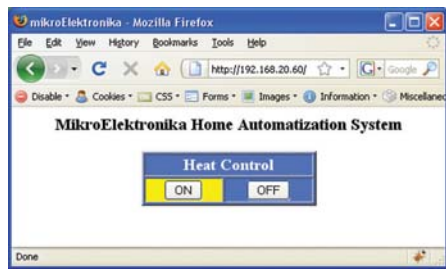


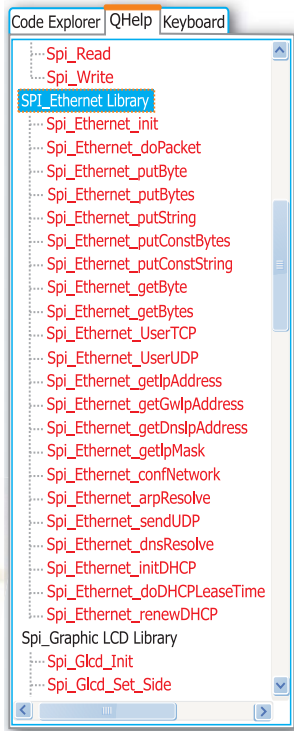
Schema 1. Aansluiten van de Seriële Ethernet-module op een PIC18F4520

Het besturen van een of ander huishoudelijk apparaat bestaat uit het in de webbrowser invoeren van het IP-adres van het controlesysteem en het specificeren van de gewenste opdrachten. Uiteraard is het mogelijk meer dan een microcontrollerpen aan te sturen, zodat u ook een groot aantal huishoudelijke apparaten als een compleet automatiseringssysteem kunt aansturen.



De schermafdruk illustreert de door de webbrowser gevoerde webpagina als het IP-adres van het controlesysteem in ons voorbeeld wordt ingevoerd. Op de ON- of OFF-knop klikken schakelt de LED in en uit en simulteert op die manier het verwarming-sregelsysteem.

Onderstaande lijst van kant en klare functies is opgenomen in de SPI Ethernet Library. Deze bibliotheek maakt deel uit van de *microPASCAL for PIC* compiler.



Spi_Ethernet_Init()*	ENC28J60-controller initialiseren
Spi_Ethernet_Enable()	Netwerkverkeer inschakelen
Spi_Ethernet_Disable()	Netwerkverkeer uitschakelen
Spi_Ethernet_doPacket()*	Ontvangen pakket verwerken
Spi_Ethernet_putByte()	Een byte opslaan
Spi_Ethernet_putBytes()	Bytes opslaan
Spi_Ethernet_putConstBytes()	Bytes continu opslaan
Spi_Ethernet_putString()*	Tekenreeks opslaan
Spi_Ethernet_putConstString()*	Tekenreeks continu opslaan
Spi_Ethernet_getByte()*	Een byte ophalen
Spi_Ethernet_getBytes()	Bytes ophalen
Spi_Ethernet_UserTCP()*	TCP-code afhandelen
Spi_Ethernet_UserUDP()	UDP-code afhandelen
Spi_Ethernet_getIpAddress()	IP-adres ophalen
Spi_Ethernet_getGwIpAddress()	Gateway-adres ophalen
Spi_Ethernet_getDnsIpAddress()	DNS-adres ophalen
Spi_Ethernet_getIpMask()	IP-masker ophalen
Spi_Ethernet_confNetwork()*	Netwerkparameters instellen
Spi_Ethernet_arpResolve()	ARP-verzoek verzenden
Spi_Ethernet_sendUDP()	UDP-pakket verzenden
Spi_Ethernet_dnsResolve()	DNS-verzoek verzenden
Spi_Ethernet_initDHCP()	DHCP-verzoek verzenden
Spi_Ethernet_doDHCPLeaseTime()	Verwerk lease time
Spi_Ethernet_renewDHCP()	Verzoek tot DHCP-vernieuwing

In het programma gebruikte *.SPI Ethernet Library-functies.

Andere in het programma gebruikte *microPASCAL for PIC*-functies.

Spi_Init()	Microcontroller SPI-module initialiseren
memcpy()	Microcontroller RAM-geheugenplaatsen kopiëren
memcmp()	Microcontroller RAM-geheugenplaatsen vergelijken

