Section 11. Timer0

HIGHLIGHTS

This section of the manual contains the following major topics:

11.1 Introduction

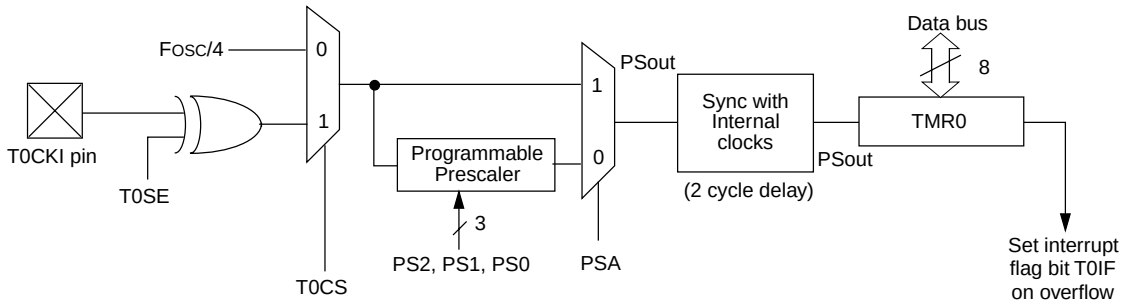
The Timer0 module has the following features:

- 8-bit timer/counter
- Readable and writable
- 8-bit software programmable prescaler
- Clock source selectable to be external or internal
- Interrupt on overflow from FFh to 00h
- Edge select for external clock

Note: To achieve a 1:1 prescaler assignment for the TMR0 register, assign the prescaler to the Watchdog Timer.

Figure 11-1 shows a simplified block diagram of the Timer0 module.

Figure 11-1: Timer0 Block Diagram



Note 1: T0CS, T0SE, PSA, PS2:PS0 (OPTION_REG<5:0>).
2: The prescaler is shared with Watchdog Timer (refer to Figure 11-6 for detailed block diagram).

11.2 Control Register

The OPTION_REG register is a readable and writable register which contains various control bits to configure the TMR0/WDT prescaler, the External INT Interrupt, TMR0, and the weak pull-ups on PORTB.

Note:	To achieve a 1:1 prescaler assignment for the TMR0 register, assign the prescaler to the Watchdog Timer.

Register 11-1: OPTION_REG Register

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$ (1)	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$ (1):** Weak Pull-up Enable bit
1 = Weak pull-ups are disabled
0 = Weak pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of INT pin
0 = Interrupt on falling edge of INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2:0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Legend		
R = Readable bit	W = Writable bit	
U = Unimplemented bit, read as '0'		- n = Value at POR reset

Note 1:	Some devices call this bit $\overline{\text{GPPU}}$. Devices that have the $\overline{\text{RBPU}}$ bit, have the weak pull-ups on PORTB, while devices that have the $\overline{\text{GPPU}}$ have the weak pull-ups on the GPIO Port.

11.3 Operation

Timer mode is selected by clearing the T0CS bit (OPTION<5>). In timer mode, the Timer0 module will increment every instruction cycle (without prescaler). If the TMR0 register is written, the increment is inhibited for the following two instruction cycles (Figure 11-2 and Figure 11-3). The user can work around this by writing an adjusted value to the TMR0 register.

Counter mode is selected by setting the T0CS bit (OPTION<5>). In counter mode, Timer0 will increment either on every rising or falling edge of the T0CKI pin. The incrementing edge is determined by the Timer0 Source Edge Select the T0SE bit (OPTION<4>). Clearing the T0SE bit selects the rising edge. Restrictions on the external clock input are discussed in detail in Subsection **11.5 "Using Timer0 with an External Clock"** .

The prescaler is mutually exclusively shared between the Timer0 module and the Watchdog Timer. The prescaler assignment is controlled in software by the PSA control bit (OPTION<3>). Clearing the PSA bit will assign the prescaler to the Timer0 module. The prescaler is not readable or writable. When the prescaler is assigned to the Timer0 module, prescale values of 1:2, 1:4,..., 1:256 are selectable. Subsection **11.6 "TMR0 Prescaler"** details the operation of the prescaler.

