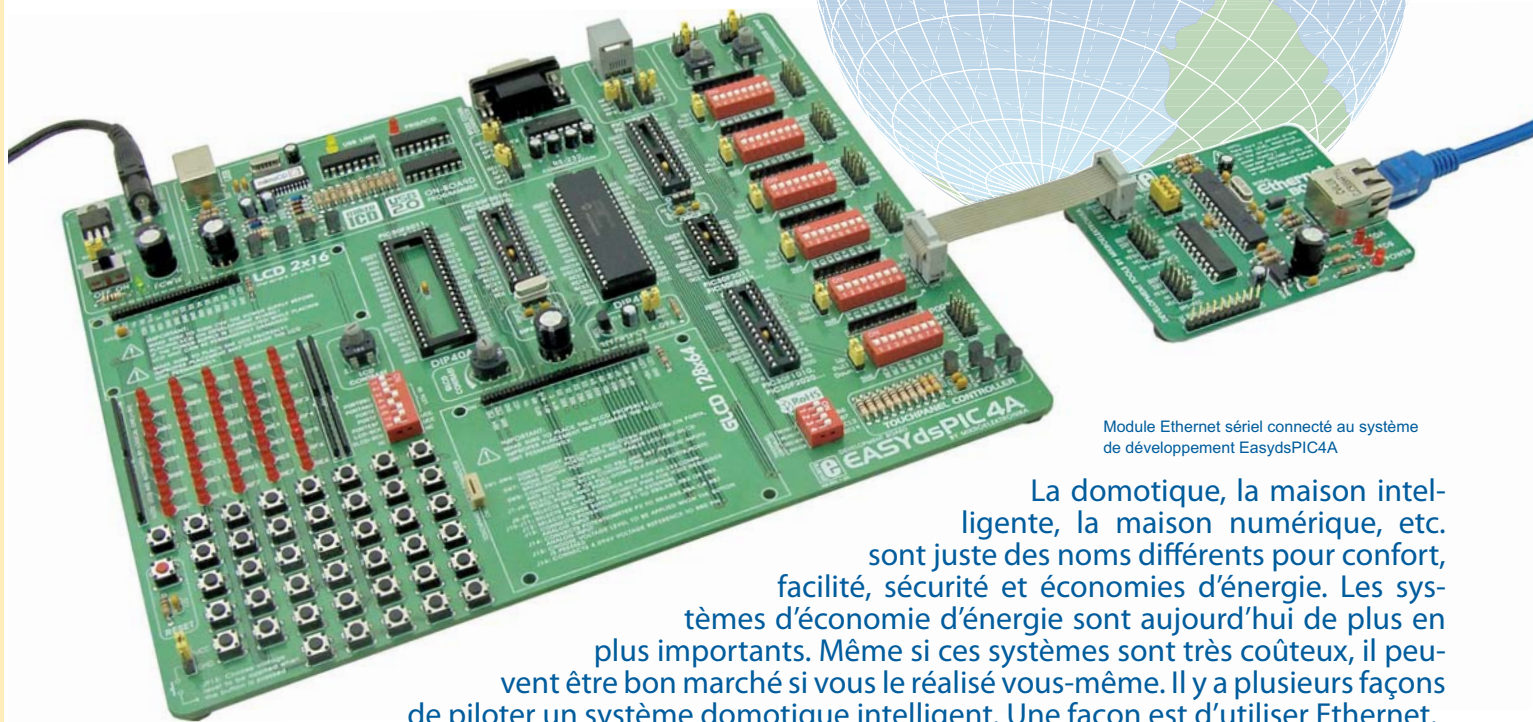


Bon. ETHERNET

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 *mikroC 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

