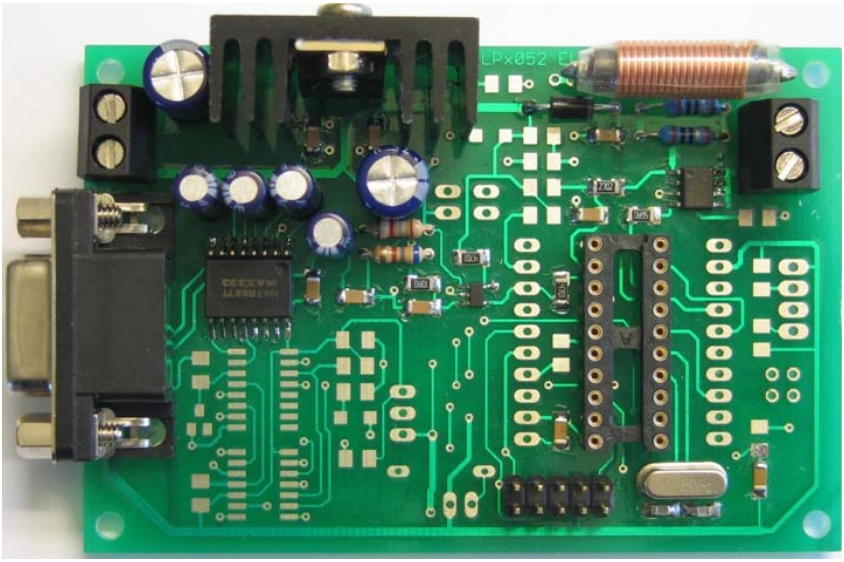
11.0 Aufbau für die Steuerung des Elliptecmotors

Die Möglichkeiten der ISP Programmierung der Ghost Entwicklungsplatine und Steuerung über die serielle Schnittstelle vereinfachen die Versuche mit der Steuerung des Elliptecmotors. Das Ghost Board dient als 8051 Beispiel, um sich grundlegend in die Thematik einarbeiten zu können. Der Elliptecmotor ist auf einer fertigen Mechanik montiert und die Beispielprogramme helfen die Funktionsweise zu veranschaulichen.

Parameter können zur Laufzeit verändert und über die serielle Schnittstelle übertragen werden. Soll eine programmtechnische Änderung des Mikrocontrollers erfolgen, dann wird einfach das veränderte Programm ebenfalls seriell übermittelt und der Mikrocontroller neu gestartet.

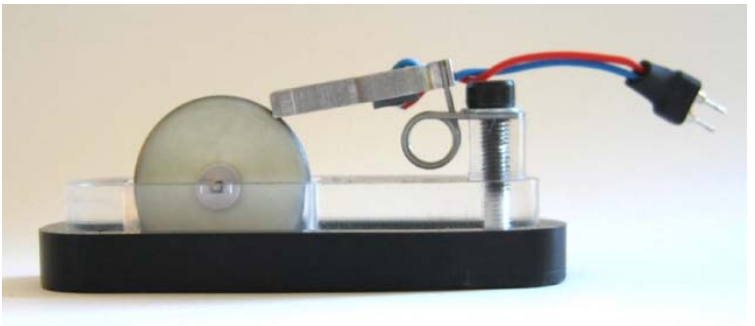
Mit einem Oszilloskop lassen sich so schnell die Signale am Piezo während der unterschiedlichsten Bedingungen über das gesamte Frequenzspektrum kontrollieren und damit die beste Soft - Hardwarekombination für die eigene Applikation finden.

Das Bild zeigt die einfachste Bestückung der Prototypenplatine für den Elliptecmotor mit serieller Schnittstelle, Leistungselektronik und serielle Steuerung über den PC.



Natürlich kann die Steuerung auch mit anderen Mikrokontrollern erfolgen. Mit mehr Erfahrungen können dann Verbesserungen für eine optimierte Ansteuerung des Elliptecmotor in Zusammenhang mit der geplanten Mechanik durchgeführt werden.

11.1 Aufbau der Original Antriebsmechanik



Das Bild zeigt eine Original Elliptec Motoreinheit. Das Antriebsrad hat einen Durchmesser von 20 mm; die Achse 1 mm. Die Kabelanschlüsse werden direkt mit dem Ghost Board verbunden.

Der Elliptecmotor muss sich in einem Winkel von 50° (+/- 1°) zur Antriebsscheibe befinden. Eine Verringerung des Winkels erhöht die Rückwärtsperformance und reduziert die Vorwärtsperformance.
Für den Kontaktpunkt an der Spitze des Stators zur Antriebsscheibe benötigt die Antriebsscheibe eine Abrundung mit einem Radius von 1.5mm.
Im Betriebsfall werden sie einen dunklen Abrieb der Antriebsscheibe feststellen. Vermeiden Sie Verunreinigungen am Rand der Antriebsscheibe z.B. durch einfache Berührung. Dieses wird dann wahrscheinlich zum Stillstand des Motors in dem verschmutzen Bereich führen.

Elliptec besitzt mehrjährige Versuchserfahrungen mit den verschiedensten Antriebsmaterialien. Die Antriebsscheibe besteht aus IXEF 1032 von Solvay. Auch andere Materialen sind denkbar. Weiche Kunststoffe würden einen hohen Abrieb nach sich ziehen. Harte Materialen, es wäre auch eine, am Umfang gerauhte Gasscheibe möglich, würden zu hörbarer Geräuschentwicklung beim Antrieb führen. Aluminium sollten Sie für Ihren Antrieb nicht verwenden, da

dieses zum verschweißen der beiden Aluminiumkomponenten führen würde.
Für Linearantriebe empfiehlt Elliptec den Kunststoff PF7595 von Bakalite.

Die Achse der Antriebsscheibe ist ohne Kugellager direkt auf dem Kunststoff gelagert. Bei einem Achsdurchmesser von 1 mm und Antriebsscheiben ab 15 mm Durchmesser entwickelt der Elliptecmotor genügend Kraft und stellt damit eine entsprechend günstige Lösung dar.

11.2 Die Steuerung des Piezo

Die Frequenz, Pulsweite und Amplitude der pulsierenden Spannung entscheiden hauptsächlich über den Betrieb des Elliptecmotors. Die Vorwärtsfrequenz eines Elliptecmotors liegt in einem Frequenzbereich von 73-85 kHz, die Rückwartsfrequenz zwischen 91 und 108 KHz. Jeder Elliptecmotor kann unterschiedliche Betriebsfrequenzen in diesen Bereichen haben. Die Betriebsfrequenz kann sich zudem mit der Temperatur verschieben. Für den Betriebsfall wird also eine mechanische oder elektronische Frequenzsuche benötigt.
Die optimale Betriebsfrequenz ist gleichzeitig der maximal zurückgelegte Weg.