Any write to the TMR0 register will cause a 2 instruction cycle (2TCY) inhibit. That is, after the TMR0 register has been written with the new value, TMR0 will not be incremented until the third instruction cycle later (Figure 11-2). When the prescaler is assigned to the Timer0 module, any write to the TMR0 register will immediately update the TMR0 register and clear the prescaler. The incrementing of Timer0 (TMR0 and Prescaler) will also be inhibited 2 instruction cycles (TCY). So if the prescaler is configured as 2, then after a write to the TMR0 register TMR0 will not increment for 4 Timer0 clocks (Figure 11-3). After that, TMR0 will increment every prescaler number of clocks later.

Figure 11-2: Timer0 Timing: Internal Clock/No Prescale

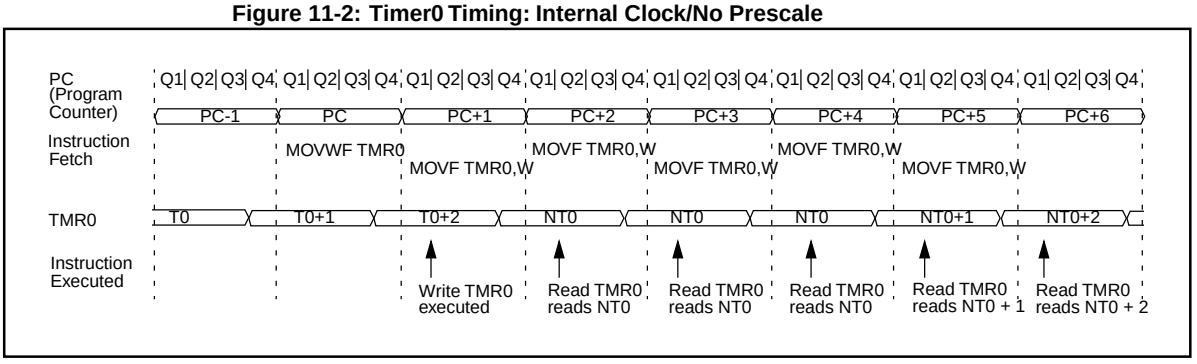
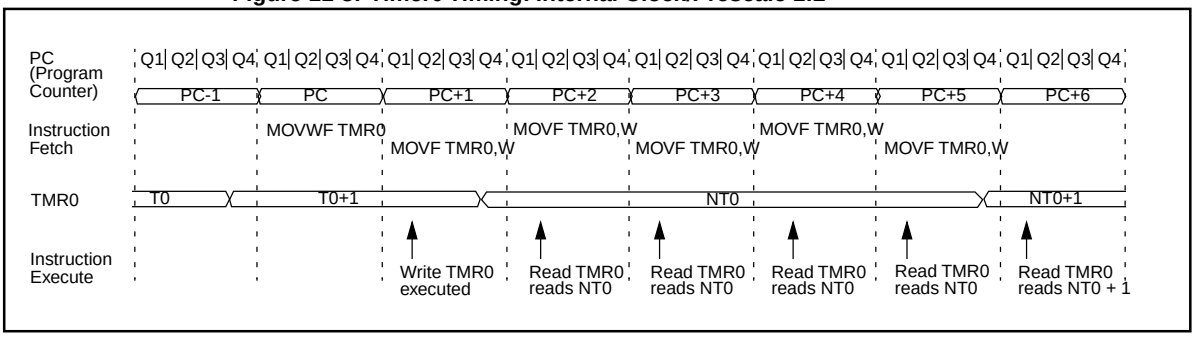