Voorbeeld 1: Programma om sturing via Ethernet te demonstreren (two files)

```

program enc_ethernet;
uses home_auto_utils;

RAM variables
var myMacAddr : array[6] of byte ; // my MAC address
    myIpAddr : array[4] of byte ; // my IP address
    gwIpAddr : array[4] of byte ; // gateway (router) IP address
    ipMask : array[4] of byte ; // network mask (for example: 255.255.255.0)
    dnsIpAddr : array[4] of byte ; // DNS server IP address

begin
ADCON1 := 0x0F; // no analog inputs
CMCON := 0x07; // turn off comparators

PORTB0 := 0; // set PORTB0 as output (relé control pin)

indexPage :=
<html><head><title>mikroElektronika</title></head><body>+
<h3 align=center>MikroElektronika Home Automatization System</h3>+
<form name= 'input' action= '/' method= 'get' >+
<table align=center width=200 bgcolor=#497AE2 border=2 <tr>+
<td align=center colspan=2><font size=4 color=white><b>Heat Control</b></td></tr>+
</table></tr><tr><td align=center bgcolor=#497AE2><input name= 'tst1' width=60 +
'type= 'submit' value= 'ON' ></td><td align=center bgcolor=#FFFF00>+
<input name= 'tst2' type= 'submit' value= 'OFF' ></td></tr></table>+
</form></body></html>;

myMacAddr[0]:=0x00; myMacAddr[1] := 0x14; myMacAddr[2] := 0xA5;
myMacAddr[3]:=0x76; myMacAddr[4] := 0x19; myMacAddr[5] := 0x3F;

ipMask[0]:=255; ipMask[1]:=255; ipMask[2]:=255; ipMask[3]:=0;
myIpAddr[0]:=192; myIpAddr[1]:=168; myIpAddr[2]:=20; myIpAddr[3]:=60;
gwIpAddr[0]:=192; gwIpAddr[1]:=168; gwIpAddr[2]:=20; gwIpAddr[3]:=60;
dnsIpAddr[0]:=192; dnsIpAddr[1]:=168; dnsIpAddr[2]:=20; dnsIpAddr[3]:=6;

// starts ENC28J60 with: reset bit on PORTC.F0, CS bit on PORTC.F1
Spi_Init(); // my MAC & IP address, full duplex
// full duplex CRC + MAC Unicast + MAC Broadcast filtering
Spi_Ethernet_Init(PORTC.0, PORTC.1, myMacAddr, myIpAddr, Spi_Ethernet_FULLDUPLEX);

// dhcp will not be used here, so use preconfigured addresses
Spi_Ethernet_confNetwork(ipMask, gwIpAddr, dnsIpAddr);

while true do
Spi_Ethernet_doPacket(); // process incoming Ethernet packets
end.

unit home_auto_utils;
const httpHeader : string[30] = 'HTTP/1.1 200 OK'+#10+'Content-type: '; // HTTP header
const httpMimeTypeHTML : string[13] = 'text/html'+#10+#10; // HTML MIME type
const httpMimeTypeScript : string[14] = 'text/plain'+#10+#10; // TEXT MIME type

// default html page
var indexPage : string[523];
var getReq : array[20] of byte; // HTTP request buffer

implementation
function SPI_Ethernet_UserTCP(var remoteHost : array[4] of byte;
remotePort, localPort, reqLength : word) : word;
var len : word; // my reply length
tmp : string[10];
begin
if(localPort < 80) then
result := 0;
end;
// I listen only to web request on port 80

// get 10 first bytes only of the request, the rest does not matter here
for len := 0 to 14 do getReq[len] := SPI_Ethernet_getByte();
getReq[len] := 0;

tmp := 'GET /';
if(memcmp(getReq, @tmp, 5) < 0) then // only GET method
result := 0;
end;

tmp := 'ON';
if(memcmp(getReq+11, @tmp, 2) = 0) then // do we have ON command
PORTB0 := 1; // set PORTB bit0
else
begin
tmp := 'OFF';
if(memcmp(getReq+11, @tmp, 3) = 0) then // do we have OFF command
PORTB0 := 0; // clear PORTB bit0
end;

if (PORTB0) then
begin
tmp := '#FFFF00'; memcpy(@indexPage+340, @tmp, 6); // highlight (yellow) ON
tmp := '#497AE2'; memcpy(@indexPage+431, @tmp, 6); // clear OFF
end;
else
begin
tmp := '#497AE2'; memcpy(@indexPage+340, @tmp, 6); // clear ON
tmp := '#FFFF00'; memcpy(@indexPage+431, @tmp, 6); // highlight (yellow) OFF
end;

len := Spi_Ethernet_putConstString(@httpHeader); // HTTP header
len := len + Spi_Ethernet_putConstString(@httpMimeTypeHTML); // with HTML MIME type
len := len + Spi_Ethernet_putString(@indexPage); // HTML page first part
result := len; // return to the library with the number of bytes to transmit
end;

function SPI_Ethernet_UserUDP(var remoteHost : array[4] of byte;
remotePort, destPort, reqLength : word) : word;
begin
result := 0; // back to the library with the length of the UDP reply
end.

```

Opn.: De voor dit voorbeeld in C, Basic en Pascal voor PIC® microcontrollers geschreven code staan, evenals de voor dsPIC® en AVR® microcontrollers geschreven programma's, op onze website: www.mikroe.com/en/article/.

