



Welcome to [E-XFL.COM](https://www.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	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	4MHz
Connectivity	-
Peripherals	Brown-out Detect/Reset, POR, WDT
Number of I/O	13
Program Memory Size	3.5KB (2K x 14)
Program Memory Type	OTP
EEPROM Size	128 x 8
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 5.5V
Data Converters	-
Oscillator Type	External
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	18-SOIC (0.295", 7.50mm Width)
Supplier Device Package	18-SOIC
Purchase URL	<a href="https://www.e-xfl.com/product-detail/microchip-technology/pic16lce625t-04-so">https://www.e-xfl.com/product-detail/microchip-technology/pic16lce625t-04-so</a>

# PIC16CE62X

## 4.2.2.4 PIE1 REGISTER

This register contains the individual enable bit for the comparator interrupt.

### REGISTER 4-4: PIE1 REGISTER (ADDRESS 8CH)

U-0	R/W-0	U-0	U-0	U-0	U-0	U-0	U-0
—	CMIE	—	—	—	—	—	—
bit7							bit0

R = Readable bit  
W = Writable bit  
U = Unimplemented bit, read as '0'  
-n = Value at POR reset  
-x = Unknown at POR reset

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

bit 6: **CMIE:** Comparator Interrupt Enable bit  
1 = Enables the Comparator interrupt  
0 = Disables the Comparator interrupt

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

## 4.2.2.5 PIR1 REGISTER

This register contains the individual flag bit for the comparator interrupt.

**Note:** Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

### REGISTER 4-5: PIR1 REGISTER (ADDRESS 0CH)

U-0	R/W-0	U-0	U-0	U-0	U-0	U-0	U-0
—	CMIF	—	—	—	—	—	—
bit7							bit0

R = Readable bit  
W = Writable bit  
U = Unimplemented bit, read as '0'  
-n = Value at POR reset  
-x = Unknown at POR reset

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

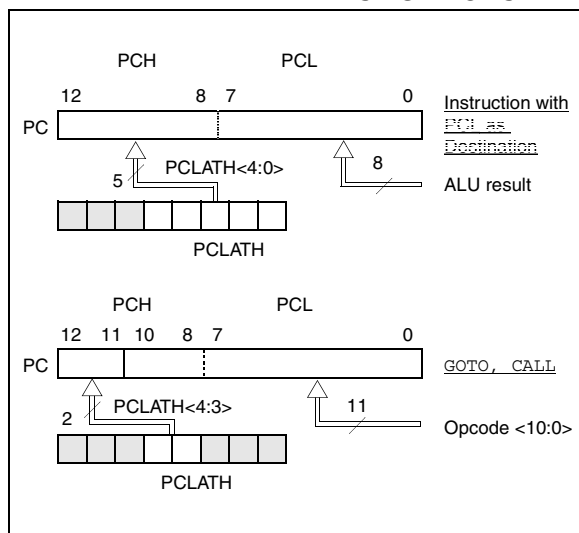
bit 6: **CMIF:** Comparator Interrupt Flag bit  
1 = Comparator input has changed  
0 = Comparator input has not changed

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

## 4.3 PCL and PCLATH

The program counter (PC) is 13 bits wide. The low byte comes from the PCL register, which is a readable and writable register. The high byte (PC<12:8>) is not directly readable or writable and comes from PCLATH. On any reset, the PC is cleared. Figure 4-6 shows the two situations for the loading of the PC. The upper example in the figure shows how the PC is loaded on a write to PCL (PCLATH<4:0> → PCH). The lower example in the figure shows how the PC is loaded during a CALL or GOTO instruction (PCLATH<4:3> → PCH).

**FIGURE 4-6: LOADING OF PC IN DIFFERENT SITUATIONS**



### 4.3.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL). When doing a table read using a computed GOTO method, care should be exercised if the table location crosses a PCL memory boundary (each 256 byte block). Refer to the application note, "Implementing a Table Read" (AN556).

### 4.3.2 STACK

The PIC16CE62X family has an 8 level deep x 13-bit wide hardware stack (Figure 4-2 and Figure 4-3). The stack space is not part of either program or data space and the stack pointer is not readable or writable. The PC is PUSHed onto the stack when a CALL instruction is executed or an interrupt causes a branch. The stack is POPed in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not affected by a PUSH or POP operation.

The stack operates as a circular buffer. This means that after the stack has been PUSHed eight times, the ninth push overwrites the value that was stored from the first push. The tenth push overwrites the second push (and so on).

**Note 1:** There are no STATUS bits to indicate stack overflow or stack underflow conditions.

**Note 2:** There are no instruction/mnemonics called PUSH or POP. These are actions that occur from the execution of the CALL, RETURN, RETLW and RETFIE instructions or the vectoring to an interrupt address.

