AVR-Mikrocontroller in BASCOM programmieren

Der I2C-Bus (TWI-Interface)

Allgemeines

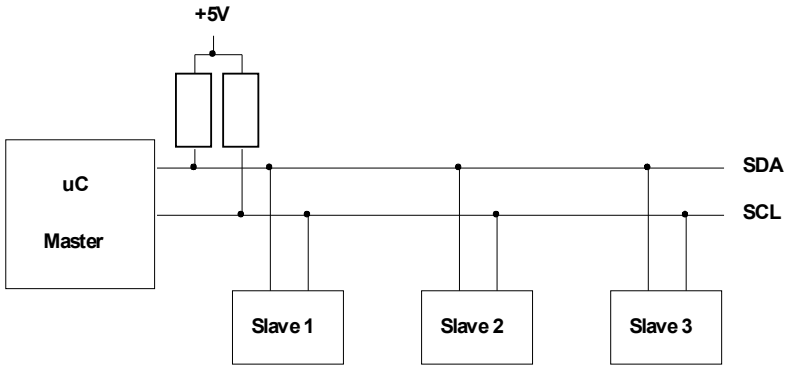
Spezifikationen des I2C-Busses:

http://www.nxp.com/acrobat_download2/literature/9398/39340011.pdf

Taktraten:

100kHz	Standard
400kHz	Fast
3.4MHz	High speed

Schaltung:



SDA: serielle Daten
SCL: Takt (wird vom Master erzeugt, der Slave kann aber die Low-Phase bei Bedarf verlängern)

Pullup-Widerstände: min. Strom 3mA
Bei 5V und 200pF Kapazität sind 1.8K angemessen.

Achtung: I2C-Geräte die einen Prozessor haben der mit 3V läuft brauchen Pullup-Widerstände nach +3V und nicht nach 5V!

Adressen der Slaves:

Das erste übertragene Byte ist immer die Adresse.

Bits 7...1	Bit0
Adresse	R/W

Das niederwertigste Bit enthält das Kontrollbit R/W: Lesen oder Schreiben des Slave:

1 = Lesen
0 = Schreiben

Die meisten I2C-ICs haben feste Adressen, die im Datenblatt angegeben sind. Oft sind die oberen 4 Bits fest, während die unteren 3 über Jumper festgelegt werden können.

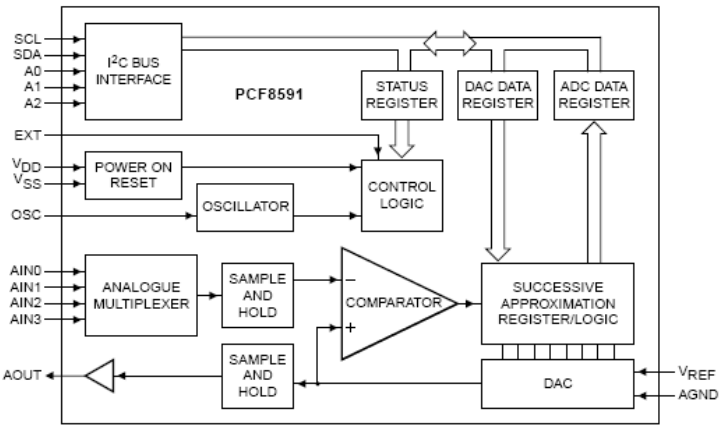
Datenaustausch

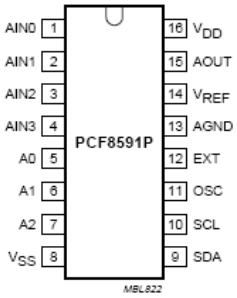
- Der Master prüft ob der Bus frei ist (beide Leitungen auf H-Pegel)
- Der Master erzeugt ein START-Signal:
  SDA wird auf L gezogen (SCL bleibt auf H)
  Nun gehört der Bus dem Master.
- Der Master sendet die Adresse des Slaves.
  Dieser zieht daraufhin SDA auf L, vom Master wird dies als ACK (Acknowledge) interpretiert.
- Nun werden 1 oder mehrere Datenbytes gesendet.
  Der Empfänger quittiert jedes mit ACK (d.h. er zieht SDA auf L).
  Wenn SDA auf H bleibt, interpretiert der Master dies als NAK (Not acknowledged)
- Am Ende der Übertragung erzeugt der Master ein STOP-Signal:
  SCL = H, SDA geht von L auf H.
  Nun ist der Bus wieder frei.

