



MICROCHIP

Section 11. Timer0

HIGHLIGHTS

This section of the manual contains the following major topics:

11.1	Introduction	11-2
11.2	Control Register	11-3
11.3	Operation	11-4
11.4	TMR0 Interrupt.....	11-5
11.5	Using Timer0 with an External Clock	11-6
11.6	TMR0 Prescaler	11-7
11.7	Design Tips	11-10
11.8	Related Application Notes.....	11-11
11.9	Revision History	11-12

PICmicro MID-RANGE MCU FAMILY

11.1 Introduction

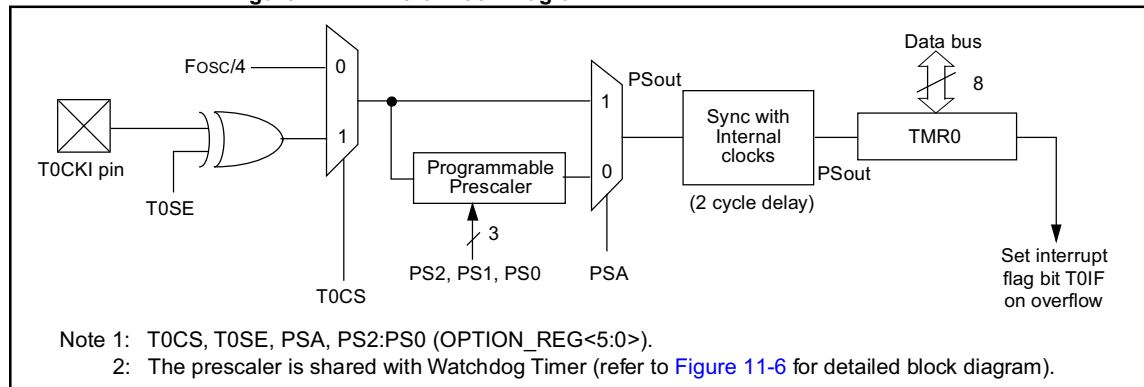
The Timer0 module has the following features:

- 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



Section 11. Timer0

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
	$\overline{\text{RBP}}\text{U}^{(1)}$	INTE $\overline{\text{D}}\text{G}$	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7								bit 0

- bit 7 **$\overline{\text{RBP}}\text{U}^{(1)}$:** Weak Pull-up Enable bit
 1 = Weak pull-ups are disabled
 0 = Weak pull-ups are enabled by individual port latch values
- bit 6 **INTE $\overline{\text{D}}\text{G}$:** 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 $\overline{\text{GPP}}\text{U}$. Devices that have the $\overline{\text{RBP}}\text{U}$ bit, have the weak pull-ups on PORTB, while devices that have the $\overline{\text{GPP}}\text{U}$ have the weak pull-ups on the GPIO Port.

PICmicro MID-RANGE MCU FAMILY

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

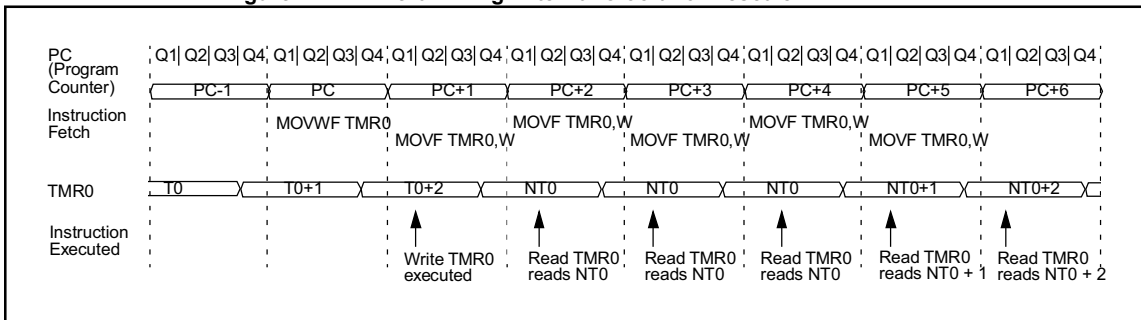
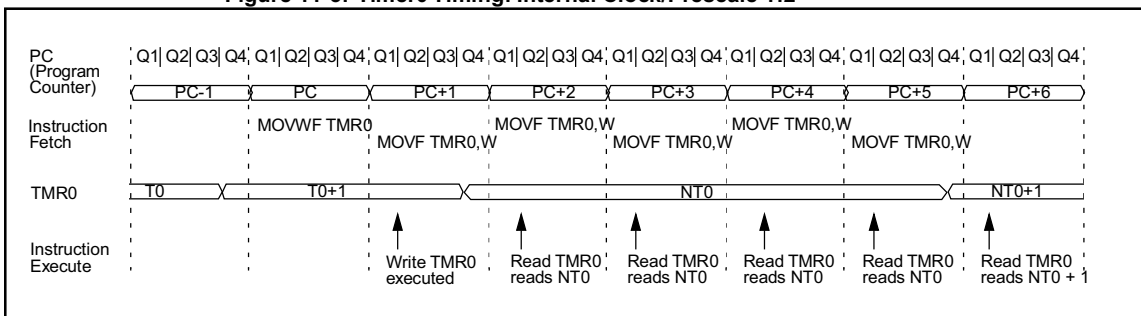


