direction register (e.g., TRISC), and data latch register (e.g., LATC). The general operation of these ports is similar to that of PORTA.2.1.

In the PIC18F452 microcontroller PORTC is multiplexed with several peripheral functions as shown in Table 2.8. On a power-on reset, PORTC pins are configured as digital inputs.

In the PIC18F452 microcontroller, PORTD has Schmitt trigger input buffers. On a power-on reset, PORTD is configured as digital input. PORTD can be configured as an 8-bit parallel slave port (i.e., a microprocessor port) by setting bit 4 of the TRISE register. Table 2.9 shows functions of PORTD pins.

In the PIC18F452 microcontroller, PORTE is only 3 bits wide. As shown in Table 2.10, port pins are shared with analog inputs and with parallel slave port read/write control bits. On a power-on reset, PORTE pins are configured as analog inputs and register ADCON1 must be programmed to change these pins to digital I/O.

2.1.9 Timers

The PIC18F452 microcontroller has four programmable timers which can be used in many tasks, such as generating timing signals, causing interrupts to be generated at specific time intervals, measuring frequency and time intervals, and so on.

This section introduces the timers available in the PIC18F452 microcontroller.

Timer 0

Timer 0 is similar to the PIC16 series Timer 0, except that it can operate either in 8-bit or in 16-bit mode. Timer 0 has the following basic features:

- 8-bit or 16-bit operation
- 8-bit programmable prescaler
- External or internal clock source
- Interupt generation on overflow

Timer 0 control register is T0CON, shown in Figure 2.24. The lower 6 bits of this register have similar functions to the PIC16-series OPTION register. The top two bits are used to select the 8-bit or 16-bit mode of operation and to enable/disable the timer.

Table 2.8: PIC18F452 PORTC pin functions

Pin	Description
RC0/T1OSO/T1CKI	
RC0	Digital I/O
T1OSO	Timer 1 oscillator output
T1CKI	Timer 1/Timer 3 external clock input
RC1/T1OSI/CCP2	
RC1	Digital I/O
T1OSI	Timer 1 oscillator input
CCP2	Capture 2 input, Compare 2 and PWM2 output
RC2/CCP1	
RC2	Digital I/O
CCP1	Capture 1 input, Compare 1 and PWM1 output
RC3/SCK/SCL	
RC3	Digital I/O
SCK	Synchronous serial clock input/output for SPI
SCL	Synchronous serial clock input/output for I^2C
RC4/SDI/SDA	
RC4	Digital I/O
SDI	SPI data in
SDA	I^2C data I/O
RC5/SDO	
RC5	Digital I/O
SDO	SPI data output
RC6/TX/CK	
RC6	Digital I/O
TX	USART transmit pin
CK	USART synchronous clock pin
RC7/RX/DT	
RC7	Digital I/O
RX	USART receive pin
DT	USART synchronous data pin

Table 2.9: PIC18F452 PORTD pin functions

Pin	Description
RD0/PSP0	
RD0	Digital I/O
PSP0	Parallel slave port bit 0
RD1/PSP1	
RD1	Digital I/O
PSP1	Parallel slave port bit 1
RD2/PSP2	
RD2	Digital I/O
PSP2	Parallel slave port bit 2
RD3/PSP3	
RD3	Digital I/O
PSP3	Parallel slave port bit 3
RD4/PSP4	
RD4	Digital I/O
PSP4	Parallel slave port bit 4
RD5/PSP5	
RD5	Digital I/O
PSP5	Parallel slave port bit 5
RD6/PSP6	
RD6	Digital I/O
PSP6	Parallel slave port bit 6
RD7/PSP7	
RD7	Digital I/O
PSP7	Parallel slave port bit 7

Table 2.10: PIC18F452 PORTE pin functions

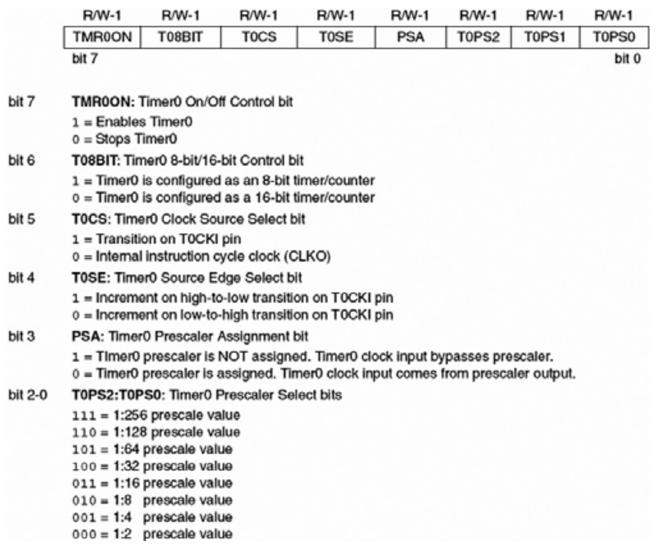
Pin	Description
RE0/RD/AN5	
RE0	Digital I/O
RD	Parallel slave port read control pin
AN5	Analog input 5
RE1/WR/AN6	
RE1	Digital I/O
WR	Parallel slave port write control pin
AN6	Analog input 6
RE2/CS/AN7	
RE2	Digital I/O
CS	Parallel slave port CS
AN7	Analog input 7

Timer 0 can be operated either as a timer or as a counter. Timer mode is selected by clearing the T0CS bit, and in this mode the clock to the timer is derived from $F_{OSC/4}$. Counter mode is selected by setting the T0CS bit, and in this mode Timer 0 is incremented on the rising or falling edge of input RA4/T0CKI. Bit T0SE of T0CON selects the edge triggering mode.

An 8-bit prescaler can be used to change the timer clock rate by a factor of up to 256. The prescaler is selected by bits PSA and T0PS2:T0PS0 of register T0CON.

8-Bit Mode Figure 2.25 shows Timer 0 in 8-bit mode. The following operations are normally carried out in a timer application:

- Clear T0CS to select clock $F_{OSC/4}$
- Use bits T0PS2:T0PS0 to select a suitable prescaler value
- Clear PSA to select the prescaler



R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT:** Timer0 8-bit/16-bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS:** Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE:** Timer0 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:** Timer0 Prescaler Assignment bit
1 = Timer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS2:T0PS0:** Timer0 Prescaler Select bits
111 = 1:256 prescale value
110 = 1:128 prescale value
101 = 1:64 prescale value
100 = 1:32 prescale value
011 = 1:16 prescale value
010 = 1:8 prescale value
001 = 1:4 prescale value
000 = 1:2 prescale value

Figure 2.24: Timer 0 control register, T0CON

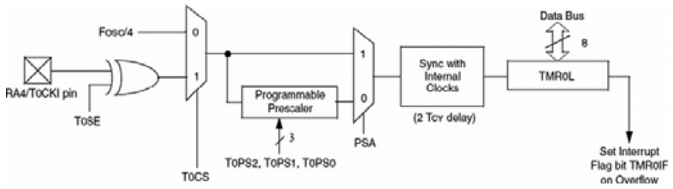


Figure 2.25: Timer 0 in 8-bit mode

- Load timer register TMR0L
- Optionally enable Timer 0 interrupts
- The timer counts up and an interrupt is generated when the timer value overflows from FFH to 00H in 8-bit mode (or from FFFFH to 0000H in 16-bit mode)

By loading a value into the TMR0 register we can control the count until an overflow occurs. The formula that follows can be used to calculate the time it will take for the timer to overflow (or to generate an interrupt) given the oscillator period, the value loaded into the timer, and the prescaler value:

$$\text{Overflow time} = 4 \times T_{OSC} \times \text{Prescaler} \times (256 - \text{TMR0}) \qquad (2.1)$$

where

Overflow time is in µs

T_{OSC} is the oscillator period in µs

Prescaler is the prescaler value

TMR0 is the value loaded into TMR0 register

For example, assume that we are using a 4MHz crystal, and the prescaler is chosen as 1:8 by setting bits PS2:PS0 to 010. Also assume that the value loaded into the timer register TMR0 is decimal 100. The overflow time is then given by:

$$\text{4MHZ clock has a period, } T = 1/f = 0.25\mu s$$

using the above formula

$$\text{Overflow time} = 4 \times 0.25 \times 8 \times (256 - 100) = 1248\mu s$$

Thus, the timer will overflow after 1.248msec, and a timer interrupt will be generated if the timer interrupt and global interrupts are enabled.

What we normally want is to know what value to load into the TMR0 register for a required overflow time. This can be calculated by modifying Equation (2.1) as follows:

$$\text{TMR0} = 256 - (\text{Overflow time})/(4 \times T_{OSC} \times \text{Prescaler}) \qquad (2.2)$$

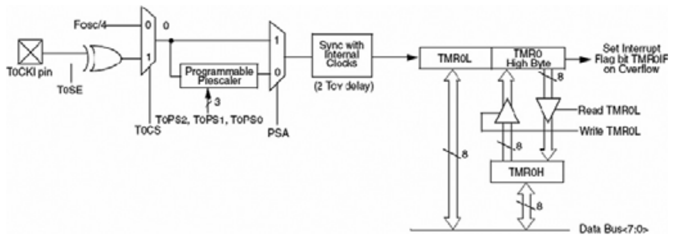


Figure 2.26: Timer 0 in 16-bit mode

For example, suppose we want an interrupt to be generated after 500μs and the clock and the prescaler values are as before. The value to be loaded into the TMR0 register can be calculated using Equation (2.2) as follows:

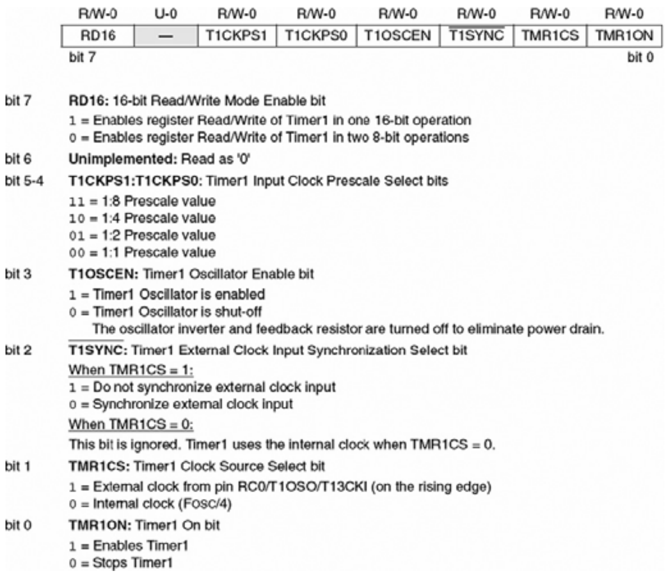
$$\text{TMR0} = 256 - 500/(4 \times 0.25 \times 8) = 193.5$$

The closest number we can load into TMR0 register is 193.

