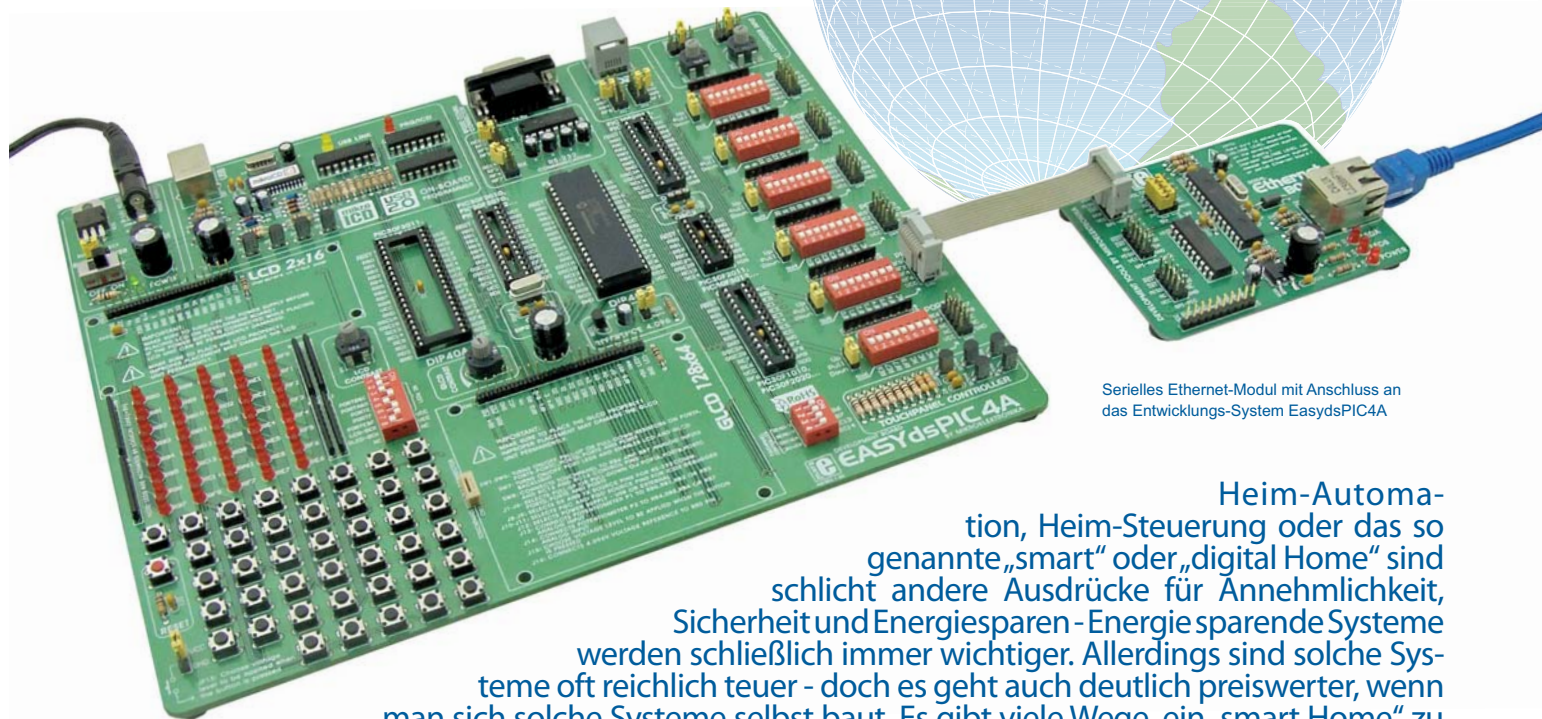


# OK. Jetzt brauchen sie... ETHERNET



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 *mikroBASIC 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.