Beispiel 1: DA-Wandlung mit PCF8591

Dieser Chip eignet sich gut für erste Experimente, da man sofort ein Ergebnis sehen kann.

In den folgenden Beispielen wurden die binären Werte nicht in Spannungen umgerechnet, da hier der Schwerpunkt auf der I2C-Übertragung liegen soll.

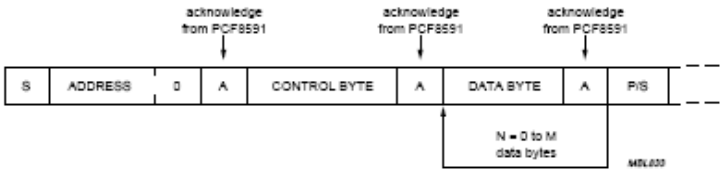




Die Adresse des Chips hat einen festen und einen einstellbaren Teil.

Der einstellbare Teil wird über die Eingangspins A2, A1, A0 festgelegt.

Ablauf der Datenübertragung:

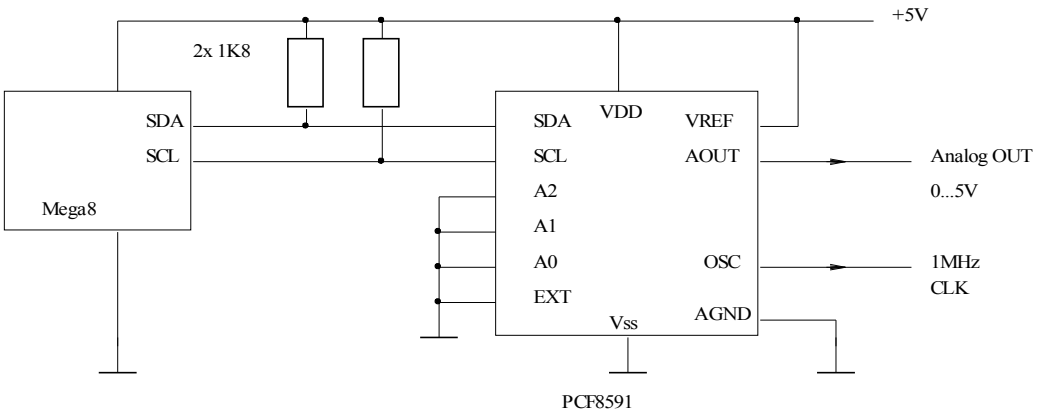


- Start-Kondition
- Adresse senden
- Kontrollbyte senden (siehe Datenblatt)
- Datenbyte senden (eventuell auch mehrere)
- Stop-Kondition

Die ganze Abfolge wird nur vom Master gesteuert.

Der Slave antwortet auf jedes Byte mit einem Acknowledge, d.h. er zieht die SDA-Leitung auf low.

