Section 11. Timers

HIGHLIGHTS

This section of the manual contains the following topics:

11.1 INTRODUCTION

The PIC24H device family offers several 16-bit timer modules. With certain exceptions, all of the 16-bit timers have the same functional circuitry, and are classified into three types according to their functional differences:

- Type A timer (Timer1)
- Type B timer (Timer2, Timer4, Timer6 and Timer8)
- Type C timer (Timer3, Timer5, Timer7 and Timer9)

The Type B and Type C timers can be combined to form a 32-bit timer.

Each timer module is a 16-bit timer/counter consisting of the following readable/writable registers:

- TMRx: 16-bit Timer Count register
- PRx: 16-bit Timer Period register associated with the timer
- TxCON: 16-bit Timer Control register associated with the timer

Each timer module also has these associated bits for interrupt control:

- Interrupt Enable Control bit (TxIE)
- Interrupt Flag Status bit (TxIF)
- Interrupt Priority Control bits (TxIP<2:0>)

Note 1: Each PIC24H device variant can have one or more timer modules. For more details, refer to the specific device data sheets.

2: An ‘x’ used in the names of pins, control/status bits and registers denotes the particular timer number (x = 1 to 9).

3: A ‘y’ used in the names of pins, control/status bits and registers denotes the particular Type C timer number (y = 3, 5, 7 and 9).

11.2 TIMER VARIANTS

This section describes the different types of timers available on the PIC24H device family.

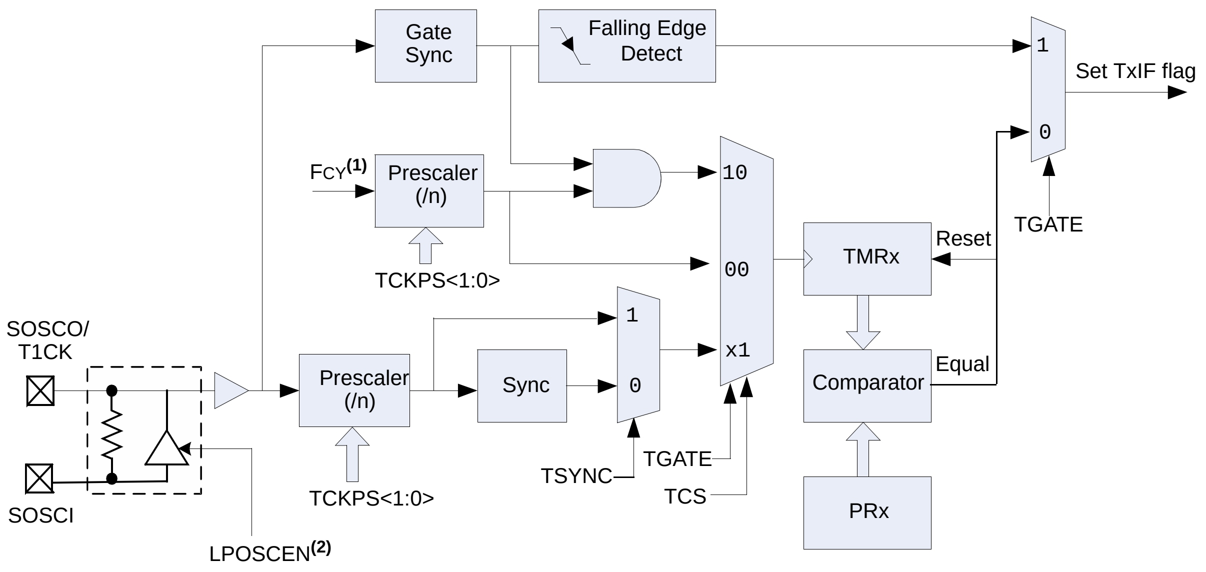
11.2.1 Type A Timer

Timer1 is a Type A timer. A Type A timer has the following unique features over other types of timers:

- Can be operated from the low-power 32 kHz crystal oscillator available on the device
- Can be operated in Asynchronous Counter mode from an external clock source
- Optionally, the external clock input (TxCK) can be synchronized to the internal device clock and clock synchronization is performed after TxCK is divided by the prescaler. The advantage of clock synchronization after division by the prescaler is explained in Section **11.4.3 “Synchronous Counter Mode”**

The unique features of a Type A timer allow it to be used for Real Time Clock (RTC) applications. Figure 11-1 shows a block diagram of a Type A timer.

Figure 11-1: Type A Timer Block Diagram



Note 1: FCY is the instruction cycle clock.

2: For information on enabling the secondary oscillator, refer to **Section 7. “Oscillator”.**

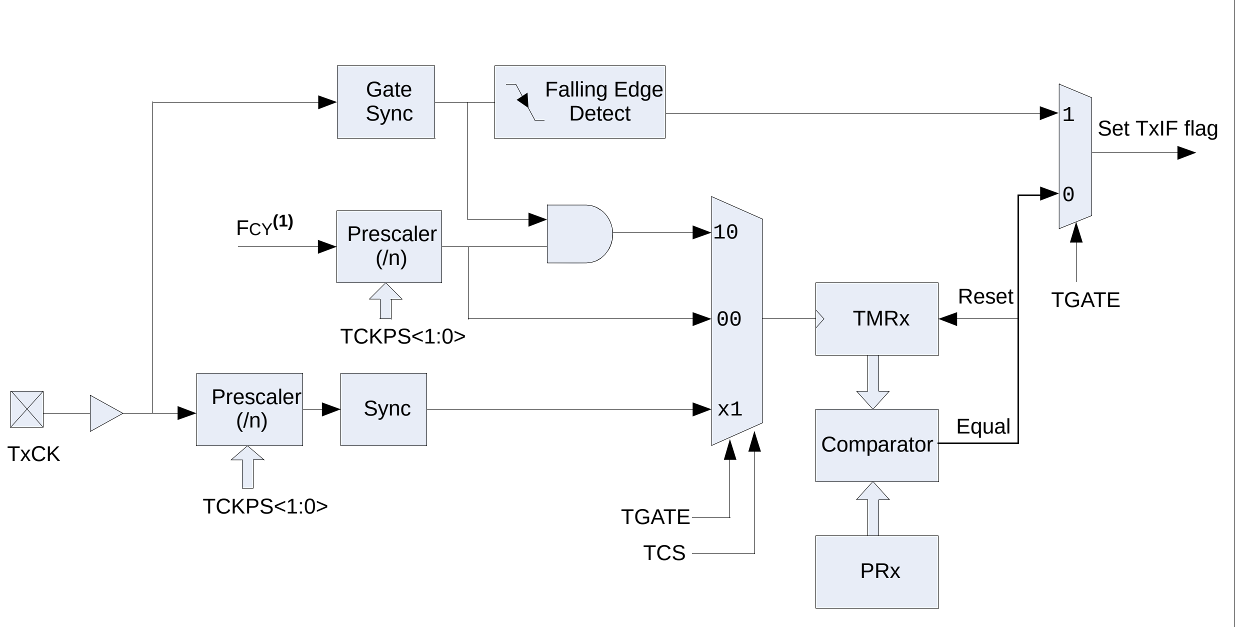
11.2.2 Type B Timer

Timer2, Timer4, Timer6 and Timer8, if present, are Type B timers. A Type B timer has the following specific features:

