W5500 模块客户端模式实验

1. 实验目的

- 了解 W5500 模块的工作原理及电气特性。
- 掌握 STC15W4K32S4 系列单片机连接 W5500 模块的硬件设计和程序设计。

2. 实验内容

- 编写程序使用查询方式控制 W5500 模块实现客户端模式下与电脑通信。
- 编写程序使用中断方式控制 W5500 模块实现客户端模式下与电脑通信。

3. 硬件设计

3.1. OSI 简介

OSI（Open System Interconnection 的缩写）是一个开放性的通信系统互连参考模型，它是一个定义得非常好的协议规范。OSI 模型有 7 层结构，如下图所示：

图 1：OSI 七层结构示意图

第一层物理层负责最后将信息编码成电流脉冲或其它信号用于网上传输。

第二层数据链路层通过物理网络链路提供可靠的数据传输。

第三层网络层负责在源和终点之间建立连接。

第四层传输层向高层提供可靠的端到端的网络数据流服务。

第五层会话层建立、管理和终止表示层与实体之间的通信会话。

第六层表示层提供多种功能用于应用层数据编码和转化，以确保以一个系统应用层发送的信息可以被另一个系统应用层识别。

第七层应用层是最接近终端用户的 OSI 层，这就意味着 OSI 应用层与用户之间是通过应用软件直接相互作用的。

✧ 注：OSI 七层网络的第一层到第三层主要面向通过网络的端到端的数据流，而第四层到第七层定义了应用程序的功能。

3.2. W5500 芯片介绍

WIZnet 是一家知名半导体公司，总部位于韩国首尔市，于 1998 年创立。目前在全球设立了三个分公司，分别在美国加利福尼亚，德国法兰克福，中国香港。WIZnet 专注于全硬件 TCP/IP 以太网芯片的研发，立足于全球市场，为单片机等嵌入式网络终端设备提供更加简洁、高效、安全、稳定的以太网接入方案。

W5500 芯片是一款全硬件 TCP/IP 嵌入式以太网控制器，为嵌入式系统提供了更加简易的互联网连接方案。W5500 芯片集成了 TCP/IP 协议栈，10/100M 以太网数据链路层（MAC）及物理层（PHY），使得用户使用单芯片就能够在他们的应用中拓展网络连接。

久经市场考验的 WIZnet 全硬件 TCP/IP 协议栈支持 TCP，UDP，IPv4，ICMP，ARP，IGMP 以及 PPPoE 协议。W5500 芯片内嵌 32K 字节片上缓存以供以太网包处理。如果你使用 W5500 芯片，你只需要一些简单的 Socket 编程就能实现以太网应用。这将会比其他嵌入式以太网方案更加快捷、简便。用户可以同时使用 8 个硬件 Socket 独立通信。

W5500 芯片提供了 SPI（外设串行接口）从而能够更加容易与外设 MCU 整合。而且，W5500 芯片的使用了新的高效 SPI 协议支持 80MHz 速率，从而能够更好的实现高速网络通讯。为了减少系统能耗，W5500 芯片提供了网络唤醒模式（WOL）及掉电模式供客户选择使用。

鉴于 W5500 芯片高性能的特点，其在下面领域有比较广泛的应用：

- 家庭网络设备: 机顶盒、个人录像机、数码媒体适配器。
- 串行转以太网: 门禁控制、 LED 显示屏、无线 AP 继电器等。
- 并行转以太网: POS/微型打印机、复印机。
- USB 转以太网: 存储设备、网络打印机。
- GPIO 转以太网: 家庭网络传感器。
- 安全系统: 数字录像机、网络摄像机、信息亭。
- 工厂和楼宇自动化控制系统。
- 医疗监测设备。
- 嵌入式服务器。

W5500 芯片内部有 32K 字节收发缓存，工作电压是 3.3V，但 I/O 信号口可耐 5V 电压。W5500 芯片的封装是 LQFP48（7x7mm，间距 0.5mm）。下面给出 W5500 方框图：

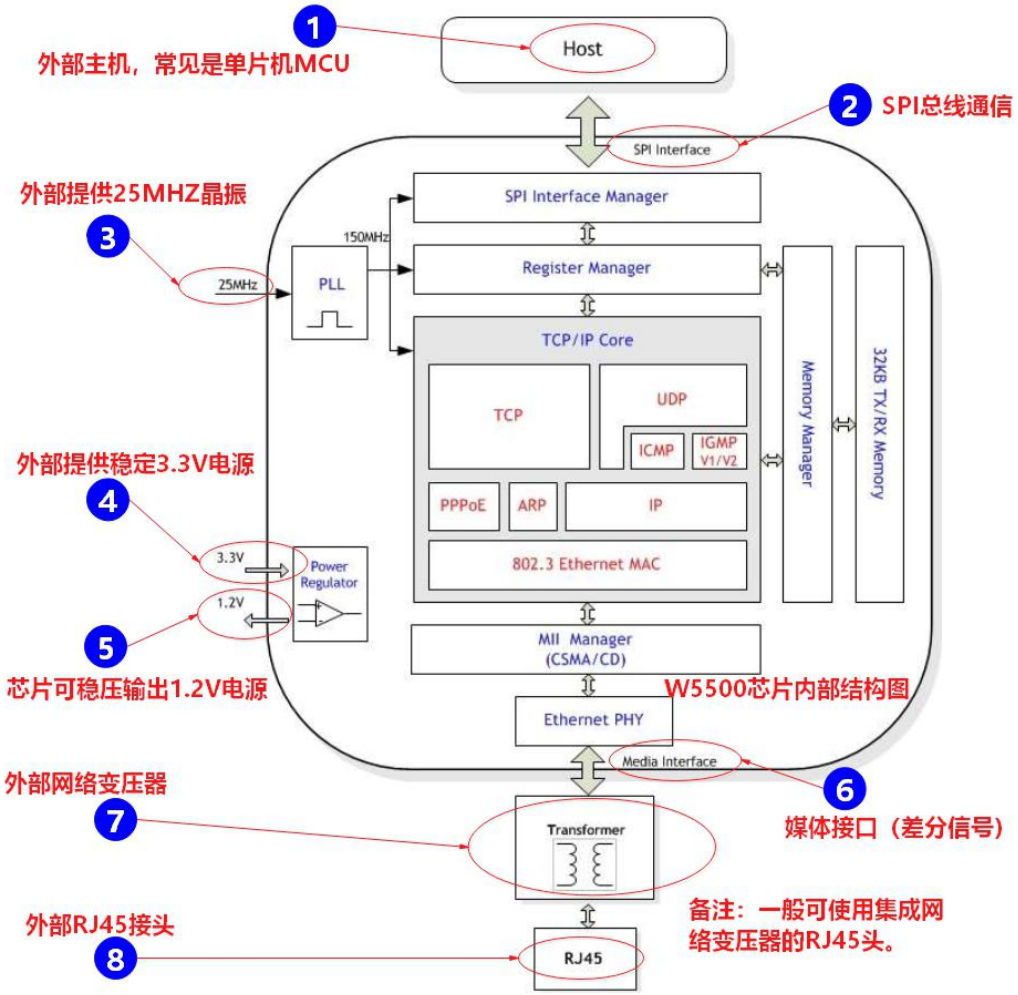

图 2：W5500 方框图

✧ 注：W5500 工作需外部 MCU 通过 SPI 总线控制 W5500 相关寄存器方可，W5500 本身的缓存仅用于收发缓存用，不会保存配置寄存器的信息。

3.3. 艾克姆科技 W5500 模块介绍

通过上节对 W5500 芯片的介绍，我们知悉该芯片功能强大，用户使用方便，但在实际使用时还需要给 W5500 芯片搭建必要的外围电路。故艾克姆科技资深工程师经深入研究和长期实践开发出一款既适合用户学习，又适合用户批量用于项目的 W5500 以太网模块。该模块外围晶振选择高精度贴片无源晶振，RJ45 选择内置网络变压器的 HR911105A，模块上的电阻 R9，R10，R11 用户可根据需要选择焊接上拉还是下拉（出厂默认均焊接上拉），以方便配置网络工作模式。

下面给出的是艾克姆 W5500 模块排针向上焊接的实物图。

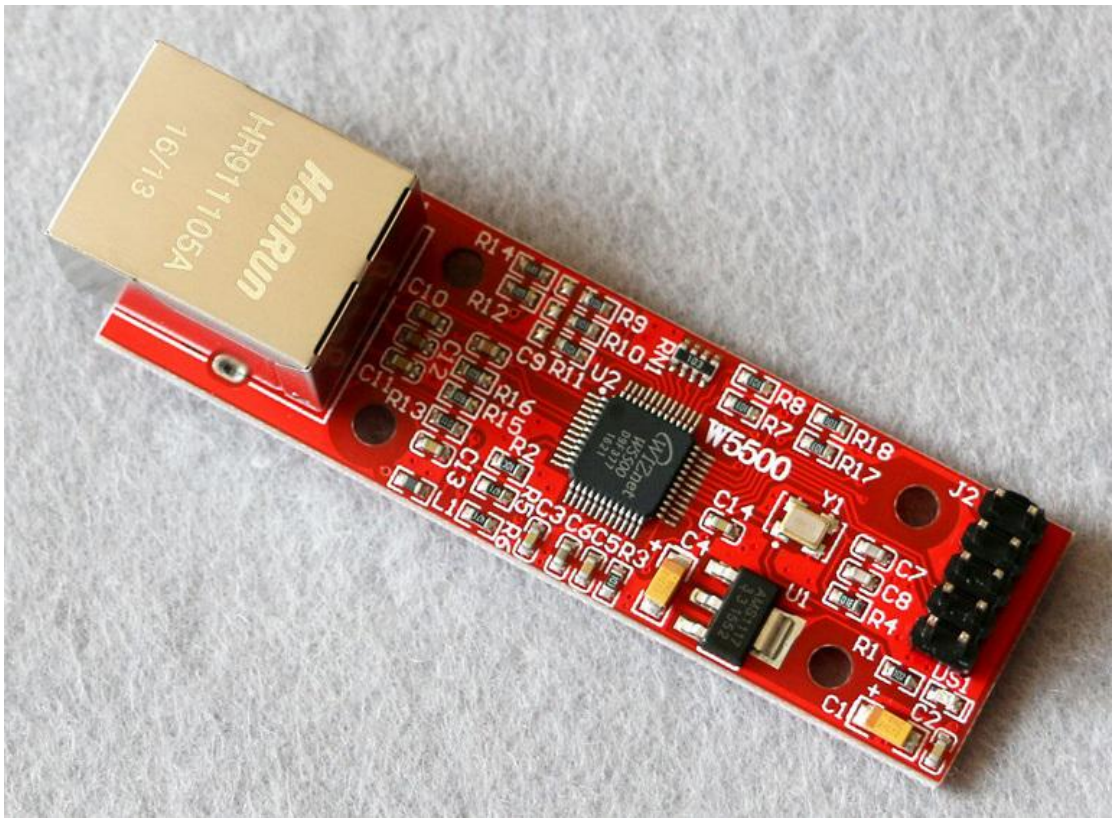

图 3：艾克姆科技 W5500 模块实物图

✧ 注：艾克姆科技 W5500 模块的排针可选择焊接排针向上和排针向下，一般艾克姆开发板如果预留有 W5500 接口，则建议用户购买选择背面直针以方便接插。

3.3.1. 艾克姆科技 W5500 以太网模块的规格参数

表 1：艾克姆 W5500 模块规格参数

参数	规格
工作电压	3.3V 或 5V
外形尺寸	71.8(L)mm × 20.4(W)mm
PCB 尺寸	67.6(L)mm ×20.4(W)mm
通信接口	高速串行外设接口（SPI 模式 0，3）
接口芯片	W5500
工作温度	-40~70 度
重量	11.3g
引脚数	10 个

