

Welcome to [E-XFL.COM](http://E-XFL.COM)

### What is "[Embedded - Microcontrollers](#)"?

"[Embedded - Microcontrollers](#)" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

### Applications of "[Embedded - Microcontrollers](#)"

#### Details

Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	32MHz
Connectivity	I <sup>2</sup> C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LCD, POR, PWM, WDT
Number of I/O	24
Program Memory Size	28KB (16K x 14)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 20x12b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	<a href="https://www.e-xfl.com/product-detail/microchip-technology/pic16lf19156-i-so">https://www.e-xfl.com/product-detail/microchip-technology/pic16lf19156-i-so</a>

## 1.1 Register and Bit Naming Conventions

### 1.1.1 REGISTER NAMES

When there are multiple instances of the same peripheral in a device, the peripheral control registers will be depicted as the concatenation of a peripheral identifier, peripheral instance, and control identifier. The control registers section will show just one instance of all the register names with an 'x' in the place of the peripheral instance number. This naming convention may also be applied to peripherals when there is only one instance of that peripheral in the device to maintain compatibility with other devices in the family that contain more than one.

### 1.1.2 BIT NAMES

There are two variants for bit names:

- Short name: Bit function abbreviation
- Long name: Peripheral abbreviation + short name

#### 1.1.2.1 Short Bit Names

Short bit names are an abbreviation for the bit function. For example, some peripherals are enabled with the EN bit. The bit names shown in the registers are the short name variant.

Short bit names are useful when accessing bits in C programs. The general format for accessing bits by the short name is *RegisterName*bits.*ShortName*. For example, the enable bit, EN, in the COG1CON0 register can be set in C programs with the instruction `COG1CON0bits.EN = 1`.

Short names are generally not useful in assembly programs because the same name may be used by different peripherals in different bit positions. When this occurs, during the include file generation, all instances of that short bit name are appended with an underscore plus the name of the register in which the bit resides to avoid naming contentions.

#### 1.1.2.2 Long Bit Names

Long bit names are constructed by adding a peripheral abbreviation prefix to the short name. The prefix is unique to the peripheral, thereby making every long bit name unique. The long bit name for the COG1 enable bit is the COG1 prefix, G1, appended with the enable bit short name, EN, resulting in the unique bit name G1EN.

Long bit names are useful in both C and assembly programs. For example, in C the COG1CON0 enable bit can be set with the `G1EN = 1` instruction. In assembly, this bit can be set with the `BSF COG1CON0,G1EN` instruction.

### 1.1.2.3 Bit Fields

Bit fields are two or more adjacent bits in the same register. Bit fields adhere only to the short bit naming convention. For example, the three Least Significant bits of the COG1CON0 register contain the mode control bits. The short name for this field is MD. There is no long bit name variant. Bit field access is only possible in C programs. The following example demonstrates a C program instruction for setting the COG1 to the Push-Pull mode:

```
COG1CON0bits.MD = 0x5;
```

Individual bits in a bit field can also be accessed with long and short bit names. Each bit is the field name appended with the number of the bit position within the field. For example, the Most Significant mode bit has the short bit name MD2 and the long bit name is G1MD2. The following two examples demonstrate assembly program sequences for setting the COG1 to Push-Pull mode.

#### EXAMPLE 1-1: ASSEMBLY SEQUENCE 1 FOR SETTING COG1 TO PUSH-PULL MODE

```
MOVLW  ~(1<<G1MD1)
ANDWF  COG1CON0,F
MOVLW  1<<G1MD2 | 1<<G1MD0
IORWF  COG1CON0,F
```

#### EXAMPLE 1-2: ASSEMBLY SEQUENCE 2 FOR SETTING COG1 TO PUSH-PULL MODE

```
BSF    COG1CON0,G1MD2
BCF    COG1CON0,G1MD1
BSF    COG1CON0,G1MD0
```

## 1.1.3 REGISTER AND BIT NAMING EXCEPTIONS

### 1.1.3.1 Status, Interrupt, and Mirror Bits

Status, interrupt enables, interrupt flags, and mirror bits are contained in registers that span more than one peripheral. In these cases, the bit name shown is unique so there is no prefix or short name variant.

### 1.1.3.2 Legacy Peripherals

There are some peripherals that do not strictly adhere to these naming conventions. Peripherals that have existed for many years and are present in almost every device are the exceptions. These exceptions were necessary to limit the adverse impact of the new conventions on legacy code. Peripherals that do adhere to the new convention will include a table in the registers section indicating the long name prefix for each peripheral instance. Peripherals that fall into the exception category will not have this table. These peripherals include, but are not limited to, the following:

- EUSART
- MSSP

## 10.0 INTERRUPTS

The interrupt feature allows certain events to preempt normal program flow. Firmware is used to determine the source of the interrupt and act accordingly. Some interrupts can be configured to wake the MCU from Sleep mode.