- It can be concatenated with a Type C timer to form a 32-bit timer
- The external clock input (TxCK) is always synchronized to the internal device clock and clock synchronization is performed after TxCK is divided by the prescaler. The advantage of clock synchronization after division by the prescaler explained in **Section 11.4.3 “Synchronous Counter Mode”**

Figure 11-2 shows a block diagram of the Type B timer.

Figure 11-2: Type B Timer Block Diagram



Note 1: FCY is the instruction cycle clock.

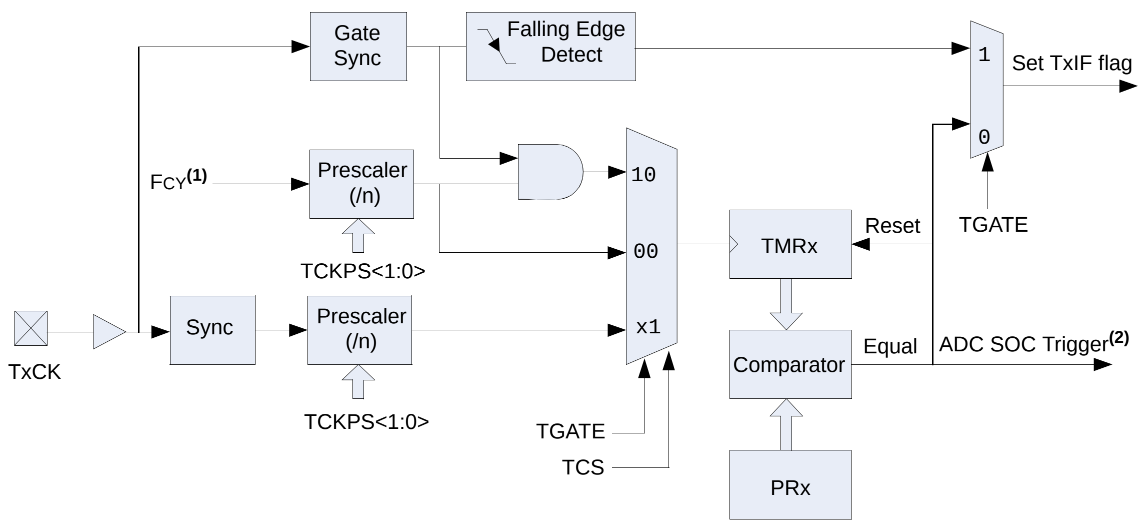
11.2.3 Type C Timer

Timer3, Timer5, Timer7 and Timer9, if present, are Type C timers. A Type C timer has the following specific features:

- It can be concatenated with a Type B timer to form a 32-bit timer
- At least one Type C timer has the ability to trigger an Analog-to-Digital (A/D) conversion
- The external clock input (TxCK) is always synchronized to the internal device clock and the clock synchronization is performed using TxCK, after which this synchronized clock is divided by the prescaler

Figure 11-3 shows a block diagram of the Type C timer.

Figure 11-3: Type C Timer Block Diagram



Note 1: FCY is the instruction cycle clock.

2: ADC trigger is available only on TMR3 and TMR5.

11.3 CONTROL REGISTERS

Register 11-1: TxCON: Type A Timer Control Register (x = 1)

R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15							bit 8

U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0
—	TGATE	TCKPS<1:0>		—	TSYNC	TCS	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15 **TON:** Timer On bit
1 = Starts the timer
0 = Stops the timer

bit 14 **Unimplemented:** Read as ‘0’

bit 13 **TSIDL:** Stop in Idle Mode bit
1 = Discontinue timer operation when device enters Idle mode
0 = Continue timer operation in Idle mode

bit 12-7 **Unimplemented:** Read as ‘0’

bit 6 **TGATE:** Timer Gated Time Accumulation Enable bit
When TCS = 1:
This bit is ignored
When TCS = 0:
1 = Gated time accumulation enabled
0 = Gated time accumulation disabled

bit 5-4 **TCKPS<1:0>:** Timer Input Clock Prescale Select bits
11 = 1:256 prescale value
10 = 1:64 prescale value
01 = 1:8 prescale value
00 = 1:1 prescale value

bit 3 **Unimplemented:** Read as ‘0’

bit 2 **TSYNC:** Timer External Clock Input Synchronization Select bit
When TCS = 1:
1 = Synchronize external clock input
0 = Do not synchronize external clock input
When TCS = 0:
This bit is ignored. Read as ‘0’. Timerx uses the internal clock when TCS = 0

bit 1 **TCS:** Timer Clock Source Select bit
1 = External clock from TxCK pin
0 = Internal clock (FOSC/2)

bit 0 **Unimplemented:** Read as ‘0’

Register 11-2: TxCON: Type B Timer Control Register (x = 2, 4, 6, 8)

R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15							bit 8

U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0
—	TGATE	TCKPS<1:0>		T32	—	TCS	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15 **TON:** Timerx On bit

When T32 = 1 (in 32-bit Timer mode):
1 = Starts 32-bit TMRx:TMRy(1) timer pair
0 = Stops 32-bit TMRx:TMRy(1) timer pair

When T32 = 0 (in 16-bit Timer mode):
1 = Starts 16-bit timer
0 = Stops 16-bit timer

bit 14 **Unimplemented:** Read as ‘0’

bit 13 **TSIDL:** Stop in Idle Mode bit

1 = Discontinue timer operation when device enters Idle mode
0 = Continue timer operation in Idle mode

bit 12-7 **Unimplemented:** Read as ‘0’

bit 6 **TGATE:** Timerx Gated Time Accumulation Enable bit

When TCS = 1:
This bit is ignored

When TCS = 0:
1 = Gated time accumulation enabled
0 = Gated time accumulation disabled

bit 5-4 **TCKPS<1:0>:** Timerx Input Clock Prescale Select bits

11 = 1:256 prescale value
10 = 1:64 prescale value
01 = 1:8 prescale value
00 = 1:1 prescale value

bit 3 **T32:** 32-Bit Timerx Mode Select bit

1 = TMRx and TMRy(1) form a 32-bit timer
0 = TMRx and TMRy(1) form separate 16-bit timer

bit 2 **Unimplemented:** Read as ‘0’

bit 1 **TCS:** Timerx Clock Source Select bit

1 = External clock from TxCK pin
0 = Internal clock (FOSC/2)

bit 0 **Unimplemented:** Read as ‘0’

