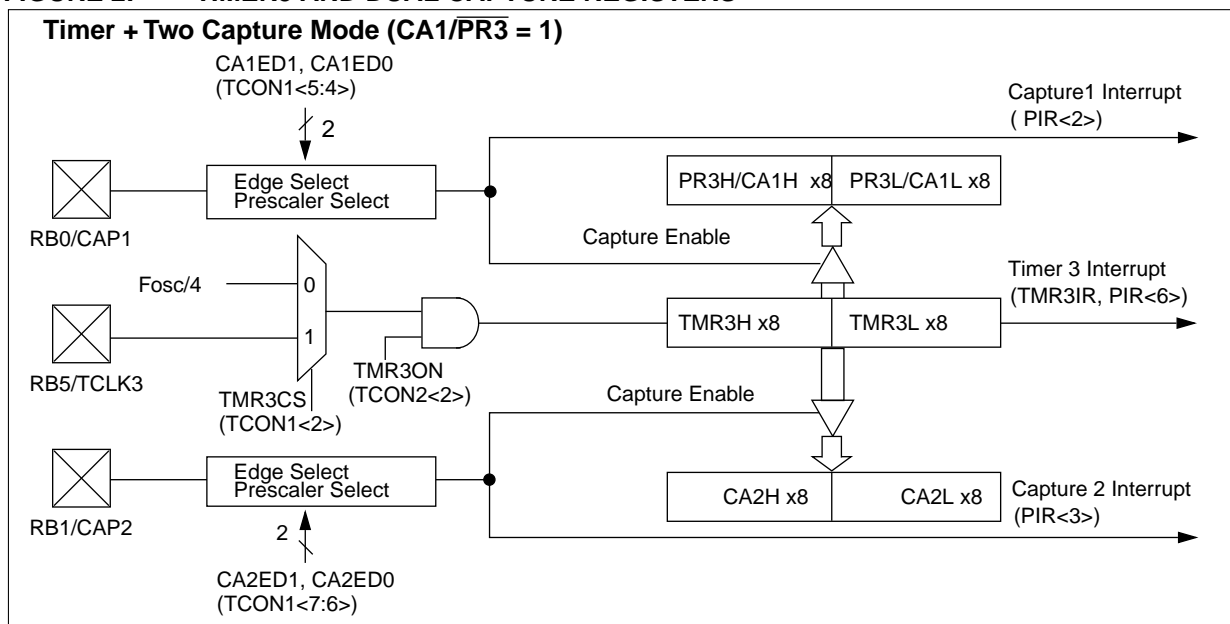




**FIGURE 2: TIMER3 AND DUAL CAPTURE REGISTERS**



The period register allows the time base of Timer3 to be something other than the  $2^{16}$  counter overflow value, which corresponds to FFFFh (65536) cycles. This is accomplished by loading the desired period value into the PR3H/CA1H:PR3L/CA1L register pair. The overflow time can be calculated by this equation:

$$TOFL = TCLK \cdot (\text{value in PR3H/CA1H:PR3L/CA1L register pair} + 1).$$

Where TCLK is either the internal system clock (Tcy) or the external clock cycle time. Table 1 shows time-out periods for different period values at different frequencies. The values in the register are the closest approximation for the period value. All examples in this application note use a Timer3 overflow value of FFFFh.

TABLE 1: TIMER3 OVERFLOW TIMES

Period Register						
Overflow Time	@ 16 MHz (250 ns)	@ 10 MHz (400 ns)	@ 8 MHz (500 ns)	@ 5 MHz (500 ns)	@ 2 MHz (2.0 $\mu$ s)	@ 32 kHz (125 $\mu$ s)
8.192 s	N.A.	N.A.	N.A.	N.A.	N.A.	0xFFFF
131.072 ms	N.A.	N.A.	N.A.	N.A.	0xFFFF	0x0418
52.428 ms	N.A.	N.A.	N.A.	0xFFFF	0x6666	0x01A3
32.7675 ms	N.A.	N.A.	0xFFFF	0x9FFF	0x3FFF	0x0106
26.214 ms	N.A.	0xFFFF	0xCE20	0x80D4	0x3388	0x00D3
16.384 ms	0xFFFF	0xA000	0x8000	0x5000	0x2000	0x0083
10.0 ms	0x9C40	0x61A8	0x4E20	0x30D4	0x1388	0x0050
4.0 ms	0x3E80	0x2710	0x1F40	0x1388	0x07D0	0x0020
1.0 ms	0x0FA0	0x09C4	0x07D0	0x04E2	0x01F4	0x0008
600 $\mu$ s	0x0960	0x05DC	0x04B0	0x02EE	0x012C	0x0005
100 $\mu$ s	0x0190	0x00FA	0x00C8	0x007D	0x0032	N.A.

The uses of an input capture are all for time based measurements. These include:

- Frequency measurement
- Duty cycle and pulse width measurements

The PIC17C42 has two pins (RB0/CAP1 and RB1/CAP2) which can be used for capturing the Timer3 value, when a specified edge occurs. The input capture can be specified to occur on one of the following four events:

- Falling Edge
- Rising Edge
- 4th Rising Edge
- 16th Rising Edge

These are specified bits 7:6 for CAP2 and 5:4 for CAP2 by the register TCON1<7:4>.

This flexibility allows an interface without the need of additional hardware to change polarity or specify an input prescaler.

The control registers that are used for by Timer3 are shown in Table 2. Shaded Boxes are control bits that are not used by the Timer3 module, the Peripheral Interrupt enable and flag bits, and the Global Interrupt enable bit.

**TABLE 2: REGISTERS ASSOCIATED WITH TIMER3 AND CAPTURE**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets (Note1)
16h, Bank 3	TCON1	CA2ED1	CA2ED0	CA1ED1	CA1ED0	T16	TMR3CS	TMR2CS	TMR1CS	0000 0000	0000 0000
17h, Bank 3	TCON2	CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/PR3	TMR3ON	TMR2ON	TMR1ON	0000 0000	0000 0000
10h, Bank 2	TMR1	Timer1 register								xxxx xxxx	uuuu uuuu
11h, Bank 2	TMR2	Timer2 register								xxxx xxxx	uuuu uuuu
16h, Bank 1	PIR	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TXIF	RCIF	0000 0010	0000 0010
17h, Bank 1	PIE	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TXIE	RCIE	0000 0000	0000 0000
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	T0	PD	—	—	--11 11--	--11 qq--
14h, Bank 2	PR1	Timer1 period register								xxxx xxxx	uuuu uuuu
15h, Bank 2	PR2	Timer2 period register								xxxx xxxx	uuuu uuuu
10h, Bank 3	PW1DCL	DC1	DC0	—	—	—	—	—	—	xx-- ----	uu-- ----
11h, Bank 3	PW2DCL	DC1	DC0	TM2PW2	—	—	—	—	—	xx0- ----	uu0- ----
12h, Bank 3	PW1DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
13h, Bank 3	PW2DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented read as a '0', q - value depends on condition, shaded cells are not used by Timer1 or Timer2.

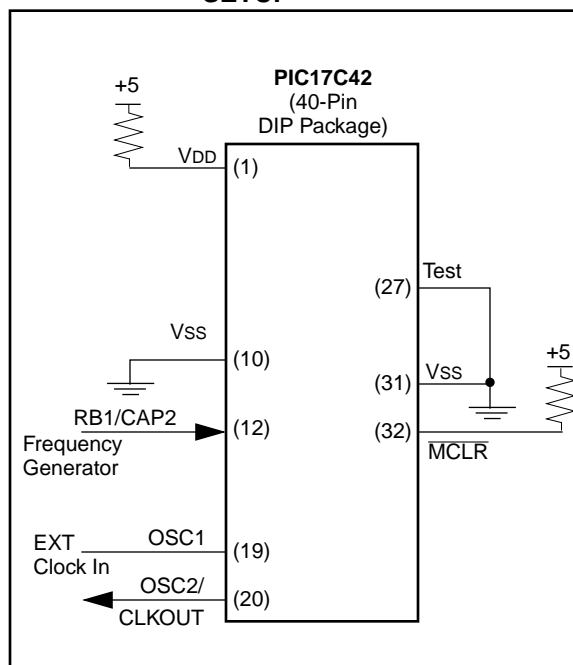
Note 1: Other (non power-up) resets include: external reset through MCLR and WDT Timer Reset.

This Application Note has 4 examples that USE the Timer3 input capture. They are:

- Frequency Counter (Period Measurement)
- Frequency Counter (Period Measurement) using a Free Running Timer
- Pulse Width Measurement
- Frequency Counter (Period Measurement) with Input Prescaler

All these examples can be run from a simple setup, this is shown in Figure 3.

**FIGURE 3: APPLICATION HARDWARE SETUP**



A discussion of each application with the operation of the software and application issues. The source listings for these are in appendices A-D.

