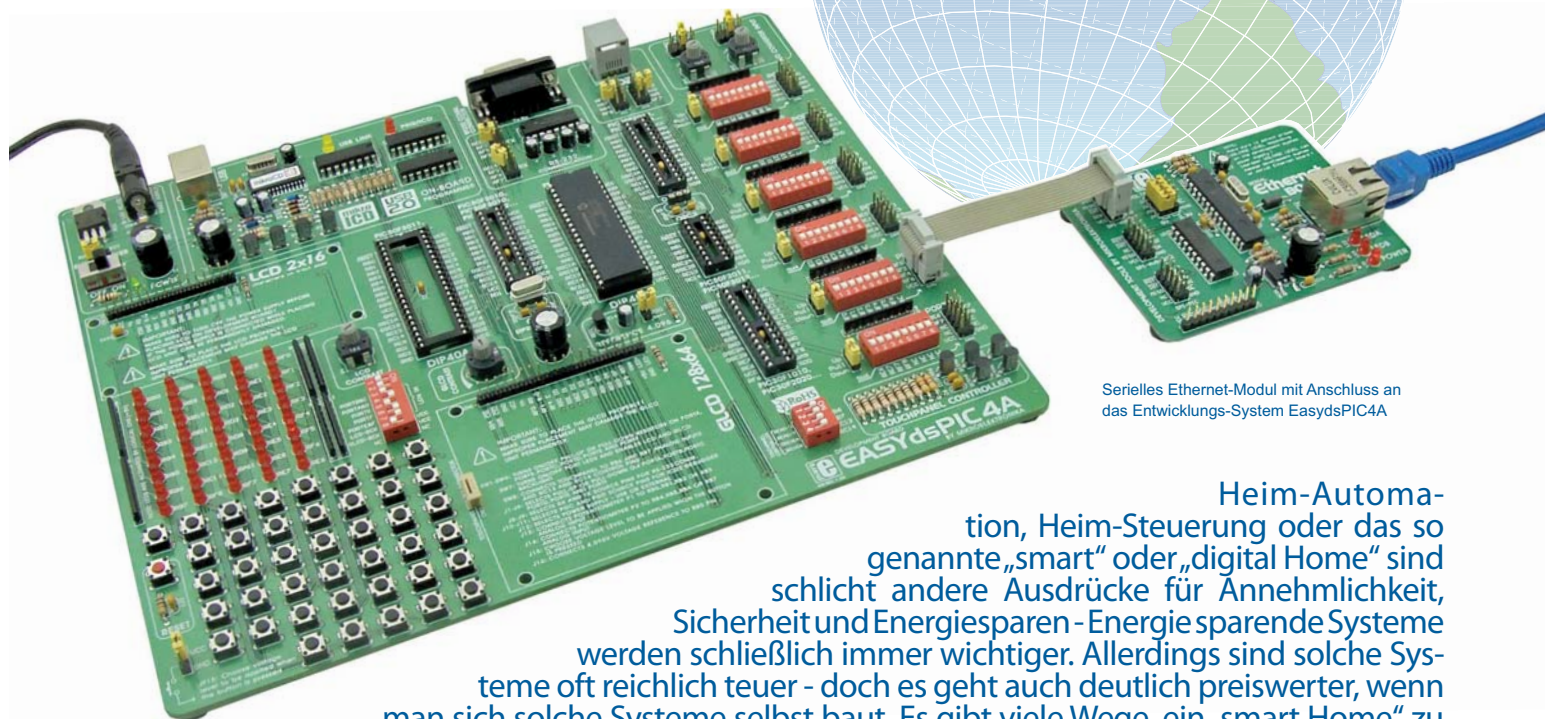


OK. Jetzt brauchen sie... ETHERNET



Serielles Ethernet-Modul mit Anschluss an das Entwicklungs-System EasydsPIC4A

Heim-Automation, Heim-Steuerung oder das sogenannte „smart“ oder „digital Home“ sind schlicht andere Ausdrücke für Annehmlichkeit, Sicherheit und Energiesparen - Energie sparende Systeme werden schließlich immer wichtiger. Allerdings sind solche Systeme oft reichlich teuer - doch es geht auch deutlich preiswerter, wenn man sich solche Systeme selbst baut. Es gibt viele Wege, ein „smart Home“ zu realisieren. Ein gut funktionierendes Verfahren ist es, Ethernet zu nutzen.