3.3.2. 艾克姆科技 W5500 以太网模块的引脚定义

表 2：艾克姆 W5500 模块引脚定义

序号	引脚名	功能描述
1	3V3	3.3V 电源正
2	5V	5V 电源正
3	MISO	SPI 接口主机输入从机（W5500）输出
4	GND	模块供电地
5	MOSI	SPI 接口主机输出从机（W5500）输入
6	RST	W5500 复位管脚（低电平有效）
7	SCS	SPI 接口片选管脚
8	INT	W5500 中断输出（低电平有效）
9	SCLK	SPI 接口时钟输入
10	NC	未使用，悬空

✧ 注：艾克姆 W5500 模块供电可选择 3.3V 供电或 5V 供电，切记仅可选择其一供电。

3.4. W5500 模块和开发板连接

艾克姆 W5500 模块可以使用各种类型的 MCU 驱动，如 STM32 系列单片机、STC15 系列单片机、LPC 系列单片机以及各种蓝牙 BLE 芯片等。该 W5500 以太网模块和艾克姆科技进取者 STC15 开发板之间的连接如下图所示。

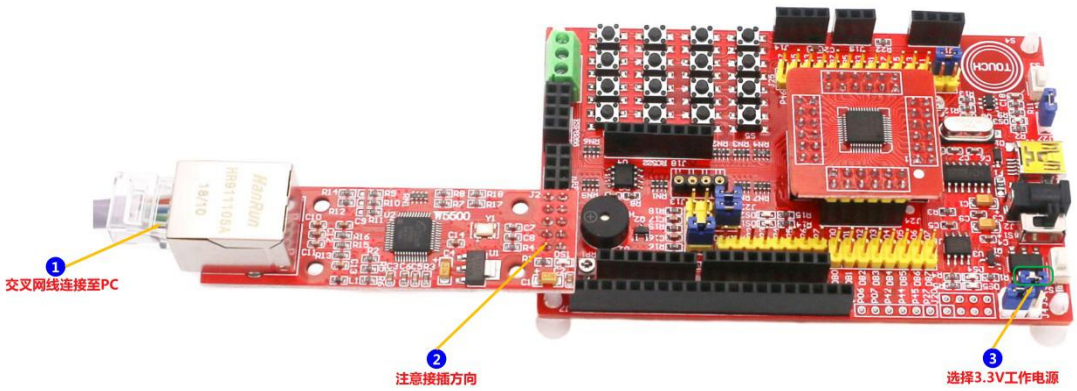

图 4：进取者 STC15 开发板与 W5500 模块连接示意图

✧ 注：W5500 模块网口与电脑连接的网线请选择交叉网线（即一端水晶头为 T568A 线序，另一端水晶头为 T568B 线序）。

1 个 W5500 以太网模块接口占用的单片机的引脚如下表：

表 3：W5500 模块接口引脚分配

序号	W5500 模块	MCU 引脚	说明
1	MISO	P41	独立 GPIO
2	MOSI	P40	独立 GPIO
3	SCLK	P43	独立 GPIO
4	SCS	P54	非独立 GPIO
5	INT	P32	独立 GPIO
6	RST	P42	独立 GPIO

✧ 注：独立 GPIO 表示开发板没有其他的电路使用这个 GPIO，非独立 GPIO 说明开发板有其他电路用到了该 GPIO。

4. 软件设计

4.1. TCP&UDP 测试工具的安装（示例电脑是 WIN7　64 位系统）

以太网实验需要一些辅助测试工具以方便进行以太网有关功能测试，这些软件安装步骤通常比较简单，界面设计仅是一些常用的选项设置，通用性比较强。下面举例周立功 TCP&UDP 测试工具的安装步骤。（仅供学习使用）

- 周立功 TCP&UDP 测试工具：解压位于“... \附 3：常用开发辅助软件\ 3 – 网络调试软件”目录下的压缩文件 TCPUDPDebug102_Setup。

1. 双击 TCPUDPDebug102_Setup.exe，弹出 TCP&UDP 测试工具的安装向导，单击【下一步】。

图 5：打开 TCPUDP 测试工具安装文件

2. 设置 TCP&UDP 测试工具的安装路径，点击【安装】开始安装，如下图所示。

此处，可以根据自己的需要选择 TCP&UDP 测试工具的安装路径，本文档设置的安装路径是默认安装路径，即安装在 C 盘。

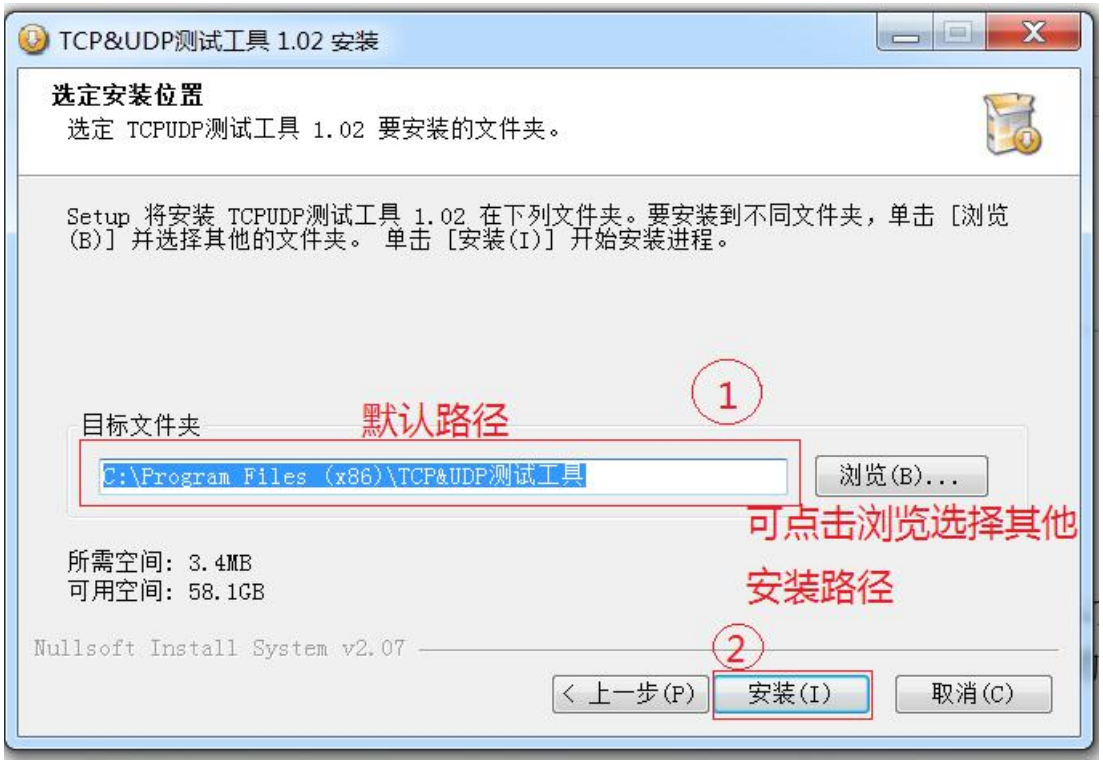

图 6：TCP&UDP 测试工具的安装路径

3. 点击【完成】以完成安装，如下图所示。

可根据需要选择是否勾选图中运行 TCPUDP 测试工具选项。

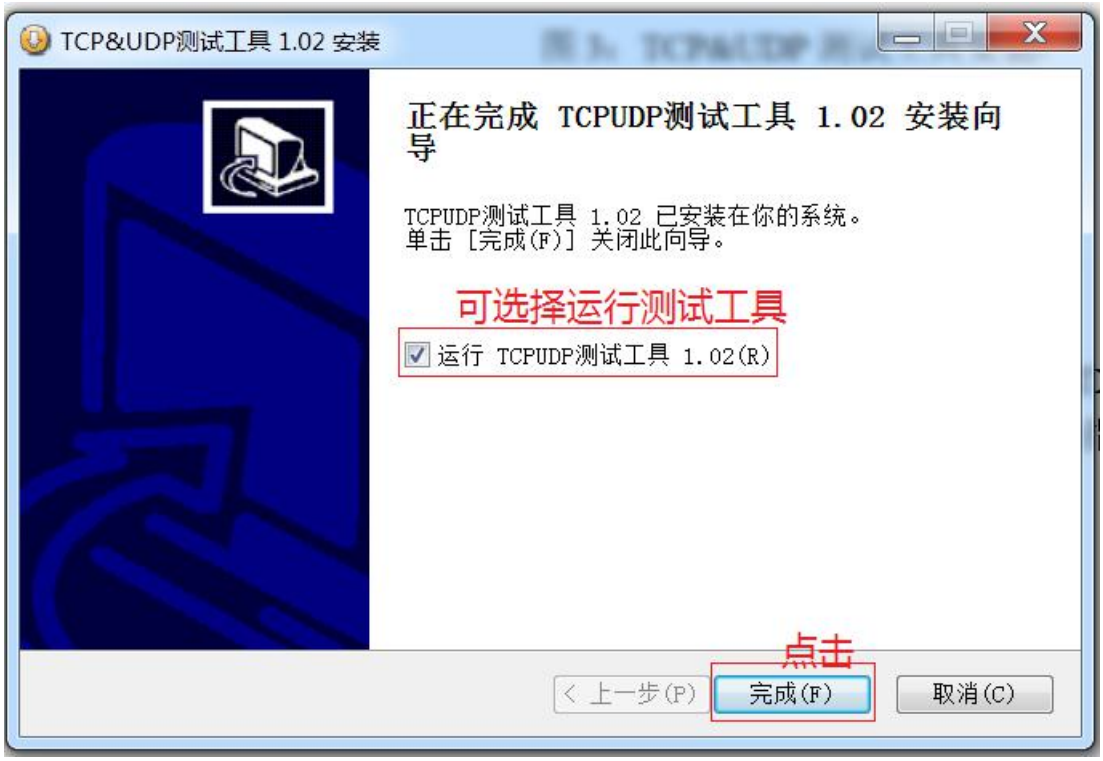

图 7：TCP&UDP 测试工具完成安装

4.2. 电脑网络参数配置（示例电脑是 WIN7　64 位系统）

因为 W5500 以太网实验例程中有默认连接的电脑 IP 等网络参数配置信息，如果电脑网络参数配置的不对，那就无法与板载有 W5500 模块的开发板正常通信。下面介绍下台式机电脑的网络配置过程。

