

## Using the Capture Module

### INTRODUCTION

The PIC16/17 family of RISC-like microcontrollers has been designed to provide advanced performance and a cost-effective solution for a variety of applications. This application report provides examples which illustrate the uses of input capture using the PIC17C42 Timer3 module. These examples may be modified to suit the specific needs of an application.

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

1. Frequency Counter (Period Measurement)
2. Frequency Counter (Period Measurement) using a Free Running Timer
3. Pulse Width Measurement
4. Frequency Counter (Period Measurement) with Input Prescaler

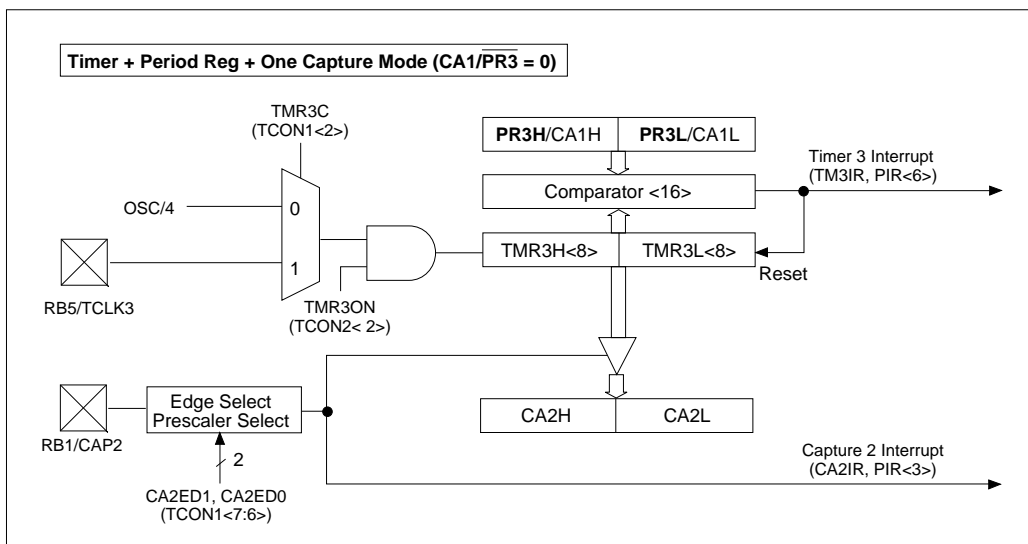
### TIMER3 DESCRIPTION

Timer3 is a 16-bit counter that has two modes of operation which can be software selected. The CA1/PR3 bit (TCON2<3>) selects the mode of operation. The two modes are:

1. Timer3 with Period Register and Single Capture Register (see Figure 1)
2. Timer3 and Dual Capture Registers (see Figure 2)

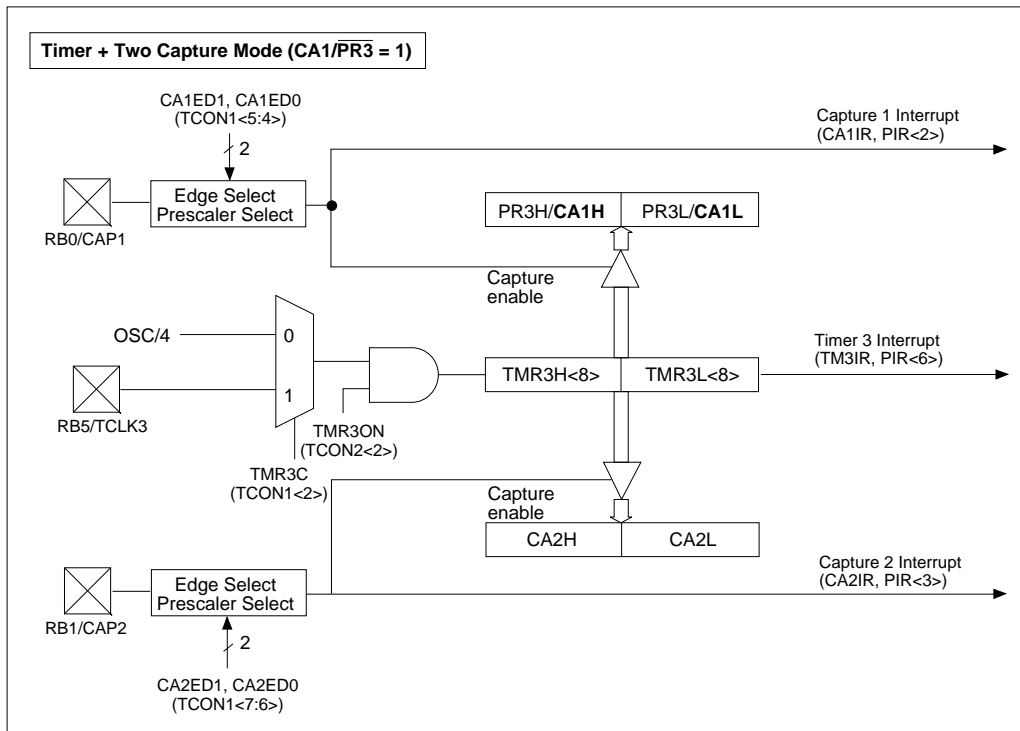
Timer3 is the time-base for capture operations.

**FIGURE 1 - TIMER3 WITH PERIOD REGISTER AND SINGLE CAPTURE REGISTER**



# Capture Module

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:

$$T_{ofl} = T_{clk} * (\text{value in PR3H/CA1H:PR3L/CA1L register pair} + 1).$$

Where  $T_{clk}$  is either the internal system clock ( $T_{cy}$ ) or the external clock cycle time. Table 1 shows time-out period 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 report uses a Timer3 overflow value of FFFFh.

**TABLE 1 - TIMER3 OVERFLOW TIMES**

Overflow Time	Period Register					
	@ 16 MHz (250 ns)	@ 10 MHz (400 ns)	@ 8 MHz (500 ns)	@ 5 MHz (800 μs)	@ 2 MHz (2.0 μs)	@ 32 KHz (125 μs)
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 μS	0x0960	0x05DC	0x04B0	0x02EE	0x012C	0x0005
100 μS	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, for both capture pins, 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.

# Capture Module

The control registers that are utilized 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:**

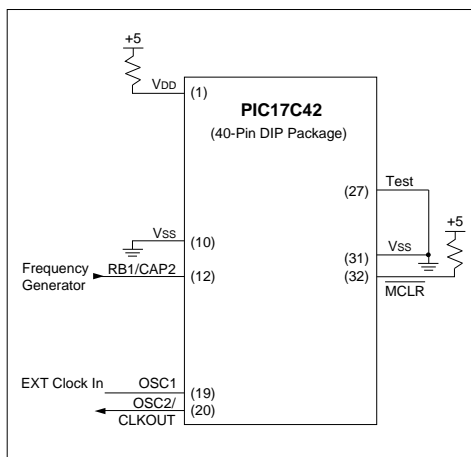
Name	BANK	ADDR	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
INTSTA	All	6h	PEIR	RTXIR	RTCIR	INTIR	PEIE	RTXIE	RTCIE	INTIE
CPUSTA	All	7h	-	-	STKAV	GLINTD	$\overline{TO}$	$\overline{PD}$	-	-
TMR3L	2	12h	Timer3 LSB Register							
TMR3H	2	13h	Timer3 MSB Register							
CA2L	3	14h	Timer3 Capture2 LSB Register							
CA2H	3	15h	Timer3 Capture2 MSB Register							
PIR	1	16h	IRB	TM3IR	TM2IR	TM1IR	CA2IR	CA1IR	TBMT	RBFL
PIE	1	17h	IEB	TM3IE	TM2IE	TM1IE	CA2IE	CA1IE	TXIE	RCIE
PR3L/ CA1L	2	16h	Timer3 - Period MSB Register/Capture1 MSB Register							
PR3H/ CA1H	2	17h	Timer3 - Period MSB Register/ Capture1 MSB Register							
TCON1	3	16h	CA2ED1	CA2ED0	CA1ED1	CA1ED0	16/8	TMR3C	TMR2C	TMR1C
TCON2	3	17h	CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/PR3	TMR3ON	TMR2ON	TMR1ON