16-Bit Mode The Timer 0 in 16-bit mode is shown in Figure 2.26. Here, two timer registers named TMR0L and TMR0 are used to store the 16-bit timer value. The low byte TMR0L is directly loadable from the data bus. The high byte TMR0 can be loaded through a buffer called TMR0H. During a read of TMR0L, the high byte of the timer (TMR0) is also loaded into TMR0H, and thus all 16 bits of the timer value can be read. To read the 16-bit timer value, first we have to read TMR0L, and then read TMR0H in a later instruction. Similarly, during a write to TMR0L, the high byte of the timer is also updated with the contents of TMR0H, allowing all 16 bits to be written to the timer. Thus, to write to the timer the program should first write the required high byte to TMR0H. When the low byte is written to TMR0L, then the value stored in TMR0H is automatically transferred to TMR0, thus causing all 16 bits to be written to the timer.

Timer 1

PIC18F452 Timer 1 is a 16-bit timer controlled by register T1CON, as shown in Figure 2.27. Figure 2.28 shows the internal structure of Timer 1.



R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

bit 7 **RD16:** 16-bit Read/Write Mode Enable bit
1 = Enables register Read/Write of Timer1 in one 16-bit operation
0 = Enables register Read/Write of Timer1 in two 8-bit operations

bit 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 = Timer1 Oscillator is enabled
0 = Timer1 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 RC0/T1OSO/T13CKI (on the rising edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR1ON:** Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

Figure 2.27: Timer 1 control register, T1CON

Timer 1 can be operated as either a timer or a counter. When bit TMR1CS of register T1CON is low, clock $F_{OSC/4}$ is selected for the timer. When TMR1CS is high, the module operates as a counter clocked from input T1OSI. A crystal oscillator circuit, enabled from bit T1OSCEN of T1CON, is built between pins T1OSI and T1OSO where a crystal up to 200KHz can be connected between these pins. This oscillator is primarily intended for a 32KHz crystal operation in real-time clock applications. A prescaler is used in Timer 1 that can change the timing rate as a factor of 1, 2, 4, or 8.

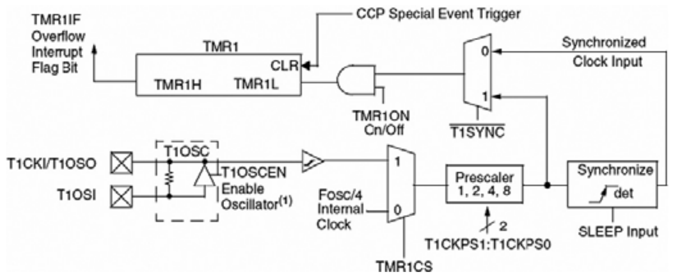


Figure 2.28: Internal structure of Timer 1

Timer 1 can be configured so that read/write can be performed either in 16-bit mode or in two 8-bit modes. Bit RD16 of register T1CON controls the mode. When RD16 is low, timer read and write operations are performed as two 8-bit operations. When RD16 is high, the timer read and write operations are as in Timer 0 16-bit mode (i.e., a buffer is used between the timer register and the data bus) (see Figure 2.29).

If the Timer 1 interrupts are enabled, an interrupt will be generated when the timer value rolls over from FFFFH to 0000H.

Timer 2

Timer 2 is an 8-bit timer with the following features:

- 8-bit timer (TMR2)
- 8-bit period register (PR2)
- Programmable prescaler
- Programmable postscaler
- Interrupt when TM2 matches PR2

Timer 2 is controlled from register T2CON, as shown in Figure 2.30. Bits T2CKPS1: T2CKPS0 set the prescaler for a scaling of 1, 4, and 16. Bits TOUTPS3:TOUTPS0 set

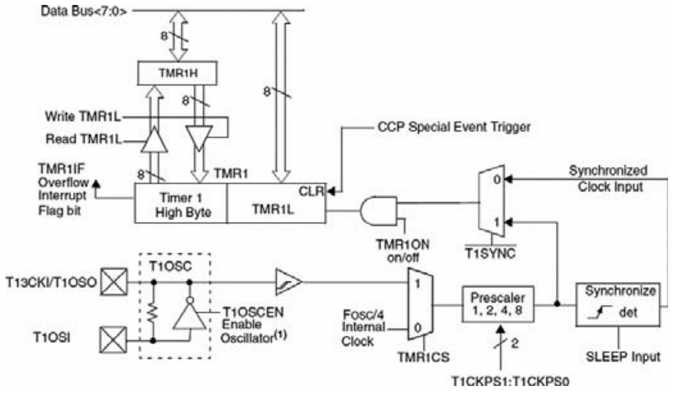
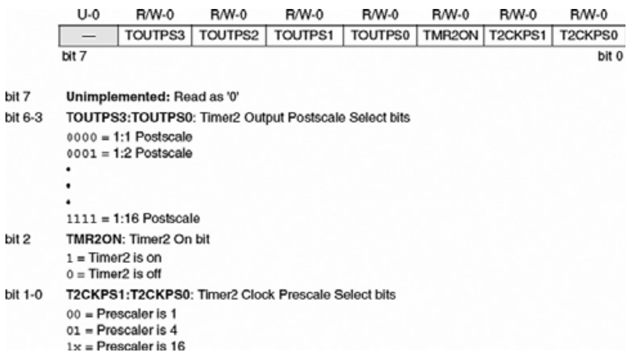


Figure 2.29: Timer 1 in 16-bit mode



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

Figure 2.30: Timer 2 control register, T2CON

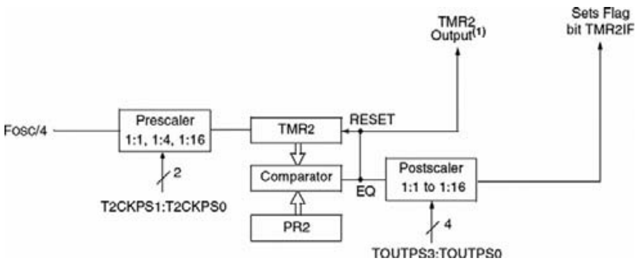


Figure 2.31: Timer 2 block diagram

the postscaler for a scaling of 1:1 to 1:16. The timer can be turned on or off by setting or clearing bit TMR2ON.

The block diagram of Timer 2 is shown in Figure 2.31. Timer 2 can be used for the PWM mode of the CCP module. The output of Timer 2 can be software selected by the SSP module as a baud clock. Timer 2 increments from 00H until it matches PR2 and sets the interrupt flag. It then resets to 00H on the next cycle.

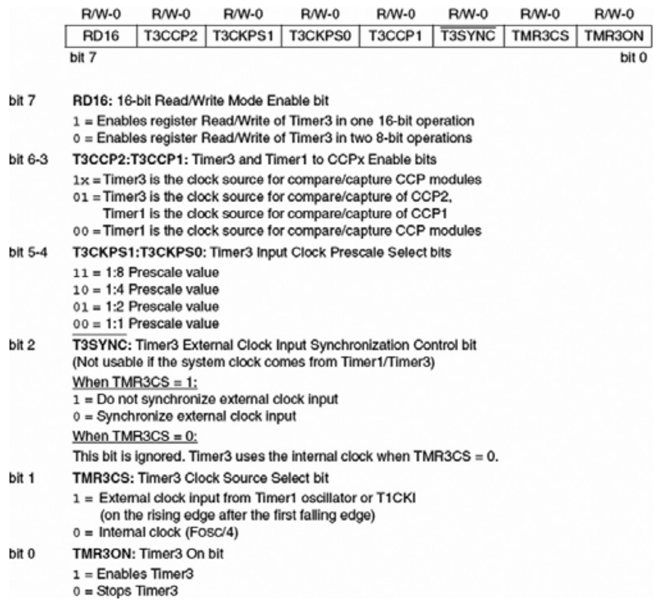
Timer 3

The structure and operation of Timer 3 is the same as for Timer 1, having registers TMR3H and TMR3L. This timer is controlled from register T3CON as shown in Figure 2.32.

The block diagram of Timer 3 is shown in Figure 2.33.

2.1.10 Capture/Compare/PWM Modules (CCP)

The PIC18F452 microcontroller has two capture/compare/PWM (CCP) modules, and they work with Timers 1, 2, and 3 to provide capture, compare, and pulse width modulation (PWM) operations. Each module has two 8-bit registers. Module 1 registers are CCPR1L and CCPR1H, and module 2 registers are CCPR2L and CCPR2H. Together, each register pair forms a 16-bit register and can be used to capture, compare, or generate waveforms with a specified duty cycle. Module 1 is controlled by register



R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON
bit 7							bit 0

bit 7 **RD16:** 16-bit Read/Write Mode Enable bit
1 = Enables register Read/Write of Timer3 in one 16-bit operation
0 = Enables register Read/Write of Timer3 in two 8-bit operations

bit 6-3 **T3CCP2:T3CCP1:** Timer3 and Timer1 to CCPx Enable bits
1x = Timer3 is the clock source for compare/capture CCP modules
01 = Timer3 is the clock source for compare/capture of CCP2,
Timer1 is the clock source for compare/capture of CCP1
00 = Timer1 is the clock source for compare/capture CCP modules

bit 5-4 **T3CKPS1:T3CKPS0:** Timer3 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 2 **$\overline{\text{T3SYNC}}$:** Timer3 External Clock Input Synchronization Control bit
(Not usable if the system clock comes from Timer1/Timer3)
When TMR3CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR3CS = 0:
This bit is ignored. Timer3 uses the internal clock when TMR3CS = 0.

bit 1 **TMR3CS:** Timer3 Clock Source Select bit
1 = External clock input from Timer1 oscillator or T1CKI
(on the rising edge after the first falling edge)
0 = Internal clock (Fosc/4)

bit 0 **TMR3ON:** Timer3 On bit
1 = Enables Timer3
0 = Stops Timer3

Figure 2.32: Timer 3 control register, T3CON

CCP1CON, and module 2 is controlled by CCP2CON. Figure 2.34 shows the bit allocations of the CCP control registers.

