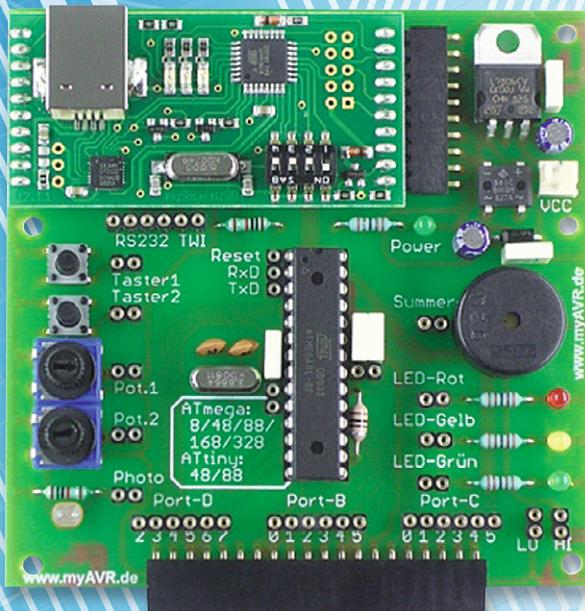




# Mikrocontroller-Einstieg

Teil 13: I<sup>2</sup>C-Lesen (weitere Anwendungen)



```
BASCOM-AVR IDE [2.0.7.5] - [C:\user\BASCOM-Programme\Blinker_attiny13.bas]
Datei Editieren Anzeigen Programmieren Werkzeuge Optionen Fenster Hilfe
Blinker_attiny13.bas
Sub
  ' BASCOM-Programm
  ' Einfacher Blinker
  ' In: -
  ' Out: LED mit Vorwiderstand an PortB.4

$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 4
$swstack = 4
$sfrsize = 10

Config PORTB.4 = Output

Do
  PORTB.4 = 1
  Waitms 500
  PORTB.4 = 0
  Waitms 500
Loop
End

' Verwendeter Chip
' Verwendete Frequenz
' Rücksprungadressen (je 2), Registersicherungen (32)
' Parameterübergaben (je 2), LOCALs (je 2)
' Parameter (Daten-Laenge), Rechenbereich Funktionen
' B.4 als Ausgang definieren

' Schleifenbeginn
' B.4 auf 1
' Wartschleife 500 ms
' B.4 auf 0
' Wartschleife 500 ms
' Schleifenende
' Programmende
```



# mit BASCOM-AVR

In diesem Teil unserer Artikelserie „Mikrocontroller-Einstieg mit BASCOM-AVR“ wird gezeigt, wie der nützliche I<sup>2</sup>C-Bus zur einfachen Ansteuerung komplexer Module genutzt wird. Mit nur zwei Daten-Leitungen plus einer GND-Leitung werden Daten zu einer DCF77-Echtzeituhr geschrieben und auch Daten eingelesen. Des Weiteren wird die Anbindung des ELV 3D-Bewegungssensors und des ELV 6D-Bewegungssensors gezeigt.

## DCF77-Funkuhr

Das ELV Real-Time-Clock-DCF-Modul RTC-DCF (Best.-Nr. J6-13 05 41) bietet eine Echtzeituhr mit DCF77-Empfänger, die sich (unter anderem) über I<sup>2</sup>C leicht in eigene BASCOM-Projekte einbinden lässt. Mit dem integrierten DCF77-Teil des Moduls wird die sekundengenaue Uhrzeit per Funk empfangen und an die ebenfalls integrierte Echtzeituhr übertragen. Die so gewonnene Uhrzeit (mit Datum) kann aus der Echtzeituhr ausgelesen und zum Beispiel für Datenlogger-Anwendungen verwendet werden.

Das Modul wird mit einer Versorgungsspannung von 1,8 bis 3,8 V betrieben. Da Pegelwandler für die Signalleitungen auf dem Modul bereits vorhanden sind, lässt sich die RTC-DCF unter Benutzung eines 3,3-V-Spannungsreglers wie in [Bild 1](#) zu sehen in 5-V-Umgebungen integrieren. Im [Bild 1](#) steht Rpu für die Pull-up-Widerstände in Klammern, da keine externen Pull-up-Widerstände benötigt werden, weil bereits welche auf dem Modul vorhanden und aktivierbar sind. Es soll in [Bild 1](#) lediglich daran erinnert werden, dass bei der Verwendung von I<sup>2</sup>C immer Pull-up-Widerstände benötigt werden. Diese sind beim RTC-DCF-Modul bereits auf dem Modul vorgesehen und müssen daher nicht außerhalb des Moduls angeschlossen werden, sondern können leicht aktiviert werden.

Für die Aktivierung der I<sup>2</sup>C-Pull-up-Widerstände werden bei J1 und J2 Lötbrücken hergestellt. Für die Aktivierung der Pegelwandler für DCF77-Empfang-Interrupt und „Periodischer Interrupt“ sind außerdem die Lötbrücken J3 und J5 zu schließen (vgl. Tabelle 2 in der Produktbeschreibung). Der 8fach-Schiebeschalter S1 wird gemäß Tabelle 1 der Produktbeschreibung (vorletzte Zeile) auf I<sup>2</sup>C eingestellt, indem Schalter 2 auf 1 und Schalter 1 auf 0 gestellt wird. Mit Schalter 3 bis 8 kann man die I<sup>2</sup>C-Slave-Adresse einstellen. An ST3 werden die Anschlüsse für den Sekunden-Interrupt und den DCF-Interrupt angeschlossen ([Bild 1](#)). Vgl. auch Tabelle 2 in der Produktbeschreibung, in der beschrieben wird, dass bei I<sup>2</sup>C-Betrieb die Pins 12 und 10 (von ST3) für den periodischen Interrupt bzw. den DCF77-Empfangs-Interrupt verwendet werden.