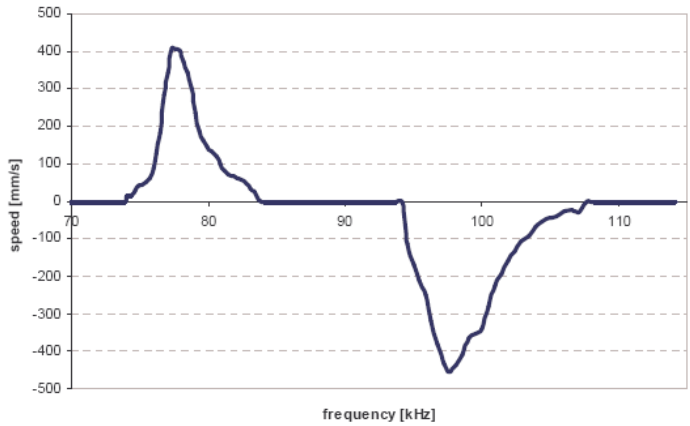


Figure 2[1]: Typical speed vs. frequency v(f)
(frequencies vary from motor to motor and depends on driver circuit)

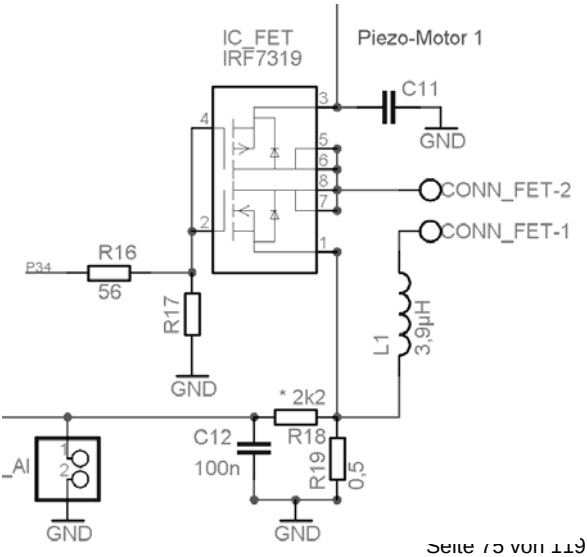
Zur Steuerung und Frequenzsuche schlägt Elliptec verschiedene Lösungsmöglichkeiten vor, die unter anderem im Internet bei www.elliptec.com zu finden sind.
Mit dem Ghost Entwicklungsboard sollen hier die Möglichkeiten zur einfachen Ansteuerung erläutert werden, um mit dem Elliptecmotor eine Antriebsscheibe vorwärts und rückwärts zu bewegen, damit ein Gefühl für eine Steuerung entwickelt werden kann.

11.3 Die Hardware zur Ansteuerung

Auf der einen Seite muss die Hardware Betriebsfrequenzen zwischen 75 und 110 KHz, möglichst mit variabler Pulsbreite generieren und auf der anderen Seite Möglichkeiten einer Frequenzsuche für die optimalen Betriebsfrequenzen unterstützen.

Die Taktsignale des Mikrocontrollers steuern den Elliptecmotor über eine Leistungselektronik, die unterschiedlich ausgelegt werden kann. Im einfachsten Fall wäre ein Schalttransistor oder FET ausreichend.
Durch die Verwendung von Spulen kann der Elliptecmotor mit geringeren Versorgungsspannungen energiesparender betrieben werden. In diesem Fall ist aber Vorsicht geboten, da steile Flanken eines Rechtecksignals hohe Spannungsspitzen erzeugen, die zur Zerstörung des Elliptecmotors führen können.
Bei geringen Spannungen könnte auch ein Transformator oder ein Vollbrückentreiber verwendet werden.

Das Ghost AT89LPx052 Entwicklungsboard ist mit einem Halbbrückentreiber bestückt. Der Piezo wird an die Schraubklemmen Conn_FET-2 angeschlossen (die blaue Zuleitung des Elliptecmotors an Klemme 1) und liegt damit in Reihe zur Spule L1 und



zum Widerstand R19. R19 wurde aus 2 parallel geschalteten ¼ Watt Widerständen a 1 Ohm gebildet (im endgültigen SMD Layout wurde R19A hinzugefügt).
Über R18 wird der Spannungsverlauf, bzw. die Stromaufnahme des Piezo als Feedback dem AT89LPx052 internen Komparator zugeführt.
Die Stromaufnahme kann bei unterschiedlichen Frequenzen für eine Betriebsfrequenzsuche herangezogen werden.

Das Taktsignal wird an PIN 3.4 des Atmel AT89LPx052 Mikrocontrollers generiert und den Gates über R16 zugeführt. R17 wird für den verwendeten IRF Typ nicht bestückt.

Wichtiger Hinweis:
Wenn Sie eigene Mikrocontroller Routinen entwickeln und den FET verwenden, dann sollten Sie entweder durch löschen von P3.4 bei der Initialisierung oder durch einen entsprechenden Widerstand R17 das Gate auf Masse legen. Auch ohne angeschlossenen Elliptecmotor werden Sie sonst, durch hohen Stromfluss über den FET, eine Überhitzung des 7805 Spannungsregler herbeiführen.

In der +5Volt Zuleitung zu Pin 3 des FET's befindet sich die Diode D3 die, je nach Ausführung, einen Spannungsabfall von 0,3 bzw. ca. 0,6 Volt erzeugt. Im Design wird eine BA146 verwendet. Diese Diode kann für andere Tests auch ganz entfallen oder durch einen Widerstand ersetzt werden.
Die Spule L1 ist eine Funkentstördrossel, die bei unterschiedlichen Anbietern in verschiedenen Ausführungen bezogen werden kann. Da manchmal 3.9 µH nicht zu erhalten sind können auch von einer 5 µH Spule einige Windungen entfernt werden. Wichtig ist eine ausreichende Auslegung der Strombelastbarkeit der Spule damit diese im Betrieb (ca. 350mA bis 1 A) nicht gesättigt wird und damit zur Zerstörung des Elliptecmotors führen würde.
Bei GMS Bentheimer Softwarehaus GmbH (www.gms2000.de) steht zudem eine Original Würth 3.9µH Spule für SMD Designs zur Verfügung, die speziell für das Original Elliptec Design entwickelt und im überarbeiteten Ghost Layout ebenfalls berücksichtigt wurde.

11.4 PWM des Atmel AT89LPx052

PWM Frequenzen an Pin 3.4 ließen sich mit den ersten Mustern der

Atmel LP Familie nur bis 1/512 bzw. 1/256 der Oszillatorfrequenz erzeugen. Zu wenig für die Ansteuerung der Elliptecmotoren. Die Single Cycle 8051 Corelogic des Atmel Mikrocontrollers erlaubt aber, wie jeder 8051 Mikrocontroller, eine interrupt gesteuerte Erzeugung von Frequenzen und Pulsbreiten mit Frequenzen von bis zu 1 MHz bei 11.059 MHz Oszillatorfrequenz. Man muss nur Obacht geben, dass der Interrupt noch schnell genug abgearbeitet werden kann bis der nächste Interrupt eintrifft, damit der Baustein nicht ins Stolpern gerät. Die verbleibende Zeit zwischen den Interrupts sollte für viele Applikationen noch ausreichend sein.

Bei 11.069MHz Taktfrequenz (ca. 90ns Cycle time) können bei 80-100 KHz Pulsfrequenz Schrittweiten von ca. 600 Hz erreicht werden. Bei 20 Mhz Taktfrequenz (50 ns Cycle time) sind ca. 350 Hz Schritte bei der Impulserzeugung erreichbar.