Figure 11-3: Timer0 Timing: Internal Clock/Prescale 1:2

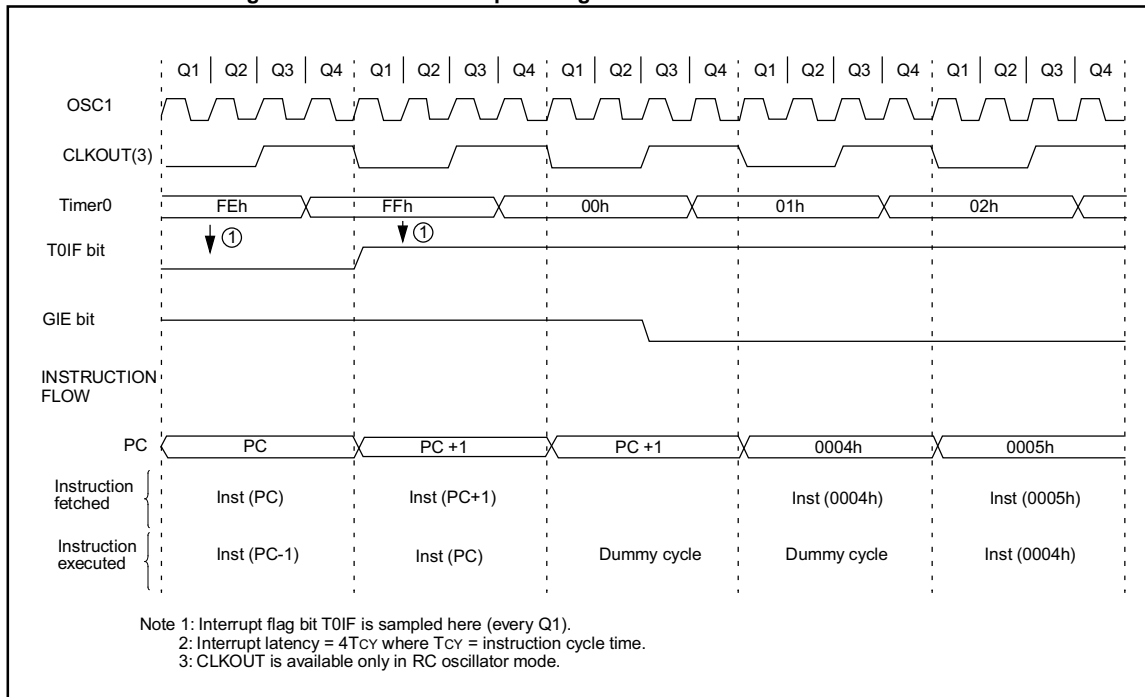


Section 11. Timer0

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



PICmicro MID-RANGE MCU FAMILY

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

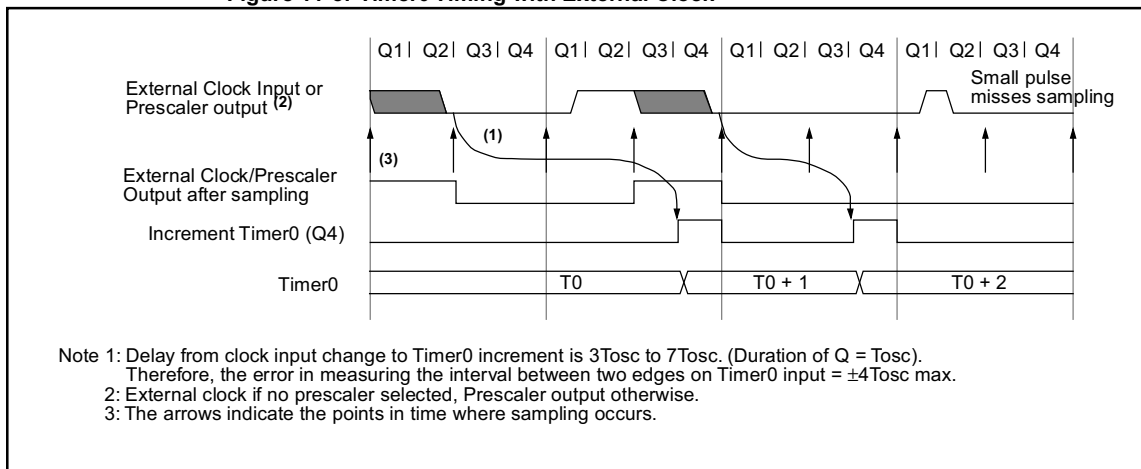
When no prescaler is used, the external clock input is the same as the prescaler output. The synchronization of T0CKI with the internal phase clocks is accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the internal phase clocks (Figure 11-5). Therefore, it is necessary for T0CKI 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