Capture Mode

In capture mode, the registers operate like a stopwatch. When an event occurs, the time of the event is recorded, although the clock continues running (a stopwatch, on the other hand, stops when the event time is recorded).

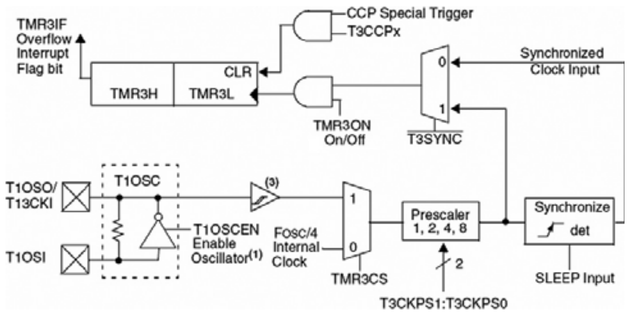


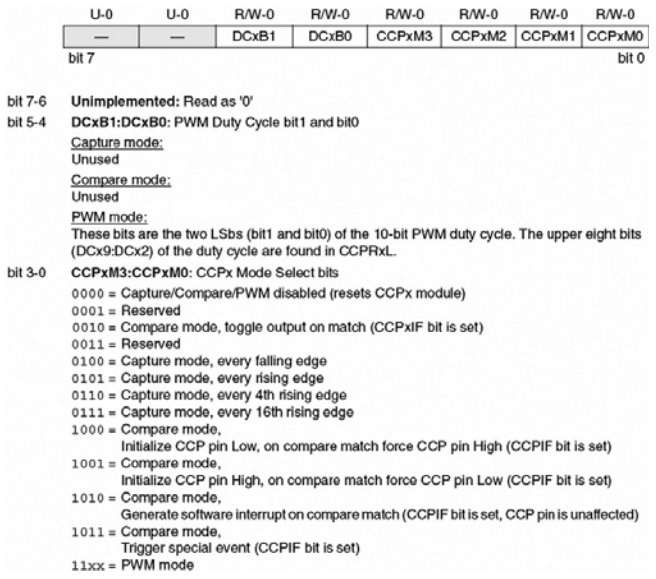
Figure 2.33: Block diagram of Timer 3

Figure 2.35 shows the capture mode of operation. Here, CCP1 will be considered, but the operation of CCP2 is identical with the register and port names changed accordingly. In this mode CCPR1H:CCPR1L captures the 16-bit value of the TMR1 or TMR3 registers when an event occurs on pin RC2/CCP1 (pin RC2/CCP1 must be configured as an input pin using TRISC). An external signal can be prescaled by 4 or 16. The event is selected by control bits CCP1M3:CCP1M0, and any of the following events can be selected:

- Every falling edge
- Every rising edge
- Every fourth rising edge
- Every sixteenth rising edge

If the capture interrupt is enabled, the occurrence of an event causes an interrupt to be generated in software. If another capture occurs before the value in register CCPR1 is read, the old captured value is overwritten by the new captured value.

Either Timer 1 or Timer 3 can be used in capture mode. They must be running in timer mode, or in synchronized counter mode, selected by register T3CON.



U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **DCxB1:DCxB0:** PWM Duty Cycle bit1 and bit0

Capture mode:
Unused

Compare mode:
Unused

PWM mode:
These bits are the two LSbs (bit1 and bit0) of the 10-bit PWM duty cycle. The upper eight bits (DCx9:DCx2) of the duty cycle are found in CCPRxL.

bit 3-0 **CCPxM3:CCPxM0:** CCPx Mode Select bits

```
0000 = Capture/Compare/PWM disabled (resets CCPx module)
0001 = Reserved
0010 = Compare mode, toggle output on match (CCPxIF bit is set)
0011 = Reserved
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode,
       Initialize CCP pin Low, on compare match force CCP pin High (CCPIF bit is set)
1001 = Compare mode,
       Initialize CCP pin High, on compare match force CCP pin Low (CCPIF bit is set)
1010 = Compare mode,
       Generate software interrupt on compare match (CCPIF bit is set, CCP pin is unaffected)
1011 = Compare mode,
       Trigger special event (CCPIF bit is set)
11xx = PWM mode
```

Figure 2.34: CCPxCON register bit allocations

Compare Mode

In compare mode, a digital comparator is used to compare the value of Timer 1 or Timer 3 to the value in a 16-bit register pair. When a match occurs, the output state of a pin is changed. Figure 2.36 shows the block diagram of compare mode in operation.

Here only module CCP1 is considered, but the operation of module CCP2 is identical.

The value of the 16-bit register pair CCPR1H:CCPR1L is continuously compared against the Timer 1 or Timer 3 value. When a match occurs, the state of the RC2/CCP1

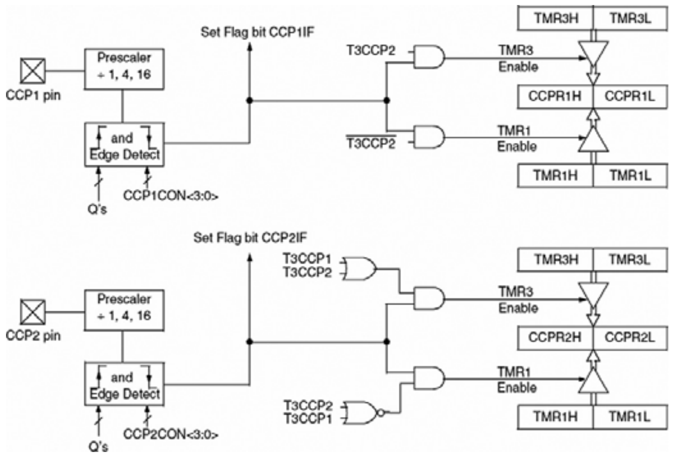


Figure 2.35: Capture mode of operation

pin is changed depending on the programming of bits CCP1M2:CCP1M0 of register CCP1CON. The following changes can be programmed:

- Force RC2/CCP1 high
- Force RC2/CCP1 low
- Toggle RC2/CCP1 pin (low to high or high to low)
- Generate interrupt when a match occurs
- No change

Timer 1 or Timer 3 must be running in timer mode or in synchronized counter mode, selected by register T3CON.

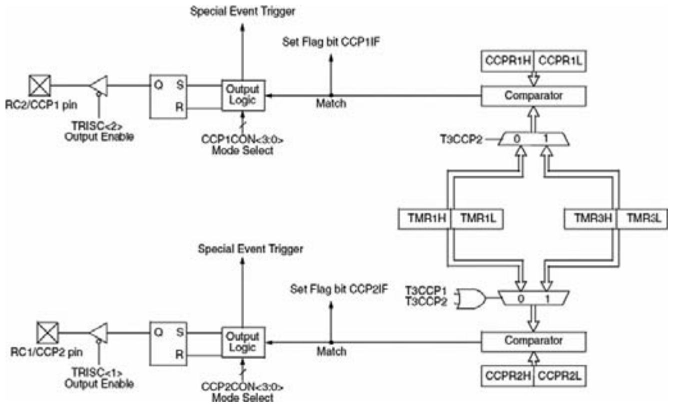


Figure 2.36: Compare mode of operation

PWM Module

The pulse width modulation (PWM) mode produces a PWM output at 10-bit resolution. A PWM output is basically a square waveform with a specified period and duty cycle. Figure 2.37 shows a typical PWM waveform.

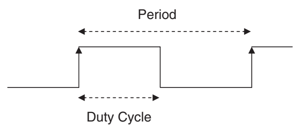


Figure 2.37: Typical PWM waveform

Figure 2.38 shows the PWM module block diagram. The module is controlled by Timer 2. The PWM period is given by:

$$\text{PWM period} = (\text{PR2} + 1)^{*}\text{TMR2PS}^{*}4^{*}\text{T}_{\text{OSC}} \tag{2.3}$$

or

$$\text{PR2} = \frac{\text{PWM period}}{\text{TMR2PS}^{*}4^{*}\text{T}_{\text{OSC}}} - 1 \tag{2.4}$$

where

PR2 is the value loaded into Timer 2 register

TMR2PS is the Timer 2 prescaler value

T_{OSC} is the clock oscillator period (seconds)

The PWM frequency is defined as 1/(PWM period).

The resolution of the PWM duty cycle is 10 bits. The PWM duty cycle is selected by writing the eight most significant bits into the CCPR1L register and the two least

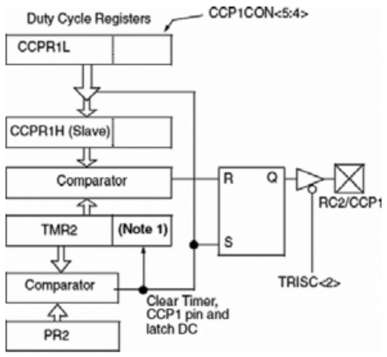


Figure 2.38: PWM module block diagram

significant bits into bits 4 and 5 of CCP1CON register. The duty cycle (in seconds) is given by:

$$\text{PWM duty cycle} = (\text{CCPR1L:CCP1CON} < 5{:}4 >)^{*}\text{TMR2PS}^{*}\text{T}_{\text{OSC}} \tag{2.5}$$

or

$$\text{CCPR1L:CCP1CON} < 5{:}4 > = \frac{\text{PWM duty cycle}}{\text{TMR2PS}^{*}\text{T}_{\text{OSC}}} \tag{2.6}$$

The steps to configure the PWM are as follows:

- Specify the required period and duty cycle.
- Choose a value for the Timer 2 prescaler (TMR2PS).
- Calculate the value to be written into the PR2 register using Equation (2.2).
- Calculate the value to be loaded into the CCPR1L and CCP1CON registers using Equation (2.6).
- Clear bit 2 of TRISC to make CCP1 pin an output pin.
- Configure the CCP1 module for PWM operation using register CCP1CON.

