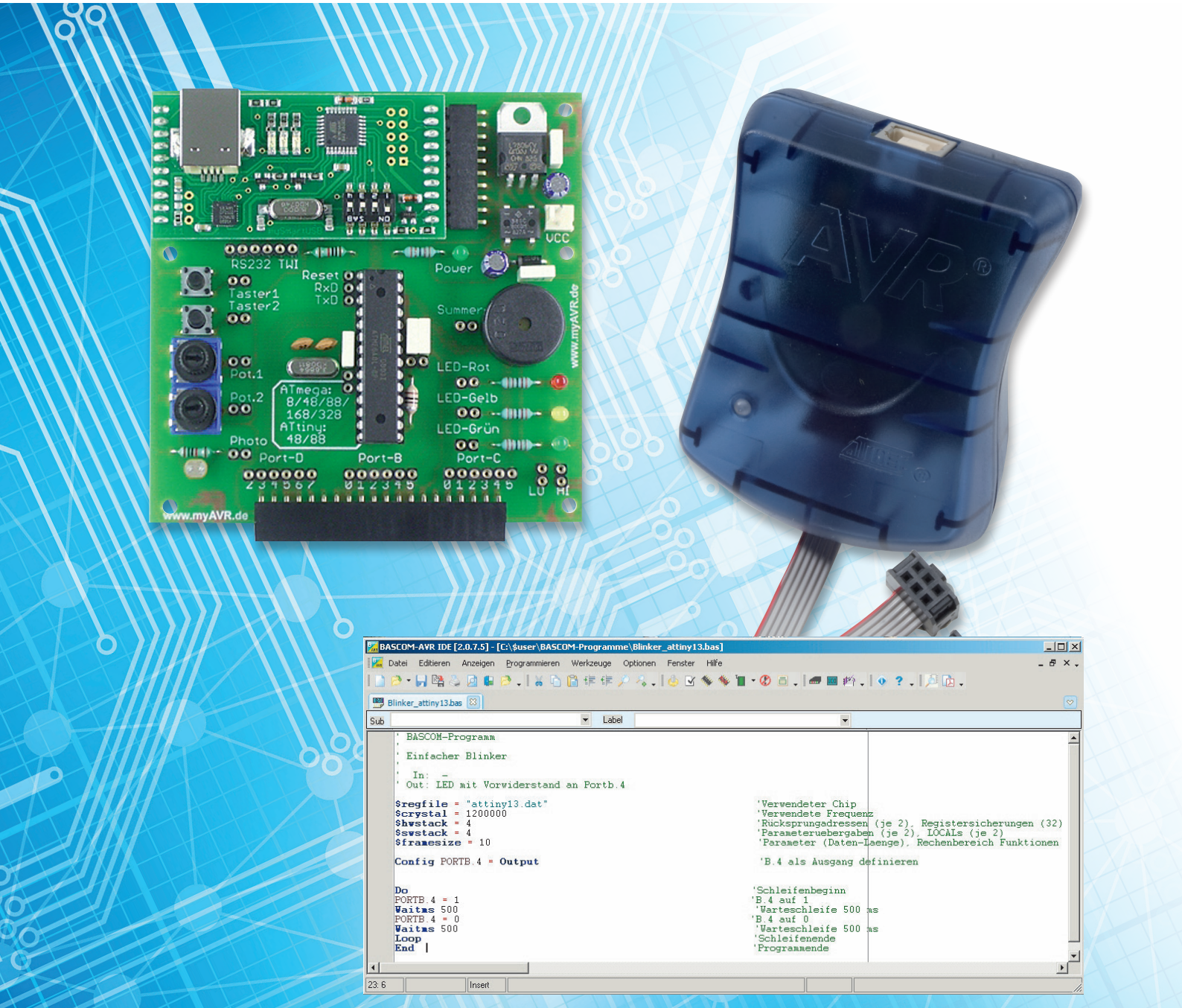
Mikrocontroller-Einstieg

Teil 15: SPI



mit BASCOM-AVR

Die SPI-Schnittstelle ist eine weit verbreitete serielle Schnittstelle, die von Motorola entwickelt wurde. SPI steht für Serial Peripheral Interface. SPI eignet sich sehr gut für die schnelle Übertragung von Daten zwischen Mikrocontrollern bzw. die Datenverbindung zwischen einem Mikrocontroller und einem oder mehreren Ausgabe- oder Eingabemodulen wie beispielsweise Displays oder Bewegungssensoren (3D-BS, 6D-BS von ELV). Auch Digital-Analog-Wandler, Analog-Digital-Wandler, EEPROM-Bausteine, die RTC-DCF von ELV (Best.-Nr. J8-13 05 41), SD-Karten und vieles mehr werden mit SPI-Schnittstelle angeboten. Die SPI-Schnittstelle arbeitet nach dem Schieberegister-Prinzip. Deshalb wird in diesem Artikel zunächst das Prinzip und die BASCOM-Ansteuerung von Schieberegistern dargestellt. Danach wird die SPI-Ansteuerung des Beschleunigungssensors 3D-BS von ELV (Best.-Nr. J8-09 15 21) beschrieben.

Schieberegister

Ein Schieberegister ist eine elektronische Schaltung, welche aus einer Anzahl in Reihe geschalteter Speicherzellen besteht, die jeweils genau ein Bit speichern können. Jede Speicherzelle kann den Inhalt 0 oder den Inhalt 1 haben. Kennzeichnend und namensgebend ist die Möglichkeit, die gespeicherten Informationen um jeweils eine Stelle weiterschieben zu können. Das Weiterschieben erfolgt synchron zu einem am Takteingang angelegten Signal (für alle Speicherstellen gleichzeitig). Der Inhalt der letzten Speicherzelle wird aus dem Schieberegister hinausgeschoben. In die erste Stelle des Schieberegisters wird die Information geschoben, die an einem Eingangspin anliegt.

Ansteuerung Schieberegister

Am Beispiel des beliebten Schieberegister-ICs 74HC595 (Best.-Nr. J8-11 37 21) wird ein konkretes Schieberegister betrachtet. Beim Schieberegister 74HC595 gibt es acht Speicherstellen. In Bild 1 sieht man das eigentliche Schieberegister in der oberen Hälfte des blau umrandeten Kastens. Im Beispiel haben die Speicherzellen den Inhalt 1-1-0-0-0-0-0-1. An Pin 11 des Schieberegister-ICs wird ein Taktsignal angelegt. Bei einem Wechsel des angelegten Taktsignals von 0 zu 1 (= steigende Flanke) wird jeweils die Information in einer Speicherstelle in die nachfolgende Speicherstelle geschoben. Die Information, die in der letzten Speicherstelle war, wird dabei an Pin 9 hinausgeschoben und die Information, die an Pin 14 anliegt, wird in die erste Stelle des Schieberegisters übernommen.

