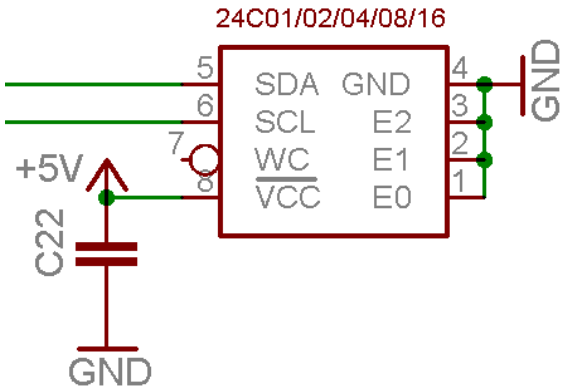
12.0 Ein serielles EEProm an der I²C Schnittstelle

Eine serielles EEProm wird häufig zur Speicherung von Systemkonfigurationen verwendet und könnte z.B. die optimalen Betriebsfrequenzen des Elliptecmotors oder die Daten einer Wav Datei für eine Soundausgabe speichern.

12.1 Die Schaltung eines 16kB EEProms

Der Anschluss eines seriellen 16kb EEProms an die I²C Schnittstelle ist denkbar einfach: Hier als Beispiel mit einem ST24C16:



SDA ist der Datenanschluss, SCL der Clock PIN und die Pins 1-3 bestimmen die letzten drei Adressbits der I²C Adresse. WC ist Write Control und kann, wenn auf Masse gelegt, ein unbeabsichtigtes Schreiben oder löschen des EEProms verhindern.
Der ST Baustein hält mindestens 1 Million Schreib- und Lesezyklen aus und unterstützt an der I²C Schnittstelle eine Frequenz bis ca. 400khz. Ein einmal beschriebener Speicher hält ohne Versorgungsspannung mindestens 40 Jahre seine Daten.

12.2 Die Funktionen des 16kB EEProms mit I²C

Der Speicher kann byteweise oder seitenweise gelesen und beschrieben werden. Beim seitenweisen Schreiben bis zu 16 Bytes wird die Speicheradresse intern automatisch immer um eins erhöht.

Nach der Start Kondition wird über I²C ein Device Select Byte (Das Adressbyte DEVSEL) mit einem Read oder Write Wert übermittelt. Die Bits B7 bis B4 der Adresse sind beim ST24Cxx mit 1010 vorgegeben. Bei dem 16kb EEprom kann hier nur ein EEPROM an I²C angeschlossen werden und die Bits B3-B1 sind damit auf High Pegel. Wie bei I²C üblich bestimmt das Bit B0 des ersten I²C Adressbytes, ob geschrieben oder gelesen werden soll.
Danach wird die 8 bit Startadresse des Speichers übermittelt ab die Daten der geschrieben oder gelesen werden sollen. Beim seitenweisen Schreiben können bis zu 16 Datenbytes übermittelt werden. Danach muss eine neue Startadresse übermittelt werden.
Beim Lesen können, nach der Startadresse alle nachfolgenden Daten ohne weitere Übermittlung der Startadresse sequentiell gelesen werden.
Eine derzeit noch rein theoretisch Frage: würde man ein vorprogrammiertes serielles EEProm als Frequenzteiler für die Erzeugung der Betriebsfrequenz eines Elliptecmotors verwenden können?

12.3 Assembler Datei: 256 Bytes des 16kB EEProms lesen

```
; 256 Bytes aus dem EEProm lesen
; DEVSEL senden, gibt bei Fehler eine 0 zurück
; Adresse senden und danach wird Leseadresse bestätigt
; und dann 256 Bytes holen
; R5 wird verwendet!

REPROM   acall RX            ; get I2C DEV SEL adr
         mov   r1,A
         acall I2Start
         jnc   REN1
         mov   A,#0
         acall TX            ; confirm adress to user, 0 =
error
         setb c
         ret                 ; return for restart

REN1     acall I2Out         ; DEV SEL
         acall TX            ; confirm to user, 0 = error
         acall RX            ; get I2C Databyte address
         acall I2Out
         acall TX            ; confirm adress to user, 0 =
error
         mov A,r1
         inc A               ; enable Read of DEV
         acall I2Start
         acall I2Out         ; DEV SEL
         acall TX            ; confirm to user, 0 = error

; und jetzt einfach sequenziell hinterher:
         mov   R5,#0FEH
REN2     acall I2In          ; read byte
         acall I2Ack         ; ein Ackn
         acall TX            ; send data to user
         djnz  R5, REN2

; und zuletzt kein ack
         acall I2In          ; read byte
         acall I2Nak
         acall I2Stop
         acall TX            ; send data to user
         clr c
         ret
; ###    END READ EEPROM
```