Section 11. Timer0

11.6 TMR0 Prescaler

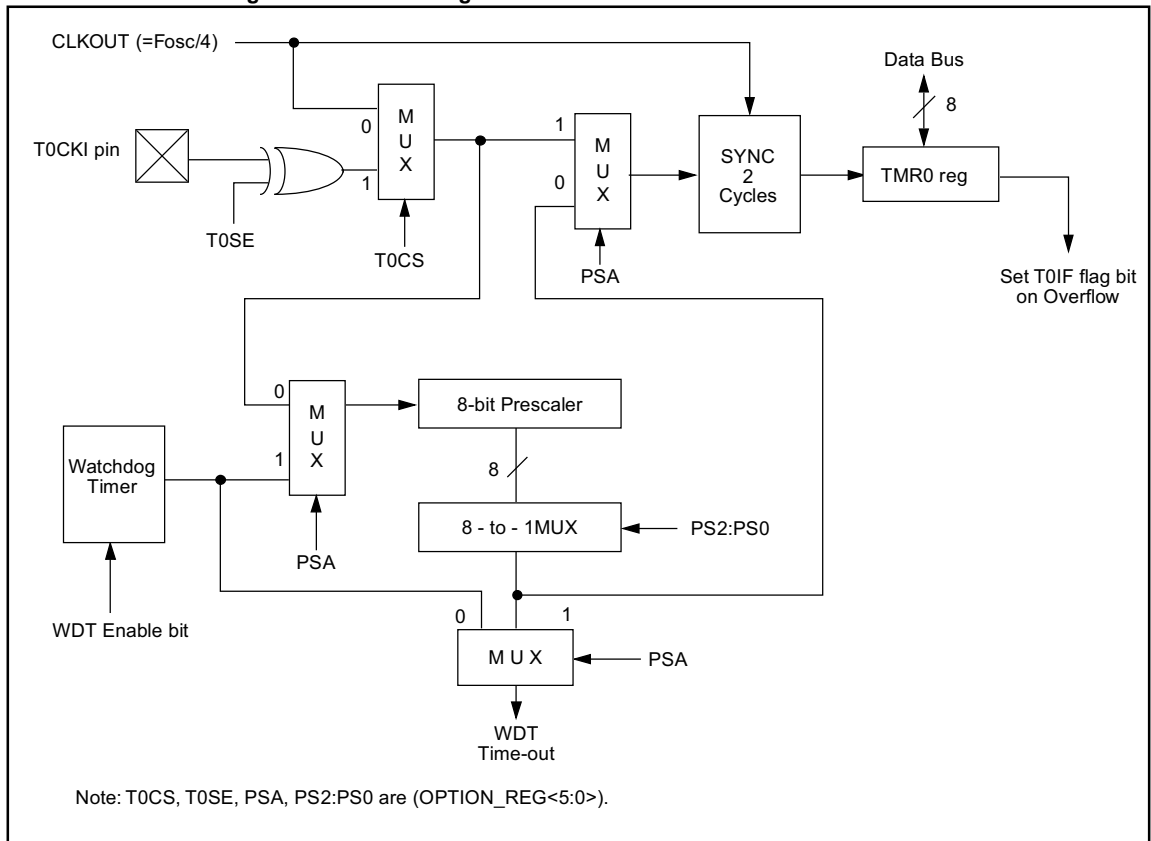
An 8-bit counter is available as a prescaler for the Timer0 module, or as a postscaler for the Watchdog Timer (Figure 11-6). For simplicity, this counter is being referred to as “prescaler” in the Timer0 description. Thus, a prescaler assignment for the Timer0 module means that there is no postscaler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