Um beim Schieben ein Flackern an den Parallelausgängen Q0 bis Q7 zu verhindern, werden die Informationen aus den Speicherstellen des Schieberegisters erst dann in ein Ausgaberegister (unten im blauen Kasten in Bild 1) übernommen, wenn an Pin 12 (Latch) ein Wechsel von 0 nach 1 (= steigende Flanke) erfolgt. Die Ansteuerung eines Schieberegister-Bausteins kann prinzipiell mit Tastern „von Hand" erfolgen. Die wahren Vorteile kommen natürlich erst zum Tragen, wenn die Ansteuerung wie in Bild 1 durch einen Mikrocontroller erfolgt. Für die Ansteuerung genügen drei Pins des Mikrocontrollers für den Takt, die neu einzuschiebende Information und die Datenübernahme in das Ausgaberegister. An die Ausgänge des Schieberegisters sind acht LEDs angeschlossen. Mit drei Ausgabepins lassen sich also acht LEDs oder andere Verbraucher schalten. Noch besser: Es gibt auch Schieberegister mit mehr als acht Speicherstellen und es ist auch möglich, mehrere Schieberegister hintereinander zu schalten. Zur Ansteuerung werden auch dann nur drei Pins des Mikrocontrollers benötigt.

Anmerkungen zum 74HC595: Mit Pin 13 kann man die Ausgabe des Schieberegisters komplett ein- oder ausschalten (Output Enable). Im Beispiel liegt der Pin 13 fest an GND, wodurch die Ausgabe immer eingeschaltet ist. Pin 10 (Reset) liegt fest an +5 V. Mit GND an Pin 10 könnte das Schieberegister komplett zurückgesetzt werden, was aber auch durch serielles Einschieben von acht Nullen erfolgen kann.

Achtung: Mit 70 mA ist die maximale Strombelastbarkeit des 74HC595 zu niedrig, um acht Standard-LEDs gleichzeitig anzusteuern. Hier wurden daher Low-Current-LEDs (J8-05 83 56) verwendet. Alternativ könnte man Treiberstufen, zum Beispiel mit Transistoren, an die Ausgabepins schalten. Es gibt auch Schieberegister mit einer höheren Strombelastbarkeit als das 74HC595, zum Beispiel TPIC6A595/TPIC6B595/TPIC6C595 (Open Drain).

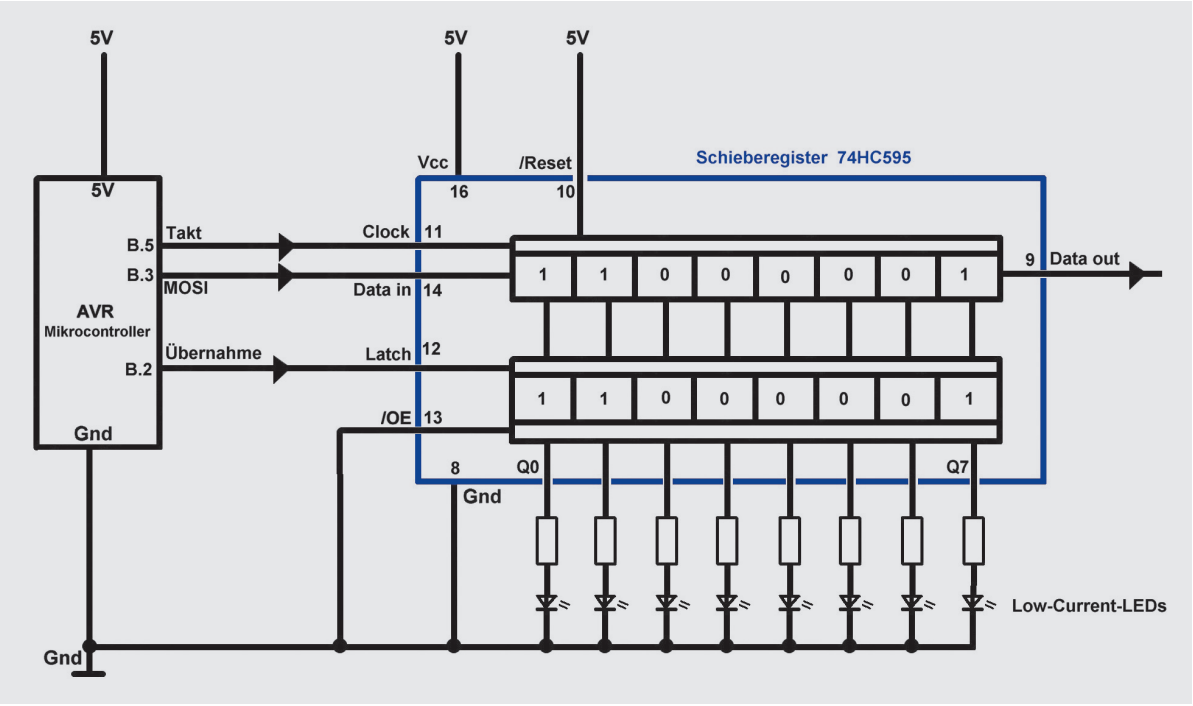
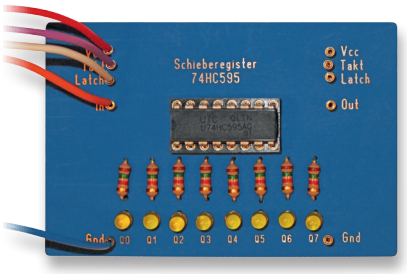


Bild 1: Schieberegister am AVR

Das folgende BASCOM-Programm zeigt die Ansteuerung eines Schieberegisters mit Standard-BASCOM-Befehlen.

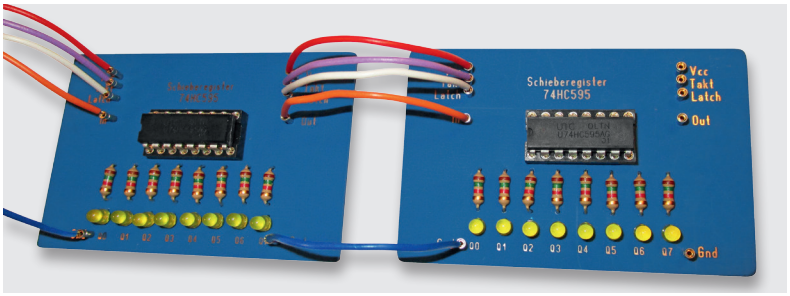
```
' BASCOM-Programm
' Schieberegister  74HC595
' Ansteuerung mit Standardbefehlen (ohne Shiftout oder SPI)
'  In: -
' Out: Takt an B.5, Daten an B.3, Uebernahme an B2
'
$regfile = „M88def.dat“                    'verwendeter Chip
$crystal = 1000000                         'verwendete Frequenz
$hwstack = 40                              'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                              'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 128                           'Parameter (Daten-Laenge), Rechenbereich Funktionen

Takt Alias Portb.5
Config Takt = Output
Daten Alias Portb.3
Config Daten = Output
Uebernahme Alias Portb.2
Config Uebernahme = Output
Dim I As Byte                              'Laufvariable
Dim Ausgabemuster As Byte                  'Ausgabemuster
Waitms 50

Takt = 0
Uebernahme = 0

Ausgabemuster = &B0000_0101                ' &B0000_0101 ist das gleiche wie &B00000101
                                           '   aber für Menschen besser lesbar
Do
  Gosub Ausgeben                           'Ausgabe an Schieberegister
Loop
End

Ausgeben:
For I = 7 To 0 Step -1                     'alle Bits in das Schieberegister schieben.
  Daten = Ausgabemuster.i
  Takt = 1                                 'positive Flanke des Taktsignals ..
                                           '.. schiebt eine Position weiter
  Takt = 0                                 'dann wieder auf 0
Next

Uebernahme = 1                             'positive Flanke an Uebernahme ...
                                           ' .. gibt Daten aus Schieberegister an Ausgaberegister
Uebernahme = 0                             'dann wieder auf 0
Return
```