## PERIOD MEASUREMENT (FREQUENCY COUNTER)

Period measurement is simply done by clearing the counter to 0000h, then starting the counter on the 1st rising edge. On the following rising edge, the capture2 register is loaded with the Timer3 value and the TMR3 register CA2H:CA2L is cleared. If the period is greater than the overflow rate of TMR3, the register overflows, causing an interrupt. With a TMR3 overflow, an interrupt occurs and the overflow counter may need to be incremented. The overflow counter should be incremented if:

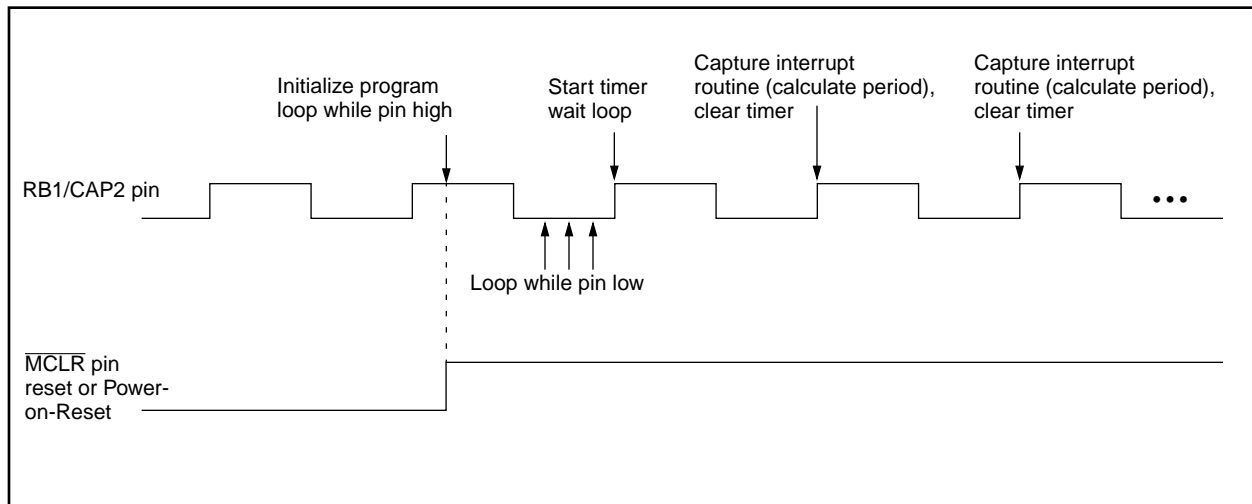
- The TMR3 overflow is the only interrupt source.
- Both the TMR3 overflow and capture2 interrupts occurred at near the same time, but the TMR3 overflow occurred first, Most Significant Byte of the Capture2 register is cleared (CA24-00h).

Once a capture has occurred, the capture registers are moved to data RAM, the capture2 interrupt flag is cleared and the TMR3 register is loaded with an offset value. This offset value is the number of cycles from the time the interrupt routine is entered to when the TMR3 register is reloaded. In this example a data RAM location is used as an overflow counter. This gives in effect a 24-bit timer. The software flow for this routine is shown in Figure 4.

The program listing in Appendix A implements this, assuming only TMR3 overflow and capture2 interrupt sources. This example may be modified to suit the particular needs of your application. The following is a performance summary for this program (@ 16 MHz):

Code size:	30 Words
RAM used:	4 Bytes
Maximum frequency that can be measured:	130 kHz
Minimum frequency that can be measured:	0.25 Hz
Measurement Accuracy:	$\pm T_{CY}$ ( $\pm 250$ ns)

**FIGURE 4: SOFTWARE TIMING FLOW RELATIVE TO INPUT SIGNAL ON RB1/CAP2 PIN**



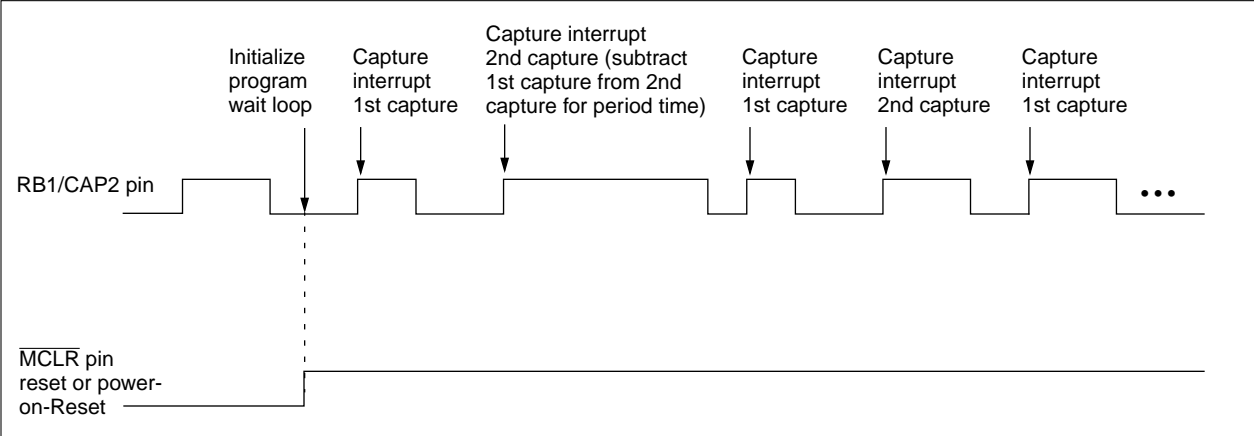
PERIOD MEASUREMENT  
(FREQUENCY COUNTER) USING A  
FREE RUNNING TIMER

In many applications the timer would need to be used for multiple tasks, and is required not to be reset (modified) by any one of these tasks. This is called a free running timer. To do period measurement in an application with a free-running timer, the program needs to store each capture in a data RAM location pair (word). The 1st capture in data RAM locations input capture2A (IC2AH:IC2AL) and the 2nd capture in data RAM locations input capture2B (IC2BH:IC2BL). Once the two captures have occurred, the values in these two words are subtracted. Since this is a free running timer, the value in input capture2B may be less than the value in input capture2A. This is if the 1st capture occurs, then the TMR3 overflows, and then the 2nd capture occurs. So an overflow counter should only be incremented if the TMR3 overflow occurs after a capture1 but before the capture2 occurs. With the use of an overflow counter this becomes an effective 24-bit period counter. The software flow for this routine is shown in Figure 5.

The program listing in Appendix B implements this, assuming only TMR3 overflow and capture2 interrupt sources. This example may be modified to suit the particular needs of your application. The following is a performance summary for this program (@ 16 MHz):

Code size:	41 Words
RAM used:	7 Bytes
Maximum frequency that can be measured:	71 kHz
Minimum frequency that can be measured:	0.25 Hz
Measurement Accuracy:	$\pm T_{CY}$ ( $\pm 250$ ns)

FIGURE 5: SOFTWARE TIMING FLOW RELATIVE TO INPUT SIGNAL ON RB1/CAP2 PIN



## PULSE WIDTH MEASUREMENT USING A FREE RUNNING TIMER

Applications that require the measurement of a pulse width can also be easily handled. The PIC17C42 can be programmed to measure either the low or the high pulse time. The software example in Appendix C measures the high pulse time. The program is initialized to capture on the rising edge of the RB1/CAP2 pin. After this event occurs, the capture mode is switched to the falling edge of the RB1/CAP2 pin. When the capture edge is modified (rising to falling, or falling to rising) a capture interrupt is generated. This "false" interrupt request must be cleared before leaving the interrupt service routine, or the program will immediately re-enter the interrupt service routine due to this "false" request. When the falling edge of the RB1/CAP2 pin occurs, the difference of the two capture values is calculated. The flow for this is shown in Figure 6.

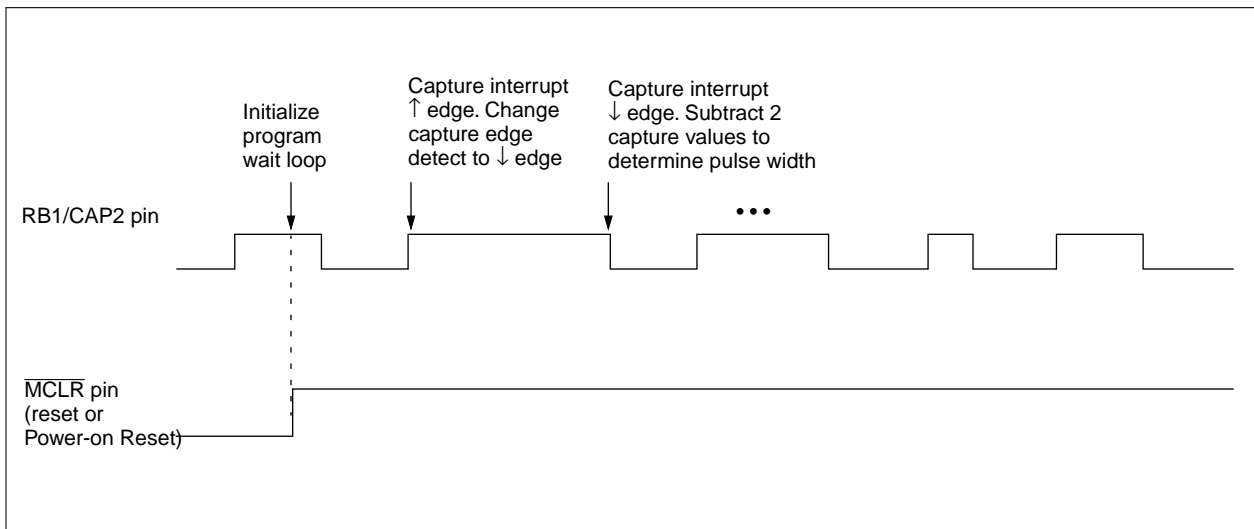
Due to the software overhead of the peripheral interrupt routine the following are the limitations on the input signal on the RB1/CAP2 pin. This does not include any software overhead that may be required in the main

routine, or if additional peripheral interrupt features need to be included. This is shown in Table 3. If you assume that the input is a square wave (high time = low time), one needs to take the worst case time of the two minimum pulse times (11  $\mu$ s) times two, to determine the period. The maximum continuous input frequency would then be approximately 45.5 kHz. For a single pulse measurement, minimum pulse width is 4.5  $\mu$ s.