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

1. Frequency Counter (Period Measurement)
2. Frequency Counter (Period Measurement) using a Free Running Timer
3. Pulse Width Measurement
4. Frequency Counter (Period Measurement) with Input Prescaler

All these examples can be run from a simple setup, this is show 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 setting the counter to 0000h, then starting the counter on the 1st rising edge. On the following rising edge, the capture 2 register is loaded with the Timer3 value and the Timer3 (TMR3) register is cleared. If the period is greater than the overflow rate of Timer3, the timer 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:

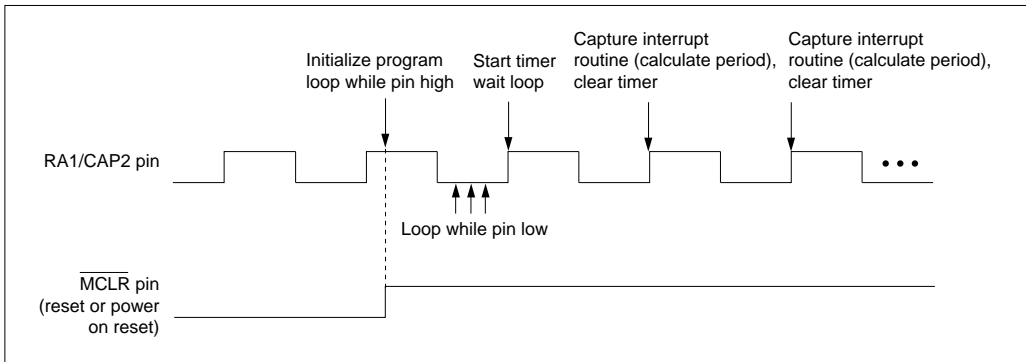
1. The TMR3 overflow is the only interrupt source.
2. Both the TMR3 overflow and capture2 interrupts occurred at near the same time, but the TMR3 overflow occurred first (Most Significant Byte of the Capture2 register = 00h).

Once a capture2 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 Timer3 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:	+/- Tcy (± 250 nsec)

**FIGURE 4 - SOFTWARE TIMING FLOW RELATIVE TO INPUT SIGNAL ON RA1/CAP2**



# Capture Module

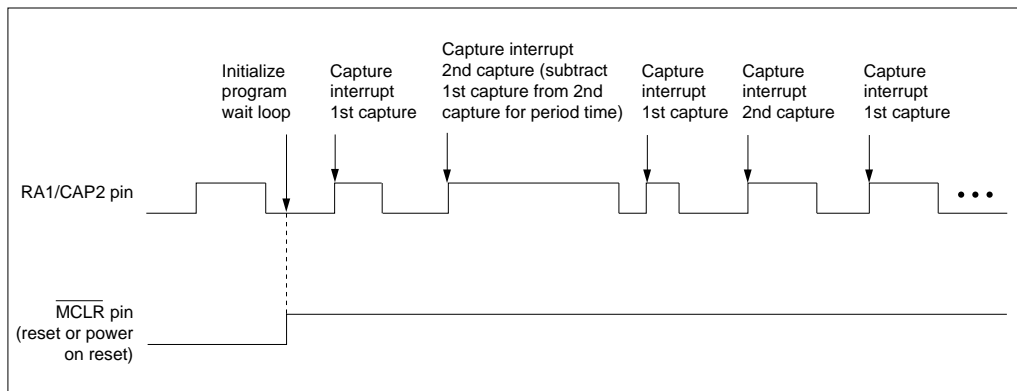
## PERIOD MEASUREMENT (FREQUENCY COUNTER) USING A FREE RUNNING TIMER

In many applications the timer (TMR3) would need to be used for multiple tasks, and could not be reset (modified) by any one of these tasks. This is called a free running timer. To do period measurement in a free running timer application, 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 is 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 Timer3 overflows, and then the 2nd capture occurs. So an overflow counter should only be incremented if the Timer3 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 Timer3 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:	+/- Tcy ( $\pm 250$ nsec)

FIGURE 5 - SOFTWARE TIMING FLOW RELATIVE TO INPUT SIGNAL ON RA1/CAP2



## PULSE WIDTH MEASUREMENT USING A FREE RUNNING TIMER

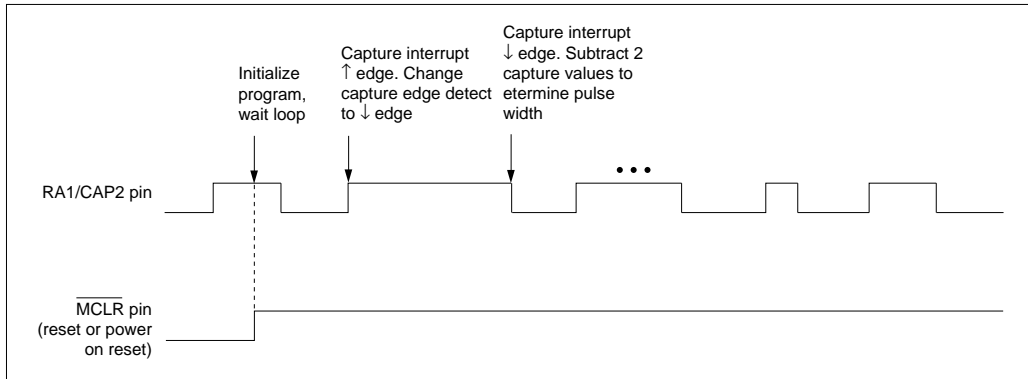
Applications that require the measurement of a pulse width can also be easily handled. The PIC17C42 can be programmed to measure either the low or the high pulse. The software example in Appendix C measures the High pulse time. The program is initialized to capture on the rising edge of the RA1/CAP2 pin. After this event occurs, the capture mode is switched to the falling edge of the RA1/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 RA1/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 RA1/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 uS) 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 uS.

The program listing in Appendix C implements this, assuming only Timer3 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:	+/- Tcy (±250 nsec)

**FIGURE 6 - SOFTWARE TIMING FLOW RELATIVE TO INPUT SIGNAL ON RA1/CAP2**



**TABLE 3 - PERIPHERAL INTERRUPT ROUTINE**

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

# Capture Module

## 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.

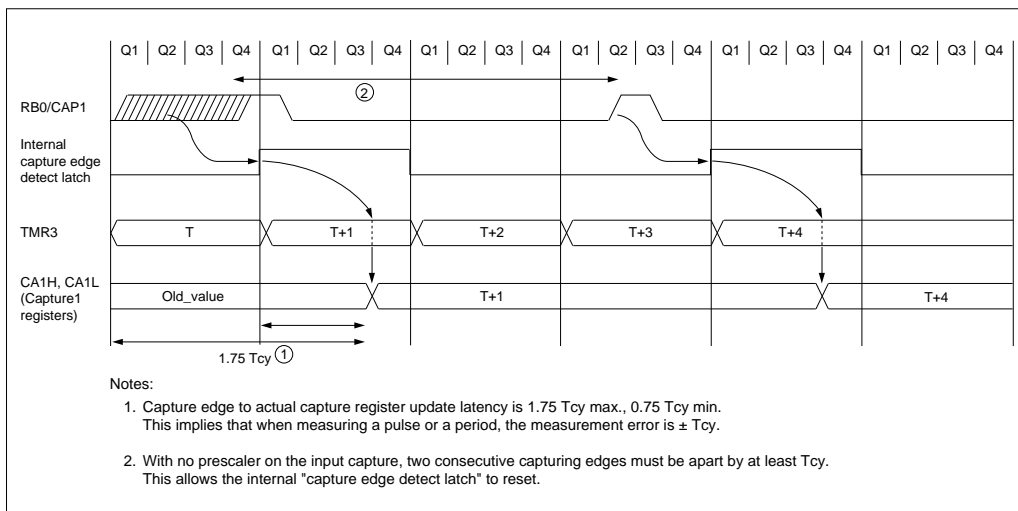
In cases where the resolution of the input frequency is important, the Prescaler can be used to reduce the input capture error. There are two components to input capture:

1. Resolution Error
2. Input Synchronization Error

These two errors add together 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 ( $T_{sync}$ ) can be up to  $1 T_{cy}$  (see 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**



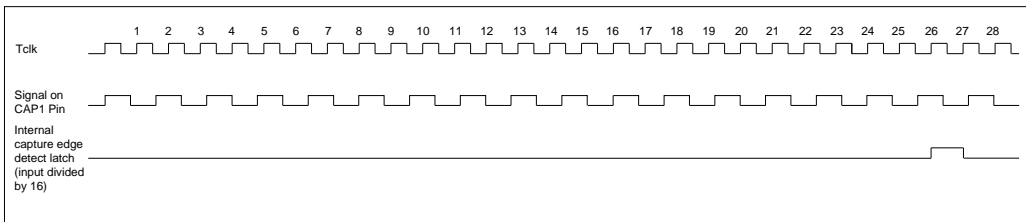


Another scenario is when the signal on the input capture pin is different than the system cycle time (Tcy), for example 1.6 Tcy. If you tried to capture this you would have a capture value of 1. If you set the prescaler to actually capture on the 16th edge you would have  $16 \times 1.6 \text{ Tcy} =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

The program listing in Appendix D implements this, assuming only Timer3 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:	+/- Tcy ( $\pm 16.625$ nsec)

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



*Author: Mark Palmer  
Logic Products Division*

# Capture Module

## APPENDIX A - PERIOD MEASUREMENT EXAMPLE CODE

MPASM 01.01 Released IC\_D16\_2.ASM 6-7-1994 11:17:12

PAGE 1

```
LOC OBJECT CODE      LINE SOURCE TEXT
0001          LIST    p=17C42, c=132
0002 ; This is the basic outline for a program that can determine the
0003 ; frequency of an input, via input capture. The input capture has
0004 ; been selected to capture on the 16th rising edge. This is useful
0005 ; for high frequency inputs, where an interrupt on each rising edge
0006 ; would not be able to be serviced (at that rate). This particular
0007 ; example can support an input signal with a period of approximately
0008 ; 625 nS. Without the divide by 16 selected, this is approximately
0009 ; 10 us. This period time increases (frequency decreases) as the
0010 ; overhead in the main routine increases.
0011 ;
0012 ; This routine uses an 8-bit register to count the times that timer3
0013 ; overflowed. At the Max crystal frequency of 16 MHz, this gives an
0014 ; overflow time of (16)(2**8 + 1)(2**16)(250 nS) > 67.37 sec. If
0015 ; measurement of longer time intervals is required, the overflow
0016 ; counter could be extended to 16 (or more) bits.
0017 ;
0018 ; Timer 3 in this example is a free running timer. The input
0019 ; capture is generated on the RB1/CAP2 pin. There is a flag
0020 ; that specifies if this is the 1st or 2nd capture.
0021 ; The first capture is the start of the period measurement. The
0022 ; second capture value gives the end of the period. In this type
0023 ; of measurement If the 2nd capture value < the 1st capture value
0024 ; then the overflow counter should be decremented.
0025 ;
0026 ; Program:   IC_D16_2.ASM
0027 ; Revision: 1.0
0028 ;
0029 ; Do the EQUate table
0030 ;
0020 0031 IC20F          EQU    0x20          ; T3 overflow register
0021 0032 IC2BH          EQU    0x21          ; T3 ICA2 MSB register (2nd Cap)
0022 0033 IC2BL          EQU    0x22          ; T3 ICA2 LSB register
0023 0034 IC2AH          EQU    0x23          ; T3 ICB2 MSB register (1st Cap)
0024 0035 IC2AL          EQU    0x24          ; T3 ICB2 LSB register
0025 0036 T3OFLCNTR      EQU    0x25          ; Temperay T3 overflow register
0026 0038 FLAG_REG       EQU    0x26          ; Register that has the Flag bits
0039 ;
0040 ; FLAG_REG bit 7 6 5 4 3 2 1 0
0041 ;          -  -  -  -  -  -  -  UFL CAP1
0042 ; CAP1 = 0, 1st Capture
0043 ;          = 1, 2nd Capture
0044 ;
0045 ; UFL = 0, No Underflow
0046 ;          = 1, Underflow during subtract
0047 ;
07FF 0048 END_OF_PROG_MEM EQU    0x07FF
0049 ;
0050 ;
0004 0051 ALUSTA          EQU    0x04
0006 0052 CPUSTA          EQU    0x06
0007 0053 INTSTA          EQU    0x07
000A 0054 W                EQU    0x0A
0055 ;
0012 0056 PORTB           EQU    0x12          ; Bank 0
0057 ;
0016 0058 PIR             EQU    0x16          ; Bank 1
0017 0059 PIE             EQU    0x17
0060 ;
0012 0061 TMR3L            EQU    0x12          ; Bank 2
0013 0062 TMR3H            EQU    0x13
0016 0063 T3PRL            EQU    0x16
0017 0064 T3PRH            EQU    0x17
```

```

0014          0065 ;
0015          0066 CA2L      EQU    0x14      ; Bank 3
0016          0067 CA2H      EQU    0x15
0017          0068 TCON1     EQU    0x16
0018          0069 TCON2     EQU    0x17
0000 C028     0071          ORG    0x0000     ; Origin for the RESET vector
0000 C028     0072          GOTO   START     ; On reset, go to the start of
0000 C028     0073          ;           ; the program
0000 C028     0074          ORG    0x0008     ; Origin for the external RA0/INT
0000 C028     0075          ;           ; interrupt vector
0008 C068     0076          GOTO   EXT_INT    ; Goto the ext. interrupt
0008 C068     0077          ;           ; on RA0/INT routine
0008 C068     0078          ORG    0x0010     ; Origin for the RTCC
0008 C068     0079          ;           ; overflow interrupt vector
0010 C069     0080          GOTO   RTCCINT    ; Goto the RTCC overflow interrupt
0010 C069     0081          ;           ; routine
0010 C069     0082          ORG    0x0018     ; Origin for the external
0010 C069     0083          ;           ; RA1/RT interrupt vector
0018 C06A     0084          GOTO   RT_INT     ; Goto the ext. interrupt on
0018 C06A     0085          ;           ; RA1/RT routine
0018 C06A     0086          ORG    0x0020     ; Origin for the interrupt vector
0018 C06A     0087          ;           ; of any enabled peripheral
0020 C03E     0088          GOTO   PER_INT    ; Goto the interrupt from a
0020 C03E     0089          ;           ; peripheral routine
0020 C03E     0091          ORG    0x0028     ; Origin for the top of
0020 C03E     0092          ;           ; program memory
0028 8406     0093 START     BSF    CPUSTA,4   ; Disable ALL interrupts via the
0028 8406     0094          ;           ; Global Interrupt Disable
0028 8406     0095          ;           ; (GLINTD) bit.
0028 8406     0096          ;
0028 8406     0097 MAIN     ;           ; Place Main program here
0029 B803     0098          MOVLB   3        ; Select register Bank 3
002A 2817     0099          CLRF    TCON2,0   ; Stop the timers, Single Capture
002B B0F0     0100          MOVLW  0x0F0     ; Initialize TCON1 so that
002C 0116     0101          MOVWF  TCON1     ; T1 (8-bit), T2 (8-bit),
002C 0116     0102          ;           ; and T3 run off the internal
002C 0116     0103          ;           ; system clock. Capture2
002C 0116     0104          ;           ; captures on the 16th rising edge
002C 0116     0105          ;
002C 0116     0106          ;           ; Initialize Timer 3, load the timer with the number of cycles that
002C 0116     0107          ;           ; the device executes (from RESET) before the timer is turned on
002C 0116     0108          ;           ; Therefore the offset is required due to software overhead.
002C 0116     0109          ;
002D B802     0110          MOVLB   2        ; Select register Bank 2
002E 280A     0111          CLRF    W,0       ; Clear the W register
002F 0126     0112          MOVWF  FLAG_REG   ; Initialize to 0
0030 0113     0113          MOVWF  TMR3H     ; Timer3 MSB = 0
0031 B000     0114          MOVLW  0x00     ; Timer3 LSB = Offset
0032 0112     0115          MOVWF  TMR3L     ;
0032 0112     0116          ;
0032 0112     0117          ;           ; Load the Timer 3 period register with 0xFFFF, which will give an
0032 0112     0118          ;           ; interrupt on the overflow of Timer3
0032 0112     0119          ;
0033 B0FF     0120          MOVLW  0xFF     ;
0034 0117     0121          MOVWF  T3PRH     ;
0035 0116     0122          MOVWF  T3PRL     ;
0035 0116     0123          ;
0035 0116     0124          ;           ; the timer should be started and interrupts enabled.
0035 0116     0125          ;
0036 B803     0126          MOVLB   3        ; Select register Bank 3
0037 8217     0127          BSF    TCON2,2   ; Turn on timer 3.
0038 8307     0128          BSF    INTSTA,3  ; Turn on Peripheral Interrupts
0039 B801     0129          MOVLB   1        ; Select register Bank 1
003A B048     0130          MOVLW  0x48     ; Enable Caputure 2 and Timer3
003B 0117     0131          MOVWF  PIE       ; Interrupts (when GLINTD = 0)
003B 0117     0132          ;
003B 0117     0133          ;           ; This is where you would do the things you wanted to do.
003B 0117     0134          ;           ; this example will only loop waiting for the interrupts.
003B 0117     0135          ;