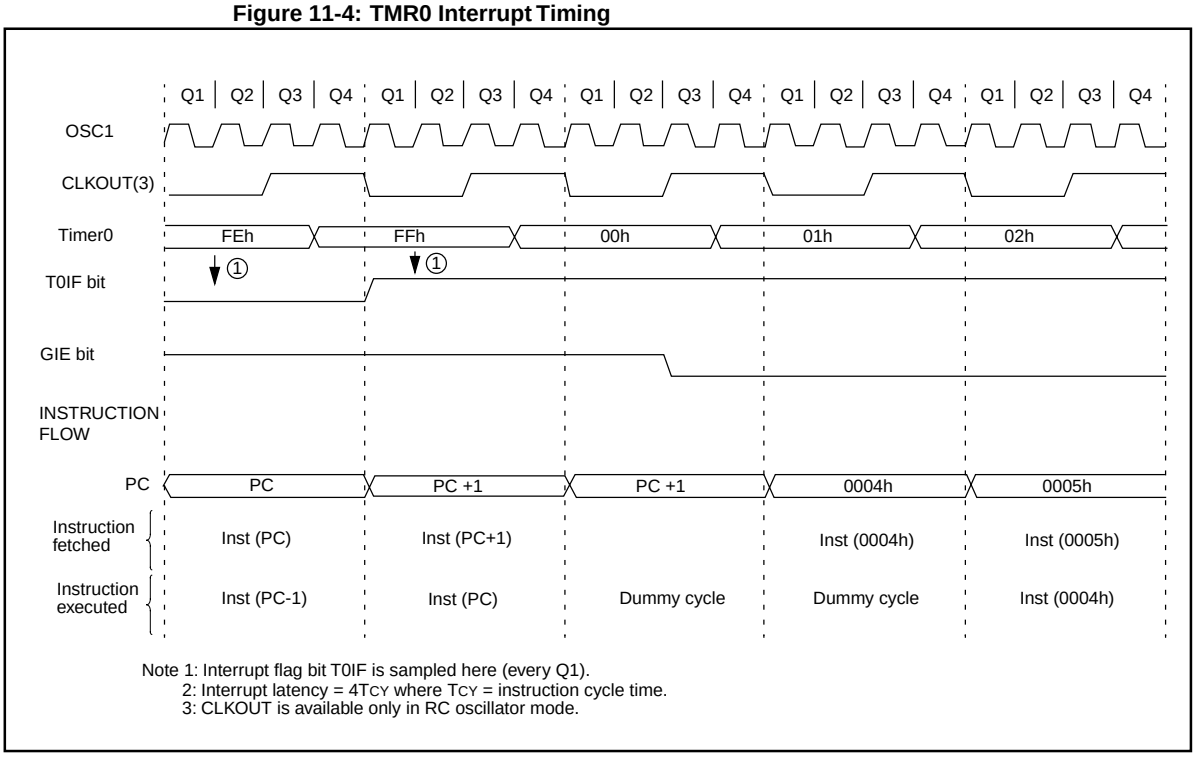
Figure 11-3: Timer0 Timing: Internal Clock/Prescale 1:2



11.4 TMR0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h. This overflow sets bit T0IF (INTCON<2>). The interrupt can be masked by clearing bit T0IE (INTCON<5>). Bit T0IF must be cleared in software by the Timer0 module interrupt service routine before re-enabling this interrupt. The TMR0 interrupt cannot awaken the processor from SLEEP since the timer is shut-off during SLEEP. See Figure 11-4 for Timer0 interrupt timing.

Figure 11-4: TMR0 Interrupt Timing



Note 1: Interrupt flag bit T0IF is sampled here (every Q1).
2: Interrupt latency = 4TCY where TCY = instruction cycle time.
3: CLKOUT is available only in RC oscillator mode.

11.5 Using Timer0 with an External Clock

When an external clock input is used for Timer0, it must meet certain requirements as detailed in **11.5.1 "External Clock Synchronization."** These requirements ensure the external clock can be synchronized with the internal phase clock (TOSC). Also, there is a delay in the actual incrementing of Timer0 after synchronization.

