

PWM, a Software Solution for the PIC16CXXX

*Author: Ole Röpcke
Consultant, Europe*

INTRODUCTION

The low cost, high performance features of a PIC16CXXX microcontroller make it a suitable device for automatic control technology applications. Sometimes, an additional PWM output is needed. For some devices, such as the PIC16C71, the addition of a software PWM adds the missing element. It is possible to use Timer0 (which also provides the system clock) and its corresponding interrupt to generate a PWM output with a duty cycle that can vary from nearly 10% to 90%. However, some applications require a greater duty cycle range.

This application note provides a software solution for a more accurate and flexible PWM output, which is characterized by the following:

1. PWM frequency up to 19.6 kHz (20 MHz crystal).
2. Variable duty cycle from 0% to 100%.
3. 8-bit resolution.
4. PWM step size of 1 Tcy (Instruction Cycle Time).
5. Simultaneous generation of a system time clock.

EXAMPLE 1: USING Tcy AS A TIME BASE FOR PULSE GENERATION (METHOD A)

```

.
.
.

PORTA    equ 04h
LENGTH   equ 0ch
COUNTER  equ 0dh

.

.

.

movlw    34          ; value for pulse length of 99 Tcy
movwf    LENGTH       ;
.

.

.

movf    LENGTH,W      ; write value to the loop counter
movwf    COUNTER      ;
bsf     PORTA,3        ; start high pulse

Loop
decfsz  COUNTER,F    ; counting pulse length
goto    Loop           ;
bcf     PORTA,3        ; end high pulse
.
.
.
```

METHODS

Before the solution is revealed, we must first examine the various software methods used to generate variable length pulses.

In the following explanations, the unit of time will be the length of an Instruction Cycle (Tcy). We will use Tcy because one instruction (if the program counter is not changed) executes in one Tcy and Timer0 (without prescaling) increments every Tcy. This provides us with a simple method to control a potentially complex timing sequence.

Making use of the time needed to execute an instruction provides a very simple method of generating a pulse. Example 1 (Method A) shows an instruction sequence, which will generate a high pulse of 99 Tcy on pin 3 of PORTA. The pulse length is controlled by the value of register LENGTH in steps of 3 Tcy. This is the computing time needed by one program loop.

The drawbacks of this method are an excessive use of computing time and a poor PWM resolution.

However, the architectural features of Microchip's midrange microcontrollers allow us to proceed in another direction. Example 2 shows an instruction sequence (Method B), which enables us to generate a high pulse with lengths varying from 1 to 5 Tcy. The addition of any number to the file register PCL increases the program counter and skips a predetermined number of instructions (depending on the number added to PCL). The length of the high pulse is the same as the computing time consumed by the number of executed BSF instructions and is controlled by the value of file register LENGTH. If LENGTH is set to 4, 4 BSF instructions will be skipped and a high pulse of 1 Tcy will be generated. If LENGTH is set to 0, no instructions will be skipped and the length of the pulse

remains 5 Tcy. A special effect takes place, when LENGTH is set to 5. All BSF instructions are skipped and therefore no high pulse occurs. This instruction sequence is able to generate pulses of various lengths including the length 0 Tcy. Between each length the step size is only 1 Tcy. This method will generate a long pulse if LENGTH is set to a small value, and a short pulse if LENGTH is set to a large value.

The drawback of this method is that it is suitable only for short pulses. Long pulses require an excessive amount of ROM and computing time.