Note 1: TMRy is a Type C timer (y = 3, 5, 7, and 9)

Register 11-3: TxCON: Type C Timer Control Register (x = 3, 5, 7, 9)

R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON[(2)]	—	TSIDL[(1)]	—	—	—	—	—
bit 15							bit 8

U-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	U-0
—	TGATE[(2)]	TCKPS<1:0>[(2)]		—	—	TCS[(2)]	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15 **TON:** Timerx On bit[(2)]
1 = Starts 16-bit Timerx
0 = Stops 16-bit Timerx

bit 14 **Unimplemented:** Read as ‘0’

bit 13 **TSIDL:** Stop in Idle Mode bit [(1)]
1 = Discontinue timer operation when device enters Idle mode
0 = Continue timer operation in Idle mode

bit 12-7 **Unimplemented:** Read as ‘0’

bit 6 **TGATE:** Timerx Gated Time Accumulation Enable bit[(2)]
When TCS = 1:
This bit is ignored
When TCS = 0:
1 = Gated time accumulation enabled
0 = Gated time accumulation disabled

bit 5-4 **TCKPS<1:0>:** Timerx Input Clock Prescale Select bits[(2)]
11 = 1:256 prescale value
10 = 1:64 prescale value
01 = 1:8 prescale value
00 = 1:1 prescale value

bit 3-2 **Unimplemented:** Read as ‘0’

bit 1 **TCS:** Timerx Clock Source Select bit[(2)]
1 = External clock from TxCK pin
0 = Internal clock (FOSC/2)

bit 0 **Unimplemented:** Read as ‘0’

Note 1: When 32-bit timer operation is enabled (T32 = 1) in Type B Timer Control (TxCON<3>) register, TSIDL bit must be cleared to operate the 32-bit timer in Idle mode.

2: These bits have no effect when the 32-bit timer operation is enabled (T32 = 1) in the Type B Timer Control (TxCON<3>) register.

11.4 MODES OF OPERATION

The timer module can operate in one of the following modes:

- Timer mode
- Gated Timer mode
- Synchronous Counter mode
- Asynchronous Counter mode (Type A timer only)

In Timer and Gated Timer modes, the input clock is derived from the internal instruction cycle clock (FCY). In Synchronous and Asynchronous Counter modes, the input clock is derived from the external clock input at the TxCK pin.

The timer modes are determined by the following bits:

- TCS (TxCON<1>): Timer Clock Source Control bit
- TSYNC (TxCON<2>): Timer Synchronization Control bit (Type A timer only)
- TGATE (TxCON<6>): Timer Gate Control bit

Timer control bit settings for different operating modes are provided in Table 11-1, as follows:

Table 11-1: Timer Modes Configuration

Mode	Bit Setting		
	TCS	TGATE(2)	TSYNC(1)
Timer	0	0	x
Gated timer	0	1	x
Synchronous counter	1	x	1
Asynchronous counter (3)	1	x	0

Note 1: TSYNC bit is available for Type A timer only and is ignored for both timer modes.
2: TGATE bit is ignored for both the counter modes.
3: Asynchronous Counter mode is supported by Type A timer only.

The input clock (FCY or TxCK) to all 16-bit timers has prescale options of 1:1, 1:8, 1:64 and 1:256. The clock prescaler is selected using the Timer Clock Prescaler (TCKPS<1:0>) bits in the Timer Control (TxCON<5:4>) register. The prescaler counter is cleared when any of the following occurs:

- A write to the Timer register (TMRx) or Timer Control (TxCON) register
- Clearing the Timer Enable (TON) bit in the Timer Control (TxCON<15>) register
- Any device Reset

The timer module is enabled or disabled using the TON bit (TxCON <15>).

11.4.1 Timer Mode

In Timer mode, the input clock to the timer is derived from the internal clock (FCY), divided by a programmable prescaler. When the timer is enabled, it increments by one on every rising edge of the input clock and generates an interrupt on a period match. Figure 11-4 illustrates the timer operation.

To configure Timer mode:

- Clear the TCS control bit (TxCON<11>) to select the internal clock source
- Clear the TGATE control bit (TxCON<6>) to disable Gated Timer mode operation

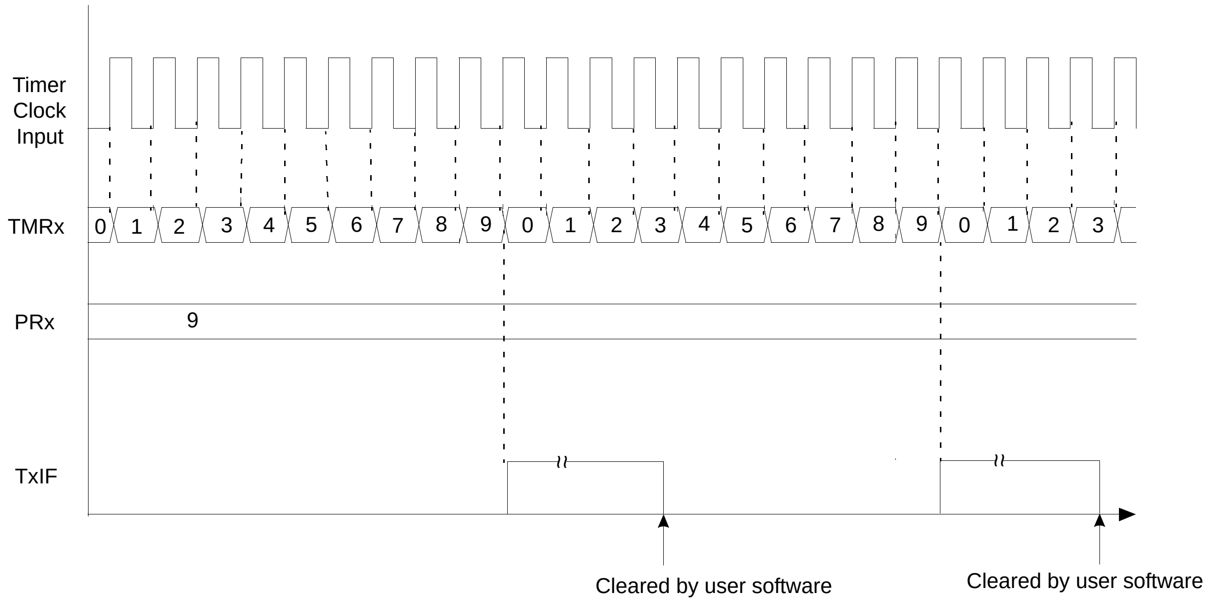
Setting the TSYNC bit (TxCON<2>) has no effect since the internal clock is always synchronized.