11.5 Die Programmierung zur Steuerung

Das erste, einfache Assembler Programm zu Erzeugung der Betriebsfrequenzen:

```
; test_elliptec piezo current with serial control
; 11.059 MHz Quatz max ca. 247 kHz
; settings with R1=R2=200 and RH0=255
; we have a frequenz of 98,742 kHz
; every step in R1 or R2 will result in
; time = time + step*1/11.0592 MHz (=90,42 ns)

#include LPx052.H
         .org 0000H
         sjmp start
         .org 000BH          ; TF0 Timer 0 overflow
         cpl P3.4            ; toggle Port pin
         jb  P3.4, pwm_on    ; next on or off ?
pwm_off: mov RL0, R2         ; set off time
         reti
pwm_on:  mov RL0, R1         ; set on time
         reti

start:   mov   P1M0,#01H     ; set ports to quasi bi-direc
         mov   P1M1,#00H     ; and P1.0 to input only
         mov   P3M0,#00H     ; (ggf auch P1.1 wenn Comp)
         mov   P3M1,#10H     ; push pull output for P3.4
         mov   SP,#20H       ; Stack pointer
         clr   TR1           ; stop timer 0 / 1
         clr   TR0
         mov   TH1,#0DCH     ; 256-6: 9600 baud
         mov   TL1,#0DCH     ; 11.059MHz for SMOD1 =0
```

```
         anl   TMOD,#00H     ; Timer1: 8 bit auto-reload
         orl   TMOD,#20H
         setb  TR1           ; TCON start timer 1
         anl   PCON,#3FH     ; PCON: clear SM0D0 and SM0D0
         mov   SCON,#50H     ; InitRS232 8 bit UART Mode1
         setb  TI
         orl   PCON,#80H     ; SMOD=1 double Baudrate
                             ; Timer 0
         mov   RL0,#0BAH     ; set first RL and RH
         mov   RH0,#0FFH     ; default settings
         mov   R1,#0C8H      ; = 98,742 khz with 11.059MHz
         mov   R2,#0C8H      ; = 200
         orl   TMOD,#1       ; 16 bit autoreload
;        setb  TR0           ; start timer 0
         clr   P3.4          ; to reduce power consumption
         mov   IE,#82H       ; Interrups EA+ET0


; RL0:  R1 ON Time, R2 OFF Time;


NEXT     acall RX
         mov   R1,A          ; R1 write ON time
         nop
         mov   A,R1          ; R1  read
         acall TX            ; confirm to user
         acall RX
         mov   R2,A          ; R2 write OFF time
         nop
         mov   A,R2          ; R2  read
         acall TX            ; confirm to user
         acall RX            ; start or stop timer 0
         anl   A,#1          ; 0 to stop - 1 to start timer
         jz    TimerOFF
         setb  TR0
         sjmp  TimerON
TimerOFF clr   TR0
         clr   P3.4          ; to reduce power consumption
TimerON  acall TX            ; confirm to user
         sjmp  NEXT

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
        .end
```

Der Timer0 Overflow erzeugt einen Interrupt (Adresse 0BH). Mit jedem Interrupt wird, je nach aktuellem Zustand von P3.4 der Ausgang umgeschaltet. Register R1 und R2 werden durch die serielle Schnittstelle gesetzt. Eine Schleife wartet auf eine serielle Übermittlung und setzt die Register R1 und R1.

Das Programm verwendet die Include Datei Lpx052.H. In dieser Datei sind u.a. die Register definiert. Die Original 8051 Header Datei wurde speziell für den Atmel LPx052 geändert:

```
;*****************************************************
;*  TASM LPx052  SFR BIT/BYTE MNEMONIC EQUATES LIST  *
;*  GMS 2005 added equations for Atmel AT89LPx0523   *
;*****************************************************

SP      .equ    081H    ;Stack pointer
DPL     .equ    082H
DPH     .equ    083H
SPDR    .equ    085H    ;SPI Data register AT89LPx0523
PCON    .equ    087H
TCON    .equ    088H
TMOD    .equ    089H
TL0     .equ    08AH
TL1     .equ    08BH
TH0     .equ    08CH
TH1     .equ    08DH
P1      .equ    090H    ;Port 1
TCONB   .equ    091H    ;Atmel AT89LPx052
RL0     .equ    092H    ;Atmel AT89LPx052
RL1     .equ    093H    ;Atmel AT89LPx052
RH0     .equ    094H    ;Atmel AT89LPx052
RH1     .equ    095H    ;Atmel AT89LPx052
ACSR    .equ    097H    ;AT89LPx052
SCON    .equ    098H
SBUF    .equ    099H
WDTRST  .equ    0A6H    ;AT89LPx052 Watchdog
WDTCON  .equ    0A7H    ;AT89LPx052 Watchdog Control register
IE      .equ    0A8H
SADDR   .equ    0A9H    ;Slave Address Atmel AT89LPx052
SPSR    .equ    0AAH    ;Atmel AT89LPx052
P3      .equ    0B0H    ;Port 3

IP      .equ    0B8H
SADEN   .equ    0B9H    ;Atmel AT89LPx052 Slave Address enable
P1M0    .equ    0C2H    ;Atmel AT89LPx052 Port config
P1M1    .equ    0C3H    ;Atmel AT89LPx052
P3M0    .equ    0C6H    ;Atmel AT89LPx052
P3M1    .equ    0C7H    ;Atmel AT89LPx052
T2CON   .equ    0C8H    ;8052, 80154 only
RCAP2L  .equ    0CAH    ;8052, 80154 only
RCAP2H  .equ    0CBH    ;8052, 80154 only
```

```
TL2     .equ    0CCH    ;8052, 80154 only
TH2     .equ    0CDH    ;8052, 80154 only
PSW     .equ    0D0H
SPCR    .equ    0D5H    ;SPI control register Atmel AT89LPx052
ACC     .equ    0E0H    ;Accumulator
B       .equ    0F0H    ;Secondary Accumulator
IOCON   .equ    0F8H    ;80154 only

;PORT 1 BITS
P1.0    .equ    090H    ;Port 1 bit 0
P1.1    .equ    091H    ;Port 1 bit 1
P1.2    .equ    092H    ;Port 1 bit 2
P1.3    .equ    093H    ;Port 1 bit 3
P1.4    .equ    094H    ;Port 1 bit 4
P1.5    .equ    095H    ;Port 1 bit 5
P1.6    .equ    096H    ;Port 1 bit 6
P1.7    .equ    097H    ;Port 1 bit 7

;PORT 3 BITS
P3.0    .equ    0B0H    ;Port 3 bit 0
P3.1    .equ    0B1H    ;Port 3 bit 1
P3.2    .equ    0B2H    ;Port 3 bit 2
P3.3    .equ    0B3H    ;Port 3 bit 3
P3.4    .equ    0B4H    ;Port 3 bit 4
P3.5    .equ    0B5H    ;Port 3 bit 5
P3.6    .equ    0B6H    ;Port 3 bit 6
P3.7    .equ    0B7H    ;Port 3 bit 7

;ACCUMULATOR BITS
ACC.0   .equ    0E0H    ;Acc bit 0
ACC.1   .equ    0E1H    ;Acc bit 1
ACC.2   .equ    0E2H    ;Acc bit 2
ACC.3   .equ    0E3H    ;Acc bit 3
ACC.4   .equ    0E4H    ;Acc bit 4
ACC.5   .equ    0E5H    ;Acc bit 5
ACC.6   .equ    0E6H    ;Acc bit 6
ACC.7   .equ    0E7H    ;Acc bit 7

;B REGISTER BITS
B.0     .equ    0F0H    ;Breg bit 0
B.1     .equ    0F1H    ;Breg bit 1
B.2     .equ    0F2H    ;Breg bit 2
B.3     .equ    0F3H    ;Breg bit 3
B.4     .equ    0F4H    ;Breg bit 4
B.5     .equ    0F5H    ;Breg bit 5
B.6     .equ    0F6H    ;Breg bit 6
B.7     .equ    0F7H    ;Breg bit 7

;PSW REGISTER BITS
P       .equ    0D0H    ;Parity flag
F1      .equ    0D1H    ;User flag 1
OV      .equ    0D2H    ;Overflow flag
RS0     .equ    0D3H    ;Register bank select 1
RS1     .equ    0D4H    ;Register bank select 0
```

```
F0      .equ    0D5H    ;User flag 0
AC      .equ    0D6H    ;Auxiliary carry flag
CY      .equ    0D7H    ;Carry flag

;TCON REGISTER BITS
IT0     .equ    088H    ;Intr 0 type control
IE0     .equ    089H    ;Intr 0 edge flag
IT1     .equ    08AH    ;Intr 1 type control
IE1     .equ    08BH    ;Intr 1 edge flag
TR0     .equ    08CH    ;Timer 0 run
TF0     .equ    08DH    ;Timer 0 overflow
TR1     .equ    08EH    ;Timer 1 run
TF1     .equ    08FH    ;Timer 1 overflow

;SCON REGISTER BITS
RI      .equ    098H    ;RX Intr flag
TI      .equ    099H    ;TX Intr flag
RB8     .equ    09AH    ;RX 9th bit
TB8     .equ    09BH    ;TX 9th bit
REN     .equ    09CH    ;Enable RX flag
SM2     .equ    09DH    ;8/9 bit select flag
SM1     .equ    09EH    ;Serial mode bit 1
SM0     .equ    09FH    ;Serial mode bit 0

;IE REGISTER BITS
EX0     .equ    0A8H    ;External intr 0
ET0     .equ    0A9H    ;Timer 0 intr
EX1     .equ    0AAH    ;External intr 1
ET1     .equ    0ABH    ;Timer 1 intr
ES      .equ    0ACH    ;Serial port intr
ET2     .equ    0ADH    ;Timer 2 intr
;Reserved       0AEH    Reserved
EA      .equ    0AFH    ;Global intr enable

;IP REGISTER BITS
PX0     .equ    0B8H    ;Priority level-External intr 0
PT0     .equ    0B9H    ;Priority level-Timer 0 intr
PX1     .equ    0BAH    ;Priority level-External intr 1
PT1     .equ    0BBH    ;Priority level-Timer 1 intr
PS      .equ    0BCH    ;Priority level-Serial port intr
PT2     .equ    0BDH    ;Priority level-Timer 2 intr
;Reserved       0BEH    Reserved
PCT     .equ    0BFH    ;Global priority level
```