**TABLE 5-3: PORTB FUNCTIONS**

Name	Bit #	Buffer Type	Function
RB0/INT	bit0	TTL/ST <sup>(1)</sup>	Input/output or external interrupt input. Internal software programmable weak pull-up.
RB1	bit1	TTL	Input/output pin. Internal software programmable weak pull-up.
RB2	bit2	TTL	Input/output pin. Internal software programmable weak pull-up.
RB3	bit3	TTL	Input/output pin. Internal software programmable weak pull-up.
RB4	bit4	TTL	Input/output pin (with interrupt on change). Internal software programmable weak pull-up.
RB5	bit5	TTL	Input/output pin (with interrupt on change). Internal software programmable weak pull-up.
RB6	bit6	TTL/ST <sup>(2)</sup>	Input/output pin (with interrupt on change). Internal software programmable weak pull-up. Serial programming clock pin.
RB7	bit7	TTL/ST <sup>(2)</sup>	Input/output pin (with interrupt on change). Internal software programmable weak pull-up. Serial programming data pin.

Legend: ST = Schmitt Trigger, TTL = TTL input

**Note 1:** This buffer is a Schmitt Trigger input when configured as the external interrupt.

**Note 2:** This buffer is a Schmitt Trigger input when used in serial programming mode.

**TABLE 5-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR	Value on All Other Resets
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111
81h	OPTION	RBP <sub>U</sub>	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: u = unchanged, x = unknown

**Note:** Shaded bits are not used by PORTB.

## 6.0 EEPROM PERIPHERAL OPERATION

The PIC16CE623/624/625 each have 128 bytes of EEPROM data memory. The EEPROM data memory supports a bi-directional, 2-wire bus and data transmission protocol. These two-wires are serial data (SDA) and serial clock (SCL), and are mapped to bit1 and bit2, respectively, of the EEINTF register (SFR 90h). In addition, the power to the EEPROM can be controlled using bit0 (EEVDD) of the EEINTF register. For most applications, all that is required is calls to the following functions:

```
; Byte_Write: Byte write routine
;   Inputs: EEPROM Address      EEADDR
;           EEPROM Data        EEDATA
;   Outputs: Return 01 in W if OK, else
;           return 00 in W
;
; Read_Current: Read EEPROM at address
;               currently held by EE device.
;   Inputs: NONE
;   Outputs: EEPROM Data        EEDATA
;           Return 01 in W if OK, else
;           return 00 in W
;
; Read_Random: Read EEPROM byte at supplied
; address
;   Inputs: EEPROM Address      EEADDR
;   Outputs: EEPROM Data        EEDATA
;           Return 01 in W if OK,
;           else return 00 in W
;
```

The code for these functions is available on our web site ([www.microchip.com](http://www.microchip.com)). The code will be accessed by either including the source code FL62XINC.ASM or by linking FLASH62X.ASM. FLASH62.IMC provides external definition to the calling program.

### 6.0.1 SERIAL DATA

SDA is a bi-directional pin used to transfer addresses and data into and data out of the memory.

For normal data transfer, SDA is allowed to change only during SCL low. Changes during SCL high are reserved for indicating the START and STOP conditions.

### 6.0.2 SERIAL CLOCK

This SCL input is used to synchronize the data transfer to and from the memory.

### 6.0.3 EEINTF REGISTER

The EEINTF register (SFR 90h) controls the access to the EEPROM. Register 6-1 details the function of each bit. User code must generate the clock and data signals.

**REGISTER 6-1: EEINTF REGISTER (ADDRESS 90h)**

U-0	U-0	U-0	U-0	U-0	R/W-1	R/W-1	R/W-1
—	—	—	—	—	EESCL	EESDA	EEVDD
bit7							bit0

R = Readable bit  
W = Writable bit  
U = Unimplemented bit, read as '0'  
- n = Value at POR reset

bit 7-3: **Unimplemented:** Read as '0'

bit 2: **EESCL:** Clock line to the EEPROM  
1 = Clock high  
0 = Clock low

bit 1: **EESDA:** Data line to EEPROM  
1 = Data line is high (pin is tri-stated, line is pulled high by a pull-up resistor)  
0 = Data line is low

bit 0: **EEVDD:** VDD control bit for EEPROM  
1 = VDD is turned on to EEPROM  
0 = VDD is turned off to EEPROM (all pins are tri-stated and the EEPROM is powered down)

**Note:** EESDA, EESCL and EEVDD will read '0' if EEVDD is turned off.

## 7.0 TIMER0 MODULE

The Timer0 module timer/counter has the following features:

- 8-bit timer/counter
- Readable and writable
- 8-bit software programmable prescaler
- Internal or external clock select
- Interrupt on overflow from FFh to 00h
- Edge select for external clock

Figure 7-1 is a simplified block diagram of the Timer0 module.

Timer mode is selected by clearing the T0CS bit (OPTION<5>). In timer mode, the TMR0 will increment every instruction cycle (without prescaler). If Timer0 is written, the increment is inhibited for the following two cycles (Figure 7-2 and Figure 7-3). The user can work around this by writing an adjusted value to TMR0.

