7.0 PWM - Pulsbreitenmodulation

PWM ist eine Abkürzung für Pulse Width Modulation (Pulsbreitenmodulation). Zwei unabhängige PWM-Ausgänge erlauben die Erzeugung von Rechtecksignalen mit einstellbarer Frequenz und einstellbarem Puls/Pausenverhältnis.

7.1 PWM Versuche in der Praxis

Für die Ausgabe der PWM Signale verwendet der Atmel AT89LPx052 Mikrocontroller die Pins P3.4 (Timer 0 Input) und P3.5 (Timer 1), die entsprechend den Vorgaben in den Special Function Registern eingerichtet werden müssen. PWM wird dabei nur in den Timer 0 und 1 Modi unterstützt.

Die Pulsbreite wird durch einen Vergleich der Register THO, TH1 und RH0 bzw. RH1 Register verändert. Damit verbleiben TL0 bzw. TL1 als 'Prescaler' für einen Frequenzteiler. In Mode 0 wird mit TL0/1 eine logarithmische Teilung erreicht, wobei jeweils 3 PSC Bits im TCON Register den Faktor bestimmen.
In Mode 1 erfolgt eine lineare Teilung, wobei TL0/1 automatisch aus RL0/1 neu geladen wird.

7.2 Ein Assembler Programm für PWM Mode 1

Das folgende Beispiel zeigt Ihnen die Programmierung der Pulsweitenmodulation in Mode 1 und erlaubt die Steuerung durch den Entwicklungsrechner mit einem einfachen VB Programm über die serielle Schnittstelle.

```
; TASM
; Serial port access to modify PWM RL0
; 11,059MHz, 19200 Baud (or change SMOD1)
#include LPx052.H
      .org 0000H

INIT  mov   P1M0,#00H       ; set port
      mov   P1M1,#00H
      mov   P3M0,#00H
      mov   P3M1,#00H
      clr   TR1             ;stop Timer 0/1
      clr   TR0
```

```
       mov   TH1,#0DCH       ;256-6: 9600 baud
       mov   TL1,#0DCH       ;for SMOD
       anl   TMOD,#00H       ;auto-reload
       orl   TMOD,#20H
       setb  TR1             ;TCON start timer
       anl   PCON,#3FH       ;clr SM0D0
       mov   SCON,#50H       ;InitRS232
       setb  TI
       orl   PCON,#80H       ;double Baudrate
       orl   TMOD,#01H       ;Timer1, M0 Mode 1
       mov   TL0,#55H        ;T0 Prescaler
       mov   RL0,#55H        ;reload TL0
       orl   TCONB,#40H      ;PWM enable P3.5
       mov   A,#55H
       mov   TH0,#55H        ;T0 counter value
       mov   RH0,#0AAH       ;Pulse width
       setb  TR0             ;start timer

NEXT   acall RX
       mov   RL0,A           ;RL0 write
       nop
       mov   A,RL0           ;RL0  read
       acall TX              ;confirm to user
       acall RX
       mov   RH0,A           ;RH0 write
       nop
       mov   A,RH0           ;RH0  read
       acall TX              ;confirm to user
       sjmp  NEXT

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

TX     jnb   TI,TX           ;commit value
       clr   TI
       mov   SBUF,A
       ret
      .end
```

Der Timer 1 wird für die serielle Datenübermittlung und Timer 0 in Mode 1 für die Ausgabe an Port 3.5 eingesetzt.

Zu Anfang werden Ports 1 und 3 in den bidirektionalen Mode gesetzt. Die serielle Übertragung wird auf 9600 Baud eingerichtet und durch SMOD1 im PCON Register auf 19.2k verdoppelt. Beachten Sie die Werte in TH1 und TL1 im Vergleich zu den AT89C2051 / AT89C4051 Vorgängern, da der Timer nicht mehr durch 12 geteilt wird.

Die Ausgabe an P3.5 wird durch Setzen des PWM1EN Bits im PCONB Registers erreicht.

Die Daten für RL0 (Teilfaktor) und RH0 (Pulsbreite) werden durch ein VB Programm vom Entwicklungsrechner seriell übermittelt und danach wiederum vom Mikrokontroller dem Programm bestätigt.