```

# Capture Module

```
003C 8C06          0136 WAIT          BCF      CPUSTA,4      ; Enable ALL interrupts
003D C03C          0137          GOTO     WAIT          ; Loop here waiting for a timer
0138              ; Interrupt
0140 ;
0141 ; The interrupt routine for any peripheral interrupt, This routine
0142 ; only deals with Timer3 (T3) interrupts.
0143 ;
0144 ; Time required to execute interrupt routine. Not including
0145 ; interrupt latency (time to enter into the interrupt routine)
0146 ;
0147 ; case1 - only T3 overflow          = 12 cycles
0148 ; case2 - 1st capture              = 14 cycles
0149 ; case3 - 2nd capture              = 30 cycles
0150 ; case4 - T3 overflow and 1st capture = 34 cycles
0151 ; case5 - T3 overflow and 2nd capture = 50 cycles
0152 ;
0153 ;
003E B801          0154 PER_INT MOVLB 1      ; Select register Bank 1
003F 9E16          0155          BTFSC   PIR,6        ; Did T3 overflow?
0156              ; If not skip next Instruction
0040 C055          0157          GOTO     T3OVFL      ; Inc overflow cntr and clear flag
0041 9316          0158 CK_CAP        BTFSS   PIR,3        ; Did the RB1/CAP2 pin cause an
0159              ; interrupt?
0042 0005          0160          RETFIE      ; No RB1/CAP2 interrupt,
0161              ; Return from Interrupt
0162 ;
0163 ; This portion of the code takes the 1st capture and stores its
0164 ; value in register pair IC2AH:IC2AL. When the 2nd capture
0165 ; is take, its value is stored in register pair IC2BH:IC2BL.
0166 ; A 16-bit subtract is performed, with the final 24-bit result
0167 ; being stored in IC2OF:IC2BH:IC2BL. This value will no longer
0168 ; be correct after the next capture occurs (IC2BH:IC2BL will
0169 ; change), so the main routine must utilize this value before
0170 ; it changes.
0171 ;
0043 8B16          0172 CAPTURE BCF      PIR,3      ; Clear Capture2 interrupt flag
0044 B803          0173          MOVLB   3          ; Select register Bank 3
0045 9826          0174          BTFSC   FLAG_REG,0    ; 1st or 2nd capture2?
0046 C04B          0175          GOTO     CAP2        ; It was the 2nd Capture
0047 5424          0176 CAP1      MOVFP   CA2L,IC2AL    ; Move the captured value to
0048 5523          0177          MOVFP   CA2H,IC2AH    ; temporary registers
0049 8026          0178          BSF     FLAG_REG,0    ; Have 1st capture2
004A 0005          0179          RETFIE      ; Return from Interrupt
0180 ;
004B 5422          0182 CAP2      MOVFP   CA2L,IC2BL    ; Move the captured value to
004C 5521          0183          MOVFP   CA2H,IC2BH    ; temporary registers
0184              ; (to prevent being overwritten)
0185 ;
004D E061          0186          CALL    SUB16      ; Call routine which subtracts
0187              ; 2 16-bit numbers.
004E 9926          0188          BTFSC   FLAG_REG,1    ; Underflow during SUB16?
004F 0725          0189          DECF   T3OFLCNTR,1    ; Since underflow, decrement the
0190              ; overflow counter.
0050 2926          0191          CLRFB  FLAG_REG,1    ; Clear the flag bits for
0192              ; underflow and 2nd capture2
0051 6A25          0193          MOVFP   T3OFLCNTR,W    ; Store the T3 input capture
0052 4A20          0194          MOVFP   W,IC2OF        ; overflow value in IC2OF
0053 2825          0195          CLRFB  T3OFLCNTR,0    ; Clear the Data register which
0196              ; counts how many times Timer 3
0197              ; overflows.
0054 0005          0198          RETFIE      ; Return from interrupt
0199 ;
0200 ; When Timer 3 has overflowed, the overflow counter only should
0201 ; be incremented when the overflow occurs after a capture 1
0202 ; but before the capture 2. The 4 possible cases when entering
0203 ; the T3OVFL section of the PER_INT routine are as follows:
0204 ; Case 1: T3 overflow (only) and FLAG_REG.0 = 0 (waiting
0205 ; for Capture 1 to occur). Do Not increment counter
0206 ; Case 2: T3 overflow (only) and FLAG_REG.0 = 1 (waiting
```

```

0207 ;           for Capture 2 to occur). Increment counter
0208 ; Case 3: T3 Overflow happened after Capture. Do Not
0209 ;           increment overflow counter
0210 ; Case 4: T3 Overflow occurred before Capture 2 and FLAG_REG.0 = 1
0211 ;           (waiting for Capture 2 to occur). Increment counter
0212
0213 ;
0055 8E16      0214 T3OVFL          BCF     PIR,6           ; Clear Overflow interrupt flag
0056 9316      0215                BTFSS   PIR,3           ; Did the RB1/CAP2 pin also
                                0216                ; cause an interrupt?
0057 C05E      0217                GOTO    FR0            ; No, Check if between 1st
                                0218                ; and 2nd capture
0058 B803      0219                MOVLB   3              ; Bank 3
0059 280A      0220                CLRF   W,0            ; W = 0
005A 3115      0221                CPFSEQ  CA2H           ; if CA2H = 0, overflow happened
005B C05E      0222                GOTO    FR0            ; first, must check FLAG_REG
                                0223                ; bit 0
005C B801      0224                MOVLB   1              ; Back to bank 1
005D C043      0225                GOTO    CAPTURE        ; Capture happened first, do NOT
                                0226                ; Increment overflow counter
                                0227                ; and do capture routine
005E 9826      0228 FR0          BTFSC   FLAG_REG,0        ; Between Capture 1 and Capture 2?
005F 1525      0229                INCF   T3OFLCNTR,1    ; Yes, Inc. the overflow counter
0060 0005      0230                RETFIE  ; Return from overflow interrupt
                                0231 ;
0061 6A24      0232 SUB16    MOVFP   IC2AL,W          ; Do the 16-bit subtraction
0062 0522      0233                SUBWF  IC2BL,1        ;
0063 6A23      0234                MOVFP  IC2AH,W        ;
0064 0321      0235                SUBWFB IC2BH,1        ;
0065 9004      0236                BTFSS  ALUSTA,0      ; Is the result pos. or neg. ?
0066 8126      0237                BSF   FLAG_REG,1     ; neg., Set the underflow flag
0067 0002      0238                RETURN  ; Return from the subroutine
                                0240 ;
                                0241 ; Other Interrupt routines. (Not utilized in this example)
                                0242 ;
0068 0005      0243 EXT_INT    RETFIE  ; RA0/INT interrupt routine
                                0244                ; (NOT used in this program)
0069 0005      0245 RTCCINT    RETFIE  ; RTCC overflow interrupt routine
                                0246                ; (NOT used in this program)
006A 0005      0247 RT_INT     RETFIE  ; RA1/RT interrupt routine
                                0248                ; (NOT used in this program)
                                0249                ;
006B C028      0250 SRESET     GOTO    START        ; If program became lost, goto
                                0251                ; START and reinitialize.
                                0252 ;
                                0253 ;
                                0254 ; When the executed address is NOT in the program range, the
                                0255 ; 16-bit address should contain all 1's (a CALL 0x1FFF). At
                                0256 ; this location you could branch to a routine to recover or
                                0257 ; shut down from the invalid program execution.
                                0258 ;
007F C06B      0259 ORG      END_OF_PROG_MEM ;
                                0260                GOTO    SRESET        ; The program has lost it's mind,
                                0261                ; do a system reset
                                0262                END
                                0263