Erläuterungen:
Zunächst werden die benötigten Ausgabepins für Taktsignal, einzuschiebende Daten und Datenübernahme mit Aliasnamen versehen und konfiguriert.

Den Byte-Variablen Ausgabemuster wird ein Bitmuster zugewiesen, welches in das Schieberegister geschoben werden soll. In der Hauptschleife wird dieses Bitmuster immer wieder ausgegeben. (In diesem Beispiel wäre eigentlich keine Schleife nötig, da ein einmaliges Hinausschieben reichen würde, aber das Programm soll als Basis für wechselnde Ausgaben dienen.) Das eigentliche Ansprechen des Schieberegisters erfolgt in dem Unterprogramm „Ausgeben". In diesem Unterprogramm werden die einzelnen Bits der Variablen Ausgabemuster in einer FOR-NEXT-Schleife durchlaufen. Jedes Bit der Variablen Ausgabemuster wird am Ausgabepin (Portb.3; Alias Daten) ausgegeben. Dann wird das Taktsignal kurz auf 1 gesetzt, was die steigende Flanke zur Datenübernahme in das Schieberegister und das Weiterschieben der bereits im Schieberegister vorhandenen Bits bedeutet. Der Taktpin wird dann wieder auf 0 gesetzt. Nachdem in der FOR-NEXT-Schleife auf diese Weise alle 8 Bits hinausgeschoben wurden, wird eine 1 an den Übernahme-Pin (Portb.2; Alias Uebernahme) gelegt, was der steigenden Flanke entspricht, die bewirkt, dass die Daten des Schieberegisters in das Ausgaberegister des 74HC595 übernommen werden.

In Bild 2 sieht man den zeitlichen Ablauf der Signale. Das Bild wurde aufgenommen mit dem Logikanalysator LogiScope von OSCIUM (ELV J8-11 53 40), der ein sehr nützliches Hilfsmittel ist, um Signalverläufe sichtbar zu machen. Mit diesem Logikanalysator können bis zu 16 Signalverläufe dargestellt werden. Das LogiScope wird an ein iOS-Gerät (iPad, iPhone) angeschlossen, welches auch als Ausgabegerät dient. Durch eine übersichtliche, kostenlose App lässt sich das LogiScope sehr gut bedienen. Bild 2 zeigt die 8 Takte (rot), den Datenlevel, der zum Zeitpunkt der steigenden Taktflanke in das Schieberegister übernommen wird (gelb) und das Übernahmesignal für die Übernahme in das Ausgaberegister (blau).



Nach dem gleichen Prinzip sind mehrere Schieberegister hintereinander ansteuerbar. Dabei werden +5 V, Takt, Latch und GND aller Schieberegister verbunden und der Ausgang eines Schieberegisters (Data out, Pin 9) wird jeweils an den Eingang des nachfolgenden Schieberegisters (Data in, Pin 14) angeschlossen. Bild 3 zeigt die Signalverläufe bei Ansteuerung von zwei hintereinander geschalteten 8-Bit-Schieberegistern vom Typ 74HC595. Man sieht die 16 Takte (rot), in der Mitte die Daten (gelb) und das Übernahmesignal (blau). Mit zwei Schieberegistern kann man auf diese Weise 16 Ausgabepins erhalten, die durch nur drei Pins des Mikrocontrollers angesteuert werden. Ebenso ist es möglich, weitere Schieberegister anzuhängen. Am Mikrocontroller werden immer nur die drei Pins (Takt, Daten, Übernahme) benötigt. Man spricht von einer „Porterweiterung durch Schieberegister".

Statt einer Byte-Variablen (8 Bits) kann man bei mehreren Schieberegistern eine Word-Variable (16 Bits) oder eine Dword-Variable (32 Bits) benutzen, um die einzelnen Bits kompakt zu speichern. Schieberegister kann man auch mit dem BASCOM-Befehl SHIFTOUT oder mit SPI-Befehlen (siehe nächste Seite) ansteuern.

Bei dem beschriebenen 74HC595 spricht man von einem SIPO-Schieberegister (= Seriell In – Parallel Out). Es gibt auch Schieberegister, bei denen Eingabedaten parallel anliegen und seriell (an den Mikrocontroller) ausgegeben werden. 74HC195 (J8-02 72 98) oder 74HC597 sind Vertreter dieser sogenannten PISO-Schieberegister (= Parallel In – Seriell Out). Beide Arten von Schieberegistern können (auch gemischt) mit BASCOM angesteuert werden.