PICmicro MID-RANGE MCU FAMILY

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 than 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 TOIF
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, TOIE ; 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, TOIF
      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 TOIF
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, TOIE ; 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, TOIF
      GOTO  T0_OVFL_WAIT
; Timer has overflowed
```

PICmicro MID-RANGE MCU FAMILY

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.

Section 11. Timer0

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

PICmicro MID-RANGE MCU FAMILY

11.9 Revision History

Revision A

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



MICROCHIP

Section 12. Timer1

HIGHLIGHTS

This section of the manual contains the following major topics:

12.1	Introduction	12-2
12.2	Control Register	12-3
12.3	Timer1 Operation in Timer Mode	12-4
12.4	Timer1 Operation in Synchronized Counter Mode.....	12-4
12.5	Timer1 Operation in Asynchronous Counter Mode.....	12-5
12.6	Timer1 Oscillator.....	12-7
12.7	Sleep Operation	12-9
12.8	Resetting Timer1 Using a CCP Trigger Output	12-9
12.9	Resetting of Timer1 Register Pair (TMR1H:TMR1L).....	12-9
12.10	Timer1 Prescaler.....	12-9
12.11	Initialization	12-10
12.12	Design Tips	12-12
12.13	Related Application Notes.....	12-13
12.14	Revision History	12-14

PICmicro MID-RANGE MCU FAMILY

12.1 Introduction

The Timer1 module is a 16-bit timer/counter consisting of two 8-bit registers (TMR1H and TMR1L) which are readable and writable. The TMR1 Register pair (TMR1H:TMR1L) increments from 0000h to FFFFh and rolls over to 0000h. The Timer1 Interrupt, if enabled, is generated on overflow which is latched in the TMR1IF interrupt flag bit. This interrupt can be enabled/disabled by setting/clearing the TMR1IE interrupt enable bit.

Timer1 can operate in one of three modes:

- As a synchronous timer
- As a synchronous counter
- As an asynchronous counter

The operating mode is determined by clock select bit, TMR1CS (T1CON<1>), and the synchronization bit, $\overline{T1SYNC}$ (Figure 12-1).

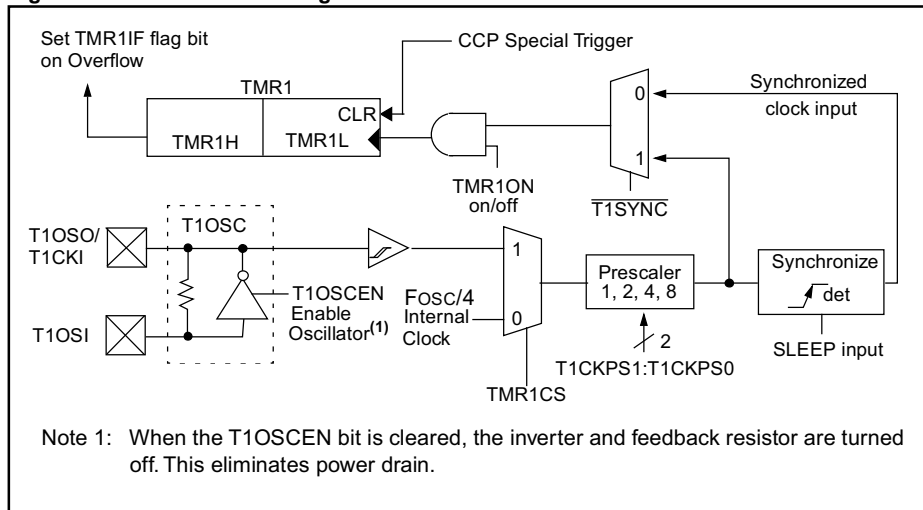
In timer mode, Timer1 increments every instruction cycle. In counter mode, it increments on every rising edge of the external clock input on pin T1CKI.

Timer1 can be turned on and off using the TMR1ON control bit (T1CON<0>).

Timer1 also has an internal "reset input", which can be generated by a CCP module.

Timer1 has the capability to operate off an external crystal. When the Timer1 oscillator is enabled (T1OSCEN is set), the T1OSI and T1OSO pins become inputs. That is, their corresponding TRIS values are ignored.

Figure 12-1: Timer1 Block Diagram



Section 12. Timer1

12.2 Control Register

Register 12-1 shows the Timer1 control register.

Register 12-1: T1CON: Timer1 Control Register

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	
bit 7								bit 0

- bit 7:6 **Unimplemented:** Read as '0'
- bit 5:4 **T1CKPS1:T1CKPS0:** Timer1 Input Clock Prescale Select bits
 - 11 = 1:8 Prescale value
 - 10 = 1:4 Prescale value
 - 01 = 1:2 Prescale value
 - 00 = 1:1 Prescale value
- bit 3 **T1OSCEN:** Timer1 Oscillator Enable bit
 - 1 = Oscillator is enabled
 - 0 = Oscillator is shut off. The oscillator inverter and feedback resistor are turned off to eliminate power drain
- bit 2 **T1SYNC:** Timer1 External Clock Input Synchronization Select bit
 - When TMR1CS = 1:
 - 1 = Do not synchronize external clock input
 - 0 = Synchronize external clock input
 - When TMR1CS = 0:
 - This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.
- bit 1 **TMR1CS:** Timer1 Clock Source Select bit
 - 1 = External clock from pin T1OSO/T1CKI (on the rising edge)
 - 0 = Internal clock (FOSC/4)
- bit 0 **TMR1ON:** Timer1 On bit
 - 1 = Enables Timer1
 - 0 = Stops Timer1

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

PICmicro MID-RANGE MCU FAMILY

12.3 Timer1 Operation in Timer Mode

Timer mode is selected by clearing the TMR1CS (T1CON<1>) bit. In this mode, the input clock to the timer is $F_{osc}/4$. The synchronize control bit, $\overline{T1SYNC}$ (T1CON<2>), has no effect since the internal clock is always synchronized.

12.4 Timer1 Operation in Synchronized Counter Mode

Counter mode is selected by setting the TMR1CS bit. In this mode the timer increments on every rising edge of clock input on the T1OSI pin when the oscillator enable bit (T1OSCEN) is set, or the T1OSO/T1CKI pin when the T1OSCEN bit is cleared.

