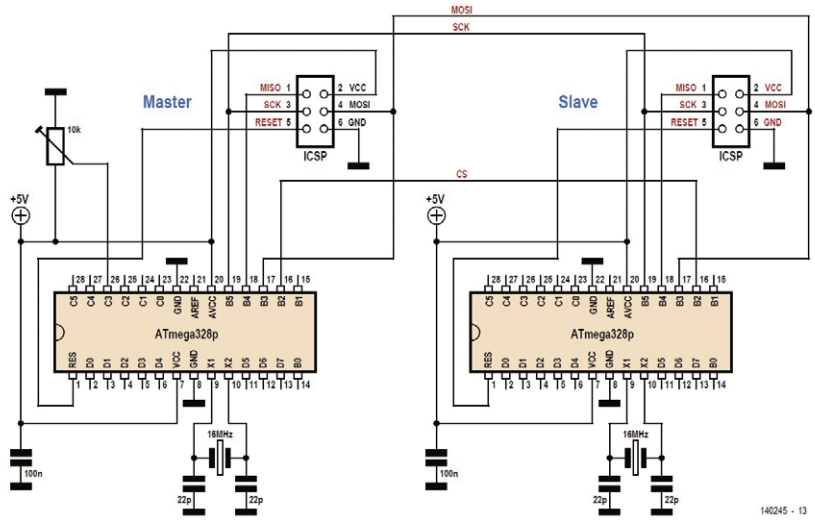
# trafic entre µC
## par le bus SPI et l'ATmega328p

**Burkhard Kainka** (Allemagne)

Voici un aperçu de l'envoi de données par le bus SPI d'un microcontrôleur à un autre. À titre d'exemple, les données consistent en lectures sur 10 bits provenant d'un convertisseur A/N. Un (autre) avantage du SPI est la liberté de format des données : que vous envoyiez 8, 10, 12 ou 16 bits, la procédure reste la même. Si le seul objectif était de connecter deux µC l'un à l'autre, il serait plus facile d'utiliser une interface série asynchrone avec les lignes TXD et RXD. Le bus SPI, en revanche, est meilleur pour commander le matériel externe et communiquer avec lui. Notre objectif est ici d'illustrer le protocole de transmission.

Outre MOSI et MISO, le schéma montre qu'une 3e ligne est impliquée




**Listage 1 : maître SPI**

```
'--------------------------------
'UNO_spi2.BAS SPI Master
'--------------------------------
$regfile = "m328pdef.dat"
$crystal = 16000000
$baud = 9600
Dim B As Bit
Dim Dout As Word
Dim N As Byte
Dim I As Byte
Sck Alias Portb.5
Ddrb.5 = 1
Mosi Alias Portb.3
Ddrb.3 = 1
Cs Alias Portb.2
Ddrb.2 = 1
Cs = 1
Mosi = 0
Sck = 0
Config Adc = Single , Prescaler = 32 ,
Reference = Avcc
Start Adc
Cls
Cursor Off
Waitms 200
Do
   Dout = Getadc(3) 'Pot
   Locate 1 , 1
   Lcd Dout
   Lcd " "
   Cs = 0
   Waitms 20
   For N = 1 To 10
     Mosi = Dout.9
     Waitms 1
     Sck = 1
     Waitms 1
     Sck = 0
     Waitms 1
     Shift Dout , Left
   Next N
   Cs = 1
   Waitms 100
Loop
End
```

**Listage 2 : esclave SPI**

```
'-----------------------------
 'UNO_spi3.BAS SPI Slave
 '-----------------------------
 $regfile = "m328pdef.dat"
 $crystal = 16000000
 $baud = 9600

Dim Addr As Byte
 Dim B As Bit
 Dim Dout As Word
 Dim Din As Word
 Dim N As Byte
 Dim I As Byte

S1 Alias Pinc.0
 Portc.0 = 1
 S2 Alias Pinc.1
 Portc.1 = 1
 Sck Alias Pinb.5
 Portb.5 = 1
 Mosi Alias Pinb.3
 Portb.3 = 1
 Cs Alias Pinb.2
 Portb.2 = 1
 ...
 Do
    Do
    Loop Until Cs = 0
    Din = 0
    For N = 1 To 10
      Shift Din , Left
      Do
      Loop Until Sck = 1
      Din = Din + Mosi
      Do
      Loop Until Sck = 0
    Next N
    Do
    Loop Until Cs = 1
    Locate 1 , 1
    Lcd Din
    Lcd " "
    Print Din
 Loop
 End
```