Und so einfach lässt sich dieses mit VB steuern:

```
SENDBYTE Wert1          R1 write ON time
Dummy =  READBYTE
SENDBYTE Wert2          R2 write OFF time
Dummy =  READBYTE
SENDBYTE 1              start oder SENDBYTE 0 = stop
```

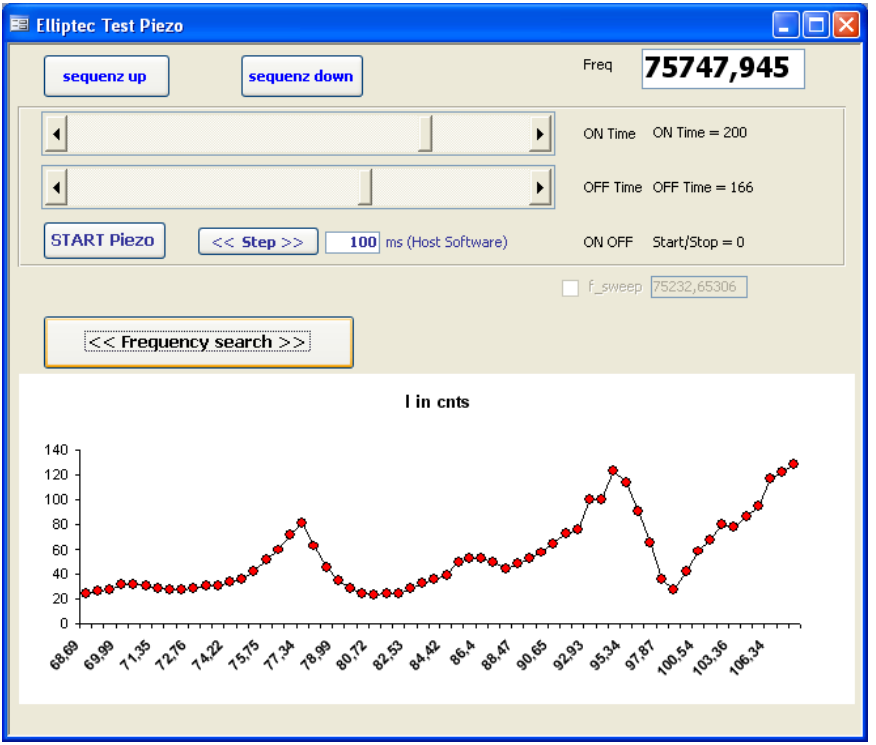
```
Dummy = Readbyte
```

Die Register R1 und R2 sind im Assemblerprogramm auf die Werte von 200 für RL0 und FFH für RH0 voreingestellt.
D.h. nach jeweils 256-200 Taktimpulsen erhalten Sie einen Interrupt der den Ausgang P3.4 umschaltet.

11,059200Mhz / (256- 200) + (256 -200) ergeben 11,059200Mhz / 112 = 98,742 KHz

Wird R1 um eins erhöht, dann erhöht sich die Periodendauer um ca. 90 ns und es ergeben sich 11,059200Mhz / 113 = 97,869 KHz. Mit der Veränderung von R1 und R2 wird das Puls/Pausenverhältnis entsprechend verschoben. In dem folgenden Access Beispiel wird die berechnete Frequenz oben rechts angezeigt und mit jedem Verändern der Schieberegler aktualisiert.

11.6 Frequenztest für den Elliptecmotor



Das Frequenztestprogramm wurde in Microsoft Access XP entwickelt. Mit den Schaltern sequenz up bzw. sequenz down wird ein Frequenzband von ca. 70-110khz schrittweise angesprochen. Die Startfrequenz ist dabei von den beiden Schiebereglern abhängig. Also, Motor anschließen, Programm starten und einfach den Schalter sequenz up betätigen. Sie sollten, sofern die Kommunikation und das Board in Ordnung sind, kurze Vor- und Rückwärtsbewegungen der Piezomotormechanik erkennen können.
Mit dem unteren Schieberegler können Sie die Frequenzen manuell einstellen und dann einfach auf Start Piezo drücken um zu sehen ob sich der Motor bei dieser Frequenz bewegt. Sie können nach dem Start auch mit dem Schieberegler die Motorfrequenz im Betriebsfall ändern.

