Mikrocontroller-Einstieg

Teil 11: I²C – Grundlagen und Schreiben

mit BASCOM-AVR

Nachdem in den vorangegangenen Teilen der Artikelserie beschrieben wurde, wie Daten vom Mikrocontroller über die serielle UART-Schnittstelle gesendet und empfangen werden, wird hier nun das spannende Thema I²C behandelt. I²C steht für Inter Integrated Circuit und wurde in den frühen 80er-Jahren von Philips Semiconductors (heute NXP Semiconductors) als bidirektionaler Datenbus für die Kommunikation von Mikrocontrollern innerhalb eines Systems (zum Beispiel innerhalb eines Fernsehers) entwickelt. Von Atmel und einigen anderen Herstellern wird der Bus Two-Wire-Interface (TWI) genannt, womit technisch das Gleiche gemeint ist wie I²C. Mit I²C werden Daten über zwei I/O-Leitungen vom Mikrocontroller zu anderen Geräten gesendet oder von anderen Geräten empfangen. Typische Beispiele für I²C-Bausteine sind Temperatursensoren, Beschleunigungssensoren, LED-Treiber, LCD-Treiber, Portexpander, EEPROM-Bausteine, Echtzeituhren (Real Time Clocks = RTCs) und vieles mehr. Ein großer Vorteil ist, dass nur zwei I/O-Leitungen für den Anschluss vieler I²C-Bausteine benötigt werden. Aus diesem Grund und wegen der guten Benutzbarkeit und der guten Unterstützung durch Programmierumgebungen wie auch durch BASCOM bieten sehr viele ELV-Produkte eine Ansteuerungsmöglichkeit mit I²C. Im Folgenden werden die Grundlagen mit Blick auf die praktische Anwendung dargestellt und etliche Beispiele mit ELV-Produkten vorgestellt.

I²C-Grundlagen

Der I²C-Bus ist ein synchroner, serieller Zweidraht-Bus. „Seriell" heißt, dass – wie auch bereits bei UART – die Informationen auf einer Leitung hintereinander übertragen werden. Diese Leitung heißt Serial Data (SDA). Im Unterschied zu UART, wo Sender und Empfänger auf eine gemeinsame Übertragungsgeschwindigkeit eingestellt sein müssen, damit eine Kommunikation stattfinden kann, gibt es bei I²C eine zweite Leitung mit dem Namen Serial Clock Line (SCL) auf der ein gemeinsamer Takt übertragen wird. Deshalb spricht man von einer „synchronen" Datenübertragung. Die Kommunikation wird über einen Master angestoßen, der auch den Takt auf der SCL-Leitung vorgibt. Außer dem Master, welcher bei uns ein AVR-Mikrocontroller mit (BASCOM-)Programm ist, sind an den I²C-Bus bis zu 112 Slaves angeschlossen, die auf Signale vom Master warten und darauf reagieren können.

In Bild 1 sieht man den Master und die Slaves, die über die zwei I/O-Leitungen SDA und SCL mit dem Master verbunden sind. Master und Slaves müssen außerdem durch eine gemeinsame Gnd-Leitung verbunden sein. (Eigentlich werden also DREI Leitungen für den Bus benötigt, aber bei der Zählung werden nur die zwei I/O-Leitungen gezählt.) Der Master und die Slaves benötigen selbstverständlich außerdem jeweils eine positive Versorgungsspannung. Ganz wichtig ist bei I²C, dass beide Bus-Leitungen (SDA und SCL) jeweils einen Pull-up-Widerstand (Rpu) zur positiven Versorgungsspannung haben.

