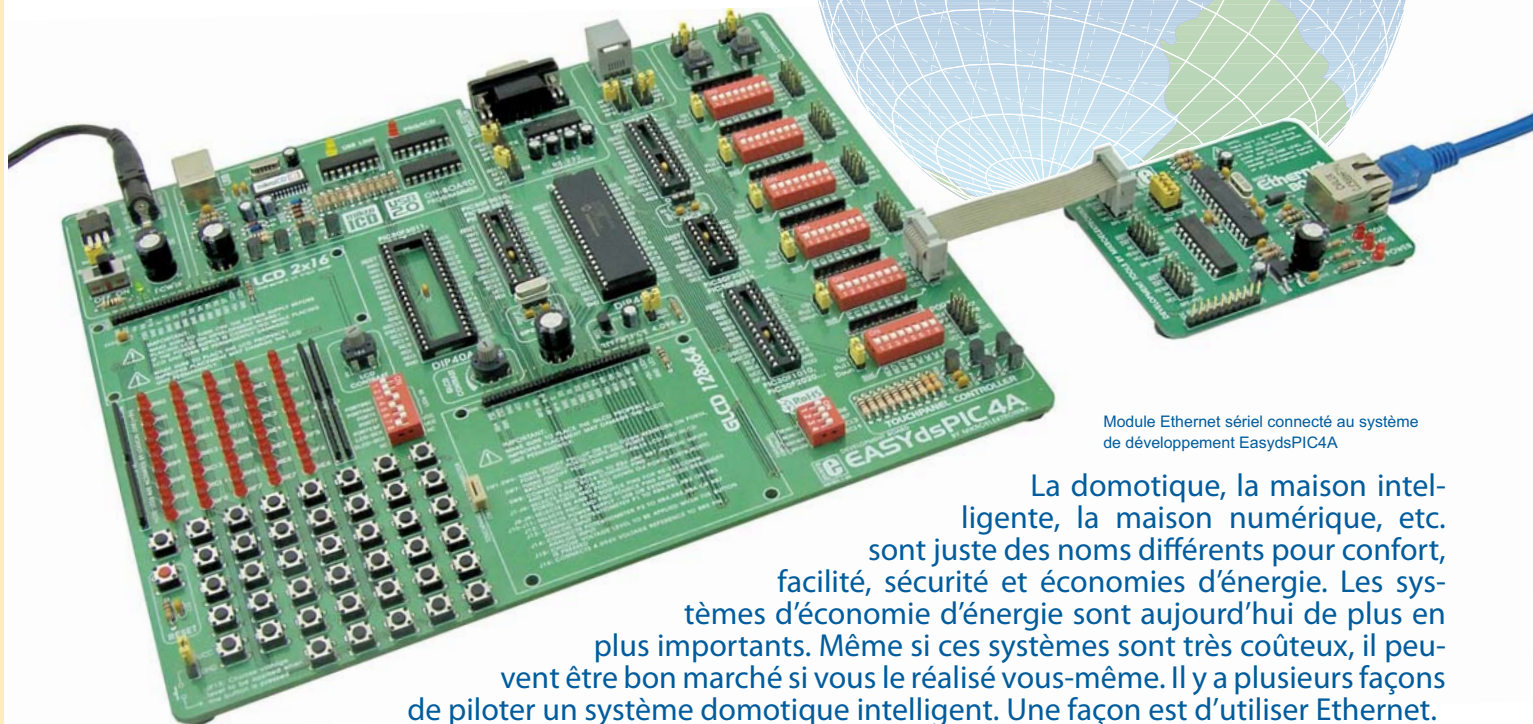


# Bon. ETHERNET

## Maintenant il vous faut...



Module Ethernet sériel connecté au système de développement EasysPIC4A

La domotique, la maison intelligente, la maison numérique, etc. sont juste des noms différents pour confort, facilité, sécurité et économies d'énergie. Les systèmes d'économie d'énergie sont aujourd'hui de plus en plus importants. Même si ces systèmes sont très coûteux, il peuvent être bon marché si vous le réalisez vous-même. Il y a plusieurs façons de piloter un système domotique intelligent. Une façon est d'utiliser Ethernet.