Lassen Sie den Motor nicht zu lange eingeschaltet, auch wenn dieser aufgrund falscher Frequenzeinspeisung nicht läuft, da die gesamte Energie in Wärme gewandelt wird und der Motor zerstört würde!

Frequency Search ermittelt die Stromaufnahme zu jeder Frequenz und wird in den nächsten Kapiteln detaillierter beschrieben.

Und hier ein Auszug aus dem Access VBA Programm:

Das folgende Programm wird durch die Schalter sequenz up bzw. sequenz down mit dem entsprechenden Index von 0 oder 1 aufgerufen:

```
Private Sub scan_frequenz(Index As Integer)
Dim scq_beginn As Integer
Dim scq_end As Integer
Dim schritt As Integer
Dim warte As Double
warte = 300
scq_beginn = 0
schritt = 1
sqq_end = 60

If Index = 1 Then  ' na dann anders herum
   scq_beginn = 60
   sqq_end = 0
   schritt = -1
End If

For i% = scq_beginn To sqq_end Step schritt
    Me.StartStopPiezo.Caption = "STOP Piezo"
    Me.HScroll2.Value = 150 + i
    Me.StartStopPiezo.Caption = "START Piezo"
    Sleep 50
    Call bt_send_Click
Next i
End Sub
```

Diese Prozedur wird von **scan_frequenz** aufgerufen:

```
Private Sub bt_send_Click()
SENDBYTE (HScroll1.Value) ' RL0
Text1.Value = "ON Time = " & READBYTE
SENDBYTE (HScroll2.Value) ' RH0
Text2.Value = "OFF Time = " & READBYTE
Me.Form.Repaint
  If Me.StartStopPiezo.Caption = "STOP Piezo" Then
     SENDBYTE 1 ' start
  Else
     SENDBYTE 0 ' stop
  End If
  Text3.Value = "Start/Stop = " & READBYTE
 ' jetzt auft 4tes Byte warten
While READBYTE = -1
Wend
Call FrequenzAnzeigen
End Sub
```

```
Private Sub FrequenzAnzeigen()
' Die Frequenz berechnen und anzeigen
Dim R_ges As Integer
Me.freq.Value = 11059200 / (512 - HScroll1.Value - Me.HScroll2.Value)
Me.f_max.Value = 11059200 / (512 + 1 - HScroll1.Value - Me.HScroll2.Value)
End Sub
```

Möchte man die Stromaufnahme des Piezo zu jeder Betriebsfrequenz ermitteln ist das Assemblerprogramm des Mikrocontrollers zu erweitern. Die Stromaufnahme wird durch ein Zählverfahren im Zusammenspiel mit dem Maxim DA/Wandler und dem Komparator des Atmel Mikrocontrollers ermittelt. In Zukunft kann dieses auch anders gelöst werden, da eine neue Atmel LP Mikrocontroller Version einen integrierten A/D Wandler besitzen wird.

11.7 Der Elliptecmotor im Schrittbetrieb

Der Elliptecmotor funktioniert optimal, wenn der Resonator mit der richtigen Frequenz schwingt. Die Frequenz liegt bei ca. 75-100KHz und kann durch heutige Elektroniken sehr gut gehandelt werden. Die Frequenz liegt außerhalb des Hörbereichs von Mensch und Tier, aber unterhalb der kritischen Frequenzen anderer Elektroniken.
Eine Frequenz steuert den Motor vorwärts und eine andere Frequenz steuert ihn rückwärts. Eine Umpolung der Versorgungsspannung, wie bei herkömmlichen DC-Motoren, ist nicht erforderlich. Die Laufrichtung wird nur von der Frequenz bestimmt.

Bei der optimalen Ansteuerfrequenz befindet sich der Resonator in seiner Resonanzfrequenz, d.h. die Spitze des Motors hat die maximale Amplitude und die Mikroschritte sind am größten. Die Geschwindigkeit des Motors nimmt ab je mehr die optimale Ansteuerfrequenz verlassen wird, bis zum Stillstand des Motors.

Je nach verwendetem Mikrocontroller kann aber die optimale Betriebsfrequenz nicht auf den Punkt genau erzeugt werden.
Bei 11.069MHz Taktfrequenz des Ghost Entwicklungsboards (ca. 90ns Cycle time) können bei 80-100 KHz Pulsfrequenz Schrittweiten von ca. 600 Hz erreicht werden. Bei 20Mhz Taktfrequenz (50 ns Cycle time) wären ca. 350 Hz Schritte für eine Impulserzeugung erreichbar.

11.7.1 Sweepen

Durch das variieren der Betriebsfrequenz im Bereich der Resonanzfrequenz (*‚sweepen'*) könnte der Elliptecmotor schrittweise in einem Frequenzbereich angesprochen werden. Diese Methode ist nicht ganz optimal, da außerhalb der optimalen Resonanzfrequenz noch einige Frequenzen vor und nach der optimalen Frequenz zur Bewegung des Motors beitragen. Für viele Anwendungen ist diese Methode jedoch ausreichend.

Jede einzelne Frequenz wird für einen kurzen Moment konstant eingestellt. Dann wird im nächsten Schritt die Frequenz erhöht und wiederum für einen Moment konstant gehalten.
Die Zeitabstände der einzelnen Frequenzschritte sollten, laut Elliptec, in einer Größenordnung von 0,5ms gehalten werden.

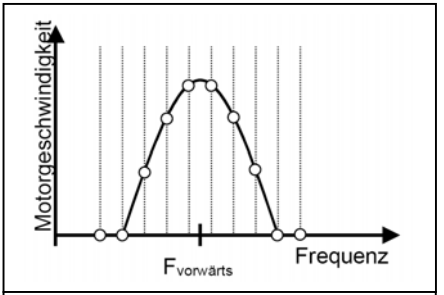


Bild: Sweepen

In diesem Beispiel würde sich der Motor bei der ersten und der zweiten Frequenz nicht bewegen. Erst bei der dritten Frequenz würde er sich langsam bewegen und mit den weiteren Frequenzen schneller werden bis zum Maximum. Im Weiteren würde die Geschwindigkeit wieder abnehmen bis zum Stillstand. Die ideale Frequenz wurde bei diesem Beispiel nicht exakt getroffen.