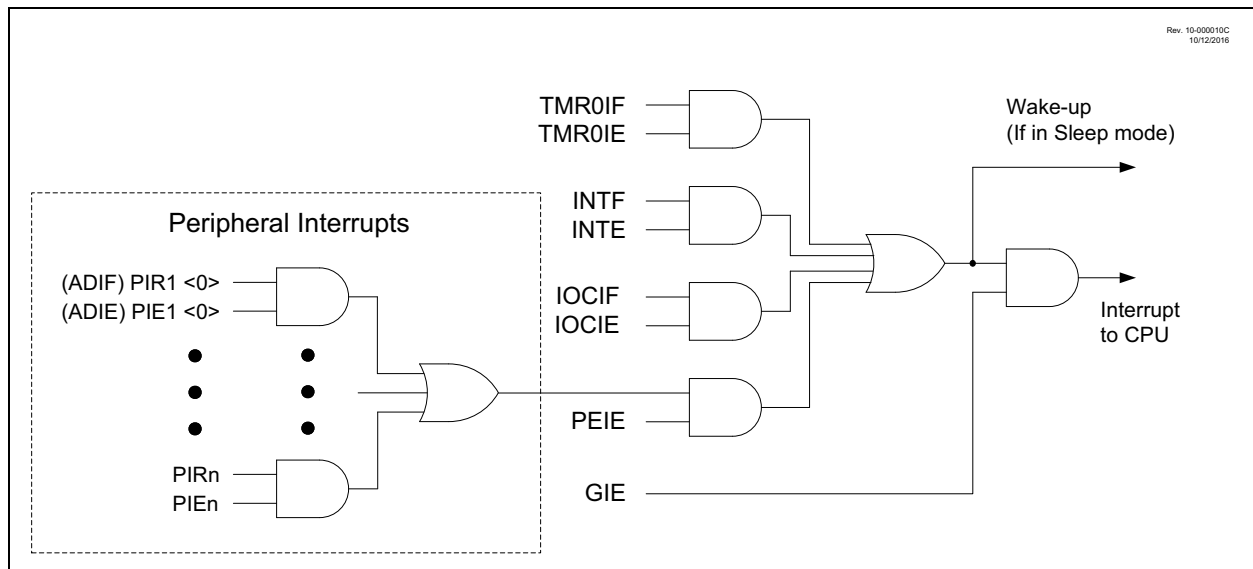
This chapter contains the following information for Interrupts:

- Operation
- Interrupt Latency
- Interrupts During Sleep
- INT Pin
- Automatic Context Saving

Many peripherals produce interrupts. Refer to the corresponding chapters for details.

A block diagram of the interrupt logic is shown in Figure 10-1.

**FIGURE 10-1: INTERRUPT LOGIC**



## 14.2 PORTA Registers

### 14.2.1 DATA REGISTER

PORTA is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISA (Register 14-2). Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., disable the output driver). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., enables output driver and puts the contents of the output latch on the selected pin). Example 14.2.8 shows how to initialize PORTA.

Reading the PORTA register (Register 14-1) reads the status of the pins, whereas writing to it will write to the PORT latch.

The PORT data latch LATA (Register 14-3) holds the output port data, and contains the latest value of a LATA or PORTA write.

#### EXAMPLE 14-1: INITIALIZING PORTA

```
; This code example illustrates
; initializing the PORTA register. The
; other ports are initialized in the same
; manner.

BANKSEL PORTA      ;
CLRF PORTA         ;Init PORTA
BANKSEL LATA       ;Data Latch
CLRF LATA          ;
BANKSEL ANSELA     ;
CLRF ANSELA        ;digital I/O
BANKSEL TRISA      ;
MOVLW B'00111000' ;Set RA<5:3> as inputs
MOVWF TRISA        ;and set RA<2:0> as
                  ;outputs
```

### 14.2.2 DIRECTION CONTROL

The TRISA register (Register 14-2) controls the PORTA pin output drivers, even when they are being used as analog inputs. The user should ensure the bits in the TRISA register are maintained set when using them as analog inputs. I/O pins configured as analog inputs always read '0'.

### 14.2.3 OPEN-DRAIN CONTROL

The ODCONA register (Register 14-6) controls the open-drain feature of the port. Open-drain operation is independently selected for each pin. When an ODCONA bit is set, the corresponding port output becomes an open-drain driver capable of sinking current only. When an ODCONA bit is cleared, the corresponding port output pin is the standard push-pull drive capable of sourcing and sinking current.

**Note:** It is not necessary to set open-drain control when using the pin for I<sup>2</sup>C; the I<sup>2</sup>C module controls the pin and makes the pin open-drain.

### 14.2.4 SLEW RATE CONTROL

The SLRCONA register (Register 14-7) controls the slew rate option for each port pin. Slew rate control is independently selectable for each port pin. When an SLRCONA bit is set, the corresponding port pin drive is slew rate limited. When an SLRCONA bit is cleared, The corresponding port pin drive slews at the maximum rate possible.

# PIC16(L)F19155/56/75/76/85/86

## REGISTER 14-39: SLRCONE: PORTE SLEW RATE CONTROL REGISTER

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	U-0	R/W-1/1	R/W-1/1
SLRE7	SLRE6	SLRE5	SLRE4	SLRE3	—	SLRE1	SLRE0
bit 7							bit 0

### Legend:

R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-3      **SLRE<7:3>**: PORTE Slew Rate Enable bits  
 For RE<7:0> pins, respectively  
 1 = Port pin slew rate is limited  
 0 = Port pin slews at maximum rate

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

bit 1-0      **SLRE<1:0>**: PORTE Slew Rate Enable bits  
 For RE<1:0> pins, respectively  
 1 = Port pin slew rate is limited  
 0 = Port pin slews at maximum rate

## REGISTER 14-40: INLVLE: PORTE INPUT LEVEL CONTROL REGISTER

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	U-0	R/W-1/1	R/W-1/1
INLVLE7	INLVLE6	INLVLE5	INLVLE4	INLVLE3	—	INLVLE1	INLVLE0
bit 7							bit 0

### Legend:

R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-3      **INLVLE<7:3>**: PORTE Input Level Select bits  
 For RE<7:3> pins, respectively  
 1 = ST input used for PORT reads and interrupt-on-change  
 0 = TTL input used for PORT reads and interrupt-on-change

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

bit 1-0      **INLVLE<1:0>**: PORTE Input Level Select bits  
 For RE<1:0> pins, respectively  
 1 = ST input used for PORT reads and interrupt-on-change  
 0 = TTL input used for PORT reads and interrupt-on-change