The following example shows how the PWM can be set up.

Example 2.1

PWM pulses must be generated from pin CCP1 of a PIC18F452 microcontroller. The required pulse period is 44µs and the required duty cycle is 50%. Assuming that the microcontroller operates with a 4MHz crystal, calculate the values to be loaded into the various registers.

Solution 2.1

$$\text{Using a 4MHz crystal}, \text{T}_{\text{OSC}} = 1/4 = 0.25 \times 10^{-6}$$

The required PWM duty cycle is 44/2 = 22µs.

From Equation (2.4), assuming a timer prescaler factor of 4, we have:

$$\text{PR2} = \frac{\text{PWM period}}{\text{TMR2PS}^{*}4^{*}\text{T}_{\text{OSC}}} - 1$$

or

$$\text{PR2} = \frac{44\times10^{-6}}{4^{*}4^{*}0.25\times10^{-6}} - 1 = 10 \qquad \text{i.e., 0AH}$$

and from Equation (2.6)

$$\text{CCPR1L:CCP1CON} < 5:4 > = \frac{\text{PWM duty cycle}}{\text{TMR2PS}^{*}\text{T}_{\text{OSC}}}$$

or

$$\text{CCPR1L:CCP1CON} < 5:4 > = \frac{22\times10^{-6}}{4^{*}0.25\times10^{-6}} = 22$$

But the equivalent of number 22 in 10-bit binary is:

"00 00010110"

Therefore, the value to be loaded into bits 4 and 5 of CCP1CON is "00." Bits 2 and 3 of CCP1CON must be set to high for PWM operation. Therefore, CCP1CON must be set to bit pattern ("X" is "don't care"):

XX001100

Taking the don't-care entries as 0, we can set CCP1CON to hexadecimal 0CH.

The value to be loaded into CCPR1L is "00010110" (i.e., hexadecimal number 16H).

The required steps are summarized as follows:

- Load Timer 2 with prescaler of 4 (i.e., load T2CON) with 00000101 (i.e., 05H).
- Load 0AH into PR2.
- Load 16H into CCPR1L.
- Load 0 into TRISC (make CCP1 pin output).
- Load 0CH into CCP1CON.

One period of the generated PWM waveform is shown in Figure 2.39.

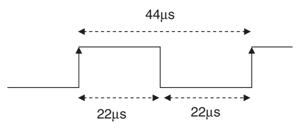


Figure 2.39: Generated PWM waveform

2.1.11 Analog-to-Digital Converter (A/D) Module

An analog-to-digital converter (A/D) is another important peripheral component of a microcontroller. The A/D converts an analog input voltage into a digital number so it can be processed by a microcontroller or any other digital system. There are many analog-to-digital converter chips available on the market, and an embedded systems designer should understand the characteristics of such chips so they can be used efficiently.

As far as the input and output voltage are concerned A/D converters can be classified as either unipolar and bipolar. Unipolar A/D converters accept unipolar input voltages in the range 0 to +0V, and bipolar A/D converters accept bipolar input voltages in the range ±V. Bipolar converters are frequently used in signal processing applications, where the signals by nature are bipolar. Unipolar converters are usually cheaper, and they are used in many control and instrumentation applications.

Figure 2.40 shows the typical steps involved in reading and converting an analog signal into digital form, a process also known as signal conditioning. Signals received from sensors usually need to be processed before being fed to an A/D converter. This

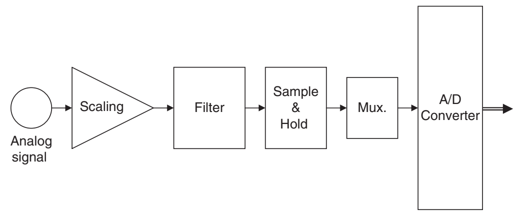


Figure 2.40: Signal conditioning and A/D conversion process

processing usually begins with scaling the signal to the correct value. Unwanted signal components are then removed by filtering the signal using classical filters (e.g., a low-pass filter). Finally, before feeding the signal to an A/D converter, the signal is passed through a sample-and-hold device. This is particularly important with fast real-time signals whose value may be changing between the sampling instants. A sample-and-hold device ensures that the signal stays at a constant value during the actual conversion process. Many applications required more than one A/D, which normally involves using an analog multiplexer at the input of the A/D. The multiplexer selects only one signal at any time and presents this signal to the A/D converter. An A/D converter usually has a single analog input and a digital parallel output. The conversion process is as follows:

- Apply the processed signal to the A/D input
- Start the conversion
- Wait until conversion is complete
- Read the converted digital data

The A/D conversion starts by triggering the converter. Depending on the speed of the converter, the conversion process itself can take several microseconds. At the end of the conversion, the converter either raises a flag or generates an interrupt to indicate that the conversion is complete. The converted parallel output data can then be read by the digital device connected to the A/D converter.

Most members of the PIC18F family contain a 10-bit A/D converter. If the chosen voltage reference is +5V, the voltage step value is:

$$\left(\frac{5\text{V}}{1023}\right) = 0.00489\text{V} \quad \text{or} \quad 4.89\text{mV}$$

Therefore, for example, if the input voltage is 1.0V, the converter will generate a digital output of 1.0/0.00489 = 205 decimal. Similarly, if the input voltage is 3.0V, the converter will generate 3.0/0.00489 = 613.

The A/D converter used by the PIC18F452 microcontroller has eight channels, named AN0–AN7, which are shared by the PORTA and PORTE pins. Figure 2.41 shows the block diagram of the A/D converter.

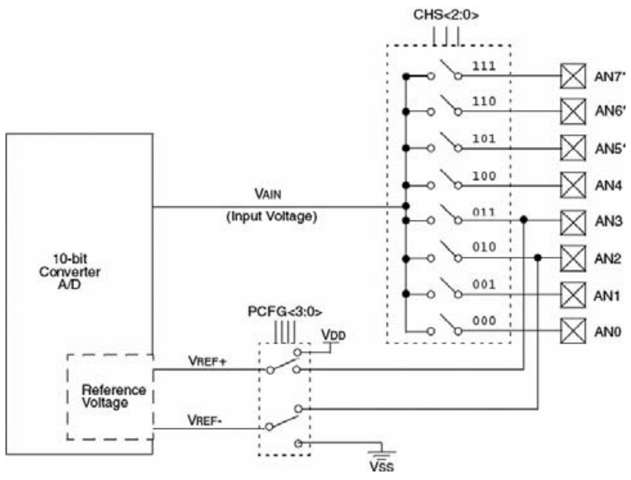
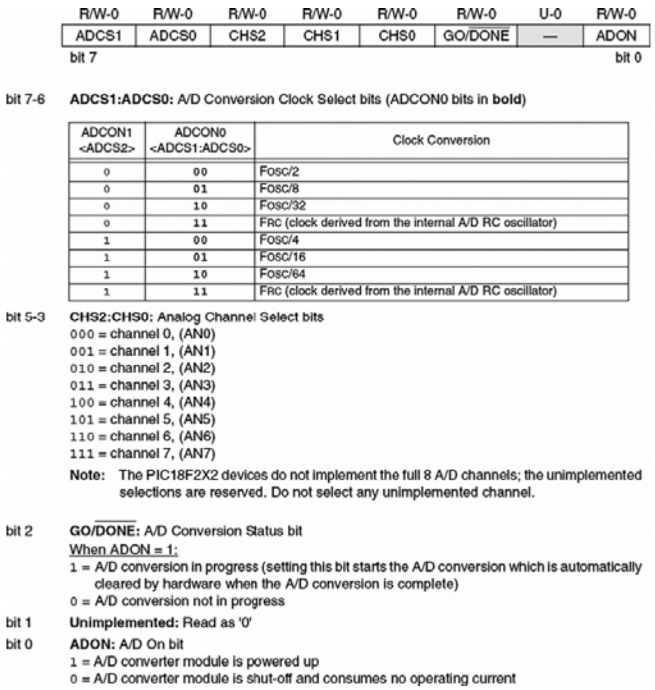


Figure 2.41: Block diagram of the PIC18F452 A/D converter

The A/D converter has four registers. Registers ADRESH and ADRESL store the higher and lower results of the conversion respectively. Register ADCON0, shown in Figure 2.42, controls the operation of the A/D module, such as selecting the conversion clock together with register ADCON1, selecting an input channel, starting a conversion, and powering up and shutting down the A/D converter.

Register ADCON1 (see Figure 2.43) is used for selecting the conversion format, configuring the A/D channels for analog input, selecting the reference voltage, and selecting the conversion clock together with register ADCON0.

A/D conversion starts by setting the GO/DONE bit of ADCON0. When the conversion is complete, the 2 bits of the converted data is written into register ADRESH, and the remaining 8 bits are written into register ADRESL. At the same time the GO/DONE bit is cleared to indicate the end of conversion. If required, interrupts can be enabled so that a software interrupt is generated when the conversion is complete.