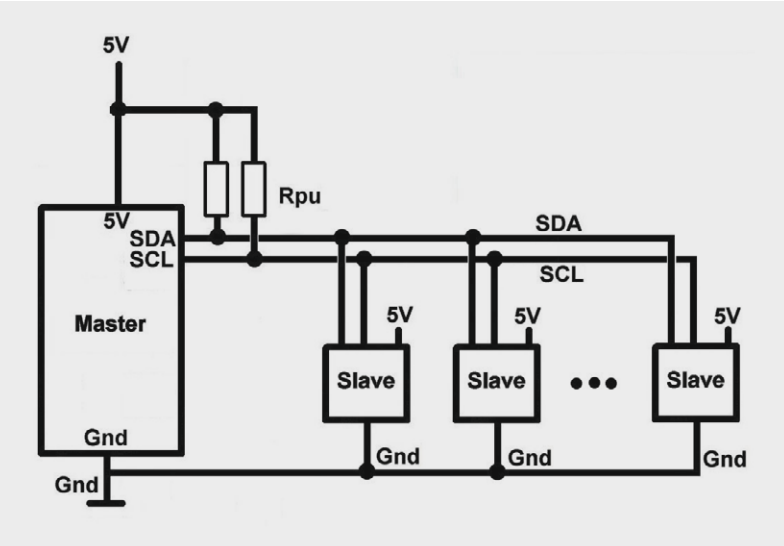


Bild 1: I²C-Bus

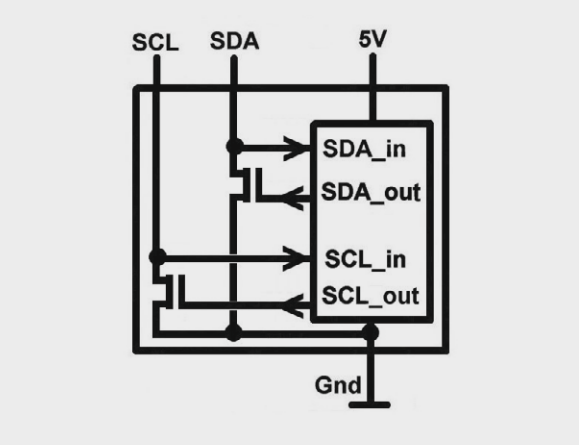


Bild 2: I²C-Slave mit Open Drain

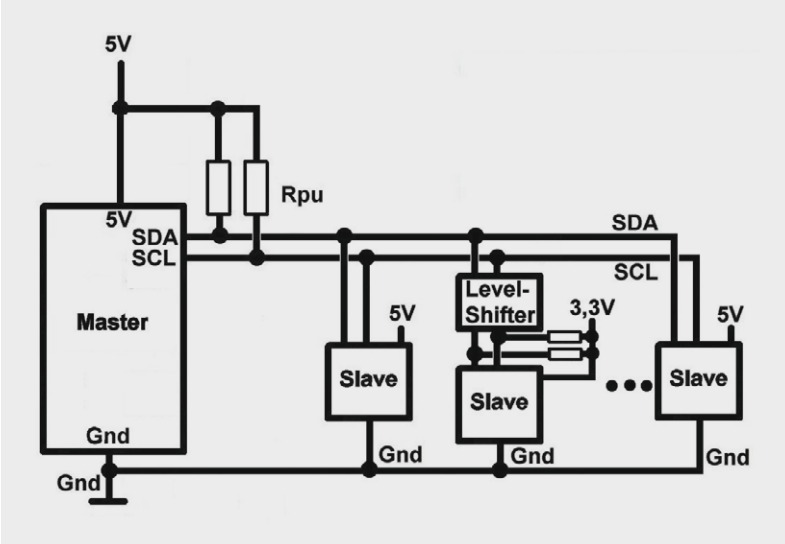


Bild 3: I²C-Bus mit Level-Shifter

Der Grund dafür ist, dass alle I²C-Komponenten intern mit einer Open-Drain- oder Open-Collector-Schaltung arbeiten (Bild 2). Durch die Pull-up-Widerstände sind die beiden Bus-Leitungen im Ruhezustand logisch HIGH und werden durch den Master bzw. einen Slave nach logisch LOW gezogen. Ohne Pull-up-Widerstände funktioniert ein I²C-Bus nicht!! Die Größe der Pull-up-Widerstände liegt zwischen 1 kΩ und 10 kΩ (typisch 4,7 kΩ). In Bild 2 ist zu sehen, dass jeder Baustein auf die Leitungen „lauschen" kann (SDA_in und SCL_in) und dass er auch die jeweilige Leitung auf LOW ziehen kann (SDA_out und SCL_out). Ein Baustein kann NICHT ein logisches HIGH auf die Busleitung schalten. Dafür gibt es die Pull-up-Widerstände, die die Leitung auf HIGH ziehen, solange nicht der Master oder ein Slave nach LOW schaltet. Jede I²C-Leitung erhält GENAU EINEN Pull-up-Widerstand.

Falls Master- und Slave-Bausteine unterschiedlich hohe Versorgungsspannungen haben, muss für eine Spannungsumsetzung auf den Busleitungen gesorgt werden. In Bild 3 sieht man einen 5-Volt-Master, der mehrere 5-Volt-Slaves und einen 3,3-Volt-Slave angebunden hat. Letzterer wird über einen Levelshifter (Spannungsumsetzer) an die zwei Busleitungen angeschlossen. Der 3,3-Volt-Slave erhält seine eigenen Pull-up-Widerstände zu seiner Versorgungsspannung (3,3 V). Ein Levelshifter besteht im einfachsten Fall aus einem MOSFET für jede der beiden I²C-Leitungen. So ein einfacher Levelshifter wird unter [1] beschrieben. Ein Elektronik-Wissen-Artikel aus dem ELVjournal erklärt die Grundlagen auf deutsch [2].

Die meisten ELV I²C-Slaves haben Pull-up-Widerstände und oft sogar Levelshifter bereits eingebaut, damit die Slaves den jeweiligen Umgebungen bequem angepasst werden können, ohne externe Pull-up-Widerstände oder Levelshifter aufbauen zu müssen. Das stellt eine wesentliche Unterstützung des Anwenders dar. Die Anleitung des jeweiligen ELV-Produkts ist sehr genau zu lesen, da für die möglichen Kombinationen (Pull-up-Widerstände aktiviert oder nicht, Levelshifter aktiviert oder nicht) Lötbrücken oder Jumper in der richtigen Kombination genutzt werden müssen.

Jeder am I²C-Bus angeschlossene Slave hat eine eindeutige I²C-Adresse, welche eine Zahl von 7 Bit Länge ist. Mit 7 Bit lassen sich 2^7 = 128 verschiedene Adressen darstellen. Von den möglichen 128 Adressen sind 16 Adressen für besondere Zwecke reserviert und daher sind 112 Slaveadressen an einem I²C-Bus möglich. (Sollte das nicht ausreichen, gibt es auch noch eine 10-Bit-Adressierung als Erweiterung der 7-Bit-Adressierung sowie spezielle Buserweiterungsbausteine!) Zu den 7 Bits für die Adressierung eines Slaves kommt noch ein Bit dazu, welches angibt, ob vom Master zum Slave geschrieben oder vom Slave zum Master gelesen werden soll. Zum Schreiben ist das Schreib-/Lese-Bit 0 und zum Lesen ist es 1. In Datenblättern steht R/W für Read/Write – also für Lesen/Schreiben (Bild 4). Insgesamt wird ein Slave also durch 8 Bit = 1 Byte angesprochen, wie in Bild 4 dargestellt.