# PIC16(L)F19155/56/75/76/85/86

**TABLE 14-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTE**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
PORTE	RE7	RE6	RE5	RE4	RE3	—	RE1	RE0	248
TRISE	TRISE7	TRISE6	TRISE5	TRISE4	TRISE3	—	TRISE1	TRISE0	248
LATE	LATE7	LATE6	LATE5	LATE4	LATE3	—	LATE1	LATE0	249
ANSELE	ANSE7	ANSE6	ANSE5	ANSE4	ANSE3	—	ANSE1	ANSE0	249
WPUE	WPUE7	WPUE6	WPUE5	WPUE4	WPUE3	—	WPUE1	WPUE0	250
ODCONE	ODCE7	ODCE6	ODCE5	ODCE4	ODCE3	—	ODCE1	ODCE0	250
SLRCONE	SLRE7	SLRE6	SLRE5	SLRE4	SLRE3	—	SLRE1	SLRE0	251
INLVLE	INLVLE7	INLVLE6	INLVLE5	INLVLE4	INLVLE3	—	INLVLE1	INLVLE0	251

**Legend:** x = unknown, u = unchanged, — = unimplemented locations read as '0'. Shaded cells are not used by PORTE.

**TABLE 19-2: COMPUTATION MODES**

Mode	ADMD	Bit Clear Conditions	Value after Trigger completion		Threshold Operations			Value at ADTIF Interrupt		
			ACC	CNT	Retrigger	Threshold Test	Interrupt	OV	FLTR	CNT
Basic	0	ADACLR = 1	Unchanged	Unchanged	No	Every Sample	If threshold=true	N/A	N/A	count
Accumulate	1	ADACLR = 1	S + ACC or (S2-S1) + ACC	If (CNT=0xFF): CNT, otherwise: CNT+1	No	Every Sample	If threshold=true	ACC Overflow	$ACC/2^{ADCRS}$	count
Average	2	ADACLR = 1 or CNT>=RPT at GO or retrigger	S + ACC or (S2-S1) + ACC	If (CNT=0xFF): CNT, otherwise: CNT+1	No	If CNT>=RPT	If threshold=true	ACC Overflow	$ACC/2^{ADCRS}$	count
Burst Average	3	ADACLR = 1 or GO set or retrigger	Each repetition: same as Average End with sum of all samples	Each repetition: same as Average End with CNT=RPT	Repeat while CNT<RPT	If CNT>=RPT	If threshold=true	ACC Overflow	$ACC/2^{ADCRS}$	RPT
Low-pass Filter	4	ADACLR = 1	$S+ACC-ACC/2^{ADCRS}$ or $(S2-S1)+ACC-ACC/2^{ADCRS}$	Count up, stop counting when CNT = 0xFF	No	If CNT>=RPT	If threshold=true	ACC Overflow	Filtered Value	count

**Note:** S1 and S2 are abbreviations for Sample 1 and Sample 2, respectively. When ADDSEN = 0, S1 = ADRES; When ADDSEN = 1, S1 = PREV and S2 = ADRES.

# PIC16(L)F19155/56/75/76/85/86

## 19.6 Register Definitions: ADC Control

**REGISTER 19-1: ADCON0: ADC CONTROL REGISTER 0**

R/W-0/0	R/W-0/0	U-0	R/W-0/0	U-0	R/W-0/0	U-0	R/W/HC-0
ON	CONT	—	CS	—	FM	—	GO
bit 7							bit 0

**Legend:**

R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	HC = Bit is cleared by hardware

- bit 7      **ON:** ADC Enable bit  
1 = ADC is enabled  
0 = ADC is disabled
- bit 6      **CONT:** ADC Continuous Operation Enable bit<sup>(2)</sup>  
1 = GO is retriggered upon completion of each conversion trigger until ADTIF is set (if ADSOI is set) or until GO is cleared (regardless of the value of ADSOI)  
0 = ADC is cleared upon completion of each conversion trigger
- bit 5      **Unimplemented:** Read as '0'
- bit 4      **CS:** ADC Clock Selection bit  
1 = Clock supplied from FRC dedicated oscillator  
0 = Clock supplied by FOSC, divided according to ADCLK register
- bit 3      **Unimplemented:** Read as '0'
- bit 2      **FM:** ADC results Format/alignment Selection  
1 = ADRES and PREV data are right-justified  
0 = ADRES and PREV data are left-justified, zero-filled
- bit 1      **Unimplemented:** Read as '0'
- bit 0      **GO:** ADC Conversion Status bit<sup>(1)</sup>  
1 = ADC conversion cycle in progress. Setting this bit starts an ADC conversion cycle. The bit is cleared by hardware as determined by the CONT bit  
0 = ADC conversion completed/not in progress

- Note 1:** This bit requires ON bit to be set.
- 2:** If cleared by software while a conversion is in progress, the results of the conversion up to this point will be transferred to ADRES and the state machine will be reset, but the ADIF interrupt flag bit will not be set; filter and threshold operations will not be performed.



## 22.9 Comparator Response Time

The comparator output is indeterminate for a period of time after the change of an input source or the selection of a new reference voltage. This period is referred to as the response time. The response time of the comparator differs from the settling time of the voltage reference. Therefore, both of these times must be considered when determining the total response time to a comparator input change. See the Comparator and Voltage Reference Specifications in Table 39-14 and Table 39-15 for more details.