R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

bit 7-6 **ADCS1:ADCS0:** A/D Conversion Clock Select bits (ADCON0 bits in **bold**)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	**00**	FOSC/2
0	**01**	FOSC/8
0	**10**	FOSC/32
0	**11**	FRC (clock derived from the internal A/D RC oscillator)
1	**00**	FOSC/4
1	**01**	FOSC/16
1	**10**	FOSC/64
1	**11**	FRC (clock derived from the internal A/D RC oscillator)

bit 5-3 **CHS2:CHS0:** Analog Channel Select bits
`000` = channel 0, (AN0)
`001` = channel 1, (AN1)
`010` = channel 2, (AN2)
`011` = channel 3, (AN3)
`100` = channel 4, (AN4)
`101` = channel 5, (AN5)
`110` = channel 6, (AN6)
`111` = channel 7, (AN7)

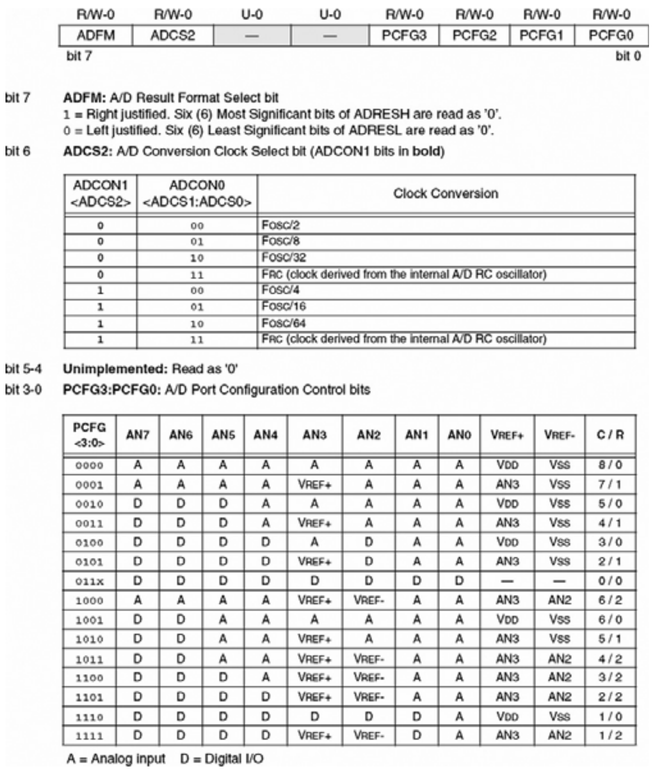
Note: The PIC18F2X2 devices do not implement the full 8 A/D channels; the unimplemented selections are reserved. Do not select any unimplemented channel.

bit 2 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
<u>When ADON = 1:</u>
`1` = A/D conversion in progress (setting this bit starts the A/D conversion which is automatically cleared by hardware when the A/D conversion is complete)
`0` = A/D conversion not in progress

bit 1 **Unimplemented:** Read as '0'

bit 0 **ADON:** A/D On bit
`1` = A/D converter module is powered up
`0` = A/D converter module is shut-off and consumes no operating current

Figure 2.42: ADCON0 register



R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7 **ADFM:** A/D Result Format Select bit
`1` = Right justified. Six (6) Most Significant bits of ADRESH are read as '0'.
`0` = Left justified. Six (6) Least Significant bits of ADRESL are read as '0'.

bit 6 **ADCS2:** A/D Conversion Clock Select bit (ADCON1 bits in **bold**)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	FOSC/2
0	01	FOSC/8
0	10	FOSC/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	FOSC/4
1	01	FOSC/16
1	10	FOSC/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

bit 5-4 **Unimplemented:** Read as '0'

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C / R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8 / 0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7 / 1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5 / 0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4 / 1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3 / 0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2 / 1
011x	D	D	D	D	D	D	D	D	—	—	0 / 0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6 / 2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6 / 0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5 / 1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4 / 2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3 / 2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2 / 2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1 / 0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1 / 2

A = Analog input D = Digital I/O

Figure 2.43: ADCON1 register

The steps in carrying out an A/D conversion are as follows:

- Use ADCON1 to configure required channels as analog and configure the reference voltage.
- Set the TRISA or TRISE bits so the required channel is an input port.
- Use ADCON0 to select the required analog input channel.
- Use ADCON0 and ADCON1 to select the conversion clock.
- Use ADCON0 to turn on the A/D module.
- Configure the A/D interrupt (if desired).
- Set the GO/DONE bit to start conversion.
- Wait until the GO/DONE bit is cleared, or until a conversion complete interrupt is generated.
- Read the converted data from ADRESH and ADRESL.
- Repeat these steps as required.

For correct A/D conversion, the A/D conversion clock must be selected to ensure a minimum bit conversion time of 1.6μs. Table 2.11 gives the recommended A/D clock sources for various microcontroller operating frequencies. For example, if the

Table 2.11: A/D conversion clock selection

A/D clock source		
Operation	**ADCS2:ADCS0**	**Maximum microcontroller frequency**
2 T_{OSC}	000	1.25 MHz
4 T_{OSC}	100	2.50 MHz
8 T_{OSC}	001	5.0 MHz
16 T_{OSC}	101	10.0 MHz
32 T_{OSC}	010	20.0 MHz
64 T_{OSC}	110	40.0 MHz
RC	011	–

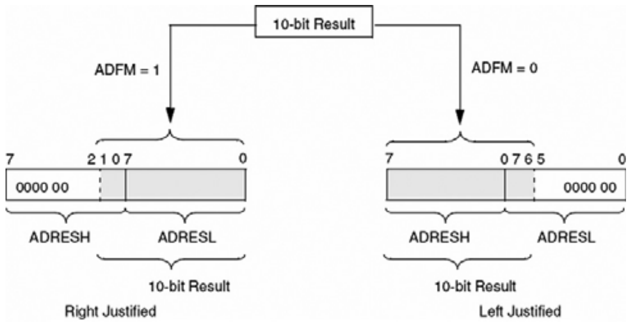


Figure 2.44: Formatting the A/D conversion result

microcontroller is operated from a 10MHz clock, the A/D clock source should be $F_{OSC/16}$ or higher (e.g., $F_{OSC/32}$).

Bit ADFM of register ADCON1 controls the format of a conversion. When ADFM is cleared, the 10-bit result is left justified (see Figure 2.44) and lower 6 bits of ADRESL are cleared to 0. When ADFM is set to 1 the result is right justified and the upper 6 bits of ADRESH are cleared to 0. This is the mode most commonly used, in which ADRESL contains the lower 8 bits, and bits 0 and 1 of ADRESH contain the upper 2 bits of the 10-bit result.

Analog Input Model and Acquisition Time

An understanding of the A/D analog input model is necessary to interface the A/D to external devices. Figure 2.45 shows the analog input model of the A/D. The analog input voltage V_{AIN} and the source resistance R_S are shown on the left side of the diagram. It is recommended that the source resistance be no greater than 2.5K. The analog signal is applied to the pin labeled ANx. There is a small capacitance (5pF) and a leakage current to the ground of approximately 500nA. R_{IC} is the interconnect resistance, which has a value of less than 1K. The sampling process is shown with switch SS having a resistance R_{SS} whose value depends on the voltage as shown in the

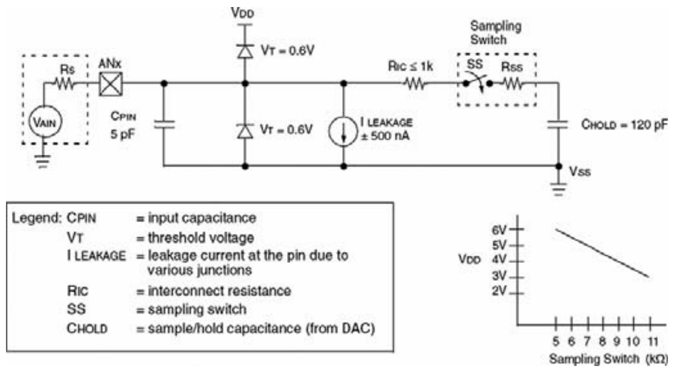


Figure 2.45: Analog input model of the A/D converter

small graph at the bottom of Figure 2.45. The value of R_{SS} is approximately 7K at 5V supply voltage.

The A/D converter is based on a switched capacitor principle, and capacitor C_{HOLD} shown in Figure 2.45 must be charged fully before the start of a conversion. This is a 120pF capacitor which is disconnected from the input pin once the conversion is started.

The acquisition time can be calculated by using Equation (2.7), provided by Microchip Inc:

$$T_{ACQ} = \text{Amplifier settling time} + \text{Holding capacitor charging time} + \text{temperature coefficient} \qquad (2.7)$$

The amplifier settling time is specified as a fixed 2μs. The temperature coefficient, which is only applicable if the temperature is above 25°C, is specified as:

$$\text{Temperature coefficient} = (\text{Temperature} - 25^\circ\text{C})(0.05\mu\text{s}/^\circ\text{C}) \qquad (2.8)$$

Equation (2.8) shows that the effect of the temperature is very small, creating about 0.5μs delay for every 10°C above 25°C. Thus, assuming a working environment

between 25°C and 35°C, the maximum delay due to temperature will be 0.5μs, which can be ignored for most practical applications.

The holding capacitor charging time as specified by Microchip Inc is:

$$\text{Holding capacitor charging time} = -(120\text{pF})(1\text{K} + R_{SS} + R_S)\text{Ln}(1/2048) \qquad (2.9)$$

Assuming that $R_{SS} = 7\text{K}$, $R_S = 2.5\text{K}$, Equation (2.9) gives the holding capacitor charging time as 9.6μs.

The acquisition time is then calculated as:

$$T_{ACQ} = 2 + 9.6 + 0.5 = 12.1\mu\text{s}$$

A full 10-bit conversion takes 12 A/D cycles, and each A/D cycle is specified at a minimum of 1.6μs. Thus, the fastest conversion time is 19.2μs. Adding this to the best possible acquisition time gives a total time to complete a conversion of 19.2 + 12.1 = 31.3μs.

When a conversion is complete, it is specified that the converter should wait for two conversion periods before starting a new conversion. This corresponds to $2 \times 1.6 = 3.2\mu\text{s}$. Adding this to the best possible conversion time of 31.3μs gives a complete conversion time of 34.5μs. Assuming the A/D converter is used successively, and ignoring the software overheads, this implies a maximum sampling frequency of about 29KHz.

2.1.12 Interrupts

An interrupt is an event that requires the CPU to stop normal program execution and then execute a program code related to the event causing the interrupt. Interrupts can be generated internally (by some event inside the chip) or externally (by some external event). An example of an internal interrupt is a timer overflowing or the A/D completing a conversion. An example of an external interrupt is an I/O pin changing state.

Interrupts can be useful in many applications such as:

- *Time critical applications*. Applications which require the immediate attention of the CPU can use interrupts. For example, in an emergency such as a power failure or fire in a plant the CPU may have to shut down the system immediately in an orderly manner. In such applications an external interrupt can force the CPU to stop whatever it is doing and take immediate action.