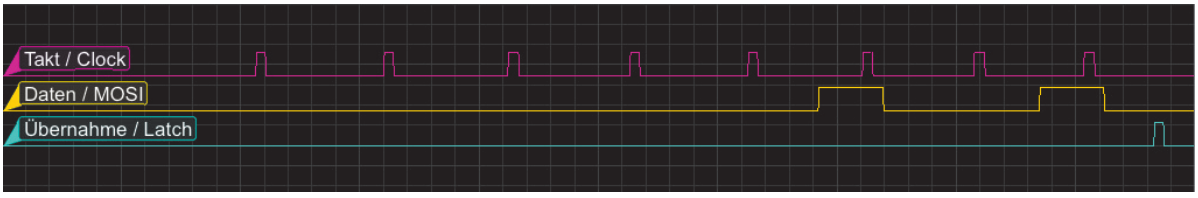


Bild 2: Signalverlauf 8-Bit-Schieberegister

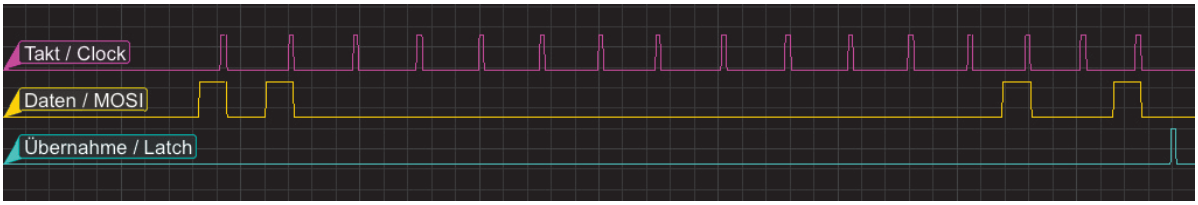


Bild 3: Signalverlauf 16-Bit-Schieberegister

SPI

Es gibt bei SPI immer einen Master, der die Kommunikation anstößt und auch den (gemeinsamen) SPI-Takt erzeugt, und einen (oder mehrere) Slaves (Bild 4). Ganz wichtig für das Verständnis von SPI ist, dass es ein Schieberegister im Master und ein Schieberegister im Slave gibt, die miteinander wie in Bild 4 verbunden sind. Wenn nun acht Bits (= ein Byte) vom Master zum Slave übertragen werden sollen, dann werden die Bits bitweise am Datenausgang des Masters hinausgeschoben und am Dateneingang des Slaves hineingeschoben. Sowohl im Master als auch im Slave werden die Bits mit jedem Takt um eine Speicherzelle weitergeschoben. Außerdem wird mit jedem Takt jeweils ein Bit aus dem Slave hinaus und in den Master hinein geschoben. Das alles geschieht gleichzeitig. Wenn also nach 8 Takten acht Bits aus dem Master hinausgeschoben wurden, dann sind die acht Bits nach diesen acht Takten im Slave und die acht Bits, die vorher im Slave gespeichert waren, sind dann im Master. Es wird gleichzeitig ein Byte vom Master zum Slave geschrieben und ein Byte vom Slave zum Master übertragen! Man spricht daher von einer Voll-Duplex-Verbindung.

Die Bezeichnungen der Anschlüsse variieren je nach Hersteller. Der Taktanschluss heißt meistens SCK (Serial Clock) oder einfach nur Clock. Der Datenausgang beim Master und der Dateneingang beim Slave wird meistens mit MOSI (Master Out – Slave In) oder beim Slave mit SDI (Serial Data In) bezeichnet. Der Dateneingang beim Master und der Datenausgang des Slaves wird meistens mit MISO (Master In – Slave Out) oder beim Slave teils auch mit SDO (Serial Data Out) bezeichnet. Die Übernahmeleitung wird mit Slave Select oder Chip Select bezeichnet. Hintergrund für diese beiden Bezeichnungen ist, dass man auch mehrere verschiedene Slaves an einen Master (sternförmig) anschließen könnte. Mit Slave Select bzw. Chip Select kann man dann den anzusprechenden Baustein auswählen.

Damit die Datenübertragung der einzelnen Bits nicht jedes Mal „von Hand" geschrieben werden muss, bietet BASCOM für SPI drei verschiedene Befehle an: SPIOUT, SPIIN und SPIMOVE (Bild 5). Mit SPIOUT werden Daten vom Master zum Slave übertragen. Es ist dabei nicht von Interesse, Daten vom Slave wieder in den Master zurück zu übertragen – wie es ja eigentlich IMMER bei SPI gleichzeitig geschieht. Man kann die Rückleitung sogar weglassen. Für obiges Porterweiterungsbeispiel mit einem 74HC595-Schieberegister reicht SPIOUT vollkommen aus. Wenn man nur daran interessiert ist, Daten von einem Slave zum Master zu übertragen, dann kann man den BASCOM-Befehl SPIIN verwenden. Für die gleichzeitige (!) Übertragung in beide Richtungen wird der BASCOM-Befehl SPIMOVE verwendet. Häufig wird dem Slave im ersten Byte, das vom Master kommt, mitgeteilt, welche Daten ab dem zweiten zum Master übertragenen Byte erwartet werden. Das erste vom Slave zum Master übertragene Byte kann dann verworfen werden. Die Nutzdaten nimmt sich der Master ab dem zweiten Byte. Siehe unten bei der Ansteuerung des 3D-BS.