12.4 256 Bytes in Visual Basic zum PC übertragen

Und hier ein Beispiel wie man in Visual Basic diese Daten dann einfach aus dem EEProm lesen kann:

```
' JH Aug 2005
DevSelNumber = 160              ‘ die I²CAdresse und read
Dim str_read As String

OPENCOM comconnection      ‘ serielle Schnittstelle öffnen

SENDBYTE DevSelNumber

If READBYTE <> DevSelNumber Then _
            MsgBox "Error Reading Device: " & DevSelNumber

SENDBYTE 0              ' send first adress 0 to read from

If READBYTE <> 0 Then _
            MsgBox "Error Reading Device " & DevSelNumber & " adress: " & i - 1

For i = 0 To 255
  str_read = str_read & Chr$(READBYTE)
Next i

, Der Inhalt ist jetzt im String str_read
```

12.5 Assembler Datei: 16kB EEProm Bytes schreiben

Ein Assembler Beispiel um Daten an eine bestimmte Adresse im Datenspeicher des EEProms zu schreiben:

```
; ### Write EEPROM
; Ein Byte an eine Speicheradresse schreiben!

WEPROM   acall RX            ; get I2C adr from user DEVSEL
         acall I2Start
         jnc   WEN1
         mov   A,#0
         acall TX            ; 0 = error
         setb c
         ret                 ; return for restart
WEN1     acall I2Out
         acall TX            ; confirm to user, 0 = error
         acall RX            ; get I2C BYTE address
         acall I2Out
         acall TX            ; confirm adress to user, 0 =
error
         acall RX            ; get I2C Data IN for ser
EEPROM
         acall I2Out         ; write data to EProm
         acall TX            ; to user, 0 = error
         acall I2Stop
         clr c
         ret
; ###    END WRITE EEPROM
```

12.6 Assembler Datei: I²C und serielle Steuerung

Und dazu noch die (Standard) I²C Programmierung und serielle Kommunikation:

```
; ### I2C Progs

Delay    nop                 ; I2C command delay
         nop                 ; and DA strech
         nop
         nop
         ret

I2Start setb SDA             ; JH carry flag added
        setb SCL
        jnb  SDA, st40       ; Verify bus available
```

```
        jnb  SCL, st40
        acall Delay
        clr   SDA
        acall Delay
        clr   SCL
        clr c
        sjmp st41
st40    setb c
st41    ret


I2Stop   clr   SCL
         clr   SDA
         acall Delay
         setb  SCL
         acall Delay
         setb  SDA
         ret

I2Ack    clr   SDA
         acall Delay
         setb  SCL
         acall Delay
         clr   SCL
         acall Delay
         ret

I2Nak    setb  SDA
         acall Delay
         setb  SCL
         acall Delay
         clr   SCL
         acall Delay
         ret

; value in A

I2Out    mov   r3,#8          ; 8 bits
S0       jb    ACC.7,S1       ; bit 7 = 1?
         clr   SDA            ; bit = 0
         sjmp  S2             ; bit = 1
S1       setb  SDA            ; bit = 1
S2       acall Delay
         setb  SCL            ; clock
         acall Delay
         clr   SCL
         rl    a              ; next bit in A
         djnz  r3,S0          ; 8 bits
         setb  SDA            ; SDA high Z
         acall Delay
         setb  SCL            ; clock 9
         acall Delay
         jb    SDA,Err        ; Ack?
         clr   SCL
```

```
         clr   SDA
         acall Delay
         ret
Err      mov   A,#0           ; error: return 0
         clr   SCL
         clr   SDA
         acall Delay
         ret

; value in A
I2In     setb  SDA            ; SDA high Z
         mov   A,#0
         mov   r3,#8          ; 8 bits
S4       rl    A              ; next bit
         setb  SCL            ; clock
         acall Delay
         jnb   SDA,S5         ; SDA = 0?
         inc   A              ; Bit 0 = 1
S5       clr   SCL
         acall Delay
         djnz  r3,S4          ; 8 bits
         ret


; ### END I2C

RX       jnb   RI,RX         ; get value from user
         mov   A,SBUF
         clr   RI
         ret

TX       jnb   TI,TX         ; committ value
         clr   TI
         mov   SBUF,A
         ret
```