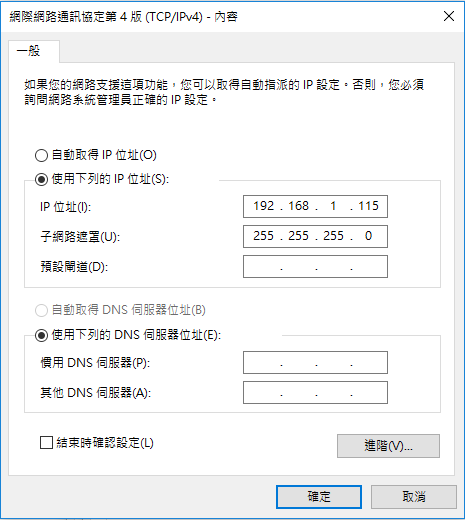
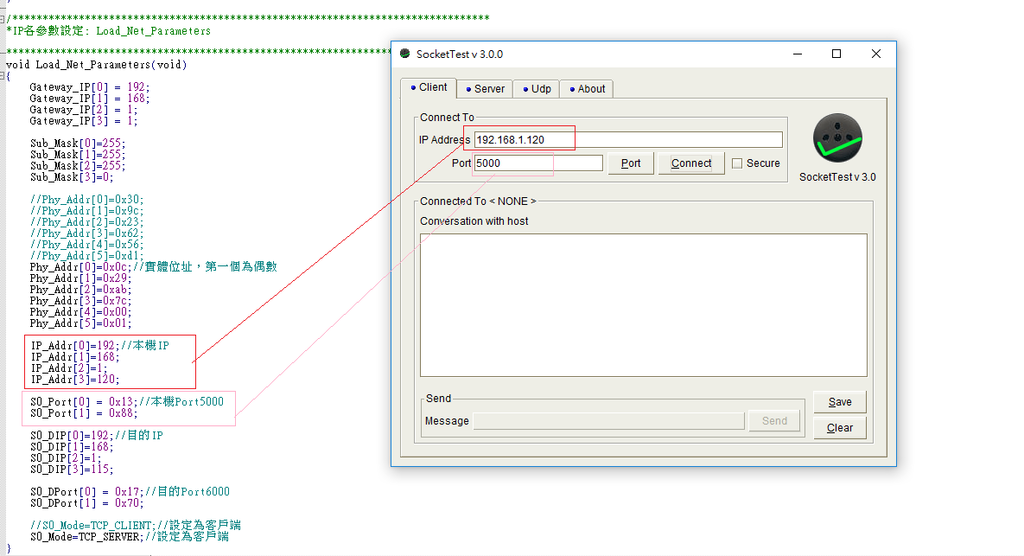
電腦網卡的IP設定需要符合程式內IP(192.168.1.115)



而且Socket Test的IP設定需要符合程式內IP(192.168.1.120  Port:5000)



程式碼在google找尋許久

大概都是STM32與W5500運用

8051與W5500不太好找

其中有一部分能編譯

實際寫入8051之後無法RUN

老男人大概花了3天時間

又是看說明書又是參考比對程式

最後才成功連線

請珍惜使用

W5500說明書

<http://file1.dzsc.com/product/14/09/02/54548_110126098.pdf>

------程式碼------

//---main.c---//

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* 硬體接線    P3^2 -> W5500\_RST     
 \*                  P3^3 -> W5500\_INT(本例未使用，可不接)       
 \*                  P3^6 -> W5500\_SCS        
 \*                  P3^7 -> W5500\_SCK      
 \*                  P3^4 -> W5500\_MISO      
 \*                  P3^5 -> W5500\_MOSI      
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*IP設定\*/  
//閘道:192.168.1.1  
//遮罩:    255.255.255.0  
//實體位址:0c 29 ab 7c 00 01  
//本機IP:192.168.1.120  
//本機通信Port:5000  
//目的:192.168.1.115  
//目的通信Port:6000

#include <reg51.h>  
#include "W5500.h"                  
#include <string.h>

void Delay(unsigned int d);            //延遲函式(ms)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* 初始化 : W5500\_Initialization

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
void W5500\_Initialization(void)  
{  
    W5500\_Init();        //初始化W5500暫存器  
    Detect\_Gateway();    //檢查網路服務器   
    Socket\_Init(0);        //指定Socket(0~7)初始化  
}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*IP各參數設定: Load\_Net\_Parameters

