown

自动补全

提交评论 退出

[Ctrl+Enter快捷键提交]



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：
· 滴滴的中长期动力源
· 奈雪的茶真的盈利了吗？
· “杀疯了”的剧本杀，如何敲开年轻人的门？
· BOSS直聘上市，到底谁在隐身？
· 穷人在内卷，富人在漏税
» 更多新闻...

单片机,物联网,上位机,…

扫一扫二维码，加入群聊。