Oftmals hat man mehrere gleichartige Slaves am I²C-Bus angeschlossen – zum Beispiel mehrere Temperatursensoren. Die (gleichartigen) Slaves haben dann einen gemeinsamen festen Adressteil (in Bild 4: 0011) und einen variablen Adressteil (in Bild 4: 111). Der variable Anteil der Adresse lässt sich am Slave meist über Lötbrücken oder Jumper einstellen. Bei manchen Bausteinen wird der variable Adressteil durch bestimmte Spannungslevel an EINEM Pin eingestellt. Bitte hierzu die Produktbeschreibung bzw. das Datenblatt des Slaves genau lesen! Im häufig vorkommenden Fall eines 3 Bit langen variablen Adressteils lassen sich 2^3 = 8 Slaves des gleichen Slave-Typs an den Bus anschließen.

Schreiben eines Bytes auf den I²C-Bus

Das schreibende Ansprechen eines I²C-Slaves erfolgt immer nach demselben Schema, welches in Bild 5 dargestellt ist.

BASCOM bietet für die Benutzung des I²C-Busses einige (wenige) BASCOM-Befehle. Am Anfang des BASCOM-Programms werden einmalig mit CONFIG SDA und CONFIG SCL die Pins für den I²C-Bus definiert. Das gesamte Schema zum Schreiben eines Datenbytes an den Slave mit der Adresse slaveadresse sieht in BASCOM so aus:

```
CONFIG SDA = ..          'Konfiguration des SDA-Pins
CONFIG SCL = ..          'Konfiguration des SCL-Pins
I2CSTART                 'Starten einer I²C-Übertragung
I2CWBYTE slaveadresse    'Slave adressieren
I2CWBYTE datenbyte       'Datenbyte übertragen
I2CSTOP                  'Bus wieder freigeben
```

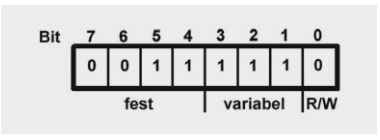


Bild 4: I²C-Adressierung

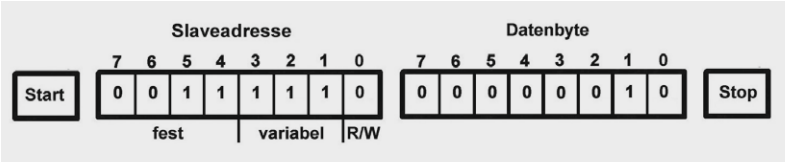


Bild 5: Ein Datenbyte schreiben

I²C-Scanner

Zum Einstieg in das Thema I²C und für Testzwecke ist ein sogenannter I²C-Scanner ein sehr nützliches Programm, mit dem geprüft wird, unter welchen Adressen sich ansprechbare Slaves am I²C-Bus befinden. Mit dem Programm werden alle geraden Adressen – also die Schreibadressen von 0 bis 254 – testweise angesprochen. Wenn sich unter der jeweiligen Adresse ein Slave „angesprochen fühlt" und er antwortet, dann ist die Systemvariable ERR gleich 0 und es wird am LC-Display die Adresse des Slaves in den Zahlenformaten Dezimal, Hexadezimal und Binär für 2 Sekunden angezeigt. Die ungeraden Leseadressen werden nicht getestet, weil sie zu der jeweiligen geradzahligen Schreibadresse gehören und man daher die geradzahlige Slaveadresse als „Basisadresse des Slaves" (und zugleich Schreibadresse) ansehen kann. Die Leseadresse entspricht der Schreibadresse mit dem Unterschied, dass das niedrigstwertige Bit (Bit 0) für die Schreibadresse auf 1 gesetzt ist.

Dieser Scanner sollte immer als Erstes eingesetzt werden, wenn man einen Slave zum ersten Mal am Bus einsetzt. Man sieht damit sehr schnell, ob der I²C-Bus funktioniert, ob die Pull-up-Widerstände richtig angeschlossen sind und welche Slaves man tatsächlich am I²C-Bus hat. Bei Slaves mit einstellbarer Slaveadresse hat man schnell die Gewissheit, dass die Einstellungen wunschgemäß vorgenommen und wirksam wurden. Die dezimale, hexadezimale und binäre Darstellung ist deswegen sinnvoll, weil man prinzipiell jede dieser Darstellungsformen benutzen kann. Manchmal wird in Produktbeschreibungen oder Datenblättern die Dezimaldarstellung (z. B. 62) benutzt, häufig werden Slaveadressen hexadezimal angegeben (z. B. h3E). Zur Kontrolle der Konfigurationspins der Slaveadressierung ist die Binärdarstellung (b00111110) sehr übersichtlich und hilfreich, weil der feste und der variable Adressenteil sehr schön zu erkennen sind.