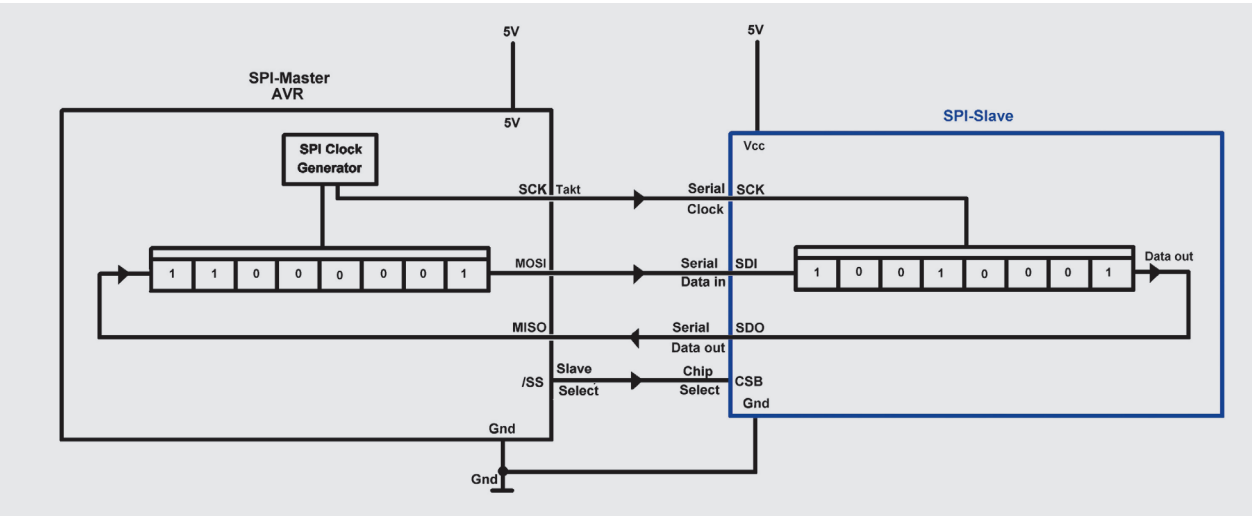


Bild 4: SPI-Prinzip

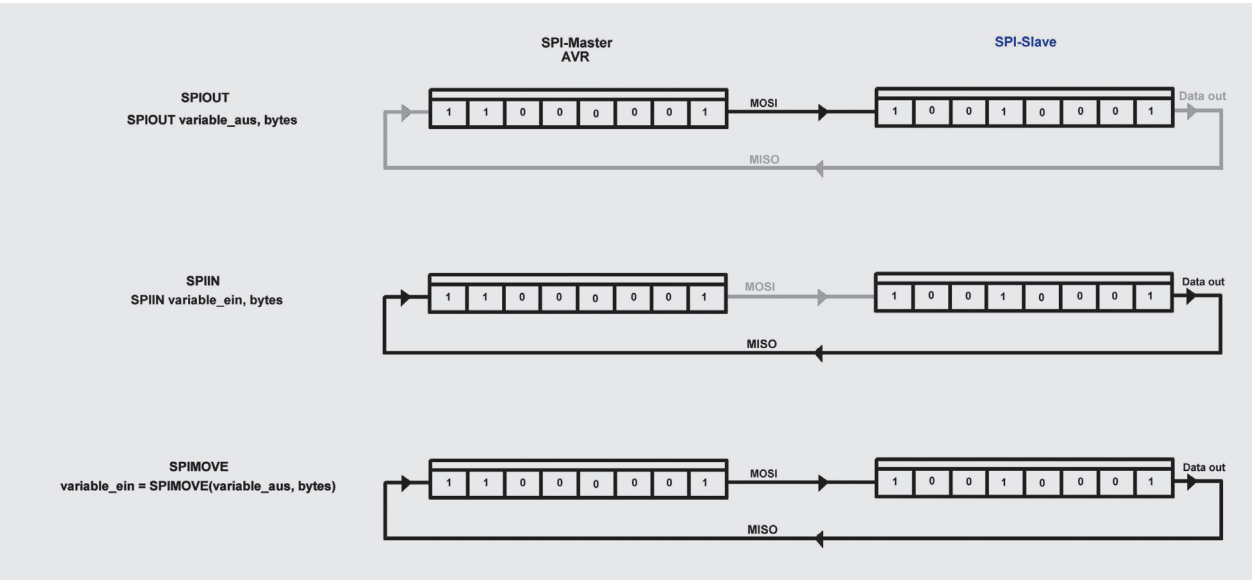


Bild 5: SPIOUT/ SPIIN/SPIMOVE mit BASCOM-Syntax

SPI-Ansteuerung des Schieberegister 74HC595

Im folgenden BASCOM-Programm werden per SPIOUT Bytes zu einem 74HC595 übertragen.

```
' BASCOM-Programm
' Schieberegister  74HC595
' Ansteuerung mit HARDWARE-SPI
'  In: -
' Out: Takt an B.5, Daten an B.3, Uebernahme an B2
$regfile = „M88adef.dat“                    'verwendeter Chip
$crystal = 1000000                          'verwendete Frequenz
$hwstack = 40                               'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                               'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 128                            'Parameter (Daten-Laenge), Rechenbereich Funktionen

'(
Takt Alias Portb.5
Config Takt = Output                       bei Hardware-SPI werden die DDR-Register automatisch gesetzt!
Daten Alias Portb.3
Config Daten = Output
Uebernahme Alias Portb.2
Config Uebernahme = Output
')

Config Spi = Hard , Master = Yes , Data_order = Msb
Spiinit                                     'SPI-Bus initialisieren; setzt die Pins auf Eingabe bzw. Ausgabe

Dim I As Byte                               'Laufvariable
Dim Ausgabemuster As Byte                   'Ausgabemuster

Ausgabemuster = &B0000_0000                 'alles dunkel
Gosub Ausgeben
Wait 2

Ausgabemuster = &B1110_1100
Gosub Ausgeben
Wait 1

Ausgabemuster = &B1000_0000                 'Anfangsmuster
Do
Gosub Ausgeben                              'Ausgabe an Schieberegister
Waitms 100
Rotate Ausgabemuster , Right , 1            'um eins nach rechts schieben
Loop
End

Ausgeben:
Spiout Ausgabemuster , 1                    ' <<---- 1 Byte per SPI ausgeben.
Return
```

Erläuterungen:

Mit dem Befehl CONFIG SPI wird dem Compiler mitgeteilt, dass der Mikrocontroller als Master agieren soll. SPI = Hard weist den Compiler an, die **Hardware-SPI**-Einheit des AVR-Mikrocontrollers zu verwenden. Dadurch sind automatisch auch die Pins festgelegt, die für SPI verwendet werden. Sie sind dem Pinlayout des verwendeten Mikrocontrollers zu entnehmen. Es gibt auch die Möglichkeit, Software-SPI zu verwenden, wobei dann die Pins frei gewählt werden können. Durch den Befehl SPIINIT wird der SPI-Bus initialisiert. Bei Hardware-SPI werden durch SPIINIT auch die I/O-Pins konfiguriert.

Wie im obigen Schieberegister-Beispiel wird nun ein Bitmuster einer Byte-Variablen zugewiesen und durch Aufruf des Unterprogramms „Ausgeben“ an das Schieberegister ausgegeben. Wie man sieht, ist das Unterprogramm durch die Verwendung des BASCOM-Befehls SPIOUT auf einen einzigen Befehl verkürzt. Man muss keine Flanken selbst erzeugen usw. Für SPIOUT muss nur angegeben werden, wie viele Bytes vom Master zum Slave geschrieben werden sollen und in welchem Speicherbereich (Variablen) die zu übertragenden Bits stehen.

Soft-SPI

Die Verwendung von SOFT-SPI ist fast identisch. Lediglich der Konfigurationsbefehl ist anders und die I/O-Pins werden selbst konfiguriert. Sie sind frei wählbar.

```
' BASCOM-Programm
' Schieberegister  74HC595
' Ansteuerung mit SOFT-SPI
'  In: -
' Out: Takt an B.5, Daten an B.3, Uebernahme an B2
$regfile = „M88adef.dat“                    'verwendeter Chip
$crystal = 1000000                          'verwendete Frequenz
$hwstack = 40                               'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                               'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 128                            'Parameter (Daten-Laenge), Rechenbereich Funktionen
```

```
Takt Alias Portb.5
Config Takt = Output
Daten Alias Portb.3
Config Daten = Output
Uebernahme Alias Portb.2
Config Uebernahme = Output

Config Spi = Soft , Din = Pinb.4 , Dout = Daten , Ss = Uebernahme , Clock = Takt , Mode = 0
Spiinit                                 'SPI-Bus initialisieren; setzt die Pins auf Eingabe bzw, Ausgabe

Dim I As Byte                           'Laufvariable
Dim Ausgabemuster As Byte               'Ausgabemuster

Ausgabemuster = &B0000_0000             'alles dunkel
Gosub Ausgeben
Wait 2

Ausgabemuster = &B1110_1100             'ein Muster ausgeben
Gosub Ausgeben
Wait 1

Ausgabemuster = &B1000_0000             'Anfangsmuster
Do
Gosub Ausgeben                          'Ausgabe an Schieberegister
Waitms 100
Rotate Ausgabemuster , Right , 1        'um eins nach rechts schieben
Loop
End

Ausgeben:
Spiout Ausgabemuster , 1                ' <<---- ein Byte über SPI ausgeben
Return
```

Erläuterungen:
Nach der Definition der Ausgabepins (inklusive Aliasnamenvergabe) wird dem Compiler mit dem CONFIG-Befehl mitgeteilt, dass **Software-SPI** verwendet werden soll. Als Konsequenz werden dann im CONFIG-Befehl die Pins für SPI angegeben. Din steht für Data in (= MISO) und wird in diesem Beispiel zwar nicht benutzt, aber dennoch konfiguriert. Dout, Ss und Clock stehen für MOSI, Slave Select und Takt und werden den Aliasnamen der entsprechenden Pins zugewiesen.

