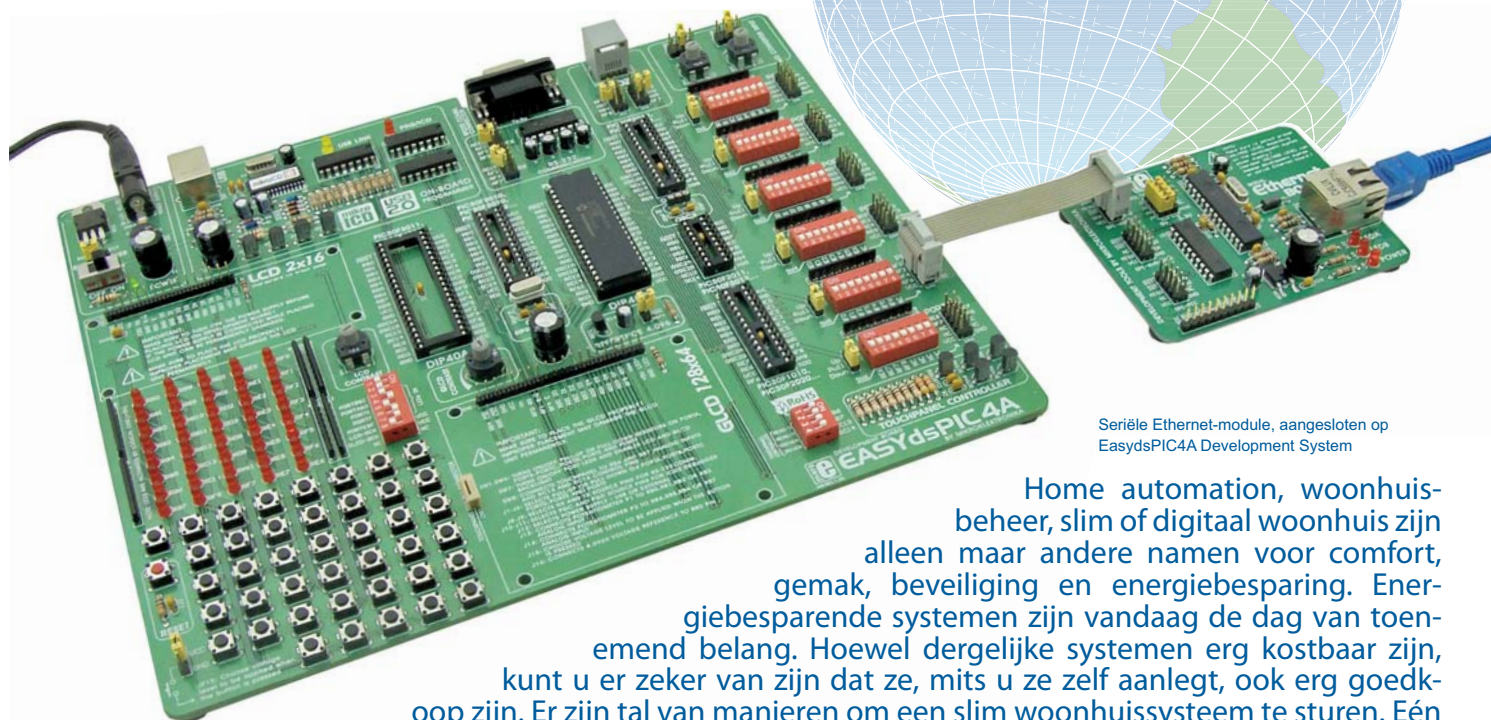


# OKÉ. Nu hebt u ... ETHERNET nodig



Seriële Ethernet-module, aangesloten op EasydsPIC4A Development System

Home automation, woonhuis-beheer, slim of digitaal woonhuis zijn alleen maar andere namen voor comfort, gemak, beveiliging en energiebesparing. Energiebesparende systemen zijn vandaag de dag van toenemend belang. Hoewel dergelijke systemen erg kostbaar zijn, kunt u er zeker van zijn dat ze, mits u ze zelf aanlegt, ook erg goedkoop zijn. Er zijn tal van manieren om een slim woonhuissysteem te sturen. Eén ervan is via Ethernet.