Example 11-1 illustrates the code sequence to set up Timer1 in 16-bit Timer mode. This code generates an interrupt on every 10 instruction cycles.

Example 11-1: Initialization Code for 16-Bit Timer Mode

```
   T1CONbits.TON = 0;        // Disable Timer
   T1CONbits.TCS = 0;        // Select internal instruction cycle clock
   T1CONbits.TGATE = 0;      // Disable Gated Timer mode
   T1CONbits.TCKPS = 0b00;   // Select 1:1 Prescaler
   TMR1 = 0x00;              // Clear timer register
   PR1 = 9;                  // Load the period value

   IPC0bits.T1IP = 0x01;     // Set Timer 1 Interrupt Priority Level
   IFS0bits.T1IF = 0;        // Clear Timer 1 Interrupt Flag
   IEC0bits.T1IE = 1;        // Enable Timer1 interrupt

   T1CONbits.TON = 1;        // Start Timer

/* Example code for Timer1 ISR*/
void __attribute__((__interrupt__, __shadow__)) _T1Interrupt(void)
 }
/* Interrupt Service Routine code goes here */
IFS0bits.T1IF = 0;            //Clear Timer1 interrupt flag
 }
```

Figure 11-4: Interrupt Timing for Timer Period Match



11.4.2 Gated Timer Mode

When the timer module operates with the internal clock (TCS = 0), Gated Timer mode can be used to measure the duration of an external gate signal. In this mode the timer increments by one on every rising edge of the input clock as long as the external gate signal at the TxCK pin is high. The timer interrupt is generated on the falling edge of the TxCK pin. Figure 11-5 illustrates Gated Timer mode operation.

To configure the Gated Timer mode:

- Set the TGATE control bit (TxCON<6>) to enable gated timer operation
- Clear the TCS control bit (TxCON<11>) to select the internal clock source

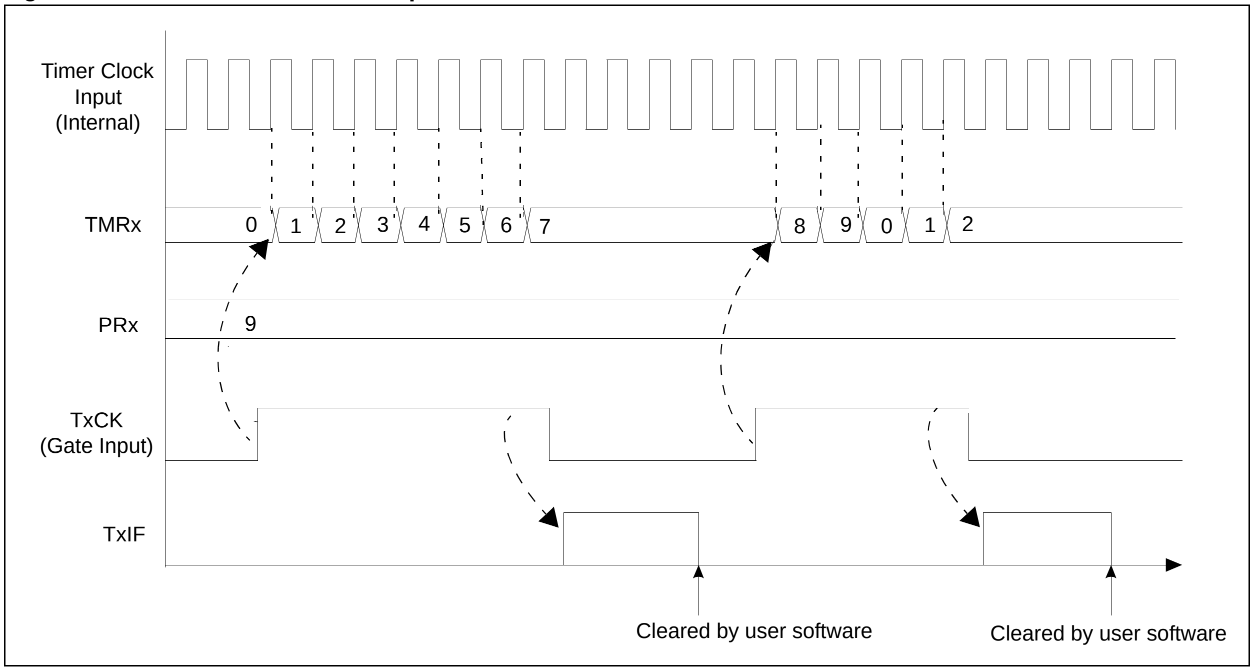
Setting the TSYNC bit (TxCON<2>) has no effect since the internal clock is always synchronized.

Example 11-2 illustrates the code sequence to measure pulse width (T1CK) in Gated Timer mode.

Example 11-2: Initialization Code for 16-Bit Gated Timer Mode

```
    T1CONbits.TON = 0;       // Disable Timer
    T1CONbits.TCS = 0;       // Select internal instruction cycle clock
    T1CONbits.TGATE = 1;     // Enable Gated Timer mode
    T1CONbits.TCKPS = 0b00;  // Select 1:1 Prescaler
    TMR1 = 0x00;             // Clear timer register
    PR1 = 9;                 // Load the period value

    IPC0bits.T1IP = 0x01;    // Set Timer 1 Interrupt Priority Level
    IFS0bits.T1IF = 0;       // Clear Timer 1 Interrupt Flag
    IEC0bits.T1IE = 1;       // Enable Timer1 interrupt

    T1CONbits.TON = 1;       // Start Timer

/* Example code for Timer1 ISR*/
void __attribute__((__interrupt__, __shadow__)) _T1Interrupt(void)
{/* Interrupt Service Routine code goes here */

    IFS0bits.T1IF = 0;       //Clear Timer1 interrupt flag
 }
```

Figure 11-5: Gated Timer Mode Operation



11.4.3 Synchronous Counter Mode

In Synchronous Counter mode, the input clock to the timer is derived from the external clock input divided by a programmable prescaler. In this mode, the external clock input is synchronized with the internal device clock. When the timer is enabled, it increments by one on every rising edge of the input clock, and generates an interrupt on a period match.