Ansteuerung 3-Achsen-Beschleunigungssensor 3D-BS mit SPI

Das ELV-Modul 3D-BS (J8-09 15 21 bzw. J8-10 48 93) ist ein 3-Achsen-Beschleunigungssensor, der als Kommunikations-Schnittstelle (unter anderem) SPI unterstützt. Es lassen sich mit dem verbauten Sensor Beschleunigungen bis zu 8g messen, und da auch in Ruhe die Erdbeschleunigung auf den Sensor wirkt, lässt der Sensor auch das Messen von Neigungen zu. Durch integrierte Pegelwandler lässt sich der 3D-BS in einem Spannungsbereich von 2,5 bis 6 V betreiben. Der Anschluss erfolgt bei Nutzung der SPI-Schnittstelle gemäß Bild 6. Außer den Spannungsversorgungsleitungen und der gemeinsamen GND-Verbindung sieht man hier die SPI-Verbindungen MOSI, MISO, SCK und Slave Select/SS. Beim Schieberegister im obigen Beispiel wurde keine MISO-Verbindung genutzt, beim Beschleunigungssensor sollen die Beschleunigungsdaten vom Sensor (Slave) zum Mikrocontroller (Master) übertragen werden.

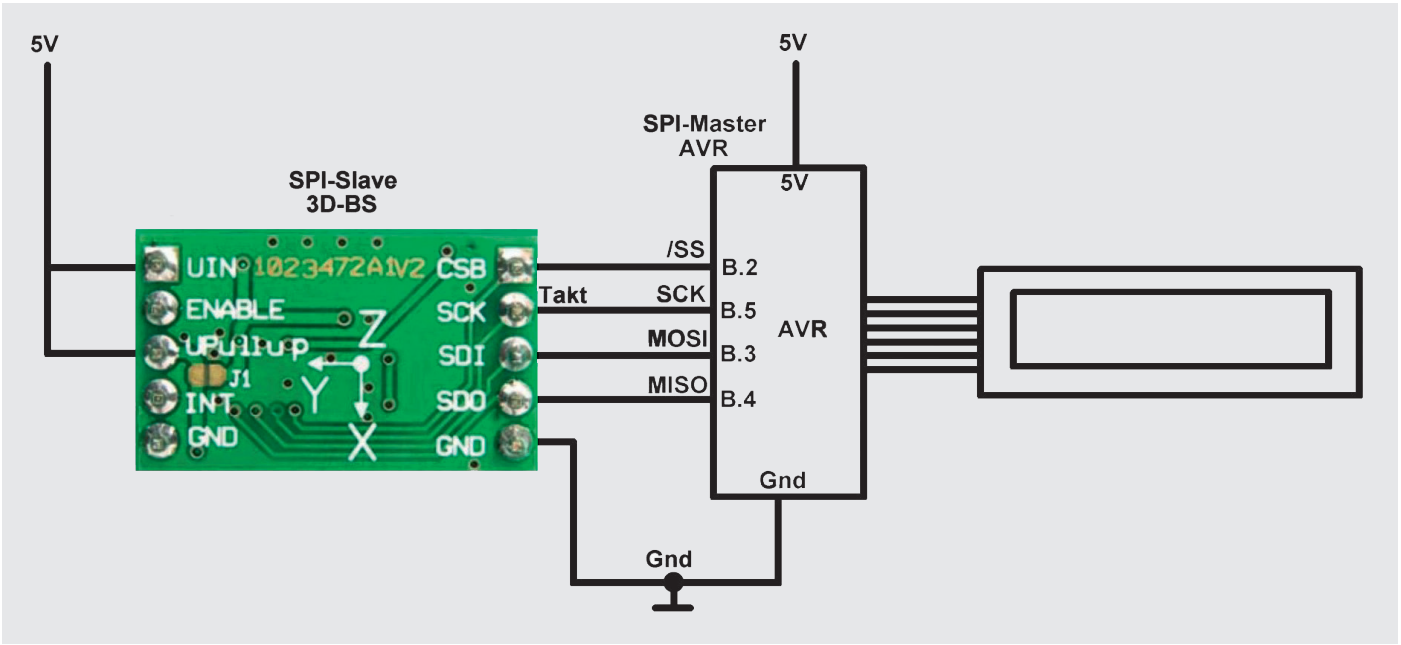


Bild 6: 3D-BS-Anschluss

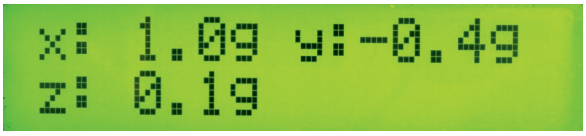
Zur Benutzung des Beschleunigungssensors werden zunächst Konfigurationsdaten vom Master zum Slave übertragen (MOSI) und dann die Beschleunigungswerte vom Slave angefordert und gelesen (MISO).

```
' BASCOM-Programm
' SPI-Ansteuerung des ELV  3DBS  mit BMA020-Sensor
' Ansteuerung mit HARDWARE-SPI   Mit Range-Schreiben
'  In: -
' Out: Takt, Daten, Uebernahme
'
$regfile = „M88adef.dat“                    'verwendeter Chip
$crystal = 1000000                          'verwendete Frequenz
$hwstack = 40                               'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                               'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 128                            'Parameter (Daten-Laenge), Rechenbereich Funktionen

Config Lcdpin = Pin , Db4 = Portd.4 , Db5 = Portd.5 , Db6 = Portd.6 , Db7 = Portd.7 , E = Portd.3 , Rs = Portd.2
Config Lcd = 16 * 2
Cls
Cursor Off
Waitms 100

Config Spi = Hard , Master = Yes , Data_order = Msb , Polarity = High , Phase = 1 , Clockrate = 128
'Bei Hardware-SPI werden die DDR-Register automatisch gesetzt!
Spiinit                                     'SPI-Bus initialisieren; setzt die Pins auf Eingabe bzw, Ausgabe

Dim Vom_sensor(7) As Byte
Dim Zum_sensor(7) As Byte
Dim X As Integer , Y As Integer , Z As Integer
Dim Zahl_string As String * 5
Dim X_g As Single , Y_g As Single , Z_g As Single
Dim Range_register As Byte
Const Range = 2                             'Angabe in g:  2g  4g  8g

Cls
Lcd "ELV"
Lowerline
Lcd "3D-BS"
Wait 1
Cls

'Bereich in Control-Register schreiben
Range_register = Range
Range_register = Range_register / 4                 '  2 -> 0   4-> 1   8-> 2
Shift Range_register , Left , 3                     'drei Stellen/Bits nach links schieben
Zum_sensor(1) = &H14                                'Register  &h14 (Control-Register) beschreiben
Zum_sensor(2) = Range_register                      'Inhalt
Spiout Zum_sensor(1) , 2                            ' <<---- in Register &h14 schreiben. = Control-Register
Zum_sensor(2) = 0                                   'zweites Byte fuer laufende Sensorabfrage auf 0

Do
Zum_sensor(1) = &H82                                'ab Register 2 LESEN
Vom_sensor(1) = Spimove(zum_sensor(1) , 7)          ' <<---- Senden und gleichzeitig empfangen
'Locate 1 , 1 : Lcd Vom_sensor(1)
'Locate 1 , 8 : Lcd Vom_sensor(2)
'Locate 2 , 1 : Lcd Vom_sensor(3)

X = Makeint(vom_sensor(2) , Vom_sensor(3))          'zwei gelesene Bytes zu einer Integerzahl zusammenfassen
Shift X , Right , 6 , Signed                        'ergibt Wert zwischen -512 und +511
Y = Makeint(vom_sensor(4) , Vom_sensor(5))
Shift Y , Right , 6 , Signed                        'ergibt Wert zwischen -512 und +511
Z = Makeint(vom_sensor(6) , Vom_sensor(7))
Shift Z , Right , 6 , Signed                        'ergibt Wert zwischen -512 und +511
'In x, y und z stehen nun die Rohdaten als Integerzahlen zur Verfügung

Locate 1 , 1
'Umrechnen in g  (Erdbeschleunigung)
X = X * Range : X_g = X / 512                       'Rohdaten umrechnen in Erdbeschleunigung x,xx g
Zahl_string = Fusing(x_g , "#.#")                   '.. aufbereiten
Lcd "x:" : If X_g >= 0 Then Lcd " " : Lcd Zahl_string ; "g" '.. und anzeigen

Y = Y * Range : Y_g = Y / 512                       'Rohdaten umrechnen in Erdbeschleunigung x,xx g
Zahl_string = Fusing(y_g , "#.#")                   '.. aufbereiten
Lcd "y:" : If Y_g >= 0 Then Lcd " " : Lcd Zahl_string ; "g" '.. und anzeigen

Lowerline
Z = Z * Range : Z_g = Z / 512                       'Rohdaten umrechnen in Erdbeschleunigung x,xx g
Zahl_string = Fusing(z_g , "#.#")                   '.. aufbereiten
Lcd "z:" : If X_g >= 0 Then Lcd " " : Lcd Zahl_string ; "g" '.. und anzeigen
Waitms 500
Loop
End
```