✧ 注：如果使用的是笔记本或带有无线的一体机，则需要首先把无线网络连接禁止，因为 W5500 模块是有线连接到计算机本地连接的网口。

1. 打开“控制面板”→“网络和 Internet” →“网络和共享中心”，出现如下图界面，点击“更改适配器设置”。

图 8：打开更改适配器设置选项

2. 在出现的界面中，右击本地连接，打开属性，如下图所示。

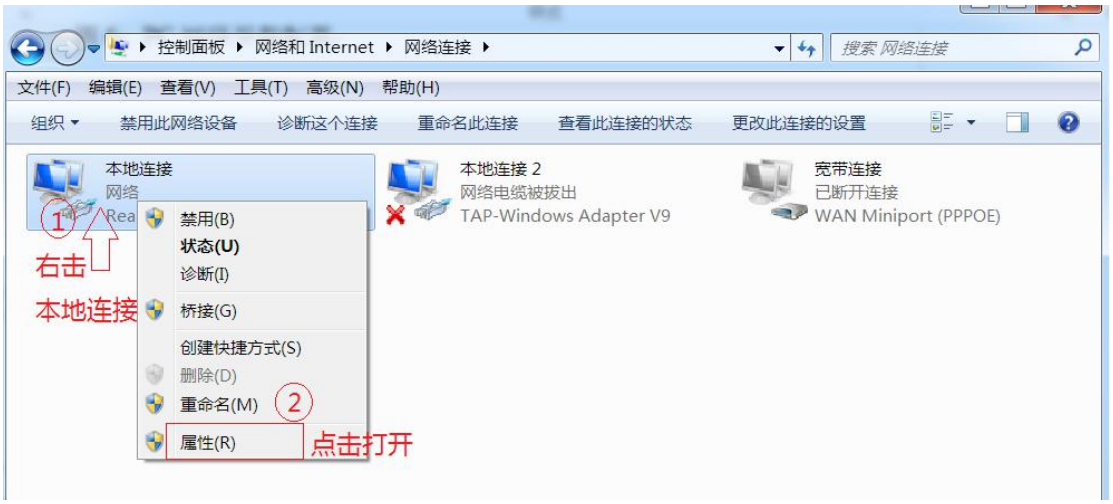

图 9：设置本地连接属性

3. 双击打开“Internet 协议版本 4（TCP/IPv4）”，如下图所示。

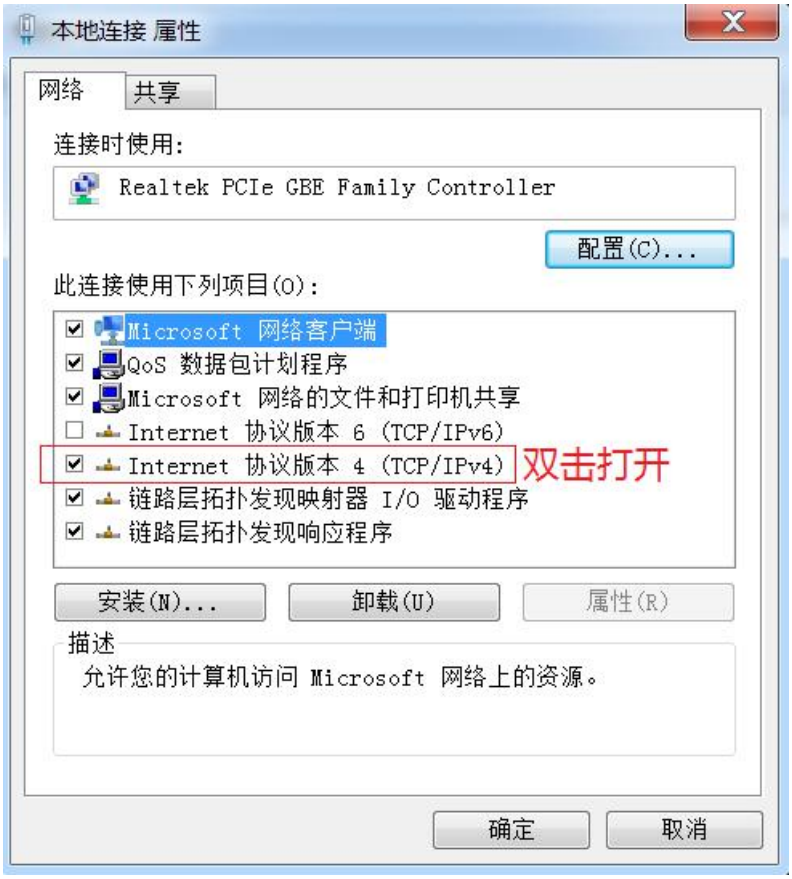

图 10：打开 TCP/IPv4 选项

4. 在打开的属性界面中，进行如下图所示配置。

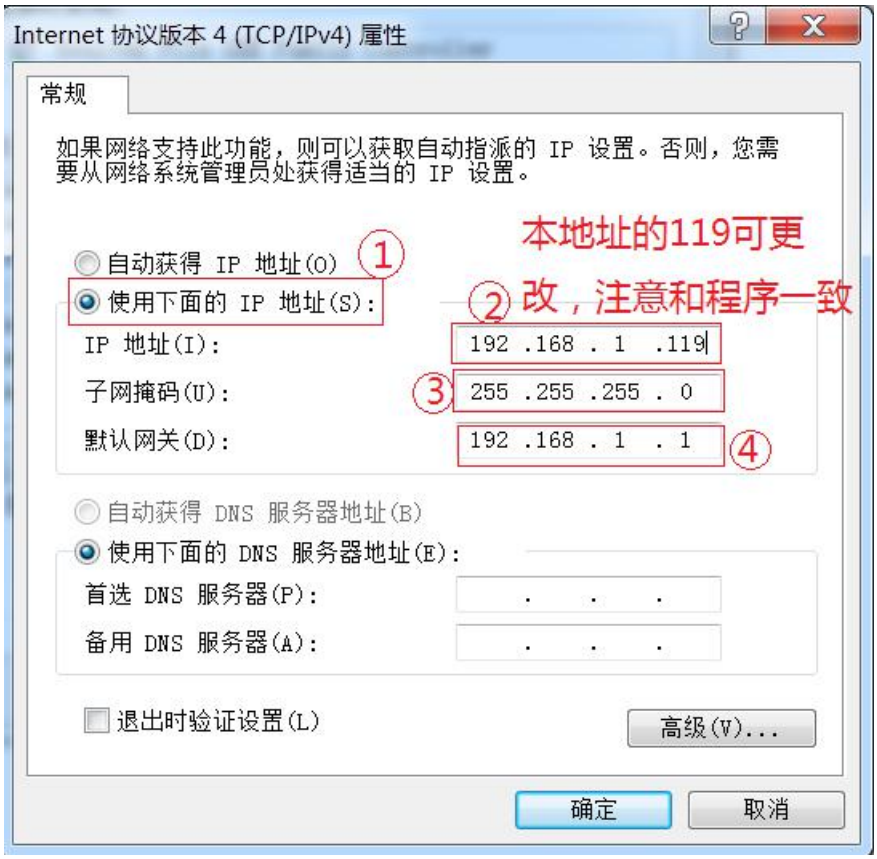

图 11：设置 TCP/IPv4 属性

✧ 注：以上网络参数的配置是根据例程匹配进行的设置，用户可同步修改下载到 MCU 的程序和电脑的有关网络配置参数进行实验。但需注意以下几点：

1、 网关 IP 地址必须与 IP 地址属于同一个子网，否则本机将无法找到网关。

2、 物理地址（ MAC）要是唯一的标识网络设备的物理地址值，不能与远程主机 （电脑）的物理相同。

3、 W5500 端口的端口号不能与远程服务器主机（电脑）的端口号相同。

4、 W5500 端口的 IP 地址与远程服务器主机（电脑）的 IP 地址不能相同，否则将会产生 IP 地址冲突。

4.3. W5500 模块客户端模式实验（查询方式）

✧ 注：本节的实验源码是在“实验 2-15-3：外接 FLASH 存储器读写单字节实验（硬件 SPI）”的基础上修改。本节对应的实验源码是：“实验 3-30-1：W5500 模块客户端模式例程（查询方式）”。

4.3.1. 工程需要用到的 c 文件

本例需要用到的 c 文件如下表所示，工程需要添加下表中的 c 文件。

表 4：实验需要用到的 c 文件

序号	文件名	后缀	功能描述
1	w5500	.c	W5500 模块有关的用户自定义函数。
2	timer	.c	定时器有关的自定义函数。
3	delay	.c	包含用户自定义延时函数。

4.3.2. 头文件引用和路径设置

■ 需要引用的头文件

```
1. #include "delay.h"
2. #include "timer.h"
3. #include "w5500.h"
```

■ 需要包含的头文件路径

本例需要包含的头文件路径如下表：

表 5：头文件包含路径

序号	路径	描述
1	..\ Source	timer.h、w5500.h 和 delay.h 头文件在该路径，所以要包含。
2	..\User	15W4KxxS4.h 头文件在该路径，所以要包含。

MDK 中点击魔术棒，打开工程配置窗口，按照下图所示添加头文件包含路径。

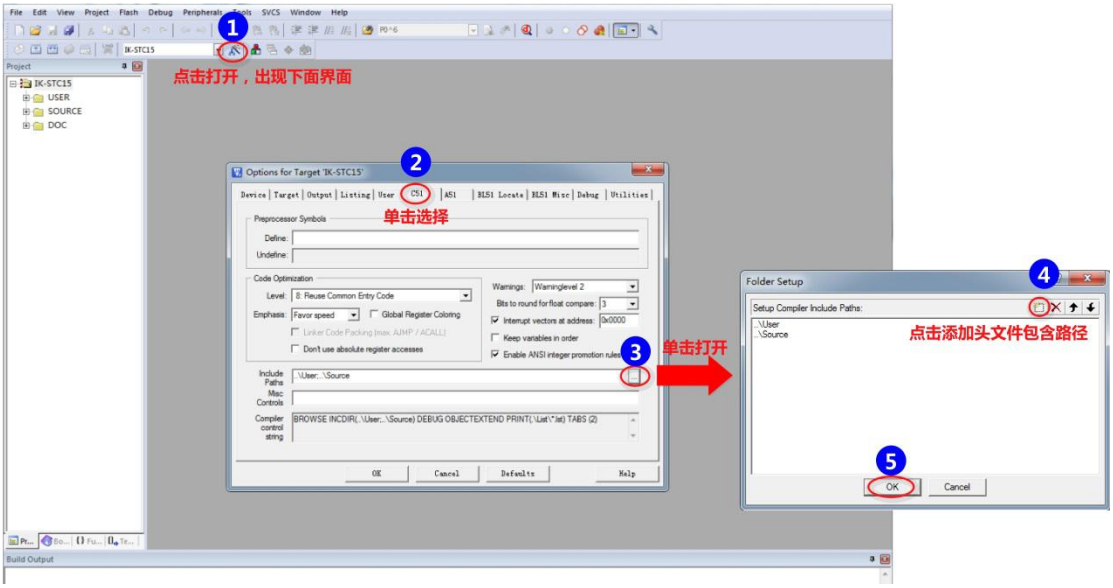

图 12：添加头文件包含路径

4.3.3. 编写代码

首先，在 w5500.c 文件中对硬件 SPI 进行初始化，并编写硬件 SPI 的读写字节函数，代码如下：

程序清单：硬件 SPI 初始化函数

```
/*******************************************************************************
 * 描 述 : 硬件 SPI 初始化
 * 入 参 : 无
 * 返回值 : 无
 *******************************************************************************/
void Init_SPI(void)
{
    AUXR1|=0X08;            //将 SPI 调整到 P4.1 P4.2 P4.3
    AUXR1&=0XFB;            //将 SPI 调整到 P4.1 P4.2 P4.3
    SPDAT = 0; //初始化 SPI 数据
    SPSTAT = SPIF | WCOL; //清除 SPI 状态位
    SPCTL = SPEN | MSTR | SSIG; //主机模式
}
```