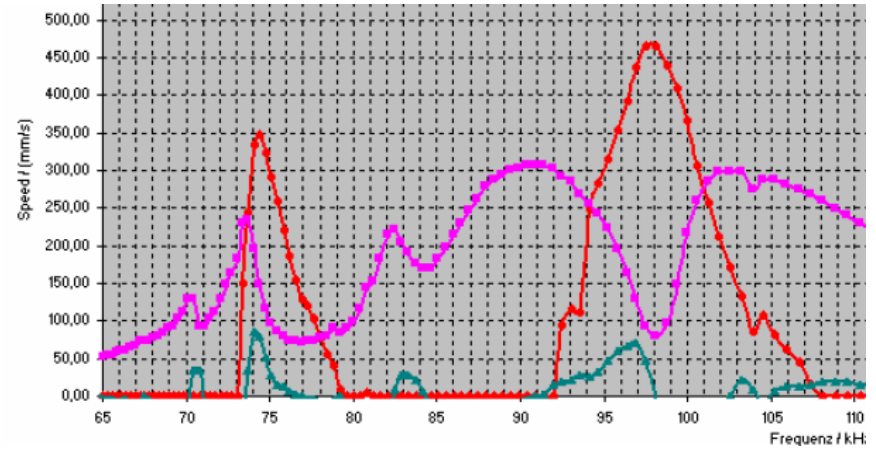
Die Wahl einer großen Frequenzbreite ist vorteilhaft für die Geräuschentwicklung und eine sichere Funktionsweise, aber die Geschwindigkeit und die Nutzungsbreite des Motors nehmen ab.

Bei der Wahl kurzer Schritte steigt die Chance die Resonanzfrequenz zu treffen. Aber bei einer Menge von kleinen Schritten steigt die Zeit zum Durchlaufen des Frequenzbereichs und der Motor läuft evtl. rauer.

11.8 Feedback

Der Elliptecmotor erlaubt sehr präzise Bewegungen. Einzelne Schritte können bei einer Weite von 3-4 Mikrometer liegen. Die genaue Schrittweite hängt von der Last, den Reibungsverhältnissen, der Genauigkeit der Steuerung und anderen Parametern ab.
Deshalb, und um den Motor mit seiner optimalen Frequenz ansteuern zu können, ist es empfehlenswert ein Feedbacksystem einzusetzen.

Eine Darstellung in den technischen Handbüchern der Elliptec AG zeigt den Zusammenhang zwischen Geschwindigkeit (rote Line) und dem Stromverbrauch an:



Betrachtet man die absolute Änderung des Stroms (die untere Linie), dann sind diese bei der maximalen Geschwindigkeit nahezu am größten. Dieses kann als erstes Kriterium für eine getrennte Suche nach Vorwärts- und Rückwärtslauf herangezogen werden. Beginnt man bei einer Frequenzsuche für den Vorwärtslauf (hier bei ca. 75khz) bei der kleineren Frequenz und für den Rückwärtslauf (hier bei ca. 98khz)

bei einer größeren, dann kann der Zwischenbereich ausgeschlossen werden.

Eine Frequenzsuche für die Vorwärtsbewegung könnte also wie folgt aussehen: bei einer schrittweisen Erhöhung der Frequenz ab 60khz wird zuerst der maximale Stromwert gesucht und gespeichert und danach der minimale Stromwert gesucht. Sofern die ermittelte Frequenz nicht genau erreicht werden kann, könnte man in diesem Bereich auch Sweepen.

Bei einer Rückwärtssuche, z.B. beginnend bei 130-140kHz bis 90kHz kann man entsprechend der Vorwärtssuche vorgehen. In diesem Fall ergibt sich die größte Stromänderung nach dem minimalen Stromwert (im Diagramm bei ca.98 kHz).

Beachten Sie:

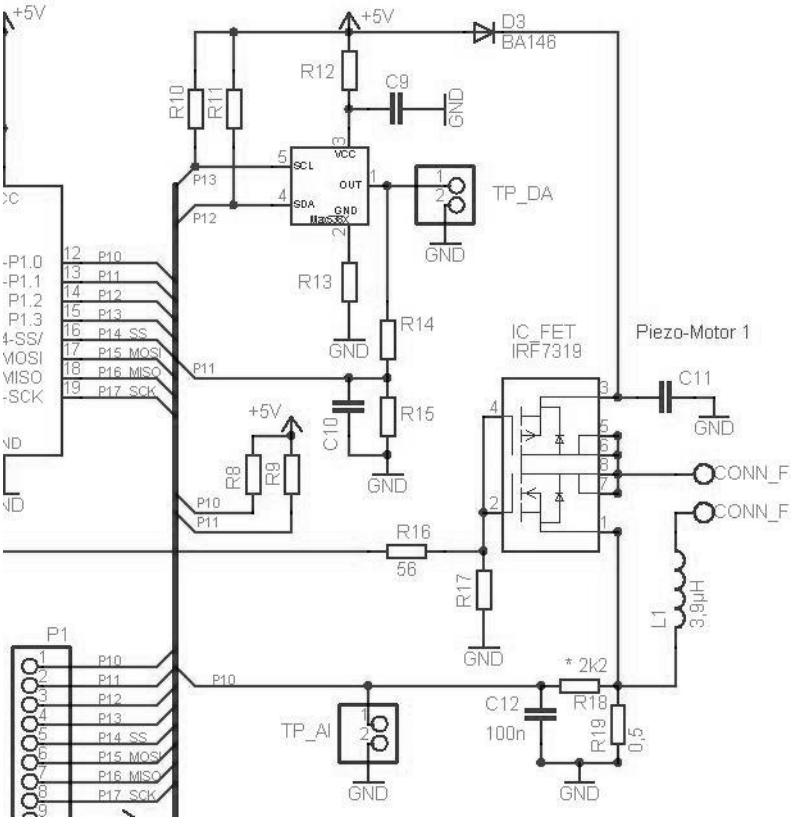
Diese Art der Suche setzt voraus, dass die Resonanzspule bei hohen Strömen (je nach Schaltung bis über 1 Ampere) nicht in die Sättigung gefahren wird!

Beachten Sie bitte weiterhin: Die gesamte Energie des Motors wird in Wärmeenergie umgewandelt, daher sollte der Elliptecmotor nicht im Dauerbetrieb eingesetzt werden. Dieses gilt auch dann, wenn der Motor bei einer Frequenz betrieben wird, die zu keiner Motorbewegung führt!

Der Motor arbeitet am besten bei Raumtemperatur. Er erwärmt sich selbst während des Betriebs und sollte nicht über 80°C erreichen. Bei ca. 150°C erreicht der Motor seine Depolarisierungstemperatur, die den Motor zerstört.
Die Betriebsfrequenz driftet mit verändern der Temperatur ab. Bei steigenden Temperaturen driftet die ideale Frequenz runter.

11.8.1 Einfache Strommessung mit A/D Wandler

Mit der Messung des Stromverbrauchs des Elliptecmotors lassen sich die optimalen Betriebsfrequenzen feststellen.



Über R18 wird der Spannungsverlauf, bzw. die Stromaufnahme des Elliptecmotors als Feedback dem AT89LPx052 internen Komparator an Port 1.0 zugeführt. Der Ausgang des D/A Wandler wird über den Widerstandsteiler R14 und R15 dem anderen Komparatoreingang an Port 1.1 zugeführt. Durch diese Teilung wird eine verbesserte Auflösung erreicht, da über den Widerstand R19 maximal ca. 0,5 Volt abfallen. Durch Ändern der Werte R19, R14 und R15 kann die Messung auf entsprechend andere Werte eingestellt werden. Der

Widerstand von 0,5 Ohm (zwei parallel geschaltete Widerstände von 1 Ohm) begrenzt den Strom bei 5 Volt auf 1 A.

