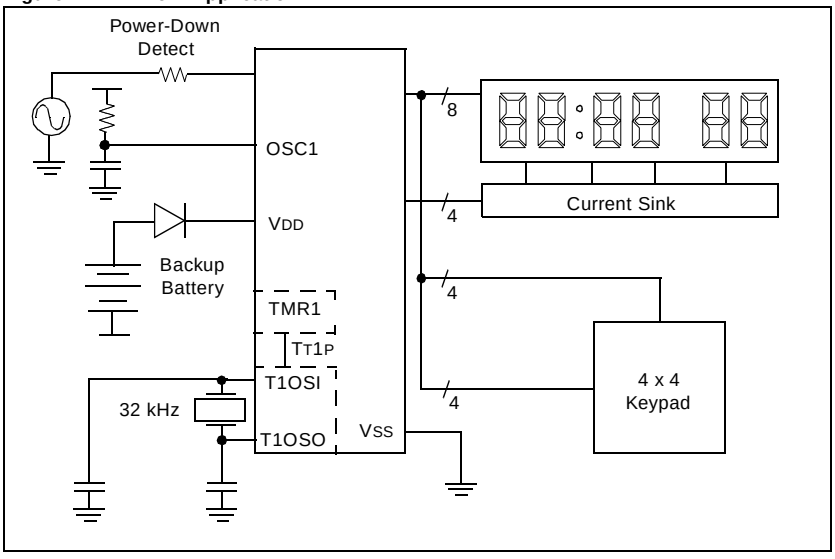
14.8 Typical Application

In Figure 14-4 an example application is given, where Timer1 is driven from an external 32 kHz oscillator. The external 32 kHz oscillator 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 wakes up the device so that the appropriate registers can be updated.

Figure 14-4: Timer1 Application



In this example, a 32 kHz crystal is used as the time base for the Real Time Clock. If the clock needs to be updated at 1 second intervals, then the Timer1 must be loaded with a value to allow the Timer1 to overflow at the desired rate. In the case of a 1 second Timer1 overflow, the TMR1H register should be loaded with a value of 80k after each overflow.

Note:	The TMR1L register should never be modified, since an external clock is asychronous to the system clock. Writes to the TRM1L register may corrupt the real time counter value causing inaccuracies.

14.14 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 that your layout uses good PCB layout techniques so noise does not couple onto the Timer1/Timer3 oscillator lines.

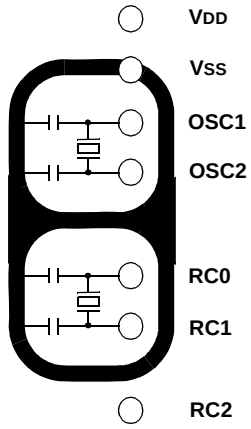
12.3.3 TIMER1 OSCILLATOR LAYOUT CONSIDERATIONS

The Timer1 oscillator circuit draws very little power during operation. Due to the low-power nature of the oscillator, it may also be sensitive to rapidly changing signals in close proximity.

The oscillator circuit, shown in Figure 12-3, should be located as close as possible to the microcontroller. There should be no circuits passing within the oscillator circuit boundaries other than VSS or VDD.

If a high-speed circuit must be located near the oscillator (such as the CCP1 pin in Output Compare or PWM mode, or the primary oscillator using the OSC2 pin), a grounded guard ring around the oscillator circuit, as shown in Figure 12-4, may be helpful when used on a single-sided PCB or in addition to a ground plane.

FIGURE 12-4: OSCILLATOR CIRCUIT WITH GROUNDED GUARD RING



Note: Not drawn to scale.

12.4 Timer1 Interrupt

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 interrupt flag bit, TMR1IF (PIR1<0>). This interrupt can be enabled or disabled by setting or clearing the Timer1 Interrupt Enable bit, TMR1IE (PIE1<0>).