程序清单：硬件 SPI 读写字节函数

```
/*******************************************************************************
 * 描 述 : 硬件 SPI 写入一个字节，并返回一个值
 * 入 参 : unsigned char byte
 * 返回值 : 无
 *******************************************************************************/
unsigned char SPI_SendByte(unsigned char byte)
{
  SPDAT = byte; //触发 SPI 发送数据
  while (!(SPSTAT & SPIF)); //等待发送完成
  SPSTAT = SPIF | WCOL; //清除 SPI 状态位
  return SPDAT; //返回 SPI 数据
}
```

然后，编写对 W5500 以太网模块的基本操作函数，如下表所示。

表 6：W5500 相关用户函数汇集

序号	函数名	功能描述
1	SPI_Send_Short	硬件 SPI 向 W5500 写入 2 字节数据。
2	Write_W5500_1Byte	硬件 SPI 向 W5500 指定地址寄存器写 1 字节数据。
3	Write_W5500_2Byte	硬件 SPI 向 W5500 指定地址寄存器写 2 字节数据。
4	Write_W5500_nByte	硬件 SPI 向 W5500 指定地址寄存器写 n 字节数据。

5	Write_W5500_SOCK_1Byte	硬件 SPI 向 W5500 指定端口寄存器写 1 字节数据。
6	Write_W5500_SOCK_2Byte	硬件 SPI 向 W5500 指定端口寄存器写 2 字节数据。
7	Write_W5500_SOCK_4Byte	硬件 SPI 向 W5500 指定端口寄存器写 4 字节数据。
8	Read_W5500_1Byte	硬件 SPI 读 W5500 指定地址寄存器 1 字节数据。
9	Read_W5500_SOCK_1Byte	硬件 SPI 读 W5500 指定端口寄存器 1 字节数据。
10	Read_W5500_SOCK_2Byte	硬件 SPI 读 W5500 指定端口寄存器 2 字节数据。
11	Read_SOCK_Data_Buffer	硬件 SPI 读 W5500 接收数据缓冲区数据。
12	Write_SOCK_Data_Buffer	硬件 SPI 向 W5500 发送数据缓冲区写数据。
13	W5500_Hardware_Reset	硬件复位 W5500。
14	W5500_Init	初始化 W5500 寄存器。
15	Detect_Gateway	检查网关服务器。
16	Socket_Init	指定 Socket(0~7)初始化。
17	Socket_Connect	设置指定 Socket(0~7)为客户端与远程服务器连接。
18	Socket_Listen	设置指定 Socket(0~7)为服务器等待远程主机连接。
19	Socket_UDP	设置指定 Socket(0~7)为 UDP 模式。
20	W5500_Interrupt_Process	W5500 中断处理程序框架。

关于每个操作 W5500 以太网模块相关的用户函数，下面给出详细代码。

程序清单：硬件 SPI 向 W5500 写入 2 字节数据函数

```
1.  /*******************************************************************************
2.  * 函数名 : SPI_Send_Short
3.  * 描述   : SPI 发送 2 个字节数据(16 位)
4.  * 输入   : dat:待发送的 16 位数据
5.  * 输出   : 无
6.  * 返回值 : 无
7.  * 说明   : 无
8.  *******************************************************************************/
9.  void SPI_Send_Short(unsigned short dat)
10. {
11.    SPI_SendByte(dat/256);      //写数据高位
12.    SPI_SendByte(dat);          //写数据低位
13. }
```

程序清单：硬件 SPI 向 W5500 指定地址寄存器写 1 字节数据函数

```
1.  /*******************************************************************************
```

```
2.  * 函数名 : Write_W5500_1Byte
3.  * 描述   : 通过 SPI1 向指定地址寄存器写 1 个字节数据
4.  * 输入   : reg:16 位寄存器地址,dat:待写入的数据
5.  * 输出   : 无
6.  * 返回值 : 无
7.  * 说明   : 无
8.  ******************************************************************************/
9.  void Write_W5500_1Byte(unsigned short reg, unsigned char dat)
10. {
11.     W5500_SCS=0;                    //置 W5500 的 SCS 为低电平
12.
13.     SPI_Send_Short(reg);             //通过 SPI 写 16 位寄存器地址
14.     SPI_SendByte(FDM1|RWB_WRITE|COMMON_R);  //通过 SPI 写控制字节,1 个字节数据长度,写数据,
    选择通用寄存器
15.     SPI_SendByte(dat);              //写 1 个字节数据
16.
17.     W5500_SCS=1;                    //置 W5500 的 SCS 为高电平
18. }
```

程序清单：硬件 SPI 向 W5500 指定地址寄存器写 2 字节数据函数

```
1.  /******************************************************************************
2.  * 函数名 : Write_W5500_2Byte
3.  * 描述   : 通过 SPI1 向指定地址寄存器写 2 个字节数据
4.  * 输入   : reg:16 位寄存器地址,dat:16 位待写入的数据(2 个字节)
5.  * 输出   : 无
6.  * 返回值 : 无
7.  * 说明   : 无
8.  ******************************************************************************/
9.  void Write_W5500_2Byte(unsigned short reg, unsigned short dat)
10. {
11.     W5500_SCS=0;                    //置 W5500 的 SCS 为低电平
12.
13.     SPI_Send_Short(reg);             //通过 SPI 写 16 位寄存器地址
14.     SPI_SendByte(FDM2|RWB_WRITE|COMMON_R);  //通过 SPI 写控制字节,2 个字节数据长度,写数据,
    选择通用寄存器
15.     SPI_Send_Short(dat);             //写 16 位数据
16.
17.     W5500_SCS=1;                    //置 W5500 的 SCS 为高电平
18. }
```

程序清单：硬件 SPI 向 W5500 指定地址寄存器写 n 字节数据函数

```
1.  /******************************************************************************
2.  * 函数名 : Write_W5500_nByte
3.  * 描述   : 通过 SPI1 向指定地址寄存器写 n 个字节数据
4.  * 输入   : reg:16 位寄存器地址,*dat_ptr:待写入数据缓冲区指针,length:待写入的数据长度
```

```
* 输出  : 无
* 返回值 : 无
* 说明  : 无
*******************************************************************************/
void Write_W5500_nByte(unsigned short reg, unsigned char *dat_ptr, unsigned short length)
{
    unsigned short i;

    W5500_SCS=0;                          //置 W5500 的 SCS 为低电平

    SPI_Send_Short(reg);                  //通过 SPI 写 16 位寄存器地址
    SPI_SendByte(VDM|RWB_WRITE|COMMON_R);  //通过 SPI 写控制字节,N 个字节数据长度,写数据,选择通用寄存器

    for(i=0;i<length;i++)                 //循环将缓冲区的 size 个字节数据写入 W5500
    {
        SPI_SendByte(*dat_ptr++);         //写一个字节数据
    }

    W5500_SCS=1;                          //置 W5500 的 SCS 为高电平
}
```

程序清单：硬件 SPI 向 W5500 指定端口寄存器写 1 字节数据函数

```
/*******************************************************************************
* 函数名 : Write_W5500_SOCK_1Byte
* 描述  : 通过 SPI1 向指定端口寄存器写 1 个字节数据
* 输入  : s:端口号,reg:16 位寄存器地址,dat:待写入的数据
* 输出  : 无
* 返回值 : 无
* 说明  : 无
*******************************************************************************/
void Write_W5500_SOCK_1Byte(SOCKET s, unsigned short reg, unsigned char dat)
{
    W5500_SCS=0;                          //置 W5500 的 SCS 为低电平

    SPI_Send_Short(reg);                  //通过 SPI 写 16 位寄存器地址
    SPI_SendByte(FDM1|RWB_WRITE|(s*0x20+0x08));//通过 SPI 写控制字节,1 个字节数据长度,写数据,选择端口 s 的寄存器
    SPI_SendByte(dat);                    //写 1 个字节数据

    W5500_SCS=1;                          //置 W5500 的 SCS 为高电平
}
```

程序清单：硬件 SPI 向 W5500 指定端口寄存器写 2 字节数据函数

```
/*******************************************************************************
* 函数名 : Write_W5500_SOCK_2Byte
* 描述   : 通过 SPI1 向指定端口寄存器写 2 个字节数据
* 输入   : s:端口号,reg:16 位寄存器地址,dat:16 位待写入的数据(2 个字节)
* 输出   : 无
* 返回值 : 无
* 说明   : 无
*******************************************************************************/
void Write_W5500_SOCK_2Byte(SOCKET s, unsigned short reg, unsigned short dat)
{
   W5500_SCS=0;                    //置 W5500 的 SCS 为低电平

   SPI_Send_Short(reg);             //通过 SPI 写 16 位寄存器地址
   SPI_SendByte(FDM2|RWB_WRITE|(s*0x20+0x08));//通过 SPI 写控制字节,2 个字节数据长度,写数据,选择端口 s 的寄存器
   SPI_Send_Short(dat);             //写 16 位数据

   W5500_SCS=1;                    //置 W5500 的 SCS 为高电平
}
```

程序清单：硬件 SPI 向 W5500 指定端口寄存器写 4 字节数据函数

```
/*******************************************************************************
* 函数名 : Write_W5500_SOCK_4Byte
* 描述   : 通过 SPI1 向指定端口寄存器写 4 个字节数据
* 输入   : s:端口号,reg:16 位寄存器地址,*dat_ptr:待写入的 4 个字节缓冲区指针
* 输出   : 无
* 返回值 : 无
* 说明   : 无
*******************************************************************************/
void Write_W5500_SOCK_4Byte(SOCKET s, unsigned short reg, unsigned char *dat_ptr)
{
   W5500_SCS=0;                    //置 W5500 的 SCS 为低电平

   SPI_Send_Short(reg);             //通过 SPI 写 16 位寄存器地址
   SPI_SendByte(FDM4|RWB_WRITE|(s*0x20+0x08));//通过 SPI 写控制字节,4 个字节数据长度,写数据,选择端口 s 的寄存器

   SPI_SendByte(*dat_ptr++);        //写第 1 个字节数据
   SPI_SendByte(*dat_ptr++);        //写第 2 个字节数据
   SPI_SendByte(*dat_ptr++);        //写第 3 个字节数据
   SPI_SendByte(*dat_ptr++);        //写第 4 个字节数据

```

```
21.    W5500_SCS=1;                    //置 W5500 的 SCS 为高电平
22. }
```

程序清单：硬件 SPI 读 W5500 指定地址寄存器 1 字节数据函数

```
1.  /*******************************************************************************
2.  * 函数名 : Read_W5500_1Byte
3.  * 描述   : 读 W5500 指定地址寄存器的 1 个字节数据
4.  * 输入   : reg:16 位寄存器地址
5.  * 输出   : 无
6.  * 返回值 : 读取到寄存器的 1 个字节数据
7.  * 说明   : 无
8.  *******************************************************************************/
9.  unsigned char Read_W5500_1Byte(unsigned short reg)
10. {
11.    unsigned char i;
12.
13.    W5500_SCS=0;                    //置 W5500 的 SCS 为低电平
14.
15.    SPI_Send_Short(reg);            //通过 SPI 写 16 位寄存器地址
16.    SPI_SendByte(FDM1|RWB_READ|COMMON_R);//通过 SPI 写控制字节,1 个字节数据长度,读数据,选
       择通用寄存器
17.
18.    i=SPI_SendByte(0xff);           //读取 1 个字节数据
19.
20.    W5500_SCS=1;                    //置 W5500 的 SCS 为高电平
21.    return i;                       //返回读取到的寄存器数据
22. }
```