EXAMPLE 2: PULSE GENERATION OPTIMIZED TO DEVICE ARCHITECTURE (METHOD B)

```
.  
PORTA    equ 04h  
LENGTH   equ 0ch  
. . .  
movlw    4          ; value for pulse length of 1 Tcy  
movwf    LENGTH     ;  
. . .  
movf    LENGTH,W    ; This is an indirectly addressed relative jump  
addwf    PCL,F      ;  
bsf     PORTA,3      ; start high pulse 5 Tcy  
bsf     PORTA,3      ; start high pulse 4 Tcy  
bsf     PORTA,3      ; start high pulse 3 Tcy  
bsf     PORTA,3      ; start high pulse 2 Tcy  
bsf     PORTA,3      ; start high pulse 1 Tcy  
bcf     PORTA,3      ; end high pulse  
. . .
```

A third method (Method C) uses the Timer0 Interrupt. After the port is set, the timer is loaded with an adjusted number corresponding to the desired pulse length. If the timer overflows, the interrupt service routine starts and resets the port. The number, which has to be loaded, is defined by the following factors:

- the counting direction of the timer (up/down)
- the time between loading the timer and setting the port
- the timer clock cycles needed to write to the timer
- the computing time between timer overflow and executing the first instruction of the interrupt service routine
- the computing time of the interrupt service routine until the port is reset
- the desired pulse length
- an additional correction number, which is related to the method of calculation

Method C is able to generate very different pulse lengths without wasting any computing time. This method is specifically useful to generate long pulses. When the prescaler is not used, the available step size is 1 TCY. But this procedure is unsuitable for generating very short pulses, because every interrupt service routine needs a minimum of computing time for execution, which defines the minimum pulse length.

Every PWM signal is a continuous succession of high and low pulses. The length of each pulse is defined by the desired duty cycle. Adding the length of a low and a high pulse gives us the PWM frequency, which should be kept constant.

Method C will only work if each of the pulses are longer than the minimum computing time of the interrupt service routine. The interrupt service routine has to find the required pulse and set or reset the port. In addition to that the interrupt service routine has to calculate the number required by the timer and write that value to the timer.

Whether it is feasible to use the timer interrupt depends on the desired pulse length and the known minimum computing time of the interrupt service routine. If Method C will not give the desired PWM signal, one has to fall back on methods A or B as described in Example 1 and Example 2.

Each method has its advantages and disadvantages. Because all three methods are software based, it is possible to change methods at any time. Therefore the microcontroller can determine, even during the interrupt service routine, which method is best suited to generate the next pulse. This is the underlying concept for this application.

The software PWM module uses methods B and C, which are shown in Table 1. If possible, both pulses are generated by the timer interrupt and two timer interrupts occur during one PWM period. If a very short pulse is needed, the interrupt service routine will generate this short pulse using method B. Afterwards it

writes a new number corresponding to the next long pulse to the timer. Here, only one interrupt occurs during one PWM period. Knowing the computing time of each part of the interrupt service routine ensures that a change in methods will not visibly affect the PWM signal or vary the PWM frequency.

Now, let us have a look at a simple application of this PWM software module. The circuit diagram is shown in Figure 1.

Potentiometer R1 adjusts voltage from 0V to VDD. A PIC16C71 converts this voltage to an 8-bit value thirty times per second and sends it to PORTB. The eight LEDs connected to PORTB show the result. The PWM signal value, which is output on pin RA3, is shown by the intensity of the connected LED. If the value is equal to 0, a continuous low signal is generated. If the value is equal to 255 a continuous high signal is generated.

The method of operation of the program is the same as the described procedure, which is shown by Table 1.

The file register COUNTER is the file register of the system clock. It is incremented by the PWM module at the rising edge of the PWM signal. Bit 7 is toggled each $255 \cdot 128 \cdot \text{TCY}$ and is used as the system clock bit.

The file register PWMDESIRED contains the desired PWM value in the range of 0 to 255. Register PWMMAX and PWMHELP are needed by the PWM module. They must not be modified by any other part of the program except during initialization.

The constant PWMADJUSTVAL contains the previously described factors, which have to be taken into account while calculating the adjusted timer value. The constant PWMMAXVAL controls the maximum pulse length, which is generated by making use of the time needed to execute BSF/BCF instructions. If the interrupt service routine is modified, then both constants will also have to be modified.

The comments in the code explain each instruction sequence.