```

# Capture Module

## APPENDIX B - PERIOD MEASUREMENT WITH FREE RUNNING TIMER EXAMPLE CODE

MPASM 01.01 Released IC\_FRT2.ASM 6-7-1994 11:16:54

PAGE 1

```
LOC OBJECT CODE LINE SOURCE TEXT
0001 LIST p=17C42, c=132
0002 ; This is the basic outline for a program that can determine the
0003 ; frequency of an input, via input capture. This routine uses an
0004 ; 8-bit register to count the times that timer3 overflowed. At the
0005 ; Max crystal frequency of 16 MHz, this gives an overflow time of
0006 ; (2**16)(256 + 1)(250 nS) > 4.21 sec or a frequency < 0.25 Hz. If
0007 ; measurement of longer time intervals is required, the overflow
0008 ; counter could be extended to 16 (or more) bits.
0009 ;
0010 ; Timer 3 in this example is a free running timer. The input
0011 ; capture is generated on the RB1/CAP2 pin. There is a flag
0012 ; that specifies if this is the 1st or 2nd capture.
0013 ; The first capture is the start of the period measurement. The
0014 ; second capture value gives the end of the period. In this type
0015 ; of measurement If the 2nd capture value < the 1st capture value
0016 ; then the overflow counter should be decremented.
0017 ;
0018 ; Program: IC_FRT2.ASM
0019 ; Revision: 1.0
0020 ;
0021 ; Do the EQUate table
0022 ;
0020 0023 IC2OF EQU 0x20 ; T3 overflow register
0021 0024 IC2BH EQU 0x21 ; T3 ICA2 MSB register (2nd Cap)
0022 0025 IC2BL EQU 0x22 ; T3 ICA2 LSB register
0023 0026 IC2AH EQU 0x23 ; T3 ICB2 MSB register (1st Cap)
0024 0027 IC2AL EQU 0x24 ; T3 ICB2 LSB register
0025 0028 T3OFLCNTR EQU 0x25 ; Temperary T3 overflow register
0029 ;
0026 0030 FLAG_REG EQU 0x26 ; Register that has the Flag bits
0031 ;
0032 ; FLAG_REG bit 7 6 5 4 3 2 1 0
0033 ; - - - - - - - UFL CAP1
0034 ; CAP1 = 0, 1st Capture
0035 ; = 1, 2nd Capture
0036 ;
0037 ; UFL = 0, No Underflow
0038 ; = 1, Underflow during subtract
0039 ;
07FF 0040 END_OF_PROG_MEM EQU 0x07FF
0041 ;
0042 ;
0004 0043 ALUSTA EQU 0x04
0006 0044 CPUSTA EQU 0x06
0007 0045 INTSTA EQU 0x07
000A 0046 W EQU 0x0A
0047 ;
0012 0048 PORTB EQU 0x12 ; Bank 0
0049 ;
0016 0050 PIR EQU 0x16 ; Bank 1
0017 0051 PIE EQU 0x17
0052 ;
0012 0053 TMR3L EQU 0x12 ; Bank 2
0054 ;
0013 0054 TMR3H EQU 0x13
0016 0055 T3PRL EQU 0x16
0017 0056 T3PRH EQU 0x17
0057 ;
0014 0058 CA2L EQU 0x14 ; Bank 3
0015 0059 CA2H EQU 0x15
0016 0060 TCON1 EQU 0x16
0017 0061 TCON2 EQU 0x17
```

```

0000 C028      0063          ORG      0x0000      ; Origin for the RESET vector
                0064          GOTO     START       ; On reset, go to the start of
                0065                      ; the program
                0066          ORG      0x0008      ; Origin for the external RA0/INT
                0067                      ; interrupt vector
0008 C068      0068          GOTO     EXT_INT     ; Goto the ext. interrupt
                0069                      ; on RA0/INT routine
                0070          ORG      0x0010      ; Origin for the RTCC
                0071                      ; overflow interrupt vector
0010 C069      0072          GOTO     RTCCINT    ; Goto the RTCC overflow interrupt
                0073                      ; routine
                0074          ORG      0x0018      ; Origin for the external
                0075                      ; RAL/RT interrupt vector
0018 C06A      0076          GOTO     RT_INT     ; Goto the ext. interrupt on
                0077                      ; RAL/RT routine
                0078          ORG      0x0020      ; Origin for the interrupt vector
                0079                      ; of any enabled peripheral
0020 C03E      0080          GOTO     PER_INT     ; Goto the interrupt from a
                0081                      ; peripheral routine
                0082          ORG      0x0028      ; Origin for the top of
                0083                      ; program memory
0028 8406      0085  START    BSF      CPUSTA,4    ; Disable ALL interrupts via the
                0086                      ; Global Interrupt Disable
                0087                      ; (GLINTD) bit.
                0088                      ;
                0089  MAIN    ; Place Main program here
0029 B803      0090          MOVLB    3          ; Select register Bank 3
002A 2817      0091          CLRF     TCON2,0    ; Stop the timers, Single Capture
002B B070      0092          MOVLW   0x070     ; Initialize TCON1 so that
002C 0116      0093          MOVWF   TCON1     ; T1 (8-bit), T2 (8-bit),
                0094                      ; and T3 run off the internal
                0095                      ; system clock. Capture2
                0096                      ; captures on the rising edge.
                0097 ;
                0098 ; Initialize Timer 3, load the timer with the number of cycles that
                0099 ; the device executes (from RESET) before the timer is turned on
                0100 ; Therefore the offset is required due to software overhead.
                0101 ;
002D B802      0102          MOVLB    2          ; Select register Bank 2
002E 280A      0103          CLRF     W,0       ; Clear the W register
002F 0126      0104          MOVWF   FLAG_REG   ; Initialize to 0
0030 0113      0105          MOVWF   TMR3H     ; Timer3 MSB = 0
0031 B013      0106          MOVLW   0x13     ; Timer3 LSB = Offset
0032 0112      0107          MOVWF   TMR3L     ;
                0108 ;
                0109 ; Load the Timer 3 period register with 0xFFFF, which will give an
                0110 ; interrupt on the overflow of Timer3
                0111 ;
0033 B0FF      0112          MOVLW   0xFF     ;
0034 0117      0113          MOVWF   T3PRH     ;
0035 0116      0114          MOVWF   T3PRL     ;
                0115 ;
                0116 ; the timer should be started and interrupts enabled.
                0117 ;
0036 B803      0118          MOVLB    3          ; Select register Bank 3
0037 8217      0119          BSF     TCON2,2    ; Turn on timer 3.
0038 8307      0120          BSF     INTSTA,3    ; Turn on Peripheral Interrupts
0039 B801      0121          MOVLB    1          ; Select register Bank 1
003A B048      0122          MOVLW   0x48     ; Enable Capture 2 and Timer3
003B 0117      0123          MOVWF   PIE      ; Interrupts (when GLINTD = 0)
                0124 ;
                0125 ; This is where you would do the things you wanted to do.
                0126 ; this example will only loop waiting for the interrupts.
                0127 ;
003C 8C06      0128  WAIT    BCF     CPUSTA,4    ; Enable ALL interrupts
003D C03C      0129          GOTO     WAIT       ; Loop here waiting for a timer
                0130                      ; Interrupt