Srdjan Tomic  
MikroElektronika – Software Department

Alles wat u nodig hebt, is een dsPIC30F4013 microcontroller en een ENC28J60 seriële Ethernet-chip. Deze chip is trouwens ook een prima oplossing voor andere microcontroller families, zoals AVR, dsPIC enzovoort. De aansluiting op het Ethernet-netwerk komt tot stand via een CviLux CJCBA8HF1Y0 RJ-45 connector. De op de PORTD.0 aangesloten LED simuleert een huishoudelijk apparaat dat we willen regelen.

De *microPASCAL for dsPIC*-compiler bevat de SPI\_Ethernet-bibliotheek die het schrijven van een programma voor de microcontroller aanzienlijk vereenvoudigt. Met enkele routines uit deze bibliotheek kunt u een programma opstellen waarmee u de elektrische huishoudelijke apparaten in uw huis via een webbrowser bestuurt.

Dit programma moet de volgende bewerkingen uitvoeren:

- Stap 1.** Aanmaken van een html-pagina om de microcontroller aan te sturen. Deze wordt in de code geïmporteerd in de vorm van een tekenreeks.
- Stap 2.** Instellen van de door uw internet-

provider verstrekte IP-, DNS- en gateway-adressen en het subnetmasker.

De parameters van uw lokale netwerk kunnen bijvoorbeeld luiden:

- IP** : 192.168.20.60 (Controlesysteem-adres)
- DNS** : 192.168.20.1 (Domain Name System-adres)
- GATEWAY** : 192.168.20.6 (gateway-adres)
- SUBNET** : 255.255.255.0 (subnetmasker)

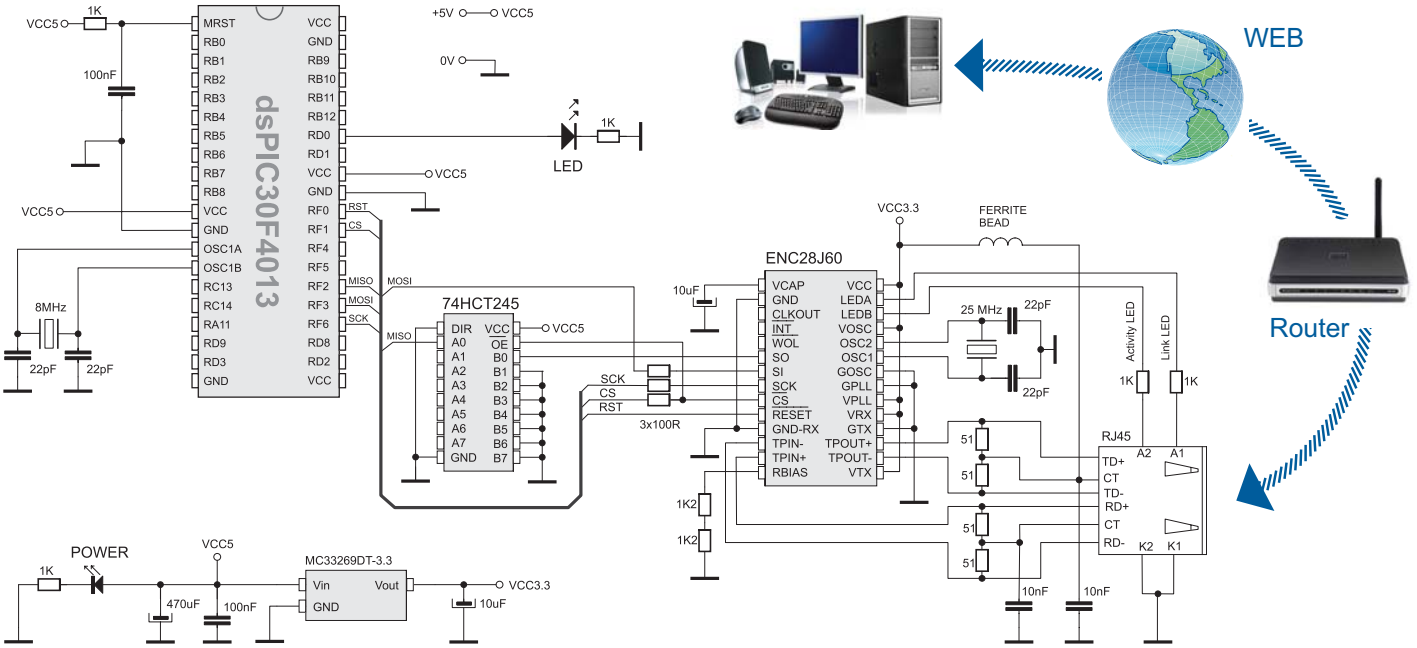
- Stap 3.** Uitschakelen van de analoge PORTD ingangen. De microcontroller-pen moet vrijgemaakt en als output geconfigureerd worden.
- Stap 4.** Initialiseren van de SPI-module van de dsPIC30F4013 microcontroller.
- Stap 5.** Initialiseren van de ENC28J60-chip op de module.
- Stap 6.** Schrijven van de code binnen de functie Spi\_Ethernet-userTCP die, na ontvangst van een opdracht via de webbrowser, de op de PORTD.0 aangesloten LED in- of uitschakelt.
- Stap 7.** De ontvangen gegevens in een eindeloze lus inlezen.