Schaltungstechnisch werden die Slaves über die zwei I²C-Leitungen mit dem Master verbunden. Beide I²C-Leitungen erhalten jeweils EINEN Pull-up-Widerstand. Eine gemeinsame Gnd-Leitung und die jeweiligen Spannungsversorgungen komplettieren die Schaltung (Bild 1 bzw. Bild 3).

```
' BASCOM-Programm
'
' I2C-Scanner mit ATmega88
' Scannen der Slaveadressen
'
'  In: beliebige I2C-Slaves  an C.4=SDA und C.5=SCL
' Out: LCD an D2 bis D.7
'      B.6 Piezo-Buzzer ohne Elektronik

$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 Portb = Output                      'Buzzer
Buzzer Alias Portb.6

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
'I2cinit

Dim Slaveadresse As Byte

Do
Cls
Lcd '"I2C Slaves"
Lowerline
Lcd "suchen..."
Wait 2

For Slaveadresse = 0 To 254 Step 2         'Für alle geraden Adressen
  I2cstart                                 'Startbedingung senden
  I2cwbyte Slaveadresse                    'Adresse senden
  If Err = 0 Then                          'Wenn Antwort vom Slave
    Cls
    Sound Buzzer , 55 , 189                'Kurzen Ton ausgeben
    Lcd "Slave dec: " ; Slaveadresse       'Slaveadresse anzeigen
    Lowerline
    Lcd "h" ; Hex(slaveadresse) ; "  b" ; Bin(slaveadresse)
    Wait 2
  End If
  I2cstop                                  'Bus freigeben
Next
```

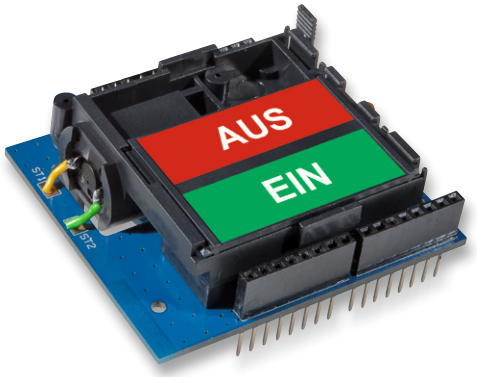
```
Cls
Lcd "Ende Scan"
Wait 2

Loop
End
```

Erläuterungen:
An diesem ersten I²C-Programm sieht man einige grundsätzliche Dinge eines I²C-BASCOM-Programms. Mit CONFIG SDA und CONFIG SCL werden die beiden Pins definiert, die für den I²C-Bus verwendet werden sollen. (I2CINIT ist nur in speziellen/kritischen Situationen nötig. Prinzipiell wird der Bus am Anfang des BASCOM-Programms bereits initialisiert.) Eine I²C-Kommunikation wird immer mit I2CSTART eingeläutet. Mit I2CWBYTE wird ein Byte auf den I²C-Bus geschrieben. In diesem Programm wird (nur) die zu testende Slaveadresse geschrieben. Und wenn sich daraufhin ein Slave am Bus meldet, weil die geschriebene Slaveadresse mit seiner eigenen Slaveadresse übereinstimmt, dann ist der Inhalt der Variablen ERR 0 und die Slaveadresse wird auf dem Display angezeigt. Eine I²C-Datenübertragung wird IMMER mit I2CSTOP beendet, wodurch der Bus wieder freigegeben wird.

Flip-Anzeige

Unter der Bezeichnung I²C-Flip-Anzeige I2C-FA (Best.-Nr. J4-10 48 63) bietet ELV ein kleines Modul zur Signalisierung von Zuständen an. Die Besonderheit ist, dass sich das Modul über den I²C-Bus ansprechen lässt und deswegen – auch bei mehreren dieser Flip-Anzeigen am Bus – außer einer gemeinsamen Gnd-Leitung nur die beiden I²C-Leitungen SDA und SCL benötigt werden (Bild 6). Mikrocontroller und Slave benötigen natürlich noch jeweils eine positive Spannungsversorgung. Über diese wenigen Leitungen lassen sich durch Adressierung bis zu acht derartige Module ansteuern. Die individuelle Slaveadresse wird am Modul durch drei Lötbrücken eingestellt. Das Modul ist für 3-V- oder 5-V-Betrieb geeignet. Die I²C-Busspannung (durch die Pull-up-Widerstände) muss 3 V sein. Falls der steuernde Mikrocontroller mit 5 V betrieben werden soll, muss ein Levelshifter, wie in Bild 3 skizziert, zwischengeschaltet werden. Pull-up-Widerstände können, wie in Bild 6 gezeigt, extern angeschlossen oder per Jumper J15 und J16 auf dem Modul aktiviert werden.

```
' BASCOM-Programm
'
' I2C-Flip-Anzeige mit ATmega88
' Testprogramm für ELV Flip-Anzeige mit I2C-Ansteuerung
'
'  In: Taster an B.0 und B.1
' Out: Flip-Anzeige  an C.4=SDA und C.5=SCL
' Out: LCD an D2 bis D.7
$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 Portb.0 = Input
Taster1 Alias Pinb.0
Portb.0 = 1                                 'Pull-up-Widerstand
Config Portb.1 = Input
Taster2 Alias Pinb.1
Portb.1 = 1                                 'Pull-up-Widerstand

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

Const Flipslave = &H3E                      'Slaveadresse Flip-Anzeige

Do
Debounce Taster1 , 0 , Flip_up , Sub
Debounce Taster2 , 0 , Flip_down , Sub
'Debounce Taster2 , 0 , Flip_toggle , Sub
Loop
End

Flip_up:
I2cstart
I2cwbyte Flipslave
I2cwbyte &H02                               'An/up
I2cstop
Return

Flip_down:
I2cstart
I2cwbyte Flipslave
I2cwbyte &H04                               'Aus/down
I2cstop
Return

Flip_toggle:
I2cstart
I2cwbyte Flipslave
I2cwbyte &H08                               'Toggle
I2cstop
Return
```