Par Srdjan Tomic  
MikroElektronika - Software Department

Tout ce dont vous avez besoin est un microcontrôleur dsPIC30F4013 et une puce Ethernet sérielle ENC28J60. Ce composant est aussi une excellente solution pour d'autres microcontrôleurs comme les AVR, dsPIC etc. Le connecteur CviLux CJCBA8HF1Y0 RJ-45 est utilisé pour la connexion avec le réseau Ethernet. Une LED connectée à PORTD.0 du microcontrôleur simule l'appareil ménager à piloter.

Le compilateur *mikroBASIC for dsPIC* possède une librairie *SPI\_Ethernet* qui simplifie beaucoup le développement du programme pour le microcontrôleur. En utilisant quelques fonctions de cette librairie, il est possible de créer un programme qui permet de commander vos appareils ménagers électriques par un navigateur Internet.

Suivez les étapes suivantes dans le programme:

- Étape 1.** Créez une page HTML pour piloter le microcontrôleur. Insérez-la dans le programme comme une trame de caractères.
- Étape 2.** Saisissez les adresses IP, DNS et Passerelle et le masque Subnet fourni par votre fournisseur Internet.

Voici un exemple de paramètres d'un réseau local:

- IP :** 192.168.20.60 (adresse du système de pilotage)
- DNS :** 192.168.20.1 (adresse du serveur des noms de domaines)
- GATEWAY :** 192.168.20.6 (adresse Passerelle)
- SUBNET :** 255.255.255.0 (masque Subnet)

- Étape 3.** Désactivez les entrées analogiques PORTD. Configurez les broches du microcontrôleur comme sortie et mettez-les à un niveau logique bas.
- Étape 4.** Initialisez le module SPI du microcontrôleur dsPIC30F4013.
- Étape 5.** Initialisez la puce Ethernet sériel ENC28J60
- Étape 6.** Écrivez le code dans la fonction `Spi_Ethernet_userTCP` qui, après avoir reçu une commande par le navigateur Internet, allumera ou éteindra la LED connectée à PORTD.0.
- Étape 7.** Lisez les données reçues dans une boucle infinie.

La fonction `Spi_Ethernet_userTCP` est la plus importante du programme, c'est elle qui traite toutes les commandes reçues. Après la réception d'une commande "GET", envoyé par le navigateur Internet sur votre ordinateur vers l'adresse IP du système de contrôle, le microcontrôleur répondra avec une page web stockée dans sa mémoire. Cette page est automatiquement affichée sur l'écran de l'ordinateur par le navigateur.

Quand la commande ON est reçue, la LED connectée à PORTD.0 sera allumée. Pareil, quand la commande OFF est reçue, la LED sera éteinte. Si vous utilisez un relais à la place de la LED, il est possible de piloter toute sorte d'appareils, comme une lumière, un système d'alarme, le chauffage, etc.