Von Srdjan Tomic
MikroElektronika - Software-Abteilung

Es wird lediglich ein dsPIC30F4013-Mikrocontroller und ein serieller Ethernet-Chip vom Typ ENC28J60 benötigt. Dieser Chip ist eine gute Ergänzung auch für andere Mikrocontroller-Familien wie dsPIC oder die Controller von Atmel etc. Für den Anschluss an das Ethernet-Netzwerk wird eine Steckverbindung vom Typ CviLux CJCBA8HF1Y0 (RJ-45) verwendet. Eine an den Pin PORTD.0 des Mikrocontrollers angeschlossene LED simuliert das zu steuernde Gerät.

Der Compiler *mikroC for dsPIC* enthält die Library *SPI_Ethernet*, welche das Schreiben von Software für den Mikrocontroller drastisch vereinfacht. Durch Verwendung nur weniger Routinen dieser Library ist es möglich, ein Programm zu erstellen, das es erlaubt, alle gesteuerten Geräte im Heim auf einfache Weise via Web-Browser fernzusteuern.

Hierzu sind folgende Schritte im Programm erforderlich:

Schritt 1. Erstelle eine HTML-Seite für den Mikrocontroller. Diese Seite wird dann als String importiert.

Schritt 2. Setzen der IP-, DNS- und Gateway-Adressen sowie der Subnetz-Maske entsprechend der Vorgaben des Internet-Providers.

Die lokalen Netzwerk-Parameter könnten zum Beispiel so aussehen:

IP : 192.168.20.60 (Adresse des Geräts)
DNS : 192.168.20.1 (Adresse des Domain-Name-Systems)
GATEWAY : 192.168.20.6 (Gateway-Adresse)
SUBNET : 255.255.255.0 (Subnetz-Maske)

Schritt 3. Deaktivieren der analogen Eingänge von PORTD. Der jeweilige Pin wird gelöscht und als Ausgang definiert.

Schritt 4. Initialisieren des SPI-Moduls des PIC18F4520-Mikrocontrollers.

Schritt 5. Initialisieren des seriellen Ethernet-Modul-Chips ENC28J60.