```

# Capture Module

```
0132 ;
0133 ; The interrupt routine for any peripheral interrupt, This routine
0134 ; only deals with Timer3 (T3) interrupts.
0135 ;
0136 ; Time required to execute interrupt routine. Not including
0137 ; interrupt latency (time to enter into the interrupt routine)
0138 ;
0139 ;         case1 - only T3 overflow           = 12 cycles
0140 ;         case2 - 1st capture              = 14 cycles
0141 ;         case3 - 2nd capture              = 30 cycles
0142 ;         case4 - T3 overflow and 1st capture = 34 cycles
0143 ;         case5 - T3 overflow and 2nd capture = 50 cycles
0144 ;
0145 ;
003E B801 0146 PER_INT MOVLB 1           ; Select register Bank 1
003F 9E16 0147         BTFSC  PIR,6       ; Did T3 overflow?
0148         ; If not skip next Instruction
0040 C055 0149         GOTO   T3OVFL          ; Inc overflow cntr and clear flag
0041 9316 0150 CK_CAP  BTFSS  PIR,3       ; Did the RB1/CAP2 pin cause an
0151         ; interrupt?
0042 0005 0152         RETFIE          ; No RB1/CAP2 interrupt,
0153         ; Return from Interrupt
0154 ;
0155 ; This portion of the code takes the 1st capture and stores its
0156 ; value in register pair IC2AH:IC2AL. When the 2nd capture
0157 ; is take, its value is stored in register pair IC2BH:IC2BL.
0158 ; A 16-bit subtract is performed, with the final 24-bit result
0159 ; being stored in IC2OF:IC2BH:IC2BL. This value will no longer
0160 ; be correct after the next capture occurs (IC2BH:IC2BL will
0161 ; change), so the main routine must utilize this value before
0162 ; it changes.
0163 ;
0043 8B16 0164 CAPTURE BCF PIR,3         ; Clear Capture2 interrupt flag
0044 B803 0165         MOVLB  3           ; Select register Bank 3
0045 9826 0166         BTFSC  FLAG_REG,0 ; 1st or 2nd capture2?
0046 C04B 0167         GOTO   CAP2       ; It was the 2nd Capture
0047 5424 0168 CAP1  MOVVPF CA2L,IC2AL    ; Move the captured value to
0048 5523 0169         MOVVPF CA2H,IC2AH   ; temporary registers
0049 8026 0170         BSF   FLAG_REG,0   ; Have 1st capture2
004A 0005 0171         RETFIE          ; Return from Interrupt
0172         ;
004B 5422 0174 CAP2  MOVVPF CA2L,IC2BL ; Move the captured value to
004C 5521 0175         MOVVPF CA2H,IC2BH   ; temporary registers
0176         ; (to prevent being overwritten)
0177         ;
004D E061 0178         CALL  SUB16        ; Call routine which subtracts
0179         ; 2 16-bit numbers.
004E 9926 0180         BTFSC  FLAG_REG,1   ; Underflow during SUB16?
004F 0725 0181         DECF  T3OFLCNTR,1 ; Since underflow, decrement the
0182         ; overflow counter.
0050 2926 0183         CLRFB FLAG_REG,1   ; Clear the flag bits for
0184         ; underflow and 2nd capture2
0051 6A25 0185         MOVFP  T3OFLCNTR,W ; Store the T3 input capture
0052 4A20 0186         MOVVPF W,IC2OF   ; overflow value in IC2OF
0053 2825 0187         CLRFB T3OFLCNTR,0 ; Clear the Data register which
0188         ; counts how many times Timer 3
0189         ; overflows.
0054 0005 0190         RETFIE          ; Return from interrupt
0191 ;
0192 ; When Timer 3 has overflowed, the overflow counter only should
0193 ; be incremented when the overflow occurs after a capture 1
0194 ; but before the capture 2. The 4 possible cases when entering
0195 ; the T3OVFL section of the PER_INT routine are as follows:
0196 ; Case 1: T3 overflow (only) and FLAG_REG.0 = 0 (waiting
0197 ;         for Capture 1 to occur). Do Not increment counter
0198 ; Case 2: T3 overflow (only) and FLAG_REG.0 = 1 (waiting
0199 ;         for Capture 2 to occur). Increment counter
0200 ; Case 3: T3 Overflow happened after Capture. Do Not
```



```

0201 ;          increment overflow counter
0202 ;   Case 4: T3 Overflow occurred before Capture 2 and FLAG_REG.0 = 1
0203 ;          (waiting for Capture 2 to occur). Increment counter
0204
0205 ;
0055 8E16 0206 T3OVFL  BCF  PIR,6          ; Clear Overflow interrupt flag
0056 9316 0207          BTFSS  PIR,3          ; Did the RBl/CAP2 pin also
                                ;   cause an interrupt?
0208
0057 C05E 0209          GOTO   FR0          ; No, Check if between 1st
                                ;   and 2nd capture
0210
0058 B803 0211          MOVLB  3          ; Bank 3
0059 280A 0212          CLRF   W,0          ; W = 0
005A 3115 0213          CPFSEQ  CA2H        ; if CA2H = 0, overflow happened
005B C05E 0214          GOTO   FR0          ;   first, must check FLAG_REG
                                ;   bit 0
0215
005C B801 0216          MOVLB  1          ; Back to bank 1
005D C043 0217          GOTO   CAPTURE      ; Capture happened first, do NOT
                                ;   Increment overflow counter
                                ;   and do capture routine
0218
005E 9826 0220 FR0      BTFSC  FLAG_REG,0    ; Between Capture 1 and Capture 2?
005F 1525 0221          INCF   T3OFLCNTR,1  ; Yes, Inc. the overflow counter
0060 0005 0222          RETFIE          ; Return from overflow interrupt
0223 ;
0061 6A24 0224 SUB16   MOVFP  IC2AL,W        ; Do the 16-bit subtraction
0062 0522 0225          SUBWF  IC2BL,1        ;
0063 6A23 0226          MOVFP  IC2AH,W        ;
0064 0321 0227          SUBWFB IC2BH,1        ;
0065 9004 0228          BTFSS  ALUSTA,0      ; Is the result pos. or neg. ?
0066 8126 0229          BSF   FLAG_REG,1    ; neg., Set the underflow flag
0067 0002 0230          RETURN          ; Return from the subroutine
0231 ;
0232 ;
0233 ; Other Interrupt routines. (Not utilized in this example)
0234 ;
0068 0005 0235 EXT_INT  RETFIE          ; RA0/INT interrupt routine
                                ;   (NOT used in this program)
0236
0069 0005 0237 RTCCINT  RETFIE          ; RTCC overflow interrupt routine
                                ;   (NOT used in this program)
0238
006A 0005 0239 RT_INT   RETFIE          ; RAl/RT interrupt routine
                                ;   (NOT used in this program)
0240
0241
006B C028 0242 SRESET  GOTO   START        ; If program became lost, goto
                                ;   START and reinitialize.
0243
0244 ;
0245 ;
0246 ; When the executed address is NOT in the program range, the
0247 ; 16-bit address should contain all 1's (a CALL 0x1FFF). At
0248 ; this location you could branch to a routine to recover or
0249 ; shut down from the invalid program execution.
0250 ;
0251          ORG   END_OF_PROG_MEM      ;
007F C06B 0252          GOTO   SRESET        ; The program has lost it's mind,
                                ;   do a system reset
0253
0254          END
0255

