MICROCHIP

AN654

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.

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 an 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.

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
     .
     .
     .
```


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 T_{CY}. 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 V_{DD}. 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 • 128 • T_{CY} 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:

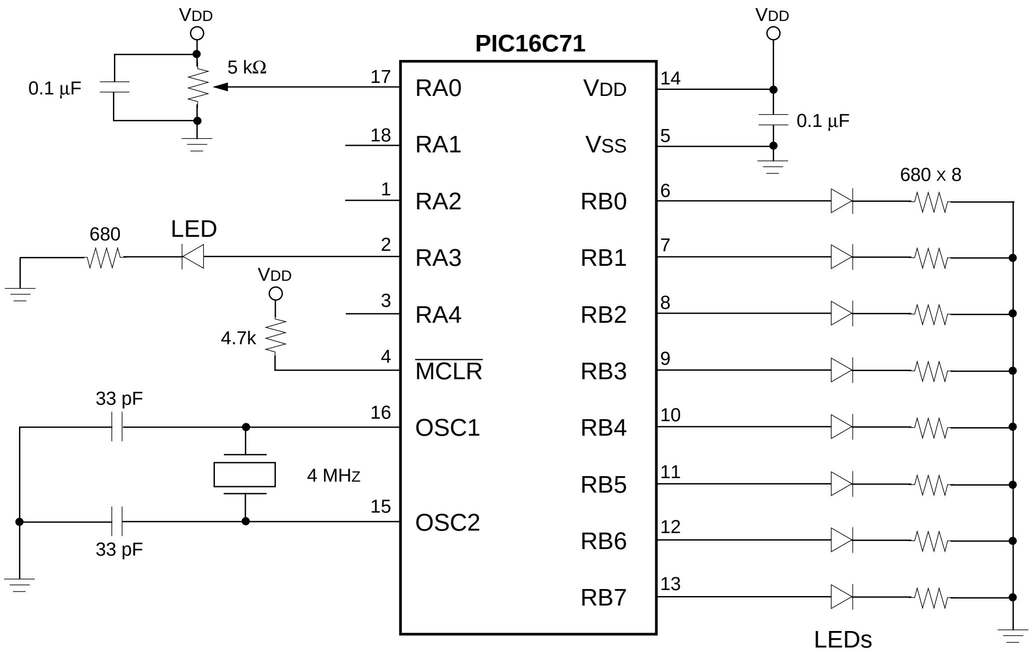


TABLE 1: OPERATION MODES OF THE SOFTWARE PWM MODULE

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     LINE SOURCE TEXT
  VALUE

                     ;******************************************************************
                     ; Filename:  PWM16CXX.ASM
                     ;******************************************************************
                     ;     Author:      Ole Ropcke
                     ;     Company:     private
                     ;     Revision:    RevA1
                     ;     Date:        01-4-96
                     ;     Assembled using MPASM rev 01.21
                     ;******************************************************************
                     ;     Include files:
                     ;       p16c71.inc
                     ;******************************************************************
                     ;     Required hardware:
                     ;******************************************************************
                     ;     What's changed:
                     ; Date      Description of Change
                     ; xx-xx-xx
                     ;******************************************************************

                      LIST P=16C71, R=DEC

                      include P16c71.INC
                             LIST
                     ; P16C71.INC  Std Hdr File, Version 1.00 Microchip Technology, Inc.
                             LIST

                     ;******************************************************************
                     ; Definitions
                     ;******************************************************************
                     ;
                     ; I/O, Interrupt and Option Definitions
                     ;
  00000088           OPTIONVAL         equ   10001000b   ; portB no pull-up, tmr0 int.
  000000A0           INTCONVAL         equ   0a0h        ; set GIE, T0IE
                                                           ; port A:
  00000000           UINPBIT           equ   00h         ; analog input for desired PWM
  00000003           PWMOUTBIT         equ   03h         ; PWM output
  00000017           TRISAVAL          equ   00010111b   ; A3 output
  00000002           ADCON1VAL         equ   2           ; A0,A1 analog; A2,A3 digital
  00000089           ADCON0VAL         equ   10001001b   ; fosc/32, channel 0
                                                         ; port B:
  00000000           TRISBVAL          equ   0           ; LED outputs
                     ;------------------------------------------------------------------
                     ; Register Definitions
                     ;
  0000000C           STACKW        equ     0ch     ; stack to push/pop the W-register
  0000000D           STACKS        equ     0dh     ; stack to push/pop the STATUS-reg
  0000000E           COUNTER       equ     0eh     ; counter: input frequency
                                                   ;   f1 = crystalfreq. / 4 / 255
  0000000F           COUNTER2      equ     0fh     ; counter2: input frequency
                                                   ;          f2 = f1 / 128
  00000010           PWMDESIRED    equ     10h     ; desired PWM value 0..255
  00000011           PWMMAX        equ     11h     ; register to support generating PWM
                                                   ; You have to put PWMMAXVAL into it!