The program listing in Appendix C implements this, assuming only TMR3 overflow and capture2 interrupt sources. This example may be modified to suit the particular needs of your application. The following is a performance summary for this program (@ 16 MHz):

Code size:	51 Words
RAM used:	7 Bytes
Maximum frequency that can be measured:	71 kHz
Minimum frequency that can be measured:	0.25 Hz
Measurement Accuracy:	$\pm T_{CY}$ ( $\pm 250$ ns)

**FIGURE 6: SOFTWARE TIMING FLOW RELATIVE TO INPUT SIGNAL ON RB1/CAP2**



**TABLE 3: PERIPHERAL INTERRUPT ROUTINE**

EVENT		# of Cycles	Time @ 16 MHz
1st CAPTURE	Capture1 only	18	4.5 $\mu$ s
	Capture1 and Timer Overflow	30	7.5 $\mu$ s
2nd CAPTURE	Capture only	35	8.75 $\mu$ s
	Capture and Timer Overflow	41	10.25 $\mu$ s
Minimum Pulse High	Capture1 and Timer Overflow + INT Latency	33	8.25 $\mu$ s
Minimum Pulse Low	Capture2 and Timer Overflow + INT Latency	44	11 $\mu$ s
Minimum Period (square wave)	2 • (Minimum Pulse Low)	88	22 $\mu$ s

## PERIOD MEASUREMENT (FREE RUNNING TIMER) WITH A PRESCALER

Occasionally the application may require a prescaler on the input signal. This may be due to application requirements, such as:

- Require higher resolution measurement of the input signal
- Reduce interrupt service overhead
- The input frequency is higher than interrupt service routine

The software selectable prescaler of the PIC17C42 allows the designer to easily implement this in their system without the cost of additional hardware. Care must be taken in determining if this option is appropriate. For example, if the input frequency is not stable (excessive frequency change per period) then the prescaler will give a less accurate capture value than the individual measurements.

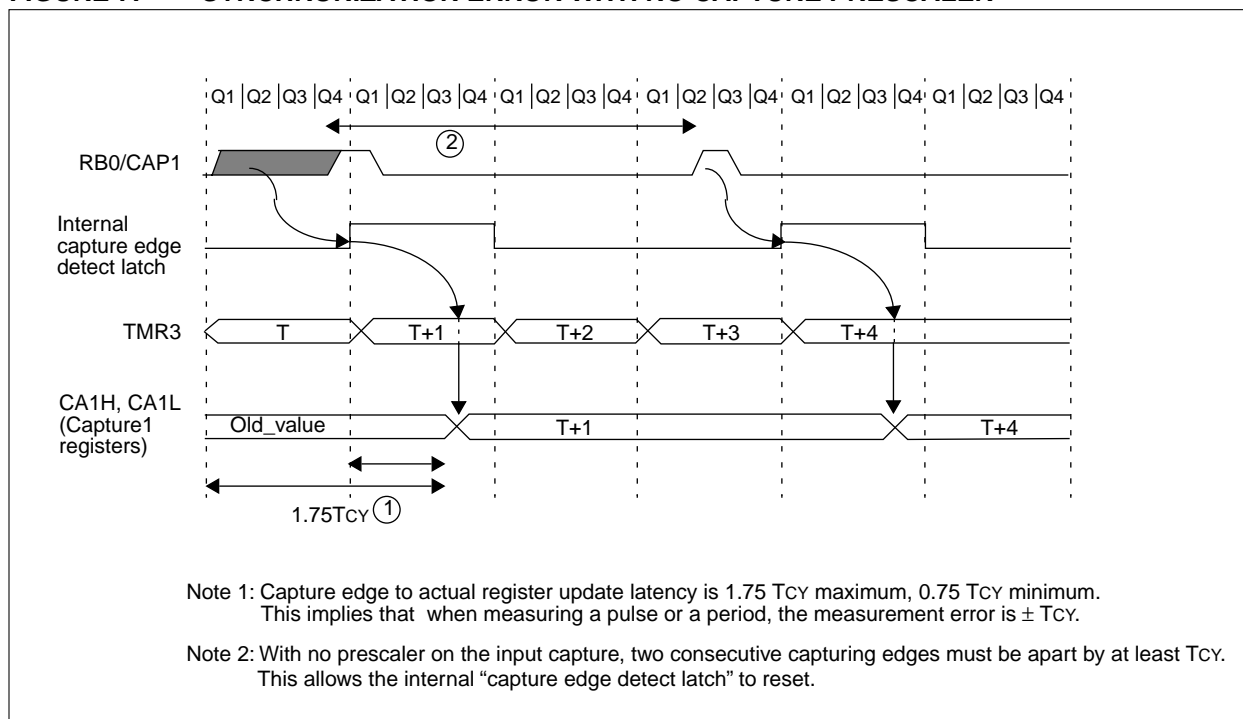
In cases where the resolution of the input frequency is important, the prescaler can be used to reduce the input capture error. There are two components to input capture:

- Resolution Error
- Input Synchronization Error

These two errors combine to form the total capture error. Resolution error is dependent on the rate at which the timer is incremented. Remember the timer may be based on an external clock (must be slower than  $T_{CY}$ ). The input synchronization error is dependent on the system clock speed ( $T_{CY}$ ), and will be less than  $T_{CY}$ .

It is easy to see that when a capture occurs the synchronization error ( $T_{ESYS}$ ) can be up to 1  $T_{CY}$  (Figure 7). This error is constant regardless of the number of edges that occur before the capture is taken. So a capture on the 1st edge gives a synchronization error per sample up to  $T_{CY}$ . While a capture taken on the 16th edge gives a synchronization error per sample only up to  $T_{CY} / 16$ , by achieving a smaller percentage of error, the captured value becomes more accurate.

**FIGURE 7: SYNCHRONIZATION ERROR WITH NO CAPTURE PRESCALER**



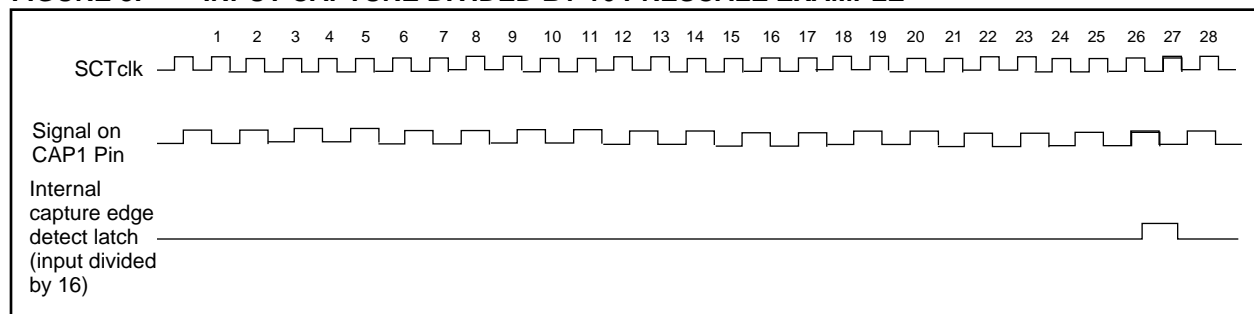


Another scenario is when the signal on the input capture pin is different than the system cycle time (T<sub>cy</sub>), for example 1.6T<sub>cy</sub>. If you tried to capture this you would have a capture value of 1. If you set the prescaler to actually capture on the 16th edge you would have  $16 * 1.6 T_{cy} = 25.6 T_{cy}$ , which would be latched on the 26th T<sub>cy</sub> (Figure 8). This 0.4 T<sub>cy</sub> error is over 16 samples, which therefore gives an effective error/sample of 0.025T<sub>cy</sub>.

The program listing in Appendix D implements this, assuming only TMR3 overflow and Capture2 interrupt sources. This example may be modified to suit the particular needs of your application. The following is a performance summary for this program (@ 16 MHz):

Code size:	41 Words
RAM used:	7 Bytes
Maximum frequency that can be measured:	80 kHz
Minimum frequency that can be measured:	0.25 Hz
Measurement Accuracy:	$\pm T_{cy}$ ( $\pm 16.625$ ns)

**FIGURE 8: INPUT CAPTURE DIVIDED BY 16 PRESCALE EXAMPLE**



Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: [www.microchip.com](http://www.microchip.com); Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