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
void Load\_Net\_Parameters(void)  
{  
    Gateway\_IP[0] = 192;  
    Gateway\_IP[1] = 168;  
    Gateway\_IP[2] = 1;  
    Gateway\_IP[3] = 1;

    Sub\_Mask[0]=255;  
    Sub\_Mask[1]=255;  
    Sub\_Mask[2]=255;  
    Sub\_Mask[3]=0;

    //Phy\_Addr[0]=0x30;  
    //Phy\_Addr[1]=0x9c;  
    //Phy\_Addr[2]=0x23;  
    //Phy\_Addr[3]=0x62;  
    //Phy\_Addr[4]=0x56;  
    //Phy\_Addr[5]=0xd1;  
    Phy\_Addr[0]=0x0c;//實體位址，第一個為偶數  
    Phy\_Addr[1]=0x29;  
    Phy\_Addr[2]=0xab;  
    Phy\_Addr[3]=0x7c;  
    Phy\_Addr[4]=0x00;  
    Phy\_Addr[5]=0x01;

    IP\_Addr[0]=192;//本機IP  
    IP\_Addr[1]=168;  
    IP\_Addr[2]=1;  
    IP\_Addr[3]=120;

    S0\_Port[0] = 0x13;//本機Port5000   
    S0\_Port[1] = 0x88;

    S0\_DIP[0]=192;//目的IP  
    S0\_DIP[1]=168;  
    S0\_DIP[2]=1;  
    S0\_DIP[3]=115;  
      
    S0\_DPort[0] = 0x17;//目的Port6000  
    S0\_DPort[1] = 0x70;

    //S0\_Mode=TCP\_CLIENT;//設定為客戶端  
    S0\_Mode=TCP\_SERVER;//設定為伺服端  
}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* TCP/UDP分類: W5500\_Socket\_Set

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
void W5500\_Socket\_Set(void)  
{  
    if(S0\_State==0)//0初始化配置  
    {  
        if(S0\_Mode==TCP\_SERVER)//TCP Server模式  
        {  
            if(Socket\_Listen(0)==TRUE)  
                S0\_State=S\_INIT;  
            else  
                S0\_State=0;  
        }  
        else if(S0\_Mode==TCP\_CLIENT)//TCP Client模式  
        {  
            if(Socket\_Connect(0)==TRUE)  
                S0\_State=S\_INIT;  
            else  
                S0\_State=0;  
        }  
        else//UDP模式  
        {  
              
            if(Socket\_UDP(0)==TRUE)  
                S0\_State=S\_INIT|S\_CONN;  
            else  
                S0\_State=0;  
        }  
    }  
}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* 接收發送數據: Process\_Socket\_Data

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
void Process\_Socket\_Data(SOCKET s)  
{  
    unsigned short size;  
    size=Read\_SOCK\_Data\_Buffer(s, Rx\_Buffer);  
    memcpy(Tx\_Buffer, Rx\_Buffer, size);      
    //將Tx\_buffer數據，寫入SD卡筆記本test.txt(沒有SD卡無視)  
              
    Write\_SOCK\_Data\_Buffer(s, Tx\_Buffer, size);  
}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* 主函式: main  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
int main(void)  
{  
    unsigned int W5500\_Send\_Delay\_Counter =0;

    Load\_Net\_Parameters();        //      
    W5500\_Hardware\_Reset();        //W5500硬體RST  
    W5500\_Initialization();                        
    W5500\_Socket\_Set();

    while (1)  
    {  
        W5500\_Socket\_Set();            //W5500端口初始化  
        W5500\_Interrupt\_Process();//W5500中斷處理程序框架

        if((S0\_Data & S\_RECEIVE) == S\_RECEIVE)//如果Socket0接收到數據  
        {  
            S0\_Data&=~S\_RECEIVE;  
            Process\_Socket\_Data(0);//W5500接收發送接收到的數據  
        }  
        else if(W5500\_Send\_Delay\_Counter >= 3000)//定時發送字元  
        {  
            if(S0\_State == (S\_INIT|S\_CONN))  
            {  
                S0\_Data&=~S\_TRANSMITOK;  
                memcpy(Tx\_Buffer,"Have a good time.  \r\n", 20);      
                Write\_SOCK\_Data\_Buffer(0, Tx\_Buffer, 20);//指定Socket(0~7)發送數據,端口0發送20字元   
            }  
            W5500\_Send\_Delay\_Counter=0;  
        }  
        W5500\_Send\_Delay\_Counter++;  
    }  
}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* 延遲函式: Delay (ms)  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay(unsigned int  x)  
{  
    unsigned int i,j;