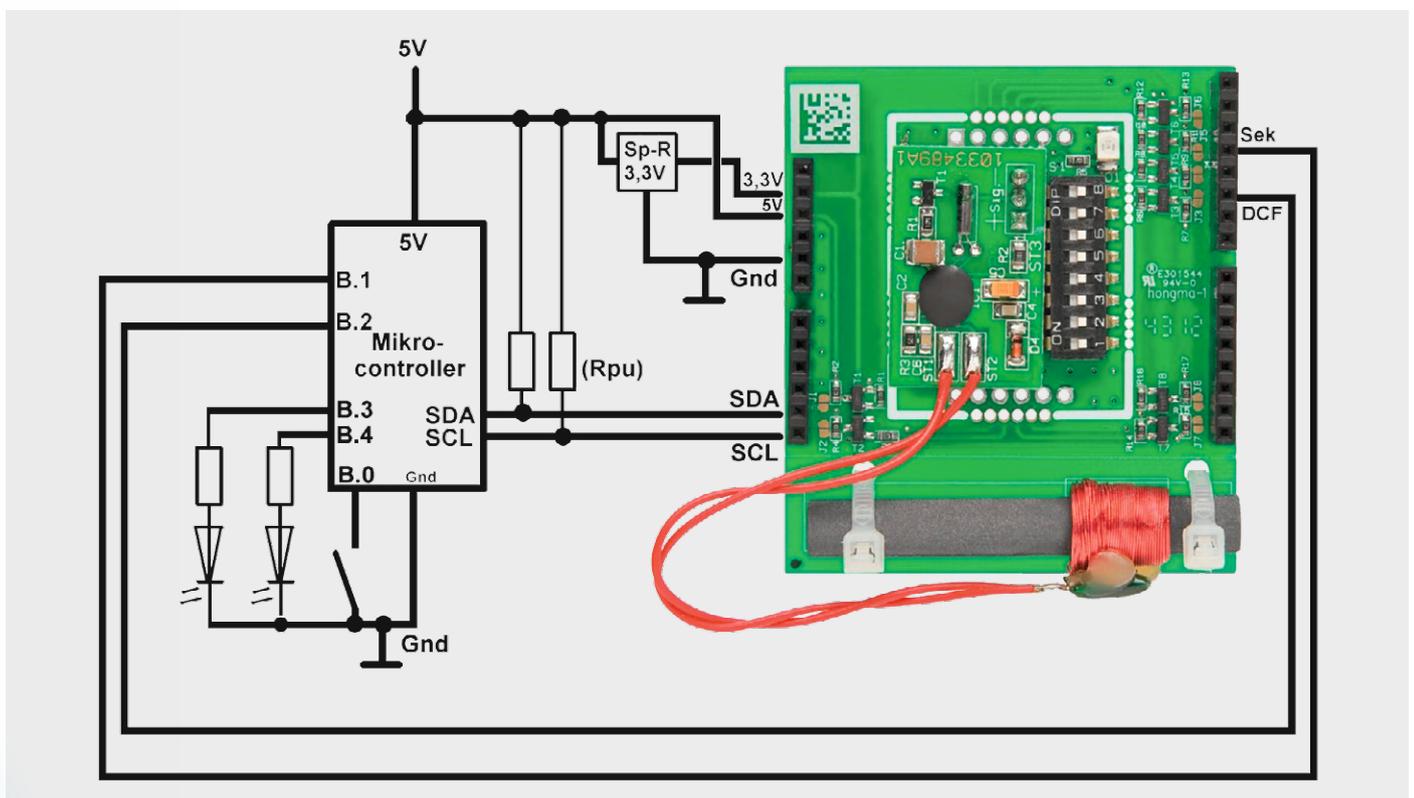


Bild 1: RTC-DCF77-Anschluss (LCD-Anschluss nicht eingezeichnet)



```
' BASCOM-Programm
'
' I2C-Real-Time-Clock mit DCF77-Empfang mit ATmega88
' Bei Tastendruck am Start Schreiben einer Anfangszeit in die RTC, dann Empfang der DCF77-Uhrzeit
'
' In: I2C-Signal von RTC-DCF an C.4=SDA und C.5=SCL
' In: Taster an B.0 für Schreiben einer Initial-Uhrzeit
' In: Interrupt-Signal von RTC-DCF an B.1 im Sekundentakt
' In: Interrupt-Signal von RTC-DCF an B.2, wenn DCF77-Uhrzeit komplett empfangen wurde
' Out: LCD an D.2 bis D.7
' Out: LED an B.3 für Sekundenblitzen
' Out: LED an B.4 für DCF-Empfang-komplett-Anzeige

$regfile = „M88def.dat“           'Verwendeter Chip
$crystal = 3686400                'Verwendete Frequenz
$hwstack = 40                    'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                    'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 60                  '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
Deflcdchar 0 , 32 , 14 , 17 , 32 , 14 , 17 , 32 , 4           ' Symbol für 'Empfang o.k.'
Cls
Cursor Off
Waitms 250

Config Sda = Portc.4
Config Scl = Portc.5

On Pcnt0 Pcnt0_isr           'PCINT0-Sprungziel definieren
Pcmsk0.1 = 1                 'Sekundeninterrupt
Pcmsk0.2 = 1                 'DCF-Empfang komplett Interrupt
Enable Pcnt0
Enable Interrupts

Config Portb.0 = Input       'Taster
Taster Alias Pinb.0         'Aliasnamen Taster für PINB.0 vergeben
Portb.0 = 1                 'Internen Pull-up-Widerstand aktivieren

Config Pinb.1 = Input       'Interrupt vom RTC-DCF bei jeder neuen Sekunde
Sekunden_interrupt Alias Pinb.1
Dim Neue_sekunde As Bit

Config Pinb.2 = Input       'Interrupt vom RTC-DCF, wenn DCF-Zeit komplett empfangen
Dcf_interrupt Alias Pinb.2

Config Portb.3 = Output     'Sekundenblinken
Led Alias Portb.3

Config Portb.4 = Output     'DCF-Empfang
Led2 Alias Portb.4

Const Slaveadresse_schreiben = &H02           'Slave-Adresse RTC-DCF schreiben
Const Slaveadresse_lesen = &H03              'Slave-Adresse RTC-DCF lesen
Dim Slaveadresse As Byte
Dim Stunde As Byte
Dim Minute As Byte
Dim Sekunde As Byte
Dim Stunde_bcd As Byte
Dim Minute_bcd As Byte
Dim Sekunde_bcd As Byte
Dim Seit_dcf As Byte
Seit_dcf = 250           'Am Anfang keine gültige DCF77-Zeit

Cls
Lcd "RTC-DCF"
Lowerline
Lcd "suchen..."
Wait 2
Do
  Locate 1 , 1
  Slaveadresse = Slaveadresse_schreiben           'RTC-DCF
  I2cstart                                       'Startbedingung senden
  I2cwbyte Slaveadresse                         'Adresse senden
  If Err = 0 Then                               'I2C-Slave gefunden?
    Lcd "RTC-DCF gefunden"
    Lowerline