To configure Synchronous Counter mode:

- Set the TSYNC control bit (TxCON<2>) for a Type A timer to enable clock synchronization. For a Type B or Type C timer, the external clock input is always synchronized
- Set the TCS control bit (TxCON<11>) to select the external clock source

A timer operating from a synchronized external clock source does not operate in Sleep mode, since the synchronization circuit is shut off during Sleep mode.

For Type C timers, it is necessary for the external clock input period to be high for at least 0.5 TCY (and an additional input buffer delay of 20 ns), and low for at least 0.5 TCY (and an additional input buffer delay of 20 ns) for proper synchronization.

The clock synchronization for a Type A and Type B timer is performed after the prescaler and the prescaler output changes on the rising edge of the input. Therefore, for a Type A and Type B timer, the external clock input period must be at least 0.5 TCY (and an additional input buffer delay of 20 ns) divided by the prescaler value.

However, the high and low time of the external clock input should not violate the minimum pulse width requirement of 10 ns nominal (or 50 MHz nominal frequency).

Note 1: For the external clock timing requirement in Synchronous Counter mode, refer to the electrical specification of the specific device data sheet.

Example 11-3 illustrates the code sequence to set up the Timer1 module in Synchronous Counter mode. This code generates an interrupt after counting 1000 rising edges in the TxCK pin.

Example 11-3: Initialization Code for 16-Bit Synchronous Counter Mode

```
   T1CONbits.TON = 0;         // Disable Timer
   T1CONbits.TCS = 1;         // Select external clock source
   T1CONbits.TSYNC = 1;       // Enable Synchronization
   T1CONbits.TCKPS = 0b00;    // Select 1:1 Prescaler
   TMR1 = 0x00;               // Clear timer register
   PR1 = 999;                 // Load the period value

   IPC0bits.T1IP = 0x01;      // Set Timer 1 Interrupt Priority Level
   IFS0bits.T1IF = 0;         // Clear Timer 1 Interrupt Flag
   IEC0bits.T1IE = 1;         // Enable Timer1 interrupt

   T1CONbits.TON = 1;         // Start Timer

/* Example code for Timer1 ISR*/
void __attribute__((__interrupt__, __shadow__)) _T1Interrupt(void)
{
/* Interrupt Service Routine code goes here */

   IFS0bits.T1IF = 0;           //Clear Timer1 interrupt flag
}
```

11.4.4 Asynchronous Counter Mode (Type A Timer only)

A Type A timer has the ability to operate in an Asynchronous Counting mode. In Asynchronous Counter mode, the input clock to the timer is derived from the external clock input (TxCK) divided by a programmable prescaler. In this mode, the external clock input is not synchronized with the internal device clock. When enabled, the timer increments by one on every rising edge of the input clock and generates an interrupt on a period match.

To configure the Asynchronous Counter mode:

- Clear the TSYNC control bit (TxCON<2>) to disable clock synchronization
- Set the TCS control bit (TxCON<11>) to select the external clock source

In Asynchronous Counter mode:

- The timer can be clocked from the low-power 32 kHz secondary crystal oscillator for Real Time Clock (RTC) applications by setting the Secondary Oscillator Enable (LPOSCEN) bit in the Oscillator Control (OSCCON<1>) register. For further details, refer to **Section 7. “Oscillator”**
- The timer can operate during Sleep mode if the external clock input is active or the secondary oscillator is enabled. It can generate an interrupt (if enabled) on a period register match to wake-up the processor from Sleep mode

In Asynchronous Counter mode, the external clock input high and low time should not violate the minimum pulse width requirement of 10 ns nominal (or 50 MHz nominal frequency).

Note: For the external clock timing requirement in Asynchronous Counter mode, refer to the electrical specification of the specific device data sheet.

Example 11-4 illustrates the code sequence to set up the Timer1 module in Asynchronous Counter mode. This code generates an interrupt every second when running on 32 kHz clock input.

Example 11-4: Initialization Code for 16-Bit Asynchronous Counter Mode

```
   T1CONbits.TON = 0;        // Disable Timer
   T1CONbits.TCS = 1;        // Select external clock
   T1CONbits.TSYNC = 0;      // Disable Synchronization
   T1CONbits.TCKPS = 0b00;   // Select 1:1 Prescaler
   TMR1 = 0x00;              // Clear timer register
   PR1 = 32767;              // Load the period value

   IPC0bits.T1IP = 0x01;     // Set Timer 1 Interrupt Priority Level
   IFS0bits.T1IF = 0;        // Clear Timer 1 Interrupt Flag
   IEC0bits.T1IE = 1;        // Enable Timer1 interrupt

   T1CONbits.TON = 1;        // Start Timer

 /* Example code for Timer1 ISR*/
void __attribute__((__interrupt__, __shadow__)) _T1Interrupt(void)
{
/* Interrupt Service Routine code goes here */

   IFS0bits.T1IF = 0;        //Clear Timer1 interrupt flag
 }
```

11.5 TIMER INTERRUPTS

A timer interrupt is generated:

- On a period match for Timer mode or Synchronous/Asynchronous Counter modes (refer to Figure 11-4)
- On the falling edge of the “gate” signal at the TxCK pin for Gated Timer mode (refer to Figure 11-5)

The Timer Interrupt Flag (TxIF) bit must be cleared in software.

A timer is enabled as a source of interrupt via the respective Timer Interrupt Enable (TxIE) bit. The interrupt priority level (TxIP<2:0>) bits must be written with a non-zero value for the timer to be a source of interrupt. For further details, refer to **Section 6. “Interrupts”**.

Note:	A special case occurs when the period register, PRx, is loaded with 0x0000 and the timer is enabled. No timer interrupts are generated for this configuration.

11.6 32-BIT TIMER CONFIGURATION

A 32-bit timer module can be formed by combining Type B and Type C 16-bit timers. For 32-bit timer operation, the T32 control bit in the Type B Timer Control (TxCON<3>) register must be set. The Type C timer holds the most significant word (msw) and the Type B timer holds the least significant word (lsw) for 32-bit operation.

When configured for 32-bit operation, only the Type B Timer Control (TxCON) register bits are required for setup and control. With the exception of the TSIDL bit, all Type C timer control register bits are ignored. For an explanation, refer to **Section 11.8.2 "Timer Operation in Idle Mode"**.

