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 T<sub>CY</sub> (Instruction Cycle Time).
5. Simultaneous generation of a system time clock.

EXAMPLE 1: USING T<sub>CY</sub> 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 (T<sub>CY</sub>). We will use T<sub>CY</sub> because one instruction (if the program counter is not changed) executes in one T<sub>CY</sub> and Timer0 (without prescaling) increments every T<sub>CY</sub>. 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 T<sub>CY</sub> on pin 3 of PORTA. The pulse length is controlled by the value of register LENGTH in steps of 3 T<sub>CY</sub>. 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 T$_{CY}$. 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 T$_{CY}$ will be generated. If LENGTH is set to 0, no instructions will be skipped and the length of the pulse remains 5 T$_{CY}$. 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 T$_{CY}$. Between each length the step size is only 1 T$_{CY}$. 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 • 128 • Tcy and is used as the system clock bit.

The file register PWMDESIRRED 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 PWMMAXVAL 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

<table>
<thead>
<tr>
<th>Duty Cycle Range</th>
<th>High Pulse</th>
<th>Low Pulse</th>
</tr>
</thead>
<tbody>
<tr>
<td>0% .. 10%</td>
<td>Skipping (Method B)</td>
<td>Timer Interrupt</td>
</tr>
<tr>
<td>10% .. 90%</td>
<td>Timer Interrupt</td>
<td>Timer Interrupt</td>
</tr>
<tr>
<td>90% .. 100%</td>
<td>Timer Interrupt</td>
<td>Skipping (Method B)</td>
</tr>
</tbody>
</table>
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

