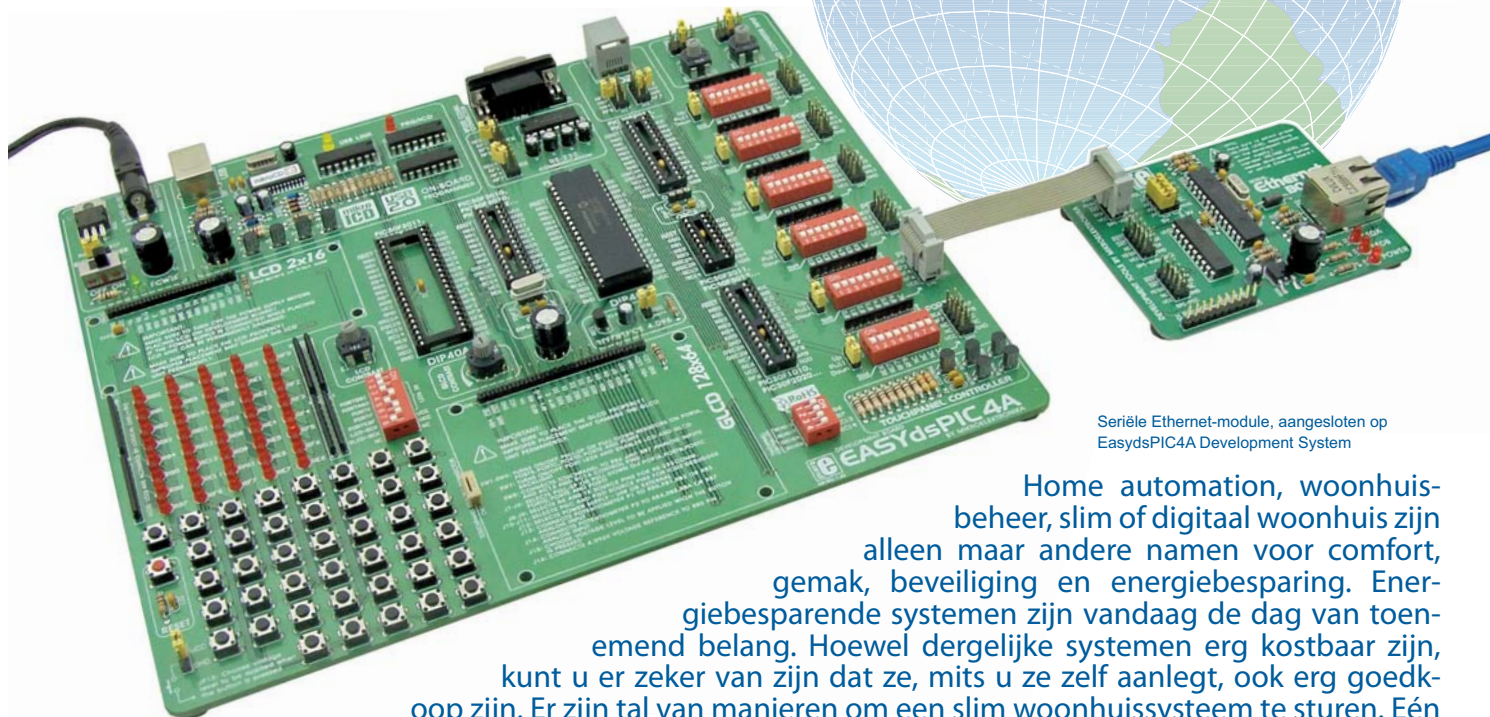


Oké. Nu hebt u ... ETHERNET nodig



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 *microC 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)