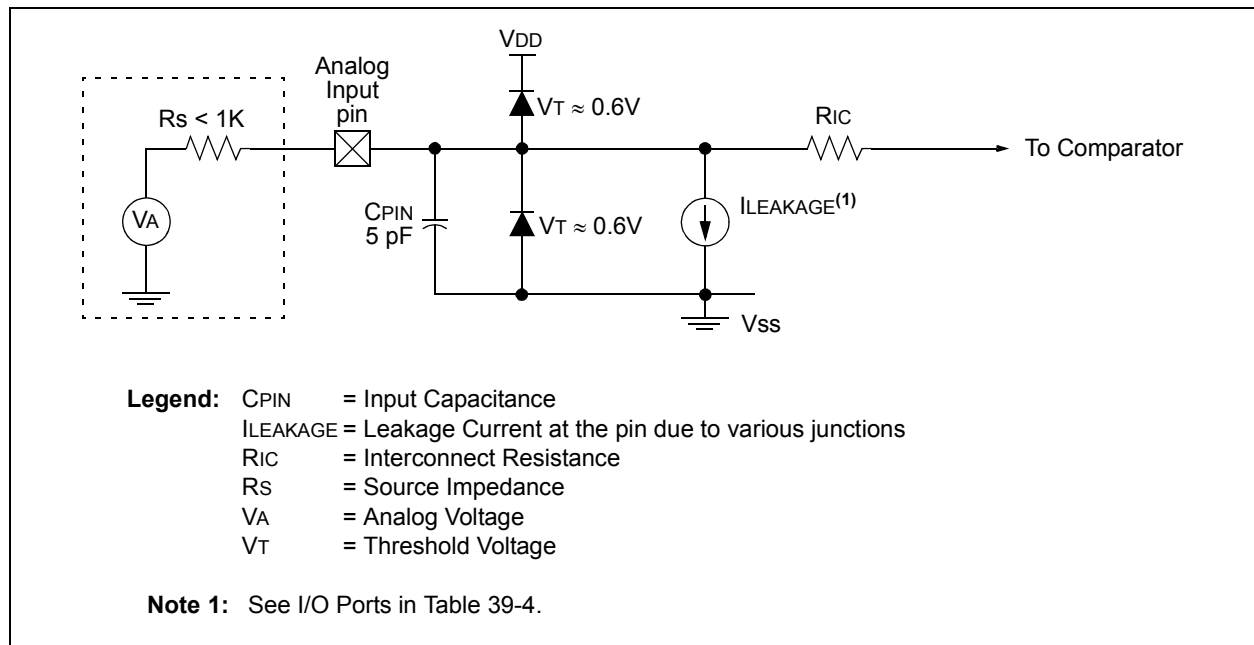
## 22.10 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 22-3. Since the analog input pins share their connection with a digital input, they have reverse biased ESD protection diodes to  $V_{DD}$  and  $V_{SS}$ . The analog input, therefore, must be between  $V_{SS}$  and  $V_{DD}$ . If the input voltage deviates from this range by more than 0.6V in either direction, one of the diodes is forward biased and a latch-up may occur.

A maximum source impedance of  $1k\Omega$  is recommended for the analog sources. Also, any external component connected to an analog input pin, such as a capacitor or a Zener diode, should have very little leakage current to minimize inaccuracies introduced.

- Note 1:** When reading a PORT register, all pins configured as analog inputs will read as a '0'. Pins configured as digital inputs will convert as an analog input, according to the input specification.
- 2:** Analog levels on any pin defined as a digital input, may cause the input buffer to consume more current than is specified.

**FIGURE 22-3: ANALOG INPUT MODEL**



**FIGURE 26-6: TIMER1 GATE SINGLE-PULSE AND TOGGLE COMBINED MODE**



## 26.11 Peripheral Module Disable

When a peripheral module is not used or inactive, the module can be disabled by setting the Module Disable bit in the PMD registers. This will reduce power consumption to an absolute minimum. Setting the PMD bits holds the module in Reset and disconnects the module's clock source. The Module Disable bit for Timer1 (TMR1MD) are in the PMD1 register. See **Section 16.0 "Peripheral Module Disable (PMD)"** for more information.

## 27.5 Operation Examples

Unless otherwise specified, the following notes apply to the following timing diagrams:

- Both the prescaler and postscaler are set to 1:1 (both the CKPS and OUTPS bits in the TxCON register are cleared).
- The diagrams illustrate any clock except  $F_{osc}/4$  and show clock-sync delays of at least two full cycles for both ON and Timer2/4\_ers. When using  $F_{osc}/4$ , the clock-sync delay is at least one instruction period for Timer2/4\_ers; ON applies in the next instruction period.
- The PWM Duty Cycle and PWM output are illustrated assuming that the timer is used for the PWM function of the CCP module as described in **Section 29.0 “Capture/Compare/PWM Modules”**. The signals are not a part of the Timer2/4 module.

### 27.5.1 SOFTWARE GATE MODE

This mode corresponds to legacy Timer2/4 operation. The timer increments with each clock input when  $ON = 1$  and does not increment when  $ON = 0$ . When the TMRx count equals the PRx period count the timer resets on the next clock and continues counting from 0. Operation with the ON bit software controlled is illustrated in Figure 27-4. With  $PRx = 5$ , the counter advances until  $TMRx = 5$ , and goes to zero with the next clock.

## 27.5.9 EDGE-TRIGGERED MONOSTABLE MODES

The Edge-Triggered Monostable modes start the timer on an edge from the external Reset signal input, after the ON bit is set, and stop incrementing the timer when the timer matches the PRx period value. The following edges will start the timer:

- Rising edge (MODE<4:0> = 10001)
- Falling edge (MODE<4:0> = 10010)
- Rising or Falling edge (MODE<4:0> = 10011)

When an Edge-Triggered Monostable mode is used in conjunction with the CCP PWM operation the PWM drive goes active with the external Reset signal edge that starts the timer, but will not go active when the timer matches the PRx value. While the timer is incrementing, additional edges on the external Reset signal will not affect the CCP PWM.

## 28.2 SMT Operation

The core of the module is the 24-bit counter, SMTxTMR combined with a complex data acquisition front-end. Depending on the mode of operation selected, the SMT can perform a variety of measurements summarized in Table .

### 28.2.1 CLOCK SOURCES

Clock sources available to the SMT include:

- FOSC
- FOSC/4
- HFINTOSC
- MFINTOSC (500 kHz and 31.25 kHz)
- LFINTOSC
- SOSC

The SMT clock source is selected by configuring the CSEL<2:0> bits in the SMTxCLK register. The clock source can also be prescaled using the PS<1:0> bits of the SMTxCON0 register. The prescaled clock source is used to clock both the counter and any synchronization logic used by the module.

