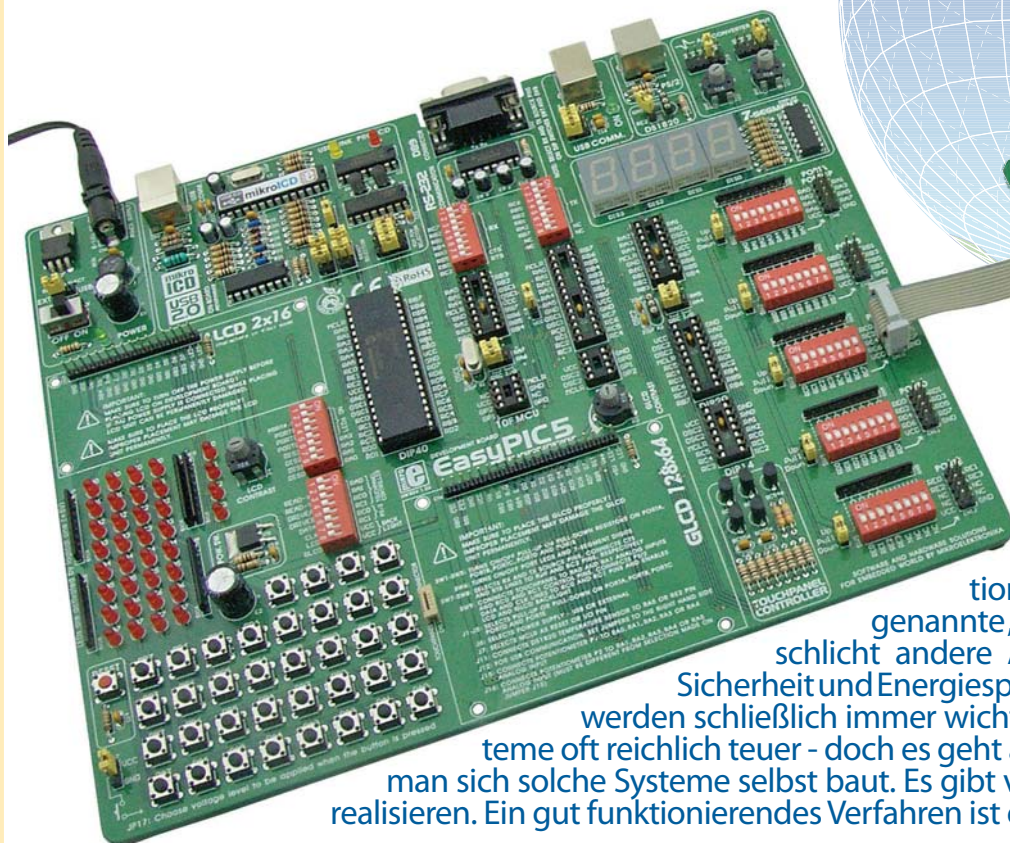


OK. Jetzt brauchen sie... ETHERNET



Serielles Ethernet-Modul mit Anschluss an das Entwicklungs-System EasyPIC5

Heim-Automation, Heim-Steuerung oder das so genannte „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 PIC18F4520-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 PORTB.0 des Mikrocontrollers angeschlossene LED simuliert das zu steuernde Gerät.

Der Compiler *mikroBASIC for PIC* 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 PORTB. 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.