Het belangrijkste onderdeel van het programma is de functie Spi\_Ethernet-UserTCP die alle ontvangen opdrachten verwerkt. Ontvangt de webbrowser een "GET"-verzoek dat door uw computer naar het IP-adres van het besturingssysteem is gezonden, dan reageert de microcontroller door een in zijn geheugen opgeslagen webpagina af te geven. Deze pagina wordt dan door de browser automatisch afgebeeld op het computerscherm.

Wordt de opdracht "ON" ontvangen, dan wordt de op PORTD.0 aangesloten LED ingeschakeld. Op dezelfde manier wordt bij ontvangst van de opdracht "OFF", de LED uitgeschakeld. Sluit u een relais in plaats van een LED aan, dan kunt u een of ander huishoudelijk apparaat besturen, bijvoorbeeld de verlichting, de beveiliging, de verwarming enzovoort.

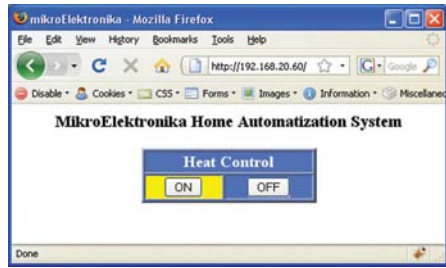


Figuur 1. MikroElektronika's Seriële Ethernet module met ENC28J60-chip



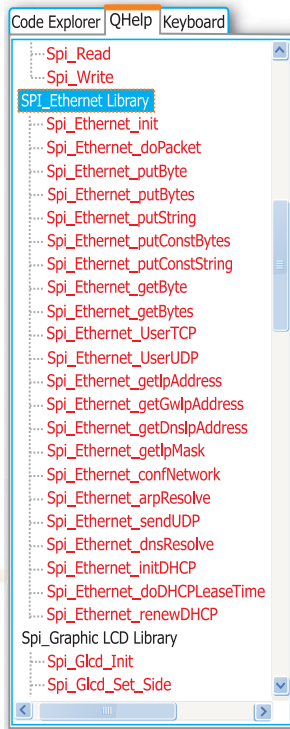
Schema 1. Aansluiten van de Seriele Ethernet-module op een dsPIC30F4013

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 simuleert 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 dsPIC* compiler.