    for(j=0;j<5;j++)  
        for(i=0;i<x;i++);  
}

////////////////////////////////////////////////////////////////

//---W5500.c---//

#include "W5500.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*----- 定義區 -----\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
unsigned char Gateway\_IP[4];  
unsigned char Sub\_Mask[4];      
unsigned char Phy\_Addr[6];    //實體位址(MAC)   
unsigned char IP\_Addr[4];    //IP

unsigned char S0\_Port[2];      
unsigned char S0\_DIP[4];      
unsigned char S0\_DPort[2];

unsigned char UDP\_DIPR[4];  
unsigned char UDP\_DPORT[2];

unsigned char S0\_Mode = 1;    //0:TCP Server,1:TCP Client,2:UDP

#define TCP\_SERVER    0x00    //TCP  
#define TCP\_CLIENT    0x01    //TCP  
#define UDP\_MODE    0x02    //UDP

unsigned char S0\_State =0;      
#define S\_INIT        0x01      
#define S\_CONN        0x02

unsigned char S0\_Data;          
#define S\_RECEIVE     0x01   
#define S\_TRANSMITOK 0x02

unsigned char Rx\_Buffer[30];      
unsigned char Tx\_Buffer[30];

unsigned char W5500\_Interrupt;

unsigned char SPI\_Read\_Byte(void)  
{  
    unsigned char i,rByte=0;  
      
    W5500\_SCLK=0;  
    for(i=0;i<8;i++)  
    {  
        W5500\_SCLK=1;  
        rByte<<=1;  
        rByte|=W5500\_MISO;  
        W5500\_SCLK=0;      
    }  
    return rByte;  
}

void SPI\_Send\_Byte(unsigned char dt)  
{  
    unsigned char i;  
          
    for(i=0;i<8;i++)  
    {      
        W5500\_SCLK=0;  
        if((dt<<i)&0x80)  
            W5500\_MOSI=1;  
        else  
            W5500\_MOSI=0;                      
        W5500\_SCLK=1;  
    }  
    W5500\_SCLK=0;  
}

void SPI\_Send\_Short(unsigned short dt)  
{  
    SPI\_Send\_Byte((unsigned char)(dt/256));      
    SPI\_Send\_Byte(dt);                          
}

void Write\_W5500\_1Byte(unsigned short reg, unsigned char dat)  
{  
    W5500\_SCS=0;

    SPI\_Send\_Short(reg);  
    SPI\_Send\_Byte(FDM1|RWB\_WRITE|COMMON\_R);  
    SPI\_Send\_Byte(dat);

    W5500\_SCS=1;   
}

void Write\_W5500\_2Byte(unsigned short reg, unsigned short dat)  
{  
    W5500\_SCS=0;  
          
    SPI\_Send\_Short(reg);  
    SPI\_Send\_Byte(FDM2|RWB\_WRITE|COMMON\_R);  
    SPI\_Send\_Short(dat);

    W5500\_SCS=1;   
}

void Write\_W5500\_nByte(unsigned short reg, unsigned char \*dat\_ptr, unsigned short size)  
{  
    unsigned short i;

    W5500\_SCS=0;      
          
    SPI\_Send\_Short(reg);  
    SPI\_Send\_Byte(VDM|RWB\_WRITE|COMMON\_R);

    for(i=0;i<size;i++)  
    {  
        SPI\_Send\_Byte(\*dat\_ptr++);  
    }

    W5500\_SCS=1;   
}

void Write\_W5500\_SOCK\_1Byte(SOCKET s, unsigned short reg, unsigned char dat)  
{  
    W5500\_SCS=0;  
          
    SPI\_Send\_Short(reg);  
    SPI\_Send\_Byte(FDM1|RWB\_WRITE|(s\*0x20+0x08));  
    SPI\_Send\_Byte(dat);

    W5500\_SCS=1;  
}

void Write\_W5500\_SOCK\_2Byte(SOCKET s, unsigned short reg, unsigned short dat)  
{  
    W5500\_SCS=0;  
              
    SPI\_Send\_Short(reg);  
    SPI\_Send\_Byte(FDM2|RWB\_WRITE|(s\*0x20+0x08));  
    SPI\_Send\_Short(dat);

    W5500\_SCS=1;  
}

void Write\_W5500\_SOCK\_4Byte(SOCKET s, unsigned short reg, unsigned char \*dat\_ptr)  
{  
    W5500\_SCS=0;  
              