Schaltung zur DA-Wandlung mit Referenzspannung +5V:



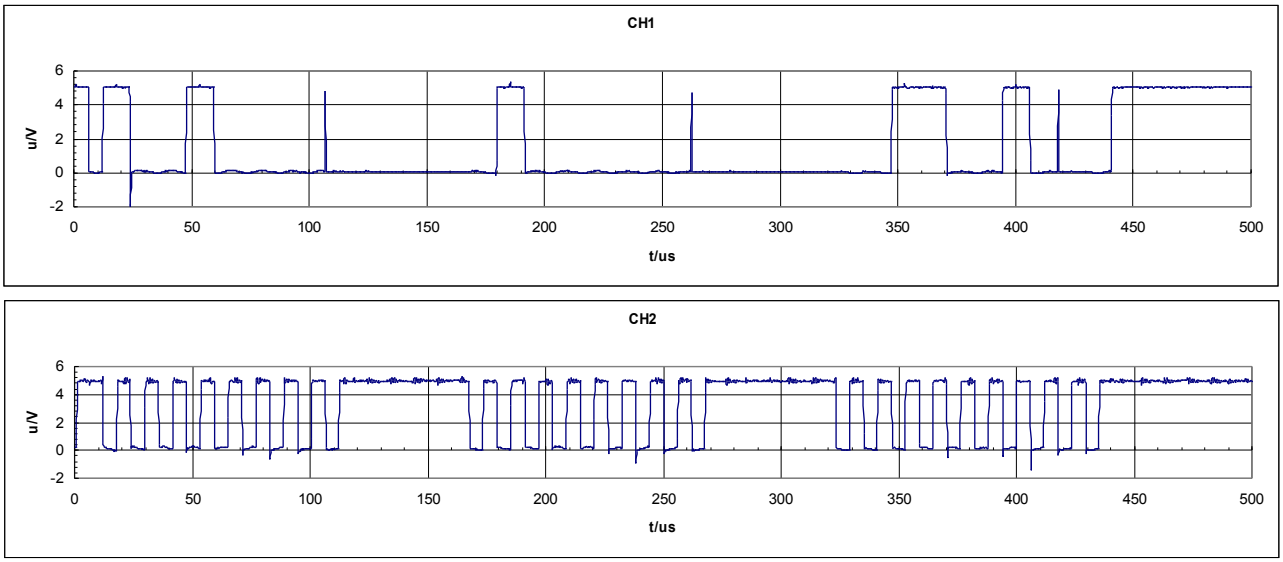
Achtung: der EXT-Pin muss auf Masse gelegt werden, damit der interne Oszillator funktioniert!
Dieser wird für den AD-Wandler als Takt benutzt, aber auch für den Refresh des Puffers am Ausgang (siehe Datenblatt). Am OSC-Ausgangspin ist dann ein 1MHz-Takt messbar.

Beispiel-Programm mit Werte-Eingabe über die serielle Schnittstelle:

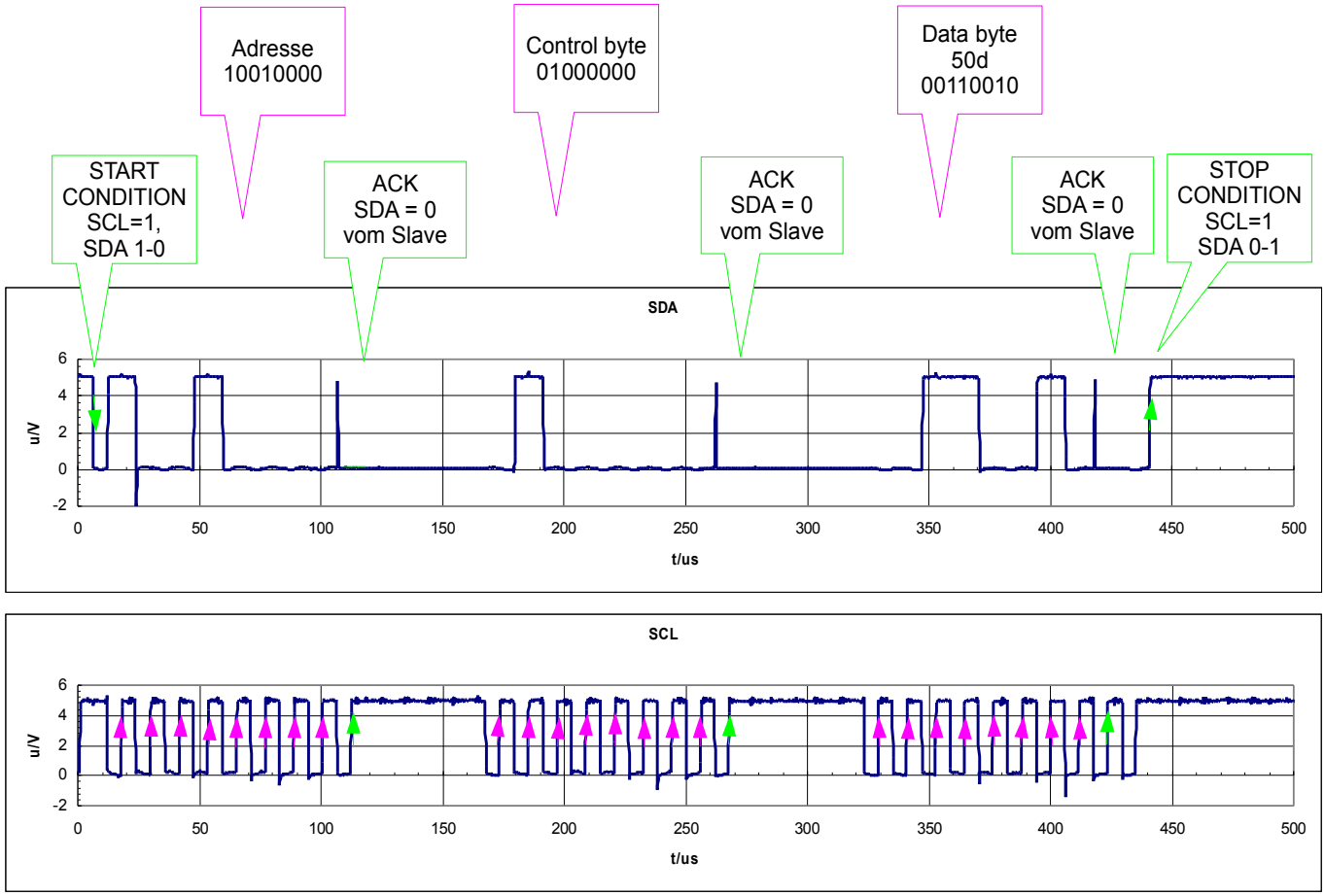
```
$crystal = 16000000
$regfile = "m8def.dat"
$hwstack = 100

$baud = 9600

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

Dim I As Byte

Do
    Input "I=" , I
    Print I
    Gosub I2cda

    Waitms 500

Loop
'-----------------------------------------------------
I2cda:
    I2cstart
    I2cwbyte &B10010000                    'address for write with A2A1A0=000
    I2cwbyte &B01000000                    'control byte with DA enable
    I2cwbyte I                             'byte for DA
    I2cstop
Return
```

Oszillogramme:

Hier wurde ein DA-Wert von 50 = 00110010dual übertragen.
Zur besseren Übersicht wurde eine kurze Pause vor jedem I2Cwbyte-Befehl eingefügt.



Analyse des Oszillogramms:



Die einzelnen Bits werden in der Reihenfolge MSB – LSB übertragen.

Die Bitwerte werden mit der ansteigenden Flanke von SCL übernommen.