程序清单：硬件 SPI 读 W5500 指定端口寄存器 1 字节数据函数

```
1.  /*******************************************************************************
2.  * 函数名 : Read_W5500_SOCK_1Byte
3.  * 描述   : 读 W5500 指定端口寄存器的 1 个字节数据
4.  * 输入   : s:端口号,reg:16 位寄存器地址
5.  * 输出   : 无
6.  * 返回值 : 读取到寄存器的 1 个字节数据
7.  * 说明   : 无
8.  *******************************************************************************/
9.  unsigned char Read_W5500_SOCK_1Byte(SOCKET s, unsigned short reg)
10. {
11.    unsigned char i;
12.
13.    W5500_SCS=0;                    //置 W5500 的 SCS 为低电平
14.
```

```
    SPI_Send_Short(reg);                    //通过 SPI 写 16 位寄存器地址
    SPI_SendByte(FDM1|RWB_READ|(s*0x20+0x08));//通过 SPI 写控制字节,1 个字节数据长度,读数据,选择端口 s 的寄存器

    i=SPI_SendByte(0xff);                   //读取 1 个字节数据

    W5500_SCS=1;                            //置 W5500 的 SCS 为高电平
    return i;                               //返回读取到的寄存器数据
}
```

程序清单：硬件 SPI 读 W5500 指定端口寄存器 2 字节数据函数

```
/*******************************************************************************
* 函数名 : Read_W5500_SOCK_2Byte
* 描述   : 读 W5500 指定端口寄存器的 2 个字节数据
* 输入   : s:端口号,reg:16 位寄存器地址
* 输出   : 无
* 返回值 : 读取到寄存器的 2 个字节数据(16 位)
* 说明   : 无
*******************************************************************************/
unsigned short Read_W5500_SOCK_2Byte(SOCKET s, unsigned short reg)
{
    unsigned short i;

    W5500_SCS=0;                            //置 W5500 的 SCS 为低电平

    SPI_Send_Short(reg);                    //通过 SPI 写 16 位寄存器地址
    SPI_SendByte(FDM2|RWB_READ|(s*0x20+0x08));//通过 SPI 写控制字节,2 个字节数据长度,读数据,选择端口 s 的寄存器

    i=SPI_SendByte(0xff);                   //读取高位数据
    i*=256;
    i+=SPI_SendByte(0xff);                  //读取低位数据后合并

    W5500_SCS=1;                            //置 W5500 的 SCS 为高电平
    return i;                               //返回读取到的寄存器数据
}
```

程序清单：硬件 SPI 读 W5500 接收数据缓冲区数据函数

```
/*******************************************************************************
* 函数名 : Read_SOCK_Data_Buffer
* 描述   : 从 W5500 接收数据缓冲区中读取数据
* 输入   : s:端口号,*dat_ptr:数据保存缓冲区指针
* 输出   : 无
* 返回值 : 读取到的数据长度,rx_size 个字节
```

```
7.  * 说明   : 无
8.  ******************************************************************************/
9.  unsigned short Read_SOCK_Data_Buffer(SOCKET s, unsigned char *dat_ptr)
10. {
11.   unsigned short rx_size;
12.   unsigned short offset, offset1;
13.   unsigned short i;
14.   unsigned char j;
15.
16.   rx_size=Read_W5500_SOCK_2Byte(s,Sn_RX_RSR);
17.   if(rx_size==0) return 0;//没接收到数据则返回
18.   if(rx_size>1460) rx_size=1460;
19.
20.   offset=Read_W5500_SOCK_2Byte(s,Sn_RX_RD);
21.   offset1=offset;
22.   offset&=(S_RX_SIZE-1);//计算实际的物理地址
23.
24.   W5500_SCS=0;                    //置 W5500 的 SCS 为低电平
25.
26.   SPI_Send_Short(offset);//写 16 位地址
27.   SPI_SendByte(VDM|RWB_READ|(s*0x20+0x18));//写控制字节,N 个字节数据长度,读数据,选择端口 s
      的寄存器
28.
29.   if((offset+rx_size)<S_RX_SIZE)//如果最大地址未超过 W5500 接收缓冲区寄存器的最大地址
30.   {
31.     for(i=0;i<rx_size;i++)//循环读取 rx_size 个字节数据
32.     {
33.       j=SPI_SendByte(0xff);//读取 1 个字节数据
34.       *dat_ptr=j;//将读取到的数据保存到数据保存缓冲区
35.       dat_ptr++;//数据保存缓冲区指针地址自增 1
36.     }
37.   }
38.   else//如果最大地址超过 W5500 接收缓冲区寄存器的最大地址
39.   {
40.     offset=S_RX_SIZE-offset;
41.     for(i=0;i<offset;i++)//循环读取出前 offset 个字节数据
42.     {
43.       j=SPI_SendByte(0xff); //读取 1 个字节数据
44.       *dat_ptr=j;//将读取到的数据保存到数据保存缓冲区
45.       dat_ptr++;//数据保存缓冲区指针地址自增 1
46.     }
47.     W5500_SCS=1;                  //置 W5500 的 SCS 为高电平
48.
49.     W5500_SCS=0;                  //置 W5500 的 SCS 为低电平
```

```
50.
51.     SPI_Send_Short(0x00);//写 16 位地址
52.     SPI_SendByte(VDM|RWB_READ|(s*0x20+0x18));//写控制字节,N 个字节数据长度,读数据,选择端口 s 的寄存器
53.
54.     for(;i<rx_size;i++)//循环读取后 rx_size-offset 个字节数据
55.     {
56.       j=SPI_SendByte(0xff);//读取 1 个字节数据
57.       *dat_ptr=j;//将读取到的数据保存到数据保存缓冲区
58.       dat_ptr++;//数据保存缓冲区指针地址自增 1
59.     }
60.   }
61.   W5500_SCS=1;                    //置 W5500 的 SCS 为高电平
62.
63.   offset1+=rx_size;//更新实际物理地址,即下次读取接收到的数据的起始地址
64.   Write_W5500_SOCK_2Byte(s, Sn_RX_RD, offset1);
65.   Write_W5500_SOCK_1Byte(s, Sn_CR, RECV);//发送启动接收命令
66.   return rx_size;//返回接收到数据的长度
67. }
```

程序清单：硬件 SPI 向 W5500 发送数据缓冲区写数据函数

```
1.  /*******************************************************************************
2.  * 函数名 : Write_SOCK_Data_Buffer
3.  * 描述   : 将数据写入 W5500 的数据发送缓冲区
4.  * 输入   : s:端口号,*dat_ptr:数据保存缓冲区指针,length:待写入数据的长度
5.  * 输出   : 无
6.  * 返回值 : 无
7.  * 说明   : 无
8.  *******************************************************************************/
9.  void Write_SOCK_Data_Buffer(SOCKET s, unsigned char *dat_ptr, unsigned short length)
10. {
11.   unsigned short offset,offset1;
12.   unsigned short i;
13.
14.   //如果是 UDP 模式,可以在此设置目的主机的 IP 和端口号
15.   if((Read_W5500_SOCK_1Byte(s,Sn_MR)&0x0f) != SOCK_UDP)//如果 Socket 打开失败
16.   {
17.     Write_W5500_SOCK_4Byte(s, Sn_DIPR, UDP_DIPR);//设置目的主机 IP
18.     Write_W5500_SOCK_2Byte(s, Sn_DPORTR, UDP_DPORT[0]*256+UDP_DPORT[1]);//设置目的主机端口号
19.   }
20.
21.   offset=Read_W5500_SOCK_2Byte(s,Sn_TX_WR);
22.   offset1=offset;
```

```
    offset&=(S_TX_SIZE-1);//计算实际的物理地址

    W5500_SCS=0;                    //置 W5500 的 SCS 为低电平

    SPI_Send_Short(offset);//写 16 位地址
    SPI_SendByte(VDM|RWB_WRITE|(s*0x20+0x10));//写控制字节,N 个字节数据长度,写数据,选择端口 s 的寄存器

    if((offset+length)<S_TX_SIZE)//如果最大地址未超过 W5500 发送缓冲区寄存器的最大地址
    {
      for(i=0;i<length;i++)//循环写入 length 个字节数据
      {
        SPI_SendByte(*dat_ptr++);//写入一个字节的数据
      }
    }
    else//如果最大地址超过 W5500 发送缓冲区寄存器的最大地址
    {
      offset=S_TX_SIZE-offset;
      for(i=0;i<offset;i++)//循环写入前 offset 个字节数据
      {
        SPI_SendByte(*dat_ptr++);//写入一个字节的数据
      }
      W5500_SCS=1;                  //置 W5500 的 SCS 为高电平

      W5500_SCS=0;                  //置 W5500 的 SCS 为低电平

      SPI_Send_Short(0x00);//写 16 位地址
      SPI_SendByte(VDM|RWB_WRITE|(s*0x20+0x10));//写控制字节,N 个字节数据长度,写数据,选择端口 s 的寄存器

      for(;i<length;i++)//循环写入 size-offset 个字节数据
      {
        SPI_SendByte(*dat_ptr++);//写入一个字节的数据
      }
    }
    W5500_SCS=1;                    //置 W5500 的 SCS 为高电平

    offset1+=length;//更新实际物理地址,即下次写待发送数据到发送数据缓冲区的起始地址
    Write_W5500_SOCK_2Byte(s, Sn_TX_WR, offset1);
    Write_W5500_SOCK_1Byte(s, Sn_CR, SEND);//发送启动发送命令
}
```

程序清单：硬件复位 W5500 函数

```
/*******************************************************************************
```

```
2.  * 函数名 : W5500_Hardware_Reset
3.  * 描述   : 硬件复位 W5500
4.  * 输入   : 无
5.  * 输出   : 无
6.  * 返回值 : 无
7.  * 说明   : W5500 的复位引脚保持低电平至少 500us 以上,才能重围 W5500
8.  ****************************************************************************/
9.  void W5500_Hardware_Reset(void)
10. {
11.    W5500_RST=0;              //复位引脚拉低
12.    delay_ms(200);
13.    W5500_RST=1;              //复位引脚拉高
14.    delay_ms(200);
15.    while((Read_W5500_1Byte(PHYCFGR)&LINK)==0);//等待以太网连接完成
16. }
```