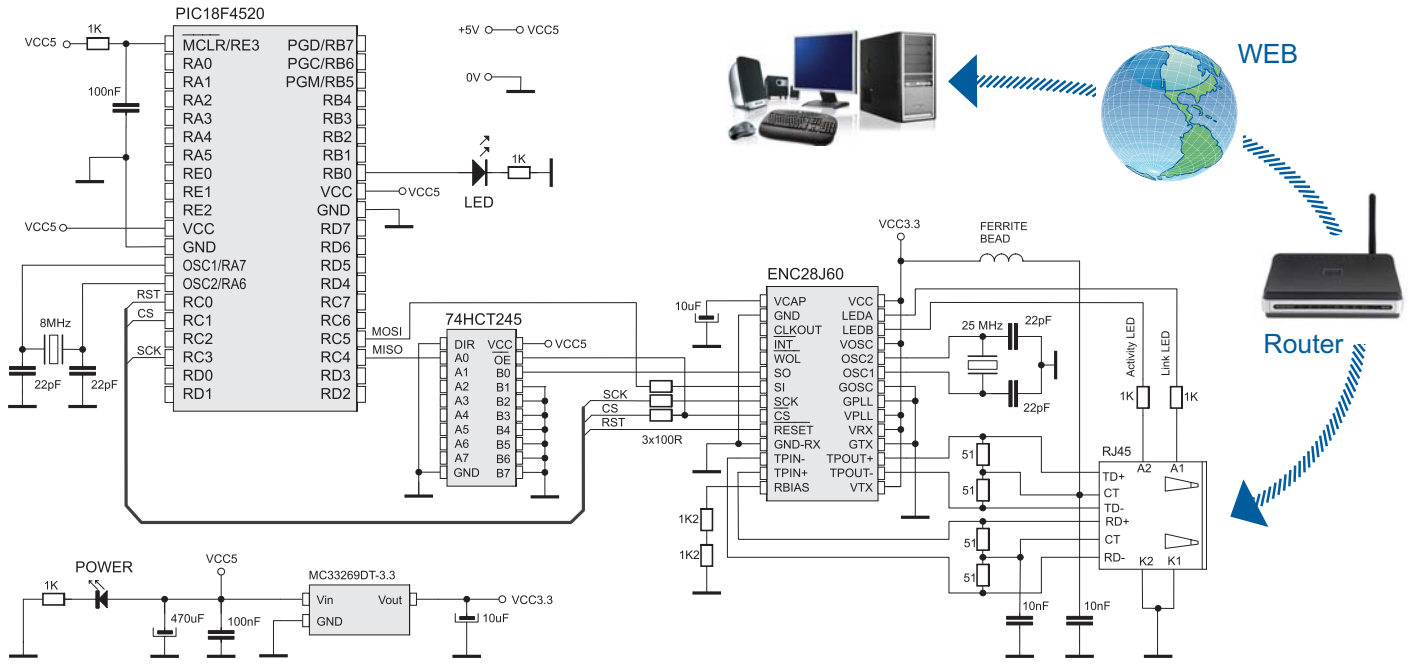
Schritt 6. Schreiben des Codes innerhalb der *SPI_Ethernet_userTCP*-Funktion, welcher nach Empfang eines Befehls via Web-Browser die an PORTB.0 angeschlossene LED ein bzw. ausschaltet.

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 PORTB.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.

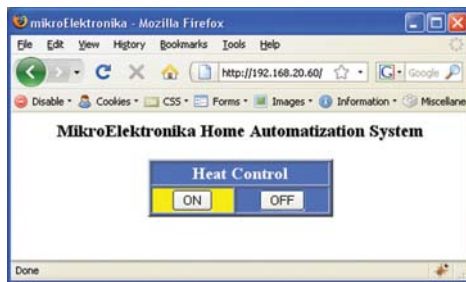


Abb. 1. MikroElektronika's Serielles Ethernet-Modul mit ENC28J60 hip



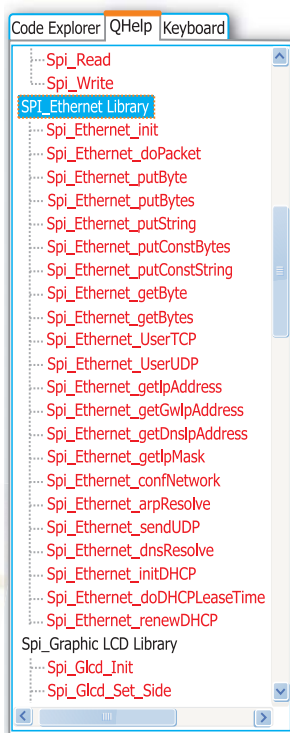
Schaltung 1. Anschluss des Serial -Ethernet-Module an einen PIC18F4520