    SPI\_Send\_Short(reg);  
    SPI\_Send\_Byte(FDM4|RWB\_WRITE|(s\*0x20+0x08));

    SPI\_Send\_Byte(\*dat\_ptr++);  
    SPI\_Send\_Byte(\*dat\_ptr++);  
    SPI\_Send\_Byte(\*dat\_ptr++);  
    SPI\_Send\_Byte(\*dat\_ptr++);

    W5500\_SCS=1;  
}

unsigned char Read\_W5500\_1Byte(unsigned short reg)  
{  
    unsigned char i;

    W5500\_SCS=0;  
              
    SPI\_Send\_Short(reg);  
    SPI\_Send\_Byte(FDM1|RWB\_READ|COMMON\_R);  
    i=SPI\_Read\_Byte();

    W5500\_SCS=1;  
    return i;  
}

unsigned char Read\_W5500\_SOCK\_1Byte(SOCKET s, unsigned short reg)  
{  
    unsigned char i;

    W5500\_SCS=0;  
              
    SPI\_Send\_Short(reg);  
    SPI\_Send\_Byte(FDM1|RWB\_READ|(s\*0x20+0x08));  
    i=SPI\_Read\_Byte();

    W5500\_SCS=1;  
    return i;  
}

unsigned short Read\_W5500\_SOCK\_2Byte(SOCKET s, unsigned short reg)  
{  
    unsigned short i;

    W5500\_SCS=0;  
              
    SPI\_Send\_Short(reg);  
    SPI\_Send\_Byte(FDM2|RWB\_READ|(s\*0x20+0x08));

    i=SPI\_Read\_Byte();  
    i\*=256;  
    i+=SPI\_Read\_Byte();  
    W5500\_SCS=1;  
    return i;  
}

unsigned short Read\_SOCK\_Data\_Buffer(SOCKET s, unsigned char \*dat\_ptr)  
{  
    unsigned short rx\_size;  
    unsigned short offset, offset1;  
    unsigned short i;  
    unsigned char j;

    rx\_size=Read\_W5500\_SOCK\_2Byte(s,Sn\_RX\_RSR);  
    if(rx\_size==0) return 0;  
    if(rx\_size>1460) rx\_size=1460;

    offset=Read\_W5500\_SOCK\_2Byte(s,Sn\_RX\_RD);  
    offset1=offset;  
    offset&=(S\_RX\_SIZE-1);  
    W5500\_SCS=0;

    SPI\_Send\_Short(offset);  
    SPI\_Send\_Byte(VDM|RWB\_READ|(s\*0x20+0x18));  
      
    if((offset+rx\_size)<S\_RX\_SIZE)  
    {  
        for(i=0;i<rx\_size;i++)  
        {  
            j=SPI\_Read\_Byte();  
            \*dat\_ptr=j;  
            dat\_ptr++;  
        }  
    }  
    else  
    {  
        offset=S\_RX\_SIZE-offset;  
        for(i=0;i<offset;i++)  
        {  
            j=SPI\_Read\_Byte();  
            \*dat\_ptr=j;  
            dat\_ptr++;  
        }  
        W5500\_SCS=1;

        W5500\_SCS=0;

        SPI\_Send\_Short(0x00);  
        SPI\_Send\_Byte(VDM|RWB\_READ|(s\*0x20+0x18));

        for(;i<rx\_size;i++)  
        {  
            j=SPI\_Read\_Byte();  
            \*dat\_ptr=j;  
            dat\_ptr++;  
        }  
    }  
    W5500\_SCS=1;

    offset1+=rx\_size;  
    Write\_W5500\_SOCK\_2Byte(s, Sn\_RX\_RD, offset1);  
    Write\_W5500\_SOCK\_1Byte(s, Sn\_CR, RECV);  
    return rx\_size;  
}

void Write\_SOCK\_Data\_Buffer(SOCKET s, unsigned char \*dat\_ptr, unsigned short size)  
{  
    unsigned short offset,offset1;  
    unsigned short i;

    if((Read\_W5500\_SOCK\_1Byte(s,Sn\_MR)&0x0f) != SOCK\_UDP)  
    {          
        Write\_W5500\_SOCK\_4Byte(s, Sn\_DIPR, UDP\_DIPR);           
        Write\_W5500\_SOCK\_2Byte(s, Sn\_DPORTR, UDP\_DPORT[0]\*256+UDP\_DPORT[1]);                  
    }

    offset=Read\_W5500\_SOCK\_2Byte(s,Sn\_TX\_WR);  
    offset1=offset;  
    offset&=(S\_TX\_SIZE-1);