7.3 PWM mit Visual Basic steuern

Eine einfache VB Oberfläche übermittelt zyklisch die eingestellten Werte zum Mikrokontroller:

Die VB Programmierung ist simple:

```
Private Sub Form_Load()
  OPENCOM "COM1:19200,N,8,1"
End Sub

Private Sub Form_Unload(Cancel As Integer)
  CLOSECOM
End Sub

Private Sub Timer1_Timer()
  SENDBYTE (HScroll1.Value) ' RL0
  Text1.Text = "RL0=" & READBYTE
  SENDBYTE (HScroll2.Value) ' RH0
  Text2.Text = "RH0=" & READBYTE
End Sub
```

Also einfach den Mikrokontroller mit dem Hex File programmieren, das VB Programm starten und mit einem Scope oder Frequenzmessgerät testen.

Die maximale Ausgangsfrequenz an P3.5 ist Oszillatorfrequenz / 256 und entsprechen ca. 43 khz bei 11.059 Mhz Oszillatorfrequenz. Sie

werden z.T. ein etwas zitterndes Signal feststellen, da die zyklische, serielle Übermittlung die Register zu undefinierten Zeitpunkten neu programmiert.

(Die erforderlichen Dll's für die unterschiedlichen Betriebssysteme finden Sie entweder bei Elexs www.elexs.de oder bei HJ Bernd http://www.hjberndt.de).

7.4 16 Bit Timer als Alternative zu PWM

PWM Frequenzen 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 Cylcle 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 *(ein guter Wert für die serielle Kommunikation)*. 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 reicht sollte für viele Applikationen noch ausreichend sein.
Bei 11.069 MHz 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.

Und hier die Assembler Datei:

```
; TASM
; test_timer_pwm with serial control
; bei 11.059 MHz Quartz ca. 247 kHz max

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

start:   mov   P1M0,#01H     ; set ports
         mov   P1M1,#00H     ; P1.0 to input
         mov   P3M0,#00H
```

```
        mov   P3M1,#10H   ; push pull P3.4
        mov   SP,#20H     ; Stack pointer
        clr   TR1         ; stop T07T1
        clr   TR0
        mov   TH1,#0DCH   ; 256-6: 9600 baud
        mov   TL1,#0DCH   ;
        anl   TMOD,#00H   ; auto-reload
        orl   TMOD,#20H
        setb  TR1         ; start timer 1
        anl   PCON,#3FH   ; SM0D0 and SM0D0
        mov   SCON,#50H   ; InitRS232
        setb  TI
        orl   PCON,#80H   ; double Baudrate
        mov   RL0,#0BAH   ; set RL and RH
        mov   RH0,#0FFH   ; default settings
        mov   R1,#0BAH    ; 80khz
        mov   R2,#0BAH
        orl   TMOD,#1     ; autoreload
        setb  TR0         ; start T0
        mov   IE,#82H     ; EA+ET0
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
        mov   RH0,A       ; RH0 pre scaler
        nop
        mov   A,RH0       ; RH0 read
        acall TX          ; confirm to user
        sjmp  NEXT

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

TX      jnb   TI,TX       ; commit value
        clr   TI
        mov   SBUF,A
        ret
       .end
```

Der Timer 1 wird für die serielle Datenübermittlung und der 16bit Timer 0 für die Interrupt Steuerung verwendet.
Die Ein- und Ausschaltzeiten werden in R1 bzw. R2 gespeichert und bei jedem Interrupt entsprechend dem Zustand neu gesetzt.

Mit R1 / R2 wird daher nicht nur die Pulsbreite sondern auch die Frequenz verändert. Anstelle R1 und R2 könnte man auch das interne RAM verwenden.
Im Gegensatz zum vorherigen VB Beispiel muss zur Steuerung über die serielle Schnittstelle ein Byte mehr übermittelt werden.

Ein Auszug aus der einfachen VB Software. Die drei Bytes für ON-OFF Time und dem Teiler können mit drei Schiebereglern eingestellt und übermittelt werden.
Die Signalausgabe kann an P3.4 überprüft werden.