If the $\overline{T1SYNC}$ bit is cleared, then the external clock input is synchronized with internal phase clocks. The synchronization is done after the prescaler stage. The prescaler is an asynchronous ripple-counter.

In this configuration, during SLEEP mode, Timer1 will not increment even if the external clock is present, since the synchronization circuit is shut off. The prescaler however will continue to increment.

12.4.1 External Clock Input Timing for Synchronized Counter Mode

When an external clock input is used for Timer1 in synchronized counter mode, it must meet certain requirements. The external clock requirement is due to internal phase clock (T_{osc}) synchronization. Also, there is a delay in the actual incrementing of TMR1 after synchronization.

When the prescaler is 1:1, the external clock input is the same as the prescaler output. The synchronization of T1CKI with the internal phase clocks is accomplished by sampling the prescaler output on alternating T_{osc} clocks of the internal phase clocks. Therefore, it is necessary for the T1CKI pin to be high for at least $2T_{osc}$ (and a small RC delay) and low for at least $2T_{osc}$ (and a small RC delay). Refer to [parameters 45, 46, and 47](#) in the “**Electrical Specifications**” section.

When a prescaler other than 1:1 is used, the external clock input is divided by the asynchronous ripple-counter prescaler so that the prescaler output is symmetrical. In order for the external clock to meet the sampling requirement, the ripple-counter must be taken into account. Therefore, it is necessary for the T1CKI pin to have a period of at least $4T_{osc}$ (and a small RC delay) divided by the prescaler value. Another requirement on the T1CKI pin high and low time is that they do not violate the minimum pulse width requirements). Refer to [parameters 40, 42, 45, 46, and 47](#) in the “**Electrical Specifications**” section.

Section 12. Timer1

12.5 Timer1 Operation in Asynchronous Counter Mode

If $\overline{T1SYNC}$ (T1CON<2>) is set, the external clock input is not synchronized. The timer continues to increment asynchronously to the internal phase clocks. The timer will continue to run during SLEEP and can generate an interrupt on overflow which will wake-up the processor. However, special precautions in software are needed to read/write the timer (Subsection 12.5.2 “Reading and Writing Timer1 in Asynchronous Counter Mode”). Since the counter can operate in sleep, Timer1 can be used to implement a true real-time clock.

In asynchronous counter mode, Timer1 cannot be used as a time-base for capture or compare operations.

12.5.1 External Clock Input Timing with Unsynchronized Clock

If the $\overline{T1SYNC}$ control bit is set, the timer will increment completely asynchronously. The input clock must meet certain minimum high time and low time requirements. Refer to the Device Data Sheet Electrical Specifications Section, timing parameters 45, 46, and 47.

12.5.2 Reading and Writing Timer1 in Asynchronous Counter Mode

Reading TMR1H or TMR1L while the timer is running from an external asynchronous clock, will guarantee a valid read (taken care of in hardware). However, the user should keep in mind that reading the 16-bit timer in two 8-bit values itself poses certain problems since the timer may overflow between the reads.

For writes, it is recommended that the user simply stop the timer and write the desired values. A write contention may occur by writing to the timer registers while the register is incrementing. This may produce an unpredictable value in the timer register.

Reading the 16-bit value requires some care, since two separate reads are required to read the entire 16-bits. Example 12-1 shows why this may not be a straight forward read of the 16-bit register.

Example 12-1: Reading 16-bit Register Issues

TMR1	Sequence 1		Sequence 2	
	Action	TMPH:TMPL	Action	TMPH:TMPL
04FFh	READ TMR1L	xxxxh	READ TMR1H	xxxxh
0500h	Store in TMPL	xxFFh	Store in TMPH	04xxh
0501h	READ TMR1H	xxFFh	READ TMR1L	04xxh
0502h	Store in TMPH	05FFh	Store in TMPL	0401h

PICmicro MID-RANGE MCU FAMILY

[Example 12-2](#) shows a routine to read the 16-bit timer value with experiencing the issues shown in [Example 12-1](#). This is useful if the timer cannot be stopped.

Example 12-2: Reading a 16-bit Free-Running Timer