### 28.2.2 PERIOD MATCH INTERRUPT

Similar to other timers, the SMT triggers an interrupt when SMTxTMR rolls over to '0'. This happens when SMTxTMR = SMTxPR, regardless of mode. Hence, in any mode that relies on an external signal or a window to reset the timer, proper operation requires that SMTxPR be set to a period larger than that of the expected signal or window.

## 28.3 Basic Timer Function Registers

The SMTxTMR time base and the SMTxCPW/SMTxPR/SMTxCPR buffer registers serve several functions and can be manually updated using software.

### 28.3.1 TIME BASE

The SMTxTMR is the 24-bit counter that is the center of the SMT. It is used as the basic counter/timer for measurement in each of the modes of the SMT. It can be reset to a value of 24'h00\_0000 by setting the RST bit of the SMTxSTAT register. It can be written to and read from software, but it is not guarded for atomic access, therefore reads and writes to the SMTxTMR should only be made when the GO = 0, or the software should have other measures to ensure integrity of SMTxTMR reads/writes.

### 28.3.2 PULSE-WIDTH LATCH REGISTERS

The SMTxCPW registers are the 24-bit SMT pulse width latch. They are used to latch in the value of the SMTxTMR when triggered by various signals, which are determined by the mode the SMT is currently in.

The SMTxCPW registers can also be updated with the current value of the SMTxTMR value by setting the CPWUP bit of the SMTxSTAT register.

### 28.3.3 PERIOD LATCH REGISTERS

The SMTxCPR registers are the 24-bit SMT period latch. They are used to latch in values of the SMTxTMR when triggered by various other signals, which are determined by the mode the SMT is currently in.

The SMTxCPR registers can also be updated with the current value of the SMTxTMR value by setting the CPRUP bit in the SMTxSTAT register.

## 28.4 Halt Operation

The counter can be prevented from rolling-over using the STP bit in the SMTxCON0 register. When halting is enabled, the period match interrupt persists until the SMTxTMR is reset (either by a manual Reset, **Section 28.3.1 "Time Base"**) or by clearing the SMTxGO bit of the SMTxCON1 register and writing the SMTxTMR values in software.

## 28.5 Polarity Control

The three input signals for the SMT have polarity control to determine whether or not they are active high/positive edge or active low/negative edge signals.

The following bits apply to Polarity Control:

- WSEL bit (Window Polarity)
- SSEL bit (Signal Polarity)
- CSEL bit (Clock Polarity)

These bits are located in the SMTxCON0 register.

## 28.6 Status Information

The SMT provides input status information for the user without requiring the need to deal with the polarity of the incoming signals.

### 28.6.1 WINDOW STATUS

Window status is determined by the WS bit of the SMTxSTAT register. This bit is only used in Windowed Measure, Gated Counter and Gated Window Measure modes, and is only valid when TS = 1, and will be delayed in time by synchronizer delays in non-Counter modes.

### 28.6.2 SIGNAL STATUS

Signal status is determined by the AS bit of the SMTxSTAT register. This bit is used in all modes except Window Measure, Time of Flight and Capture modes, and is only valid when TS = 1, and will be delayed in time by synchronizer delays in non-Counter modes.

## 32.2 CLCx Interrupts

An interrupt will be generated upon a change in the output value of the CLCx when the appropriate interrupt enables are set. A rising edge detector and a falling edge detector are present in each CLC for this purpose.

The CLCxIF bit of the associated PIR5 register will be set when either edge detector is triggered and its associated enable bit is set. The LCxINTP enables rising edge interrupts and the LCxINTN bit enables falling edge interrupts. Both are located in the CLCxCON register.

To fully enable the interrupt, set the following bits:

- CLCxIE bit of the PIE5 register
- LCxINTP bit of the CLCxCON register (for a rising edge detection)
- LCxINTN bit of the CLCxCON register (for a falling edge detection)
- PEIE and GIE bits of the INTCON register

The CLCxIF bit of the PIR5 register, must be cleared in software as part of the interrupt service. If another edge is detected while this flag is being cleared, the flag will still be set at the end of the sequence.

## 32.3 Output Mirror Copies

Mirror copies of all LCxCON output bits are contained in the CLCxDATA register. Reading this register reads the outputs of all CLCs simultaneously. This prevents any reading skew introduced by testing or reading the LCxOUT bits in the individual CLCxCON registers.

## 32.4 Effects of a Reset

The CLCxCON register is cleared to zero as the result of a Reset. All other selection and gating values remain unchanged.

## 32.5 Operation During Sleep

The CLC module operates independently from the system clock and will continue to run during Sleep, provided that the input sources selected remain active.

The HFINTOSC remains active during Sleep when the CLC module is enabled and the HFINTOSC is selected as an input source, regardless of the system clock source selected.

In other words, if the HFINTOSC is simultaneously selected as the system clock and as a CLC input source, when the CLC is enabled, the CPU will go idle during Sleep, but the CLC will continue to operate and the HFINTOSC will remain active.

This will have a direct effect on the Sleep mode current.

## 32.6 CLCx Setup Steps

The following steps should be followed when setting up the CLCx:

- Disable CLCx by clearing the LCxEN bit.
- Select desired inputs using CLCxSEL0 through CLCxSEL3 registers (See Table 32-2).
- Clear any associated ANSEL bits.
- Enable the chosen inputs through the four gates using CLCxGLS0, CLCxGLS1, CLCxGLS2, and CLCxGLS3 registers.
- Select the gate output polarities with the LCxGyPOL bits of the CLCxPOL register.
- Select the desired logic function with the LCxMODE<2:0> bits of the CLCxCON register.
- Select the desired polarity of the logic output with the LCxPOL bit of the CLCxPOL register. (This step may be combined with the previous gate output polarity step).
- If driving a device pin, set the desired pin PPS control register and also clear the TRIS bit corresponding to that output.
- If interrupts are desired, configure the following bits:
  - Set the LCxINTP bit in the CLCxCON register for rising event.
  - Set the LCxINTN bit in the CLCxCON register for falling event.
  - Set the CLCxIE bit of the PIE5 register.
  - Set the GIE and PEIE bits of the INTCON register.
- Enable the CLCx by setting the LCxEN bit of the CLCxCON register.