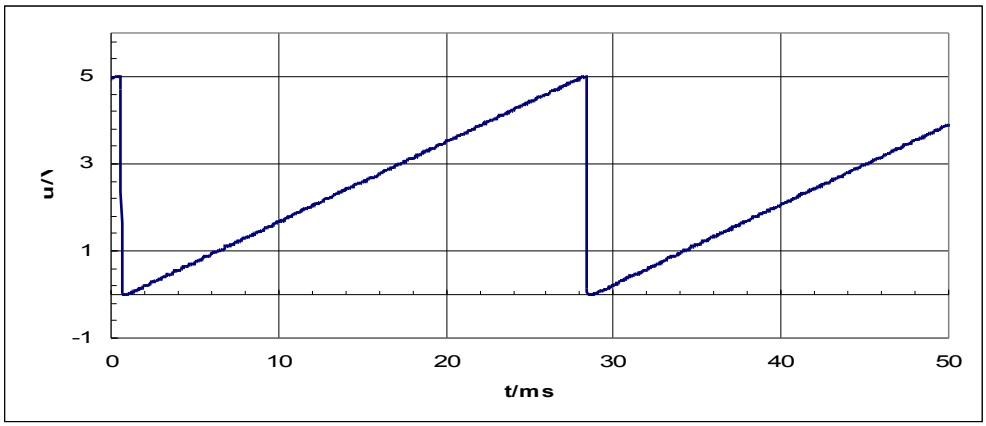
Nach jedem Byte wird ein neunter Taktimpuls vom Mikrocontroller generiert. Zu diesem Zeitpunkt muss der PCF8591 als Slave die Datenleitung auf Low gezogen haben.
Dies wird vom Mikrocontroller als Acknowledge interpretiert.

Übertragen mehrerer Bytes hintereinander

Sind viele Werte zu übertragen, so können diese direkt hintereinander übertragen werden.
Eine Stop-Kondition auf dem Bus beendet den Datentransfer.
Der Vorteil dieser Methode ist eine schnellere Übertragung.

Beispiel für eine sägezahnförmige Ausgangsspannung:

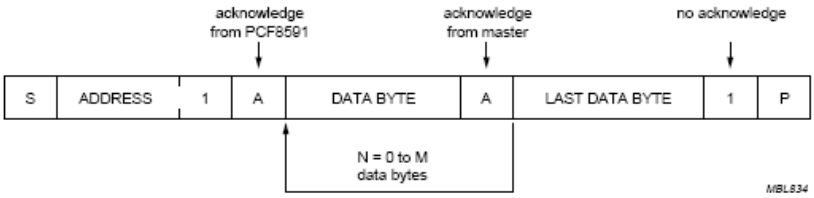
```
$crystal = 16000000
$regfile = "m8def.dat"
$hwstack = 100

$baud = 9600

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

Dim I As Byte

Do
 Gosub I2cda256bytes
Loop
'-----------------------------------------------------
I2cda256bytes:
 I2cstart
 I2cwbyte &B10010000      'address for write with A2A1A0=000
 I2cwbyte &B01000000                                    'control byte with DA enable

    For I = 0 To 255
      I2cwbyte I                                          'bytes for DA
    Next I

    I2cstop
Return
```



Für dieses Beispiel ergibt sich eine Periodendauer von ca. 28ms, bei Übertragung nach dem Schema des ersten Beispiels wären es ca. 95ms.

Beispiel 2: AD-Wandlung mit PCF8591

Ablauf der Datenübertragung:



- Start-Kondition (Master)
- Adresse senden (Master)
  Achtung: Bit 0 ist jetzt 1 : Lesen

  Vom PCF8591 kommt ein Acknowledge zurück (SDA auf low)
- Datenbyte(s) vom PCF8591 empfangen (eventuell auch mehrere)
  Nach jedem empfangenen Byte schickt der Master ein Acknowledge

  Will der Master kein Byte mehr empfangen, schickt er kein Acknowledge mehr, d.h. er lässt SDA auf H-Pegel.

- Stop-Kondition (Master)

Da der PCF8591 4 AD-Kanäle hat, muss eventuell vorher noch durch Schicken eines Control Bytes der gewünschte Kanal ausgewählt werden (im Beispiel Kanal 1).

(Es gibt noch interessante Möglichkeiten, z.B. die 4 Kanäle hintereinander zu übertragen, Differenzmessungen usw., siehe Datenblatt).

Man muss noch ein wenig aufpassen: nach dem Adressieren führt der PCF8591 eine Messung durch, er überträgt aber den Wert der vorigen Messung. Um den aktuellen Wert zu erhalten, muss man also 2 Bytes übertragen.