    Lcd "h" ; Hex(slaveadresse) ; " b" ; Bin(slaveadresse) ; " "
    Wait 2
  Else                                         'Kein RTC-DCF gefunden
    Lcd "Kein RTC-DCF "
    Lowerline
    Lcd "suche weiter... "
  End If
  I2cstop                                       'Bus freigeben
Loop Until Err = 0                             'Bis RTC-DCF am Bus gefunden

Cls
Lcd "Betrieb"
Wait 2
```



```

Gosub Rtc_dcf_initialisieren
If Taster = 0 Then
  Stunde = 20
  Minute = 15
  Gosub Zeit_in_rtc_dcf_schreiben
  Cls : Lcd "Initialisiert" : Wait 2
End If

Cls
Lcd "ELV"
Lowerline
Lcd "RTC-DCF"

Do
If Neue Sekunde = 1 Then
  Neue Sekunde = 0
  Set Led
  Waitms 50
  Reset Led

  Gosub Zeit_aus_rtc_dcf_lesen
  Locate 1 , 5
  If Stunde < 10 Then Lcd "0" : Lcd Stunde ; ":"
  If Minute < 10 Then Lcd "0" : Lcd Minute ; ":"
  If Sekunde < 10 Then Lcd "0" : Lcd Sekunde

  If Seit_dcf = 0 Then
    Set Led2 : Waitms 500 : Reset Led2
    Gosub Dcf_interrupt_flag_loeschen
  End If
  If Seit_dcf < 250 Then Incr Seit_dcf
  Locate 2 , 10 : Lcd Seit_dcf ; " "

  Locate 2 , 16
  If Seit_dcf < 240 Then
    Lcd Chr(0)
  Else
    Lcd " "
  End If
  'Hier ggf. weitere Anweisungen für sekundliche Ausführung
End If
'..
'hier ggf. andere Anweisungen für dauernde Ausführung
'...
Loop
End
Point0_isr:
'Interruptgruppe wird ausgelöst, wenn Sekundeninterrupt oder Interrupt für DCF-Zeit komplett
If Sekunden_interrupt = 0 Then Neue Sekunde = 1
If Dcf_interrupt = 0 Then Seit_dcf = 0
Return

Rtc_dcf_initialisieren:
'RTC-DCF konfigurieren
I2cstart
I2cwbyte Slaveadresse_schreiben
I2cwbyte &H0A
I2cwbyte &B00000000
I2cwbyte 0
I2cwbyte &B00000010
'Achtung: PILED funktioniert erst nach Firmware-Update (siehe ELV-Produktseite!)
I2cwbyte &B00000111
I2cstop
Return

Zeit_in_rtc_dcf_schreiben:
'Uhrzeit in RTC schreiben
Stunde_bcd = Makebcd(stunde)
Minute_bcd = Makebcd(minute)
Sekunde_bcd = 0
I2cstart
I2cwbyte Slaveadresse_schreiben
I2cwbyte &H00
I2cwbyte Sekunde_bcd
I2cwbyte Minute_bcd
I2cwbyte Stunde_bcd
I2cstop
Return

Zeit_aus_rtc_dcf_lesen:
'Uhrzeit aus RTC-DCF auslesen
I2cstart
I2cwbyte Slaveadresse_schreiben
I2cwbyte &H00
I2cstart
I2cwbyte Slaveadresse_lesen
I2cwbyte Sekunde_bcd , Ack
I2cwbyte Minute_bcd , Ack
I2cwbyte Stunde_bcd , Nack
I2cstop
Stunde = Makedec(stunde_bcd)
Minute = Makedec(minute_bcd)
Sekunde = Makedec(sekunde_bcd)
Return

```

'Wenn Taster gedrückt ist, dann 20:15 als Zeit einstellen

'Bei jeder neuen Sekunde ...  
'Flag zurücksetzen  
'LED zur Kontrolle im Sekundentakt blinken lassen

'Uhrzeit aus RTC-DCF-Modul auslesen ...  
' ... und anzeigen

'Neuer DCF-Empfang ..  
'... dann Kontroll-LED kurz an  
'... und Flag löschen

'Sekunden seit letztem DCF-Empfang hochzählen  
' und zu Kontrollzwecken anzeigen