- *Performing routine tasks*. Many applications require the CPU to perform routine work at precise times, such as checking the state of a peripheral device exactly every millisecond. A timer interrupt scheduled with the required timing can divert the CPU from normal program execution to accomplish the task at the precise time required.
- *Task switching in multi-tasking applications*. In multi-tasking applications, each task may have a finite time to execute its code. Interrupt mechanisms can be used to stop a task should it consume more than its allocated time.
- *To service peripheral devices quickly*. Some applications may need to know when a task, such as an A/D conversion, is completed. This can be accomplished by continuously checking the completion flag of the A/D converter. A more elegant solution would be to enable the A/D completion interrupt so the CPU is forced to read the converted data as soon as it becomes available.

The PIC18F452 microcontroller has both core and peripheral interrupt sources. The core interrupt sources are:

- External edge-triggered interrupt on INT0, INT1, and INT2 pins.
- PORTB pins change interrupts (any one of the RB4–RB7 pins changing state)
- Timer 0 overflow interrupt

The peripheral interrupt sources are:

- Parallel slave port read/write interrupt
- A/D conversion complete interrupt
- USART receive interrupt
- USART transmit interrupt
- Synchronous serial port interrupt
- CCP1 interrupt
- TMR1 overflow interrupt
- TMR2 overflow interrupt
- Comparator interrupt

- EEPROM/FLASH write interrupt
- Bus collision interrupt
- Low-voltage detect interrupt
- Timer 3 overflow interrupt
- CCP2 interrupt

Interrupts in the PIC18F family can be divided into two groups: high priority and low priority. Applications that require more attention can be placed in the higher priority group. A high-priority interrupt can stop a low-priority interrupt that is in progress and gain access to the CPU. However, high-priority interrupts cannot be stopped by low-priority interrupts. If the application does not need to set priorities for interrupts, the user can choose to disable the priority scheme so all interrupts are at the same priority level. High-priority interrupts are vectored to address 00008H and low-priority ones to address 000018H of the program memory. Normally, a user program code (interrupt service routine, ISR) should be at the interrupt vector address to service the interrupting device.

In the PIC18F452 microcontroller there are ten registers that control interrupt operations. These are:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2
- PIE1, PIE2
- IPR1, IPR2

Every interrupt source (except INT0) has three bits to control its operation. These bits are:

- A flag bit to indicate whether an interrupt has occurred. This bit has a name ending in . . .**IF**

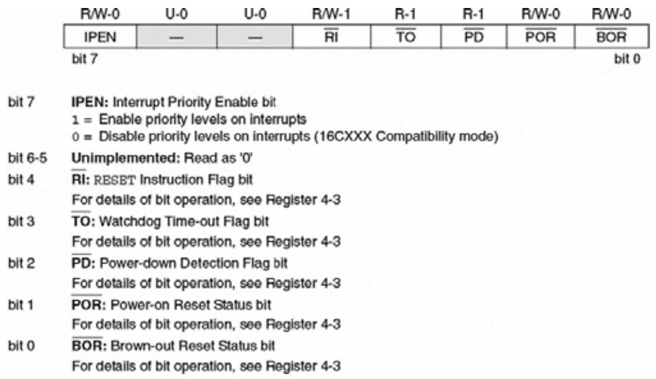
- An interrupt enable bit to enable or disable the interrupt source. This bit has the name ending in . . .**IE**
- A priority bit to select high or low priority. This bit has a name ending in . . .**IP**

RCON Register

The top bit of the RCON register, called IPEN, is used to enable the interrupt priority scheme. When IPEN = 0, interrupt priority levels are disabled and the microcontroller interrupt structure is similar to that of the PIC16 series. When IPEN = 1, interrupt priority levels are enabled. Figure 2.46 shows the bits of register RCON.

Enabling/Disabling Interrupts—No Priority Structure

When the IPEN bit is cleared, the priority feature is disabled. All interrupts branch to address 00008H of the program memory. In this mode, bit PEIE of register INTCON enables/disables all peripheral interrupt sources. Similarly, bit GIE of INTCON enables/disables all interrupt sources. Figure 2.47 shows the bits of register INTCON.



R/W-0	U-0	U-0	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	—	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 0

bit 7 **IPEN:** Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts (16CXXX Compatibility mode)

bit 6-5 **Unimplemented:** Read as '0'

bit 4 **$\overline{RI}$:** $\overline{RESET}$ Instruction Flag bit
For details of bit operation, see Register 4-3

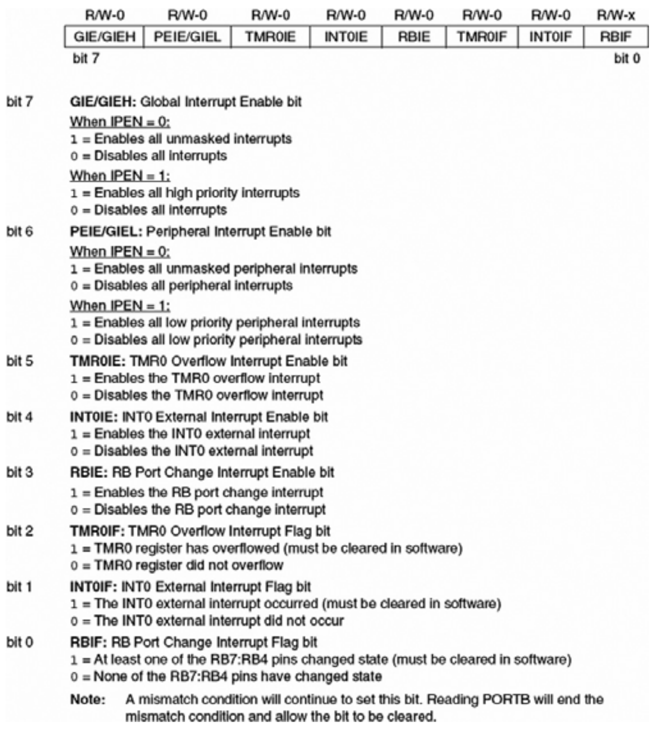
bit 3 **$\overline{TO}$:** Watchdog Time-out Flag bit
For details of bit operation, see Register 4-3

bit 2 **$\overline{PD}$:** Power-down Detection Flag bit
For details of bit operation, see Register 4-3

bit 1 **$\overline{POR}$:** Power-on Reset Status bit
For details of bit operation, see Register 4-3

bit 0 **$\overline{BOR}$:** Brown-out Reset Status bit
For details of bit operation, see Register 4-3

Figure 2.46: RCON register bits



R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
bit 7							bit 0

bit 7 **GIE/GIEH:** Global Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked interrupts
0 = Disables all interrupts
When IPEN = 1:
1 = Enables all high priority interrupts
0 = Disables all interrupts

bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts
When IPEN = 1:
1 = Enables all low priority peripheral interrupts
0 = Disables all low priority peripheral interrupts

bit 5 **TMR0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 overflow interrupt
0 = Disables the TMR0 overflow interrupt

bit 4 **INT0IE:** INT0 External Interrupt Enable bit
1 = Enables the INT0 external interrupt
0 = Disables the INT0 external interrupt

bit 3 **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INT0IF:** INT0 External Interrupt Flag bit
1 = The INT0 external interrupt occurred (must be cleared in software)
0 = The INT0 external interrupt did not occur

bit 0 **RBIF:** RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Note: A mismatch condition will continue to set this bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared.

Figure 2.47: INTCON register bits

For an interrupt to be accepted by the CPU the following conditions must be satisfied:

- The interrupt enable bit of the interrupt source must be enabled. For example, if the interrupt source is external interrupt pin INT0, then bit INT0IE of register INTCON must be set to 1.
- The interrupt flag of the interrupt source must be cleared. For example, if the interrupt source is external interrupt pin INT0, then bit INT0IF of register INTCON must be cleared to 0.
- The peripheral interrupt enable/disable bit PEIE of INTCON must be set to 1 if the interrupt source is a peripheral.
- The global interrupt enable/disable bit GIE of INTCON must be set to 1.

With an external interrupt source we normally have to define whether the interrupt should occur on the low-to-high or high-to-low transition of the interrupt source. With INT0 interrupts, for example, this is done by setting/clearing bit INTEDG0 of register INTCON2.

When an interrupt occurs, the CPU stops its normal flow of execution, pushes the return address onto the stack, and jumps to address 00008H in the program memory where the user interrupt service routine program resides. Once the CPU is in the interrupt service routine, the global interrupt enable bit (GIE) is cleared to disable further interrupts. When multiple interrupt sources are enabled, the source of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bits must be cleared in the software before reenabling interrupts to avoid recursive interrupts. When the CPU has returned from the interrupt service routine, the global interrupt bit GIE is automatically set by the software.

Enabling/Disabling Interrupts—Priority Structure

When the IPEN bit is set to 1, the priority feature is enabled and the interrupts are grouped into two: low priority and high priority. Low-priority interrupts branch to address 00008H and high-priority interrupts branch to address 000018H of the program memory. Setting the priority bit makes the interrupt source a high-priority interrupt, and clearing this bit makes the interrupt source a low-priority interrupt.

Setting the GIEH bit of INTCON enables all high-priority interrupts that have the priority bit set. Similarly, setting the GIEL bit of INTCON enables all low-priority interrupts (the priority is bit cleared).

For a high-priority interrupt to be accepted by the CPU, the following conditions must be satisfied:

- The interrupt enable bit of the interrupt source must be enabled. For example, if the interrupt source is external interrupt pin INT1, then bit INT1IE of register INTCON3 must be set to 1.
- The interrupt flag of the interrupt source must be cleared. For example, if the interrupt source is external interrupt pin INT1, then bit INT1IF of register INTCON3 must be cleared to 0.
- The priority bit must be set to 1. For example, if the interrupt source is external interrupt INT1, then bit INT1P of register INTCON3 must be set to 1.
- The global interrupt enable/disable bit GIEH of INTCON must be set to 1.