CONCLUSION

This software based PWM module is able to generate a PWM frequency of up to 19.6 kHz with a variable duty cycle of 0% to 100%. The consumption of RAM is 3 bytes, the consumption of ROM less than 10% (nearly 100 instructions) and the consumption of computing time, in the worst case, amounts to $57/266 = 22.4\%$. The remaining computing time is more than the total available computing time of a 8051 microcontroller with a 12 MHz crystal.

FIGURE 1:

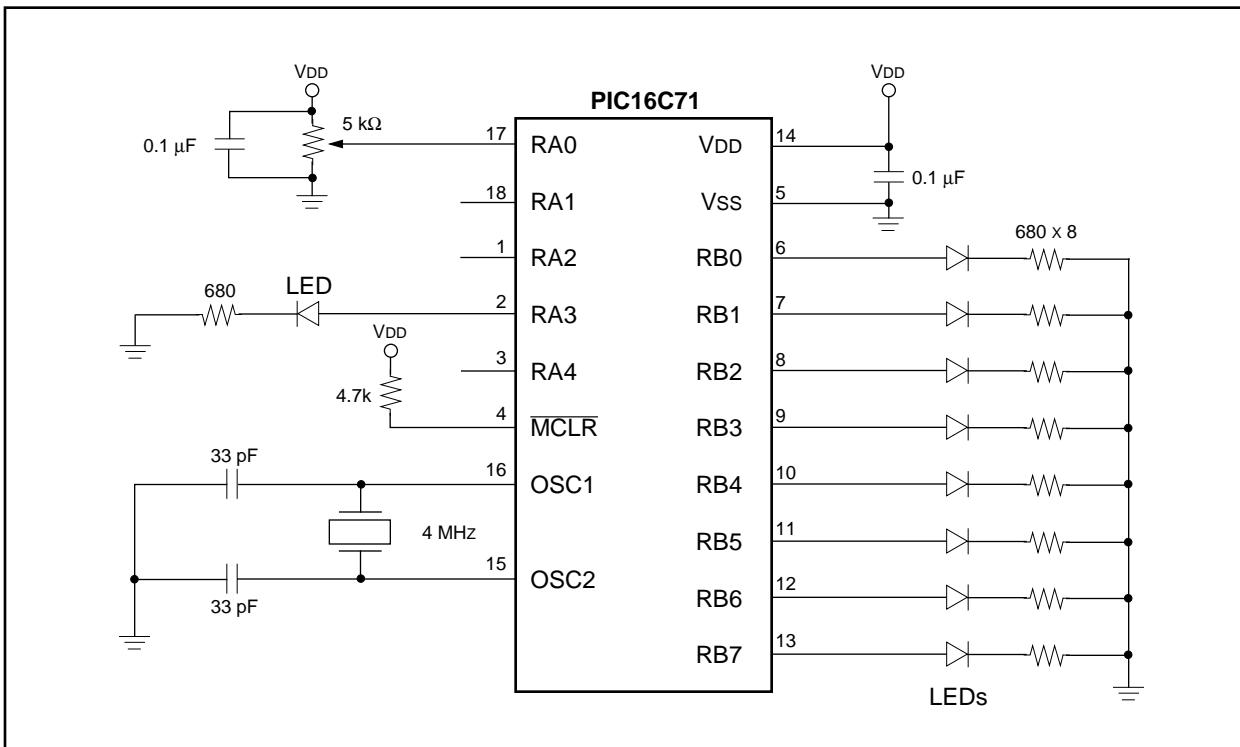


TABLE 1: OPERATION MODES OF THE SOFTWARE PWM MODULE

Duty Cycle Range	High Pulse	Low Pulse
0% .. 10%	Skipping (Method B)	Timer Interrupt
10% .. 90%	Timer Interrupt	Timer Interrupt
90% .. 100%	Timer Interrupt	Skipping (Method B)