Beispiel für das Auslesen von Kanal 1:

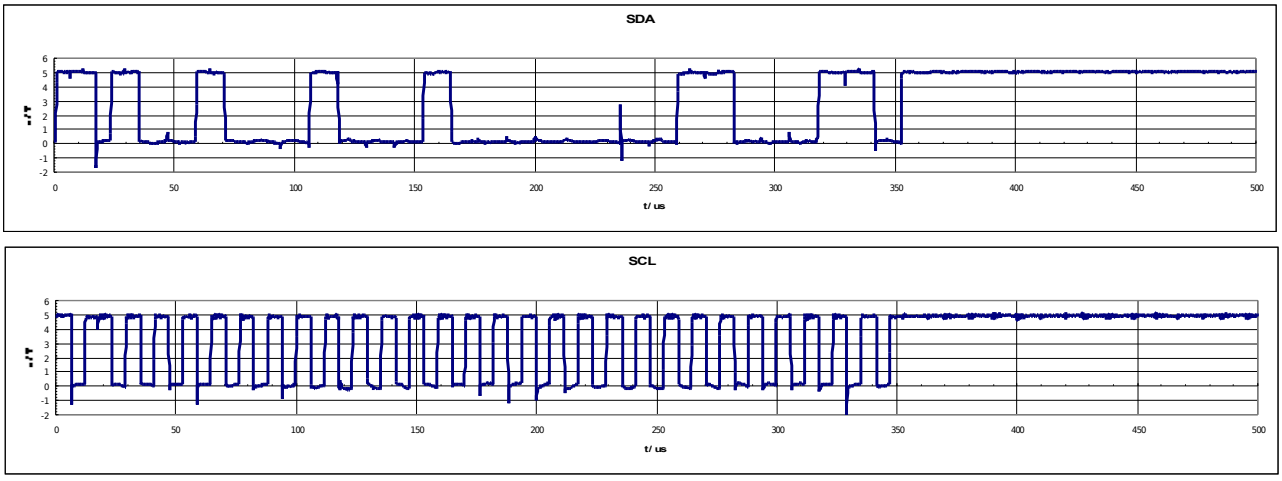
```
$crystal = 16000000
$regfile = "m8def.dat"
$hwstack = 100
$baud = 9600

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

Dim X As Byte                       'dummy byte for result of last conversion
Dim Y As Byte                       'result of actual conversion

Do

 Gosub I2cad
 Print X;                       'this gives the result from the last conversion
 Print "    "
 Print Y                        'this is the actual AD value
 Wait 1
```

```
Loop

'-----------------------------------------------------
I2cad:

'select channel (send control byte)
    I2cstart
    I2cwbyte &B10010000                'address for write with A2A1A0=000
    I2cwbyte &B01000001                'control byte with DA enable, channel 1
    I2cstop

'read from AD
    I2cstart
    I2cwbyte &B10010001                 'address for read with A2A1A0=000
    I2crbyte X , Ack                    'read byte from last conversion
    I2crbyte Y , Nack                   'read byte from actual conversion
    I2cstop
Return
```