## APPENDIX A: PERIOD MEASUREMENT EXAMPLE CODE

MPASM 01.40 Released

IC\_D16\_2.ASM 1-16-1997 15:15:17

PAGE 1

```

LOC  OBJECT CODE      LINE SOURCE TEXT
VALUE

00001          LIST    P = 17C42, n = 66
00002 ;
00003 ;*****
00004 ;
00005 ; This is the basic outline for a program that can determine the
00006 ; frequency of an input, via input capture. The input capture has
00007 ; been selected to capture on the 16th rising edge. This is useful
00008 ; for high frequency inputs, where an interrupt on each rising edge
00009 ; would not be able to be serviced (at that rate). This particular
00010 ; example can support an input signal with a period of approximatly
00011 ; 625 nS. Without the divide by 16 selected, this is approximatly
00012 ; 10 us. This period time increases (frequency decreases) as the
00013 ; overhead in the main routine increases.
00014 ;
00015 ; This routine uses an 8-bit register to count the times that timer3
00016 ; overflowed. At the Max crystal frequency of 16 MHz, this gives an
00017 ; overflow time of (16)(2**8 + 1)(2**16)(250 nS) > 67.37 sec. If
00018 ; measurement of longer time intervals is required, the overflow
00019 ; counter could be extended to 16 (or more) bits.
00020 ;
00021 ; Timer 3 in this example is a free running timer. The input
00022 ; capture is generated on the RB1/CAP2 pin. There is a flag
00023 ; that specifies if this is the 1st or 2nd capture.
00024 ; The first capture is the start of the period measurement. The
00025 ; second capture value gives the end of the period. In this type
00026 ; of measurement If the 2nd capture value < the 1st captue value
00027 ; then the overflow counter should be decremented.
00028 ;
00029 ;      Program:          IC_D16_2.ASM
00030 ;      Revision Date:
00031 ;                      1-14-97      Compatibility with MPASMWIN 1.40
00032 ;
00033 ;*****
00034 ;
00035 ;
00036 ; Do the EQUate table
00037 ;
00000020      00038 IC2OF          EQU      0x20          ; T3 overflow register
00000021      00039 IC2BH          EQU      0x21          ; T3 ICA2 MSB register (2nd Cap)
00000022      00040 IC2BL          EQU      0x22          ; T3 ICA2 LSB register
00000023      00041 IC2AH          EQU      0x23          ; T3 ICB2 MSB register (1st Cap)
00000024      00042 IC2AL          EQU      0x24          ; T3 ICB2 LSB register
00000025      00043 T3OFLCNTR      EQU      0x25          ; Temperay T3 overflow register
00044 ;
00000026      00045 FLAG_REG        EQU      0x26          ; Register that has the Flag bits
00046 ;
00047 ;      FLAG_REG bit    7    6    5    4    3    2    1    0
00048 ;                  -    -    -    -    -    -    -    UFL CAP1
00049 ;      CAP1 = 0,  1st Capture
00050 ;              = 1,  2nd Capture
00051 ;
00052 ;      UFL = 0,  No Underflow
00053 ;              = 1,  Underflow during subtract
00054 ;
000007FF      00055 END_OF_PROG_MEM      EQU      0x07FF

```

```

00056 ;
00057 ;
00000004 00058 ALUSTA EQU 0x04
00000006 00059 CPUSTA EQU 0x06
00000007 00060 INTSTA EQU 0x07
0000000A 00061 W EQU 0x0A
00062 ;
00000012 00063 PORTB EQU 0x12 ; Bank 0
00064 ;
00000016 00065 PIR EQU 0x16 ; Bank 1
00000017 00066 PIE EQU 0x17
00067 ;
00000012 00068 TMR3L EQU 0x12 ; Bank 2
00000013 00069 TMR3H EQU 0x13
00000016 00070 T3PRL EQU 0x16
00000017 00071 T3PRH EQU 0x17
00072 ;
00000014 00073 CA2L EQU 0x14 ; Bank 3
00000015 00074 CA2H EQU 0x15
00000016 00075 TCON1 EQU 0x16
00000017 00076 TCON2 EQU 0x17
00077 PAGE
0000 00078 ORG 0x0000 ; Origin for the RESET vector
0000 C028 00079 GOTO START ; On reset, go to the start of
00080 ; the program
0008 00081 ORG 0x0008 ; Origin for the external RA0/INT
00082 ; interrupt vector
0008 C068 00083 GOTO EXT_INT ; Goto the ext. interrupt
00084 ; on RA0/INT routine
0010 00085 ORG 0x0010 ; Origin for the TMR0
00086 ; overflow interrupt vector
0010 C069 00087 GOTO TMR0INT ; Goto the TMR0 overflow interrupt
00088 ; routine
0018 00089 ORG 0x0018 ; Origin for the external
00090 ; RB1/T0CKI interrupt vector
0018 C06A 00091 GOTO T0INT ; Goto the ext. interrupt on
00092 ; RB1/T0CKI routine
0020 00093 ORG 0x0020 ; Origin for the interrupt vector
00094 ; of any enabled peripheral
0020 C03E 00095 GOTO PER_INT ; Goto the interrupt from a
00096 ; peripheral routine
00097 PAGE
0028 00098 ORG 0x0028 ; Origin for the top of
00099 ; program memory
0028 8406 00100 START BSF CPUSTA,4 ; Disable ALL interrupts via the
00101 ; Global Interrupt Disable
00102 ; (GLINTD) bit.
00103 ;
0029 00104 MAIN ; Place Main program here
0029 B803 00105 MOVLB 3 ; Select register Bank 3
002A 2817 00106 CLRF TCON2,0 ; Stop the timers, Single Capture
002B B0F0 00107 MOVLW 0x0F0 ; Initialize TCON1 so that
002C 0116 00108 MOVWF TCON1 ; T1 (8-bit), T2 (8-bit),
00109 ; and T3 run off the internal
00110 ; system clock. Capture2 captures
00111 ; on the 16th rising edge.
00112 ;
00113 ; Initialize Timer 3, load the timer with the number of cycles that
00114 ; the device executes (from RESET) before the timer is turned on
00115 ; Therefore the offset is required due to software overhead.
00116 ;
002D B802 00117 MOVLB 2 ; Select register Bank 2
002E 280A 00118 CLRF W,0 ; Clear the W register
002F 0126 00119 MOVWF FLAG_REG ; Initialize to 0
0030 0113 00120 MOVWF TMR3H ; Timer3 MSB = 0
0031 B000 00121 MOVLW 0x00 ; Timer3 LSB = Offset

```

```

0032 0112          00122          MOVWF    TMR3L          ;
                   00123 ;
                   00124 ; Load the Timer 3 period register with 0xFFFF, which will give an
                   00125 ; interrupt on the overflow of Timer3
                   00126 ;
0033 B0FF          00127          MOVLW    0xFF          ;
0034 0117          00128          MOVWF    T3PRH          ;
0035 0116          00129          MOVWF    T3PRL          ;
                   00130 ;
                   00131 ; the timer should be started and interrupts enabled.
                   00132 ;
0036 B803          00133          MOVLB    3              ; Select register Bank 3
0037 8217          00134          BSF      TCON2,2        ; Turn on timer 3.
0038 8307          00135          BSF      INTSTA,3       ; Turn on Peripheral Interrupts
0039 B801          00136          MOVLB    1              ; Select register Bank 1
003A B048          00137          MOVLW    0x48          ; Enable Capture 2 and Timer3
003B 0117          00138          MOVWF    PIE           ; Interrupts (when GLINTD = 0)
                   00139 ;
                   00140 ; This is where you would do the things you wanted to do.
                   00141 ; this example will only loop waiting for the interrupts.
                   00142 ;
003C 8C06          00143 WAIT      BCF      CPUSTA,4      ; Enable ALL interrupts
003D C03C          00144          GOTO     WAIT           ; Loop here waiting for a timer
                   00145 ; Interrupt
                   00146 PAGE
                   00147 ;
                   00148 ; The interrupt routine for any peripheral interrupt, This routine
                   00149 ; only deals with Timer3 (T3) interrupts.
                   00150 ;
                   00151 ; Time required to execute interrupt routine. Not including
                   00152 ; interrupt latency (time to enter into the interrupt routine)
                   00153 ;
                   00154 ;      case1 - only T3 overflow          = 12 cycles
                   00155 ;      case2 - 1st capture          = 14 cycles
                   00156 ;      case3 - 2nd capture          = 30 cycles
                   00157 ;      case4 - T3 overflow and 1st capture = 34 cycles
                   00158 ;      case5 - T3 overflow and 2nd capture = 50 cycles
                   00159 ;
                   00160 ;
003E B801          00161 PER_INT   MOVLB    1              ; Select register Bank 1
003F 9E16          00162          BTFSC    PIR,6          ; Did T3 overflow?
                   00163 ; If not skip next Instruction
                   00164          GOTO     T3OVFL          ; Inc overflow cntr and clear flag
0040 C055          00165 CK_CAP    BTFSS    PIR,3          ; Did the RB1/CAP2 pin cause an
0041 9316          00166          ; interrupt?
0042 0005          00167          RETFIE          ; No RB1/CAP2 interrupt,
                   00168 ; Return from Interrupt
                   00169 ;
                   00170 ; This portion of the code takes the 1st capture and stores its
                   00171 ; value in register pair IC2AH:IC2AL. When the 2nd capture
                   00172 ; is take, its value is stored in register pair IC2BH:IC2BL.
                   00173 ; A 16-bit subtract is performed, with the final 24-bit result
                   00174 ; being stored in IC20F:IC2BH:IC2BL. This value will no longer
                   00175 ; be correct after the next capture occurs (IC2BH:IC2BL will
                   00176 ; change), so the main routine must utilize this value before
                   00177 ; it changes.
                   00178 ;
0043 8B16          00179 CAPTURE   BCF      PIR,3          ; Clear Capture2 interrupt flag
0044 B803          00180          MOVLB    3              ; Select register Bank 3
0045 9826          00181          BTFSC    FLAG_REG,0     ; 1st or 2nd capture2?
0046 C04B          00182          GOTO     CAP2           ; It was the 2nd Capture
0047 5424          00183 CAP1      MOVFPF    CA2L,IC2AL    ; Move the captured value to
0048 5523          00184          MOVFPF    CA2H,IC2AH    ; temporary registers
0049 8026          00185          BSF      FLAG_REG,0     ; Have 1st capture2
004A 0005          00186          RETFIE          ; Return from Interrupt
                   00187 ;