'Wenn in letzten 4 Minuten DCF77 empfangen wurde ...  
'... dann DCF o.k. anzeigen

'... sonst nicht

'Interruptgruppe wird ausgelöst, wenn Sekundeninterrupt oder Interrupt für DCF-Zeit komplett  
'Neue Sekunde  
'DCF-Zeit komplett empfangen

'Adresse senden  
'Ab Register A: Interrupt-Config-Register  
'Register A: Ein-Sekunden-Interrupt konfigurieren  
'Register B: Kein Alarm  
'Register C: Sekunden-Interrupt einschalten.  
'Achtung: PILED funktioniert erst nach Firmware-Update (siehe ELV-Produktseite!)  
'D: DCF-LED, DCF-Int, DCF77-Empfang einschalten



'Schreibadresse RTC-DCF  
'Ab Adresse 0 schreiben  
'Sekunde im BCD-Format  
'Minute im BCD-Format  
'Stunde im BCD-Format

'Schreibadresse RTC-DCF  
'Ab Adresse 0

'Leseadresse RTC-DCF  
'Sekunde im BCD-Format lesen  
'Minute im BCD-Format lesen  
'Stunde im BCD-Format lesen



```

Dcf_interrupt_flag_loeschen:
'In RTC-DCF das DCF-Interrupt-Flag wieder zurücksetzen
I2cstart
I2cwbyte Slaveadresse_schreiben          'Slave-Adresse senden
I2cwbyte &H0E                             'Ab Register E: Status-Register
I2cwbyte &B00000001                       'Rücksetzen des DCF-Flags
I2cstop
Return

```

#### Erläuterungen:

Nach diversen Grundeinstellungen bzw. Dimensionierungen wird in der Routine `Rtc_dcf_initialisieren` zunächst das Verhalten des Moduls definiert, indem die Konfigurationsregister per I<sup>2</sup>C beschrieben werden. Das geschieht nach dem üblichen I<sup>2</sup>C-Schema: Nach einem I2CSTART wird die Slave-Schreibadresse und dann die Adresse des ersten zu beschreibenden Registers gesendet. Danach werden die Werte für die Register gesendet und die Kommunikation mit I2CSTOP beendet.

Falls beim Start des Programms der Taster gedrückt wird (IF Taster = 0), dann wird eine Initial-Uhrzeit in die Echtzeituhr des Moduls geschrieben. Wieder ist das Schema: I2CSTART – Slave-Adresse schreiben – Registernummer schreiben – Werte schreiben – I2CSTOP. Eine DCF77-Uhrzeit wurde bis hierher noch nicht empfangen.

In der Hauptschleife (DO-LOOP) wird bei jeder neuen Sekunde die Uhrzeit aus der Echtzeituhr gelesen und auf dem LC-Display angezeigt.

Bei jeder neuen Sekunde oder bei einem kompletten DCF-Empfang wird jeweils ein Signal vom Modul erzeugt, das einen Interrupt der PCINT0-Gruppe auslöst. In der Interrupt-Routine wird ausgewertet, welches der beiden hier verwendeten Ereignisse stattgefunden hat. Bei einem Sekunden-Interrupt wird ein Flag gesetzt, das in der Hauptschleife ausgewertet wird. Wenn ein Interrupt empfangen wurde, der anzeigt, dass ein DCF-Signal komplett empfangen wurde, dann wird ein entsprechender Zähler zurückgestellt. Dadurch kann in der Hauptschleife ermittelt werden, wie lange kein DCF-Signal empfangen worden ist. Zu Kontrollzwecken wird in der zweiten LCD-Zeile angezeigt, für wie viele Sekunden KEIN korrekter DCF-Empfang erfolgt ist. Die Uhrzeit wird dennoch aus der Echtzeituhr ausgelesen. Wenn in den vergangenen 240 Sekunden ein DCF-Empfang erfolgt ist, dann wird ein Kontrollsignal unten rechts im LC-Display angezeigt. Dadurch weiß man sofort, ob die Uhrzeit funkuhrgenau ist oder „nur“ aus der Echtzeituhr stammt. In einer praktischen Anwendung müsste man selbstverständlich nicht unbedingt sekundlich die Uhrzeit (und das Datum) auslesen, sondern könnte dies auch zum Beispiel einmal pro Nacht tun und die Uhrzeit den Rest der Zeit mit einer mikrocontroller-internen Uhr weiterlaufen lassen.

## Beschleunigungen messen mit 3D-Bewegungssensor

In einigen Projekten möchte man ermitteln, in welche Richtung ein Objekt gehalten wird oder wie sich ein Objekt in eine Richtung bewegt. Jedes Smartphone hat Sensoren für derartige Informationen eingebaut. Dadurch lassen sich zum Beispiel elektronische Wasserwaagen realisieren, Alarmanlagen können bei Bewegung einen Alarm auslösen, Geräte können bei Bewegung aus einem Stromspar-Modus geweckt werden oder im Modellbau können Lagen erkannt werden. Mit dem 3-Achsen-Beschleunigungssensor 3D-BS (Best.-Nr. J6-10 48 93 bzw. J6-09 15 21) können derartige Lage- bzw. Beschleunigungswerte ermittelt werden. Die Werte für die drei Achsen x, y und z können über die I<sup>2</sup>C-Schnittstelle aus dem Modul ausgelesen und dann im BASCOM-Programm verwendet werden. Dabei ist zu beachten, dass auch im Ruhezustand eine Erdbeschleunigung von 1g senkrecht in Richtung Erdmittelpunkt wirkt. Mit dem 3D-BS-Modul kann man jederzeit ermitteln, in welche Richtung diese Erdbeschleunigung auf das Modul wirkt. Der Anschluss des Moduls ist sehr einfach und erfolgt gemäß Bild 2. Außer der Versorgungsspannung und einer gemeinsamen GND-Leitung sind lediglich die beiden I<sup>2</sup>C-Leitungen SDA und SCL nötig. Das Modul kann mit 2,5 bis 6 V betrieben werden. Ein Spannungsregler und Pegelwandler mit Pull-up-Widerständen für die Daten-Leitungen befinden sich auf dem Modul. Die I<sup>2</sup>C-Slave-Adresse ist mit &h70 fest vorgegeben.