  00000012           PWMHELP       equ     12h     ; register to support generating PWM
                                                   ; used as temp storage of PWMDESIRED
                     ;------------------------------------------------------------------
                     ; PWM-module-constant
  00000016           PWMADJUSTVAL  equ     .22
```



```
                     ; correction number, defined by the following factors:
                     ; time from timer interrupt to executing PC 004 + 3 cycles
                     ; computing time from PC=004 to required edge   +18 cycles
                     ; lost timer cycles due to writing the timer    + 2 cycles
                     ; cal desired PWM value to timer loading value  + 2 cycles
                     ; time from timer loading to gen required edge  - 1 cycle
                     ; valid value for hardware (unknown diff to the data sheet)
                     ; 3+18+0+2-1=22
                     ; valid value for PICSIM version 5.11 (error of PICSIM):
                     ; 0+18+2+2-1=21
  0000001D     PWMMAXVAL       equ     .29
                     ; loading value for PWMMAX
                     ; If n is the maximum length of a high pulse, which has to be
                     ; generated by the skipping method, then is PWMMAXVAL = n+1.
                     ; The max length of a low pulse using the skip method is n-1.
               ;=================================================================
0000                    ORG     0
0000 2870               goto    PowerOn
               ;--- PWM-Generator -----------------------------------------------
0004                    ORG     04h                ; timer interrupt
0004 1801               btfsc   TMR0,0             ; compensate comp time of 1/2 cyc
0005 2806               goto    PwmInt             ; instruc when timer int occured
0006           PwmInt
0006 008C               movwf   STACKW             ; copy W register... to "stack"
0007 0E8C               swapf   STACKW,F           ; ...to "stack"
0008 0E03               swapf   STATUS,W           ; copy STATUS register...
0009 008D               movwf   STACKS             ; ...to "stack"
000A 110B               bcf     INTCON,T0IF        ; clear interrupt flag
000B 1985               btfsc   PORTA,PWMOUTBIT    ; which edge is required?
000C 2840               goto    Lowpulse           ; -> goto falling edge
000D           Highpulse
000D 0910               comf    PWMDESIRED,W       ; get desired PWM value
000E 0092               movwf   PWMHELP            ; store val for the foll low pulse
000F 0791               addwf   PWMMAX,F           ; calc number of inst’s to skip
0010 1C03               btfss   STATUS,C           ; which method to use?
0011 2836               goto    HighImpInt         ; -> using interrupt
0012           HighImpShrt
0012 0811               movf    PWMMAX,W           ; get number of inst’s to skip
0013 0782               addwf   PCL,F              ; skip n instructions
0014 1585               bsf     PORTA,PWMOUTBIT    ; rising edge, 28 cycles hi pulse
0015 1585               bsf     PORTA,PWMOUTBIT    ; 27 cycles
0016 1585               bsf     PORTA,PWMOUTBIT    ; 26 cycles
0017 1585               bsf     PORTA,PWMOUTBIT    ; 25 cycles
0018 1585               bsf     PORTA,PWMOUTBIT    ; 24 cycles
0019 1585               bsf     PORTA,PWMOUTBIT    ; 23 cycles
001A 1585               bsf     PORTA,PWMOUTBIT    ; 22 cycles
001B 1585               bsf     PORTA,PWMOUTBIT    ; 21 cycles
001C 1585               bsf     PORTA,PWMOUTBIT    ; 20 cycles
001D 1585               bsf     PORTA,PWMOUTBIT    ; 19 cycles
001E 1585               bsf     PORTA,PWMOUTBIT    ; 18 cycles
001F 1585               bsf     PORTA,PWMOUTBIT    ; 17 cycles
0020 1585               bsf     PORTA,PWMOUTBIT    ; 16 cycles
0021 1585               bsf     PORTA,PWMOUTBIT    ; 15 cycles
0022 1585               bsf     PORTA,PWMOUTBIT    ; 14 cycles
0023 1585               bsf     PORTA,PWMOUTBIT    ; 13 cycles
0024 1585               bsf     PORTA,PWMOUTBIT    ; 12 cycles
0025 1585               bsf     PORTA,PWMOUTBIT    ; 11 cycles
0026 1585               bsf     PORTA,PWMOUTBIT    ; 10 cycles