## 33.2.4 SPI SLAVE MODE

In Slave mode, the data is transmitted and received as external clock pulses appear on SCK. When the last bit is latched, the SSPxIF interrupt flag bit is set.

Before enabling the module in SPI Slave mode, the clock line must match the proper Idle state. The clock line can be observed by reading the SCK pin. The Idle state is determined by the CKP bit of the SSPxCON1 register.

While in Slave mode, the external clock is supplied by the external clock source on the SCK pin. This external clock must meet the minimum high and low times as specified in the electrical specifications.

While in Sleep mode, the slave can transmit/receive data. The shift register is clocked from the SCK pin input and when a byte is received, the device will generate an interrupt. If enabled, the device will wake-up from Sleep.

### 33.2.4.1 Daisy-Chain Configuration

The SPI bus can sometimes be connected in a daisy-chain configuration. The first slave output is connected to the second slave input, the second slave output is connected to the third slave input, and so on. The final slave output is connected to the master input. Each slave sends out, during a second group of clock pulses, an exact copy of what was received during the first group of clock pulses. The whole chain acts as one large communication shift register. The daisy-chain feature only requires a single Slave Select line from the master device.

Figure 33-7 shows the block diagram of a typical daisy-chain connection when operating in SPI mode.

In a daisy-chain configuration, only the most recent byte on the bus is required by the slave. Setting the BOEN bit of the SSPxCON3 register will enable writes to the SSPxBUF register, even if the previous byte has not been read. This allows the software to ignore data that may not apply to it.

## 33.2.5 SLAVE SELECT SYNCHRONIZATION

The Slave Select can also be used to synchronize communication. The Slave Select line is held high until the master device is ready to communicate. When the Slave Select line is pulled low, the slave knows that a new transmission is starting.

If the slave fails to receive the communication properly, it will be reset at the end of the transmission, when the Slave Select line returns to a high state. The slave is then ready to receive a new transmission when the Slave Select line is pulled low again. If the Slave Select line is not used, there is a risk that the slave will eventually become out of sync with the master. If the slave misses a bit, it will always be one bit off in future transmissions. Use of the Slave Select line allows the slave and master to align themselves at the beginning of each transmission.

The  $\overline{SS}$  pin allows a Synchronous Slave mode. The SPI must be in Slave mode with  $\overline{SS}$  pin control enabled (SSPxCON1<3:0> = 0100).

When the  $\overline{SS}$  pin is low, transmission and reception are enabled and the SDO pin is driven.

When the  $\overline{SS}$  pin goes high, the SDO pin is no longer driven, even if in the middle of a transmitted byte and becomes a floating output. External pull-up/pull-down resistors may be desirable depending on the application.

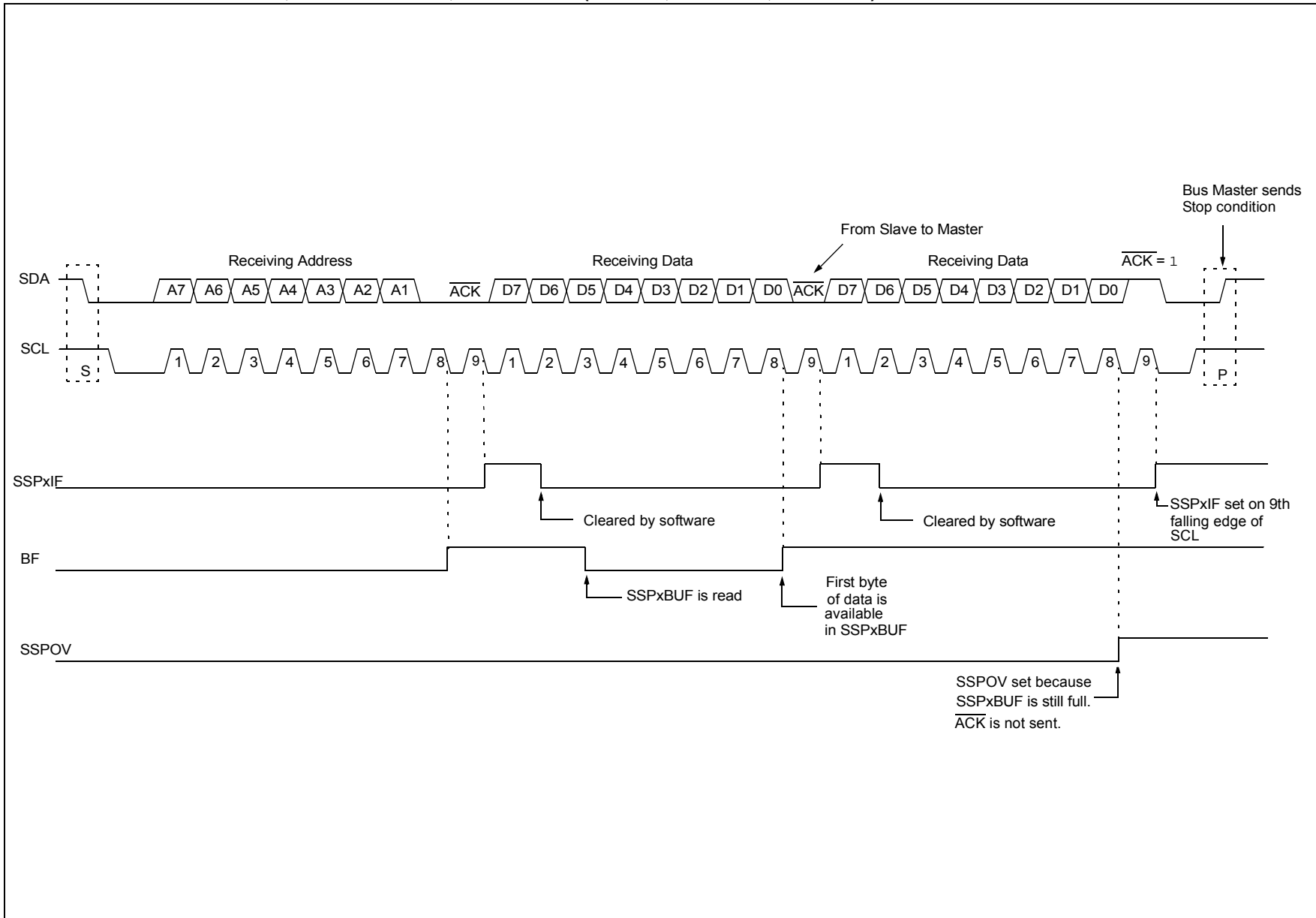
- |   |
|---|
| <p><b>Note 1:</b> When the SPI is in Slave mode with <math>\overline{SS}</math> pin control enabled (SSPxCON1&lt;3:0&gt; = 0100), the SPI module will reset if the <math>\overline{SS}</math> pin is set to VDD.</p> <p><b>2:</b> When the SPI is used in Slave mode with CKE set; the user must enable <math>\overline{SS}</math> pin control.</p> <p><b>3:</b> While operated in SPI Slave mode the SMP bit of the SSPxSTAT register must remain clear.</p> |
|---|