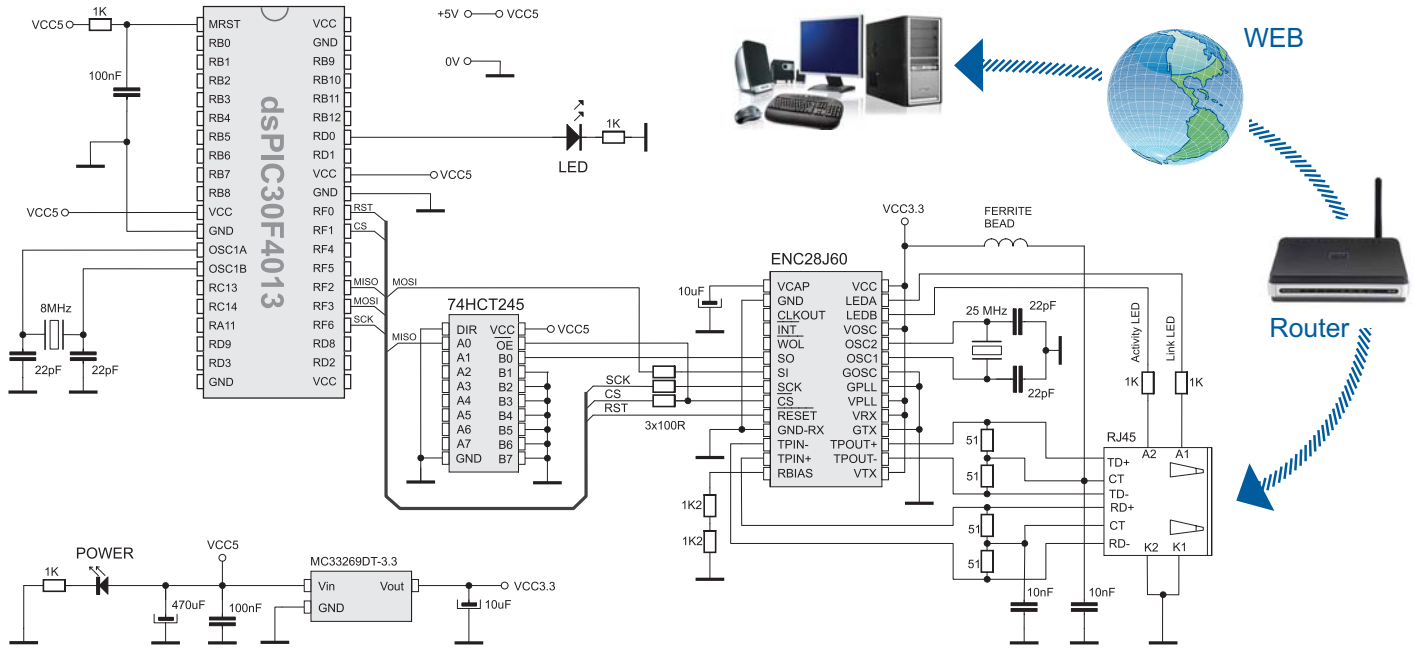
Schritt 6. Schreiben des Codes innerhalb der *Spi_Ethernet_userTCP*-Funktion, welcher nach Empfang eines Befehls via Web-Browser die an PORTD.0 angeschlossene LED ein bzw. ausschaltet.

Schritt 7. Lesen der zu empfangenden Daten in einer Endlos-Schleife.

Der wichtigste Teil des Programms ist die *Spi_Ethernet_userTCP*-Funktion, die empfangene Befehle ausführt. Nachdem eine „GET“-Anfrage des Web-Browsers empfangen wurde, die Ihr PC an die IP-Adresse des zu steuernden Geräts geschickt hat, wird der Mikrocontroller mit einer Web-Seite antworten, die in seinem Speicher abgelegt ist. Diese Web-Seite wird dann automatisch im Browser des PCs angezeigt werden. Wenn ein „ON“-Befehl empfangen wurde, wird die an PORTD.0 angeschlossene LED leuchten. Entsprechend wird ein empfangener „OFF“-Befehl die LED wieder verlöschen lassen. Ist ein Relais anstelle einer LED angeschlossen, kann das Gerät diverse andere Geräte wie Lampen, Alarmanlagen, Heizungen etc. steuern.

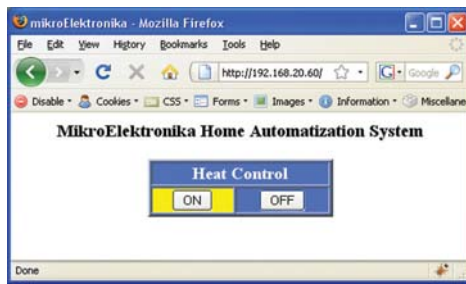


Abb. 1. MikroElektronika's Serielles Ethernet-Modul mit ENC28J60 hip



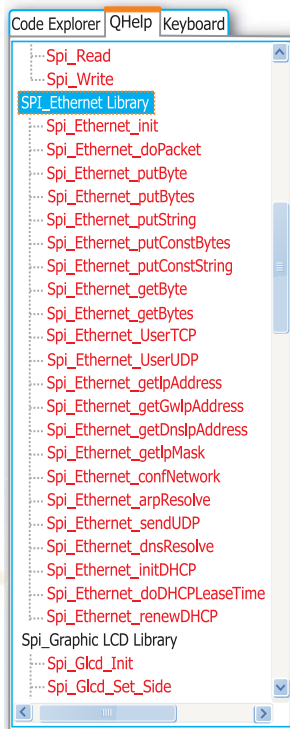
Schaltung 1. Anschluss des Serial -Ethernet-Module an einen dsPIC30F4013

Die Steuerung von Geraten erfolgt prinzipiell, indem die IP-Adresse des Gerats in die Adresszeile des Web-Browsers eingegeben wird und die gewnschten Befehle spezifiziert werden. Selbstverstandlich kann mehr als nur ein einziger Pin eines Mikrocontrollers gesteuert werden. Auf diese Weise knnen viele unterschiedliche Gerate gesteuert oder auch ein komplettes Home-Automation-System realisiert werden.