Figure 1. MikroElektronika's Module Ethernet sériel avec une puce ENC28J60

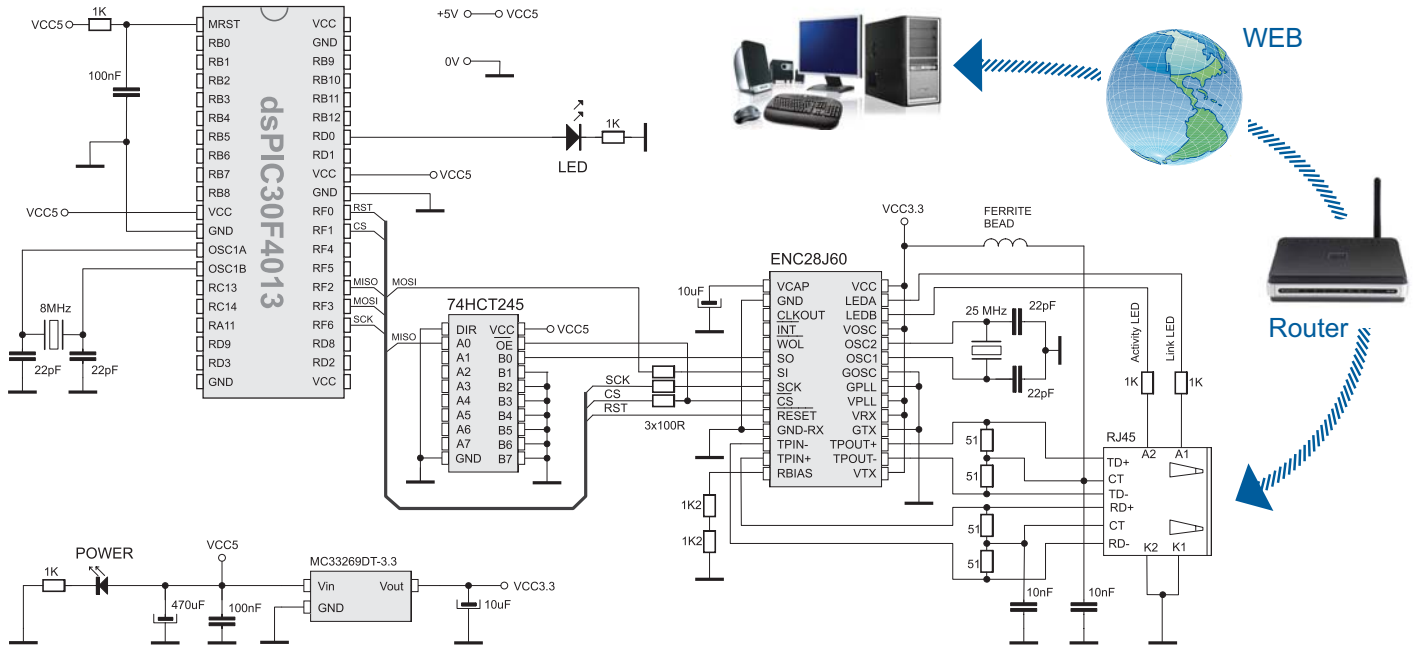
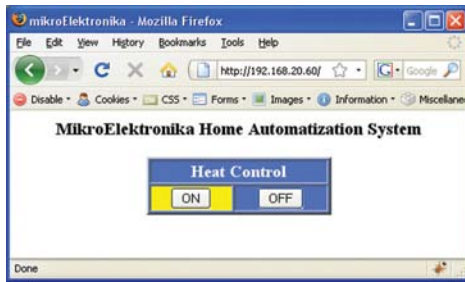


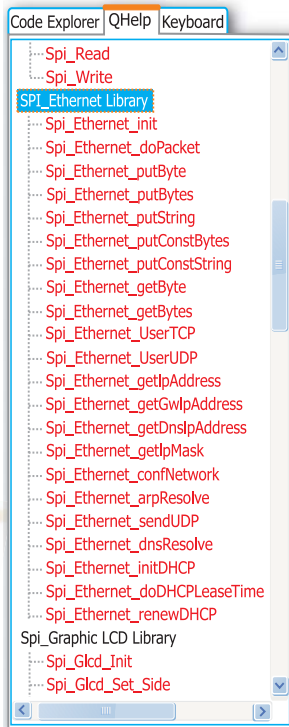
Schéma 1. Comment connecter le module Ethernet sériel à un dsPIC30F4013

Le pilotage d'un appareil se fait en saisissant l'adresse IP du système de contrôle dans le navigateur Internet et de spécifier les commandes souhaitées. Bien sûr, il est possible de piloter plus d'une broche du microcontrôleur, ce qui vous permet de commander un grand nombre d'appareils ou un système domotique complet.