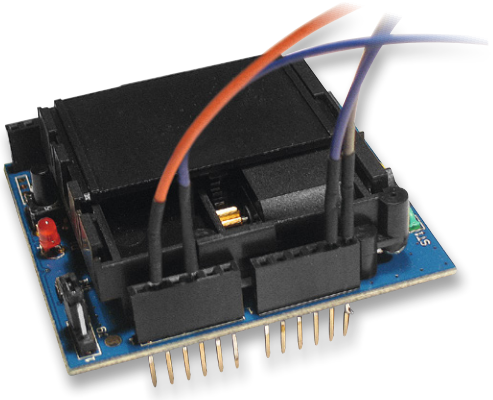
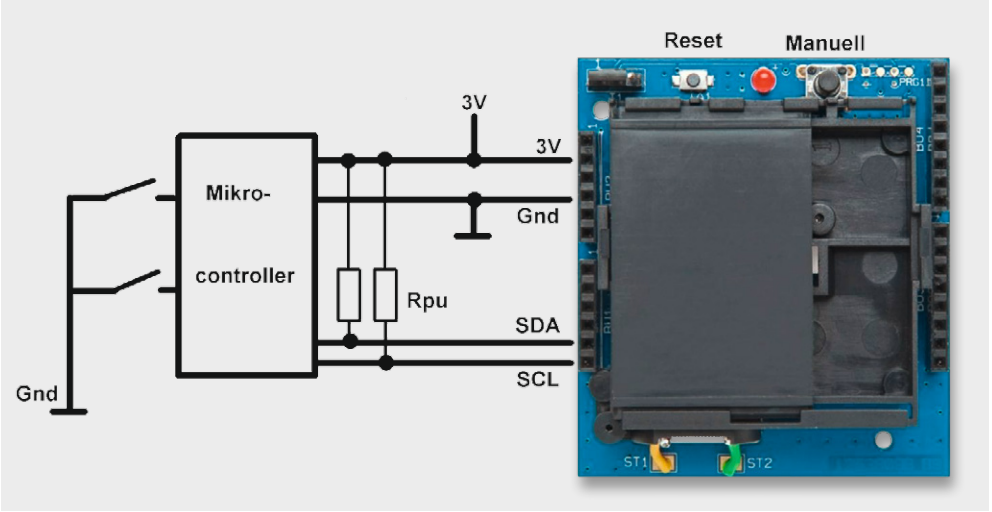


Bild 6: Anschluss Flip-Anzeige



Erläuterungen:
Im Programm sind wieder die grundlegenden Befehle CONFIG SDA und CONFIG SCL zu sehen, durch die die I²C-Pins definiert werden. Beim Drücken des jeweiligen Tasters (ausgewertet mit dem BASCOM-Entprellbefehl DEBOUNCE) wird ein dem Taster entsprechendes Unterprogramm aufgerufen, in dem die Flip-Anzeige hoch- bzw. runtergeschaltet wird (Flip_up bzw. Flip_down) oder umgeschaltet wird (Flip_toggle). In jeder dieser Unterroutinen ist wieder der I²C-typische Ablauf zu erkennen: Mit I2CSTART wird die Kommunikation begonnen. Mit I2CWBYTE wird über die Slaveadresse eine Flip-Anzeige angesprochen. Mit einem weiteren I2CWBYTE wird das Datenbyte an den Slave gesendet, welches den Slave zu der jeweiligen Aktion veranlasst (h02, h04 bzw. h08). Mit I2CSTOP wird der I²C-Bus wieder freigegeben.

Schreiben mehrerer Bytes auf den I²C-Bus

Je nach Aufbau des I²C-Slaves können auch mehrere Datenbytes hintereinander vom Master zum Slave geschrieben werden. Wie in Bild 7 zu erkennen bleibt das Schreibschema (START-Slaveadresse-Datenbyte(s)-STOP) gleich. Statt EINES Datenbytes werden hier nun mehrere (in Bild 7: 2) Datenbytes an den Slave geschrieben.