APPENDIX A: PWM.ASM

MPASM 01.40 Released

PWM.ASM 10-2-1996 10:29:19

PAGE 1

LOC	OBJECT CODE VALUE	LINE SOURCE TEXT
		00001 ;*****
		00002 ; Filename: PWM16CXX.ASM
		00003 ;*****
		00004 ; Author: Ole Ropcke
		00005 ; Company: private
		00006 ; Revision: RevA1
		00007 ; Date: 01-4-96
		00008 ; Assembled using MPASM rev 01.21
		00009 ;*****
		00010 ; Include files:
		00011 ; p16c71.inc
		00012 ;*****
		00013 ; Required hardware:
		00014 ;*****
		00015 ; What's changed:
		00016 ; Date Description of Change
		00017 ; xx-xx-xx
		00018 ;*****
		00019 00020 LIST P=16C71, R=DEC
		00021
		00022 include P16C71.INC
		00001 LIST
		00002 ; P16C71.INC Std Hdr File, Version 1.00 Microchip Technology, Inc.
		00142 LIST
		00023
		00024 ;*****
		00025 ; Definitions
		00026 ;*****
		00027 ;
		00028 ; I/O, Interrupt and Option Definitions
		00029 ;
00000088		00030 OPTIONVAL equ 10001000b ; portB no pull-up, tmr0 int.
000000A0		00031 INTCONVAL equ 0a0h ; set GIE, TOIE
		00032 ; port A:
00000000		00033 UINPBIT equ 00h ; analog input for desired PWM
00000003		00034 PWMOUTBIT equ 03h ; PWM output
00000017		00035 TRISAVAL equ 00010111b ; A3 output
00000002		00036 ADCON1VAL equ 2 ; A0,A1 analog; A2,A3 digital
00000089		00037 ADCON0VAL equ 10001001b ; fosc/32, channel 0
		00038 ; port B:
00000000		00039 TRISBVAL equ 0 ; LED outputs
		00040 -----
		00041 ; Register Definitions
		00042 ;
0000000C		00043 STACKW equ 0ch ; stack to push/pop the W-register
0000000D		00044 STACKS equ 0dh ; stack to push/pop the STATUS-reg
0000000E		00045 COUNTER equ 0eh ; counter: input frequency
		00046 ; f1 = crystalfreq. / 4 / 255
0000000F		00047 COUNTER2 equ 0fh ; counter2: input frequency
		00048 ; f2 = f1 / 128
00000010		00049 PWMDESIRED equ 10h ; desired PWM value 0..255
00000011		00050 PWMMAX equ 11h ; register to support generating PWM
		00051 ; You have to put PWMMAXVAL into it!
00000012		00052 PWMHELP equ 12h ; register to support generating PWM
		00053 ; used as temp storage of PWMDESIRED
		00054 -----
		00055 ; PWM-module-constant
00000016		00056 PWMADJUSTVAL equ .22

```

00057      ; correction number, defined by the following factors:
00058      ; time from timer interrupt to executing PC 004 + 3 cycles
00059      ; computing time from PC=004 to required edge    +18 cycles
00060      ; lost timer cycles due to writing the timer    + 2 cycles
00061      ; cal desired PWM value to timer loading value + 2 cycles
00062      ; time from timer loading to gen required edge - 1 cycle
00063      ; valid value for hardware (unknown diff to the data sheet)
00064      ; 3+18+0+2-1=22
00065      ; valid value for PICSIM version 5.11 (error of PICSIM):
00066      ; 0+18+2+2-1=21