```

```

00188 PAGE
004B 5422 00189 CAP2      MOVFP  CA2L,IC2BL ; Move the captured value to
004C 5521 00190      MOVFP  CA2H,IC2BH ;   temporary registers
                                ;   (to prevent being overwritten)
                                ;
004D E061 00193      CALL   SUB16   ; Call routine which subtracts
                                ;   2 16-bit numbers.
004E 9926 00195      BTFSC  FLAG_REG,1 ; Underflow during SUB16?
004F 0725 00196      DECF   T3OFLCNTR,1 ; Since underflow, decrement the
                                ;   overflow counter.
0050 2926 00198      CLRF   FLAG_REG,1 ; Clear the flag bits for
                                ;   underflow and 2nd capture2
0051 6A25 00200      MOVFP  T3OFLCNTR,W ; Store the T3 input capture
0052 4A20 00201      MOVFP  W,IC2OF   ;   overflow value in IC2OF
0053 2825 00202      CLRF   T3OFLCNTR,0 ; Clear the Data register which
                                ;   counts how many times Timer 3
                                ;   overflows.
0054 0005 00205      RETFIE                ; Return from interrupt
                                ;
00206 ;
00207 ; When Timer 3 has overflowed, the overflow counter only should
00208 ; be incremented when the overflow occurs after a capture 1
00209 ; but before the capture 2. The 4 possible cases when entering
00210 ; the T3OVFL section of the PER_INT routine are as follows:
00211 ; Case 1: T3 overflow (only) and FLAG_REG.0 = 0 (waiting
00212 ;         for Capture 1 to occur). Do Not increment counter
00213 ; Case 2: T3 overflow (only) and FLAG_REG.0 = 1 (waiting
00214 ;         for Capture 2 to occur). Increment counter
00215 ; Case 3: T3 Overflow happened after Capture. Do Not
00216 ;         increment overflow counter
00217 ; Case 4: T3 Overflow occurred before Capture 2 and FLAG_REG.0 = 1
00218 ;         (waiting for Capture 2 to occur). Increment counter
00219 ;
00220 ;
0055 8E16 00221 T3OVFL    BCF   PIR,6      ; Clear Overflow interrupt flag
0056 9316 00222      BTFSS  PIR,3      ; Did the RBl/CAP2 pin also
                                ;   cause an interrupt?
0057 C05E 00224      GOTO   FR0       ; No, Check if between 1st
                                ;   and 2nd capture
0058 B803 00226      MOVLB   3        ; Bank 3
0059 280A 00227      CLRF   W,0       ; W = 0
005A 3115 00228      CPFSEQ  CA2H     ; if CA2H = 0, overflow happened
005B C05E 00229      GOTO   FR0       ;   first, must check FLAG_REG
                                ;   bit 0
005C B801 00231      MOVLB   1        ; Back to bank 1
005D C043 00232      GOTO   CAPTURE   ; Capture happened first, do NOT
                                ;   Increment overflow counter
                                ;   and do capture routine
005E 9826 00235 FR0      BTFSC  FLAG_REG,0 ; Between Capture 1 and Capture 2?
005F 1525 00236      INCF   T3OFLCNTR,1 ; Yes, Inc. the overflow counter
0060 0005 00237      RETFIE                ; Return from overflow interrupt
                                ;
00238 ;
0061 6A24 00239 SUB16    MOVFP  IC2AL,W   ; Do the 16-bit subtraction
0062 0522 00240      SUBWF  IC2BL,1      ;
0063 6A23 00241      MOVFP  IC2AH,W     ;
0064 0321 00242      SUBWFB IC2BH,1      ;
0065 9004 00243      BTFSS  ALUSTA,0    ; Is the result pos. or neg. ?
0066 8126 00244      BSF    FLAG_REG,1 ; neg., Set the underflow flag
0067 0002 00245      RETURN                ; Return from the subroutine
00246 PAGE
00247 ;
00248 ; Other Interrupt routines. (Not utilized in this example)
00249 ;
0068 0005 00250 EXT_INT RETFIE                ; RA0/INT interrupt routine
                                ;   (NOT used in this program)
0069 0005 00252 TMR0INT RETFIE                ; TMR0 overflow interrupt routine
                                ;   (NOT used in this program)
00253

```

# AN545

---

```
006A 0005      00254 T0INT  RETFIE                      ; RA1/T0CKI interrupt routine
                00255                      ; (NOT used in this program)
                00256                      ;
006B C028      00257 SRESET GOTO    START                ; If program became lost, goto
                00258                      ; START and reinitialize.
                00259 ;
                00260 ;
                00261 ; When the executed address is NOT in the program range, the
                00262 ; 16-bit address should contain all 1's (a CALL 0x1FFF). At
                00263 ; this location you could branch to a routine to recover or
                00264 ; shut down from the invalid program execution.
                00265 ;
007FF          00266                      ORG            END_OF_PROG_MEM ;
007FF C06B     00267                      GOTO    SRESET    ; The program has lost it's mind,
                00268                      ; do a system reset
                00269                      END
```

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

```
0000 : X-----X----- X-----X----- X-----XXXXXXXXXXXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXX-----
07C0 : -----X-----X-----X-----X-----X-----X-----X-----X-----X
```

All other memory blocks unused.

Program Memory Words Used: 74

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

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: [www.microchip.com](http://www.microchip.com); Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

## APPENDIX B: PERIOD MEASUREMENT, FREE RUNNING TIMER EXAMPLE CODE

MPASM 01.40 Released

IC\_FRT2.ASM 1-16-1997 15:15:48

PAGE 1

```

LOC  OBJECT CODE      LINE SOURCE TEXT
VALUE

00001          LIST    P = 17C42, n = 66
00002 ;
00003 ;*****
00004 ;
00005 ; This is the basic outline for a program that can determine the
00006 ; frequency of an input, via input capture. This routine uses an
00007 ; 8-bit register to count the times that timer3 overflowed. At the
00008 ; Max crystal frequency of 16 MHz, this gives an overflow time of
00009 ; (2*16)(256 + 1)(250 nS) > 4.21 sec or a frequency < 0.25 Hz. If
00010 ; measurement of longer time intervals is required, the overflow
00011 ; counter could be extended to 16 (or more) bits.
00012 ;
00013 ; Timer 3 in this example is a free running timer. The input
00014 ; capture is generated on the RB1/CAP2 pin. There is a flag
00015 ; that specifies if this is the 1st or 2nd capture.
00016 ; The first capture is the start of the period measurement. The
00017 ; second capture value gives the end of the period. In this type
00018 ; of measurement If the 2nd capture value < the 1st capture value
00019 ; then the overflow counter should be decremented.
00020 ;
00021 ; Do the EQUate table
00022 ;
00000020      00023 IC2OF          EQU      0x20      ; T3 overflow register
00000021      00024 IC2BH          EQU      0x21      ; T3 ICA2 MSB register (2nd Cap)
00000022      00025 IC2BL          EQU      0x22      ; T3 ICA2 LSB register
00000023      00026 IC2AH          EQU      0x23      ; T3 ICB2 MSB register (1st Cap)
00000024      00027 IC2AL          EQU      0x24      ; T3 ICB2 LSB register
00000025      00028 T3OFLCNTR     EQU      0x25      ; Temperay T3 overflow register
00029 ;
00000026      00030 FLAG_REG       EQU      0x26      ; Register that has the Flag bits
00031 ;
00032 ;      FLAG_REG bit   7   6   5   4   3   2   1   0
00033 ;                -   -   -   -   -   -   -   UFL CAP1
00034 ;      CAP1 = 0,  1st Capture
00035 ;                = 1,  2nd Capture
00036 ;
00037 ;      UFL = 0,  No Underflow
00038 ;                = 1,  Underflow during subtract
00039 ;
00040 ;      Program:          IC_FRT2.ASM
00041 ;      Revision Date:
00042 ;                      1-14-97  Compatibility with MPASMWIN 1.40
00043 ;
00044 ;*****
00045 ;
00046 ;
000007FF      00047 END_OF_PROG_MEM EQU 0x07FF
00048 ;
00049 ;
00000004      00050 ALUSTA          EQU      0x04
00000006      00051 CPUSTA          EQU      0x06
00000007      00052 INTSTA          EQU      0x07
0000000A      00053 W              EQU      0x0A