程序清单：初始化 W5500 寄存器函数

```
1.  /****************************************************************************
2.  * 函数名 : W5500_Init
3.  * 描述   : 初始化 W5500 寄存器函数
4.  * 输入   : 无
5.  * 输出   : 无
6.  * 返回值 : 无
7.  * 说明   : 在使用 W5500 之前，先对 W5500 初始化
8.  ****************************************************************************/
9.  void W5500_Init(void)
10. {
11.    unsigned char i=0;
12.
13.    Write_W5500_1Byte(MR, RST);//软件复位 W5500,置 1 有效,复位后自动清 0
14.    delay_ms(10);//延时 10ms,自己定义该函数
15.
16.    //设置网关(Gateway)的 IP 地址,Gateway_IP 为 4 字节 unsigned char 数组,自己定义
17.    //使用网关可以使通信突破子网的局限，通过网关可以访问到其它子网或进入 Internet
18.    Write_W5500_nByte(GAR, Gateway_IP, 4);
19.
20.    //设置子网掩码(MASK)值,SUB_MASK 为 4 字节 unsigned char 数组,自己定义
21.    //子网掩码用于子网运算
22.    Write_W5500_nByte(SUBR,Sub_Mask,4);
23.
24.    //设置物理地址,PHY_ADDR 为 6 字节 unsigned char 数组,自己定义,用于唯一标识网络设备的物理地
   址值
25.    //该地址值需要到 IEEE 申请，按照 OUI 的规定，前 3 个字节为厂商代码，后三个字节为产品序号
26.    //如果自己定义物理地址，注意第一个字节必须为偶数
```

```
27.    Write_W5500_nByte(SHAR,Phy_Addr,6);
28.
29.    //设置本机的 IP 地址,IP_ADDR 为 4 字节 unsigned char 数组,自己定义
30.    //注意，网关 IP 必须与本机 IP 属于同一个子网，否则本机将无法找到网关
31.    Write_W5500_nByte(SIPR,IP_Addr,4);
32.
33.    //设置发送缓冲区和接收缓冲区的大小，参考 W5500 数据手册
34.    for(i=0;i<8;i++)
35.    {
36.      Write_W5500_SOCK_1Byte(i,Sn_RXBUF_SIZE, 0x02);//Socket Rx memory size=2k
37.      Write_W5500_SOCK_1Byte(i,Sn_TXBUF_SIZE, 0x02);//Socket Tx mempry size=2k
38.    }
39.
40.    //设置重试时间，默认为 2000(200ms)
41.    //每一单位数值为 100 微秒,初始化时值设为 2000(0x07D0),等于 200 毫秒
42.    Write_W5500_2Byte(RTR, 0x07d0);
43.
44.    //设置重试次数，默认为 8 次
45.    //如果重发的次数超过设定值,则产生超时中断(相关的端口中断寄存器中的 Sn_IR 超时位(TIMEOUT)
       置“1”)
46.    Write_W5500_1Byte(RCR,8);
47. }
```

程序清单：检查网关服务器函数

```
1.  /*******************************************************************************
2.  * 函数名 : Detect_Gateway
3.  * 描述   : 检查网关服务器
4.  * 输入   : 无
5.  * 输出   : 无
6.  * 返回值 : 成功返回 W_TRUE(0xFF),失败返回 W_FALSE(0x00)
7.  * 说明   : 无
8.  *******************************************************************************/
9.  unsigned char Detect_Gateway(void)
10. {
11.    unsigned char ip_adde[4];
12.    ip_adde[0]=IP_Addr[0]+1;
13.    ip_adde[1]=IP_Addr[1]+1;
14.    ip_adde[2]=IP_Addr[2]+1;
15.    ip_adde[3]=IP_Addr[3]+1;
16.
17.    //检查网关及获取网关的物理地址
18.    Write_W5500_SOCK_4Byte(0,Sn_DIPR,ip_adde);//向目的地址寄存器写入与本机 IP 不同的 IP 值
19.    Write_W5500_SOCK_1Byte(0,Sn_MR,MR_TCP);//设置 socket 为 TCP 模式
20.    Write_W5500_SOCK_1Byte(0,Sn_CR,OPEN);//打开 Socket
```

```
21.   delay_ms(5);//延时 5ms
22.
23.   if(Read_W5500_SOCK_1Byte(0,Sn_SR) != SOCK_INIT)//如果 socket 打开失败
24.   {
25.     Write_W5500_SOCK_1Byte(0,Sn_CR,CLOSE);//打开不成功,关闭 Socket
26.     return W_FALSE;//返回 FALSE(0x00)
27.   }
28.
29.   Write_W5500_SOCK_1Byte(0,Sn_CR,CONNECT);//设置 Socket 为 Connect 模式
30.
31.   do
32.   {
33.     unsigned char j=0;
34.     j=Read_W5500_SOCK_1Byte(0,Sn_IR);//读取 Socket0 中断标志寄存器
35.     if(j!=0)
36.     Write_W5500_SOCK_1Byte(0,Sn_IR,j);
37.     delay_ms(5);//延时 5ms
38.     if((j&IR_TIMEOUT) == IR_TIMEOUT)
39.     {
40.       return W_FALSE;
41.     }
42.     else if(Read_W5500_SOCK_1Byte(0,Sn_DHAR) != 0xff)
43.     {
44.       Write_W5500_SOCK_1Byte(0,Sn_CR,CLOSE);//关闭 Socket
45.       return W_TRUE;
46.     }
47.   }while(1);
48. }
```

程序清单：指定 Socket(0~7)初始化函数

```
1.  /*******************************************************************************
2.  * 函数名 : Socket_Init
3.  * 描述   : 指定 Socket(0~7)初始化
4.  * 输入   : s:待初始化的端口
5.  * 输出   : 无
6.  * 返回值 : 无
7.  * 说明   : 无
8.  *******************************************************************************/
9.  void Socket_Init(SOCKET s)
10. {
11.   //设置分片长度，参考 W5500 数据手册，该值可以不修改
12.   Write_W5500_SOCK_2Byte(0, Sn_MSSR, 1460);//最大分片字节数=1460(0x5b4)
13.   //设置指定端口
14.   switch(s)
```

```
15.     {
16.         case 0:
17.             //设置端口 0 的端口号
18.             Write_W5500_SOCK_2Byte(0, Sn_PORT, S0_Port[0]*256+S0_Port[1]);
19.             //设置端口 0 目的(远程)端口号
20.             Write_W5500_SOCK_2Byte(0, Sn_DPORTR, S0_DPort[0]*256+S0_DPort[1]);
21.             //设置端口 0 目的(远程)IP 地址
22.             Write_W5500_SOCK_4Byte(0, Sn_DIPR, S0_DIP);
23.
24.             break;
25.
26.         case 1:
27.             break;
28.
29.         case 2:
30.             break;
31.
32.         case 3:
33.             break;
34.
35.         case 4:
36.             break;
37.
38.         case 5:
39.             break;
40.
41.         case 6:
42.             break;
43.
44.         case 7:
45.             break;
46.
47.         default:
48.             break;
49.     }
50. }
```

程序清单：设置指定 Socket(0~7)为客户端与远程服务器连接函数

```
1. /*******************************************************************************
2. * 函数名 : Socket_Connect
3. * 描述   : 设置指定 Socket(0~7)为客户端与远程服务器连接
4. * 输入   : s:待设定的端口
5. * 输出   : 无
6. * 返回值 : 成功返回 W_TRUE(0xFF),失败返回 W_FALSE(0x00)
```

```
* 说明   ：当本机 Socket 工作在客户端模式时,引用该程序,与远程服务器建立连接
*         如果启动连接后出现超时中断，则与服务器连接失败,需要重新调用该程序连接
*         该程序每调用一次,就与服务器产生一次连接
*****************************************************************************/
unsigned char Socket_Connect(SOCKET s)
{
  Write_W5500_SOCK_1Byte(s,Sn_MR,MR_TCP);//设置 socket 为 TCP 模式
  Write_W5500_SOCK_1Byte(s,Sn_CR,OPEN);//打开 Socket
  delay_ms(5);  //延时 5ms
  if(Read_W5500_SOCK_1Byte(s,Sn_SR)!=SOCK_INIT)//如果 socket 打开失败
  {
    Write_W5500_SOCK_1Byte(s,Sn_CR,CLOSE);//打开不成功,关闭 Socket
    return W_FALSE;  //返回 FALSE(0x00)
  }
  Write_W5500_SOCK_1Byte(s,Sn_CR,CONNECT);//设置 Socket 为 Connect 模式
  return W_TRUE;  //返回 TRUE,设置成功
}
```

程序清单：设置指定 Socket(0~7)为服务器等待远程主机连接函数

```
/*****************************************************************************
* 函数名 : Socket_Listen
* 描述   : 设置指定 Socket(0~7)作为服务器等待远程主机的连接
* 输入   : s:待设定的端口
* 输出   : 无
* 返回值 : 成功返回 W_TRUE(0xFF),失败返回 W_FALSE(0x00)
* 说明   : 当本机 Socket 工作在服务器模式时,引用该程序,等等远程主机的连接
*         该程序只调用一次,就使 W5500 设置为服务器模式
*****************************************************************************/
unsigned char Socket_Listen(SOCKET s)
{
  Write_W5500_SOCK_1Byte(s,Sn_MR,MR_TCP);//设置 socket 为 TCP 模式
  Write_W5500_SOCK_1Byte(s,Sn_CR,OPEN);//打开 Socket
  delay_ms(5);//延时 5ms
  if(Read_W5500_SOCK_1Byte(s,Sn_SR)!=SOCK_INIT)//如果 socket 打开失败
  {
    Write_W5500_SOCK_1Byte(s,Sn_CR,CLOSE);//打开不成功,关闭 Socket
    return W_FALSE;//返回 FALSE(0x00)
  }
  Write_W5500_SOCK_1Byte(s,Sn_CR,LISTEN);//设置 Socket 为侦听模式
  delay_ms(5);//延时 5ms
  if(Read_W5500_SOCK_1Byte(s,Sn_SR)!=SOCK_LISTEN)//如果 socket 设置失败
  {
    Write_W5500_SOCK_1Byte(s,Sn_CR,CLOSE);//设置不成功,关闭 Socket
    return W_FALSE;//返回 FALSE(0x00)
```

```
26.    }
27.    return W_TRUE;
28.
29.    //至此完成了 Socket 的打开和设置侦听工作,至于远程客户端是否与它建立连接,则需要等待 Socket 中断,
30.    //以判断 Socket 的连接是否成功。参考 W5500 数据手册的 Socket 中断状态
31.    //在服务器侦听模式不需要设置目的 IP 和目的端口号
32. }
```

程序清单：设置指定 Socket(0~7)为 UDP 模式函数

```
1.  /*****************************************************************************
2.  * 函数名 : Socket_UDP
3.  * 描述   : 设置指定 Socket(0~7)为 UDP 模式
4.  * 输入   : s:待设定的端口
5.  * 输出   : 无
6.  * 返回值 : 成功返回 W_TRUE(0xFF),失败返回 W_FALSE(0x00)
7.  * 说明   : 如果 Socket 工作在 UDP 模式,引用该程序,在 UDP 模式下,Socket 通信不需要建立连接
8.  *         该程序只调用一次，就使 W5500 设置为 UDP 模式
9.  *****************************************************************************/
10. unsigned char Socket_UDP(SOCKET s)
11. {
12.    Write_W5500_SOCK_1Byte(s,Sn_MR,MR_UDP);//设置 Socket 为 UDP 模式*/
13.    Write_W5500_SOCK_1Byte(s,Sn_CR,OPEN);//打开 Socket*/
14.    delay_ms(5);//延时 5ms
15.    if(Read_W5500_SOCK_1Byte(s,Sn_SR)!=SOCK_UDP)//如果 Socket 打开失败
16.    {
17.       Write_W5500_SOCK_1Byte(s,Sn_CR,CLOSE);//打开不成功,关闭 Socket
18.       return W_FALSE;//返回 FALSE(0x00)
19.    }
20.    else
21.       return W_TRUE;
22.
23.    //至此完成了 Socket 的打开和 UDP 模式设置,在这种模式下它不需要与远程主机建立连接
24.    //因为 Socket 不需要建立连接,所以在发送数据前都可以设置目的主机 IP 和目的 Socket 的端口号
25.    //如果目的主机 IP 和目的 Socket 的端口号是固定的,在运行过程中没有改变,那么也可以在这里设置
26. }
```