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
00023
00024 ;------------------------------------------------------------------
00025 ; Register Definitions
00026 ;------------------------------------------------------------------
00027 ;
00028 ; I/O, Interrupt and Option Definitions
00029 ;
00030 OPTIONVAL equ 10001000b ; portB no pull-up, tmr0 int.
00031 INTCONVAL equ 0a0h ; set GIE, TOIE
00032 ; port A:
00033 UINPBIT equ 00h ; analog input for desired PWM
00034 PWMOUTBIT equ 03h ; PWM output
00035 TRISVAL equ 00100111b ; A3 output
00036 ADCONIVAL equ 00010111b ; A0,A1 analog; A2,A3 digital
00037 ADCON0VAL equ 0001001b ; fosc/32, channel 0
00038 ; port B:
00039 TRISBVAL equ 0 ; LED outputs
00040 ;------------------------------------------------------------------
00041 ; Register Definitions
00042 ;
00043 STACKW equ 0ch ; stack to push/pop the W-register
00044 STACKS equ 0dh ; stack to push/pop the STATUS-reg
00045 COUNTER equ 0eh ; counter: input frequency
00046 ; f1 = crystalfreq / 4 / 255
00047 COUNTER2 equ 0fh ; counter2: input frequency
00048 ; f2 = f1 / 128
00049 PWMDESIRED equ 10h ; desired PWM value 0..255
00050 PWMMAX equ 11h ; register to support generating PWM
00051 ; You have to put PWMMAXVAL into it!
00052 PWNHELP equ 12h ; register to support generating PWM
00053 ; used as temp storage of PWMDESIRED
00054 ;------------------------------------------------------------------
00055 ; PWM-module-constant
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
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 ;==================================================================
00073 ORG 0
00074 goto PowerOn
00075 ;--- PWM-Generator ------------------------------------------------
00076 ORG 04h ; timer interrupt
00077 btfscc TMRO,0 ; compensate comp time of 1/2 cyc
00078 goto PwmInt ; instruc when timer int occured
00079 PwmInt
00080 movf STACKW ; copy W register... to "stack"
00081 swapf STACKW,F ; ...to "stack"
00082 swapf STATUS,W ; copy STATUS register...
00083 movwf STACKS ; ...to "stack"
00084 bcf INTCON,T0IF ; clear interrupt flag
00085 btfs PORTA,PWMOUTBIT ; which edge is required?
00086 goto Lowpulse ; -> goto falling edge
00087 Highpulse
00088 movf PWMDESIRED,W ; get desired PWM value
00089 addwf PWMMAX,F ; store val for the foll low pulse
00090 btfs STATUS,C ; which method to use?
00091 goto HighImpInt ; -> using interrupt
00092 HighImpShrt
00093 movf PWMMAX,W ; get number of inst’s to skip
00094 addwf PCL,F ; skip n instructions
00095 bsf PORTA,PWMOUTBIT ; rising edge, 28 cycles hi pulse
00096 bsf PORTA,PWMOUTBIT ; 27 cycles
00097 bsf PORTA,PWMOUTBIT ; 26 cycles
00098 bsf PORTA,PWMOUTBIT ; 25 cycles
00099 bsf PORTA,PWMOUTBIT ; 24 cycles
00100 bsf PORTA,PWMOUTBIT ; 23 cycles
00101 bsf PORTA,PWMOUTBIT ; 22 cycles
00102 bsf PORTA,PWMOUTBIT ; 21 cycles
00103 bsf PORTA,PWMOUTBIT ; 20 cycles
00104 bsf PORTA,PWMOUTBIT ; 19 cycles
00105 bsf PORTA,PWMOUTBIT ; 18 cycles
00106 bsf PORTA,PWMOUTBIT ; 17 cycles
00107 bsf PORTA,PWMOUTBIT ; 16 cycles
00108 bsf PORTA,PWMOUTBIT ; 15 cycles
00109 bsf PORTA,PWMOUTBIT ; 14 cycles
00110 bsf PORTA,PWMOUTBIT ; 13 cycles
00111 bsf PORTA,PWMOUTBIT ; 12 cycles
00112 bsf PORTA,PWMOUTBIT ; 11 cycles
00113 bsf PORTA,PWMOUTBIT ; 10 cycles
00114 bsf PORTA,PWMOUTBIT ; 9 cycles
00115 bsf PORTA,PWMOUTBIT ; 8 cycles
00116 bsf PORTA,PWMOUTBIT ; 7 cycles
00117 bsf PORTA,PWMOUTBIT ; 6 cycles
00118 bsf PORTA,PWMOUTBIT ; 5 cycles
00119 bsf PORTA,PWMOUTBIT ; 4 cycles
00120 bsf PORTA,PWMOUTBIT ; 3 cycles
00121 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 0791 00128  addwf PWMMAX,F ; get required pulse length
00129  ; calculate timer loading value
00130  ; Edge was generated 5 cycles before
00131  ; usual point of time.
0033 0092 00132  movwf PWMMAX ; put value into timer
00133  goto LowImpInt2 ; low pulse using int is running
0034 3E1B 00134  addlw PWMADJUSTVAL+5 ; calculate timer loading value
00135  ; HighImpInt
0035 3E16 00136  addlw PWMADJUSTVAL ; calculate timer loading value
00137  ; HighImpInt2
0036 0081 00138  bcf PORTA,PWMOUTBIT ; generate rising edge
00139  incf COUNTER,F ; trigger counter, because there
00140  ; was a rising edge
0037 0083 00141  movwf TMR0 ; put value into timer
00142  ; HighImpInt
0038 0791 00143  addwf PWMMAX,F ; calculate number of inst’s to skip
00144  ; which method is to use?