12.5 Resetting Timer1 Using the CCP Special Event Trigger

If either of the CCP modules is configured in Compare mode to generate a Special Event Trigger (CCP1M3:CCP1M0 or CCP2M3:CCP2M0 = `1011`), this signal will reset Timer1. The trigger from CCP2 will also start an A/D conversion if the A/D module is enabled (see **Section 15.3.4 “Special Event Trigger”** for more information).

The module must be configured as either a timer or a synchronous counter to take advantage of this feature. When used this way, the CCPRH:CCPRL register pair effectively becomes a period register for Timer1.

If Timer1 is running in Asynchronous Counter mode, this Reset operation may not work.

In the event that a write to Timer1 coincides with a Special Event Trigger, the write operation will take precedence.

Note:	The Special Event Triggers from the CCP2 module will not set the TMR1IF interrupt flag bit (PIR1<0>).

12.6 Using Timer1 as a Real-Time Clock

Adding an external LP oscillator to Timer1 (such as the one described in **Section 12.3 “Timer1 Oscillator”**) gives users the option to include RTC functionality to their applications. This is accomplished with an inexpensive watch crystal to provide an accurate time base and several lines of application code to calculate the time. When operating in Sleep mode and using a battery or supercapacitor as a power source, it can completely eliminate the need for a separate RTC device and battery backup.

The application code routine, `RTCisr`, shown in Example 12-1, demonstrates a simple method to increment a counter at one-second intervals using an Interrupt Service Routine. Incrementing the TMR1 register pair to overflow triggers the interrupt and calls the routine, which increments the seconds counter by one. Additional counters for minutes and hours are incremented as the previous counter overflows.

Since the register pair is 16 bits wide, counting up to overflow the register directly from a 32.768 kHz clock would take 2 seconds. To force the overflow at the required one-second intervals, it is necessary to preload it. The simplest method is to set the MSb of TMR1H with a `BSF` instruction. Note that the TMR1L register is never preloaded or altered; doing so may introduce cumulative error over many cycles.

For this method to be accurate, Timer1 must operate in Asynchronous mode and the Timer1 overflow interrupt must be enabled (PIE1<0> = `1`) as shown in the routine, `RTCinit`. The Timer1 oscillator must also be enabled and running at all times.

EXAMPLE 12-1: IMPLEMENTING A REAL-TIME CLOCK USING A TIMER1 INTERRUPT SERVICE

```
RTCinit
            MOVLW    80h              ; Preload TMR1 register pair
            MOVWF    TMR1H            ; for 1 second overflow
            CLRF     TMR1L
            MOVLW    b'00001111'      ; Configure for external clock,
            MOVWF    T1OSC            ; Asynchronous operation, external oscillator
            CLRF     secs             ; Initialize timekeeping registers
            CLRF     mins             ;
            MOVLW    d'12'
            MOVWF    hours
            BSF      PIE1, TMR1IE     ; Enable Timer1 interrupt
            RETURN
RTCisr
            BSF      TMR1H, 7         ; Preload for 1 sec overflow
            BCF      PIR1, TMR1IF     ; Clear interrupt flag
            INCF     secs, F          ; Increment seconds
            MOVLW    d'59'            ; 60 seconds elapsed?
            CPFSGT   secs
            RETURN                    ; No, done
            CLRF     secs             ; Clear seconds
            INCF     mins, F          ; Increment minutes
            MOVLW    d'59'            ; 60 minutes elapsed?
            CPFSGT   mins
            RETURN                    ; No, done
            CLRF     mins             ; clear minutes
            INCF     hours, F         ; Increment hours
            MOVLW    d'23'            ; 24 hours elapsed?
            CPFSGT   hours
            RETURN                    ; No, done
            MOVLW    d'01'            ; Reset hours to 1
            MOVWF    hours
            RETURN                    ; Done
```

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	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	51
PIR1	SPPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	54
PIE1	SPPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	54
IPR1	SPPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	54
TMR1L	Timer1 Register Low Byte								52
TMR1H	TImer1 Register High Byte								52
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	52

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.

Note 1: These bits are unimplemented on 28-pin devices; always maintain these bits clear.