0000001D 00067 PWMMAXVAL    equ     .29
00068      ; loading value for PWMMAX
00069      ; If n is the maximum length of a high pulse, which has to be
00070      ; generated by the skipping method, then is PWMMAXVAL = n+1.
00071      ; The max length of a low pulse using the skip method is n-1.
00072 =====
0000 00073 ORG 0
0000 2870 00074 goto PowerOn
00075 ;--- PWM-Generator -----
0004 00076 ORG 04h          ; timer interrupt
0004 1801 00077 btfsc TMRO,0 ; compensate comp time of 1/2 cyc
0005 2806 00078 goto PwmInt ; instruc when timer int occured
0006 00079 PwmInt
0006 008C 00080 movwf STACKW ; copy W register... to "stack"
0007 0E8C 00081 swapf STACKW,F ; ...to "stack"
0008 0E03 00082 swapf STATUS,W ; copy STATUS register...
0009 008D 00083 movwf STACKS ; ...to "stack"
000A 110B 00084 bcf INTCON,T0IF ; clear interrupt flag
000B 1985 00085 btfsc PORTA,PWMOUTBIT ; which edge is required?
000C 2840 00086 goto Lowpulse ; -> goto falling edge
000D 00087 Highpulse
000D 0910 00088 comf PWMDESIRED,W ; get desired PWM value
000E 0092 00089 movwf PWMHELP ; store val for the foll low pulse
000F 0791 00090 addwf PWMMAX,F ; calc number of inst's to skip
0010 1C03 00091 btfss STATUS,C ; which method to use?
0011 2836 00092 goto HighImpInt ; -> using interrupt
0012 00093 HighImpShrt
0012 0811 00094 movf PWMMAX,W ; get number of inst's to skip
0013 0782 00095 addwf PCL,F ; skip n instructions
0014 1585 00096 bsf PORTA,PWMOUTBIT ; rising edge, 28 cycles hi pulse
0015 1585 00097 bsf PORTA,PWMOUTBIT ; 27 cycles
0016 1585 00098 bsf PORTA,PWMOUTBIT ; 26 cycles
0017 1585 00099 bsf PORTA,PWMOUTBIT ; 25 cycles
0018 1585 00100 bsf PORTA,PWMOUTBIT ; 24 cycles
0019 1585 00101 bsf PORTA,PWMOUTBIT ; 23 cycles
001A 1585 00102 bsf PORTA,PWMOUTBIT ; 22 cycles
001B 1585 00103 bsf PORTA,PWMOUTBIT ; 21 cycles
001C 1585 00104 bsf PORTA,PWMOUTBIT ; 20 cycles
001D 1585 00105 bsf PORTA,PWMOUTBIT ; 19 cycles
001E 1585 00106 bsf PORTA,PWMOUTBIT ; 18 cycles
001F 1585 00107 bsf PORTA,PWMOUTBIT ; 17 cycles
0020 1585 00108 bsf PORTA,PWMOUTBIT ; 16 cycles
0021 1585 00109 bsf PORTA,PWMOUTBIT ; 15 cycles
0022 1585 00110 bsf PORTA,PWMOUTBIT ; 14 cycles
0023 1585 00111 bsf PORTA,PWMOUTBIT ; 13 cycles
0024 1585 00112 bsf PORTA,PWMOUTBIT ; 12 cycles
0025 1585 00113 bsf PORTA,PWMOUTBIT ; 11 cycles
0026 1585 00114 bsf PORTA,PWMOUTBIT ; 10 cycles
0027 1585 00115 bsf PORTA,PWMOUTBIT ; 9 cycles
0028 1585 00116 bsf PORTA,PWMOUTBIT ; 8 cycles
0029 1585 00117 bsf PORTA,PWMOUTBIT ; 7 cycles
002A 1585 00118 bsf PORTA,PWMOUTBIT ; 6 cycles
002B 1585 00119 bsf PORTA,PWMOUTBIT ; 5 cycles
002C 1585 00120 bsf PORTA,PWMOUTBIT ; 4 cycles
002D 1585 00121 bsf PORTA,PWMOUTBIT ; 3 cycles
002E 1585 00122 bsf PORTA,PWMOUTBIT ; 2 cycles