    W5500\_SCS=0;

    SPI\_Send\_Short(offset);  
    SPI\_Send\_Byte(VDM|RWB\_WRITE|(s\*0x20+0x10));

    if((offset+size)<S\_TX\_SIZE)  
    {  
        for(i=0;i<size;i++)  
        {  
            SPI\_Send\_Byte(\*dat\_ptr++);          
        }  
    }  
    else  
    {  
        offset=S\_TX\_SIZE-offset;  
        for(i=0;i<offset;i++)  
        {  
            SPI\_Send\_Byte(\*dat\_ptr++);  
        }  
        W5500\_SCS=1;

        W5500\_SCS=0;

        SPI\_Send\_Short(0x00);  
        SPI\_Send\_Byte(VDM|RWB\_WRITE|(s\*0x20+0x10));

        for(;i<size;i++)  
        {  
            SPI\_Send\_Byte(\*dat\_ptr++);  
        }  
    }  
    W5500\_SCS=1;

    offset1+=size;  
    Write\_W5500\_SOCK\_2Byte(s, Sn\_TX\_WR, offset1);  
    Write\_W5500\_SOCK\_1Byte(s, Sn\_CR, SEND);                  
}

void W5500\_Hardware\_Reset(void)  
{  
    W5500\_RST=0;  
    Delay(200);  
    W5500\_RST=1;  
    Delay(200);  
    while((Read\_W5500\_1Byte(PHYCFGR)&LINK)==0);  
}

void W5500\_Init(void)  
{  
    unsigned char i=0;

    Write\_W5500\_1Byte(MR, RST);  
    Delay(10);

    Write\_W5500\_nByte(GAR, Gateway\_IP, 4);

    Write\_W5500\_nByte(SUBR,Sub\_Mask,4);

    Write\_W5500\_nByte(SHAR,Phy\_Addr,6);

    Write\_W5500\_nByte(SIPR,IP\_Addr,4);        

    for(i=0;i<8;i++)  
    {  
        Write\_W5500\_SOCK\_1Byte(i,Sn\_RXBUF\_SIZE, 0x02);//Socket Rx memory size=2k  
        Write\_W5500\_SOCK\_1Byte(i,Sn\_TXBUF\_SIZE, 0x02);//Socket Tx mempry size=2k  
    }

    Write\_W5500\_2Byte(RTR, 0x07d0);

    Write\_W5500\_1Byte(RCR,8);  
}

unsigned char Detect\_Gateway(void)  
{  
    unsigned char ip\_adde[4];  
    ip\_adde[0]=IP\_Addr[0]+1;  
    ip\_adde[1]=IP\_Addr[1]+1;  
    ip\_adde[2]=IP\_Addr[2]+1;  
    ip\_adde[3]=IP\_Addr[3]+1;

    Write\_W5500\_SOCK\_4Byte(0,Sn\_DIPR,ip\_adde);  
    Write\_W5500\_SOCK\_1Byte(0,Sn\_MR,MR\_TCP);  
    Write\_W5500\_SOCK\_1Byte(0,Sn\_CR,OPEN);      
    Delay(5);       
      
    if(Read\_W5500\_SOCK\_1Byte(0,Sn\_SR) != SOCK\_INIT)  
    {  
        Write\_W5500\_SOCK\_1Byte(0,Sn\_CR,CLOSE);  
        return FALSE;  
    }

    Write\_W5500\_SOCK\_1Byte(0,Sn\_CR,CONNECT);

    do  
    {  
        unsigned char j=0;  
        j=Read\_W5500\_SOCK\_1Byte(0,Sn\_IR);  
        if(j!=0)  
        Write\_W5500\_SOCK\_1Byte(0,Sn\_IR,j);  
        Delay(5);  
        if((j&IR\_TIMEOUT) == IR\_TIMEOUT)  
        {  
            return FALSE;      
        }  
        else if(Read\_W5500\_SOCK\_1Byte(0,Sn\_DHAR) != 0xff)  
        {  
            Write\_W5500\_SOCK\_1Byte(0,Sn\_CR,CLOSE);  
            return TRUE;                              
        }  
    }while(1);  
}

void Socket\_Init(SOCKET s)  
{      
    Write\_W5500\_SOCK\_2Byte(0, Sn\_MSSR, 30);  
      