0027 1585               bsf     PORTA,PWMOUTBIT    ;  9 cycles
0028 1585               bsf     PORTA,PWMOUTBIT    ;  8 cycles
0029 1585               bsf     PORTA,PWMOUTBIT    ;  7 cycles
002A 1585               bsf     PORTA,PWMOUTBIT    ;  6 cycles
002B 1585               bsf     PORTA,PWMOUTBIT    ;  5 cycles
002C 1585               bsf     PORTA,PWMOUTBIT    ;  4 cycles
002D 1585               bsf     PORTA,PWMOUTBIT    ;  3 cycles
002E 1585               bsf     PORTA,PWMOUTBIT    ;  2 cycles
```

```
002F 1585              bsf     PORTA,PWMOUTBIT  ;  1 cycle
0030 1185              bcf     PORTA,PWMOUTBIT  ; fall edge;start of the following
                                                ; low pulse using the interrupt
0031 0A8E              incf    COUNTER,F        ; trigger COUNTER, cause there was
                                                ; a rising edge
0032 0912              comf    PWMHELP,W        ; get required low pulse length
0033 3E1B              addlw   PWMADJUSTVAL+5   ; calculate timer loading value
                                                ; Edge was generated 5 cycles before
                                                ; usual point of time.
0034 0081              movwf   TMR0             ; put value into timer
0035 2869              goto    LowImpInt2       ; low pulse using int is running
0036             HighImpInt                     ; high pulse using interrupt
0036 3E16              addlw   PWMADJUSTVAL     ; calculate timer loading value
0037 0081              movwf   TMR0             ; put value into timer
0038             HighImpInt2
0038 1585              bsf     PORTA,PWMOUTBIT  ; generate rising edge
0039 0A8E              incf    COUNTER,F        ; trigger counter, because there
                                                ; was a rising edge
003A 301C              movlw   PWMMAXVAL-1      ; "repair"...
003B 0091              movwf   PWMMAX           ; ...support register
003C 0E0D              swapf   STACKS,W         ; restore...
003D 0083              movwf   STATUS           ; ...STATUS register
003E 0E0C              swapf   STACKW,W         ; restore W register
003F 0009              retfie                   ; return to main program
0040             Lowpulse
0040 0912              comf    PWMHELP,W        ; get required pulse length
0041 0791              addwf   PWMMAX,F         ; calc number of inst’s to skip
0042 1C03              btfss   STATUS,C         ; which method is to use?
0043 2867              goto    LowImpInt        ; ->using interrupt
0044             LowImpShrt
0044 0811              movf    PWMMAX,W         ; get number of inst’s to skip
0045 0782              addwf   PCL,F            ; skip n instructions
0046 1185              bcf     PORTA,PWMOUTBIT  ; falling edge, 27 cycles low pulse
0047 1185              bcf     PORTA,PWMOUTBIT  ; 26 cycles
0048 1185              bcf     PORTA,PWMOUTBIT  ; 25 cycles
0049 1185              bcf     PORTA,PWMOUTBIT  ; 24 cycles
004A 1185              bcf     PORTA,PWMOUTBIT  ; 23 cycles
004B 1185              bcf     PORTA,PWMOUTBIT  ; 22 cycles
004C 1185              bcf     PORTA,PWMOUTBIT  ; 21 cycles
004D 1185              bcf     PORTA,PWMOUTBIT  ; 20 cycles
004E 1185              bcf     PORTA,PWMOUTBIT  ; 19 cycles
004F 1185              bcf     PORTA,PWMOUTBIT  ; 18 cycles
0050 1185              bcf     PORTA,PWMOUTBIT  ; 17 cycles
0051 1185              bcf     PORTA,PWMOUTBIT  ; 16 cycles
0052 1185              bcf     PORTA,PWMOUTBIT  ; 15 cycles
0053 1185              bcf     PORTA,PWMOUTBIT  ; 14 cycles
0054 1185              bcf     PORTA,PWMOUTBIT  ; 13 cycles
0055 1185              bcf     PORTA,PWMOUTBIT  ; 12 cycles
0056 1185              bcf     PORTA,PWMOUTBIT  ; 11 cycles
0057 1185              bcf     PORTA,PWMOUTBIT  ; 10 cycles
0058 1185              bcf     PORTA,PWMOUTBIT  ;  9 cycles
0059 1185              bcf     PORTA,PWMOUTBIT  ;  8 cycles
005A 1185              bcf     PORTA,PWMOUTBIT  ;  7 cycles
005B 1185              bcf     PORTA,PWMOUTBIT  ;  6 cycles
005C 1185              bcf     PORTA,PWMOUTBIT  ;  5 cycles
005D 1185              bcf     PORTA,PWMOUTBIT  ;  4 cycles
005E 1185              bcf     PORTA,PWMOUTBIT  ;  3 cycles
005F 1185              bcf     PORTA,PWMOUTBIT  ;  2 cycles