Erläuterungen:

Dank der BASCOM-SPI-Befehle ist die Ansteuerung des 3D-BS-Beschleunigungssensors recht einfach. Im CONFIG-Befehl wird dem Compiler wiederum mitgeteilt, dass der Mikrocontroller als Master verwendet werden soll. In diesem Beispiel wird wieder Hardware-SPI konfiguriert. Mit den Parametern Polarity und Phase werden zwei SPI-Übertragungsparameter eingestellt, die mit denen des Slaves übereinstimmen müssen (vgl. „Elektronikwissen“).

Im Datenblatt des Bewegungssensors BMA020 [3] zeigt das Diagramm „Figure 4“ auf Seite 24 (Timing-Diagramm), dass das Taktsignal (SCK) im Ruhezustand 1 sein soll. Deshalb wird im BASCOM-Programm die Polarity auf „high“ gesetzt. Im Absatz 4.1.1



auf Seite 23 des Sensordatenblattes ist beschrieben, dass die Datenübernahme von SDI mit der steigenden Flanke und die Datenausgabe an SDO mit der fallenden Flanke erfolgt. Deshalb wird in der BASCOM-Konfiguration die SPI-Phase auf 1 gesetzt. Details sind dem ELV-Fachbeitrag zum USB-SPI-Interface [1] und dem Datenblatt des Sensors [3] zu entnehmen. Nach dem CONFIG-Befehl wird wiederum mit SPIINIT die SPI-Schnittstelle initialisiert. Zur Vorbereitung des Schreibens werden die ersten Elemente des Arrays Zum_Sensor mit Werten belegt. Das Register &h14 ist das Kontrollregister, welches beschrieben werden soll (vgl. Tabelle 1 in der Produktbeschreibung des 3D-BS).

Mit dem Befehl Spiout Zum_sensor(1) , 2 schreibt BASCOM zwei Bytes per SPI zum Slave, nämlich den Inhalt der Variablen Zum_Sensor(1) und Zum_Sensor(2). Bild 7 zeigt den zugehörigen Signalverlauf. Außer der eigentlichen Signaldarstellung bietet der OSCIUM-Logikanalyzer auch die Möglichkeit, verschiedene Protokolle, unter anderem SPI, zu dekodieren. Im unteren Teil von Bild 7 sieht man dadurch sehr schön, dass erst &h14 als Registeradresse und danach &h00 als Parameter zum Slave übertragen werden. Vom Slave zum Master werden in diesem Falle zwei &hFF-Bytes übertragen, die hier im Master nicht zur Auswertung kommen.

In der DO-LOOP-Schleife sollen permanent die Beschleunigungswerte aus dem Sensor gelesen werden, die ab Register 2 im Slave gespeichert sind. Für LESE-Zugriff muss laut Datenblatt [6] das Register &h82 adressiert werden, um ab Register &h02 zu LESEN. Mit Vom_sensor(1) = Spimove(zum_sensor(1) , 7) werden 7 Bytes geschoben!

Bild 8 zeigt den Signalverlauf beim Lesen der Beschleunigungswerte. Auf der MOSI-Leitung wird als erstes Byte die &h82 an den Slave übertragen. Das erste Byte, welches gleichzeitig auf der MISO-Leitung vom Slave zurückkommt, ist irrelevant und hier immer &hFF. Danach sieht man in Bild 8 die Übertragung der Beschleunigungswerte vom Slave zum Master (MISO), während gleichzeitig der Master auf der MOSI-Leitung Nullen an den Slave überträgt.

Die vom Slave empfangenen Daten befinden sich dann im Array Vom_Sensor und müssen nur noch verarbeitet und angezeigt werden.

Wenn einmal das Schieberegister-Prinzip von SPI (Bild 4) verstanden wurde, lassen sich nach dem gezeigten Schema andere SPI-Komponenten analog ansteuern.

Ausblick

Im nächsten Teil der Artikelserie wird betrachtet, wie man BASCOM in Zusammenhang mit Hardware-Plattformen wie Arduino, Asuro, NiboBee oder Raspberry Pi nutzen kann.