```

```

00000012      00054 ;
00000012      00055 PORTB      EQU      0x12      ; Bank 0
00000012      00056 ;
00000016      00057 PIR        EQU      0x16      ; Bank 1
00000017      00058 PIE        EQU      0x17
00000017      00059 ;
00000012      00060 TMR3L      EQU      0x12      ; Bank 2
00000013      00061 TMR3H      EQU      0x13
00000016      00062 T3PRL      EQU      0x16
00000017      00063 T3PRH      EQU      0x17
00000017      00064 ;
00000014      00065 CA2L       EQU      0x14      ; Bank 3
00000015      00066 CA2H       EQU      0x15
00000016      00067 TCON1      EQU      0x16
00000017      00068 TCON2      EQU      0x17
00000017      00069 PAGE
0000      00070
0000 C028      00071          ORG      0x0000      ; Origin for the RESET vector
0000      00072          GOTO     START          ; On reset, go to the start of
0000      00073                      ; the program
0008      00074          ORG      0x0008      ; Origin for the external RA0/INT
0008      00075          GOTO     EXT_INT        ; Goto the ext. interrupt
0008 C068      00076                      ; on RA0/INT routine
0010      00077          ORG      0x0010      ; Origin for the TMR0
0010      00078          GOTO     TMR0INT        ; Goto the TMR0 overflow interrupt
0010 C069      00079                      ; routine
0018      00080          ORG      0x0018      ; Origin for the external
0018      00081          GOTO     T0INT          ; Goto the ext. interrupt on
0018 C06A      00082                      ; RB1/RT interrupt vector
0020      00083          ORG      0x0020      ; Origin for the interrupt vector
0020      00084          GOTO     PER_INT        ; Goto the interrupt from a
0020 C03E      00085                      ; peripheral routine
0020      00086          ORG      0x0028      ; Origin for the top of
0020      00087          GOTO     CPUSTA,4        ; Disable ALL interrupts via the
0020      00088          ORG      0x0028      ; Global Interrupt Disable
0020      00089          ORG      0x0028      ; (GLINTD) bit.
0028      00090          ORG      0x0028      ;
0028      00091          ORG      0x0028      ;
0028 8406      00092 START      BSF      CPUSTA,4
0028      00093          ORG      0x0028      ;
0028      00094          ORG      0x0028      ;
0028      00095          ORG      0x0028      ;
0029      00096 MAIN          ORG      0x0028      ; Place Main program here
0029 B803      00097          MOVLB   3          ; Select register Bank 3
002A 2817      00098          CLRFB   TCON2,0    ; Stop the timers, Single Capture
002B B070      00099          MOVWF  TCON1        ; Initialize TCON1 so that
002C 0116      00100          MOVWF  TCON1        ; T1 (8-bit), T2 (8-bit),
002C      00101                      ; and T3 runs off the internal
002C      00102                      ; system clock. Capture2
002C      00103                      ; captures on the rising edge.
002C      00104 ;
002C      00105 ; Initialize Timer 3, load the timer with the number of cycles that
002C      00106 ; the device executes (from RESET) before the timer is turned on
002C      00107 ; Therefore the offset is required due to software overhead.
002C      00108 ;
002D B802      00109          MOVLB   2          ; Select register Bank 2
002E 280A      00110          CLRFB   W,0        ; Clear the W register
002F 0126      00111          MOVWF  FLAG_REG    ; Initialize to 0
0030 0113      00112          MOVWF  TMR3H      ; Timer3 MSB = 0
0031 B013      00113          MOVWF  TMR3L      ; Timer3 LSB = Offset
0032 0112      00114          MOVWF  TMR3L      ;
0032      00115 ;
0032      00116 ; Load the Timer 3 period register with 0xFFFF, which will give an
0032      00117 ; interrupt on the overflow of Timer3
0032      00118 ;
0033 B0FF      00119          MOVWF  0xFF      ;

```



```

0034 0117      00120      MOVWF    T3PRH      ;
0035 0116      00121      MOVWF    T3PRL      ;
                00122      ;
                00123      ; the timer should be started and interrupts enabled.
                00124      ;
0036 B803      00125      MOVLB     3          ; Select register Bank 3
0037 8217      00126      BSF       TCON2,2    ; Turn on timer 3.
0038 8307      00127      BSF       INTSTA,3   ; Turn on Peripheral Interrupts
0039 B801      00128      MOVLB     1          ; Select register Bank 1
003A B048      00129      MOVLW     0x48       ; Enable Capture 2 and Timer3
003B 0117      00130      MOVWF    PIE        ; Interrupts (when GLINTD = 0)
                00131      ;
                00132      ; This is where you would do the things you wanted to do.
                00133      ; this example will only loop waiting for the interrupts.
                00134      ;
003C 8C06      00135 WAIT      BCF     CPUSTA,4  ; Enable ALL interrupts
003D C03C      00136      GOTO     WAIT        ; Loop here waiting for a timer
                00137      ; Interrupt
                00138 PAGE
                00139      ;
                00140      ; The interrupt routine for any peripheral interrupt, This routine
                00141      ; only deals with Timer3 (T3) interrupts.
                00142      ;
                00143      ; Time required to execute interrupt routine. Not including
                00144      ; interrupt latency (time to enter into the interrupt routine)
                00145      ;
                00146      ;      case1 - only T3 overflow          = 12 cycles
                00147      ;      case2 - 1st capture             = 14 cycles
                00148      ;      case3 - 2nd capture             = 30 cycles
                00149      ;      case4 - T3 overflow and 1st capture = 34 cycles
                00150      ;      case5 - T3 overflow and 2nd capture = 50 cycles
                00151      ;
                00152      ;
003E B801      00153 PER_INT    MOVLB     1          ; Select register Bank 1
003F 9E16      00154      BTFSF    PIR,6       ; Did T3 overflow?
                00155      ; If not skip next Instruction
                00156      GOTO     T3OVFL      ; Inc overflow cntr and clear flag
0040 C055      00157 CK_CAP     BTFSF    PIR,3   ; Did the RB1/CAP2 pin cause an
                00158      ; interrupt?
0042 0005      00159      RETFIE           ; No RB1/CAP2 interrupt,
                00160      ; Return from Interrupt
                00161      ;
                00162      ; This portion of the code takes the 1st capture and stores its
                00163      ; value in register pair IC2AH:IC2AL. When the 2nd capture
                00164      ; is taken, its value is stored in register pair IC2BH:IC2BL.
                00165      ; A 16-bit subtract is performed, with the final 24-bit result
                00166      ; being stored in IC20F:IC2BH:IC2BL. This value will no longer
                00167      ; be correct after the next capture occurs (IC2BH:IC2BL will
                00168      ; change), so the main routine must utilize this value before
                00169      ; it changes.
                00170      ;
0043 8B16      00171 CAPTURE    BCF     PIR,3    ; Clear Capture2 interrupt flag
0044 B803      00172      MOVLB     3          ; Select register Bank 3
0045 9826      00173      BTFSF    FLAG_REG,0  ; 1st or 2nd capture2?
0046 C04B      00174      GOTO     CAP2        ; It was the 2nd Capture
0047 5424      00175 CAP1      MOVFPF   CA2L,IC2AL ; Move the captured value to
0048 5523      00176      MOVFPF   CA2H,IC2AH ; temporary registers
0049 8026      00177      BSF       FLAG_REG,0  ; Have 1st capture2
004A 0005      00178      RETFIE           ; Return from Interrupt
                00179      ;
                00180 PAGE
004B 5422      00181 CAP2      MOVFPF   CA2L,IC2BL ; Move the captured value to
004C 5521      00182      MOVFPF   CA2H,IC2BH ; temporary registers
                00183      ; (to prevent being overwritten)
                00184      ;
004D E061      00185      CALL     SUB16      ; Call routine which subtracts

```

```

00186                                ; 2 16-bit numbers.
004E 9926      00187      BTFSC    FLAG_REG,1 ; Underflow during SUB16?
004F 0725      00188      DECF     T3OFLCNTR,1 ; Since underflow, decrement the
                                ; overflow counter.
0050 2926      00190      CLRF     FLAG_REG,1 ; Clear the flag bits for
                                ; underflow and 2nd capture2
0051 6A25      00192      MOVFP    T3OFLCNTR,W ; Store the T3 input capture
0052 4A20      00193      MOVFP    W,IC2OF ; overflow value in IC2OF
0053 2825      00194      CLRF     T3OFLCNTR,0 ; Clear the Data register which
                                ; counts how many times Timer 3
                                ; overflows.
0054 0005      00197      RETFIE    ; Return from interrupt
                                00198 ;
                                00199 ; When Timer 3 has overflowed, the overflow counter only should
                                00200 ; be incremented when the overflow occurs after a capture 1
                                00201 ; but before the capture 2. The 4 possible cases when entering
                                00202 ; the T3OVFL section of the PER_INT routine are as follows:
                                00203 ; Case 1: T3 overflow (only) and FLAG_REG.0 = 0 (waiting
                                00204 ; for Capture 1 to occur). Do Not increment counter
                                00205 ; Case 2: T3 overflow (only) and FLAG_REG.0 = 1 (waiting
                                00206 ; for Capture 2 to occur). Increment counter
                                00207 ; Case 3: T3 Overflow happened after Capture. Do Not
                                00208 ; increment overflow counter
                                00209 ; Case 4: T3 Overflow occurred before Capture 2 and FLAG_REG.0 = 1
                                00210 ; (waiting for Capture 2 to occur). Increment counter
                                00211
                                00212 ;
0055 8E16      00213 T3OVFL      BCF     PIR,6 ; Clear Overflow interrupt flag
0056 9316      00214      BTFSS    PIR,3 ; Did the RB1/CAP2 pin also
                                ; cause an interrupt?
0057 C05E      00216      GOTO     FR0 ; No, Check if between 1st
                                ; and 2nd capture
0058 B803      00218      MOVLB    3 ; Bank 3
0059 280A      00219      CLRF     W,0 ; W = 0
005A 3115      00220      CPFSEQ   CA2H ; if CA2H = 0, overflow happened
005B C05E      00221      GOTO     FR0 ; first, must check FLAG_REG
                                ; bit 0
005C B801      00223      MOVLB    1 ; Back to bank 1
005D C043      00224      GOTO     CAPTURE ; Capture happened first, do NOT
                                ; Increment overflow counter
                                ; and do capture routine
005E 9826      00227 FR0      BTFSC    FLAG_REG,0 ; Between Capture 1 and Capture 2?
005F 1525      00228      INCF     T3OFLCNTR,1 ; Yes, Inc. the overflow counter
0060 0005      00229      RETFIE    ; Return from overflow interrupt
                                00230 ;
0061 6A24      00231 SUB16      MOVFP    IC2AL,W ; Do the 16-bit subtraction
0062 0522      00232      SUBWF    IC2BL,1 ;
0063 6A23      00233      MOVFP    IC2AH,W ;
0064 0321      00234      SUBWFB   IC2BH,1 ;
0065 9004      00235      BTFSS    ALUSTA,0 ; Is the result pos. or neg. ?
0066 8126      00236      BSF      FLAG_REG,1 ; neg., Set the underflow flag
0067 0002      00237      RETURN    ; Return from the subroutine
                                00238 PAGE
                                00239 ;
                                00240 ; Other Interrupt routines. (Not utilized in this example)
                                00241 ;
0068 0005      00242 EXT_INT RETFIE ; RA0/INT interrupt routine
                                ; (NOT used in this program)
0069 0005      00244 TMR0INT RETFIE ; TMR0 overflow interrupt routine
                                ; (NOT used in this program)
006A 0005      00246 T0INT RETFIE ; RB1/RT interrupt routine
                                ; (NOT used in this program)
                                00247 ;
                                00248 ;
006B C028      00249 SRESET GOTO START ; If program became lost, goto
                                ; START and reinitialize.
                                00250
                                00251 ;