Der Zustand des Komparators kann über den, nicht herausgeführten, Pin 3.6 im Polling Verfahren oder über einen Interrupt abgefragt werden.

Zum Assembler Programm im nachfolgenden Kapitel 11.8.2 :

Bei der seriellen Übermittlung werden die Registerwerte R1 und R2 für die Frequenz und das Start Stopp Signal übermittelt. Gleichzeitig wird der Wert des D/A Wandlers auf in R3 auf 0 gesetzt. Der Wert des D/A Wandlers wird in der nachfolgend abgebildeten Schleife solange um eins erhöht bis der Ausgang des Komparators kippt. Dann wird der Wert des D/A Wandlers in Register R3 wieder an den PC übermittelt. Wurden 255 Schritte überschritten wird der Zählvorgang abgebrochen.

Der eingesetzte Maxim Baustein 5382MEUK besitzt die feste Adresse 62H. Die SDA und SCL Signale sind durch das Hardwaredesign mit 92H bzw. 93H vorgegeben.

Ein Auszug aus der Erweiterung der vorherigen Assemblerdatei mit Programmierung des Maxim D/A Wandlers

```
nextval  acall I2Start
         mov   A,#62H         ; I2C adr
         acall I2Out          ;
         mov   A,R3           ; I2C data
         acall I2Out
         acall I2Stop
                              ; wait for settling time ?
         inc   R3             ; increase D/A value until
         jnb   P3.6,sendval   ; poll comparator P3.6
         mov   A,R3           ; up to 255
         jnz   nextval
sendval  acall TX             ; send to user
         sjmp  NEXT           ; next frequency
```

Dieses Verfahren ist nicht ganz optimal, da die analogen Signale auch mit Störimpulsen beaufschlagt sind. Zusätzlich ist direkt nach den steigenden und fallenden Flanken des Eingangssignals das Einschwingverhalten ein Problem.
Das Einschwingverhalten wurde durch eine Verzögerung in der Mikrocontrollersoftware eliminiert, Störimpulse einfach durch dreifach

wiederholendes Lesen der Hostsoftware. Dadurch konnte eine ausreichend gute Wiederholbarkeit für eine Frequenzsuche erreicht werden. Andere Möglichkeiten wären die Nutzung der Debouncing Möglichkeiten des Komparators (siehe Atmel Datenblatt) oder auch den Wert des A/D Wandlers z.B. immer nur bei einem bestimmten Pegel des Ausgangssignals zu setzen und abzufragen.

11.8.2 Assemblerdatei mit Strommessung (mit MAX D/A Wandler)

Für die Access Benutzeroberfläche Elliptec.mdb wird folgendes Mikrokontroller Programm eingesetzt:

```
; test_elliptec piezo current with serial control
; settings with R1=R2=200 and RH0=255:98,745 kHz
; every step in R1 or R2 will result in time = time +
; step*1/11.0592 MHz (=90,42 ns)

SDA      .equ  92h             ; Port 1.2
SCL      .equ  93h             ; Port 1.3

#include LPx052.H
         .org 0000H
         sjmp start
         .org 000BH          ; TF0 Timer 0 overflow
         cpl P3.4            ; toggle Port pin
         jb  P3.4, pwm_on    ; next on or off ?
pwm_off: mov RL0, R2         ; set off time
         reti
pwm_on:  mov RL0, R1         ; set on time
         reti
start:   mov   P1M0,#03H     ; set ports to quasi bi-direc
         mov   P1M1,#00H     ; and P1.0 /P1.1 to input only
         mov   P3M0,#00H
         mov   P3M1,#10H     ; push pull output for P3.4
         mov   SP,#20H       ; Stack pointer
         clr   TR1           ; stop timer 0 / 1
         clr   TR0
         mov   TH1,#0DCH     ; 256-6: 9600 baud
         mov   TL1,#0DCH     ; 11.059MHz for SMOD1 =0
         anl   TMOD,#00H     ; Timer1: 8 bit auto-reload
         orl   TMOD,#20H
         setb  TR1           ; TCON start timer 1
         anl   PCON,#3FH     ; PCON: clear SM0D0 and SM0D0
         mov   SCON,#50H     ; InitRS232 8 bit UART Mode1
         setb  TI
         orl   PCON,#80H     ; SMOD=1 double Baudrate
                             ; Timer 0
         mov   RL0,#0BAH     ; set first RL and RH
```

```
         mov   RH0,#0FFH     ; default settings, maximum needed
!
         mov   R1,#0C8H      ; = 98,745 khz with 11.059Mhz Osz
         mov   R2,#0C8H      ; = 200
         orl   TMOD,#1       ; 16 bit autoreload
;        setb  TR0           ; start timer 0 by Software
         orl   ACSR,#08H     ; activate comparator inputs
         clr   P3.4          ; to reduce power consumption
         nop
         mov   IE,#82H       ; Interrups EA+ET0


; RL0:  R1 ON Time, R2 OFF Time;
; R3:  DA Value R4: DA settling time
; R5:  for I2Out

NEXT     mov   R3,#0         ; set DA value to 0 for next run
         acall I2Start
         mov   A,#62H        ; I2C adr
         acall I2Out         ; return A= 0 = error
         mov   A,R3          ; I2C data
         acall I2Out
         acall I2Stop
         acall RX
         mov   R1,A          ; R1 write ON time
         nop
         mov   A,R1          ; R1  read
         acall TX            ; confirm to user
         acall RX
         mov   R2,A          ; R2 write OFF time
         nop
         mov   A,R2          ; R2  read
         acall TX            ; confirm to user
         acall RX            ; start /stop by Host software
         anl   A,#1          ; 0 to stop, 1 to start
         jz    TimerOFF
         setb  TR0
         sjmp  TimerON
TimerOFF clr   TR0
         clr   P3.4          ; clear to reduce current
TimerON  acall TX            ; confirm to user

; wait at least some (70 us) for Piezo signal
         mov   R4,#255       ; MAX DA settling time
loopr    djnz  R4,loopr      ; R4 * 90ns * 3cyc

         mov   R4,#255       ; MAX DA settling time
looprn   djnz  R4,looprn     ; R4 * 90ns * 3cyc


; MAX 5382 MEUK : adr=62, SCL max 400 khz = 2,5 us
; settling time 20 us

nextval  acall I2Start
```

```
         mov   A,#62H        ; I2C adr
         acall I2Out         ; return A= 0 = error
         mov   A,R3          ; I2C data
         acall I2Out
         acall I2Stop
; ggf. Signalzustand abfragen:
;        jnb   P3.4,nextval  ; wait for good signal
         inc   R3            ; increase DA value until
         jnb   P3.6,sendval  ; poll comparator P3.6
         mov   A,R3          ; up to 255
         jnz   nextval       ; until comparator toggles
sendval  acall TX            ; send to user
         sjmp  NEXT

; ----------------------------
; ### I2C Progs

Delay    nop                 ; I2C command delay
         nop
         nop
         nop
         ret
I2Init   setb  SDA
         setb  SCL
         ret
I2Start  clr   SDA
         acall Delay
         clr   SCL
         ret
I2Stop   clr   SCL            ; ????
         clr   SDA
         acall Delay
         setb  SCL
         acall Delay
         setb  SDA
         ret
I2Out    mov   R5,#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  R5,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
; ### 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
        .end
```