```
; All interrupts are disabled
MOVF  TMR1H, W    ; Read high byte
MOVWF TMPH       ;
MOVF  TMR1L, W    ; Read low byte
MOVWF TMPL       ;
MOVF  TMR1H, W    ; Read high byte
SUBWF TMPH, W    ; Sub 1st read with 2nd read
BTFSC STATUS,Z   ; Is result = 0
GOTO  CONTINUE   ; Good 16-bit read

;
; TMR1L may have rolled over between the read of the high and low bytes.
; Reading the high and low bytes now will read a good value.
;
MOVF  TMR1H, W    ; Read high byte
MOVWF TMPH       ;
MOVF  TMR1L, W    ; Read low byte
MOVWF TMPL       ;
; Re-enable the Interrupt (if required)
CONTINUE         ; Continue with your code
```

Writing a 16-bit value to the 16-bit TMR1 register is straight forward. First the TMR1L register is cleared to ensure that there are many Timer1 clock/oscillator cycles before there is a rollover into the TMR1H register. The TMR1H register is then loaded, and finally the TMR1L register is loaded. [Example 12-3](#) shows this:

Example 12-3: Writing a 16-bit Free Running Timer

```
; All interrupts are disabled
CLRF  TMR1L       ; Clear Low byte, Ensures no
                  ; rollover into TMR1H
MOVLW HI_BYTE    ; Value to load into TMR1H
MOVWF TMR1H, F   ; Write High byte
MOVLW LO_BYTE    ; Value to load into TMR1L
MOVWF TMR1H, F   ; Write Low byte
; Re-enable the Interrupt (if required)
CONTINUE         ; Continue with your code
```

Section 12. Timer1

12.6 Timer1 Oscillator

A crystal oscillator circuit is built in between pins T1OSI (input) and T1OSO (amplifier output). It is enabled by setting the T1OSCEN control bit (T1CON<3>). The oscillator is a low power oscillator, rated up to 200 kHz operation. It will continue to run during SLEEP. It is primarily intended for a 32 kHz crystal, which is an ideal frequency for real-time keeping. Table 12-1 shows the capacitor selection for the Timer1 oscillator.

The Timer1 oscillator is identical to the LP oscillator. The user must provide a software time delay to ensure proper oscillator start-up.

Note: This allows the counter to operate (increment) when the device is in sleep mode, which allows Timer1 to be used as a real-time clock.

Table 12-1: Capacitor Selection for the Timer1 Oscillator

Osc Type	Freq	C1	C2
LP	32 kHz	33 pF	33 pF
	100 kHz	15 pF	15 pF
	200 kHz	15 pF	15 pF
Crystals Tested:			
32.768 kHz	Epson C-001R32.768K-A		± 20 PPM
100 kHz	Epson C-2100.00 KC-P		± 20 PPM
200 kHz	STD XTL 200.000 kHz		± 20 PPM

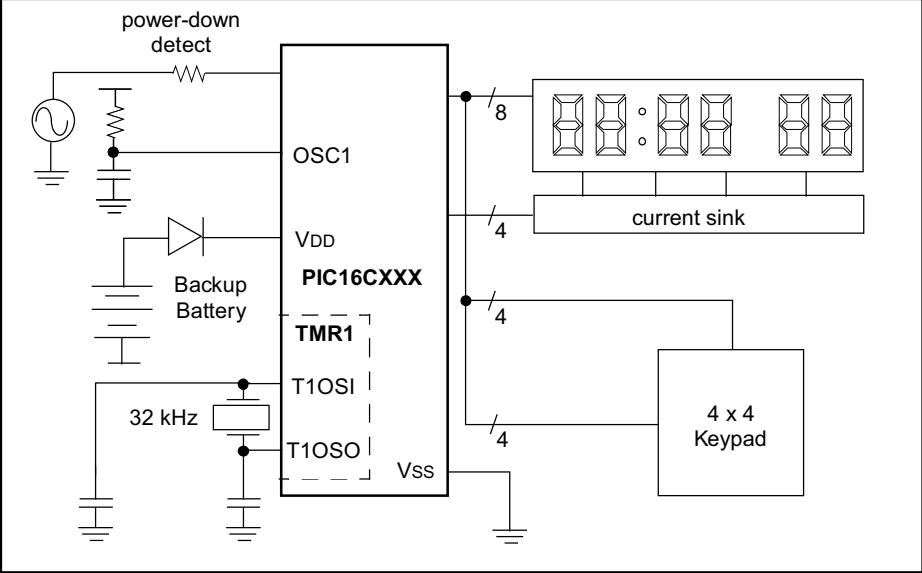
- Note 1: Higher capacitance increases the stability of oscillator but also increases the start-up time.
- 2: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

PICmicro MID-RANGE MCU FAMILY

12.6.1 Typical Application

This feature is typically used in applications where real-time needs to be kept, but it is also desirable to have the lowest possible power consumption. The Timer1 oscillator allows the device to be placed in sleep, while the timer continues to increment. When Timer1 overflows the interrupt could wake-up the device so that the appropriate registers could be updated.

Figure 12-2: Timer1 Application



Section 12. Timer1

12.7 Sleep Operation

When Timer1 is configured for asynchronous operation, the TMR1 registers will continue to increment for each timer clock (or prescale multiple of clocks). When the TMR1 register overflows, the TMR1IF bit will get set, and if enabled generate an interrupt that will wake the processor from sleep mode.

The Timer1 oscillator will add a delta current, due to the operation of this circuitry. That is, the power-down current will no longer only be the leakage current of the device, but also the active current of the Timer1 oscillator and other Timer1 circuitry.

12.8 Resetting Timer1 Using a CCP Trigger Output

If a CCP module is configured in compare mode to generate a “special event trigger” (CCP1M3:CCP1M0 = 1011), this signal resets Timer1.

Note: The special event trigger from the CCP module does not set interrupt flag bit TMR1IF.

Timer1 must be configured for either timer or synchronized counter mode to take advantage of the special event trigger feature. If Timer1 is running in asynchronous counter mode, this reset operation may not work, and should not be used.

In the event that a write to Timer1 coincides with a special event trigger from the CCP module, the write will take precedence.

In this mode of operation, the CCPRxH:CCPRxL register pair effectively becomes the period register for Timer1.

12.9 Resetting of Timer1 Register Pair (TMR1H:TMR1L)

TMR1H and TMR1L registers are not reset on a POR or any other reset, only by the CCP special event triggers.

T1CON register is reset to 00h on a Power-on Reset or a Brown-out Reset. In any other reset, the register is unaffected.

12.10 Timer1 Prescaler

The prescaler counter is cleared on writes to the TMR1H or TMR1L registers.

Table 12-2: Registers Associated with Timer1 as a Timer/Counter

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
INTCON	GIE	PEIE	TOIE	INTE	RBIE ⁽²⁾	TOIF	INTF	RBIF ⁽²⁾	0000 000x	0000 000u
PIR	TMR1IF ⁽¹⁾								0	0
PIE	TMR1IE ⁽¹⁾								0	0
TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	--uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'.

Shaded cells are not used by the Timer1 module.

Note 1: The placement of this bit is device dependent.

2: These bits may also be named GPIE and GPIF.

PICmicro MID-RANGE MCU FAMILY

12.11 Initialization

Since Timer1 has a software programmable clock source, there are three examples to show the initialization of each mode. [Example 12-4](#) shows the initialization for the internal clock source, [Example 12-5](#) shows the initialization for the external clock source, and [Example 12-6](#) shows the initialization of the external oscillator mode.

Example 12-4: Timer1 Initialization (Internal Clock Source)

```
CLRF  T1CON      ; Stop Timer1, Internal Clock Source,
                ; T1 oscillator disabled, prescaler = 1:1
CLRF  TMR1H     ; Clear Timer1 High byte register
CLRF  TMR1L     ; Clear Timer1 Low byte register
CLRF  INTCON    ; Disable interrupts
BSF   STATUS, RP0 ; Bank1
CLRF  PIE1      ; Disable peripheral interrupts
BCF   STATUS, RP0 ; Bank0
CLRF  PIR1      ; Clear peripheral interrupts Flags
MOVLW 0x30      ; Internal Clock source with 1:8 prescaler
MOVWF T1CON     ; Timer1 is stopped and T1 osc is disabled
BSF   T1CON, TMR1ON ; Timer1 starts to increment

;
; The Timer1 interrupt is disabled, do polling on the overflow bit
;
T1_OVFL_WAIT
BTFSS PIR1, TMR1IF
GOTO  T1_OVFL_WAIT

;
; Timer has overflowed
;
BCF   PIR1, TMR1IF
```

Example 12-5: Timer1 Initialization (External Clock Source)