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 mikroBASIC for PIC 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

* Im Programm verwendete Funktionen der Library „SPI Ethernet“

Andere im Programm verwendete Funktionen von mikroBASIC for PIC:

- 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 (two files)

```

program home_auto
include "home_auto_utils"

dim myMacAddr as byte[6]
myIpAddr as byte[4]
gwIpAddr as byte[4]
ipMask as byte[4]
dnsIpAddr as byte[4]

main:
ADCON1 = 0x0F
CMCON = 0x07

PORTB.0 = 0
TRISB.0 = 0

indexPage =
"<html><head><title>mikroElektronika</title></head><body>"+
"<h3 align=center>MikroElektronika Home Automation System</h3>"+
"<form name="+Chr(34)+"input"+Chr(34)+" action="+Chr(34)+" method="+
Chr(34)+" get"+Chr(34)+">"+table align=center width=200 bgcolor=#4974E2 border=2><tr>+
"<td align=center colspan=2><font size=4 color=white>Heat Control</font>"+
"</td></tr><tr><td align=center bgcolor=#4974E2><input name="+Chr(34)+"tst1"+
Chr(34)+" width=60 type="+Chr(34)+" submit"+Chr(34)+" value="+Chr(34)+"ON"+
Chr(34)+"></td><td align=center bgcolor=#4974E2><input name="+Chr(34)+"tst2"+
Chr(34)+" type="+Chr(34)+" submit"+Chr(34)+" value="+Chr(34)+"OFF"+Chr(34)+">"+
"</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]=6
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,
' my MAC & IP address, full duplex
Spi_Init()
' 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
Spi_Ethernet_doPacket()
wend
end

module home_auto_utils
const httpHeader as string[31] = "HTTP/1.1 200 OK"+chr(10)+"Content-type:"
const httpMimeTypeHTML as string[13] = "text/html"+chr(10)
const httpMimeTypeScript as string[14] = "text/plain"+chr(10)

' default html page
dim indexPage as string[523]
dim getRequest as byte[20]

implements
sub function Spi_Ethernet_UserTCP(dim byref remoteHost as byte[4],
dim remotePort, localPort, reqLength as word) as word
dim cnt as word
tmp as string[10]
' my reply length

if(localPort < 80) then
exit
end if
' listen only to web request on port 80

' get 10 first bytes only of the request, the rest does not matter here
dim cnt = 0 to 14
getRequest[cnt] = Spi_Ethernet_getByte()
next cnt

tmp = "GET "
if(memcmp@getRequest, @tmp, 5) < 0 then
result = 0
exit
end if

tmp = "ON"
if(memcmp@getRequest+11, @tmp, 2) = 0 then
PORTB.0 = 1
set PORTB bit 0
else
tmp = "OFF"
if(memcmp@getRequest+11, @tmp, 3) = 0 then
PORTB.0 = 0
clear PORTB bit 0
end if
end if

if(PORTB.0) then
tmp = "FFFF00" memcpy@indexPage+340, @tmp, 6
' highlight (yellow) ON
tmp = "#4974E2" memcpy@indexPage+431, @tmp, 6
' clear OFF
else
tmp = "#4974E2" memcpy@indexPage+340, @tmp, 6
' clear ON
tmp = "FFFF00" memcpy@indexPage+431, @tmp, 6
' highlight (yellow) OFF
end if

cnt = Spi_Ethernet_putConstString(@httpHeader)
cnt = cnt + Spi_Ethernet_putConstString(@httpMimeTypeHTML)
cnt = cnt + Spi_Ethernet_putString(@indexPage)
result = cnt
' return to the library with the number of bytes to transmit
end sub

sub function Spi_Ethernet_UserUDP(dim byref remoteHost as byte[4],
dim remotePort, destPort, reqLength as word) as word
result = 0
' back to the library with the length of the UDP reply
end.

```

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