程序清单：W5500 中断处理程序框架函数

```
1.  /*****************************************************************************
2.  * 函数名 : W5500_Interrupt_Process
3.  * 描述   : W5500 中断处理程序框架
4.  * 输入   : 无
5.  * 输出   : 无
6.  * 返回值 : 无
7.  * 说明   : 无
```

```
8.  ********************************************************************************/
9.  void W5500_Interrupt_Process(void)
10. {
11.   unsigned char i,j;
12.
13. IntDispose:
14.   i=Read_W5500_1Byte(SIR);//读取端口中断标志寄存器
15.   if((i & S0_INT) == S0_INT)//Socket0 事件处理
16.   {
17.     j=Read_W5500_SOCK_1Byte(0,Sn_IR);//读取 Socket0 中断标志寄存器
18.     Write_W5500_SOCK_1Byte(0,Sn_IR,j);
19.     if(j&IR_CON)//在 TCP 模式下,Socket0 成功连接
20.     {
21.       S0_State|=S_CONN;//网络连接状态 0x02,端口完成连接，可以正常传输数据
22.     }
23.     if(j&IR_DISCON)//在 TCP 模式下 Socket 断开连接处理
24.     {
25.       Write_W5500_SOCK_1Byte(0,Sn_CR,CLOSE);//关闭端口,等待重新打开连接
26.       Socket_Init(0);    //指定 Socket(0~7)初始化,初始化端口 0
27.       S0_State=0;//网络连接状态 0x00,端口连接失败
28.     }
29.     if(j&IR_SEND_OK)//Socket0 数据发送完成,可以再次启动 S_tx_process()函数发送数据
30.     {
31.       S0_Data|=S_TRANSMITOK;//端口发送一个数据包完成
32.     }
33.     if(j&IR_RECV)//Socket 接收到数据,可以启动 S_rx_process()函数
34.     {
35.       S0_Data|=S_RECEIVE;//端口接收到一个数据包
36.     }
37.     if(j&IR_TIMEOUT)//Socket 连接或数据传输超时处理
38.     {
39.       Write_W5500_SOCK_1Byte(0,Sn_CR,CLOSE);// 关闭端口,等待重新打开连接
40.       S0_State=0;//网络连接状态 0x00,端口连接失败
41.     }
42.   }
43.
44.   if(Read_W5500_1Byte(SIR) != 0)
45.     goto IntDispose;
46. }
```

然后，介绍下装载网络参数的函数，本例程使用的是 Socket0，目的 IP 是电脑的 IP，本机 IP 地址针对的是 W5500 模块的，同理，目的端口号指的也是电脑，而加载的端口号指的就是 W5500 的。由于本实验做的是 W5500 模块的客户端实验，所以模式配置成 TCP_CLIENT。

代码清单：W5500 装载网络参数

```
/*******************************************************************************
* 函数名 : Load_Net_Parameters
* 描述   : 装载网络参数
* 输入   : 无
* 输出   : 无
* 返回值 : 无
* 说明   : 网关、掩码、物理地址、本机 IP 地址、端口号、目的 IP 地址、目的端口号、端口工作模式
*******************************************************************************/
void Load_Net_Parameters(void)
{
    Gateway_IP[0] = 192;//加载网关参数
    Gateway_IP[1] = 168;
    Gateway_IP[2] = 1;
    Gateway_IP[3] = 1;

    Sub_Mask[0]=255;//加载子网掩码
    Sub_Mask[1]=255;
    Sub_Mask[2]=255;
    Sub_Mask[3]=0;

    Phy_Addr[0]=0x0c;//加载物理地址
    Phy_Addr[1]=0x29;
    Phy_Addr[2]=0xab;
    Phy_Addr[3]=0x7c;
    Phy_Addr[4]=0x00;
    Phy_Addr[5]=0x01;

    IP_Addr[0]=192;//加载本机 IP 地址
    IP_Addr[1]=168;
    IP_Addr[2]=1;
    IP_Addr[3]=101;

    S0_Port[0] = 0x13;//加载端口 0 的端口号 5000
    S0_Port[1] = 0x88;

    S0_DIP[0]=192;//加载端口 0 的目的 IP 地址
    S0_DIP[1]=168;
    S0_DIP[2]=1;
    S0_DIP[3]=119;

    S0_DPort[0] = 0x17;//加载端口 0 的目的端口号 6000
    S0_DPort[1] = 0x70;
```

```
43.
44.    S0_Mode=TCP_CLIENT;//加载端口 0 的工作模式,TCP 客户端模式
45. }
```

最后，在主函数中对与控制 W5500 有关的函数进行初始化后，循环向目的 IP 发送固定的字符串。

代码清单：主函数

```
1.  int main(void)
2.  {
3.
4.  //  unsigned int W5500_Delay_Counter =0;
5.  ////////////////////////////////////////////////////////
6.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
7.  //     高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
8.  //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
9.  //       P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
10. ////////////////////////////////////////////////////////
11.    P3M1  &= 0xFB;  P3M0 &= 0xFB;        //设置 P3.2 为准双向口
12.    P4M1  &= 0xF0;  P4M0 &= 0xF0;        //设置 P4.0~P4.3 为准双向口
13.    P5M1  &= 0xEF;  P5M0 &= 0xEF;        //设置 P5.4 为准双向口
14.
15.    W5500_SCS=1;                         //置 W5500 的 SCS 为高电平
16.    W5500_RST=0;                         //复位引脚拉低
17.    Timer0Init();                        //定时器 0 初始化
18.    Init_SPI();                          //硬件 SPI 初始化
19.    EA = 1;                              //使能总中断
20.    delay_ms(10);                        //初始化后延时
21.    Load_Net_Parameters();               //装载网络参数
22.    W5500_Hardware_Reset();              //硬件复位 W5500
23.    W5500_Initialization();              //W5500 初始货配置
24.
25.    while (1)
26.    {
27.      W5500_Socket_Set();//W5500 端口初始化配置
28.
29.      W5500_Interrupt_Process();//W5500 中断处理程序框架
30.
31.      if((S0_Data & S_RECEIVE) == S_RECEIVE)//如果 Socket0 接收到数据
32.      {
33.        S0_Data&=~S_RECEIVE;
34.        Process_Socket_Data(0);//W5500 接收并发送接收到的数据
35.      }
36.      else if(W5500_Send_Delay_Counter >= 500)//定时发送字符串
```

```
37.     {
38.       if(S0_State == (S_INIT|S_CONN))
39.       {
40.         S0_Data&=~S_TRANSMITOK;
41.         memcpy(Tx_Buffer, "\r\nWelcome To IKMSIK!\r\n", 20);
42.         Write_SOCK_Data_Buffer(0, Tx_Buffer, 20);//指定 Socket(0~7)发送数据处理,端口 0 发送 20 字节数据
43.       }
44.       W5500_Send_Delay_Counter =0;
45.     }
46.   }
47. }
```

4.3.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 2 – 传感器实验程序\”目录下的压缩文件“实验 3-30-1：W5500 模块客户端模式例程（查询方式）”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\W5500\projec”目录下的工程“W5500.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“W5500.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 使用网线连接 W5500 模块和电脑网口，程序运行后，打开 TCP&UDP 测试工具，如下图所示。