```
CLRF  T1CON      ; Stop Timer1, Internal Clock Source,
                ; T1 oscillator disabled, prescaler = 1:1
CLRF  TMR1H     ; Clear Timer1 High byte register
CLRF  TMR1L     ; Clear Timer1 Low byte register
CLRF  INTCON    ; Disable interrupts
BSF   STATUS, RP0 ; Bank1
CLRF  PIE1      ; Disable peripheral interrupts
BCF   STATUS, RP0 ; Bank0
CLRF  PIR1      ; Clear peripheral interrupts Flags
MOVLW 0x32      ; External Clock source with 1:8 prescaler
MOVWF T1CON     ; Clock source is synchronized to device
BSF   T1CON, TMR1ON ; Timer1 starts to increment

;
; The Timer1 interrupt is disabled, do polling on the overflow bit
;
T1_OVFL_WAIT
BTFSS PIR1, TMR1IF
GOTO  T1_OVFL_WAIT

;
; Timer has overflowed
;
BCF   PIR1, TMR1IF
```

Section 12. Timer1

Example 12-6: Timer1 Initialization (External Oscillator Clock Source)

```
CLRF  T1CON      ; Stop Timer1, Internal Clock Source,
                  ; T1 oscillator disabled, prescaler = 1:1
CLRF  TMR1H      ; Clear Timer1 High byte register
CLRF  TMR1L      ; Clear Timer1 Low byte register
CLRF  INTCON     ; Disable interrupts
BSF   STATUS, RP0 ; Bank1
CLRF  PIE1       ; Disable peripheral interrupts
BCF   STATUS, RP0 ; Bank0
CLRF  PIR1       ; Clear peripheral interrupts Flags
MOVLW 0x3E       ; External Clock source with oscillator
MOVWF T1CON      ; circuitry, 1:8 prescaler, Clock source
                  ; is asynchronous to device
                  ; Timer1 is stopped
BSF   T1CON, TMR1ON ; Timer1 starts to increment
;
; The Timer1 interrupt is disabled, do polling on the overflow bit
;
T1_OVFL_WAIT
  BTFSS PIR1, TMR1IF
  GOTO  T1_OVFL_WAIT
;
; Timer has overflowed
;
  BCF   PIR1, TMR1IF
```

PICmicro MID-RANGE MCU FAMILY

12.12 Design Tips

Question 1: *Timer1 does not seem to be keeping accurate time.*

Answer 1:

There are a few reasons that this could occur

1. You should never write to Timer1, where that could cause the loss of time. In most cases that means you should not write to the TMR1L register, but if the conditions are ok, you may write to the TMR1H register. Normally you write to the TMR1H register if you want the Timer1 overflow interrupt to be sooner than the full 16-bit time-out.
2. You should ensure the your layout uses good PCB layout techniques so that noise does not couple onto the Timer1 oscillator lines.

Section 12. Timer1

12.13 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 Timer1 are:

Title	Application Note #
Using Timer1 in Asynchronous Clock Mode	AN580
Low Power Real Time Clock	AN582
Yet another Clock using the PIC16C92X	AN649

PICmicro MID-RANGE MCU FAMILY

12.14 Revision History

Revision A

This is the initial released revision of the Timer1 module description.



MICROCHIP

Section 13. Timer2

HIGHLIGHTS

This section of the manual contains the following major topics:

13.1	Introduction	13-2
13.2	Control Register	13-3
13.3	Timer Clock Source.....	13-4
13.4	Timer (TMR2) and Period (PR2) Registers.....	13-4
13.5	TMR2 Match Output.....	13-4
13.6	Clearing the Timer2 Prescaler and Postscaler.....	13-4
13.7	Sleep Operation	13-4
13.8	Initialization	13-5
13.9	Design Tips	13-6
13.10	Related Application Notes.....	13-7
13.11	Revision History	13-8

PICmicro MID-RANGE MCU FAMILY

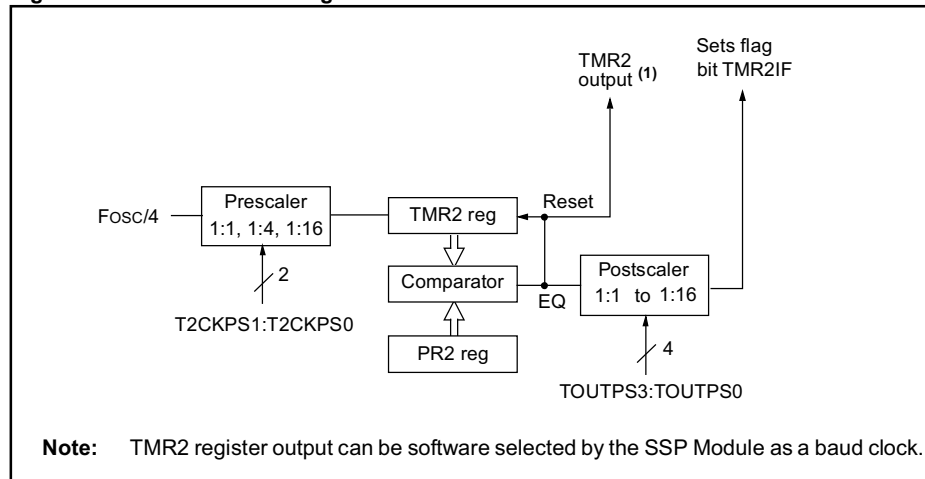
13.1 Introduction

Timer2 is an 8-bit timer with a prescaler, a postscaler, and a period register. Using the prescaler and postscaler at their maximum settings, the overflow time is the same as a 16-bit timer.

Timer2 is the PWM time-base when the CCP module(s) is used in the PWM mode.

Figure 13-1 shows a block diagram of Timer2. The postscaler counts the number of times that the TMR2 register matched the PR2 register. This can be useful in reducing the overhead of the interrupt service routine on the CPU performance.

Figure 13-1: Timer2 Block Diagram



Section 13. Timer2

13.2 Control Register

Register 13-1 shows the Timer2 control register.

Register 13-1: T2CON: Timer2 Control Register

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

- bit 7 **Unimplemented:** Read as '0'
- bit 6:3 **TOUTPS3:TOUTPS0:** Timer2 Output Postscale Select bits
 - 0000 = 1:1 Postscale
 - 0001 = 1:2 Postscale
 -
 -
 -
 - 1111 = 1:16 Postscale
- bit 2 **TMR2ON:** Timer2 On bit
 - 1 = Timer2 is on
 - 0 = Timer2 is off
- bit 1:0 **T2CKPS1:T2CKPS0:** Timer2 Clock Prescale Select bits
 - 00 = Prescaler is 1
 - 01 = Prescaler is 4
 - 1x = Prescaler is 16

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

PICmicro MID-RANGE MCU FAMILY

13.3 Timer Clock Source

The Timer2 module has one source of input clock, the device clock ($F_{osc}/4$). A prescale option of 1:1, 1:4 or 1:16 is software selected by control bits T2CKPS1:T2CKPS0 (T2CON<1:0>).

13.4 Timer (TMR2) and Period (PR2) Registers

The TMR2 register is readable and writable, and is cleared on all device resets. Timer2 increments from 00h until it matches PR2 and then resets to 00h on the next increment cycle. PR2 is a readable and writable register.

TMR2 is cleared when a WDT, POR, \overline{MCLR} , or a BOR reset occurs, while the PR2 register is set.

Timer2 can be shut off (disabled from incrementing) by clearing the TMR2ON control bit (T2CON<2>). This minimizes the power consumption of the module.

13.5 TMR2 Match Output

The match output of TMR2 goes to two sources:

1. Timer2 Postscaler
2. SSP Clock Input

There are four bits which select the postscaler. This allows the postscaler a 1:1 to 1:16 scaling (inclusive). After the postscaler overflows, the TMR2 interrupt flag bit (TMR2IF) is set to indicate the Timer2 overflow. This is useful in reducing the software overhead of the Timer2 interrupt service routine, since it will only execute once every postscaler # of matches.

The match output of TMR2 is also routed to the Synchronous Serial Port module, which may software select this as the clock source for the shift clock.

13.6 Clearing the Timer2 Prescaler and Postscaler

The prescaler and postscaler counters are cleared when any of the following occurs:

- a write to the TMR2 register
- a write to the T2CON register

Note: When T2CON is written TMR2 does not clear.

- any device reset (Power-on Reset, \overline{MCLR} reset, Watchdog Timer Reset, Brown-out Reset, or Parity Error Reset)

13.7 Sleep Operation

During sleep, TMR2 will not increment. The prescaler will retain the last prescale count, ready for operation to resume after the device wakes from sleep.

Table 13-1: Registers Associated with Timer2

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR, PER	Value on all other resets
INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
PIR	TMR2IF ⁽¹⁾								0	0
PIE	TMR2IE ⁽¹⁾								0	0
TMR2	Timer2 module's register								0000 0000	0000 0000
T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
PR2	Timer2 Period Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'.

Shaded cells are not used by the Timer2 module.

Note 1: The position of this bit is device dependent.

Section 13. Timer2

13.8 Initialization

Example 13-1 shows how to initialize the Timer2 module, including specifying the Timer2 prescaler and postscaler.

Example 13-1: Timer2 Initialization