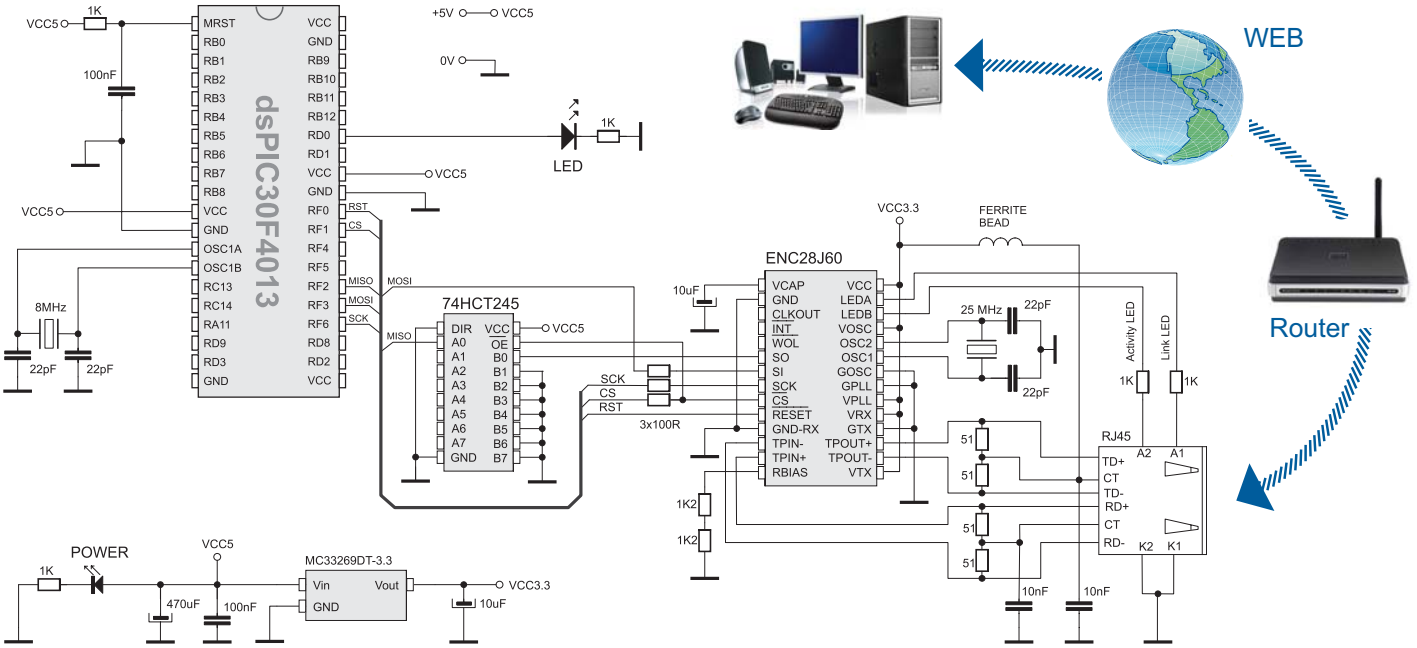
- Stap 3.** Uitschakelen van de analoge PORTD ingangen. De microcontroller-pen moet vrijgemaakt en als output geconfigureerd worden.
- Stap 4.** Initialiseren van de SPI-module van de dsPIC30F4013 microcontroller.
- Stap 5.** Initialiseren van de ENC28J60-chip op de module.
- Stap 6.** Schrijven van 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.

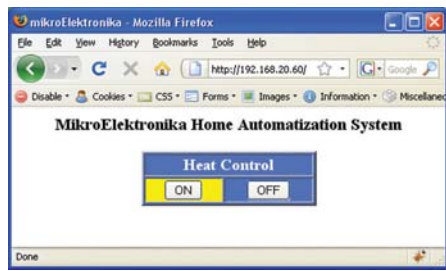


Figuur 1. MikroElektronika's Seriële Ethernet module met ENC28J60-chip



Schema 1. Aansluiten van de Seriële 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 geïmporteerde 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 verwarmingssysteem.