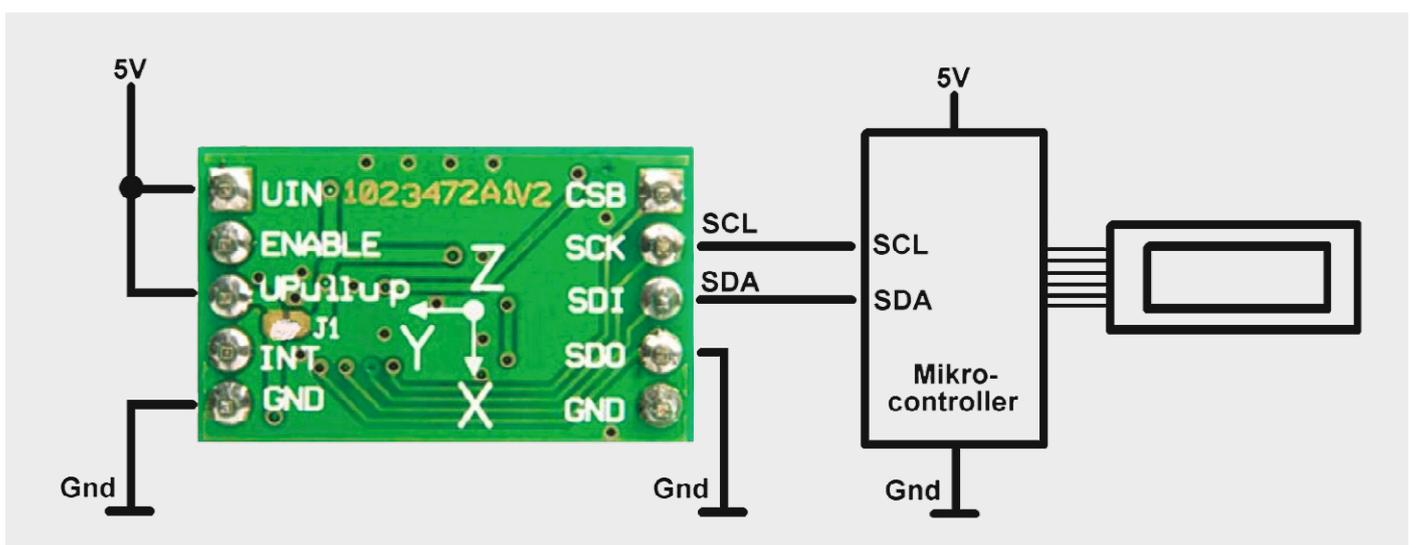
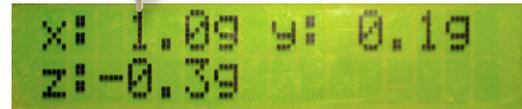
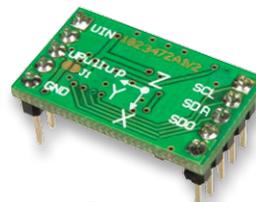
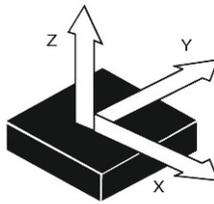


Bild 2: Anschluss des 3-Achsen-Beschleunigungssensor-Moduls 3D-BS



```

' BASCOM-Programm
'
' 3D-Beschleunigungssensor mit ATmega88
' Sensor: BMA020
'
' In: 3D-Beschleunigungssensor an C.4=SDA und C.5=SCL
' Out: Lcd an D2 bis D.7
$regfile = „M88def.dat“
$crystal = 3686400
$hwstack = 40
$swstack = 40
$framesize = 60

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
Deflcdchar 0 , 32 , 14 , 17 , 32 , 14 , 17 , 32 , 4
Cls
Cursor Off
Waitms 250

Config Sda = Portc.4
Config Scl = Portc.5
Const 3d_bs_schreibadresse = &H70
Const 3d_bs leseadresse = &H71

Dim Sensordaten(6) 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

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

'Bereich in Control-Register schreiben
Range_register = Range
Range_register = Range_register / 4 '2->0 (=00) 4->1 (=01) 8->2 (=10)
Shift Range_register , Left , 3
I2cstart
I2cwbyte 3d_bs_schreibadresse
I2cwbyte &H14
I2cwbyte Range_register
I2cstop

Do
I2cstart
I2cwbyte 3d_bs_schreibadresse
I2cwbyte &H02
I2cstart
I2cwbyte 3d_bs leseadresse
I2cwbyte Sensordaten(1) , Ack
I2cwbyte Sensordaten(2) , Ack
I2cwbyte Sensordaten(3) , Ack
I2cwbyte Sensordaten(4) , Ack
I2cwbyte Sensordaten(5) , Ack
I2cwbyte Sensordaten(6) , Nack
I2cstop

X = Makeint(sensordaten(1) , Sensordaten(2))
Shift X , Right , 6 , Signed
Y = Makeint(sensordaten(3) , Sensordaten(4))
Shift Y , Right , 6 , Signed
Z = Makeint(sensordaten(5) , Sensordaten(6))
Shift Z , Right , 6 , Signed
'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
Zahl_string = Fusing(x_g , "#.#")
Lcd "x:" : If X_g >= 0 Then Lcd " " : Lcd Zahl_string ; "g"

Y = Y * Range : Y_g = Y / 512
Zahl_string = Fusing(y_g , "#.#")
Lcd "y:" : If Y_g >= 0 Then Lcd " " : Lcd Zahl_string ; "g"

Lowerline
Z = Z * Range : Z_g = Z / 512
Zahl_string = Fusing(z_g , "#.#")
Lcd "z:" : If Z_g >= 0 Then Lcd " " : Lcd Zahl_string ; "g"

Waitms 500
Loop
End