    switch(s)  
    {  
        case 0:  
            Write\_W5500\_SOCK\_2Byte(0, Sn\_PORT, S0\_Port[0]\*256+S0\_Port[1]);  
            Write\_W5500\_SOCK\_2Byte(0, Sn\_DPORTR, S0\_DPort[0]\*256+S0\_DPort[1]);  
            Write\_W5500\_SOCK\_4Byte(0, Sn\_DIPR, S0\_DIP);              
              
            break;

        case 1:  
            break;

        case 2:  
            break;

        case 3:  
            break;

        case 4:  
            break;

        case 5:  
            break;

        case 6:  
            break;

        case 7:  
            break;

        default:  
            break;  
    }  
}

unsigned char Socket\_Connect(SOCKET s)  
{  
    Write\_W5500\_SOCK\_1Byte(s,Sn\_MR,MR\_TCP);  
    Write\_W5500\_SOCK\_1Byte(s,Sn\_CR,OPEN);  
    Delay(5);//晊奀5ms  
    if(Read\_W5500\_SOCK\_1Byte(s,Sn\_SR)!=SOCK\_INIT)  
    {  
        Write\_W5500\_SOCK\_1Byte(s,Sn\_CR,CLOSE);  
        return FALSE;  
    }  
    Write\_W5500\_SOCK\_1Byte(s,Sn\_CR,CONNECT);  
    return TRUE;  
}

unsigned char Socket\_Listen(SOCKET s)  
{  
    Write\_W5500\_SOCK\_1Byte(s,Sn\_MR,MR\_TCP);   
    Write\_W5500\_SOCK\_1Byte(s,Sn\_CR,OPEN);      
    Delay(5);//晊奀5ms  
    if(Read\_W5500\_SOCK\_1Byte(s,Sn\_SR)!=SOCK\_INIT)  
    {  
        Write\_W5500\_SOCK\_1Byte(s,Sn\_CR,CLOSE);  
        return FALSE;  
    }      
    Write\_W5500\_SOCK\_1Byte(s,Sn\_CR,LISTEN);      
    Delay(5);  
    if(Read\_W5500\_SOCK\_1Byte(s,Sn\_SR)!=SOCK\_LISTEN)  
    {  
        Write\_W5500\_SOCK\_1Byte(s,Sn\_CR,CLOSE);  
        return FALSE;  
    }

    return TRUE;

}

unsigned char Socket\_UDP(SOCKET s)  
{  
    Write\_W5500\_SOCK\_1Byte(s,Sn\_MR,MR\_UDP);  
    Write\_W5500\_SOCK\_1Byte(s,Sn\_CR,OPEN);  
    Delay(5);  
    if(Read\_W5500\_SOCK\_1Byte(s,Sn\_SR)!=SOCK\_UDP)  
    {  
        Write\_W5500\_SOCK\_1Byte(s,Sn\_CR,CLOSE);  
        return FALSE;  
    }  
    else  
        return TRUE;

}

void W5500\_Interrupt\_Process(void)  
{  
    unsigned char i,j;

IntDispose:

    i=Read\_W5500\_1Byte(SIR);      
    if((i & S0\_INT) == S0\_INT)  
    {  
        j=Read\_W5500\_SOCK\_1Byte(0,Sn\_IR);  
        Write\_W5500\_SOCK\_1Byte(0,Sn\_IR,j);  
        if(j&IR\_CON)   
        {  
            S0\_State|=S\_CONN;  
        }  
        if(j&IR\_DISCON)  
        {  
            Write\_W5500\_SOCK\_1Byte(0,Sn\_CR,CLOSE);  
            Socket\_Init(0);          
            S0\_State=0;  
        }  
        if(j&IR\_SEND\_OK)   
        {  
            S0\_Data|=S\_TRANSMITOK;   
        }  
        if(j&IR\_RECV)  
        {  
            S0\_Data|=S\_RECEIVE;  
        }  
        if(j&IR\_TIMEOUT)  
        {  
            Write\_W5500\_SOCK\_1Byte(0,Sn\_CR,CLOSE);          
            S0\_State=0;  
        }  
    }

    if(Read\_W5500\_1Byte(SIR) != 0)   
        goto IntDispose;  
}

////////////////////////////////////////////////////////////////

//---W5500.h---//

#ifndef    \_W5500\_H\_  
#define    \_W5500\_H\_

#include <reg51.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Common Register \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
#define MR        0x0000  
    #define RST        0x80  
    #define WOL        0x20  
    #define PB        0x10  
    #define PPP        0x08  
    #define FARP    0x02