Voorbeeld 1: Programma om sturing via Ethernet te demonstreren

```

// duplex config flags
#define Spi_Ethernet_HALFDUPLEX 0x00 // half duplex
#define Spi_Ethernet_FULLDUPLEX 0x01 // full duplex

const char httpHeader[] = "HTTP/1.1 200 OK\r\nContent-type: "; // HTTP header
const char httpMimeTypeHTML[] = "text/html\r\n"; // HTML MIME type
const char httpMimeTypeScript[] = "text/plain\r\n"; // TEXT MIME type

// default html page
char indexPage[] =
"<html><head><title>mikroElektronika Home Automatization System</title></head><body>\r\n"
"<h3 align=center>MikroElektronika Home Automatization System</h3>\r\n"
"<form name='input' action='/' method='get'>\r\n"
"<table align=center width=200 bgcolor=#4974E2 border=2><tr>\r\n"
"<td align=center colspan=2><font size=4 color=white><b>Heat Control</b></font>\r\n"
"</td></tr><tr><td align=center bgcolor=#4974E2><input name='tst1' width=60\r\n"
" type='submit' value='ON'></td><td align=center bgcolor=#FFFFFF>\r\n"
"<input name='tst2' type='submit' value='OFF'></td></tr></table>\r\n"
"</form></body></html>";

// network parameters
char myMacAddr[6] = {0x00, 0x14, 0xA5, 0x76, 0x19, 0x3F}; // my MAC address
char myIpAddr[4] = {192, 168, 20, 60}; // my IP address
// end network parameters

unsigned char getRequest[20]; // HTTP request buffer

unsigned int putConstString(const char *s) {
  unsigned int ctr = 0;
  while(*s) Spi_Ethernet_putByte(*s++), ctr++;
  return(ctr);
}

unsigned int putString(char *s) {
  unsigned int ctr = 0;
  while(*s) Spi_Ethernet_putByte(*s++), ctr++;
  return(ctr);
}

unsigned int SPI_Ethernet_UserTCP(char *remoteHost, unsigned int remotePort,
  unsigned int localPort, unsigned int reqLength)
{
  unsigned int len; // my reply length
  if(localPort != 80) return(0); // 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; len < 15; len++) getRequest[len] = Spi_Ethernet_getByte();
  getRequest[len] = 0;

  if(memcmp(getRequest, "GET /", 5)) return(0); // only GET method

  if(!memcmp(getRequest+11, "ON", 2)) // do we have ON command
    PORTD.F0 = 1; // set PORTD bit 0
  else if(!memcmp(getRequest+11, "OFF", 3)) // do we have OFF command
    PORTD.F0 = 0; // clear PORTD bit 0

  Delay_1us();

  if (PORTD.F0)
  {
    memcpy(indexPage+340, "#FFFF", 6); // highlight (yellow) ON
    memcpy(indexPage+431, "#4974E2", 6); // clear OFF
  }
  else
  {
    memcpy(indexPage+340, "#4974E2", 6); // clear ON
    memcpy(indexPage+431, "#FFFF", 6); // highlight (yellow) OFF
  }

  len = putConstString(httpHeader); // HTTP header
  len += putConstString(httpMimeTypeHTML); // with HTML MIME type
  len += putString(indexPage); // HTML page first part
  return len; // return to the library with the number of bytes to transmit
}

unsigned int SPI_Ethernet_UserUDP(char *remoteHost, unsigned int remotePort,
  unsigned int destPort, unsigned int reqLength)
{
  return 0; // back to the library with the length of the UDP reply
}

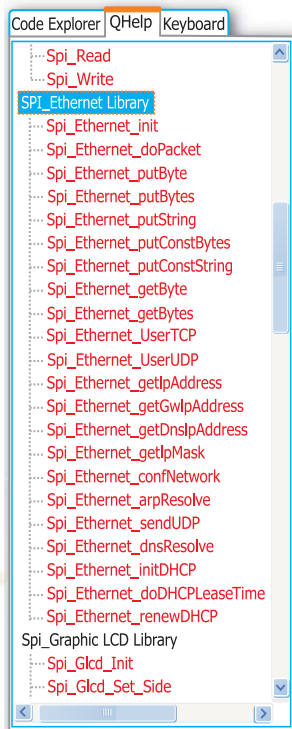
void main()
{
  ADPCFG |= 0xFFFF; // no analog inputs
  PORTD.F0 = 0;
  TRISD.F0 = 0; // set PORTD.B0 as output (rele control pin)

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

  while(1) { // do forever
    Spi_Ethernet_doPacket(); // process incoming Ethernet packets
  }
}

```

Onderstaande lijst van kant en klare functies is opgenomen in de SPI Ethernet Library. Deze bibliotheek maakt deel uit van de *microC for dsPIC* 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 microC for dsPIC-functies.

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

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/.