For a low-priority interrupt to be accepted by the CPU, the following conditions must be satisfied:

- The interrupt enable bit of the interrupt source must be enabled. For example, if the interrupt source is external interrupt pin INT1, then bit INT1IE of register INTCON3 must be set to 1.
- The interrupt flag of the interrupt source must be cleared. For example, if the interrupt source is external interrupt pin INT1, then bit INT1IF of register INTCON3 must be cleared to 0.
- The priority bit must be cleared to 0. For example, if the interrupt source is external interrupt INT1, then bit INT1P of register INTCON3 must be cleared to 0.
- Low-priority interrupts must be enabled by setting bit GIEL of INTCON to 1.
- The global interrupt enable/disable bit GIEH of INTCON must be set to 1.

Table 2.12 gives a listing of the PIC18F452 microcontroller interrupt bit names and register names for every interrupt source.

Table 2.12: PIC18F452 interrupt bits and registers

Interrupt source	Flag bit	Enable bit	Priority bit
INT0 external	INT0IF	INT0IE	–
INT1 external	INT1IF	INT1IE	INT1IP
INT2 external	INT2IF	INT2IE	INT2IP
RB port change	RBIF	RBIE	RBIP
TMR0 overflow	TMR0IF	TMR0IE	TMR0IP
TMR1overflow	TMR1IF	TMR1IE	TMR1IP
TMR2 match PR2	TMR2IF	TMR2IE	TMR2IP
TMR3 overflow	TMR3IF	TMR3IE	TMR3IP
A/D complete	ADIF	ADIE	ADIP
CCP1	CCP1IF	CCP1IE	CCP1IP
CCP2	CCP2IF	CCP2IE	CCP2IP
USART RCV	RCIF	RCIE	RCIP
USART TX	TXIF	TXIE	TXIP
Parallel slave port	PSPIF	PSPIE	PSPIP
Sync serial port	SSPIF	SSPIE	SSPIP
Low-voltage detect	LVDIF	LVDIE	LVDIP
Bus collision	BCLIF	BCLIE	BCLIP
EEPROM/FLASH write	EEIF	EEIE	EEIP

Figures 2.48 to 2.55 show the bit definitions of interrupt registers INTCON2, INTCON3, PIR1, PIR2, PIE1, PIE2, IPR1, and IPR2.

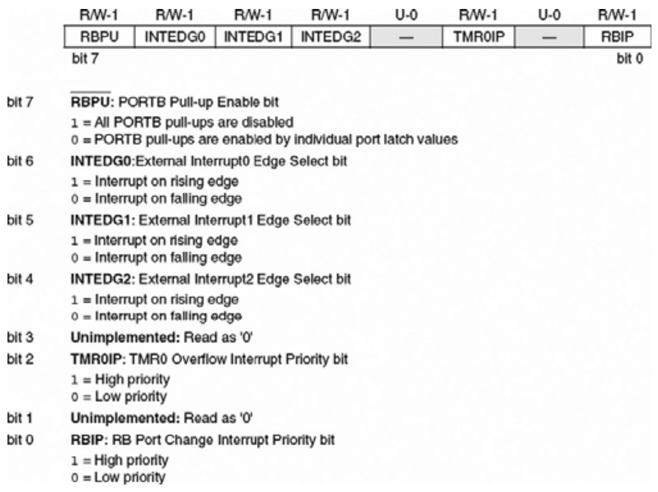
Examples are given in this section to illustrate how the CPU can be programmed for an interrupt.

Example 2.2

Set up INT1 as a falling-edge triggered interrupt input having low priority.

Solution 2.2

The following bits should be set up before the INT1 falling-edge triggered interrupts can be accepted by the CPU in low-priority mode:



R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP
bit 7							bit 0

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

bit 6 **INTEDG0:**External Interrupt0 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 5 **INTEDG1:** External Interrupt1 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 4 **INTEDG2:** External Interrupt2 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

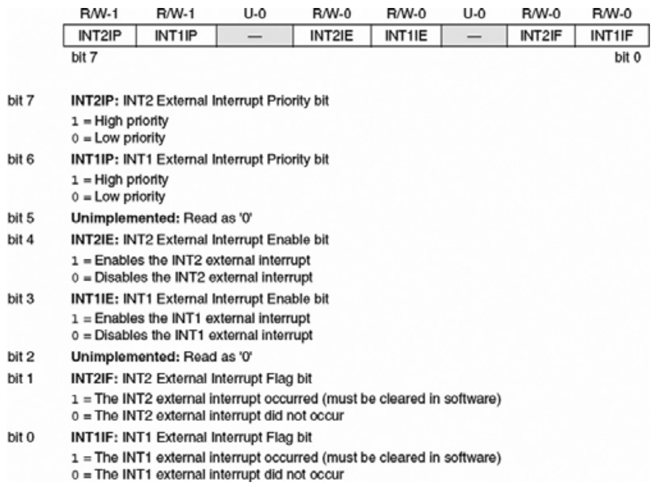
bit 3 **Unimplemented:** Read as '0'

bit 2 **TMR0IP:** TMR0 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **Unimplemented:** Read as '0'

bit 0 **RBIP:** RB Port Change Interrupt Priority bit
1 = High priority
0 = Low priority

Figure 2.48: INTCON2 bit definitions



R/W-1	R/W-1	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF
bit 7							bit 0

bit 7 **INT2IP:** INT2 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **INT1IP:** INT1 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **Unimplemented:** Read as '0'

bit 4 **INT2IE:** INT2 External Interrupt Enable bit
1 = Enables the INT2 external interrupt
0 = Disables the INT2 external interrupt

bit 3 **INT1IE:** INT1 External Interrupt Enable bit
1 = Enables the INT1 external interrupt
0 = Disables the INT1 external interrupt

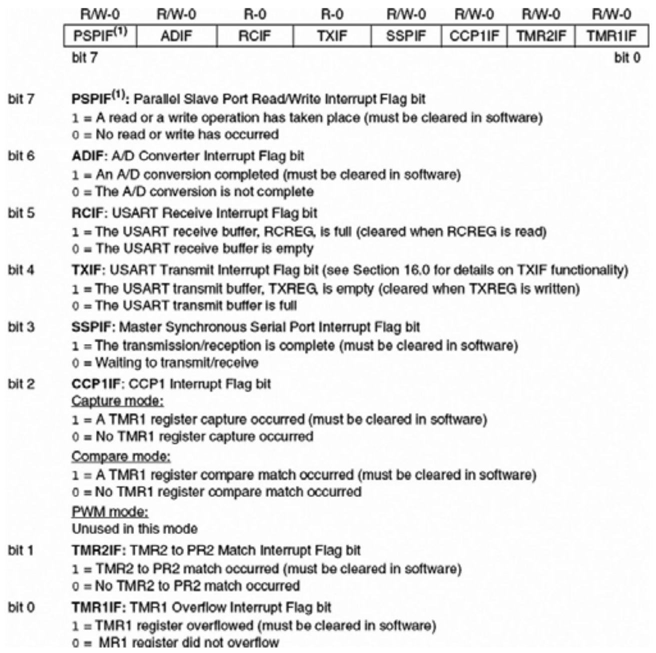
bit 2 **Unimplemented:** Read as '0'

bit 1 **INT2IF:** INT2 External Interrupt Flag bit
1 = The INT2 external interrupt occurred (must be cleared in software)
0 = The INT2 external interrupt did not occur

bit 0 **INT1IF:** INT1 External Interrupt Flag bit
1 = The INT1 external interrupt occurred (must be cleared in software)
0 = The INT1 external interrupt did not occur

Figure 2.49: INTCON3 bit definitions

- Enable the priority structure. Set IPEN = 1
- Make INT1 an input pin. Set TRISB = 1
- Set INT1 interrupts for falling edge. SET INTEDG1 = 0
- Enable INT1 interrupts. Set INT1IE = 1
- Enable low priority. Set INT1IP = 0
- Clear INT1 flag. Set INT1IF = 0
- Enable low-priority interrupts. Set GIEL = 1
- Enable all interrupts. Set GIEH = 1



R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

bit 7 **PSPIF(1):** Parallel Slave Port Read/Write Interrupt Flag bit
1 = A read or a write operation has taken place (must be cleared in software)
0 = No read or write has occurred

bit 6 **ADIF:** A/D Converter Interrupt Flag bit
1 = An A/D conversion completed (must be cleared in software)
0 = The A/D conversion is not complete

bit 5 **RCIF:** USART Receive Interrupt Flag bit
1 = The USART receive buffer, RCREG, is full (cleared when RCREG is read)
0 = The USART receive buffer is empty

bit 4 **TXIF:** USART Transmit Interrupt Flag bit (see Section 16.0 for details on TXIF functionality)
1 = The USART transmit buffer, TXREG, is empty (cleared when TXREG is written)
0 = The USART transmit buffer is full

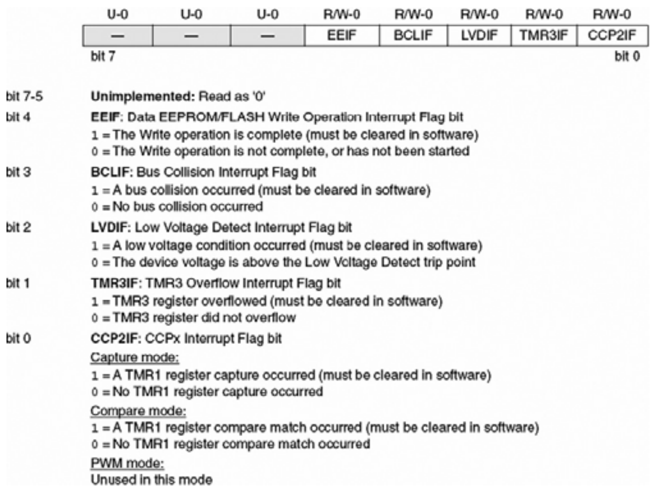
bit 3 **SSPIF:** Master Synchronous Serial Port Interrupt Flag bit
1 = The transmission/reception is complete (must be cleared in software)
0 = Waiting to transmit/receive

bit 2 **CCP1IF:** CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode

bit 1 **TMR2IF:** TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF:** TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = MR1 register did not overflow

Figure 2.50: PIR1 bit definitions



U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF
bit 7							bit 0

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **EEIF**: Data EEPROM/FLASH Write Operation Interrupt Flag bit
1 = The Write operation is complete (must be cleared in software)
0 = The Write operation is not complete, or has not been started

bit 3 **BCLIF**: Bus Collision Interrupt Flag bit
1 = A bus collision occurred (must be cleared in software)
0 = No bus collision occurred

bit 2 **LVDIF**: Low Voltage Detect Interrupt Flag bit
1 = A low voltage condition occurred (must be cleared in software)
0 = The device voltage is above the Low Voltage Detect trip point

bit 1 **TMR3IF**: TMR3 Overflow Interrupt Flag bit
1 = TMR3 register overflowed (must be cleared in software)
0 = TMR3 register did not overflow

bit 0 **CCP2IF**: CCPx Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode

Figure 2.51: PIR2 bit definitions

When an interrupt occurs, the CPU jumps to address 00008H in the program memory to execute the user program at the interrupt service routine.