11.5.1 External Clock Synchronization

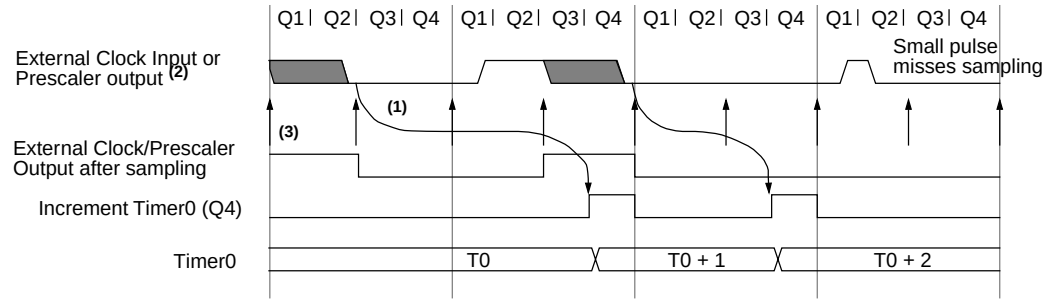
When no prescaler is used, the external clock input is the same as the prescaler output. The synchronization of T0CKI with the internal phase clocks is accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the internal phase clocks (Figure 11-5). Therefore, it is necessary for T0CKI to be high for at least 2Tosc (and a small RC delay of 20 ns) and low for at least 2Tosc (and a small RC delay of 20 ns). Refer to parameters 40, 41 and 42 in the electrical specification of the desired device.

When a prescaler is used, the external clock input is divided by an asynchronous ripple-counter type prescaler so that the prescaler output is symmetrical. For the external clock to meet the sampling requirement, the ripple-counter must be taken into account. Therefore, it is necessary for T0CKI to have a period of at least 4Tosc (and a small RC delay of 40 ns) divided by the prescaler value. The only requirement on T0CKI high and low time is that they do not violate the minimum pulse width requirement of 10 ns. Refer to parameters 40, 41 and 42 in the electrical specification of the desired device.

11.5.2 TMR0 Increment Delay

Since the prescaler output is synchronized with the internal clocks, there is a small delay from the time the external clock edge occurs to the time the Timer0 module is actually incremented. Figure 11-5 shows the delay from the external clock edge to the timer incrementing.

Figure 11-5: Timer0 Timing with External Clock



Note 1: Delay from clock input change to Timer0 increment is 3Tosc to 7Tosc. (Duration of Q = Tosc). Therefore, the error in measuring the interval between two edges on Timer0 input = ±4Tosc max.
2: External clock if no prescaler selected, Prescaler output otherwise.
3: The arrows indicate the points in time where sampling occurs.

11.6 TMR0 Prescaler

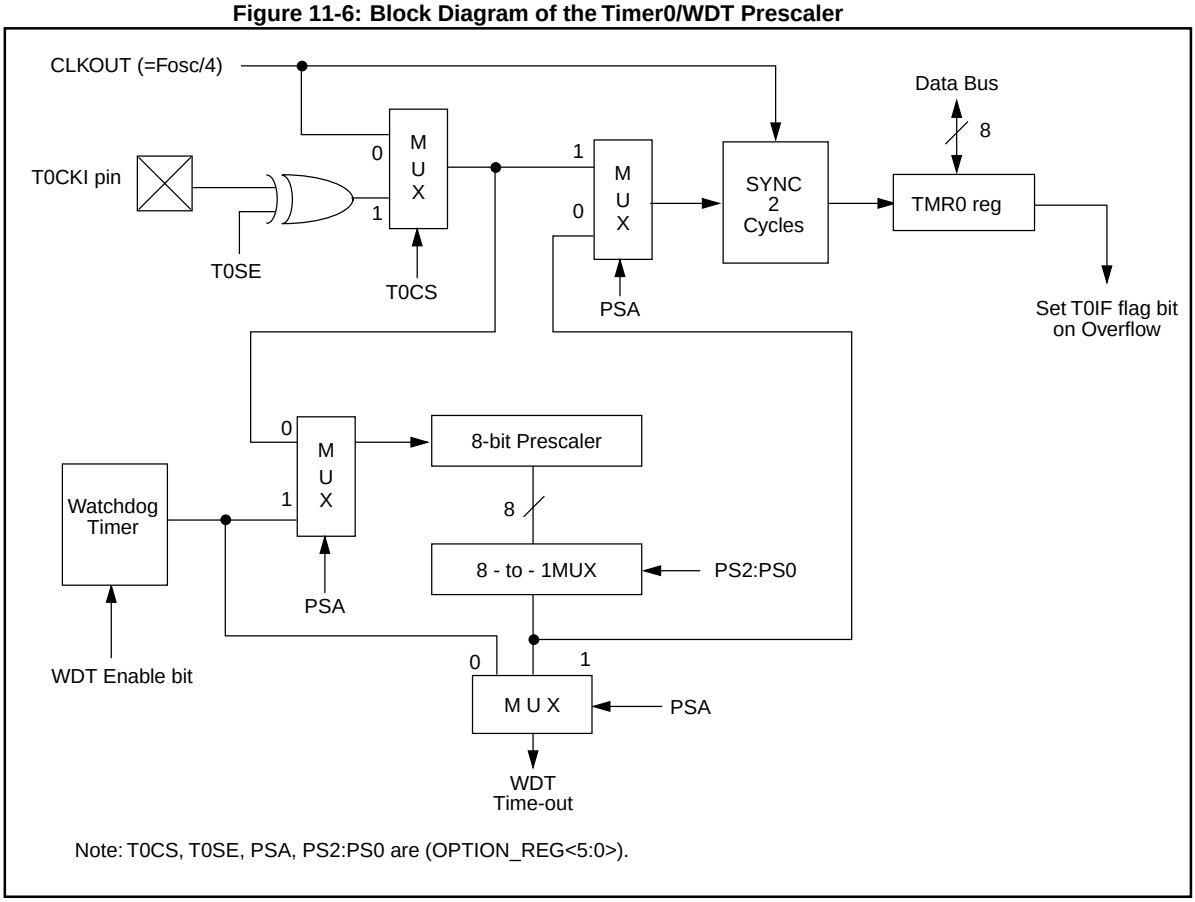
An 8-bit counter is available as a prescaler for the Timer0 module, or as a postscaler for the Watchdog Timer (Figure 11-6). For simplicity, this counter is being referred to as “prescaler” in the Timer0 description. Thus, a prescaler assignment for the Timer0 module means that there is no postscaler for the Watchdog Timer, and vice-versa.