– dans ce cas la ligne de sélection de la puce /CS. La barre oblique (/) signifie que le signal sur cette ligne est actif au niveau bas. /CS permet de connecter plusieurs esclaves à un seul maître. Dans ce cas, ils partagent les lignes de données et d'horloge, mais chacun a sa propre ligne de sélection de puce. Lorsque cette ligne est basse, l'esclave correspondant sait qu'il est sélectionné. L'utilisation d'une ligne de sélection des puces présente un autre avantage. S'il y a un retard dans l'activation de l'esclave, il peut y avoir une certaine confusion sur les bits déjà transférés. Cependant, si l'esclave attend de voir un front descendant sur son entrée CS (transition du niveau haut vers le niveau bas), il sait que le transfert commence.
Et si une impulsion de bruit vient à se faufiler comme signal d'horloge, le reste des données pour ce transfert sont perdues, mais dès le prochain accès, tout rentre dans l'ordre.

L'ATmega328 illustré ici utilise également le bus SPI pour le téléchargement de programmes à partir d'un appareil de programmation externe. Les lignes suivantes sont donc disponibles sur le connecteur de programmation à six broches de la carte Arduino et sur un écran d'extension comme celui décrit à [1] (ICSP dans le schéma) :

- ligne d'horloge Serial Clock (SCK) sur B5 ;
- la ligne de données d'écriture Master Out Slave In (MOSI) sur B3 ;
- la ligne de données de lecture Master In Slave Out (MISO) sur B4.

Il n'y a pas de ligne de sélection de puce sur les connecteurs, mais la ligne de réinitialisation a le même effet car la programmation se fait avec la ligne de réinitialisation forcée au niveau bas. Nous voulons maintenant utiliser ces lignes exactement comme prévu. Cela a l'avantage de nous permettre d'utiliser l'unité SPI du µC, si celui-ci en possède une. Avec le SPI matériel, nous n'avons pas besoin d'utiliser un code de programme pour mettre chaque bit individuellement sur la ligne de données comme dans les exemples précédents, et tout est beaucoup plus rapide. Cependant, il nous faut une ligne de sélection de puce, et dans ce cas, ce sera la ligne B2 à cette fin.
Le maître utilise la ligne MOSI comme sortie et distribue les signaux d'horloge et de sélection de puce comme indiqué dans le **listage 1**. Le processus est ralenti un peu par trois retards de 1 ms chacun qui permettent de distinguer les signaux sur l'oscilloscope. En outre, nous ne voulons pas compliquer les choses pour l'esclave. Vous pouvez tester les limites en réduisant progressivement ces retards jusqu'à ce qu'apparaissent des erreurs de transmission.

Les trois lignes sont des entrées pour l'esclave qui exécute le programme du **listage 2**. Il attend constamment des fronts de signal spécifiques sur les lignes /CS et SCK, puis lit un bit de la ligne MOSI. Comme tout ici est géré par le logiciel, le code doit attendre chaque front dans une boucle `Do`.
Cela prend un peu de temps, la transmission des données doit donc être plus lente qu'avec une SPI matérielle. Les données reçues sont affichées sur l'écran et sur l'émulateur de terminal.
Lorsque vous tournez l'ajustable sur la carte maîtresse, le changement est visible sur l'écran LCD de l'extension [1] et transmis avec les données provenant de l'esclave.

200202-02

## LIEN

[1] **My First Shield :-)** : www.elektormagazine.com/140009