For interrupt control, the combined 32-bit timer uses the interrupt enable, interrupt flag, and interrupt priority control bits of the Type C timer. The interrupt control and status bits for the Type B timer are ignored during 32-bit timer operation.

Table 11-2 lists the Type B and Type C timers that can be combined to form a 32-bit timer.

Table 11-2: 32-bit Timer Combinations

TYPE B timer (lsw)	TYPE C timer (msw)
Timer2	Timer3
Timer4	Timer5
Timer6	Timer7
Timer8	Timer9

A block diagram representation of the 32-bit timer module is shown in Figure 11-6. The 32-timer module can operate in any of the following modes:

- Timer
- Gated Timer
- Synchronous Counter

In Timer and Gated Timer modes, the input clock is derived from the internal instruction cycle clock (FCY). In Synchronous Counter mode, the input clock is derived from the Type B timer external clock input at the TxCK pin.

The 32-bit timer modes are determined by the following bits in the Type B timer control registers:

- TCS (TxCON<1>): Timer Clock Source Control bit
- TGATE (TxCON<6>): Timer Gate Control bit

Timer control bit settings for different operating modes are provided in the Table 11-3.

Table 11-3: Timer Mode Configuration

Mode	Bit Setting	
	TCS	TGATE
Timer	0	0
Gated Timer	0	1
Synchronous Counter	1	x

Note:	Type B and Type C timers do not support the Asynchronous External Clock mode; therefore, 32-bit Asynchronous Counter mode is not supported.

The input clock (FCY or TxCK) to all 32-bit timers has prescale options of 1:1, 1:8, 1:64 and 1:256. The clock prescaler is selected using the Timer Clock Prescaler (TCKPS<1:0>) bits in the Type B Timer Control (TxCON<5:4>) register. The prescaler counter is cleared when any of the following occurs:

- A write to the Type B Timer register (TMRx) or Type B Timer Control (TxCON) register
- Clearing the Timer Enable (TON) bit in Type B Timer Control (TxCON<15>) register
- Any device Reset

The 32-bit timer module is enabled or disabled using the TON bit (TxCON <15>) in the Type B timer control registers.

For 32-bit read/write operations to be synchronized between the lsw and msw of the 32-bit timer, additional control logic and holding registers are used (refer to Figure 11-6). Each Type C timer has a register called TMRyHLD that is used when reading or writing the timer register pair. The TMRyHLD registers are used only when their respective timers are configured for 32-bit operation.

Assuming TMR3:TMR2 form a 32-bit timer pair, the user application should first read the lsw of the timer value from the TMR2 register. The read of the lsw automatically transfers the contents of TMR3 into the TMR3HLD register. The user application can then read TMR3HLD to get the msw of the timer value.

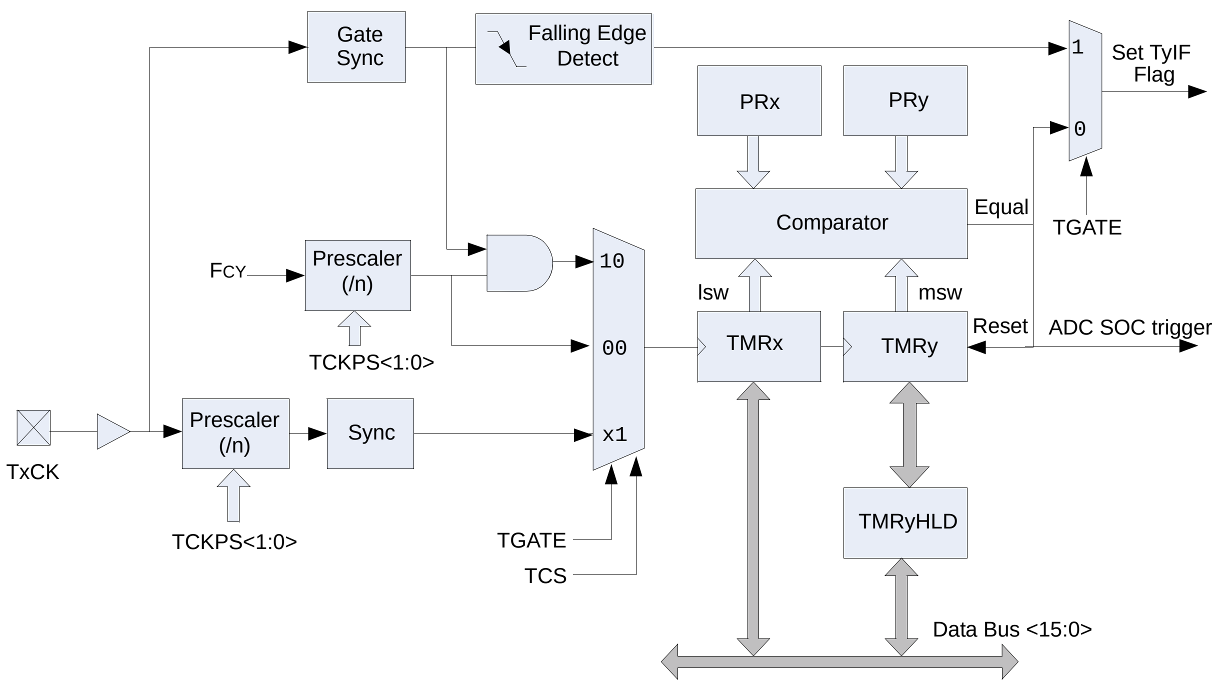
To write a value to the TMR3:TMR2 register pair, the user application should first write the msw to the TMR3HLD register. When the ISW of the timer value is written to TMR2, the contents of TMR3HLD is automatically transferred to the TMR3 register.

The code for accessing the 32-bit timer is shown in Example 11-5, as follows:

Example 11-5: 32-Bit Timer Access

```
// Reading from 32-bit timer
    lsw = TMR2;        //Read lsw from the Type B timer register
    msw = TMR3HLD;     //Read msw from the Type C timer holding register

// Writing to 32-bit timer
    TMR3HLD = msw;     //Write msw to the Type C timer holding register
    TMR2 = lsw;        //Write lsw to the Type B timer register
```

Figure 11-6: Type B/Type C Timer Pair Block Diagram (32-Bit Timer)



Note 1: ADC trigger is available only on TMR3:TMR2 and TMR5:TMR2 32-bit timers.

2: Timer x is a Type B Timer (x = 2, 4, 6, and 8).

3: Timer y is a Type C Timer (y = 3, 5, 7, and 9).

11.7 32-BIT TIMER MODES OF OPERATION

11.7.1 Timer Mode

The 32-bit timer operates similarly to a 16-bit timer in Timer mode. Example 11-6 illustrates the code sequence to set up Timer2 and Timer3 in 32-bit Timer mode.