```

```

002F 1585      00123    bsf     PORTA,PWMOUTBIT ; 1 cycle
0030 1185      00124    bcf     PORTA,PWMOUTBIT ; fall edge;start of the following
                           00125    ; low pulse using the interrupt
0031 0A8E      00126    incf    COUNTER,F   ; trigger COUNTER, cause there was
                           00127    ; a rising edge
0032 0912      00128    comf    PWMHELP,W   ; get required low pulse length
0033 3E1B      00129    addlw   PWMADJUSTVAL+5 ; calculate timer loading value
                           00130    ; Edge was generated 5 cycles before
                           00131    ; usual point of time.
0034 0081      00132    movwf   TMRO      ; put value into timer
0035 2869      00133    goto    LowImpInt2 ; low pulse using int is running
0036           00134    HighImpInt
0036 3E16      00135    addlw   PWMADJUSTVAL ; calculate timer loading value
0037 0081      00136    movwf   TMRO      ; put value into timer
0038           00137    HighImpInt2
0038 1585      00138    bsf     PORTA,PWMOUTBIT ; generate rising edge
0039 0A8E      00139    incf    COUNTER,F   ; trigger counter, because there
                           00140    ; was a rising edge
003A 301C      00141    movlw   PWMMAXVAL-1 ; "repair"...
003B 0091      00142    movwf   PWMMAX      ; ...support register
003C 0E0D      00143    swapf   STACKS,W   ; restore...
003D 0083      00144    movwf   STATUS      ; ...STATUS register
003E 0E0C      00145    swapf   STACKW,W   ; restore W register
003F 0009      00146    retfie
                           00147    Lowpulse
0040 0912      00148    comf    PWMHELP,W   ; get required pulse length
0041 0791      00149    addwf   PWMMAX,F   ; calc number of inst's to skip
0042 1C03      00150    btfss   STATUS,C   ; which method is to use?
0043 2867      00151    goto    LowImpInt ; ->using interrupt
0044           00152    LowImpShrt
0044 0811      00153    movf    PWMMAX,W   ; get number of inst's to skip
0045 0782      00154    addwf   PCL,F      ; skip n instructions
0046 1185      00155    bcf     PORTA,PWMOUTBIT ; falling edge, 27 cycles low pulse
0047 1185      00156    bcf     PORTA,PWMOUTBIT ; 26 cycles
0048 1185      00157    bcf     PORTA,PWMOUTBIT ; 25 cycles
0049 1185      00158    bcf     PORTA,PWMOUTBIT ; 24 cycles
004A 1185      00159    bcf     PORTA,PWMOUTBIT ; 23 cycles
004B 1185      00160    bcf     PORTA,PWMOUTBIT ; 22 cycles
004C 1185      00161    bcf     PORTA,PWMOUTBIT ; 21 cycles
004D 1185      00162    bcf     PORTA,PWMOUTBIT ; 20 cycles
004E 1185      00163    bcf     PORTA,PWMOUTBIT ; 19 cycles
004F 1185      00164    bcf     PORTA,PWMOUTBIT ; 18 cycles
0050 1185      00165    bcf     PORTA,PWMOUTBIT ; 17 cycles
0051 1185      00166    bcf     PORTA,PWMOUTBIT ; 16 cycles
0052 1185      00167    bcf     PORTA,PWMOUTBIT ; 15 cycles
0053 1185      00168    bcf     PORTA,PWMOUTBIT ; 14 cycles
0054 1185      00169    bcf     PORTA,PWMOUTBIT ; 13 cycles
0055 1185      00170    bcf     PORTA,PWMOUTBIT ; 12 cycles
0056 1185      00171    bcf     PORTA,PWMOUTBIT ; 11 cycles
0057 1185      00172    bcf     PORTA,PWMOUTBIT ; 10 cycles
0058 1185      00173    bcf     PORTA,PWMOUTBIT ; 9 cycles
0059 1185      00174    bcf     PORTA,PWMOUTBIT ; 8 cycles
005A 1185      00175    bcf     PORTA,PWMOUTBIT ; 7 cycles
005B 1185      00176    bcf     PORTA,PWMOUTBIT ; 6 cycles
005C 1185      00177    bcf     PORTA,PWMOUTBIT ; 5 cycles
005D 1185      00178    bcf     PORTA,PWMOUTBIT ; 4 cycles
005E 1185      00179    bcf     PORTA,PWMOUTBIT ; 3 cycles
005F 1185      00180    bcf     PORTA,PWMOUTBIT ; 2 cycles
0060 1185      00181    bcf     PORTA,PWMOUTBIT ; 1 cycle
0061 1585      00182    bsf     PORTA,PWMOUTBIT ; rising edge; start of the next
                           00183    ; high pulse using the interrupt
0062 0910      00184    comf    PWMDESIRED,W ; get desired PWM value
0063 0092      00185    movwf   PWMHELP    ; store val for the next lo pulse
0064 3E1B      00186    addlw   PWMADJUSTVAL+5 ; calculate timer loading value
                           00187    ; Edge was gen'd 5 cycles before
                           00188    ; usual point of time.