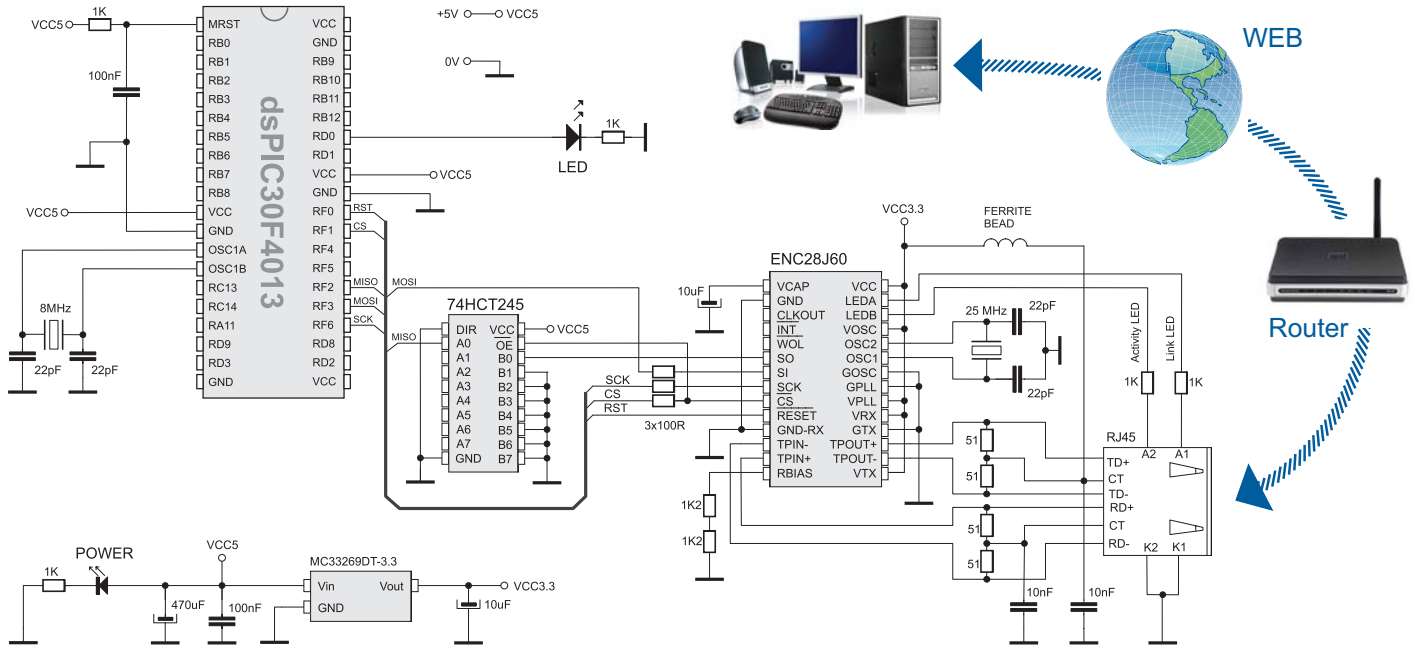
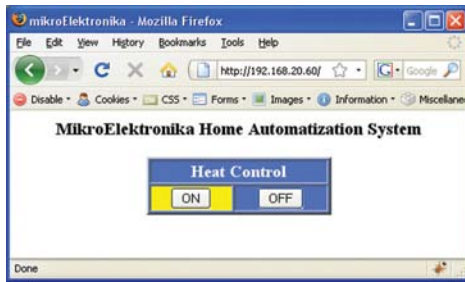


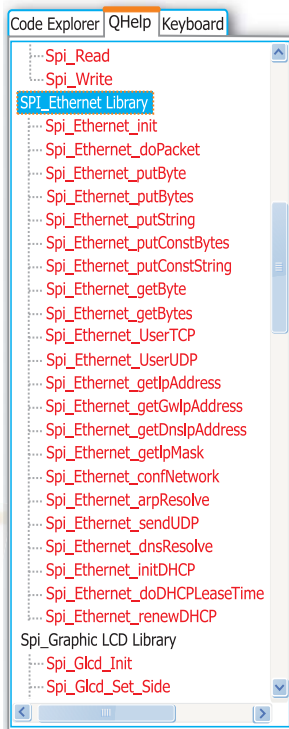
Schéma 1. Comment connecter le module Ethernet sériel à un dsPIC30F4013

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 *mikroC 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 mikroC for dsPIC utilisées dans le programme:	
Spi_Init()	Initialiser module SPI du microcontrôleur
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

```

// duplex config flags
#define Spi_Ethernet_HALFDUPLEX 0x00 // half duplex
#define Spi_Ethernet_FULLLDUPLEX 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>\
<h3 align=center>MikroElektronika Home Automatization System</h3>\
<form name='input' action='/' method='get'>\
<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='tst1' width=60\
type='submit' value='ON'></td><td align=center bgcolor=#FFFFFF>\
<input name='tst2' type='submit' value='OFF'></td></tr></table>\
</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, "#FFFFFF", 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_FULLLDUPLEX);

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

```



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/