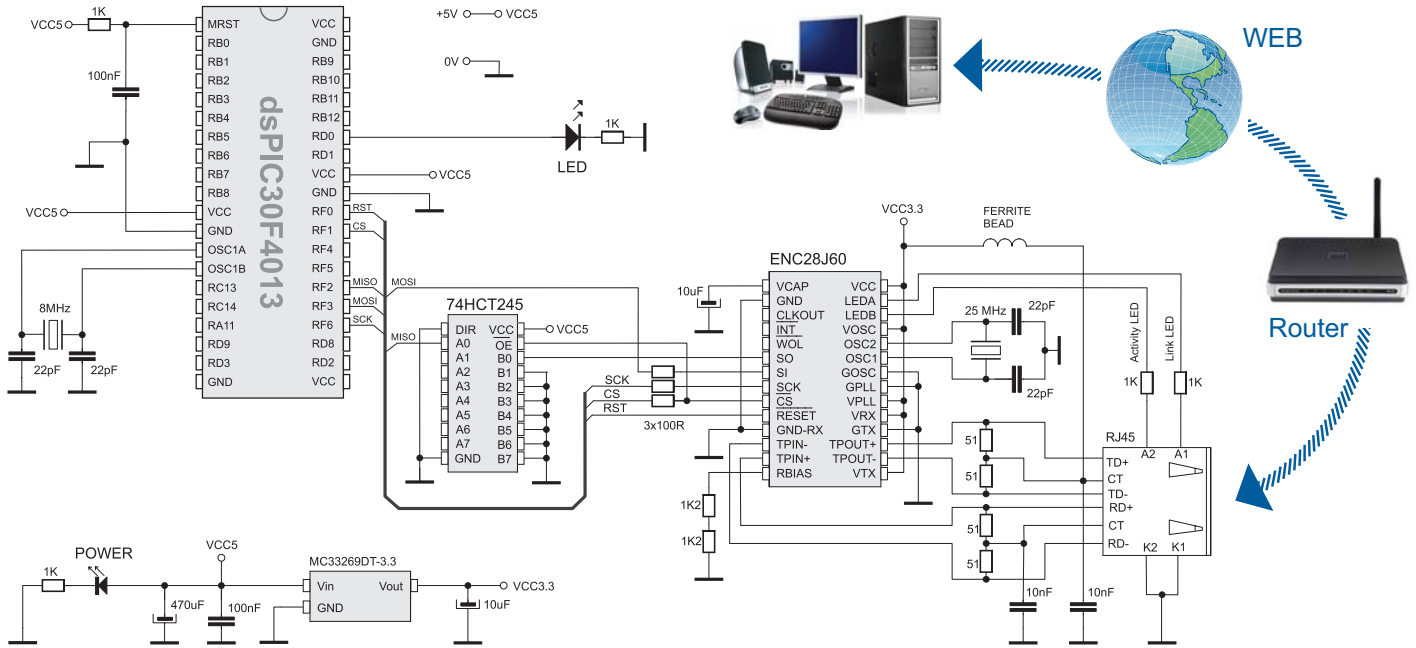
**Schritt 6.** Schreiben des Codes innerhalb der *Spi\_Ethernet\_userTCP*-Funktion, welcher nach Empfang eines Befehls via Web-Browser die an PORTD.0 angeschlossene LED ein bzw. ausschaltet.

**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.

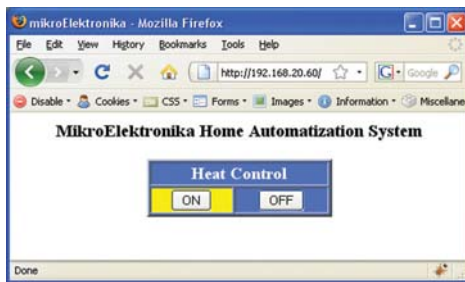


Abb. 1. MikroElektronika's Serielles Ethernet-Modul mit ENC28J60 hip



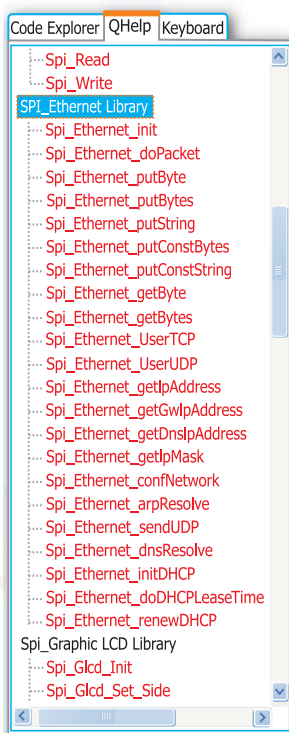
Schaltung 1. Anschluss des Serial -Ethernet-Module an einen dsPIC30F4013