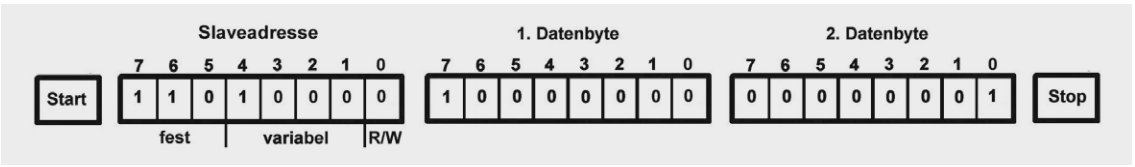


Bild 7: Mehrere Datenbytes schreiben

16-LED-Treiber

Das ELV-Produkt LED-I2C-Steuertreiber (Best.-Nr. J4-09 83 77) ist ein LED-Treiber für 16 Kanäle. Durch Jumper lassen sich 16 Adressen am Modul einstellen. Da es zwei reservierte Adressen gibt (Resetadresse und Alle_Adresse), lassen sich 14 derartige Treibermodule adressieren und man kann dadurch 14 * 16 = 224 LEDs individuell oder gruppenweise einschalten, ausschalten oder dimmen! Der Anschluss der Module geschieht wiederum über nur zwei Datenleitungen (Bild 8). 224 LEDs mit drei Leitungen (SDA, SCL und Gnd) anzusteuern ist ausreichend, um zum Beispiel eine komplette Modellbahnbeleuchtung, einen Sternenhimmel oder eine Uhrenanzeige mit einem kleinen Mikrocontroller anzusteuern. Das Modul ist für 5-Volt-Betrieb aufgebaut. Pull-up-Widerstände für den I²C-Bus lassen sich per Lötbrücken auf dem Modul aktivieren oder extern verwenden.

```
'BASCOM-Programm
'
' TLC59116 bzw. I2C-Bus-16LED-Treiber mit ATmega88
' Ansteuerung von 16 LED-Treibern
'
'   In: -
' Out: TLC59116 bzw. I2C-Bus-16LED-Treiber an C.4=SDA und C.5=SCL
'      LCD an D2 bis D.7

$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
Cls
Cursor Off
Waitms 250
```

```
Const Tlc_reset = &HD6                                  'Adresse für Reset: &hD6=&b1101_0110
                                                        Vgl. S. 22 im TLC59116 Datenblatt
Const Tlc_alle = &HD0                                   'Alle TLCs  &hD0=&b1101_0000  Vgl. Seite 10 im DB
Const Tlc = &HDE                                        'Einzelner TLC  &hDE=&b1101_1110  Vgl. S. 10 im DB
                                                        'J4 bis J7 gesteckt.  Vgl. Tabelle 2 in Produktbeschreibung.

Config Sda = Portc.4                                    'Pins für I2C konfigurieren
Config Scl = Portc.5                                    '    und
I2cinit                                                 'Bus initialisieren (Nach Reset eig. unnötig)

Dim I As Byte                                           'Laufvariable
Dim Led_status As Word                                  '16-Bit-Variable mit ein/aus-Zuständen der LEDs 0 bis 15
Declare Sub Leds_schalten(byval Pwm_kanaele As Word)    'Unterprogramm zum Schalten der LEDs


'0.) Reset  Setzt alles auf Standardwerte.  Vgl. Seite 22 im Datenblatt
I2cstart
I2cwbyte Tlc_reset                      'I2C-Slaveadresse für Resetten
I2cwbyte &HA5                           '1. Byte der Bytefolge für Resetten
I2cwbyte &H5A                           '2. Byte
I2cstop
Cls
Lcd "Ende Reset " ; Err                 'Err=0 heißt alles o.k.
Wait 2

'Prinzip immer:
'i2cstart
'i2cwbyte i2c_Slaveadresse  =   Adressierung des Bausteines
'i2cwbyte Kontrollregister  =   AAAD_DDDD   Bestimmt Auto-Inkrement-Modus (AAA) (S. 11 im DB) und Start-Registeradresse (D_DDDD) (S. 14)
'i2cwbyte Register          =   Inhalt, der in das jeweilige Register geschrieben werden soll
'ggf weitere Register       =                   -  "  -
'i2cstop

'Initialisierung
'1.) Bit 4 im Mode1-Register muss 0=Normal Mode sein.  Default ist aber 1. Vgl. Seite 15
I2cstart
I2cwbyte Tlc_alle
I2cwbyte &B1000_0000                    'Kontrollregister AI2:AI0 D4:D0 Hier: Autoinkrement ab &h00
I2cwbyte &B0000_0001                    'Mode1-Register &h00  Vgl. Seite 15
I2cstop
Cls
Lcd "Ende Init1 " ; Err                 'Err=0 heißt: alles o.k.
Wait 2


'2.) LED output state muss jeweils 255 = 1111_1111 sein.  Default ist aber 0.   Vgl. Seite 17 im Datenblatt.
I2cstart
I2cwbyte Tlc_alle
I2cwbyte &B1001_0100                    'Kontrollregister AI2:AI0 D4:D0 Hier: Autoinkrement ab &h14
I2cwbyte &B1111_1111                    'Register &h14 LEDOUT0: Alles an
I2cwbyte &B1111_1111                    'Register &h15 LEDOUT1: Alles an
I2cwbyte &B1111_1111                    'Register &h16 LEDOUT2: Alles an
I2cwbyte &B1111_1111                    'Register &h17 LEDOUT3: Alles an
I2cstop
Cls
Lcd "Ende Init2 " ; Err                 'Err=0 heißt: alles o.k.
Wait 2
```