Note:	There is only one prescaler available which is mutually exclusively shared between the Timer0 module and the Watchdog Timer.

The PSA and PS2:PS0 bits (OPTION<3:0>) determine the prescaler assignment and prescale ratio.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g., `CLRF TMR0, MOVWF TMR0, BSF TMR0,`x....etc.) will clear the prescaler. When assigned to WDT, a `CLRWDT` instruction will clear the prescaler along with the Watchdog Timer. The prescaler is not readable or writable.

Figure 11-6: Block Diagram of the Timer0/WDT Prescaler



Note: T0CS, T0SE, PSA, PS2:PS0 are (OPTION_REG<5:0>).

11.6.1 Switching Prescaler Assignment

The prescaler assignment is fully under software control, i.e., it can be changed "on the fly" during program execution.

Note:	To avoid an unintended device RESET, the following instruction sequence (shown in Example 11-1) must be executed when changing the prescaler assignment from Timer0 to the WDT. This sequence must be followed even if the WDT is disabled.

In Example 11-1, the first modification of the OPTION_REG does not need to be included if the final desired prescaler is other then 1:1. If the final prescaler value is to be 1:1, then a temporary prescale value is set (other than 1:!), and the final prescale value is set in the last modification of OPTION_REG.

Example 11-1: Changing Prescaler (Timer0→WDT)

Lines 2 and 3 do NOT have to be included if the final desired prescale value is other than 1:1. If 1:1 is final desired value, then a temporary prescale value is set in lines 2 and 3 and the final prescale value will be set in lines 10 and 11.

```
1)  BSF    STATUS, RP0   ;Bank1
2)  MOVLW  b'xx0x0xxx'   ;Select clock source and prescale value of
3)  MOVWF  OPTION_REG    ;other than 1:1
4)  BCF    STATUS, RP0   ;Bank0
5)  CLRF   TMR0          ;Clear TMR0 and prescaler
6)  BSF    STATUS, RP1   ;Bank1
7)  MOVLW  b'xxxx1xxx'   ;Select WDT, do not change prescale value
8)  MOVWF  OPTION_REG    ;
9)  CLRWDT               ;Clears WDT and prescaler
10) MOVLW  b'xxxx1xxx'   ;Select new prescale value and WDT
11) MOVWF  OPTION_REG    ;
12) BCF    STATUS, RP0   ;Bank0
```

To change prescaler from the WDT to the Timer0 module use the sequence shown in Example 11-2.