When the SPI module resets, the bit counter is forced to '0'. This can be done by either forcing the  $\overline{SS}$  pin to a high level or clearing the SSPEN bit.

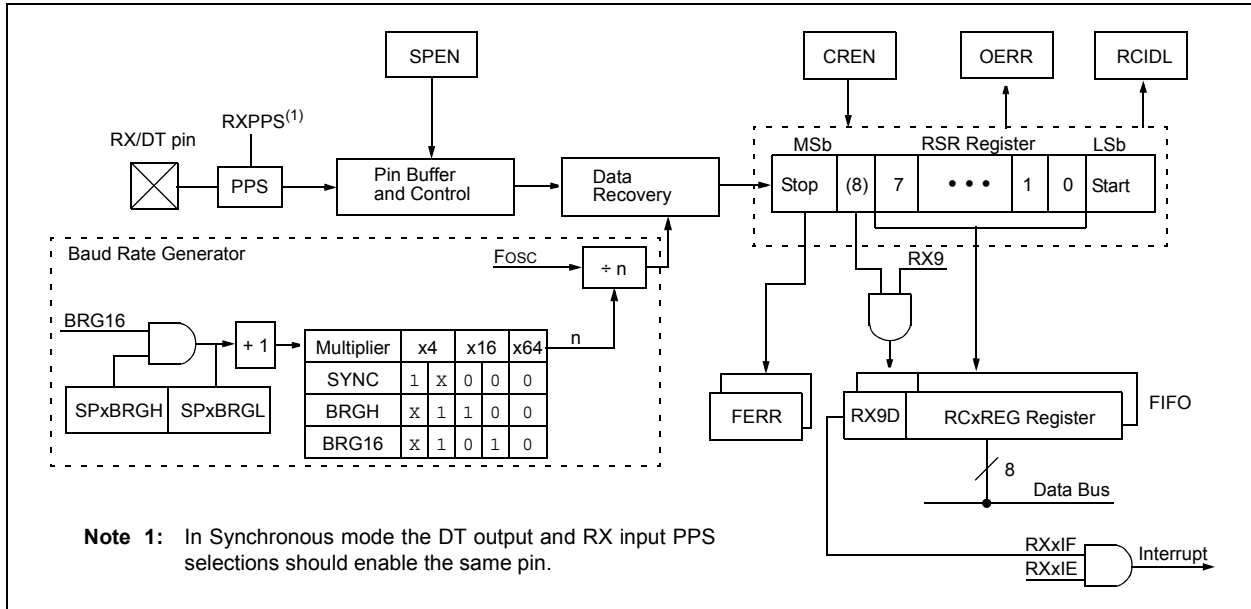




FIGURE 33-14: I<sup>2</sup>C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 0, DHEN = 0)



**FIGURE 34-2: EUSART RECEIVE BLOCK DIAGRAM**



The operation of the EUSART module is controlled through three registers:

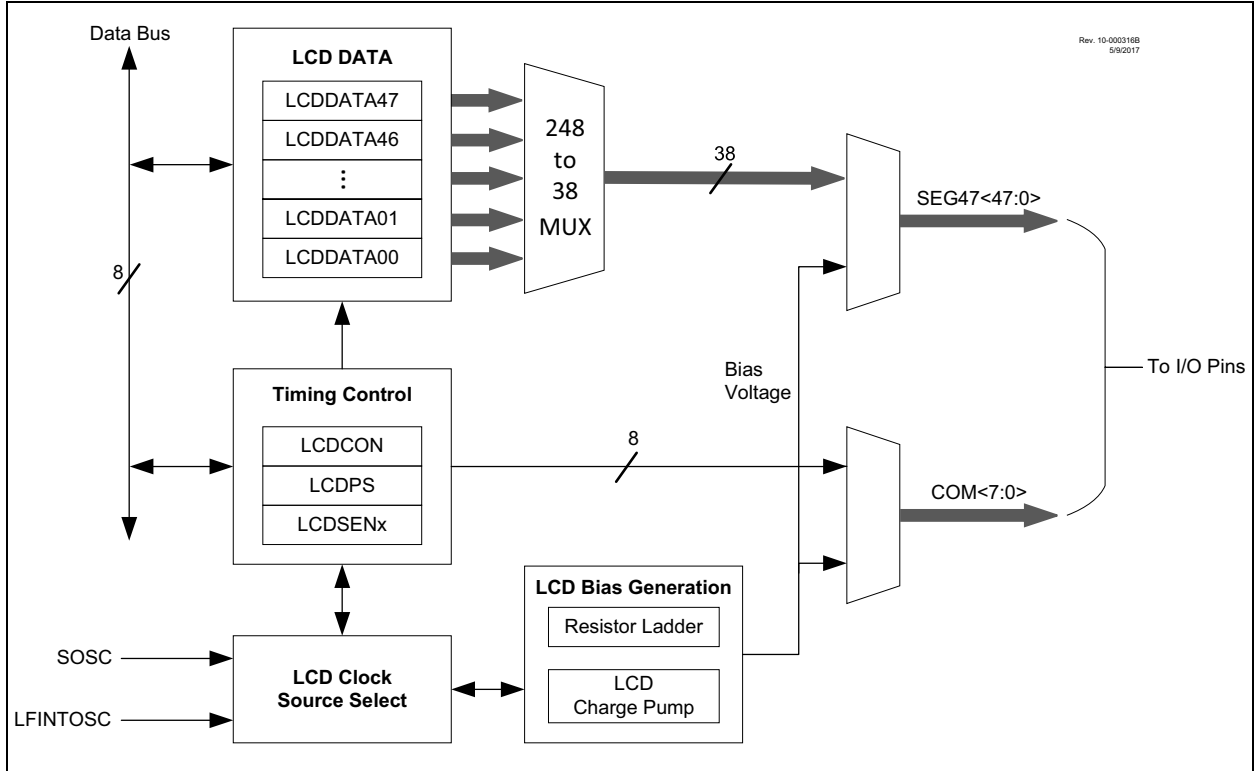
- Transmit Status and Control (TXxSTA)
- Receive Status and Control (RCxSTA)
- Baud Rate Control (BAUDxCON)