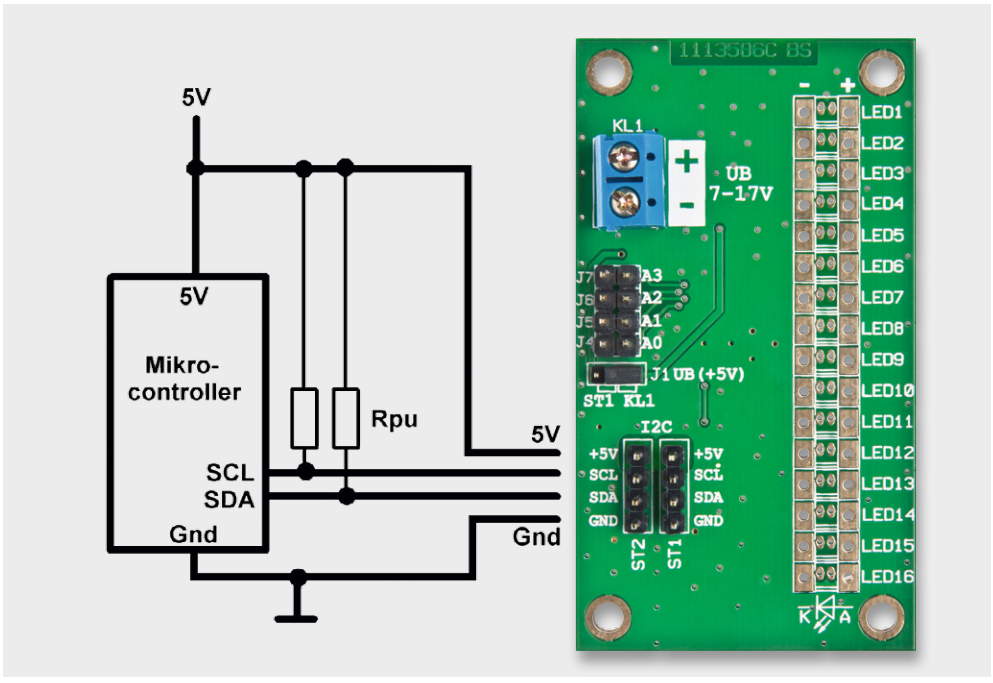
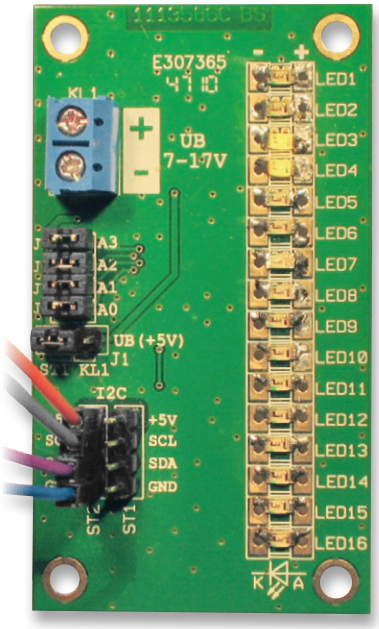


Bild 8: 16-LED-Treiber-Anschluss

```
'3.) LEDs beispielhaft schalten
I2cstart
I2cwbyte Tlc                            'oder Tlc_alle
I2cwbyte &B1010_0010                    'Kontrollregister AI2:AI0 D4:D0 Hier: Autoinkrement ab &h02
I2cwbyte 255                            'Register PWM0 LED0:  Voll an
I2cwbyte 50                             'Register PWM1 LED1:  Teil an
I2cwbyte 10                             'Register PWM2 LED2:  Sehr dunkel
I2cwbyte 0                              'Register PWM3 LED3:  Aus
I2cwbyte 255                            'Register PWM4 LED4:  Voll an
I2cstop
Cls
Lcd "Ende LEDs " ; Err                  'Err=0 heißt: alles o.k.
Wait 2

'Schleife mit Lauflicht
Do
'EINE LED hin und her laufen lassen
For I = 0 To 15
  Led_status = 0                        'Alle aus
  Led_status.i = 1                      'Eine LED an
  Call Leds_schalten(led_status)        'LEDs entsprechend schalten
  Waitms 100
Next
For I = 14 To 0 Step -1
  Led_status = 0                        'Alle aus
  Led_status.i = 1                      'Eine LED an
  Call Leds_schalten(led_status)        'LEDs entsprechend schalten
  Waitms 100
Next

'ZWEI LEDs hin und her laufen lassen
Led_status = &B0000_0000_0000__0011
For I = 1 To 14
  Call Leds_schalten(led_status)        'LEDs entsprechend schalten
  Waitms 100
  Shift Led_status , Left , 1
Next
For I = 1 To 14
  Call Leds_schalten(led_status)        'LEDs entsprechend schalten
  Waitms 100
  Shift Led_status , Right , 1
Next

Loop
End

Sub Leds_schalten(byval Pwm_kanaele As Word)
'LEDs schalten.  Vgl. Seite 16 im Datenblatt.
Local J As Byte
I2cstart
I2cwbyte Tlc                            'oder tlc_alle
I2cwbyte &B1010_0010                    'Kontrollregister AI2:AI0 D4:D0 Hier: AI ab &h02
For J = 0 To 15
  If Pwm_kanaele.j = 1 Then
    I2cwbyte 255                        'Register PWM
  Else
    I2cwbyte 0                          'Register PWM
  End If
Next
I2cstop
End Sub
```



Erläuterungen:

Dieses Anwendungsbeispiel zeigt deutlich, dass es unumgänglich ist, sich mit der Produktbeschreibung und mit dem Datenblatt des im Modul verwendeten ICs (TLC59116) zu beschäftigen. Das Schema bei I²C ist immer gleich, aber man muss für jede Situation nachlesen, welche Slaveadresse(n) angesprochen werden muss/müssen, welche Register betroffen sind und welche Inhalte als Bytes geschrieben bzw. gelesen werden. Am Anfang des Programms werden hier wieder die I²C-Leitungen mit CONFIG SDA und CONFIG SCL definiert. Unter der Überschrift 0.) Reset wird der I²C-Baustein in einen sinnvollen Grundzustand versetzt, indem die Reset-Sequenz wie im Datenblatt des TLC59116 auf Seite 22 beschrieben ausgeführt wird. Dafür gibt es eine reservierte Slaveadresse für den Reset und zwei definierte Bytes, die geschrieben werden müssen. Unter 1.) und 2.) im Quelltext werden Grundeinstellungen im I²C-Baustein vorgenommen. Gesendet wird an die reservierte Slaveadresse, mit der alle angeschlossenen TLC-Bausteine angesprochen werden. Im ersten nach der Slaveadresse gesendeten Byte wird angegeben, ob bei weiteren Schreibvorgängen automatisch die Registernummer hochgezählt (inkrementiert) werden soll und ab welchem Register geschrieben werden soll.

Unter 3.) ist wieder das bekannte I²C-Prinzip zu erkennen: Starten der I²C-Kommunikation mit I2CSTART. Dann Ansprechen eines Slaves durch Schreiben seiner Slaveadresse. Danach folgt die Angabe, wie das Auto-Inkrementieren aussehen soll und ab welchem Register geschrieben werden soll. Im Anschluss werden die Datenbytes geschrieben (Seite 16 im Datenblatt) und schließlich die I²C-Kommunikation mit I2CSTOP beendet.

Ausblick

In diesem Teil der BASCOM-Artikelserie wurden Daten vom Master (Mikrocontroller) zum Slave übertragen. Im nächsten ELVjournal wird gezeigt, wie der Master Daten vom Slave lesen kann. Ein Thermometer und eine Echtzeituhr (Real Time Clock = RTC) werden beispielhaft mit BASCOM und entsprechenden ELV-Modulen dargestellt.

Weitere Infos:

[1] www.nxp.com/documents/application_note/AN10441.pdf
[2] www.elv.de/output/controller.aspx?cid=758&detail=10&detail2=7

- 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
- www.atmel.com
- www.nxp.com/documents
- www.nxp.com/documents/user_manual/UM10204.pdf
- www.rn-wissen.de/index.php/I2C und www.timmermann.org/ralph/index.htm?
- www.ralph.timmermann.org/elektronik/i2c.htm
- Produktübersicht BASCOM: **www.elv.de/bascom.html**

Empfohlene Produkte/Bauteile:		**Best.-Nr.**	**Preis**
BASCOM-(Demo-)Lizenz von MCS Electronics www.mcselec.com		–	–
Atmel-AVRISP-mkII-Programmer		J4-10 03 55	€ 39,95
oder myAVR-Board MK2		J4-10 90 00	€ 49,–
Netzteil für myAVR-Board MK2		J4-10 90 01	€ 6,95
ATmega88		J4-10 07 62	€ 3,95
100-nF-Kondensator		J4-10 03 17	€ 0,08
Batteriehalter für 3x Mignon		J4-08 15 30	€ 0,75
Batterieclip für 9-V-Block-Batterie		J4-08 01 28	€ 0,30
BASCOM-Buch		J4-10 90 02	€ 54,–
Experimentier-Board 1202B		J4-07 72 89	€ 12,95
Schaltdraht-Sortiment		J4-05 47 68	€ 5,95
LED-Set		J4-10 63 56	€ 3,95
oder Leuchtdioden		J4-10 66 60	€ 1,65
und Widerstände		J4-10 66 57	€ 1,85
Piezo-Signalgeber		J4-00 73 87	€ 0,95
Mikroschalter und -taster		J4-10 66 67	€ 2,80
LC-Display, 2 x 16 Zeichen		J4-05 41 84	€ 6,95
oder myAVR-LCD-Add-on-			
Pin-Ausrichter		J4-00 84 63	€ 4,95
I²C-Flip-Anzeige I2C-FA		J4-10 48 63	€ 8,95
LED-I²C-Steuertreiber, 16 Kanäle		J4-09 83 77	€ 12,95
I²C-4-Digit-LED-Display I2C-4LED		J4-10 56 97	€ 16,95
I²C-Real-Time-Clock I2C-RTC		J4-10 34 13	€ 6,50
Real Time Clock mit DCF77 RTC-DCF		J4-13 05 41	€ 11,95
3-Achsen-Beschleunigungssensor 3D-BS	Komplettbausatz	J4-09 15 21	€ 6,95
	Fertiggerät	J4-10 48 93	€ 9,95
6-Achsen-Bewegungssensor 6D-BS		J4-13 05 98	€ 21,50
I²C-Bus Displaymodul I2C-LCD		J4-09 92 53	€ 13,95
LED-Bussystem LED-B6		J4-08 53 20	€ 14,95
I²C-Kabel		J4-08 56 89	€ 2,95
2-pol. Anschlussleitung passend für Miniatur-Stiftbuchse		J4-07 60 55	€ 0,99
Verbindungskabel 2 Module		J4-08 56 90	€ 2,95
Adapterplatine AP-Si4735		J4-10 34 39	€ 18,95
Intelligentes Schrittmotor Treibermodul iSMT		J4-09 27 20	€ 24,95
USB-I²C-Interface USB-I2C	Komplettbausatz	J4-09 22 55	€ 34,95
	Fertiggerät	J4-08 41 23	€ 24,95

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

Preisstellung Juni 2014 – aktuelle Preise im Web-Shop