```

```

0065 0081      00189     movwf   TMRO      ; put value into timer
0066 2838      00190     goto    HighImpInt2 ; high pulse using int is running
0067          00191 LowImpInt
0067 3E16      00192     addlw   PWMADJUSTVAL ; low pulse using interrupt
0068 0081      00193     movwf   TMRO      ; calculate timer loading value
0069          00194 LowImpInt2
0069 1185      00195     bcf    PORTA,PWMOUTBIT ; generate falling edge
006A 301D      00196     movlw   PWMMAXVAL ; "repair" ...
006B 0091      00197     movwf   PWMMAX    ; ...support register
006C 0E0D      00198     swapf  STACKS,W   ; restore ...
006D 0083      00199     movwf   STATUS    ; ...STATUS register
006E 0E0C      00200     swapf  STACKW,W   ; restore W register
006F 0009      00201     retfie  ; return to main program
00202 ;--- power on routine -----
0070          00203 PowerOn
00204     ; configuration of the PWM module
0070 0181      00205     clrf    TMRO      ; reset timer
0071 0190      00206     clrf    PWMDESIRED ; reset value of PWM is 0
0072 1185      00207     bcf    PORTA,PWMOUTBIT ; reset PWM-port before port A is
00208           ; changed from input to output to
00209           ; suppress an uncontrolled spike
0073 301D      00210     movlw   PWMMAXVAL ; set support register
0074 0091      00211     movwf   PWMMAX    ;
00212     ; configuration of the Pic
0075 1683      00213     bsf    STATUS,RP0  ; register page 1
0076 3017      00214     movlw   TRISAVAL  ; configurate ...
0077 0085      00215     movwf   TRISA     ; ...port A
0078 3000      00216     movlw   TRISBVAL  ; configurate ...
0079 0086      00217     movwf   TRISB     ; ...port B
007A 3002      00218     movlw   ADCON1VAL ; set inputs of ...
007B 0088      00219     movwf   ADCON1    ; ...adc
007C 3088      00220     movlw   OPTIONVAL  ; configurate ...
007D 0081      00221     movwf   OPTION_REG ; ...Pic
007E 1283      00222     bcf    STATUS,RP0  ; register page 0
007F 30A0      00223     movlw   INTCONVAL ; configure interrupts and ...
0080 008B      00224     movwf   INTCON    ; ...enable interrupts
00225 ;--- main idle -----
0081          00226 Idle
0081 0064      00227     clrwdt  ; toggle watchdog
0082 1F8E      00228     btfss   COUNTER,07h ; if MSB isn't set, ...
0083 2881      00229     goto    Idle      ; ...then waiting
0084 0A8F      00230     incf    COUNTER2,F ; ...else toggle COUNTER2
00231           ; If the crystal freq is 4 MHz,
00232           ; the toggle freq is nearly 30.6 Hz
0085 138E      00233     bcf    COUNTER,07h ; reset MSB
0086 3089      00234     movlw   ADCON0VAL ; set ...
0087 0088      00235     movwf   ADCON0    ; ...ADC configuration
0088          00236 WaitNoInt
0088 0801      00237     movf    TMRO,W   ; waiting until enough time
0089 3CD0      00238     sublw   0d0h    ; for one conversion before start
008A 1C03      00239     btfss   STATUS,C  ; of the next timer interrupt.
008B 2888      00240     goto    WaitNoInt ; (Conv can be disturbed by
00241           ; an interrupt.
008C 1508      00242     bsf    ADCON0,GO ; start ADC
008D          00243 WaitAdc
008D 1908      00244     btfsc  ADCON0,GO ; waiting until ADC ...
008E 288D      00245     goto    WaitAdc ; ... is ready
008F 0809      00246     movf    ADRES,W  ; put result into W-reg and then
0090 0090      00247     movwf   PWMDESIRED ; desired PWM-value into PWMDESIRED
0091 0086      00248     movwf   PORTB    ; show result at port B (LEDs)
0092 2881      00249     goto    Idle    ;
00250 ;-----
00251     END