#define GAR        0x0001  
#define SUBR    0x0005  
#define SHAR    0x0009  
#define SIPR    0x000f

#define INTLEVEL    0x0013  
#define IR        0x0015  
    #define CONFLICT    0x80  
    #define UNREACH        0x40  
    #define PPPOE        0x20  
    #define MP            0x10

#define IMR        0x0016  
    #define IM\_IR7        0x80  
    #define IM\_IR6        0x40  
    #define IM\_IR5        0x20  
    #define IM\_IR4        0x10

#define SIR        0x0017  
    #define S7\_INT        0x80  
    #define S6\_INT        0x40  
    #define S5\_INT        0x20  
    #define S4\_INT        0x10  
    #define S3\_INT        0x08  
    #define S2\_INT        0x04  
    #define S1\_INT        0x02  
    #define S0\_INT        0x01

#define SIMR    0x0018  
    #define S7\_IMR        0x80  
    #define S6\_IMR        0x40  
    #define S5\_IMR        0x20  
    #define S4\_IMR        0x10  
    #define S3\_IMR        0x08  
    #define S2\_IMR        0x04  
    #define S1\_IMR        0x02  
    #define S0\_IMR        0x01

#define RTR        0x0019  
#define RCR        0x001b

#define PTIMER    0x001c  
#define PMAGIC    0x001d  
#define PHA        0x001e  
#define PSID    0x0024  
#define PMRU    0x0026

#define UIPR    0x0028  
#define UPORT    0x002c

#define PHYCFGR    0x002e  
    #define RST\_PHY        0x80  
    #define OPMODE        0x40  
    #define DPX            0x04  
    #define SPD            0x02  
    #define LINK        0x01

#define VERR    0x0039

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Socket Register \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
#define Sn\_MR        0x0000  
    #define MULTI\_MFEN        0x80  
    #define BCASTB            0x40  
    #define    ND\_MC\_MMB        0x20  
    #define UCASTB\_MIP6B    0x10  
    #define MR\_CLOSE        0x00  
    #define MR\_TCP        0x01  
    #define MR\_UDP        0x02  
    #define MR\_MACRAW        0x04

#define Sn\_CR        0x0001  
    #define OPEN        0x01  
    #define LISTEN        0x02  
    #define CONNECT        0x04  
    #define DISCON        0x08  
    #define CLOSE        0x10  
    #define SEND        0x20  
    #define SEND\_MAC    0x21  
    #define SEND\_KEEP    0x22  
    #define RECV        0x40

#define Sn\_IR        0x0002  
    #define IR\_SEND\_OK        0x10  
    #define IR\_TIMEOUT        0x08  
    #define IR\_RECV            0x04  
    #define IR\_DISCON        0x02  
    #define IR\_CON            0x01

#define Sn\_SR        0x0003  
    #define SOCK\_CLOSED        0x00  
    #define SOCK\_INIT        0x13  
    #define SOCK\_LISTEN        0x14  
    #define SOCK\_ESTABLISHED    0x17  
    #define SOCK\_CLOSE\_WAIT        0x1c  
    #define SOCK\_UDP        0x22  
    #define SOCK\_MACRAW        0x02

    #define SOCK\_SYNSEND    0x15  
    #define SOCK\_SYNRECV    0x16  
    #define SOCK\_FIN\_WAI    0x18  
    #define SOCK\_CLOSING    0x1a  
    #define SOCK\_TIME\_WAIT    0x1b  
    #define SOCK\_LAST\_ACK    0x1d

#define Sn\_PORT        0x0004  
#define Sn\_DHAR           0x0006  
#define Sn\_DIPR        0x000c  
#define Sn\_DPORTR    0x0010

#define Sn\_MSSR        0x0012  
#define Sn\_TOS        0x0015  
#define Sn\_TTL        0x0016

#define Sn\_RXBUF\_SIZE    0x001e  
#define Sn\_TXBUF\_SIZE    0x001f  
#define Sn\_TX\_FSR    0x0020  
#define Sn\_TX\_RD    0x0022  
#define Sn\_TX\_WR    0x0024  
#define Sn\_RX\_RSR    0x0026  
#define Sn\_RX\_RD    0x0028  
#define Sn\_RX\_WR    0x002a

#define Sn\_IMR        0x002c  
    #define IMR\_SENDOK    0x10  
    #define IMR\_TIMEOUT    0x08  
    #define IMR\_RECV    0x04  
    #define IMR\_DISCON    0x02  
    #define IMR\_CON        0x01