These registers are detailed in Register 34-1, Register 34-2 and Register 34-3, respectively.

The RX input pin is selected with the RXPPS. The CK input is selected with the TXPPS register. TX, CK, and DT output pins are selected with each pin's RxyPPS register. Since the RX input is coupled with the DT output in Synchronous mode, it is the user's responsibility to select the same pin for both of these functions when operating in Synchronous mode. The EUSART control logic will control the data direction drivers automatically.

# PIC16(L)F19155/56/75/76/85/86

FIGURE 35-1: LCD CONTROLLER MODULE BLOCK DIAGRAM



## 35.13 LCD Interrupts

The LCD timing generation provides an interrupt that defines the LCD frame timing. This interrupt can be used to coordinate the writing of the pixel data with the start of a new frame, which produces a visually crisp transition of the image.

This interrupt can also be used to synchronize external events to the LCD. For example, the interface to an external segment driver can be synchronized for segment data updates to the LCD frame.

A new frame is defined as beginning at the leading edge of the COM0 common signal. The interrupt will be set immediately after the LCD controller completes accessing all pixel data required for a frame. This will occur at a fixed interval before the frame boundary (TFINT), as shown in Figure 35-20.

The LCD controller will begin to access the next frame between the interrupt and when the controller accesses the data (TFWR). New data must be written within TFWR, as this is when the LCD controller will begin to access the data for the next frame.

When the LCD driver is running with Type-B waveforms, and the LMUX<3:0> bits are not equal to '0001', the following issues may arise.

Since the DC voltage on the pixel takes two frames to maintain 0V, the pixel data must not change between subsequent frames. If the pixel data were allowed to change, the waveform for the odd frames would not necessarily be the complement of the waveform generated in the even frames and a DC component would be introduced into the panel.

Because of this, using Type-B waveforms requires synchronizing the LCD pixel updates to occur within a sub-frame after the frame interrupt.

To correctly sequence writing in Type-B, the interrupt only occurs on complete phase intervals. If the user attempts to write when the write is disabled, the WERR bit (LCDCON<5>) is set.

**Note:** The interrupt is not generated when the Type-A waveform is selected and when the Type-B with no multiplex (static) is selected.

**FIGURE 35-20: EXAMPLE WAVEFORMS AND INTERRUPT TIMING IN QUARTER DUTY CYCLE DRIVE**



## 41.6 MPLAB X SIM Software Simulator

The MPLAB X SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB X SIM Software Simulator fully supports symbolic debugging using the MPLAB XC Compilers, and the MPASM and MPLAB Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

## 41.7 MPLAB REAL ICE In-Circuit Emulator System

The MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs all 8, 16 and 32-bit MCU, and DSC devices with the easy-to-use, powerful graphical user interface of the MPLAB X IDE.

The emulator is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with in-circuit debugger systems (RJ-11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

The emulator is field upgradeable through future firmware downloads in MPLAB X IDE. MPLAB REAL ICE offers significant advantages over competitive emulators including full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, logic probes, a ruggedized probe interface and long (up to three meters) interconnection cables.

## 41.8 MPLAB ICD 3 In-Circuit Debugger System

The MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost-effective, high-speed hardware debugger/programmer for Microchip Flash DSC and MCU devices. It debugs and programs PIC Flash microcontrollers and dsPIC DSCs with the powerful, yet easy-to-use graphical user interface of the MPLAB IDE.

The MPLAB ICD 3 In-Circuit Debugger probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with a connector compatible with the MPLAB ICD 2 or MPLAB REAL ICE systems (RJ-11). MPLAB ICD 3 supports all MPLAB ICD 2 headers.

## 41.9 PICkit 3 In-Circuit Debugger/Programmer

The MPLAB PICkit 3 allows debugging and programming of PIC and dsPIC Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB IDE. The MPLAB PICkit 3 is connected to the design engineer's PC using a full-speed USB interface and can be connected to the target via a Microchip debug (RJ-11) connector (compatible with MPLAB ICD 3 and MPLAB REAL ICE). The connector uses two device I/O pins and the Reset line to implement in-circuit debugging and In-Circuit Serial Programming™ (ICSP™).

## 41.10 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at VDDMIN and VDDMAX for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages, and a modular, detachable socket assembly to support various package types. The ICSP cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices, and incorporates an MMC card for file storage and data applications.