```

```

00252 ;
00253 ; When the executed address is NOT in the program range, the
00254 ; 16-bit address should contain all 1's (a CALL 0x1FFF). At
00255 ; this location you could branch to a routine to recover or
00256 ; shut down from the invalid program execution.
00257 ;
07FF      00258          ORG          END_OF_PROG_MEM ;
07FF C06B 00259          GOTO      SRESET      ; The program has lost it's mind,
00260          ; do a system reset
00261          END
MEMORY USAGE MAP ('X' = Used, '-' = Unused)

0000 : X-----X----- X-----X----- X-----XXXXXXXXXXXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXX-----
07C0 : -----X

```

All other memory blocks unused.

Program Memory Words Used: 74

```

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

```

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: [www.microchip.com](http://www.microchip.com); Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

## APPENDIX C: PULSE WIDTH MEASUREMENT EXAMPLE CODE

MPASM 01.40 Released

PW02.ASM 1-16-1997 15:16:15

PAGE 1

```

LOC  OBJECT CODE      LINE SOURCE TEXT
VALUE

00001          LIST    P = 17C42, n = 66
00002 ;
00003 ;*****
00004 ;
00005 ; This is the basic outline for a program that can determine the
00006 ; Pulse Width of an input, via input capture. This routine uses an
00007 ; 8-bit register to count the times that timer3 overflowed. At the
00008 ; Max crystal frequency of 16 MHz, this gives an overflow time of
00009 ; (2**16)(256 + 1)(250 nS) > 4.21 sec or a frequency < 0.25 Hz. If
00010 ; measurement of longer time intervals is required, the overflow
00011 ; counter could be extended to 16 (or more) bits.
00012 ;
00013 ; Do the EQUate table
00014 ;
00000020      00015 IC2OF      EQU    0x20      ; T3 overflow register
00000021      00016 IC2BH      EQU    0x21      ; T3 ICA2 MSB register (2nd Cap)
00000022      00017 IC2BL      EQU    0x22      ; T3 ICA2 LSB register
00000023      00018 IC2AH      EQU    0x23      ; T3 ICB2 MSB register (1st Cap)
00000024      00019 IC2AL      EQU    0x24      ; T3 ICB2 LSB register
00000025      00020 T3OFLCNTR EQU    0x25      ; Temperature T3 overflow register
00021 ;
00000026      00022 FLAG_REG    EQU 0x26      ; Register that has the Flag bits
00023 ;
00024 ;   FLAG_REG bit   7   6   5   4   3   2   1   0
00025 ;           -   -   -   -   -   -   -   UFL CAP1
00026 ;   CAP1 = 0,  1st Capture
00027 ;           = 1,  2nd Capture
00028 ;
00029 ;   UFL = 0,  No Underflow
00030 ;           = 1,  Underflow during subtract
00031 ;
00032 ;   Program:          PW02.ASM
00033 ;   Revision Date:
00034 ;                   1-14-97   Compatibility with MPASMWIN 1.40
00035 ;
00036 ;*****
00037 ;
00038 ;
000007FF      00039 END_OF_PROG_MEM EQU    0x07FF
00040 ;
00000004      00041 ALUSTA      EQU    0x04
00000006      00042 CPUSTA      EQU    0x06
00000007      00043 INTSTA      EQU    0x07
0000000A      00044 W          EQU    0x0A
00045 ;
00000012      00046 PORTB      EQU    0x12      ; Bank 0
00047 ;
00000016      00048 PIR        EQU    0x16      ; Bank 1
00000017      00049 PIE        EQU    0x17
00050 ;
00000012      00051 TMR3L      EQU    0x12      ; Bank 2
00000013      00052 TMR3H      EQU    0x13
00000016      00053 T3PRL      EQU    0x16
00000017      00054 T3PRH      EQU    0x17

```

```

00000014      00055      ;
00000015      00056 CA2L      EQU      0x14      ; Bank 3
00000016      00057 CA2H      EQU      0x15
00000017      00058 TCON1     EQU      0x16
00000017      00059 TCON2     EQU      0x17
00060 PAGE
00061 ;
0000      00062      ORG      0x0000      ; Origin for the RESET vector
0000 C028      00063      GOTO     START      ; On reset, go to the start of
00064      ; the program
0008      00065      ORG      0x0008      ; Origin for the external RA0/INT
00066      ; interrupt vector
0008 C072      00067      GOTO     EXT_INT     ; Goto the ext. interrupt
00068      ; on RA0/INT routine
0010      00069      ORG      0x0010      ; Origin for the TMR0
00070      ; overflow interrupt vector
0010 C073      00071      GOTO     TMR0INT     ; Goto the TMR0 overflow interrupt
00072      ; routine
0018      00073      ORG      0x0018      ; Origin for the external
00074      ; RA1/T0CKI interrupt vector
0018 C074      00075      GOTO     T0CKI_INT    ; Goto the ext. interrupt on
00076      ; RA1/T0CKI routine
0020      00077      ORG      0x0020      ; Origin for the interrupt vector
00078      ; of any enabled peripheral
0020 C03E      00079      GOTO     PER_INT     ; Goto the interrupt from a
00080      ; peripheral routine
0028      00081      ORG      0x0028      ; Origin for the top of
00082      ; program memory
00083 PAGE
0028 8406      00084 START     BSF      CPUSTA,4      ; Disable ALL interrupts via the
00085      ; Global Interrupt Disable
00086      ; (GLINTD) bit.
00087      ;
0029      00088 MAIN      ; Place Main program here
0029 B803      00089      MOVLB     3      ; Select register Bank 3
002A 2817      00090      CLRWF     TCON2,0      ; Stop the timers, Single Capture
002B B070      00091      MOVLW     0x070      ; Initialize TCON1 so that
002C 0116      00092      MOVWF     TCON1      ; T1 (8-bit), T2 (8-bit),
00093      ; and T3 run off the internal
00094      ; system clock. Capture2
00095      ; captures on the rising edge.
00096 ;
00097 ; Initialize Timer 3, load the timer with the number of cycles that
00098 ; the device executes (from RESET) before the timer is turned on
00099 ; Therefore the offset is required due to software overhead.
00100 ;
002D B802      00101      MOVLB     2      ; Select register Bank 2
002E 280A      00102      CLRWF     W,0      ; Clear the W register
002F 0126      00103      MOVWF     FLAG_REG    ; Initialize to 0
0030 0113      00104      MOVWF     TMR3H      ; Timer3 MSB = 0
0031 B013      00105      MOVLW     0x13      ; Timer3 LSB = Offset
0032 0112      00106      MOVWF     TMR3L      ;
00107 ;
00108 ; Load the Timer 3 period register with 0xFFFF, which will give an
00109 ; interrupt on the overflow of Timer3
00110 ;
0033 B0FF      00111      MOVLW     0xFF      ;
0034 0117      00112      MOVWF     T3PRH      ;
0035 0116      00113      MOVWF     T3PRL      ;
00114 ;
00115 ; the timer should be started and interrupts enabled.
00116 ;
0036 B803      00117      MOVLB     3      ; Select register Bank 3
0037 8217      0011      BSF      TCON2,2      ; Turn on timer 3.
0038 8307      00119      BSF      INTSTA,3      ; Turn on Peripheral Interrupts
0039 B801      00120      MOVLB     1      ; Select register Bank 1