```

MEMORY USAGE MAP ('X' = Used, '-' = Unused)

```
0000 : X---XXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX  
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX  
0080 : XXXXXXXXXXXXXXXX XXX----- ----- ----- ----- ----- ----- -----
```

All other memory blocks unused.

Program Memory Words Used: 144
Program Memory Words Free: 880

Errors : 0
Warnings : 0 reported, 0 suppressed
Messages : 0 reported, 0 suppressed

APPENDIX B: FLOWCHARTS

FIGURE B-1: MAIN ROUTINE

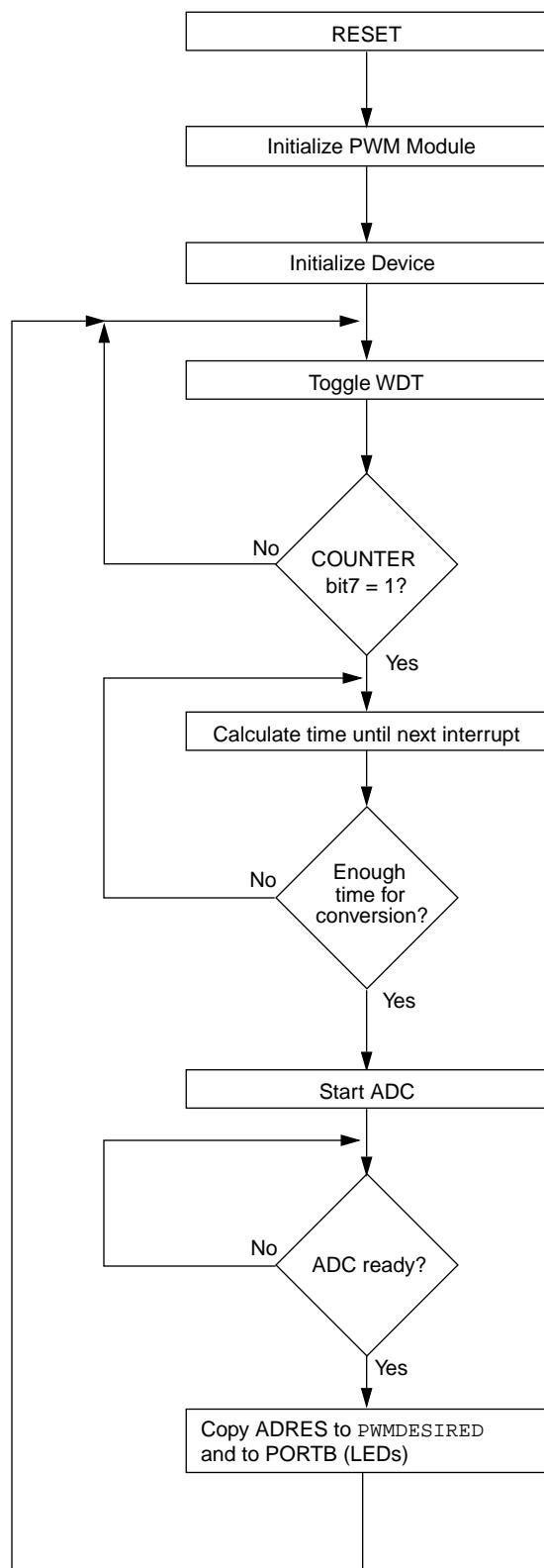


FIGURE B-2: PWM GENERATOR ROUTINE