Die Steuerung von Geräten erfolgt prinzipiell, indem die IP-Adresse des Geräts in die Adresszeile des Web-Browsers eingegeben wird und die gewünschten Befehle spezifiziert werden. Selbstverständlich kann mehr als nur ein einziger Pin eines Mikrocontrollers gesteuert werden. Auf diese Weise können viele unterschiedliche Geräte 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 Geräts eingegeben hat. In unserem Beispiel bewirken Klicks auf die Schaltflächen „ON“ und „OFF“, dass eine angeschlossene LED ein- oder ausgeschaltet wird, womit man die korrekte Funktion z.B. einer Heizung simulieren könnte.

Nachfolgend die Liste vorgefertigter Befehle, wie sie in der Library „SPI Ethernet“ enthalten sind. Diese Library ist im Compiler mikroBASIC 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_getIpAdress()	Get IP address
Spi_Ethernet_getGwIpAdress()	Get Gateway address
Spi_Ethernet_getDnsIpAdress()	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 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 (two files)

```

program home_auto
include "home_auto_utils"

dim myMacAddr as byte[6]      'my MAC address
myIpAddr as byte[4]         'my IP address

main:
ADPCFG = 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 Automation System</h3>"+
"<form name="+Chr(34)+"input"+Chr(34)+" action="+Chr(34)+"#"+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><b>Heat Control</b></td>"+
"</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=#FFFF00><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
myIpAddr[0]=192 myIpAddr[1]=168 myIpAddr[2]=20 myIpAddr[3]=60

' 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 true
Spi_Ethernet_doPacket() 'do forever
wend 'process incoming Ethernet packets

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

' default html page
dim indexPage as string[523]
dim getRequest as byte[20] ' HTTP request buffer
implements
sub function putConstString(dim const s as ^byte) as word
result = 0
while s <> 0
Spi_Ethernet_putByte(s) s = s + 1 result = result + 1
wend
end sub
sub function putString(dim byref s as byte[100]) as word
result = 0
while s[result] <> 0
Spi_Ethernet_putByte(s[result]) result = result + 1
wend
end sub
sub function SPI_Ethernet_UserTCP(dim byref remoteHost as byte[4],
dim remotePort, localPort, reqlength as word) as word
tmp as string[10] ' my reply length
if localPort <> 80 then ' I listen only to web request on port 80
result = 0
exit
end if
' get 10 first bytes only of the request, the rest does not matter here
for cnt = 0 to 14 getReq[ cnt ] = Spi_Ethernet_getByte() next cnt
getReq[ cnt ] = 0
tmp = "GET/"
if memncmp( getReq, @tmp, 5 ) > 0 then ' only GET method
result = 0
exit
end if
tmp = "ON"
if memncmp( getReq+11, @tmp, 2 ) = 0 then ' do we have ON command
PORTD.0 = 1
else
tmp = "OFF"
if memncmp( getReq+11, @tmp, 3 ) = 0 then ' do we have OFF command
PORTD.0 = 0
exit
end if
Delay_us(1)
if (PORTD.0) then
tmp = "#FFFF00" memcpy( @indexPage+340, @tmp, 6) ' highlight (yellow) ON
else
tmp = "#4974E2" memcpy( @indexPage+431, @tmp, 6) ' clear OFF
tmp = "#4974E2" memcpy( @indexPage+340, @tmp, 6) ' clear ON
tmp = "#FFFF00" memcpy( @indexPage+431, @tmp, 6) ' highlight (yellow) OFF
end if
cnt = putConstString( @httpHeader) ' HTTP header
cnt = cnt + putConstString( @httpMimeTypeHTML) ' with HTML MIME type
cnt = cnt + putString( indexPage) ' HTML page first part
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 sub
end.

```



HINWEIS: Beispiel-Code für PIC®-Mikrocontroller in C, Basic und Pascal sowie andere Programme für dsPIC®- und AVR®-Mikrocontroller finden sich auf unserer Web-Seite: [www.mikroe.com/en/article/](http://www.mikroe.com/en/article/)