```

'Verwendeter Chip  
'Verwendete Frequenz  
'Rücksprungadressen (je 2), Registersicherungen (32)  
'Parameteruebergaben (je 2), LOCALS (je 2)  
'Parameter (Daten-Laenge), Rechenbereich Funktionen

' Symbol für 'Empfang o.k.'

'fest! Also nur ein Sensor je I2C-Bus.

'Empfindlichkeit. Mögliche Werte 2 oder 4 oder 8

'Für Position im Register &h14

'In Register &h14 = Control-Register

'Schreibadresse  
'ab Register &h02

'Leseadresse  
'Vgl. Tabelle 1 in Artikelbeschreibung

'Zwei gelesene Bytes zu einer Integerzahl zusammenfassen  
'ergibt Wert zwischen -512 und +511

'ergibt Wert zwischen -512 und +511

'ergibt Wert zwischen -512 und +511

'Rohdaten umrechnen in Erdbeschleunigung x, yg  
'... aufbereiten  
'... und anzeigen

'Rohdaten umrechnen in Erdbeschleunigung x, yg  
'... aufbereiten  
'... und anzeigen

'Rohdaten umrechnen in Erdbeschleunigung x, yg  
'... aufbereiten  
'... und anzeigen

### Erläuterungen:

Das Auslesen der Beschleunigungswerte für die x-, y- und z-Achse mit BASCOM ist sehr einfach. Zunächst wird in das Register &h14 die Empfindlichkeit (Range) geschrieben. Die Range gibt an, ob im Bereich  $-2g$  bis  $+2g$  (Range 2),  $-4g$  bis  $+4g$  (Range 4) oder  $-8g$  bis  $+8g$  (Range 8) gemessen werden soll. Durch Division durch 4 ergeben sich die drei möglichen Werte 0, 1 oder 2, die nach dreimaligem Linksschieben an die richtige Position in das entsprechende Register &h14 im Beschleunigungssensor geschrieben werden. Siehe auch Tabelle 3 im Datenblatt [1] des Beschleunigungssensors BMA020.

In der Hauptschleife werden permanent die Werte ab Register &h02 eingelesen. Ab Register &h02 findet man im Beschleunigungssensor die Beschleunigungswerte für die drei Achsen. Vgl. Tabelle 1 in der ELV-Produktbeschreibung. Die Rohdaten werden zunächst in ein Byte-Array `Sensordaten()` eingelesen und dann mit `MAKEINT` in Integerzahlen umgewandelt, um 6 Stellen nach rechts verschoben und nach Korrektur um den Range-Wert auf dem LC-Display angezeigt.

Weitere Aktionen können dann entsprechend der eigenen Applikation erfolgen.

## Beschleunigungen und Rotationen messen mit 6D-Bewegungssensor

Falls außer den Beschleunigungswerten der 3 Achsen x, y und z auch Rotationen gemessen werden sollen, dann eignet sich das 6-Achsen-Bewegungssensor-Modul 6D-BS (Best.-Nr. J6-13 05 98). 6D steht für 6 Dimensionen für die drei Achsen x, y und z sowie jeweils Rotationen um die x-, y- und z-Achse. Der Beschleunigungssensor für die x-, y- und z-Achse ist mit dem Pin `SDO_A` für zwei verschiedene Slave-Adressen konfigurierbar. Der Rotationssensor ist mit dem Pin `SDO_G` ebenfalls für zwei verschiedene Slave-Adressen konfigurierbar. Beide Sensoren (Beschleunigungssensor und Rotationssensor) sind intern an einem I<sup>2</sup>C-Bus und daher über die gemeinsamen Leitungen SCL und SDA ansprechbar.

Der Anschluss dieses Moduls ist sehr einfach – wie in Bild 3 zu sehen ist. Das Modul ist für Betriebsspannungen von 2,5 bis 6 V ausgelegt, und dank integrierter Pegelwandler mit Pull-up-Widerständen ist die I<sup>2</sup>C-Kommunikation mit 3-V-, 3,3-V- oder 5-V-Systemen einfach möglich.