Oszillogramme

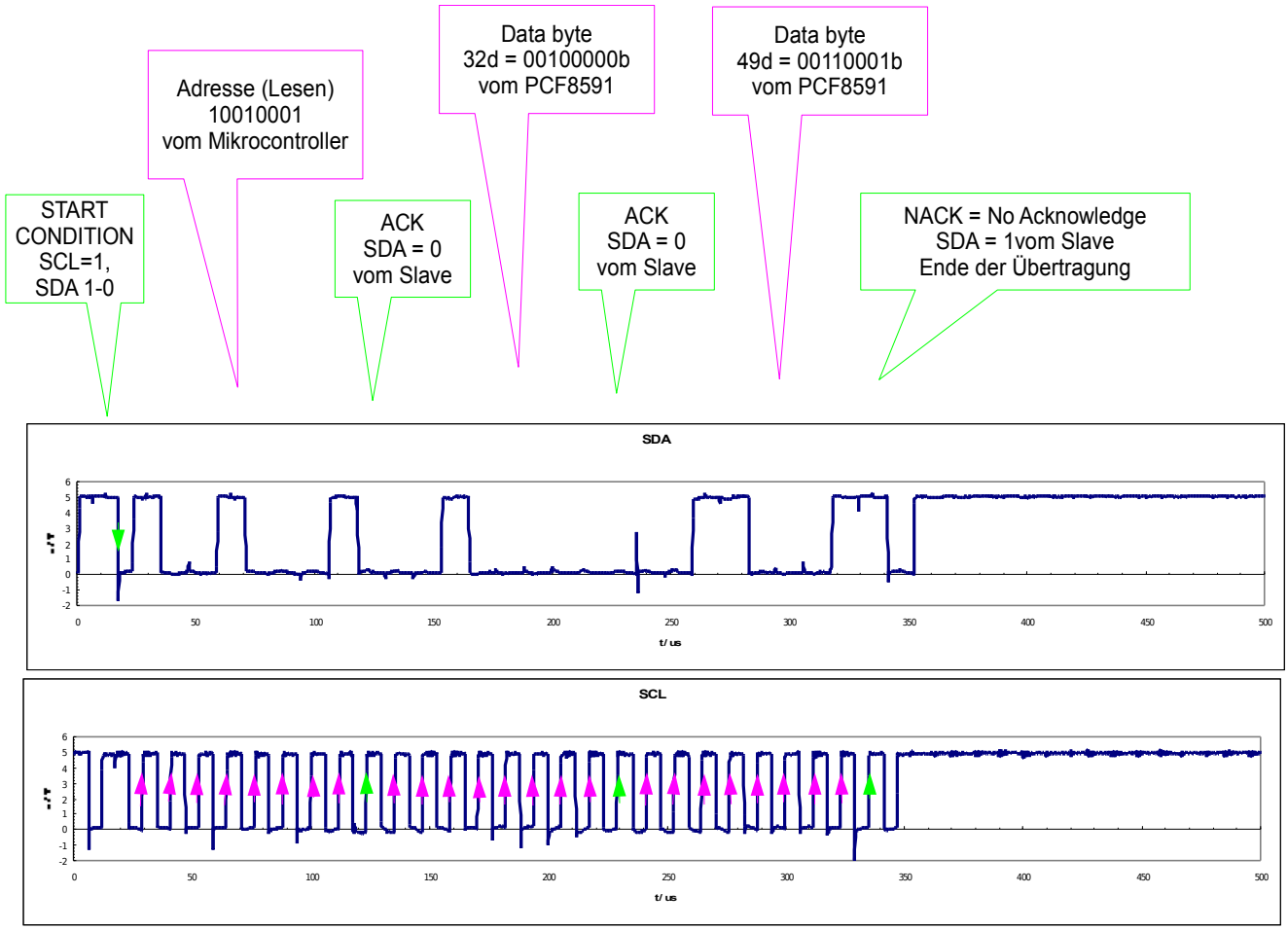
Beim Debuggen mit dem Oszilloskop ist es schwierig, wenn nicht unmöglich, die ganze Impulsfolge anzuschauen. Ein guter Trick ist dabei, einen Portpin als Ausgang zu nutzen und dort ein Synchronisationssignal zum externen Triggern des Oszilloskops auszugeben (Port setzen und zu gegebener Zeit wieder rücksetzen).

Dies wurde bei den folgenden Oszillogrammen gemacht, so dass nur der für das Lesen interessante Teil sichtbar ist.



Die gelesenen Bytes waren hier 32 (voriger Wert) und 49 (aktueller Wert).