Example 11-6: Initialization Code for 32-Bit Timer

```
    T3CONbits.TON = 0;        // Stop any 16-bit Timer3 operation
    T2CONbits.TON = 0;        // Stop any 16/32-bit Timer3 operation
    T2CONbits.T32 = 1;        // Enable 32-bit Timer mode
    T2CONbits.TCS = 0;        // Select internal instruction cycle clock
    T2CONbits.TGATE = 0;      // Disable Gated Timer mode
    T2CONbits.TCKPS = 0b00    // Select 1:1 Prescaler
    TMR3 = 0x00;              // Clear 32-bit Timer (msw)
    TMR2 = 0x00;              // Clear 32-bit Timer (lsw)
    PR3 = 0x0002;             // Load 32-bit period value (msw)
    PR2 = 0x0000;             // Load 32-bit period value (lsw)

    IPC2bits.T3IP = 0x01;     // Set Timer3 Interrupt Priority Level
    IFS2bits.T3IF = 0;        // Clear Timer3 Interrupt Flag
    IEC0bits.T3IE = 1;        // Enable Timer3 interrupt

    T2CONbits.TON = 1;        // Start 32-bit Timer

  /* Example code for Timer3 ISR*/
  void __attribute__((__interrupt__, __shadow__)) _T3Interrupt(void)
  {
  /* Interrupt Service Routine code goes here */

    IFS0bits.T3IF = 0;        //Clear Timer3 interrupt flag
  }
```

11.7.2 Gated Timer Mode

The 32-bit timer operates similarly to a 16-bit timer in Gated Timer mode. Example 11-7 illustrates the code sequence to set up Timer2 and Timer3 in 32-bit Gated Timer mode, as follows:

Example 11-7: Initialization Code for 32-Bit Gated Timer Mode

```
    T3CONbits.TON = 0;     // Stop any 16-bit Timer3 operation
    T2CONbits.TON = 0;     // Stop any 16/32-bit Timer3 operation
    T2CONbits.T32 = 1;     // Enable 32-bit Timer mode
    T2CONbits.TCS = 0;     // Select internal instruction cycle clock
    T2CONbits.TGATE = 1;   // Enable Gated Timer mode
    T2CONbits.TCKPS = 0b00 // Select 1:1 Prescaler
    TMR3 = 0x00;           // Clear 32-bit Timer (msw)
    TMR2 = 0x00;           // Clear 32-bit Timer (lsw)
    PR3 = 0x0002;          // Load 32-bit period value (msw)
    PR2 = 0x0000;          // Load 32-bit period value (lsw)

    IPC2bits.T3IP = 0x01;  // Set Timer3 Interrupt Priority Level
    IFS2bits.T3IF = 0;     // Clear Timer3 Interrupt Flag
    IEC0bits.T3IE = 1;     // Enable Timer3 interrupt

    T2CONbits.TON = 1;     // Start 32-bit Timer

  /* Example code for Timer3 ISR*/
  void __attribute__((__interrupt__, __shadow__)) _T3Interrupt(void)
  {
  /* Interrupt Service Routine code goes here */

    IFS0bits.T3IF = 0;     //Clear Timer3 interrupt flag
  }
```

11.7.3 Synchronous Counter Mode

The 32-bit timer operates similarly to a 16-bit timer in Synchronous Counter mode. Example 11-8 illustrates the code sequence to set up Timer2 and Timer3 in 32-bit Synchronous Counter mode.

Example 11-8: Initialization Code for 32-Bit Synchronous Counter Mode

```
    T3CONbits.TON = 0;     // Stop any 16-bit Timer3 operation
    T2CONbits.TON = 0;     // Stop any 16/32-bit Timer3 operation
    T2CONbits.T32 = 1;     // Enable 32-bit Timer mode
    T2CONbits.TCS = 1;     // Select External clock
    T2CONbits.TCKPS = 0b00 // Select 1:1 Prescaler
    TMR3 = 0x00;           // Clear 32-bit Timer (msw)
    TMR2 = 0x00;           // Clear 32-bit Timer (lsw)
    PR3 = 0x0002;          // Load 32-bit period value (msw)
    PR2 = 0x0000;          // Load 32-bit period value (lsw)

    IPC2bits.T3IP = 0x01;  // Set Timer3 Interrupt Priority Level
    IFS2bits.T3IF = 0;     // Clear Timer3 Interrupt Flag
    IEC0bits.T3IE = 1;     // Enable Timer3 interrupt

    T2CONbits.TON = 1;     // Start 32-bit Timer

  /* Example code for Timer3 ISR*/
  void __attribute__((__interrupt__, __shadow__)) _T3Interrupt(void)
  {
  /* Interrupt Service Routine code goes here */

    IFS0bits.T3IF = 0;     //Clear Timer3 interrupt flag
  }
```

11.8 TIMER OPERATION IN POWER-SAVING STATES

11.8.1 Timer Operation in Sleep Mode

When the device enters Sleep mode, the system clock is disabled. If the timer module is running from the internal clock source (FCY), it is disabled as well.

A Type A timer is different from the other timers because it can operate asynchronously from the system clock source. Because of this distinction, the Type A timer can continue to operate during Sleep mode. To operate in Sleep mode, the Type A timer must be configured as follows:

- Clear the TSYNC control bit (TxCON<2>) to disable clock synchronization
- Set the TCS control bit (TxCON<11>) to select external clock source
- Enable the secondary oscillator if the external clock input (TxCK) is not active

Note: The secondary oscillator is enabled by setting the Secondary Oscillator Enable (LPOSCEN) bit in the Oscillator Control (OSCCON<1>) register. For further details,, refer to **Section 7. “Oscillator”**. The 32 kHz watch crystal must be connected to the SOSCO/SOSCI device pins.

When all of these conditions are met, the timer continues to count and detect period matches when the device is in Sleep mode. When a match between the timer and the period register occurs, the TxIF bit is set. The timer interrupt is generated if the timer interrupt is enabled (TxIE = 1).

The timer interrupt wakes up the device from Sleep, and the following occurs:

- If the assigned priority for the interrupt is less than, or equal to, the current CPU priority, the device wakes up and continues code execution from the instruction following the `PWRSAV` instruction that initiated Sleep mode
- If the assigned priority level for the interrupt source is greater than the current CPU priority, the device wakes up and the CPU exception process begins. Code execution continues from the first instruction of the timer Interrupt Service Routine (ISR)

For further details, refer to **Section 9. “Watchdog Timer and Power Saving Modes”**.