11.8.3 Das D/A- A/D Wandlerverfahren verbessern

Bei der Strommessung nach der Zählmethode wird das analoge Ausgangssignal des Maxim D/A Wandlers solange erhöht und an den Komparator geliefert, bis der Wert größer als die analoge Eingangsspannung ist. Die Umsetzungszeit ist abhängig von der Eingangsgröße, im ungünstigsten Fall muss der Zähler alle Stufen durchlaufen, da der Zähler im Assemblerprogramm des Atmel Mikrocontrollers jedes Mal bei 0 startet.
Die Zahl der möglichen Wandervorgänge kann jedoch durch vor- und rückwärts zählen erheblich reduziert werden. Gleichzeitig ergibt sich in Zusammenhang mit dem Elliptecmotor eine einfachere Möglichkeit der Frequenzsuche.
Der Komparator, der die Spannung am Shuntwiderstand des Komparators mit der Ausgangsspannung des Maxim DA Wandlers vergleicht, arbeitet dynamisch. Die Stromaufnahme des Elliptecmotors ändert sich nach einem Frequenzschritt und damit ändert sich auch automatisch der Zustand des Komparators oder auch nicht, je nachdem ob sich die Spannung beim nächsten Frequenzschritt reduziert oder erhöht hat. Dann muss vom vorherigen Zählerzustand ausgehend weiter aufwärts oder abwärts gezählt werden, bis der Komparator seinen Ausgangszustand wieder ändert. Ist der Ausgangszustand nach einem Frequenzschritt gleich geblieben, dann hat sich die

Eingangsspannung reduziert und es muss rückwärts gezählt werden. Hat sich der Zustand geändert, dann ist die Eingangsspannung, und damit die Stromaufnahme des Elliptecmotors, weiter gestiegen und es muss weiter vorwärts gezählt werden.

Die größte Flankensteilheit der Spannung ergibt sich dann aus der größten Zählerdifferenz vom aktuellen zum vorherigen Schritt.

Für dieses Verfahren muss der Ausgangszustand des Komparators und der aktuelle Zählerstand des Maxim D/A Wandlers zu jedem Frequenzschritt für den nachfolgenden Schritt zwischengespeichert werden.

11.9 Die optimale Betriebsfrequenz

Die optimale Betriebsfrequenz lässt sich anhand des Stromverlaufs ermitteln. Es gibt von Elliptec AG entsprechende Applikationshinweise mit dem Tiny26, die sich auch hier anwenden lassen.
Die optimalen Frequenzen liegen in der Nähe der größten Stromänderungen und der geringsten Stromaufnahme. In der obigen Darstellung in etwa bei 78 KHz für die Vorwärtsbewegung und 97 KHz für die Rückwärtsbewegung.
Für eine Vorwärtssuche zwischen 75 und 82 KHz ist nach der Suche des maximalen Stromwertes der minimale Stromwert zu suchen. Eine Rückwärtssuche ab ca. 130 KHz bis 85 KHz kann ähnlich verwendet werden. In meinen Testfällen ergab sich die optimale Rückwärtsfrequenz dann fast immer in der Nähe der niedrigsten Stromaufnahme und der größten Flankensteilheit des Stromverlaufs.

Nun lassen sich nicht alle Betriebsfrequenzen mit einem Mikrocontroller so genau wie vielleicht gewünscht erzeugen. In diesem Fall kann man den Motor pulsierend betrieben werden, z.B. 5 ms mit etwas geringerer Frequenz, 5 ms keine Ansteuerung, 5 ms mit etwas höherer Frequenz. Durch das Pulsieren kann auch die Motorgeschwindigkeit verändert werden. In diesem Fall kann es aber auch, je nach Anwendung, zu hörbaren Geräuschentwicklungen kommen.
Da die kurze Zeit von 5 ms nicht über die serielle Schnittstelle gesteuert werden kann, ist eine Implementierung in die Mikrokontrollersoftware notwendig. Je nach Anwendung und welche Timer- und Interruptmöglichkeiten im Mikrocontroller verwendet werden, ist eine unterschiedliche Umsetzung möglich. 5 ms sind bei 11 MHz Betriebsfrequenz ca. 55.000 Takte, eine Ewigkeit für den Atmel AT89LPx052.

Die optimale Betriebsfrequenz ist zugleich von der Motortemperatur abhängig. Mit steigender Temperatur verringern sich die Betriebsfrequenzen (bis zu ca. 2khz) Im Dauerbetrieb wird, bei gleicher Ansteuerungsfrequenz, der Motor also nach kurzer Zeit stehen bleiben.

11.10 Mechanische Feedbacksysteme

Ein Encoder oder ein Linearmaßstab kann als Feedbacksystem eingesetzt werden. Über ein Feedbacksystem lässt sich die

Geschwindigkeit, die relative oder absolute Position und die ideale Ansteuerfrequenz generieren.
Mögliche Encoder wären zum Beispiel von PWB Ruhlatec oder Agilent, die dann in der Hardware einen Interrupt im Mikrocontroller auslösen könnten. Auf dem Ghost Board ist dazu der Anschluss Conn_OE vorgesehen.

11.11 Elektronische Geschwindigkeitsreglung

[Elliptec] Die Maximalgeschwindigkeit eines Elliptecmotors beträgt typischerweise über 300mm/s. Um die Geschwindigkeit per Software zu reduzierten, gibt es zwei elektronische Methoden:

1. Geschwindigkeitsreduktion durch Änderung der Pulsbreite
Zur Ansteuerung des Elliptecmotors gibt der Mikrocontroller ein Rechtecksignal aus. Den maximalen Schub erhält man, wenn Einschaltzeit und Ausschaltzeit gleich sind, das entspricht einem Verhältnis von 1:1. Ändert man dieses Verhältnis, so reduziert sich die vom Motor aufgenommene Leistung und somit die Geschwindigkeit. Nachteilig bei dieser Methode ist, dass auch die Schubkraft verringert wird. Bei kleinen Geschwindigkeiten besteht die Gefahr, dass der Motor stehen bleibt.

2. Geschwindigkeitsreduktion durch Ansteuerung in kleinen Schritten
Dabei wird der Motor für eine sehr kurze Zeit mit seiner Betriebsfrequenz (Tastverhältnis 1:1) angesteuert und erreicht in dieser Einschaltperiode vollen Schub und volle Geschwindigkeit. Anschließend folgt eine kleine Pause, in der der Motor nicht angesteuert wird und das angetriebene Element nicht bewegt. Das angetriebene Element wird somit in kleinen Schritten bewegt. Da diese Schritte sehr klein und sehr häufig sind, nimmt das Auge die Bewegung als kontinuierlich wahr. Die Längen der Ein- und Ausschaltperiode bestimmen die entstehende Geschwindigkeit. Auch die Schrittgröße und –frequenz hängen von diesen beiden Zeiten ab. Vorteil dieser Methode ist, dass die Schubkraft konstant bleibt.

Beide Methoden lassen sich kombinieren und somit Kraft, Geschwindigkeit und sogar Geräusche optimieren.