#define Sn\_FRAG        0x002d  
#define Sn\_KPALVTR    0x002f

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SPI Control Byte \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
/\* Operation mode bits \*/  
#define VDM        0x00  
#define FDM1    0x01  
#define    FDM2    0x02  
#define FDM4    0x03

/\* Read\_Write control bit \*/  
#define RWB\_READ    0x00  
#define RWB\_WRITE    0x04

/\* Block select bits \*/  
#define COMMON\_R    0x00

/\* Socket 0 \*/  
#define S0\_REG        0x08  
#define S0\_TX\_BUF    0x10  
#define S0\_RX\_BUF    0x18

/\* Socket 1 \*/  
#define S1\_REG        0x28  
#define S1\_TX\_BUF    0x30  
#define S1\_RX\_BUF    0x38

/\* Socket 2 \*/  
#define S2\_REG        0x48  
#define S2\_TX\_BUF    0x50  
#define S2\_RX\_BUF    0x58

/\* Socket 3 \*/  
#define S3\_REG        0x68  
#define S3\_TX\_BUF    0x70  
#define S3\_RX\_BUF    0x78

/\* Socket 4 \*/  
#define S4\_REG        0x88  
#define S4\_TX\_BUF    0x90  
#define S4\_RX\_BUF    0x98

/\* Socket 5 \*/  
#define S5\_REG        0xa8  
#define S5\_TX\_BUF    0xb0  
#define S5\_RX\_BUF    0xb8

/\* Socket 6 \*/  
#define S6\_REG        0xc8  
#define S6\_TX\_BUF    0xd0  
#define S6\_RX\_BUF    0xd8

/\* Socket 7 \*/  
#define S7\_REG        0xe8  
#define S7\_TX\_BUF    0xf0  
#define S7\_RX\_BUF    0xf8

#define TRUE    0xff  
#define FALSE    0x00

#define S\_RX\_SIZE    2048  
#define S\_TX\_SIZE    2048  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*----- W5500 GPIO\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
sbit W5500\_RST = P3^2;      
sbit W5500\_INT = P3^3;      
sbit W5500\_SCS = P3^6;          
sbit W5500\_SCLK = P3^7;          
sbit W5500\_MISO = P3^4;       
sbit W5500\_MOSI = P3^5;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*----- IP SET -----\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
extern unsigned char Gateway\_IP[4];       
extern unsigned char Sub\_Mask[4];      
extern unsigned char Phy\_Addr[6];    //(MAC)  
extern unsigned char IP\_Addr[4];

extern unsigned char S0\_Port[2];    //PORT(5000)   
extern unsigned char S0\_DIP[4];           
extern unsigned char S0\_DPort[2];    //PORT(6000)

extern unsigned char UDP\_DIPR[4];      
extern unsigned char UDP\_DPORT[2];

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*----- Mode-----\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
extern unsigned char S0\_Mode;    //0:TCP Server,1:TCP Client,2:UDP  
#define TCP\_SERVER        0x00    //TCP Server  
#define TCP\_CLIENT        0x01    //TCP Client  
#define UDP\_MODE        0x02    //UDP

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*----------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
extern unsigned char S0\_State;       
#define S\_INIT            0x01      
#define S\_CONN            0x02

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*----------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
extern unsigned char S0\_Data;          
#define S\_RECEIVE        0x01          
#define S\_TRANSMITOK    0x02          
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*----- -----\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
extern unsigned char Rx\_Buffer[30];  
extern unsigned char Tx\_Buffer[30];

extern unsigned char W5500\_Interrupt;      
typedef unsigned char SOCKET;

extern void Delay(unsigned int d);  
extern void W5500\_GPIO\_Configuration(void);  
extern void W5500\_NVIC\_Configuration(void);  
extern void SPI\_Configuration(void);//W5500 SPI (STM32 SPI1)  
extern void W5500\_Hardware\_Reset(void);  
extern void W5500\_Init(void);  
extern unsigned char Detect\_Gateway(void);  
extern void Socket\_Init(SOCKET s);  
extern unsigned char Socket\_Connect(SOCKET s);  
extern unsigned char Socket\_Listen(SOCKET s);  
extern unsigned char Socket\_UDP(SOCKET s);  
extern unsigned short Read\_SOCK\_Data\_Buffer(SOCKET s, unsigned char \*dat\_ptr);  
extern void Write\_SOCK\_Data\_Buffer(SOCKET s, unsigned char \*dat\_ptr, unsigned short size);   
extern void W5500\_Interrupt\_Process(void);

#endif