Example 11-2: Changing Prescaler (WDT→Timer0)

```
    CLRWDT                  ; Clear WDT and prescaler
    BSF      STATUS, RP0    ; Bank1
    MOVLW    b'xxxx0xxx'    ; Select TMR0, new prescale
    MOVWF    OPTION_REG     ; value and clock source
    BCF      STATUS, RP0    ; Bank0
```

11.6.2 Initialization

Since Timer0 has a software programmable clock source, there are two examples to show the initialization of Timer0 with each source. Example 11-3 shows the initialization for the internal clock source (timer mode), while Example 11-4 shows the initialization for the external clock source (counter mode).

Example 11-3: Timer0 Initialization (Internal Clock Source)

```
        CLRF    TMR0            ; Clear Timer0 register
        CLRF    INTCON          ; Disable interrupts and clear T0IF
        BSF     STATUS, RP0     ; Bank1
        MOVLW   0xC3            ; PortB pull-ups are disabled,
        MOVWF   OPTION_REG      ;   Interrupt on rising edge of RB0
                                ;   Timer0 increment from internal clock
                                ;   with a prescaler of 1:16.
        BCF     STATUS, RP0     ; Bank0
;**     BSF     INTCON, T0IE    ; Enable TMR0 interrupt
;**     BSF     INTCON, GIE     ; Enable all interrupts
;
; The TMR0 interrupt is disabled, do polling on the overflow bit
;
T0_OVFL_WAIT
        BTFSS   INTCON, T0IF
        GOTO    T0_OVFL_WAIT
; Timer has overflowed
```

Example 11-4: Timer0 Initialization (External Clock Source)

```
    CLRF    TMR0            ; Clear Timer0 register
    CLRF    INTCON          ; Disable interrupts and clear T0IF
    BSF     STATUS, RP0     ; Bank1
    MOVLW   0x37            ; PortB pull-ups are enabled,
    MOVWF   OPTION_REG      ;   Interrupt on falling edge of RB0
                            ;   Timer0 increment from external clock
                            ;   on the high-to-low transition of T0CKI
                            ;   with a prescaler of 1:256.
    BCF     STATUS, RP0     ; Bank0
;** BSF     INTCON, T0IE    ; Enable TMR0 interrupt
;** BSF     INTCON, GIE     ; Enable all interrupts
;
; The TMR0 interrupt is disabled, do polling on the overflow bit
;
T0_OVFL_WAIT
    BTFSS   INTCON, T0IF
    GOTO    T0_OVFL_WAIT
; Timer has overflowed
```

11.7 Design Tips

Question 1: ***I am implementing a counter/clock, but the clock loses time or is inaccurate.***

Answer 1:

If you are polling TMR0 to see if it has rolled over to zero. You could do this by executing:

```
wait    MOVF    TMR0,W      ; read the timer into W
        BTFSS   STATUS,Z    ; see if it was zero, if so,
                            ;    break from loop
        GOTO     wait       ; if not zero yet, keep waiting
```

Two possible scenarios to lose clock cycles are:

1. If you are incrementing TMR0 from the internal instruction clock, or an external source that is about as fast, the overflow could occur during the two cycle GOTO, so you could miss it. In this case the TMR0 source should be prescaled.
   Or you could do a test to see if it has rolled over by checking for less than a nominal value:

   ```
   Wait    movlw  3
           subwf  TMR0,W
           btfsc  STATUS,C
           goto   Wait
   ```

2. When writing to TMR0, two instruction clock cycles are lost. Often you have a specific time period you want to count, say 100 decimal. In that case you might put 156 into TMR0 (256 - 100 = 156). However, since two instruction cycles are lost when you write to TMR0 (for internal logic synchronization), you should actually write 158 to the timer.

11.8 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to Timer0 are:

Title	Application Note #
Frequency Counter Using PIC16C5X	AN592
A Clock Design using the PIC16C54 for LED Display and Switch Inputs	AN590

11.9 Revision History

Revision A

This is the initial released revision of the Timer0 Module description.