Section 12. Timer1

HIGHLIGHTS

This section of the manual contains the following major topics:

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, $\overline{\text{T1SYNC}}$ (Figure 12-1).

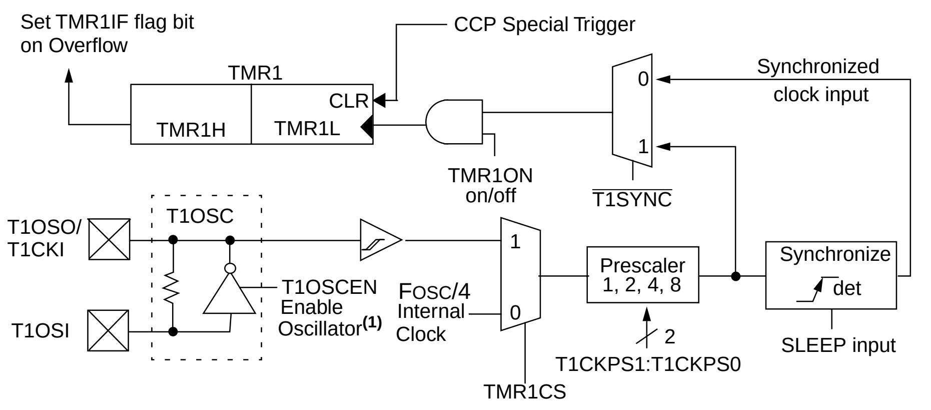
In timer mode, Timer1 increments every instruction cycle. In counter mode, it increments on every rising edge of the external clock input on pin T1CKI.

Timer1 can be turned on and off using theTMR1ON 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



Note 1: When the T1OSCEN bit is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.

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	$\overline{\text{T1SYNC}}$	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 **$\overline{\text{T1SYNC}}$**: 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

12

Timer1

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 FOSC/4. The synchronize control bit, $\overline{\text{T1SYNC}}$ (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 $\overline{\text{T1SYNC}}$ 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 (Tosc) 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 Tosc clocks of the internal phase clocks. Therefore, it is necessary for the T1CKI pin to be high for at least 2Tosc (and a small RC delay) and low for at least 2Tosc (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 4Tosc (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.

12.5 Timer1 Operation in Asynchronous Counter Mode

If $\overline{\text{T1SYNC}}$ (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 $\overline{\text{T1SYNC}}$ 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

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
```

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-2 100.00 KC-P	± 20 PPM	
200 kHz	STD XTL 200.000 kHz	± 20 PPM	

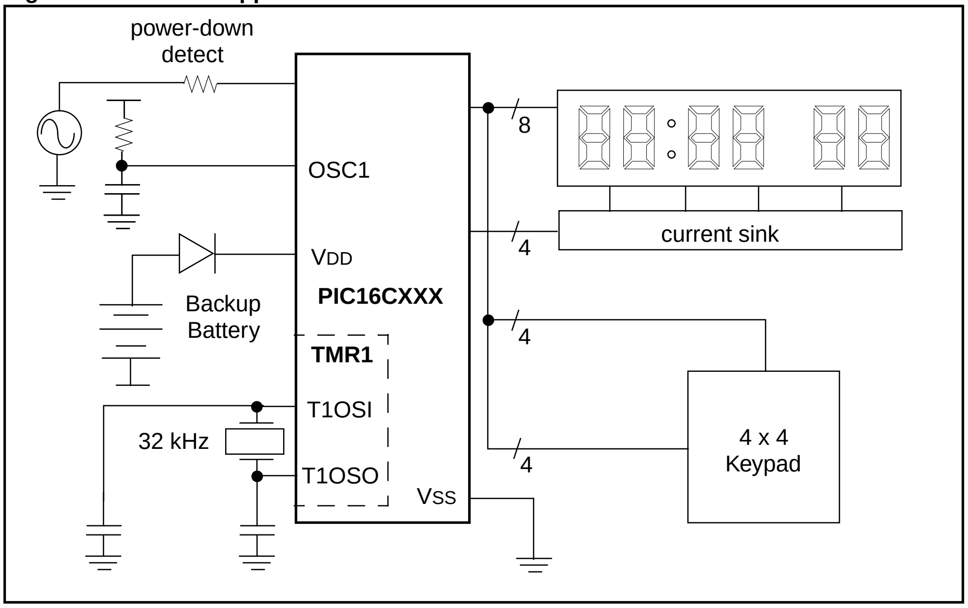
Note 1: Higher capacitance increases the stability of oscillator but also increases the start-up time.

2: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

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



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 over-flows, 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	T0IE	INTE	RBIE(2)	T0IF	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	$\overline{\text{T1SYNC}}$	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.

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
                         ;   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 12-6: 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
```

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 then 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.

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

12.14 Revision History

Revision A

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