0039 0782 00145  btfss STATUS,C ; -->using interrupt
00146  ; LowImpInt
0040 0811 00147  movf PWMMAX,W ; get number of inst’s to skip
00148  ; falling edge, 27 cycles low pulse
0041 0791 00149  addwf PWMMAX,F ; skip n instructions
00150  ; PWMOUTBIT
0042 1185 00151  bcf PORTA,PWMOUTBIT ; 26 cycles
00152  ; PWMOUTBIT
0043 1185 00153  bcf PORTA,PWMOUTBIT ; 25 cycles
00154  ; PWMOUTBIT
0044 1185 00155  bcf PORTA,PWMOUTBIT ; 24 cycles
00156  ; PWMOUTBIT
0045 1185 00157  bcf PORTA,PWMOUTBIT ; 23 cycles
00158  ; PWMOUTBIT
0046 1185 00159  bcf PORTA,PWMOUTBIT ; 22 cycles
00160  ; PWMOUTBIT
0047 1185 00161  bcf PORTA,PWMOUTBIT ; 21 cycles
00162  ; PWMOUTBIT
0048 1185 00163  bcf PORTA,PWMOUTBIT ; 20 cycles
00164  ; PWMOUTBIT
0049 1185 00165  bcf PORTA,PWMOUTBIT ; 19 cycles
00166  ; PWMOUTBIT
0050 1185 00167  bcf PORTA,PWMOUTBIT ; 18 cycles
00168  ; PWMOUTBIT
0051 1185 00169  bcf PORTA,PWMOUTBIT ; 17 cycles
00170  ; PWMOUTBIT
0052 1185 00171  bcf PORTA,PWMOUTBIT ; 16 cycles
00172  ; PWMOUTBIT
0053 1185 00173  bcf PORTA,PWMOUTBIT ; 15 cycles
00174  ; PWMOUTBIT
0054 1185 00175  bcf PORTA,PWMOUTBIT ; 14 cycles
00176  ; PWMOUTBIT
0055 1185 00177  bcf PORTA,PWMOUTBIT ; 13 cycles
00178  ; PWMOUTBIT
0056 1185 00179  bcf PORTA,PWMOUTBIT ; 12 cycles
00180  ; PWMOUTBIT
0057 1185 00181  bcf PORTA,PWMOUTBIT ; 11 cycles
00182  ; PWMOUTBIT
0058 1185 00183  bcf PORTA,PWMOUTBIT ; 10 cycles
00184  ; PWMOUTBIT
0059 1185 00185  bcf PORTA,PWMOUTBIT ; 9 cycles
00186  ; PWMOUTBIT
0060 3E1B 00187  movwf PWMMAX ; store val for the next lo pulse
00188  ; PWMOUTBIT
0061 0092 00189  movf PWMMAX,W ; get desired PWM value
00190  ; PWMOUTBIT
0062 0791 00191  addwf PWMMAX,F ; calculate timer loading value
00192  ; Edge was gen’d 5 cycles before
00193  ; usual point of time.
0065 0081 00189  movwf  TMR0  ; put value into timer
0066 2838 00190  goto  HighImpInt2  ; high pulse using int is running
0067 00191  LowImpInt  ; low pulse using interrupt
0067 3E16 00192  addlw  PWMADJUSTVAL  ; calculate timer loading value
0068 0081 00193  movwf  TMR0  ; put value into timer
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 00E0 00198  swapf  STACKS,W  ; restore ...
006D 0083 00199  movwf  STATUS  ; ...STATUS register
006E 00E0 00200  swapf  STACKW,W  ; restore W register
006F 0009 00201  retfie  ; return to main program
00202 ;--- power on routine ---------------------------------------------
0070 00203  PowerOn
0070 0181 00204  ; configuration of the PWM module
0071 0190 00205  clrf  TMR0  ; reset timer
0072 0185 00206  clrf  PWMDESIRED  ; reset value of PWM is 0
0072 0195 00207  bcf  PORTA, PWMOUTBIT  ; reset PWM-port before port A is
0072 00208  changed from input to output to
0072 00209  suppress an uncontrolled spike
0073 019D 00210  movlw  PWMMAXVAL  ; set support register
0074 0091 00211  movwf  PWMMAX  ;
0074 00212  ; configuration of the Pic
0075 1683 00213  bsf  STATUS, RP0  ; register page 1
0076 0085 00214  movlw  TRISAVAL  ; configure ...
0076 0081 00215  movwf  TRISA  ; ...port A
0077 3000 00216  movlw  TRISBVAL  ; configure ...
0078 0086 00217  movwf  TRISB  ; ...port B
007A 0082 00218  movlw  ADCON1VAL  ; set inputs of ...
007B 0088 00219  movwf  ADCON1  ; ...adc
007C 0082 00220  movlw  OPTIONVAL  ; configure ...
007D 0081 00221  movwf  OPTION_REG  ; ...Pic
007E 1283 00222  bcf  STATUS, RP0  ; register page 0
007F 0080 00223  movlw  INTOCONVAL  ; configure interrupts and ...
0080 0088 00224  movwf  INTOCON  ; ...enable interrupts
00225 ;--- main idle ----------------------------------------------------
0081 0226  Idle
0081 0064 00227  clrwdt  ; toggle watchdog
0082 1FBE 00228  btfs 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 13BF 00233  bcf  COUNTER, 07h  ; reset MSB
0086 3089 00234  movlw  ADCON0VAL  ; set ...
0087 008E 00235  movwf  ADCON0  ; ...ADC configuration
0088 0236  WaitNoInt
0088 0801 00237  movf  TMRO,W  ; waiting until enough time
0089 3C00 00238  sublw  0d0h  ; for one conversion before start
008A 1C03 00239  btfs  STATUS,C  ; of the next timer interrupt.
008B 2888 00240  goto  WaitNoInt  ; (Conv can be disturbed by
00241  an interrupt.
008C 1508 00242  bcf  ADCON0, GO  ; start ADC
008D 0243  WaitAdc
008D 1908 00244  btfs  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  movf  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)