Bei der Analyse des Oszillogramms fällt auf, dass man mit der Interpretation aufpassen muss: trotz Triggerung ist der Beginn des Oszillogramms nicht exakt der Beginn der Start-Kondition, sondern es wird schon ein kleines Stück davor aufgezeichnet. Dies ist möglicherweise von Oszilloskop zu Oszilloskop verschieden.



Geschwindigkeit der Übertragung

Die Taktrate der Übertragung wird von einer internen Zeitverzögerung gesteuert.

Dies ist mit dem Befehl `Config I2cdelay= n` einstellbar.

Dabei ist n eine Zahl zwischen 0 und 255.

Der Defaultwert 5 soll eine Taktrate von 100kHz ergeben. Gemessen habe ich eine etwas geringere Frequenz mit einer Periodendauer von 12µs.

Bei einem Wert von 1 ergab sich eine Periodendauer von etwa 4µs, wobei einige Perioden zwischendurch verlängert waren. Dies ist nicht verwunderlich da der PCF8591 nur für 100kHz spezifiziert ist.

Hardware- oder Software I2C?

Software I2C

Standardmäßig bietet BASCOM eine Software-Emulation des I2C-Protokolls.
Dies hat den Vorteil dass im Prinzip jeder Pin des Controllers als SDA bzw. SCL genutzt werden kann.
Es funktioniert also z.B. beim Mega8 nicht nur

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

sondern auch

```
Config Sda = Portc.0
Config Scl = Portc.1
```

Diese Möglichkeit kann von Vorteil sein, wenn man noch genügend freie Pins hat und mehrere Slaves anschließen will, die zufällig die gleiche Adresse haben.

Hardware-I2C

Die modernen Controller haben eine Hardware-I2C-Schnittstelle.
Diese soll laut BASCOM-Hilfe nutzbar sein, indem man die entsprechende Bibliothek einbindet.
Das geschieht mit dem Befehl

```
$lib "i2c_TWI.Lbx"
```

Bei Roland Walter findet sich eine reine BASIC-Ansteuerung der Hardware-I2C-Register.

I2C Bus Scan

Oft möchte man wissen, welche I2C-Adressen am Bus präsent sind, oder ob ein komplexer Chip überhaupt irgendwie reagiert. In der BASCOM-Hilfe ist eine Methode beschrieben die einfach zu realisieren ist: Es wird für jede gerade Adresse[1] von 0...254 geprüft, ob es einen Slave gibt, der ein Acknowledge zurücksendet. Dies ist aus der internen BASCOM-Variablen ERR ersichtlich. Wenn ein ACK kommt, ist ERR = 0.

```
.....
'I2C
Config Sda = Portc.1
Config Scl = Portc.0
Dim Busaddress As Byte
Dim Busaddress_read As Byte
Dim Chipaddress As Byte

.....
'-------------------------------------------------------------------------------
Scanbus:
'Scan bus for valid I2C addresses on bus
' ( test for ACK to come back from Busaddress )
   Print
   Print "Scan start"
   For Busaddress = 0 To 254 Step 2     'for all even addresses
      I2cstart                          'send start
      I2cwbyte Busaddress               'send Busaddress
      If Err = 0 Then                   'we got an ack
            Print "Slave at : " ; Busaddress ; "d,  =  " ; Hex(busaddress) ; "hex"
            Chipaddress = Busaddress \ 2
            Print " with chip address " ; Hex(chipaddress ) ; "h“
            Print
       End If
      I2cstop                           'free bus
   Next
   Print "End Scan"
Return
'-------------------------------------------------------------------------------
```

1 Jeder Chip belegt 2 Adressen: eine zum Lesen und eine zum Schreiben.

Interessante Links

http://www.qsl.net/pa3ckr/bascom%20and%20avr/i2c/index.html

Chip-Übersicht:

http://www.rn-wissen.de/index.php/I2C_Chip-%C3%9Cbersicht

Rückmeldungen und Kritik bitte an
jean-claude.feltes@education.lu

Anhang: Control Byte des PCF8591