```

## APPENDIX C - PULSE WIDTH MEASUREMENT EXAMPLE CODE

```

0001      LIST      p=17C42, c=132
0002 ; This is the basic outline for a program that can determine the
0003 ; Pulse Width of an input, via input capture. This routine uses an
0004 ; 8-bit register to count the times that timer3 overflowed. At the
0005 ; Max crystal frequency of 16 MHz, this gives an overflow time of
0006 ; (2**16)/(256 + 1)/(250 nS) > 4.21 sec or a frequency < 0.25 Hz. If
0007 ; measurement of longer time intervals is required, the overflow
0008 ; counter could be extended to 16 (or more) bits.
0009 ;
0010 ; Program: PW02.ASM
0011 ; Revision: 1.0
0012 ;
0013 ; Do the EQUate table
0014 ;
0015 IC20F      EQU 0x20      ; T3 overflow register
0016 IC2BH     EQU 0x21      ; T3 ICA2 MSB register (2nd Cap)
0017 IC2BL     EQU 0x22      ; T3 ICA2 LSB register
0018 IC2AH     EQU 0x23      ; T3 ICB2 MSB register (1st Cap)
0019 IC2AL     EQU 0x24      ; T3 ICB2 LSB register
0020 T3OFLCNTR EQU 0x25      ; Temperary T3 overflow register
0021 ;
0022 FLAG_REG   EQU 0x26      ; Register that has the Flag bits
0023 ;
0024 ; FLAG_REG bit 7 6 5 4 3 2 1 0
0025 ;
0026 ; CAP1 = 0, 1st Capture
0027 ; = 1, 2nd Capture
0028 ;
0029 ; UFL = 0, No Underflow
0030 ; = 1, Underflow during subtract
0031 ;
0032 END_OF_PROG_MEM EQU 0x07FF
0033 ;
0034 ALUSTA     EQU 0x04
0035 CPOSTA     EQU 0x06
0036 INTSTA     EQU 0x07
0037 W          EQU 0x0A
0038
0039 PORTB      EQU 0x12      ; Bank 0
0040
0041 PIR         EQU 0x16      ; Bank 1
0042 PIE        EQU 0x17      ;
0043
0044 TMR3L       EQU 0x12      ; Bank 2

```



# Capture Module

```
0091 ; the device executes (from RESET) before the timer is turned on
0092 ; Therefore the offset is required due to software overhead.
0093 ;
0094          MOVLEB 2          ; Select register Bank 2
0095          CLRF  W_0         ; Clear the W register
0096          MOVWF FLAG_REG   ; Initialize to 0
0097          MOVWF TMR3H      ; Timer3 MSB = 0
0098          MOVLEW 0x13     ; Timer3 LSB = Offset
0099          MOVWF  TMR3L    ;
0100 ;
0101 ; Load the Timer 3 period register with 0xFFFF, which will give an
0102 ; interrupt on the overflow of Timer3
0103 ;
0104          MOVLEW 0xFF      ;
0105          MOVWF  T3PRH    ;
0106          MOVWF  T3PRL   ;
0107 ;
0108 ; the timer should be started and interrupts enabled.
0109 ;
0110          MOVLEB 3        ; Select register Bank 3
0111          BSF  TCON2,2    ; Turn on timer 3.
0112          BSF  INTSTA,3   ; Turn on Peripheral Interrupts
0113          MOVLEB 1        ; Select register Bank 1
0114          MOVLEW 0x48    ; Enable Capture 2 and Timer3
0115          MOVWF  PIE      ; Interrupts (when GLINTD = 0)
0116 ;
0117 ; This is where you would do the things you wanted to do.
0118 ; this example will only loop waiting for the interrupts.
0119 ;
0120          WAIT  BCF  CPUSTA,4          ; Enable ALL interrupts
0121          GOTO  WAIT                  ; Loop here waiting for a timer
0122                                     ; interrupt

0124 ;
0125 ; The interrupt routine for any peripheral interrupt, This routine
0126 ; only deals with Timer3 (T3) interrupts.
0127 ;
0128 ; Time required to execute interrupt routine. Not including
0129 ; interrupt latency (time to enter into the interrupt routine)
0130 ;
0131 ; case1 - only T3 overflow = 12 cycles
0132 ; case2 - 1st capture = 20 cycles
0133 ; case3 - 2nd capture = 34 cycles
0134 ; case4 - T3 overflow and 1st capture = 32 cycles
0135 ; case5 - T3 overflow and 2nd capture = 44 cycles
0136 ;
```

```

003E B801          MOVLB 1          ; Select register Bank 1
003F 9E16          BTFSC PIR,6        ; Did T3 overflow?
0137 ;              ; If not skip next instruction
0138 PER_INT       MOVLB 1          ; Select register Bank 1
0139               BTFSC PIR,6        ; Did T3 overflow?
0140               ; If not skip next instruction
0040 C05A          GOTO T3OVFL      ; Inc overflow cntnr and clear flag
0041 9316          BTFSS PIR,3      ; Did the RB1/CAP2 pin cause an
0141               ; interrupt?
0142 CK_CAP        GOTO T3OVFL      ; No RB1/CAP2 interrupt,
0143               BTFSS PIR,3      ; Return from Interrupt
0144               RETFIE
0145
0146
0147 ;
0148 ; This portion of the code takes the 1st capture and stores its
0149 ; value in register pair IC2AH:IC2AL. When the 2nd capture
0150 ; is take, its value is stored in register pair IC2BH:IC2BL.
0151 ; A 16-bit subtract is performed, with the final 24-bit result
0152 ; being stored in IC20F:IC2BH:IC2BL. This value will no longer
0153 ; be correct after the next capture occurs (IC2BH:IC2BL will
0154 ; change), so the main routine must utilize this value before
0155 ; it changes.
0156 ;
0043 B803          MOVLB 3          ; Select register Bank 3
0044 9826          BTFSC FLAG_REG,0   ; Capture on rising or falling edge?
0045 C04B          GOTO FALLING      ; It was the 2nd Capture
0046 5424          MOVPP CA2L,IC2AL   ; Move the captured value to
0047 5523          MOVPP CA2H,IC2AH   ; temporary registers
0048 8026          BSF FLAG_REG,0    ; Set flag for 1st capture
0049 8E16          BCF TCONL,6       ; Change edge from rising
0163
0164
0165 GOTO FALSE_C   ; to falling
0166 ;** With the changing of the capture
0167 ;** edge, we have a false capture
;
0169 FALLING       MOVPP CA2L,IC2BL   ; Move the captured value to
0170               MOVPP CA2H,IC2BH   ; temporary registers
0171               ; (to prevent being overwritten)
0172
0173 CALL SUB16     ; Call routine which subtracts
0174               ; 2 16-bit numbers.
004E 9926       BTFSC FLAG_REG,1    ; Underflow during SUB16?
004F 0725       DECF T3OFLCNTR,1    ; Since underflow, decrement the
0177               ; overflow counter.
0178 CLRF FLAG_REG,1 ; Clear the flag bits for
0179               ; underflow and 2nd capture2
0180 MOVPP T3OFLCNTR,W ; Store the T3 input capture
0181 MOVPP W,IC2OF  ; overflow value in IC2OF
0050 2926       CLRF T3OFLCNTR,0    ; Clear the Data register which
0182               ; counts how many times Timer 3
0183               ; overflows.
0184