<table>
<thead>
<tr>
<th>Address</th>
<th>Memory Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000-003F</td>
<td>X---XXXXXXXXX X........X........X X........X........X</td>
</tr>
<tr>
<td>0040-007F</td>
<td>XX........X........X X........X........X X........X........X</td>
</tr>
<tr>
<td>0080-00B7</td>
<td>XX........X........X X------------- -------------- --------------</td>
</tr>
</tbody>
</table>

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
FIGURE B-1: MAIN ROUTINE

- **RESET**
- Initialize PWM Module
- Initialize Device
- Toggle WDT
- **No** COUNTER bit7 = 1?
  - **Yes** Calculate time until next interrupt
  - **No** Enough time for conversion?
    - **Yes** Start ADC
    - **No** ADC ready?
      - **Yes** Copy ADRES to PWMDESIRED and to PORTB (LEDs)
FIGURE B-2: PWM GENERATOR ROUTINE

PWMInt

Push W and STATUS register

Is PWM output high or low?

Calculate required pulse lengths and put low pulse length into PWMHELP

Calculate number n of BSF instructions to skip

Yes

n < 29?

No

toggle system clock divider

calculate required timer value for low pulse

Generate timer value

Put value into timer

Pop W and STATUS registers

Return

Get required low pulse length from PWMHELP

calculate number n of BSF instructions to skip

No

n < 29?

Yes

toggle system clock divider

calculate required timer value for high pulse

calculate required timer value for low pulse

Generate timer value

Put value into timer

Pop W and STATUS registers

Return
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.
WORLDWIDE SALES AND SERVICE

AMERICAS
Corporate Office
2335 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
2335 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: 770-640-0075

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: 972-818-7924 Fax: 972-818-2924

Detroit
Tri-Aria 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: 765-864-8397

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: 631-273-5305

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: 61-2-9888-6733 Fax: 61-2-9888-6755

China - Beijing
Microchip Technology Consulting (Shanghai)
Co., Ltd., Beijing Liaison Office
Unit 915
Beijing, 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: 86-28-6766699

China - Fuzhou
Microchip Technology Consulting (Shanghai)
Co., Ltd., Fuzhou Liaison Office
Unit 7F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503505 Fax: 86-591-7503521

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, 189880
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
Lastrup Høj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France
Microchip Technology SARL
Parc d’Activité du Moulin de Massy
43 Rue du Saule Trapu
Bâtiment A - Ier Etage
91900 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

© 2002 Microchip Technology Inc.