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

In het programma gebruikte \*.SPI Ethernet Library-functies. **Andere in het programma gebruikte microPASCAL for dsPIC-functies.**

<b>Spi_Init()</b>	<b>Microcontroller SPI-module initialiseren</b>
<b>memcpy()</b>	<b>Microcontroller RAM-geheugenplaatsen kopiëren</b>
<b>memcmp()</b>	<b>Microcontroller RAM-geheugenplaatsen vergelijken</b>

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

begin
  ADRCFG := 0xFFFF; // no analog inputs
  PORTD.0 := 0;
  TRISD.0 := 0; // set PORTD.B0 as output (rele control pin)

  indexPage :=
  <html><head><title>mikroElektronika</title></head><body>+
  <h3 align=center>MikroElektronika Home Automatization System</h3>+
  <table align=center width=200 bgcolor=#974E2 border=2><tr>+
  <td align=center colspan=2><font size=4 color=white><b>Heat Control</b></font>+
  </td></tr><tr><td align=center bgcolor=#4974E2><input name='tst1' width=60 +
  <type='submit' value='ON' ></td><td align=center bgcolor=#FFFFFF>+
  <input name='tst2' type='submit' value='OFF' ></td></tr></table>+
  </body></html>;

  myMacAddr[0]:=0x00; myMacAddr[1] := 0x14; myMacAddr[2] := 0xA5;
  myMacAddr[3]:=0x76; myMacAddr[4] := 0x19; myMacAddr[5] := 0x3F;
  myIpAddr[0]:=192; myIpAddr[1]:=168; myIpAddr[2]:=20; myIpAddr[3]:=60;

  // starts ENC28J60 with: reset bit on PORTFE.F0, CS bit on PORTFE.F1
  // my MAC & IP address, full duplex
  Spi_Init();
  Spi_Ethernet_Init(PORTF.0, PORTF.1,
  myMacAddr, myIpAddr, Spi_Ethernet_FULLDUPLEX);

  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 httpMimeTypesHTML : string[13] = 'text/html'+#10+#10; // HTML MIME type
const httpMimeTypesScript : string[14] = 'text/plain'+#10+#10; // TEXT MIME type

// default html page
var indexPage : string[5231];
var getRequest : array[20] of byte; // HTTP request buffer

implementation

function putConstString(const s : ^byte) : word;
begin
  result := 0;
  while(s^ <> 0) do
    Spi_Ethernet_putByte(s^); s := s + 1; result := result + 1;
  end;
end;

function putString(var s : array[100] of byte) : word;
begin
  result := 0;
  while(s[result] <> 0) do
    Spi_Ethernet_putByte(s[result]); result := result + 1;
  end;
end;

function SPI_Ethernet_UserTCP(var remoteHost : array[4] of byte;
remotePort, localPort, reqLength : word) : word;
var len : word;
tmp : string[10];
begin
  if(localPort < 80) then // I listen only to web request on port 80
    begin
      result := 0;
      exit;
    end;
  // get 10 first bytes only of the request, the rest does not matter here
  for len := 0 to 14 do getRequest[len] := Spi_Ethernet_getByte();
  getRequest[len] := 0;

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

  tmp := 'ON';
  if(memcmp(getRequest+11, @tmp, 2) = 0) then // do we have ON command
    PORTD.0 := 1 // set PORTD bit0
  else
    tmp := 'OFF';
    if(memcmp(getRequest+11, @tmp, 3) = 0) then // do we have OFF command
      PORTD.0 := 0; // clear PORTD bit0
    end;
  Delay 1uS();
  if (PORTD.0) then
    begin
      tmp := '#FFFFFF0'; memcpy(@indexPage+340, @tmp, 6); // highlight (yellow) ON
      tmp := '#4974E2'; memcpy(@indexPage+431, @tmp, 6); // clear OFF
    end
  else
    begin
      tmp := '#4974E2'; memcpy(@indexPage+340, @tmp, 6); // clear ON
      tmp := '#FFFFFF00'; memcpy(@indexPage+431, @tmp, 6); // highlight (yellow) OFF
    end;

  len := putConstString(@httpHeader); // HTTP header
  len := len + putConstString(@httpMimeTypesHTML); // with HTML MIME type
  len := len + 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 dsPIC® microcontrollers geschreven code staan, evenals de voor PIC® en AVR® microcontrollers geschreven programma's, op onze website: [www.mikroe.com/en/article/](http://www.mikroe.com/en/article/).

