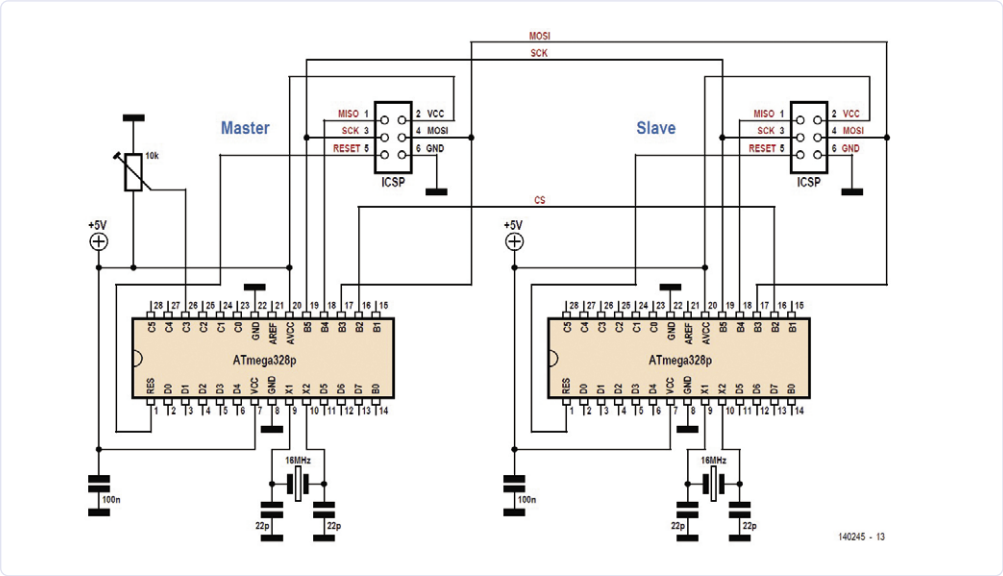
Inter-µC-communicatie
met de SPI-bus en ATmega328p

Burkhard Kainka (Duitsland)



Hier is een snelle inleiding tot het verzenden van gegevens via de SPI-bus van de ene microcontroller naar de andere. De gegevens betreffen in dit geval 10-bit metingen van bijvoorbeeld een A/D-converter. U ziet hier meteen een ander voordeel van SPI, namelijk dat de gegevensbreedte niet vast is. Of u nu 8, 10, 12 of 16 bits verstuurt, de procedure is altijd hetzelfde. Als het er alleen maar om zou gaan twee microcontrollers met elkaar te verbinden, zou het eigenlijk eenvoudiger zijn om een asynchrone seriële interface te gebruiken met de TXD- en RXD-lijnen. De SPI-bus is daarentegen beter voor het besturen van en communiceren met externe hardware. Hier gaat het er ons vooral om het transmissieprotocol te illustreren.

In het schema zien we naast MOSI en MISO nog een derde verbinding – in dit geval de chip select-lijn /CS. De slash (/) betekent dat het signaal op deze lijn actief laag is. Dankzij /CS kunt u meerdere slave-componenten op één master aansluiten. In dat geval delen

Listing 1. SPI-master.

```
'--------------------------------
'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
```

Listing 2. SPI-slave.

```
'-----------------------------
 '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
```

ze de data- en kloklijnen, maar elk heeft zijn eigen chip select-lijn. Als die laag is, weet de betreffende slave dat deze is geselecteerd. Er is nog een ander voordeel van het gebruik van een chip select-lijn. Als er vertraging optreedt bij het inschakelen van de slave, kan er enige verwarring bestaan over welke bits al zijn overgedragen. Als de slaaf echter wacht totdat hij een dalende flank op zijn CS-ingang ziet (dus een overgang van hoog naar laag), weet hij dat de overdracht begint.
En als een stoorpuls als een kloksignaal wordt beschouwd, is de rest van de gegevens voor die overdracht ongeldig, maar bij de volgende transmissie is alles weer zoals het zou moeten zijn.

De ATmega328 die hier als voorbeeld dient, gebruikt de SPI-bus ook voor het downloaden van programma's vanaf een extern programmeerapparaat. De volgende aansluitingen zijn daarom beschikbaar op de zespolige programmeerconnector op het Arduino-board en op een experimenteer-shield zoals beschreven bij [1] (ICSP in het schema):

- kloklijn Serial Clock (SCK) op B5;
- de write data-lijn Master Out Slave In (MOSI) op B3;
- de read data-lijn Master In Slave Out (MISO) op B4.

Er is geen chip select-lijn op de connectoren, maar de resetlijn heeft hetzelfde effect omdat het programmeren plaatsvindt met de resetlijn laag. Nu willen we deze lijnen precies gebruiken zoals bedoeld. Dat heeft het voordeel dat we de hardware-SPI-eenheid van de microcontroller kunnen gebruiken (als die er is). Met hardware-SPI hoeven we geen programmacode te gebruiken om elk bit afzonderlijk op de datalijn te plaatsen zoals in de vorige voorbeelden, en alles is een stuk sneller. We hebben echter nog steeds een chip select-lijn nodig en in dit geval gebruiken we hiervoor de B2-lijn.

De master gebruikt de MOSI-lijn als uitgang en genereert de klok- en chip select-signalen zoals aangegeven in **listing 1**. We verlangzamen het proces een beetje met drie vertragingen van 1 milliseconde, zodat alle signalen goed zichtbaar zijn op de oscilloscoop. Bovendien willen we het de slave niet te moeilijk maken. Als u daar zin in hebt, kunt u proberen wat de maximale snelheid is door de vertragingen steeds verder te reduceren tot er transmissiefouten optreden.
De drie lijnen zijn ingangen voor de slave-component waarop het programma van **listing 2** draait. Het wacht voortdurend op specifieke signaalflanken op de /CS- en SCK-lijnen en leest dan een bit in van de MOSI-lijn. Aangezien alles hier door software wordt afgehandeld, moet de code in een `Do`-lus steeds op elke flank wachten.
Dat kost wat tijd, en daarom is de gegevensoverdracht langzamer dan bij een hardware-SPI-implementatie. De ontvangen gegevens worden getoond op het display en op de terminal-emulator.
Wanneer u de instelpot op het moederbord verdraait, is de verandering zichtbaar op het LC-display van het uitbreidingsshield [1].

200202-03

WEBLINK

[1] **Experimenteer-shield :-)**: www.elektormagazine.nl/magazine/elektor-201407/26971