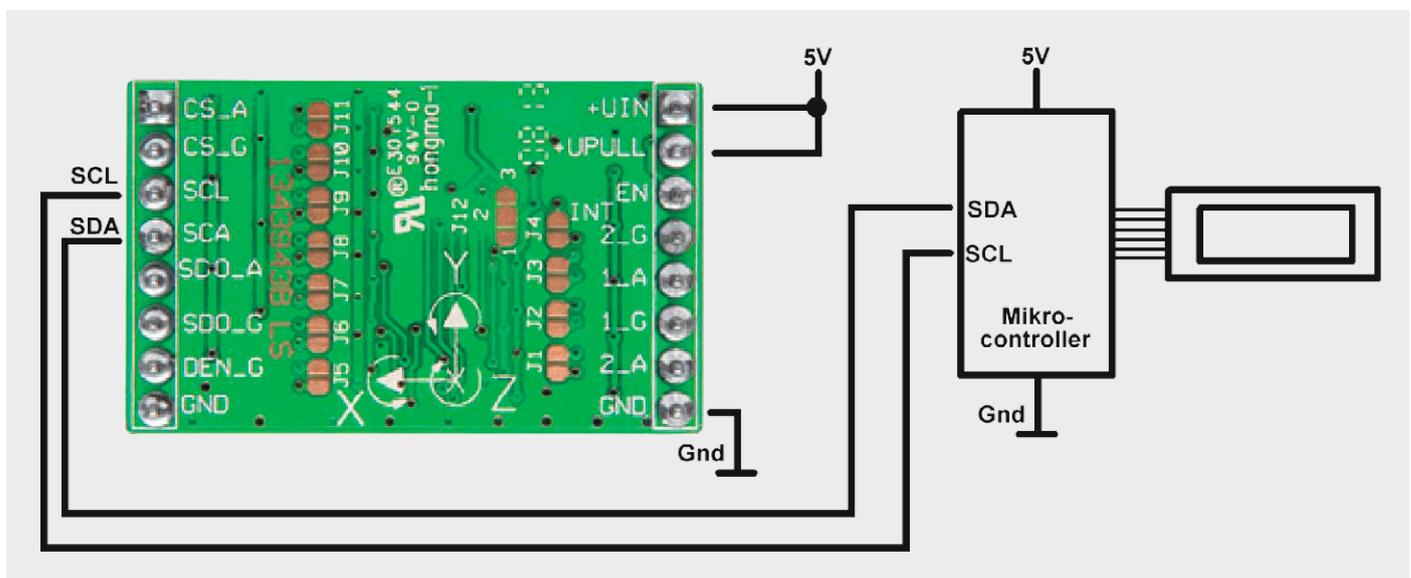
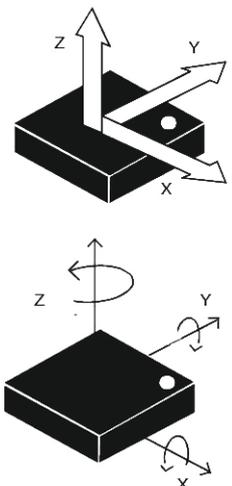


Bild 3: Anschluss des 6-Achsen-Bewegungssensor-Moduls 6D-BS

```
' BASCOM-Programm
'
' 6D-Beschleunigungssensor mit ATmega88
' Sensor: LSM330
'
' In: 6D-Beschleunigungssensor an C.4=SDA und C.5=SCL
' Out: Lcd an D2 bis D.7
$regfile = „M88def.dat“
$crystal = 3686400
$hwstack = 40
$swstack = 40
$framesize = 60
'Verwendeter Chip
'Verwendete Frequenz
'Rücksprungadressen (je 2), Registersicherungen (32)
'Parameteruebergaben (je 2), LOCALs (je 2)
'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 250

Config Sda = Portc.4
Config Scl = Portc.5
```





```

Const 6d_bs_schreibadresse = &H32
Const 6d_bs leseadresse = &H33
Dim Sensordaten(6) As Byte
Dim X As Integer , Y As Integer , Z As Integer
Dim ZAHL_STRING AS STRING * 6

CLS
Lcd "ELV"
LOWERLINE
Lcd "6D-BS"
WAIT 2

'Grundkonfiguration
I2cstart
I2cwbyte 6d_bs_schreibadresse
I2cwbyte &H20
I2cwbyte &H77
I2cstop

Do
I2cstart
I2cwbyte 6d_bs_schreibadresse
I2cwbyte &HA8
I2cstart
I2cwbyte 6d_bs leseadresse
I2crbyte Sensordaten(1) , Ack
I2crbyte Sensordaten(2) , Ack
I2crbyte Sensordaten(3) , Ack
I2crbyte Sensordaten(4) , Ack
I2crbyte Sensordaten(5) , Ack
I2crbyte Sensordaten(6) , Nack
I2cstop

X = Makeint(sensordaten(1) , Sensordaten(2))
Y = Makeint(sensordaten(3) , Sensordaten(4))
Z = Makeint(sensordaten(5) , Sensordaten(6))
'In x, y und z stehen nun die Rohdaten als Integerzahlen zur Verfügung
'Wertebereich nur durch Erdbeschleunigung je Achse (x, y, z) -16000 ... +16000

Locate 1 , 1
Lcd "x:" : Zahl_string = Str(x) : Lcd Format(zahl_string , "+00000")
Locate 1 , 9
Lcd "y:" : Zahl_string = Str(y) : Lcd Format(zahl_string , "+00000")

Lowerline
Lcd "z:" : Zahl_string = Str(z) : Lcd Format(zahl_string , "+00000")

Waitms 500
Loop
End

```

'Default ist &h32. Alternativ &h30, wenn Pin SDO\_A auf Gnd

'ab Register &h20 = Control-Register  
'&b0111\_0111 400 Hz und alle Achsen aktiv

'Register &h28 und folgende Register. Vgl. Bedienungsanleitung

'Sensordaten einlesen

'x

'y

'z



'Zwei gelesene Bytes zu einer Integerzahl zusammenfassen

'.. und anzeigen

'.. und anzeigen

'.. und anzeigen

### Erläuterungen:

Im Programm wird das Ansprechen des Beschleunigungssensors dargestellt. Der ebenfalls integrierte Rotationsensor wird analog angesprochen. Vor der DO-LOOP-Hauptschleife wird in das Konfigurationsregister &h20 die gewünschte Grundkonfiguration geschrieben (vgl. Tabelle 3 in der Artikelbeschreibung sowie Tabellen 20 und 21 im Datenblatt [2] des LSM330).