0060 1185              bcf     PORTA,PWMOUTBIT  ;  1 cycle
0061 1585              bsf     PORTA,PWMOUTBIT  ; rising edge; start of the next
                                                ; high pulse using the interrupt
0062 0910              comf    PWMDESIRED,W     ; get desired PWM value
0063 0092              movwf   PWMHELP          ; store val for the next lo pulse
0064 3E1B              addlw   PWMADJUSTVAL+5   ; calculate timer loading value
                                                ; Edge was gen’d 5 cycles before
                                                ; usual point of time.
```



```
0065 0081                   movwf   TMR0              ; put value into timer
0066 2838                   goto    HighImpInt2       ; high pulse using int is running
0067            LowImpInt                             ; low pulse using interrupt
0067 3E16                   addlw   PWMADJUSTVAL      ; calculate timer loading value
0068 0081                   movwf   TMR0              ; put value into timer
0069            LowImpInt2
0069 1185                   bcf     PORTA,PWMOUTBIT   ; generate falling edge
006A 301D                   movlw   PWMMAXVAL         ; "repair" ...
006B 0091                   movwf   PWMMAX            ; ...support register
006C 0E0D                   swapf   STACKS,W          ; restore ...
006D 0083                   movwf   STATUS            ; ...STATUS register
006E 0E0C                   swapf   STACKW,W          ; restore W register
006F 0009                   retfie                    ; return to main program
                ;--- power on routine --------------------------------------------
0070            PowerOn
                            ; configuration of the PWM module
0070 0181                   clrf    TMR0              ; reset timer
0071 0190                   clrf    PWMDESIRED        ; reset value of PWM is 0
0072 1185                   bcf     PORTA,PWMOUTBIT   ; reset PWM-port before port A is
                                                      ; changed from input to output to
                                                      ; suppress an uncontrolled spike
0073 301D                   movlw   PWMMAXVAL         ; set support register
0074 0091                   movwf   PWMMAX            ;
                            ; configuration of the Pic
0075 1683                   bsf     STATUS,RP0        ; register page 1
0076 3017                   movlw   TRISAVAL          ; configurate ...
0077 0085                   movwf   TRISA             ; ...port A
0078 3000                   movlw   TRISBVAL          ; configurate ...
0079 0086                   movwf   TRISB             ; ...port B
007A 3002                   movlw   ADCON1VAL         ; set inputs of ...
007B 0088                   movwf   ADCON1            ; ...adc
007C 3088                   movlw   OPTIONVAL         ; configurate ...
007D 0081                   movwf   OPTION_REG        ; ...Pic
007E 1283                   bcf     STATUS,RP0        ; register page 0
007F 30A0                   movlw   INTCONVAL         ; configure interrupts and ...
0080 008B                   movwf   INTCON            ; ...enable interrupts
                ;--- main idle ----------------------------------------------------
0081            Idle
0081 0064                   clrwdt                    ; toggle watchdog
0082 1F8E                   btfss   COUNTER,07h       ; if MSB isn't set, ...
0083 2881                   goto    Idle              ; ...then waiting
0084 0A8F                   incf    COUNTER2,F        ; ...else toggle COUNTER2
                                                      ; If the crystal freq is 4 MHz,
                                                      ; the toggle freq is nearly 30.6 Hz
0085 138E                   bcf     COUNTER,07h       ; reset MSB
0086 3089                   movlw   ADCON0VAL         ; set ...
0087 0088                   movwf   ADCON0            ; ...ADC configuration
0088            WaitNoInt
0088 0801                   movf    TMR0,W            ; waiting until enough time
0089 3CD0                   sublw   0d0h              ; for one conversion before start
008A 1C03                   btfss   STATUS,C          ; of the next timer interrupt.