```

# Capture Module

```

0054 8616          BSF          TCON1,6          ; Change edge from falling
0185          ;          to rising
0186
0187 ;
0188 ; Note when you change the edge of the capture, an additional capture
0189 ; is generated. The capture register must be read before the capture
0190 ; flag is cleared.
0191 ;
0192 FALSE_C      MOVPPF CA2H,W          ; Dummy read of Capture 2
0193          MOVPPF CA2L,W          ;
0194 ;
0195          MOVVLB 1          ; Select register Bank 1
0196          BCF     PIR, 3          ; Clear Capture2 interrupt flag
0197          RETFIE          ; Return from interrupt, wait for
0198          ;          T3 overflow or falling edge
0199          ;          capture
0200          ;

0202 ;
0203 ; When Timer 3 has overflowed, the overflow counter only should
0204 ; be incremented when the overflow occurs after a capture 1
0205 ; but before the capture 2. The 6 possible cases when entering
0206 ; the T3OVFL section of the PER_INT routine are as follows:
0207 ;
0208 ; Case 1: T3 overflow (only) and FLAG_REG.0 = 0 (waiting
0209 ; for Capture 1 to occur). Do Not increment counter
0210 ; Case 2: T3 overflow (only) and FLAG_REG.0 = 1 (waiting
0211 ; for Capture 2 to occur). Increment counter
0212 ; Case 3: T3 Overflow, Then Capture1 happened. Do Not
0213 ; increment overflow counter
0214 ; Case 4: T3 Overflow, Then Capture 2 happened
0215 ; Increment counter
0216 ; Case 5: Capture1, Then T3 Overflow happened
0217 ; Increment counter
0218 ; Case 6: Capture2, Then T3 Overflow happened. Do Not
0219 ; Increment counter
0220 ;
005A 8E16          BCF     PIR,6          ; Clear Overflow interrupt flag
005B 9316          BTFSS  PIR, 3          ; Did the RB1/CAP2 pin also
0222          ;          cause an interrupt?
0223          ; No, only overflow interrupt
005C C068          GOTO   FRO          ; Bank 3
0224          MOVVLB 3          ; W = 0
005D B803          CLRF   W,0          ;
0225          BTFSC  FLAG_REG,0          ; T3 overflow with Capture 1
005E 280A          ;          or Capture 2?
0226          ;          Overflow with Capture 1
005F 9826          GOTO   OF_C1          ; if CA2H = 0, overflow happened
0227          CPFSEQ  CA2H          ; first
0228          INCF   T3OFLCNTR,1          ; Increment counter
0229          GOTO   CAPTURE          ; Do capture routine
0230 OF_C2
0231
0232
0233

```

```

0064 3115      CPFSEQ CA2H      ; if CA2H = 0, overflow happened
0235          ; first
0236          GOTO CAPTURE   ; Capture happened first, do NOT
0237          ; Increment overflow counter
0238          INCF T3OFLCNTR,1 ; and do capture routine
0239          GOTO CAPTURE   ; Yes, Inc. the overflow counter
0240          ; Do capture routine
0241          ;
0242          ; Only increment overflow counter if between 1st and 2nd capture
0243          ;
0244          BTFSF FLAG_REG,0 ; Between Capture 1 and Capture 2?
0245          INCF T3OFLCNTR,1 ; Yes, Inc. the overflow counter
0246          RETFIE        ; Return from overflow interrupt
0247          ;
0248          MOVFP IC2AL,W    ; Do the 16-bit subtraction
0249          SUBWF IC2BL,1    ;
0250          MOVFP IC2AH,W    ;
0251          SUBWFB IC2BH,1   ;
0252          BTFSF ALUSTA,0   ; Is the result pos. or neg. ?
0253          BSF FLAG_REG,1 ; neg., Set the underflow flag
0254          RETURN         ; Return from the subroutine

0256          ;
0257          ; Other Interrupt routines. (Not utilized in this example)
0258          ;
0259          EXT_INT        ; RA0/INT interrupt routine
0260          RETFIE        ; (NOT used in this program)
0261          RTCCINT       ; RTCC overflow interrupt routine
0262          RETFIE        ; (NOT used in this program)
0263          RT_INT       ; RAI/RT interrupt routine
0264          RETFIE        ; (NOT used in this program)
0265          ;
0266          SRESET       ; If program became lost, goto
0267          GOTO START   ; START and reinitialize.
0268          ;
0269          ;
0270          ; When the executed address is NOT in the program range, the
0271          ; 16-bit address should contain all 1's (a CALL 0x1FFF). At
0272          ; this location you could branch to a routine to recover or
0273          ; shut down from the invalid program execution.
0274          ;
0275          ORG END_OF_PROG_MEM ;
0276          GOTO SRESET
0277          ;
0278          END
0279          ;

```

# Capture Module

---

NOTES:



---

---

# WORLDWIDE SALES & SERVICE

---

---

## AMERICAS

### Corporate Office

Microchip Technology Inc.  
2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 602 786-7200 Fax: 602 786-7277  
Technical Support: 602 786-7627  
Web: <http://www.mchip.com/microhip>

### Atlanta

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

### Boston

Microchip Technology Inc.  
5 Mount Royal Avenue  
Marlborough, MA 01752  
Tel: 508 480-9990 Fax: 508 480-8575

### Chicago

Microchip Technology Inc.  
333 Pierce Road, Suite 180  
Itasca, IL 60143  
Tel: 708 285-0071 Fax: 708 285-0075

### Dallas

Microchip Technology Inc.  
14651 Dallas Parkway, Suite 816  
Dallas, TX 75240-8809  
Tel: 214 991-7177 Fax: 214 991-8588

### Dayton

Microchip Technology Inc.  
35 Rockridge Road  
Englewood, OH 45322  
Tel: 513 832-2543 Fax: 513 832-2841

### Los Angeles

Microchip Technology Inc.  
18201 Von Karman, Suite 455  
Irvine, CA 92715  
Tel: 714 263-1888 Fax: 714 263-1338

### New York

Microchip Technology Inc.  
150 Motor Parkway, Suite 416  
Hauppauge, NY 11788  
Tel: 516 273-5305 Fax: 516 273-5335

## AMERICAS (continued)

### San Jose

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

## ASIA/PACIFIC

### Hong Kong

Microchip Technology  
Unit No. 3002-3004, Tower 1  
Metroplaza  
223 Hing Fong Road  
Kwai Fong, N.T. Hong Kong  
Tel: 852 2 401 1200 Fax: 852 2 401 3431

### Korea

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

### Singapore

Microchip Technology  
200 Middle Road  
#10-03 Prime Centre  
Singapore 188980  
Tel: 65 334 8870 Fax: 65 334 8850

### Taiwan

Microchip Technology  
10F-1C 207  
Tung Hua North Road  
Taipei, Taiwan, ROC  
Tel: 886 2 717 7175 Fax: 886 2 545 0139

## EUROPE

### United Kingdom

Arizona Microchip Technology Ltd.  
Unit 6, The Courtyard  
Meadow Bank, Furlong Road  
Bourne End, Buckinghamshire SL8 5AJ  
Tel: 44 0 1628 851077 Fax: 44 0 1628 850259

### France

Arizona Microchip Technology SARL  
2 Rue du Buisson aux Fraises  
91300 Massy - France  
Tel: 33 1 69 53 63 20 Fax: 33 1 69 30 90 79

### Germany

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

### Italy

Arizona Microchip Technology SRL  
Centro Direzionale Colleoni  
Palazzo Pegaso Ingresso No. 2  
Via Paracelso 23, 20041  
Agrate Brianza (MI) Italy  
Tel: 39 039 689 9939 Fax: 39 039 689 9883

## JAPAN

Microchip Technology Intl. Inc.  
Benex S-1 6F  
3-18-20, Shin Yokohama  
Kohoku-Ku, Yokohama  
Kanagawa 222 Japan  
Tel: 81 45 471 6166 Fax: 81 45 471 6122

9/22/95

All rights reserved. © 1995, Microchip Technology Incorporated, USA.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. 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. The Microchip logo and name are registered trademarks of Microchip Technology Inc. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.