In der Hauptschleife werden die Rohdaten ab Register &h28 (vgl. Tabelle 3 in der Produktbeschreibung) zunächst in ein Byte-Array eingelesen, dann mit MAKEINT in Integerzahlen umgewandelt und schließlich auf dem LC-Display angezeigt. Dabei ist zu beachten, dass für das Hintereinander-Auslesen mehrerer Register das höchstwertige Bit der Registeradresse auf 1 gesetzt wird. Es soll auf die Register ab &h28 zugegriffen werden. Die Registernummer &h28 entspricht in Binärdarstellung &b0010\_0100. Das höchstwertige Bit soll 1 werden. Das ergibt &b1010\_0100, was wiederum in hexadezimaler Schreibweise &hA8 ist.

Die ausgelesenen Daten können nun im BASCOM-Programm verwendet werden.

### Ausblick

In drei Teilen der Artikelserie „Mikrocontroller-Einstieg mit BASCOM-AVR“ wurde die Verwendung des I<sup>2</sup>C-Busses anhand der Einbindung beliebiger ELV-Module gezeigt. Viele weitere I<sup>2</sup>C-Module werden von ELV bzw. anderen Anbietern angeboten, deren Ansteuerung über den I<sup>2</sup>C-Bus nach dem gleichen I<sup>2</sup>C-Prinzip erfolgt. Im nächsten Teil der Artikelserie wird ein Einblick in die Grundlagen des 1-Wire-Bus gegeben und die BASCOM-Anbindung eines sehr verbreiteten Temperatursensors gezeigt.





## Weitere Infos:

### [1] Datenblatt BMA020:

Unter [www.elv.de](http://www.elv.de) bei der Artikelbeschreibung hinterlegt.  
Geben Sie dazu bitte einfach die Best.-Nr. J6-09 15 21 im Suchfeld ein.

### [2] Datenblatt LSM330DLC:

[www.st.com/web/en/resource/technical/document/datasheet/DM00037200.pdf](http://www.st.com/web/en/resource/technical/document/datasheet/DM00037200.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](http://www.bascom-buch.de)
- [www.mcselec.com](http://www.mcselec.com)
- [www.atmel.com](http://www.atmel.com)
- Produktübersicht BASCOM: [www.elv.de/bascom.html](http://www.elv.de/bascom.html)

### Empfohlene Produkte/Bauteile:

	Best.-Nr.	Preis
BASCOM-(Demo-)Lizenz von MCS Electronics, <a href="http://www.mcselec.com">www.mcselec.com</a>	-	-
Atmel-AVRISP-mkII-Programmer	J6-10 03 55	€ 39,95
<b>oder</b> myAVR-Board MK2	J6-10 90 00	€ 49,-
Netzteil für myAVR-Board MK2	J6-10 90 01	€ 6,95
ATtiny13	J6-10 03 39	€ 1,95
ATmega8	J6-05 29 71	€ 3,20
ATmega88	J6-10 07 62	€ 3,95
100-nF-Kondensator	J6-10 03 17	€ 0,15
Batteriehalter für 3x Mignon	J6-08 15 30	€ 0,75
Batterieclip für 9-V-Block-Batterie	J6-08 01 28	€ 0,30
BASCOM-Buch	J6-10 90 02	€ 54,-
Experimentier-Board 1202B	J6-07 72 89	€ 12,95
Schaltdraht-Sortiment	J6-05 47 68	€ 5,95
LED-Set	J6-10 63 56	€ 3,95
<b>oder</b> Leuchtdioden	J6-10 66 60	€ 1,65
und Widerstände	J6-10 66 57	€ 1,85
Piezo-Signalgeber	J6-00 73 87	€ 0,95
Mikroschalter und -taster	J6-10 66 67	€ 2,80
LC-Display, 2x 16 Zeichen	J6-05 41 84	€ 6,95
<b>oder</b> myAVR-LCD-Add-on-		
Pin-Ausrichter	J6-00 84 63	€ 4,95
I <sup>2</sup> C-Flip-Anzeige I2C-FA	J6-10 48 63	€ 8,95
LED-I <sup>2</sup> C-Steuertreiber, 16 Kanäle	J6-09 83 77	€ 12,95
I <sup>2</sup> C-4-Digit-LED-Display I2C-4DLED	J6-10 56 97	€ 16,95
I <sup>2</sup> C-Realtime-Clock I2C-RTC	J6-10 34 13	€ 6,50
Realtime-Clock mit DCF77 RTC-DCF	J6-13 05 41	€ 11,95
3-Achsen-Beschleunigungssensor 3D-BS	Komplettbausatz Fertiggerät	J6-09 15 21 € 6,95
		J6-10 48 93 € 9,95
6-Achsen-Bewegungssensor 6D-BS	J6-13 05 98	€ 21,50
I <sup>2</sup> C-Bus-Displaymodul I2C-LCD	J6-09 92 53	€ 13,95
LED-Bussystem LED-B6	J6-08 53 20	€ 14,95
I <sup>2</sup> C-Kabel	J6-08 56 89	€ 2,95
2-pol. Anschlussleitung passend für Miniatur-Stiftbuchse	J6-07 60 55	€ 1,25
Verbindungskabel 2 Module	J6-08 56 90	€ 2,95
Adapterplatine AP-Si4735	J6-10 34 39	€ 18,95
Intelligentes Schrittmotor-Treibermodul iSMT	J6-09 27 20	€ 24,95
USB-I <sup>2</sup> C-Interface USB-I2C	Komplettbausatz Fertiggerät	J6-09 22 55 € 34,95
		J6-08 41 23 € 24,95