008B 2888                   goto    WaitNoInt         ; (Conv can be disturbed by
                                                      ; an interrupt.
008C 1508                   bsf     ADCON0,GO         ; start ADC
008D            WaitAdc
008D 1908                   btfsc   ADCON0,GO         ; waiting until ADC ...
008E 288D                   goto    WaitAdc           ; ... is ready
008F 0809                   movf    ADRES,W           ; put result into W-reg and then
0090 0090                   movwf   PWMDESIRED        ; desired PWM-value into PWMDESIRED
0091 0086                   movwf   PORTB             ; show result at port B (LEDs)
0092 2881                   goto    Idle              ;
                ;-----------------------------------------------------------------
                            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

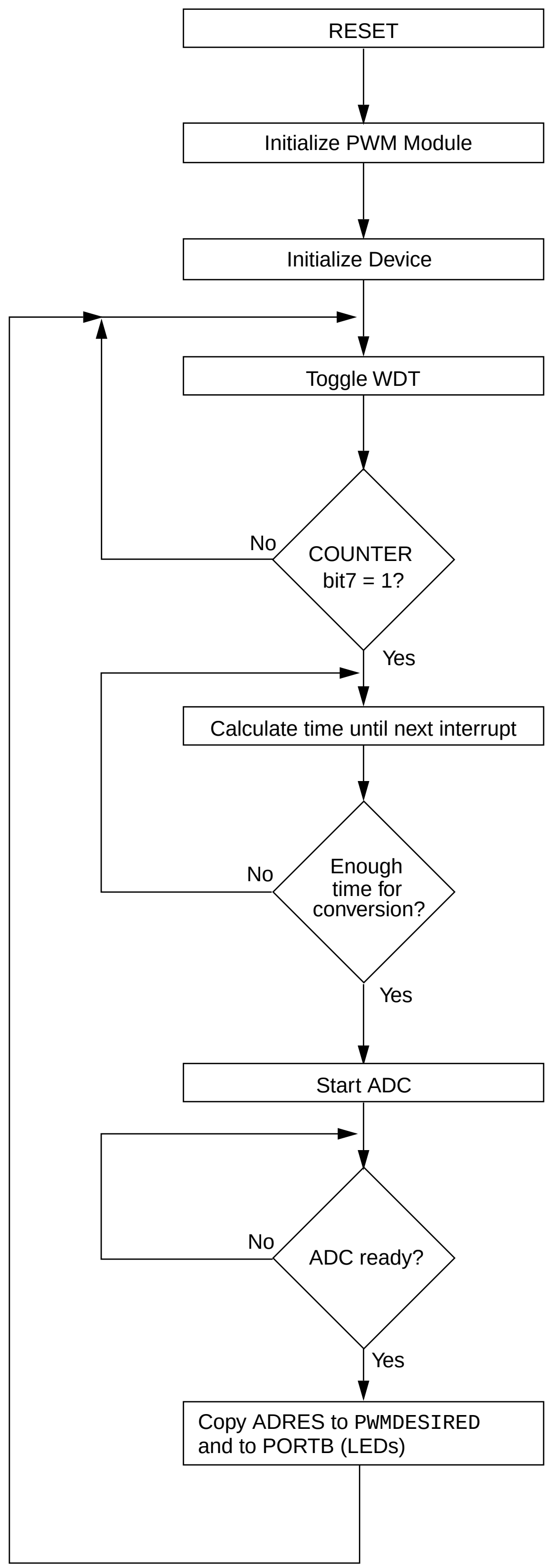
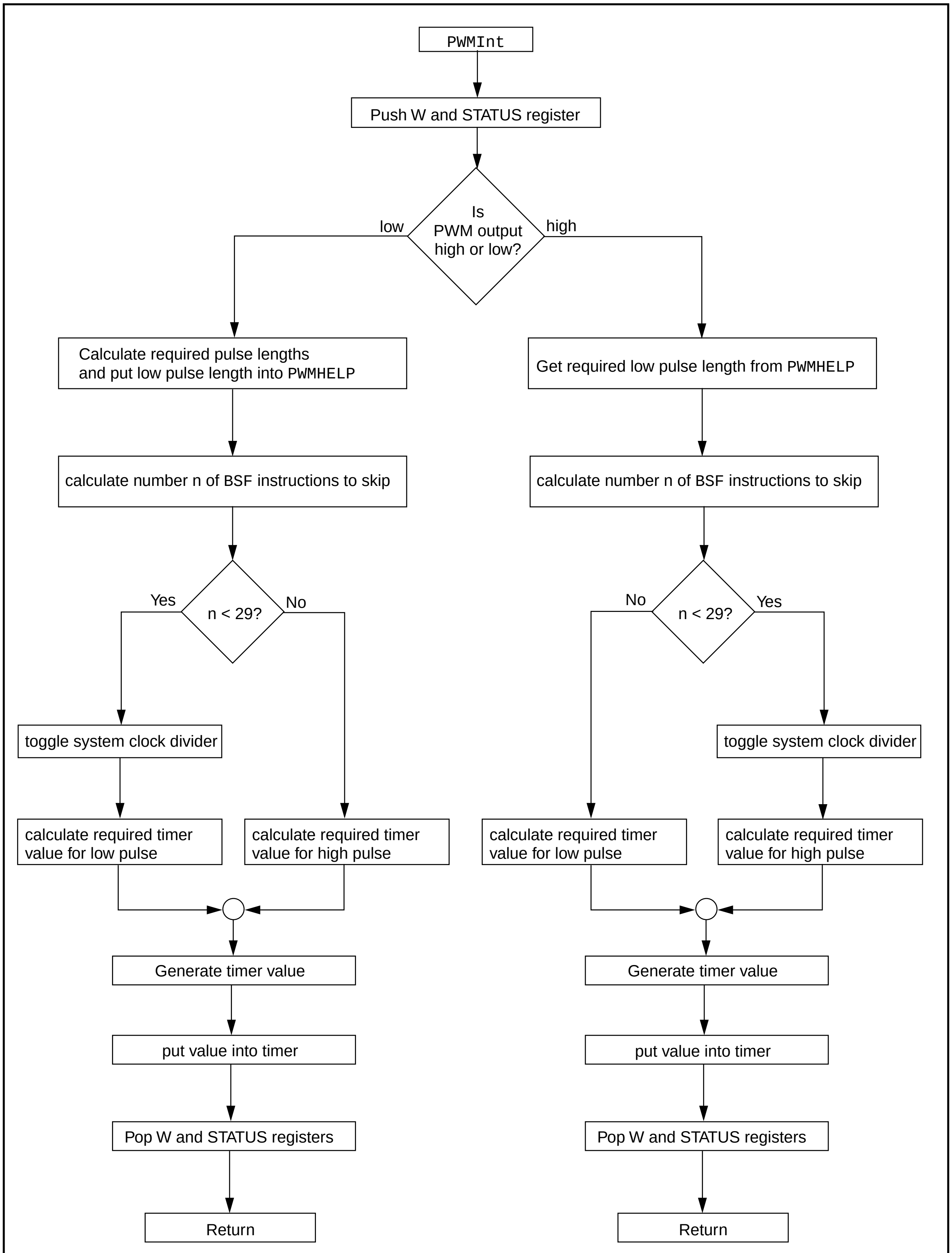


FIGURE B-2: PWM GENERATOR ROUTINE



NOTES:

Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company’s quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001 certified.

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Rocky Mountain
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

Atlanta
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Chicago
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Detroit
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Kokomo
2767 S. Albright Road
Kokomo, Indiana 46902
Tel: 765-864-8360 Fax: 765-864-8387

Los Angeles
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia
Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing
Microchip Technology Consulting (Shanghai) Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu
Microchip Technology Consulting (Shanghai) Co., Ltd., Chengdu Liaison Office
Rm. 2401, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-6766200 Fax: 86-28-6766599

China - Fuzhou
Microchip Technology Consulting (Shanghai) Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

China - Shanghai
Microchip Technology Consulting (Shanghai) Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

China - Shenzhen
Microchip Technology Consulting (Shanghai) Co., Ltd., Shenzhen Liaison Office
Rm. 1315, 13/F, Shenzhen Kerry Centre,
Renminnan Lu
Shenzhen 518001, China
Tel: 86-755-2350361 Fax: 86-755-2366086

Hong Kong
Microchip Technology Hongkong Ltd.
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India
Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan
Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Singapore
Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan
Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Denmark
Microchip Technology Nordic ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France
Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany
Microchip Technology GmbH
Gustav-Heinemann Ring 125
D-81739 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy
Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

United Kingdom
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

01/18/02