```

```

003A B048      00121      MOVLW    0x48      ; Enable Capture 2 and Timer3
003B 0117      00122      MOVWF    PIE        ; Interrupts (when GLINTD = 0)
               00123      ;
               00124      ; This is where you would do the things you wanted to do.
               00125      ; this example will only loop waiting for the interrupts.
               00126      ;
003C 8C06      00127 WAIT      BCF      CPUSTA,4 ; Enable ALL interrupts
003D C03C      00128      GOTO     WAIT      ; Loop here waiting for a timer
               00129      ; Interrupt
               00130      PAGE
               00131      ;
               00132      ; The interrupt routine for any peripheral interrupt, This routine
               00133      ; only deals with Timer3 (T3) interrupts.
               00134      ;
               00135      ; Time required to execute interrupt routine. Not including
               00136      ; interrupt latency (time to enter into the interrupt routine)
               00137      ;
               00138      ;      case1 - only T3 overflow          = 12 cycles
               00139      ;      case2 - 1st capture              = 20 cycles
               00140      ;      case3 - 2nd capture              = 34 cycles
               00141      ;      case4 - T3 overflow and 1st capture = 32 cycles
               00142      ;      case5 - T3 overflow and 2nd capture = 44 cycles
               00143      ;
               00144      ;
003E B801      00145 PER_INT    MOVLB    1        ; Select register Bank 1
003F 9E16      00146      BTFSC    PIR,6      ; Did T3 overflow?
               00147      ; If not skip next Instruction
               00148      GOTO     T3OVFL     ; Inc overflow cntr and clear flag
0040 C05A      00149 CK_CAP     BTFSS    PIR,3      ; Did the RB1/CAP2 pin cause an
0041 9316      00150      ; interrupt?
0042 0005      00151      RETFIE   ; No RB1/CAP2 interrupt,
               00152      ; Return from Interrupt
               00153      ;
               00154      ;
               00155      ; This portion of the code takes the 1st capture and stores its
               00156      ; value in register pair IC2AH:IC2AL. When the 2nd capture
               00157      ; is take, its value is stored in register pair IC2BH:IC2BL.
               00158      ; A 16-bit subtract is performed, with the final 24-bit result
               00159      ; being stored in IC2OF:IC2BH:IC2BL. This value will no longer
               00160      ; be correct after the next capture occurs (IC2BH:IC2BL will
               00161      ; change), so the main routine must utilize this value before
               00162      ; it changes.
               00163      ;
0043 B803      00164 CAPTURE    MOVLB    3        ; Select register Bank 3
0044 9826      00165      BTFSC    FLAG_REG,0 ; Capture on rising or falling edge?
0045 C04B      00166      GOTO     FALLING    ; It was the 2nd Capture
0046 5424      00167 RISING     MOVPF    CA2L,IC2AL ; Move the captured value to
0047 5523      00168      MOVPF    CA2H,IC2AH ; temporary registers
0048 8026      00169      BSF      FLAG_REG,0 ; Set flag for 1st capture
0049 8E16      00170      BCF      TCON1,6    ; Change edge from rising
               00171      ; to falling
004A C055      00172      GOTO     FALSE_C    ;** With the changing of the capture
               00173      ;** edge, we have a false capture
               00174      ;
               00175      PAGE
004B 5422      00176 FALLING     MOVPF    CA2L,IC2BL ; Move the captured value to
004C 5521      00177      MOVPF    CA2H,IC2BH ; temporary registers
               00178      ; (to prevent being overwritten)
               00179      ;
004D E06B      00180      CALL     SUB16     ; Call routine which subtracts
               00181      ; 2 16-bit numbers.
004E 9926      00182      BTFSC    FLAG_REG,1 ; Underflow during SUB16?
004F 0725      00183      DECF     T3OFLCNTR,1 ; Since underflow, decrement the
               00184      ; overflow counter.
0050 2926      00185      CLR     FLAG_REG,1 ; Clear the flag bits for
               00186      ; underflow and 2nd capture2

```

```

0051 6A25      00187      MOVFP   T3OFLCNTR,W; Store the T3 input capture
0052 4A20      00188      MOVFP   W,IC2OF   ; overflow value in IC2OF
0053 2825      00189      CLRFP   T3OFLCNTR,0; Clear the Data register which
                                ; counts how many times Timer 3
                                ; overflows.
0054 8616      00192      BSF      TCON1,6   ; Change edge from falling
                                ; to rising
00194 ;
00195 ; Note when you change the edge of the capture, an additional capture
00196 ; is generated. The capture register must be read before the capture
00197 ; flag is cleared.
00198 ;
0055 550A      00199 FALSE_C  MOVFP   CA2H,W   ; Dummy read of Capture 2
0056 540A      00200      MOVFP   CA2L,W   ;
00201 ;
0057 B801      00202      MOVLB   1         ; Select register Bank 1
0058 8B16      00203      BCF     PIR,3     ; Clear Capture2 Interrupt flag
0059 0005      00204      RETFIE          ; Return from interrupt, wait for
                                ; T3 overflow or falling edge
                                ; capture
00206 ;
00207 ;
00208 PAGE
00209 ;
00210 ; When Timer 3 has overflowed, the overflow counter only should
00211 ; be incremented when the overflow occurs after a capture 1
00212 ; but before the capture 2. The 6 possible cases when entering
00213 ; the T3OVFL section of the PER_INT routine are as follows:
00214 ;
00215 ; Case 1: T3 overflow (only) and FLAG_REG.0 = 0 (waiting
00216 ; for Capture 1 to occur). Do Not increment counter
00217 ; Case 2: T3 overflow (only) and FLAG_REG.0 = 1 (waiting
00218 ; for Capture 2 to occur). Increment counter
00219 ; Case 3: T3 Overflow, Then Capture1 happened. Do Not
00220 ; increment overflow counter
00221 ; Case 4: T3 Overflow, Then Capture 2 happened
00222 ; Increment counter
00223 ; Case 5: Capture1, Then T3 Overflow happened
00224 ; Increment counter
00225 ; Case 6: Capture2, Then T3 Overflow happened. Do Not
00226 ; Increment counter
00227 ;
005A 8E16      00228 T3OVFL   BCF     PIR,6     ; Clear Overflow interrupt flag
005B 9316      00229      BTFS    PIR,3     ; Did the RB1/CAP2 pin also
00230 ; cause an interrupt?
005C C068      00231      GOTO    FR0      ; No, only overflow interrupt
005D B803      00232      MOVLB   3         ; Bank 3
005E 280A      00233      CLRF    W,0       ; W = 0
005F 9826      00234      BTFS    FLAG_REG,0 ; T3 overflow with Capture 1
00235 ; or Capture 2?
0060 C064      00236      GOTO    OF_C1    ; Overflow with Capture 1
0061 3115      00237 OF_C2    CPFSEQ   CA2H    ; if CA2H = 0, overflow happened
00238 ; first
0062 1525      00239      INCF     T3OFLCNTR,1; Increment counter
0063 C043      00240      GOTO    CAPTURE ; Do capture routine
0064 3115      00241 OF_C1    CPFSEQ   CA2H    ; if CA2H = 0, overflow happened
00242 ; first
0065 C043      00243      GOTO    CAPTURE ; Capture happened first, do NOT
00244 ; Increment overflow counter
00245 ; and do capture routine
0066 1525      00246      INCF     T3OFLCNTR,1; Yes, Inc. the overflow counter
0067 C043      00247      GOTO    CAPTURE ; Do capture routine
00248 ;
00249 ; Only increment overflow counter if between 1st and 2nd capture
00250 ;
0068 9826      00251 FR0     BTFS    FLAG_REG,0 ; Between Capture 1 and Capture 2?
0069 1525      00252      INCF     T3OFLCNTR,1; Yes, Inc. the overflow counter

```

```

006A 0005      00253      RETFIE      ; Return from overflow interrupt
                00254 ;
006B 6A24      00255 SUB16      MOVFP   IC2AL,W      ; Do the 16-bit subtraction
006C 0522      00256      SUBWF   IC2BL,1      ;
006D 6A23      00257      MOVFP   IC2AH,W      ;
006E 0321      00258      SUBWFB  IC2BH,1      ;
006F 9004      00259      BTFSS   ALUSTA,0      ; Is the result pos. or neg. ?
0070 8126      00260      BSF     FLAG_REG,1    ; neg., Set the underflow flag
0071 0002      00261      RETURN     ; Return from the subroutine
                00262 PAGE
                00263 ;
                00264 ; Other Interrupt routines. (Not utilized in this example)
                00265 ;
0072 0005      00266 EXT_INT RETFIE      ; RA0/INT interrupt routine
                00267      ; (NOT used in this program)
0073 0005      00268 TMR0INT RETFIE     ; TMR0 overflow interrupt routine
                00269      ; (NOT used in this program)
0074 0005      00270 T0INT  RETFIE     ; RA1/T0CKI interrupt routine
                00271      ; (NOT used in this program)
                00272 ;
0075 C028      00273 SRESET GOTO   START      ; If program became lost, goto
                00274      ; START and reinitialize.
                00275 ;
                00276 ;
                00277 ; When the executed address is NOT in the program range, the
                00278 ; 16-bit address should contain all 1's (CALL 0x1FFF). At
                00279 ; this location you could branch to a routine to recover or
                00280 ; shut down from the invalid program execution.
                00281 ;
007F          00282      ORG     END_OF_PROG_MEM;
007F C075      00283      GOTO    SRESET      ; The program has lost it's mind,
                00284      ; do a system reset
                00285      END

```

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

```

0000 : X-----X----- X-----X----- X-----XXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXX-----
07C0 : -----X-----X-----X-----X-----X-----X

```

All other memory blocks unused.

Program Memory Words Used: 84

```

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

```



---

**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.

---

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

#### **Trademarks**


The Microchip name and logo, the Microchip logo, FilterLab, KEELOQ, microID, MPLAB, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

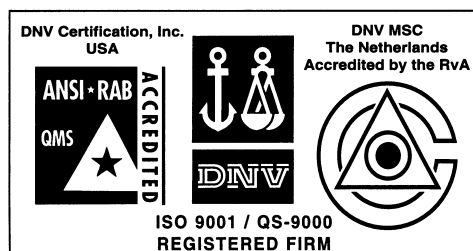
dsPIC, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, MXDEV, PICC, PICDEM, PICDEM.net, rPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2002, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.



*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: 65-6334-8870 Fax: 65-6334-8850

### 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 høj 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 - 1er 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

03/01/02