La photo d'écran montre la page web affichée par le navigateur Internet après la saisie de l'adresse IP du système de contrôle. Dans notre exemple, les clics sur les boutons ON et OFF changent l'état de la LED, ainsi simulant le système de chauffage.

Voici la liste des fonctions déjà disponibles dans la librairie SPI Ethernet (ci-dessous). Cette librairie est fournie avec le compilateur mikroBASIC for dsPIC.



Spi_Ethernet_Init()	Initialiser contrôleur ENC28J60
Spi_Ethernet_Enable()	Activer trafic réseau
Spi_Ethernet_Disable()	Désactiver trafic réseau
Spi_Ethernet_doPacket()	Traiter un paquet reçu
Spi_Ethernet_putByte()	Stocker un octet
Spi_Ethernet_putBytes()	Stocker plusieurs octets
Spi_Ethernet_putConstBytes()	Stocker plusieurs octets constants
Spi_Ethernet_putString()	Stocker une trame de caractères
Spi_Ethernet_putConstString()	Stocker une trame de caractères constantes
Spi_Ethernet_getByte()	Lire un octet
Spi_Ethernet_getBytes()	Lire plusieurs octets
Spi_Ethernet_UserTCP()	Gestionnaire TCP
Spi_Ethernet_UserUDP()	Gestionnaire UDP
Spi_Ethernet_getIpAddress()	Lire l'adresse IP
Spi_Ethernet_getGwIpAddress()	Lire l'adresse Passerelle
Spi_Ethernet_getDnsIpAddress()	Lire l'adresse DNS
Spi_Ethernet_getIpMask()	Lire le masque IP
Spi_Ethernet_confNetwork()	Saisir les paramètres réseau
Spi_Ethernet_arpResolve()	Envoyer une requête ARP
Spi_Ethernet_sendUDP()	Envoyer un paquet UDP
Spi_Ethernet_dnsResolve()	Envoyer une requête DNS
Spi_Ethernet_initDHCP()	Envoyer une requête DHCP
Spi_Ethernet_doDHCPLeaseTime()	Gérer temps de bail
Spi_Ethernet_renewDHCP()	Rafraîchir requête DHCP

\* Fonctions de la librairie SPI Ethernet utilisées dans le programme

Autres fonctions de mikroBASIC for dsPIC utilisées dans le programme:

- memcpy() Copier de la mémoire RAM du microcontrôleur
- memcmp() Comparer de la mémoire RAM du microcontrôleur

Exemple 1 : Exemple d'un programme de pilotage par 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

PORTD0 = 0                  'set PORTD.B0 as output (rele control pin)
TRISD0 = 0

indexPage =
<html><head><title>mikroElektronika</title></head><body>+
<h3 align=center>MikroElektronika Home Automatization 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>+
</tr></table></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 PORTF0, CS bit on PORTF1,
' 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]
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 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 if
tmp = "ON"
if memncmp[ getReq+11, @tmp, 2 ] = 0 then 'do we have ON command
PORTD0 = 1
set PORTD bit 0
else
tmp = "OFF"
if memncmp[ getReq+11, @tmp, 3 ] = 0 then 'do we have OFF command
PORTD0 = 0
clear PORTD bit 0
end if
end if
Delay_us(1)
if (PORTD0) 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 = putConstString[ @httpHeader ]
cnt = cnt + putConstString[ @httpMimeTypeHTML ]
cnt = cnt + 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 sub
end.

```

NOTE: Les codes source de cet exemple en C, BASIC et PASCAL pour microcontrôleurs dsPIC®, ainsi que tous les programmes écrits pour les microcontrôleurs PIC® et AVR® sont disponibles sur notre site Internet : [www.mikroe.com/en/article/](http://www.mikroe.com/en/article/)