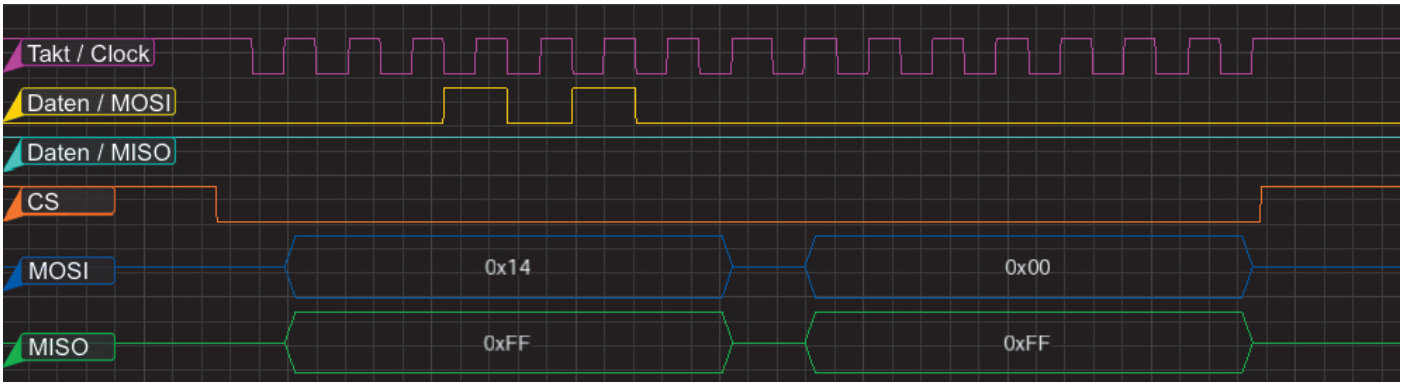


Bild 7: Signalverlauf 3D-BS bei Konfiguration

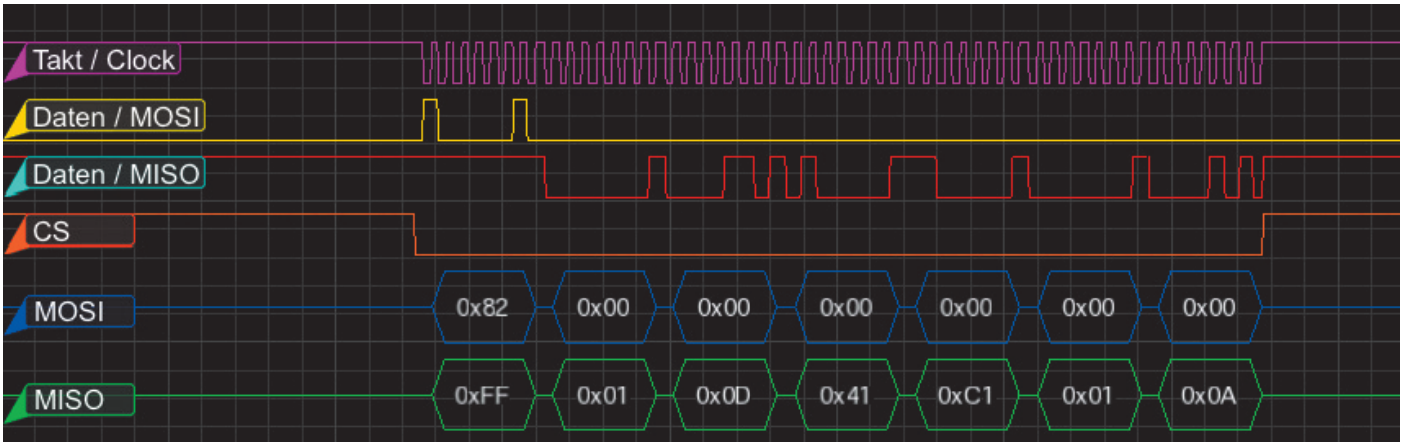


Bild 8: Signalverlauf 3D-BS im Betrieb

SPI-Modi

Motorola hat für die SPI-Kommunikation kein explizites Protokoll festgelegt, aber es hat sich die Unterscheidung von vier Modi (0, 1, 2, 3) durchgesetzt. Die Modi geben an, auf welchem logischen Level das Taktsignal im Ruhezustand sein soll und bei welchen Signalflanken Daten übernommen bzw. ausgegeben werden.

Der Ruhezustand des Taktsignals wird mit CPOL (= Clock Polarität; Polarity) gekennzeichnet. CPOL = 0 bedeutet, dass das Taktsignal im Ruhezustand 0 ist. CPOL = 1 bedeutet einen Ruhezustand des Taktsignals mit Level 1.

Mit CPHA (= Clock Phase, Phase) wird angegeben, ob die Daten bei der ersten Flanke (CPHA=0) oder bei der zweiten Flanke (CPHA=1) übernommen werden. Bei der jeweils anderen Flanke werden die Daten an den Bus ausgegeben. Die Werte sind dem Datenblatt der jeweiligen SPI-Komponente zu entnehmen und im Programm entsprechend zu berücksichtigen.

Mode	CPOL (Takt im Ruhezustand)	CPHA (Datenübernahme bei ...)	Datenübernahme bei ...	Datenausgabe bei ...
0	0	0 (... erster Flanke)	... steigender Flanke	... fallender Flanke
1	0	1 (... zweiter Flanke)	... fallender Flanke	... steigender Flanke
2	1	0 (... erster Flanke)	... fallender Flanke	... steigender Flanke
3	1	1 (.... zweiter Flanke)	... steigender Flanke	... fallender Flanke

Steigende Flanke: Wechsel des Signals von 0 nach 1.
Fallende Flanke: Wechsel des Signals von 1 nach 0.

Weitere Infos:

[1] ELV-Fachbeitrag zu USB-SPI-Interface www.elv.de/controller.aspx?cid=726&detail=45491

[2] Datenblatt 74HC595:
http://files.elv.de/Assets/Produkte/11/1137/113721/Downloads/113721_ic_data.pdf

[3] Datenblatt Beschleunigungssensor BMA020:
http://files.elv.de/Assets/Produkte/9/915/91521/Downloads/91521_bma020_data.pdf

- Stefan Hoffmann: Einfacher Einstieg in die Elektronik mit AVR-Mikrocontroller und BASCOM. Systematische Einführung und Nachschlagewerk mit vielen Anregungen. ISBN 978-3-8391-8430-1
- www.bascom-buch.de
- www.mcselec.com
- Produktübersicht BASCOM: **www.elv.de/bascom.html**

Empfohlene Produkte/Bauteile:	Best.-Nr.	Preis
BASCOM-(Demo-)Lizenz von MCS Electronics, www.mcselec.com	–	–
ATmega88	J8-10 07 62	€ 3,95
BASCOM-Buch	J8-10 90 02	€ 54,–
Schieberegister 74HC595	J8-11 37 21	€ 0,20
Schieberegister 74HC165	J8-02 72 98	€ 0,36
Low-Current-LEDs	J8-05 83 56	€ 0,37
3-Achsen-Beschleunigungssensor 3D-BS, Komplettbausatz	J8-09 15 21	€ 6,95
3-Achsen-Beschleunigungssensor 3D-BS, Fertiggerät	J8-10 48 93	€ 9,95
6-Achsen-Beschleunigungssensor-Modul 6D-BS	J8-13 05 98	€ 21,50
Real-Time-Clock-DCF-Modul	J8-13 05 41	€ 11,95
USB-SPI-Interface	J8-13 12 92	€ 29,95
Oscium Logikanalysator LogiScope	J8-11 53 40	€ 199,95
microSD-Karten-Adapter MSDA1	J8-13 15 91	€ 7,95

Alle Infos zu den Produkten/Bauteilen finden Sie im Web-Shop.

Preisstellung Februar 2015 – aktuelle Preise im Web-Shop