```
CLRF  T2CON          ; Stop Timer2, Prescaler = 1:1,
                    ; Postscaler = 1:1
CLRF  TMR2           ; Clear Timer2 register
CLRF  INTCON         ; Disable interrupts
BSF   STATUS, RP0   ; Bank1
CLRF  PIE1           ; Disable peripheral interrupts
BCF   STATUS, RP0   ; Bank0
CLRF  PIR1           ; Clear peripheral interrupts Flags
MOVLW 0x72           ; Postscaler = 1:15, Prescaler = 1:16
MOVWF T2CON          ; Timer2 is off
BSF   T2CON, TMR2ON ; Timer2 starts to increment
;
; The Timer2 interrupt is disabled, do polling on the overflow bit
;
T2_OVFL_WAIT
    BTFSS PIR1, TMR2IF ; Has TMR2 interrupt occurred?
    GOTO  T2_OVFL_WAIT ; NO, continue loop
;
; Timer has overflowed
;
    BCF   PIR1, TMR2IF ; YES, clear flag and continue.
```

PICmicro MID-RANGE MCU FAMILY

13.9 Design Tips

No related Design Tips at this time.

Section 13. Timer2

13.10 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 the Timer2 Module are:

Title	Application Note #
Using the CCP Module	AN594
Air Flow Control using Fuzzy Logic	AN600
Adaptive Differential Pulse Code Modulation using PICmicros	AN643

PICmicro MID-RANGE MCU FAMILY

13.11 Revision History

Revision A

This is the initial released revision of the Tlmer2 module description.