Das Bildschirmfoto zeigt die Web-Seite, die der Web-Browser anzeigt, nachdem man die IP-Adresse eines steuerbaren Gerats eingegeben hat. In unserem Beispiel bewirken Klicks auf die Schaltflachen „ON“ und „OFF“, dass eine angeschlossene LED ein- oder ausgeschaltet wird, womit man die korrekte Funktion z.B. einer Heizung simulieren knnte.

Nachfolgend die Liste vorgefertigter Befehle, wie sie in der Library „SPI Ethernet“ enthalten sind. Diese Library ist im Compiler mikroC for dsPIC integriert.



Spi_Ethernet_Init()*	Init ENC28J60 controller
Spi_Ethernet_Enable()	Enable network traffic
Spi_Ethernet_Disable()	Disable network traffic
Spi_Ethernet_doPacket()*	Process received packet
Spi_Ethernet_putByte()	Store a byte
Spi_Ethernet_putBytes()	Store bytes
Spi_Ethernet_putConstBytes()	Store const bytes
Spi_Ethernet_putString()*	Store string
Spi_Ethernet_putConstString()*	Store const string
Spi_Ethernet_getByte()*	Fetch a byte
Spi_Ethernet_getBytes()	Fetch bytes
Spi_Ethernet_UserTCP()*	TCP handling code
Spi_Ethernet_UserUDP()	UDP handling code
Spi_Ethernet_getIpAddress()	Get IP address
Spi_Ethernet_getGwIpAddress()	Get Gateway address
Spi_Ethernet_getDnsIpAddress()	Get DNS address
Spi_Ethernet_getIpMask()	Get IP mask
Spi_Ethernet_confNetwork()*	Set network parameters
Spi_Ethernet_arpResolve()	Send an ARP request
Spi_Ethernet_sendUDP()	Send an UDP packet
Spi_Ethernet_dnsResolve()	Send an DNS request
Spi_Ethernet_initDHCP()	Send an DHCP request
Spi_Ethernet_doDHCPLeaseTime()	Process lease time
Spi_Ethernet_renewDHCP()	DHCP renewal request

Andere im Programm verwendete Funktionen von mikroC for dsPIC:

- Spi_Init() Initialisiere das Mikrocontroller-SPI-Modul
- memcpy() Kopiere den RAM-Inhalt des Mikrocontrollers
- memcmp() Vergleiche die RAM-Inhalte des Mikrocontrollers

Beispiel 1: Demo-Programm zur Steuerung via Ethernet

```

// 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</title></head><body>\r\n"
"<h3 align=center>MikroElektronika Home Automation 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></td></tr>\r\n"
"<tr><td align=center bgcolor=#4974E2><input name='tst1' width=60\r\n"
" type='submit' value='ON'></td><td align=center bgcolor=#FFFFFF00>\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, "#FFFF00", 6); // highlight (yellow) ON
    memcpy(indexPage+431, "#4974E2", 6); // clear OFF
  }
  else
  {
    memcpy(indexPage+340, "#4974E2", 6); // clear ON
    memcpy(indexPage+431, "#FFFFFF00", 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) {
    Spi_Ethernet_doPacket(); // process incoming Ethernet packets
  }
}

```



HINWEIS: Beispiel-Code fr PIC-Mikrocontroller in C, Basic und Pascal sowie andere Programme fr dsPIC- und AVR-Mikrocontroller finden sich auf unserer Web-Seite: www.mikroe.com/en/article/