11.8.2 Timer Operation in Idle Mode

When the device enters Idle mode, the system clock sources remain functional and the CPU stops executing code. The Timer Stop-in Idle (TSIDL) bit (TxCON<13>) in the Timer Control register determines whether the module stops in Idle mode or continues to operate in Idle mode.

If TSIDL = `0`, the timer continues to operate in Idle mode providing full functionality. For 32-bit timer operation, the TSIDL bit (TxCON<13>) must be cleared in Type B and Type C Timer Control registers for a timer to operate in Idle mode.

If TSIDL = `1`, the timer performs the same functions when stopped in Idle mode as in Sleep mode (refer to **Section 11.8.1 “Timer Operation in Sleep Mode”**).

11.9 PERIPHERALS USING TIMER MODULES

11.9.1 Time Base for Input Capture and Output Compare

The input capture and output compare peripherals can select Timer2 or Timer3 as their time base. For further details, refer to **Section 12. “Input Capture”, Section 13. “Output Compare”**, and the specific device data sheet.

11.9.2 A/D Special Event Trigger

On each device variant, one Type C timer has the capability to generate a special A/D conversion trigger signal on a period match, in both 16- and 32-bit modes. The timer module provides a conversion start signal to the A/D sampling logic.

- If T32 = `0`, when a match occurs between the 16-bit timer register (TMRx) and the respective 16-bit period register (PRx), the A/D Special Event Trigger signal is generated
- If T32 = `1`, when a match occurs between the 32-bit timer (TMRx:TMRy) and the 32-bit respective combined period register (PRx:PRy), the A/D Special Event Trigger signal is generated

The Special Event Trigger signal is always generated by the timer. The trigger source must be selected in the A/D converter control registers. For additional information, refer to **Section 16. “10/12-Bit ADC with DMA”, Section 28. “10/12-Bit ADC without DMA”** and the specific device data sheet.

11.9.3 Timer as an External Interrupt Pin

The external clock input pin for each timer can be used as an additional interrupt pin. To provide the interrupt, the timer period register, PRx, is written with a non-zero value and the TMRx register is initialized to a value of one less than the value written to the period register. The timer must be configured for a 1:1 clock prescaler. An interrupt is generated when the next rising edge of the external clock signal is detected.

11.9.4 I/O Pin Control

When a timer module is enabled and configured for external clock or gate operation, the user application must ensure the I/O pin direction is configured for an input. Enabling the timer module does not configure the pin direction.

11.10 REGISTER MAPS

Summaries of the Special Function Registers associated with the PIC24H timer module are provided in Table 11-4 and Table 11-5

Table 11-4: Timer Register Map

SFR Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
TMR1	Timer1 Register																xxxx
PR1	Period Register 1																FFFF
T1CON	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS<1:0>		—	TSYNC	TCS	—	0000
TMR2	Timer2 Register																xxxx
TMR3HLD	Timer3 Holding Register (for 32-bit timer operations only)																xxxx
TMR3	Timer3 Register																xxxx
PR2	Period Register 2																FFFF
PR3	Period Register 3																FFFF
T2CON	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS<1:0>		T32	—	TCS	—	0000
T3CON	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS<1:0>		—	—	TCS	—	0000
TMR4	Timer4 Register																xxxx
TMR5HLD	Timer5 Holding Register (for 32-bit operations only)																xxxx
TMR5	Timer5 Register																xxxx
PR4	Period Register 4																FFFF
PR5	Period Register 5																FFFF
T4CON	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS<1:0>		T32	—	TCS	—	0000
T5CON	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS<1:0>		—	—	TCS	—	0000
TMR6	Timer6 Register																xxxx
TMR7HLD	Timer7 Holding Register (for 32-bit operations only)																xxxx
TMR7	Timer7 Register																xxxx
PR6	Period Register 6																FFFF
PR7	Period Register 7																FFFF
T6CON	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS<1:0>		T32	—	TCS	—	0000
T7CON	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS<1:0>		—	—	TCS	—	0000
TMR8	Timer8 Register																xxxx
TMR9HLD	Timer9 Holding Register (for 32-bit operations only)																xxxx
TMR9	Timer9 Register																xxxx
PR8	Period Register 8																FFFF
PR9	Period Register 9																FFFF
T8CON	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS<1:0>		T32	—	TCS	—	0000
T9CON	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS<1:0>		—	—	TCS	—	0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

Table 11-5: Interrupt Control Register Map

SFR Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
IFS0	—	—	—	—	—	—	—	T3IF	T2IF	—	—	—	T1IF	—	—	—	0000
IFS1	—	—	—	T5IF	T4IF	—	—	—	—	—	—	—	—	—	—	—	0000
IFS2	T6IF	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
IFS3	—	—	—	—	—	—	—	—	—	—	—	T9IF	T8IF	—	—	T7IF	0000
IEC0	—	—	—	—	—	—	—	T3IE	T2IE	—	—	—	T1IE	—	—	—	0000
IEC1	—	—	—	T5IE	T4IE	—	—	—	—	—	—	—	—	—	—	—	0000
IEC2	T6IE	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
IEC3	—	—	—	—	—	—	—	—	—	—	—	T9IE	T8IE	—	—	T7IE	0000
IPC0	—	T1IP<2:0>			—	—	—	—	—	—	—	—	—	—	—	—	4444
IPC1	—	T2IP<2:0>			—	—	—	—	—	—	—	—	—	—	—	—	4444
IPC2	—	—	—	—	—	—	—	—	—	—	—	—	—	T3IP<2:0>			4444
IPC6	—	T4IP<2:0>			—	—	—	—	—	—	—	—	—	—	—	—	4444
IPC7	—	—	—	—	—	—	—	—	—	—	—	—	—	T5IP<2:0>			4444
IPC11	—	T6IP<2:0>			—	—	—	—	—	—	—	—	—	—	—	—	4444
IPC12	—	T8IP<2:0>			—	—	—	—	—	—	—	—	—	T7IP<2:0>			4444
IPC13	—	—	—	—	—	—	—	—	—	—	—	—	—	T9IP<2:0>			4444

Legend: x = unknown value on Reset, — = unimplemented, read as ‘0’. Reset values are shown in hexadecimal.

11.11 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 PIC24H device family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the Timer modules are:

Title	**Application Note #**
Using Timer1 in Asynchronous Clock Mode	AN580

Note:	For additional application notes and code examples for the PIC24H device family, visit the Microchip web site (www.microchip.com).

11.12 REVISION HISTORY

Revision A (May 2007)

This is the initial released version of this document.