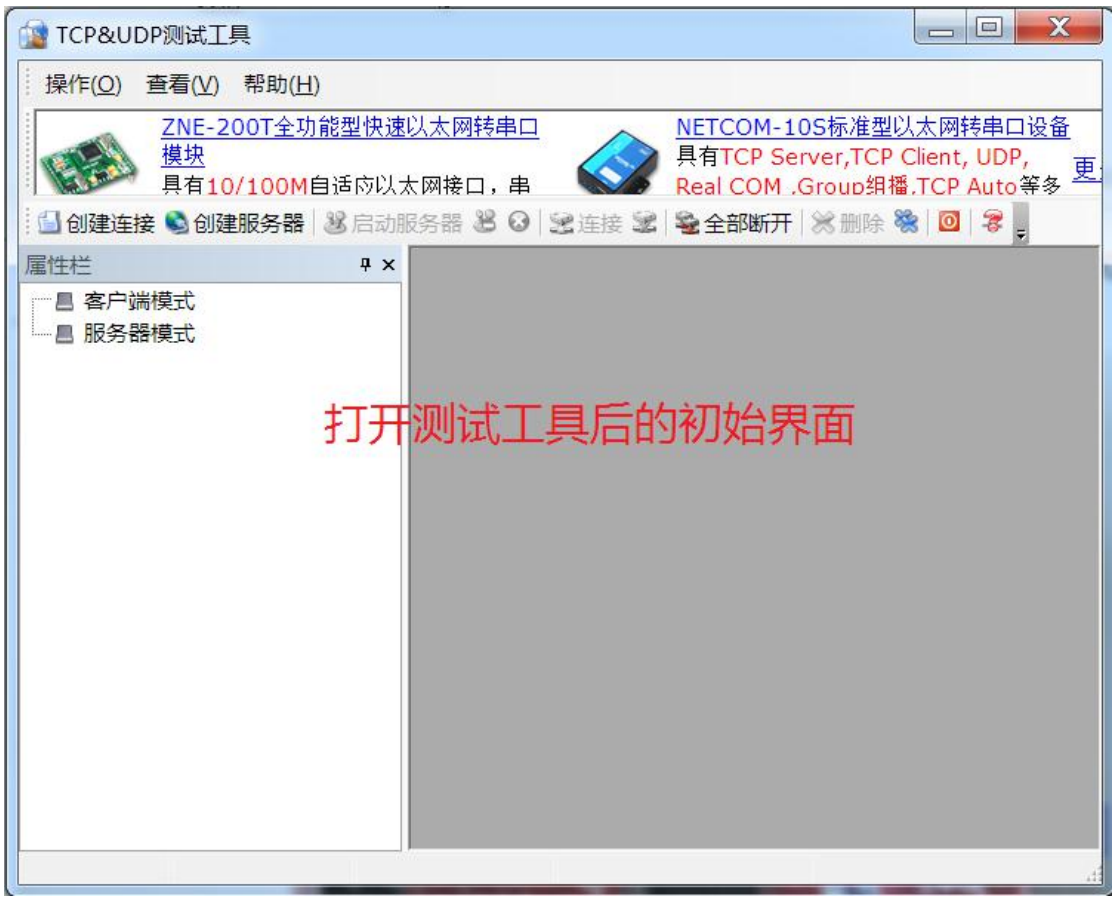

图 13：打开 TCP&UDP 测试工具

7. 点击创建服务器，本机端口选择 6000，如下图所示。

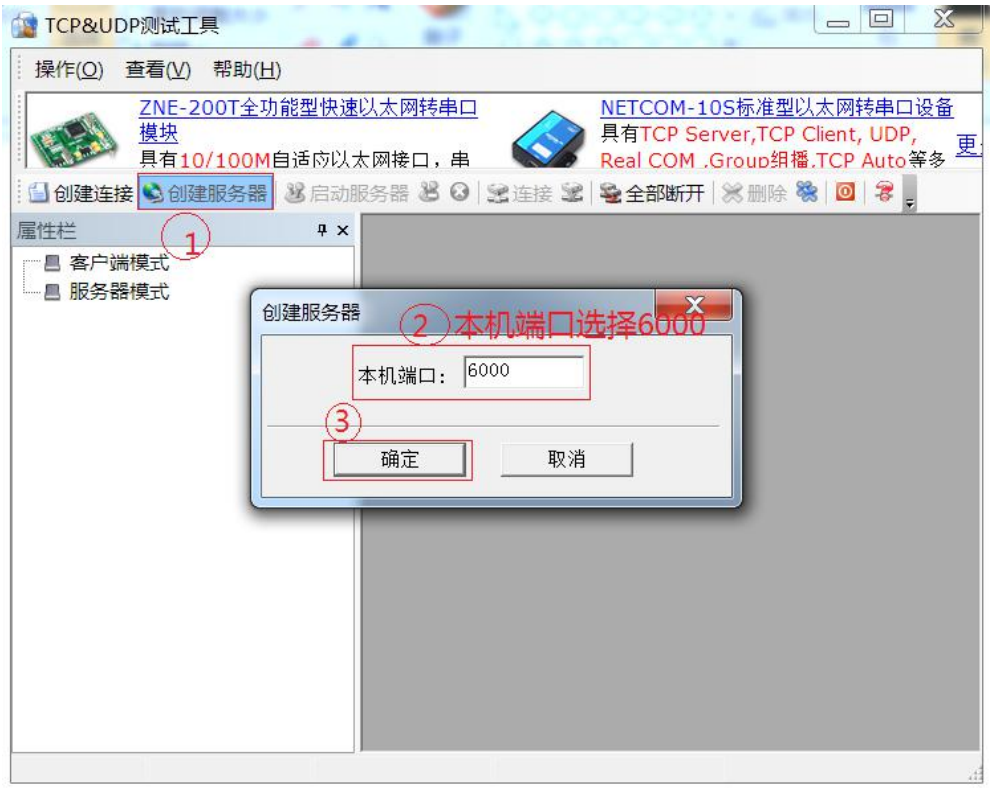

图 14：创建服务器界面

✧ 注：端口号设置 6000，是为了与例程中的目的端口号对应。

8. 创建服务器成功，出现下图界面。

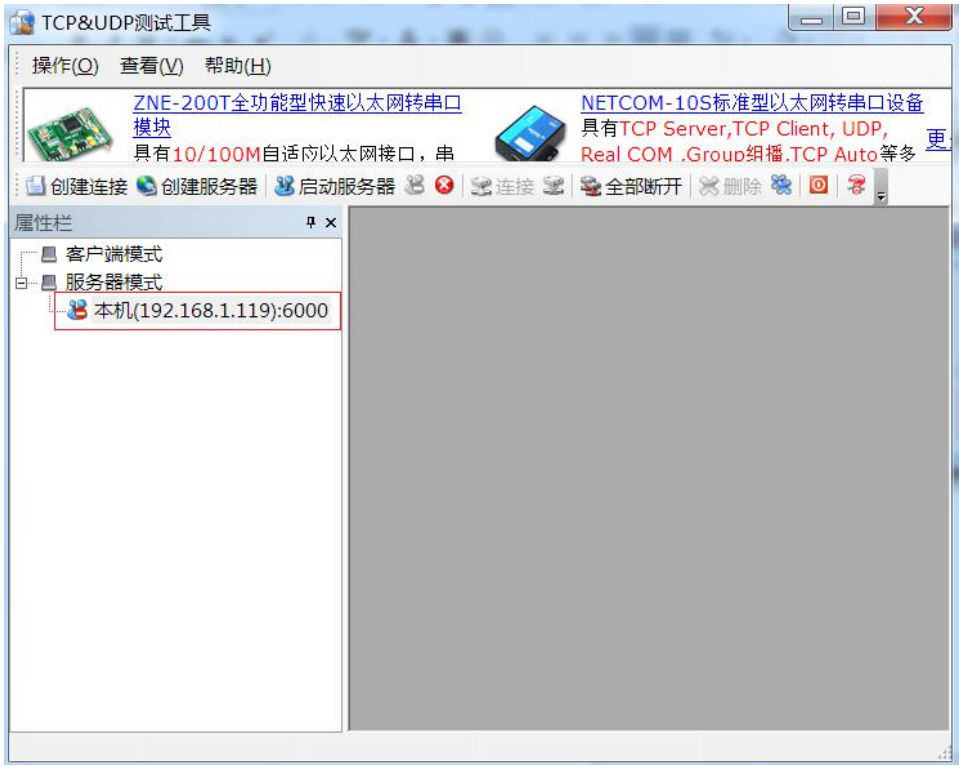

图 15：创建服务器成功

9. 点击启动服务器，可以观察到如下图所示的通信界面。说明本实验测试成功。

图 16：客户端模式正常通信界面

✧ 注：在实验过程中 W5500 模块的 W5500 芯片会发热，只要发热不过于剧烈，均属于正常现象。

4.4. W5500 模块客户端模式实验（中断方式）

✧ 注：本节的实验源码是在“实验 3-30-1：W5500 模块客户端模式例程（查询方式）”的基础上修改。本节对应的实验源码是：“实验 3-30-4：W5500 模块客户端模式例程（中断方式）”。

4.4.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 3-30-1：W5500 模块客户端模式例程（查询方式）”部分。

4.4.2. 编写代码

首先，在 w5500.c 文件中编写操作 W5500 以太网模块会用到的基本函数，请参考“实验 3-30-1：W5500 模块客户端模式例程（查询方式）”部分。但因为用到了中断引脚，所以新增加了外部中断 0 的初始化函数，并在 W5500 中断处理程序框架函数 W5500_Interrupt_Process 中增加了中断部分。

程序清单：初始化 W5500 用中断引脚

```
/*******************************************************************************
功能描述：外部中断 0 初始化
入口参数：无
返回值：无
*******************************************************************************/
void INT0_Init(void)
{
    IE0  = 0;      //将 INT0 中断请求标志位清"0"
    EX0 = 1;       //使能 INT0 中断允许位
    IT0 = 1;       //选择 INT0 为下降沿触发方式
}
```

程序清单：外部中断服务函数

```
/*******************************************************************************
功能描述：外部中断服务程序
入口参数：无
返回值：无
*******************************************************************************/
void INT0_Isr (void) interrupt INT0_VECTOR
{
    W5500_Interrupt=1;      //W5500 中断标志
}
```

程序清单：W5500 中断处理程序框架函数

```
/*******************************************************************************
* 函数名 : W5500_Interrupt_Process
* 描述   : W5500 中断处理程序框架
* 输入   : 无
* 输出   : 无
* 返回值 : 无
* 说明   : 无
*******************************************************************************/
void W5500_Interrupt_Process(void)
{
    unsigned char i,j;

IntDispose:
    W5500_Interrupt=0;//清零中断标志
    i = Read_W5500_1Byte(IR);//读取中断标志寄存器
    Write_W5500_1Byte(IR, (i&0xf0));//回写清除中断标志

    if((i & CONFLICT) == CONFLICT)//IP 地址冲突异常处理
```

```
19.     {
20.         //自己添加代码
21.     }
22.
23.     if((i & UNREACH) == UNREACH)//UDP 模式下地址无法到达异常处理
24.     {
25.       //自己添加代码
26.     }
27.
28.     i=Read_W5500_1Byte(SIR);//读取端口中断标志寄存器
29.     if((i & S0_INT) == S0_INT)//Socket0 事件处理
30.     {
31.       j=Read_W5500_SOCK_1Byte(0,Sn_IR);//读取 Socket0 中断标志寄存器
32.       Write_W5500_SOCK_1Byte(0,Sn_IR,j);
33.       if(j&IR_CON)//在 TCP 模式下,Socket0 成功连接
34.       {
35.         S0_State|=S_CONN;//网络连接状态 0x02,端口完成连接，可以正常传输数据
36.       }
37.       if(j&IR_DISCON)//在 TCP 模式下 Socket 断开连接处理
38.       {
39.         Write_W5500_SOCK_1Byte(0,Sn_CR,CLOSE);//关闭端口,等待重新打开连接
40.         Socket_Init(0);    //指定 Socket(0~7)初始化,初始化端口 0
41.         S0_State=0;//网络连接状态 0x00,端口连接失败
42.       }
43.       if(j&IR_SEND_OK)//Socket0 数据发送完成,可以再次启动 S_tx_process()函数发送数据
44.       {
45.         S0_Data|=S_TRANSMITOK;//端口发送一个数据包完成
46.       }
47.       if(j&IR_RECV)//Socket 接收到数据,可以启动 S_rx_process()函数
48.       {
49.         S0_Data|=S_RECEIVE;//端口接收到一个数据包
50.       }
51.       if(j&IR_TIMEOUT)//Socket 连接或数据传输超时处理
52.       {
53.         Write_W5500_SOCK_1Byte(0,Sn_CR,CLOSE);// 关闭端口,等待重新打开连接
54.         S0_State=0;//网络连接状态 0x00,端口连接失败
55.       }
56.     }
57.
58.     if(Read_W5500_1Byte(SIR) != 0)
59.       goto IntDispose;
60. }
```

然后，在主函数中对与控制 W5500 有关的函数进行初始化后，循环向目的 IP 发送固定的字符串。

代码清单：主函数

```
1.  int main(void)
2.  {
3.
4.  ////////////////////////////////////////////////////////
5.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
6.  //     高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
7.  //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
8.  //       P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
9.  ////////////////////////////////////////////////////////
10.    P3M1  &= 0xFB;  P3M0 &= 0xFB;        //设置 P3.2 为准双向口
11.    P4M1  &= 0xF0;  P4M0 &= 0xF0;        //设置 P4.0~P4.3 为准双向口
12.    P5M1  &= 0xEF;  P5M0 &= 0xEF;        //设置 P5.4 为准双向口
13.
14.    W5500_SCS=1;                         //置 W5500 的 SCS 为高电平
15.    W5500_RST=0;                         //复位引脚拉低
16.    INT0_Init();                        //外部中断 0 的初始化配置
17.    Timer0Init();                       //定时器 0 初始化
18.    Init_SPI();                        //硬件 SPI 初始化
19.    EA = 1;                            //使能总中断
20.    delay_ms(10);                       //初始化后延时
21.    Load_Net_Parameters();                  //装载网络参数
22.    W5500_Hardware_Reset();                 //硬件复位 W5500
23.    W5500_Initialization();                //W5500 初始货配置
24.
25.    while (1)
26.    {
27.      W5500_Socket_Set();//W5500 端口初始化配置
28.
29.    if(W5500_Interrupt)       //处理 W5500 中断
30.     {
31.        W5500_Interrupt_Process();//W5500 中断处理程序框架
32.      }
33.
34.      if((S0_Data & S_RECEIVE) == S_RECEIVE)//如果 Socket0 接收到数据
35.      {
36.        S0_Data&=~S_RECEIVE;
37.        Process_Socket_Data(0);//W5500 接收并发送接收到的数据
38.      }
39.      else if(W5500_Send_Delay_Counter >= 500)//定时发送字符串
```

```
40.     {
41.       if(S0_State == (S_INIT|S_CONN))
42.       {
43.         S0_Data&=~S_TRANSMITOK;
44.         memcpy(Tx_Buffer, "\r\nWelcome To IKMSIK!\r\n", 20);
45.         Write_SOCK_Data_Buffer(0, Tx_Buffer, 20);//指定 Socket(0~7)发送数据处理,端口 0 发送 20 字节
    数据
46.       }
47.       W5500_Send_Delay_Counter =0;
48.     }
49.   }
50. }
```

4.4.3. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 2 – 传感器实验程序\”目录下的压缩文件“实验 3-30-4：W5500 模块客户端模式例程（中断方式）”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\W5500\projec”目录下的工程“W5500.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“W5500.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 使用网线连接 W5500 模块和电脑网口，程序运行后，按照 4.3.4 中的步骤操作 TCP&UDP 测试工具做通客户端模式下的通信实验