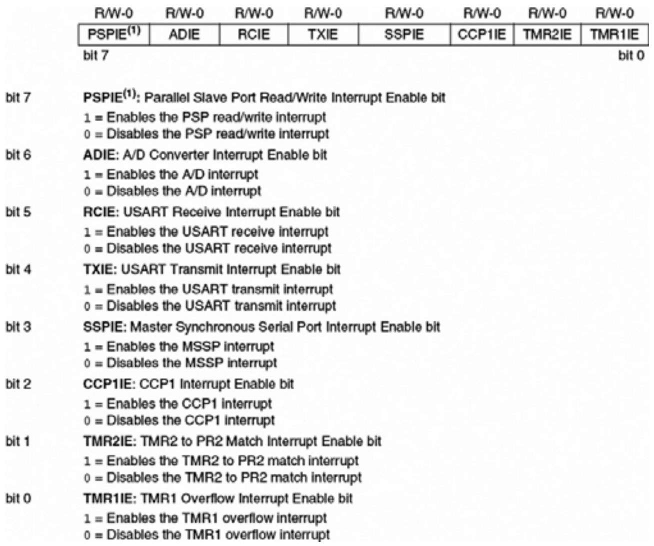
Example 2.3

Set up INT1 as a rising-edge triggered interrupt input having high priority.

Solution 2.3

The following bits should be set up before the INT1 rising-edge triggered interrupts can be accepted by the CPU in high-priority mode:

- Enable the priority structure. Set IPEN = 1
- Make INT1 an input pin. Set TRISB = 1



R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7 **PSPIE(1)**: Parallel Slave Port Read/Write Interrupt Enable bit
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt

bit 6 **ADIE**: A/D Converter Interrupt Enable bit
1 = Enables the A/D interrupt
0 = Disables the A/D interrupt

bit 5 **RCIE**: USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit 4 **TXIE**: USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

bit 3 **SSPIE**: Master Synchronous Serial Port Interrupt Enable bit
1 = Enables the MSSP interrupt
0 = Disables the MSSP interrupt

bit 2 **CCP1IE**: CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

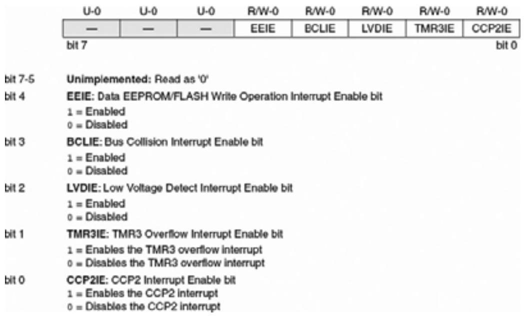
bit 1 **TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE**: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

Figure 2.52: PIE1 bit definitions

- Set INT1 interrupts for rising edge. SET INTEDG1 = 1
- Enable INT1 interrupts. Set INT1IE = 1
- Enable high priority. Set INT1IP = 1
- Clear INT1 flag. Set INT1IF = 0
- Enable all interrupts. Set GIEH = 1

When an interrupt occurs, the CPU jumps to address 000018H of the program memory to execute the user program at the interrupt service routine.



U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE
bit 7							bit 0

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **EEIE:** Data EEPROM/FLASH Write Operation Interrupt Enable bit
1 = Enabled
0 = Disabled

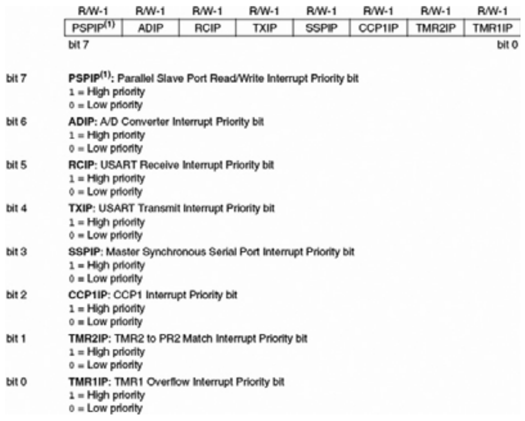
bit 3 **BCLIE:** Bus Collision Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **LVDIE:** Low Voltage Detect Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **TMR3IE:** TMR3 Overflow Interrupt Enable bit
1 = Enables the TMR3 overflow interrupt
0 = Disables the TMR3 overflow interrupt

bit 0 **CCP2IE:** CCP2 Interrupt Enable bit
1 = Enables the CCP2 interrupt
0 = Disables the CCP2 interrupt

Figure 2.53: PIE2 bit definitions



R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
PSPIP[(1)]	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
bit 7							bit 0

bit 7 **PSPIP[(1)]:** Parallel Slave Port Read/Write Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **ADIP:** A/D Converter Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **RCIP:** USART Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **TXIP:** USART Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

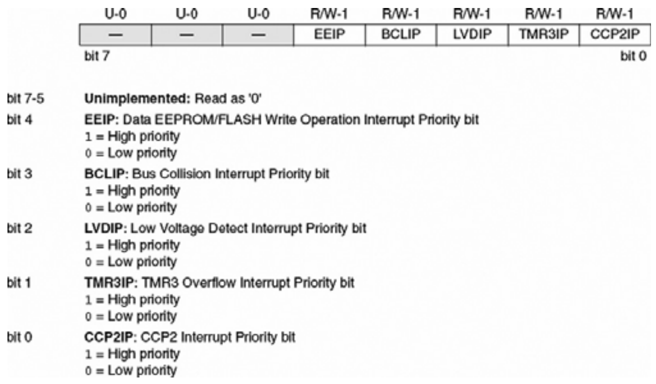
bit 3 **SSPIP:** Master Synchronous Serial Port Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **CCP1IP:** CCP1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR2IP:** TMR2 to PR2 Match Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR1IP:** TMR1 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

Figure 2.54: IPR1 bit definitions



U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP
bit 7							bit 0

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **EEIP:** Data EEPROM/FLASH Write Operation Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **BCLIP:** Bus Collision Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **LVDIP:** Low Voltage Detect Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR3IP:** TMR3 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **CCP2IP:** CCP2 Interrupt Priority bit
1 = High priority
0 = Low priority

Figure 2.55: IPR2 bit definitions

2.2 Summary

This chapter has described the architecture of the PIC18F family of microcontrollers. The PIC18F452 was used as a typical sample microcontroller in this family. Other members of the same family, such as the PIC18F242, have smaller pin counts and less functionality. And some, such as the PIC18F6680, have larger pin counts and more functionality.

Important parts and peripheral circuits of the PIC18F series have been described, including data memory, program memory, clock circuits, reset circuits, watchdog timer, general purpose timers, capture and compare module, PWM module, A/D converter, and the interrupt structure.

2.3 Exercises

1. Describe the data memory structure of the PIC18F452 microcontroller. What is a bank? How many banks are there?
2. Explain the differences between a general purpose register (GPR) and a special function register (SFR).

3. Explain the various ways the PIC18F microcontroller can be reset. Draw a circuit diagram to show how an external push-button switch can be used to reset the microcontroller.
4. Describe the various clock sources that can be used to provide a clock to a PIC18F452 microcontroller. Draw a circuit diagram to show how a 10MHz crystal can be connected to the microcontroller.
5. Draw a circuit diagram to show how a resonator can be connected to a PIC18F microcontroller.
6. In a non-time-critical application a clock must be provided for a PIC18F452 microcontroller using an external resistor and a capacitor. Draw a circuit diagram to show how this can be done and find the component values for a required clock frequency of 5MHz.
7. Explain how an external clock can provide clock pulses to a PIC18F microcontroller.
8. What are the registers of PORTA? Explain the operation of the port by drawing the port block diagram.
9. The watchdog timer must be set to provide an automatic reset every 0.5 seconds. Describe how to do this, including the appropriate register bits.
10. PWM pulses must be generated from pin CCP1 of a PIC18F452 microcontroller. The required pulse period is 100μs, and the required duty cycle is 50%. Assuming the microcontroller is operating with a 4MHz crystal, calculate the values to be loaded into the various registers.
11. Again, with regard to PWM pulses generated from pin CCP1 of a PIC18F452 microcontroller: If the required pulse frequency is 40KHz, and the required duty cycle is 50%, and assuming the microcontroller is operating with a 4MHz crystal, calculate the values to be loaded into the various registers.
12. An LM35DZ-type analog temperature sensor is connected to analog port AN0 of a PIC18F452 microcontroller. The sensor provides an analog output voltage proportional to the temperature (i.e., V0 = 10 mV/°C). Show the steps required to read the temperature.
13. Explain the difference between a priority interrupt and a nonpriority interrupt.

14. Show the steps required to set up INT2 as a falling-edge triggered interrupt input having low priority. What is the interrupt vector address?
15. Show the steps required to set up both INT1 and INT2 as falling-edge triggered interrupt inputs having low priority.
16. Show the steps required to set up INT1 as falling-edge triggered and INT2 as rising-edge triggered interrupt inputs having high priorities. Explain how to find the source of the interrupt when an interrupt occurs.
17. Show the steps required to set up Timer 0 to generate interrupts every millisecond with a high priority. What is the interrupt vector address?
18. In an application the CPU registers have been configured to accept interrupts from external sources INT0, INT1, and INT2. An interrupt has been detected. Explain how to find the source of the interrupt.