Counter mode is selected by setting the T0CS bit. In this mode Timer0 will increment either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the source edge (T0SE) control

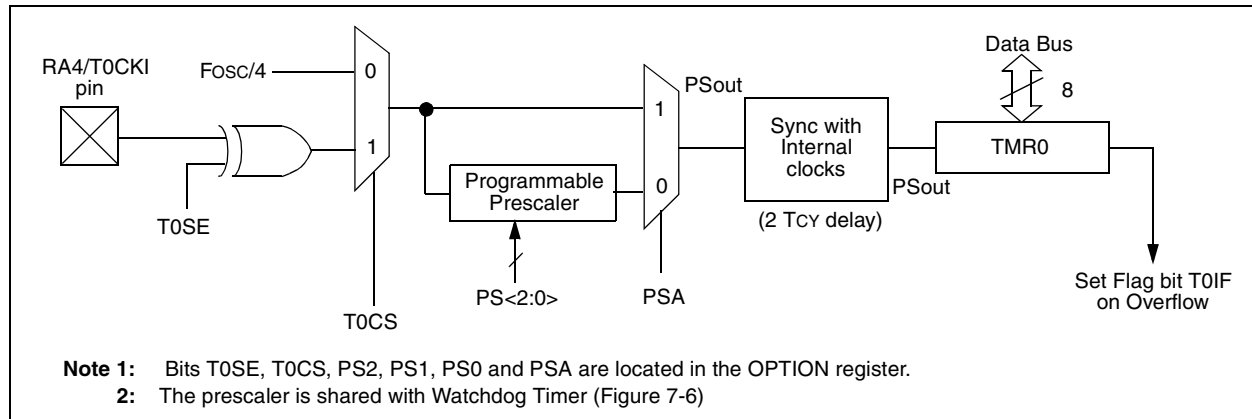
bit (OPTION<4>). Clearing the T0SE bit selects the rising edge. Restrictions on the external clock input are discussed in detail in Section 7.2.

The prescaler is shared between the Timer0 module and the Watchdog Timer. The prescaler assignment is controlled in software by the control bit PSA (OPTION<3>). Clearing the PSA bit will assign the prescaler to Timer0. The prescaler is not readable or writable. When the prescaler is assigned to the Timer0 module, prescale value of 1:2, 1:4, ..., 1:256 are selectable. Section 7.3 details the operation of the prescaler.

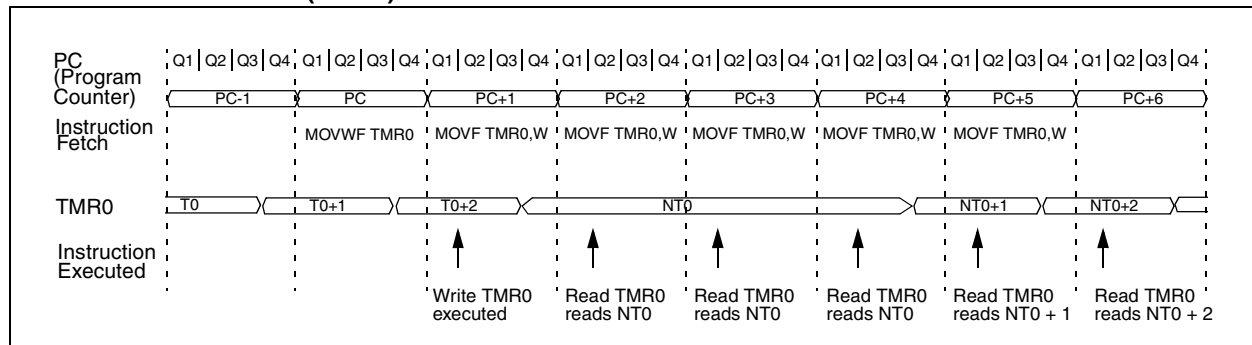
### 7.1 Timer0 Interrupt

Timer0 interrupt is generated when the TMR0 register timer/counter overflows from FFh to 00h. This overflow sets the T0IF bit. The interrupt can be masked by clearing the T0IE bit (INTCON<5>). The T0IF bit (INTCON<2>) must be cleared in software by the Timer0 module interrupt service routine before re-enabling this interrupt. The Timer0 interrupt cannot wake the processor from SLEEP since the timer is shut off during SLEEP. See Figure 7-4 for Timer0 interrupt timing.

**FIGURE 7-1: TIMER0 BLOCK DIAGRAM**



**FIGURE 7-2: TIMER0 (TMR0) TIMING: INTERNAL CLOCK/NO PRESCALER**



# PIC16CE62X

---

NOTES:

## 10.3 Reset

The PIC16CE62X differentiates between various kinds of reset:

- Power-on reset (POR)
- $\overline{\text{MCLR}}$  reset during normal operation
- $\overline{\text{MCLR}}$  reset during SLEEP
- WDT reset (normal operation)
- WDT wake-up (SLEEP)
- Brown-out Reset (BOD)

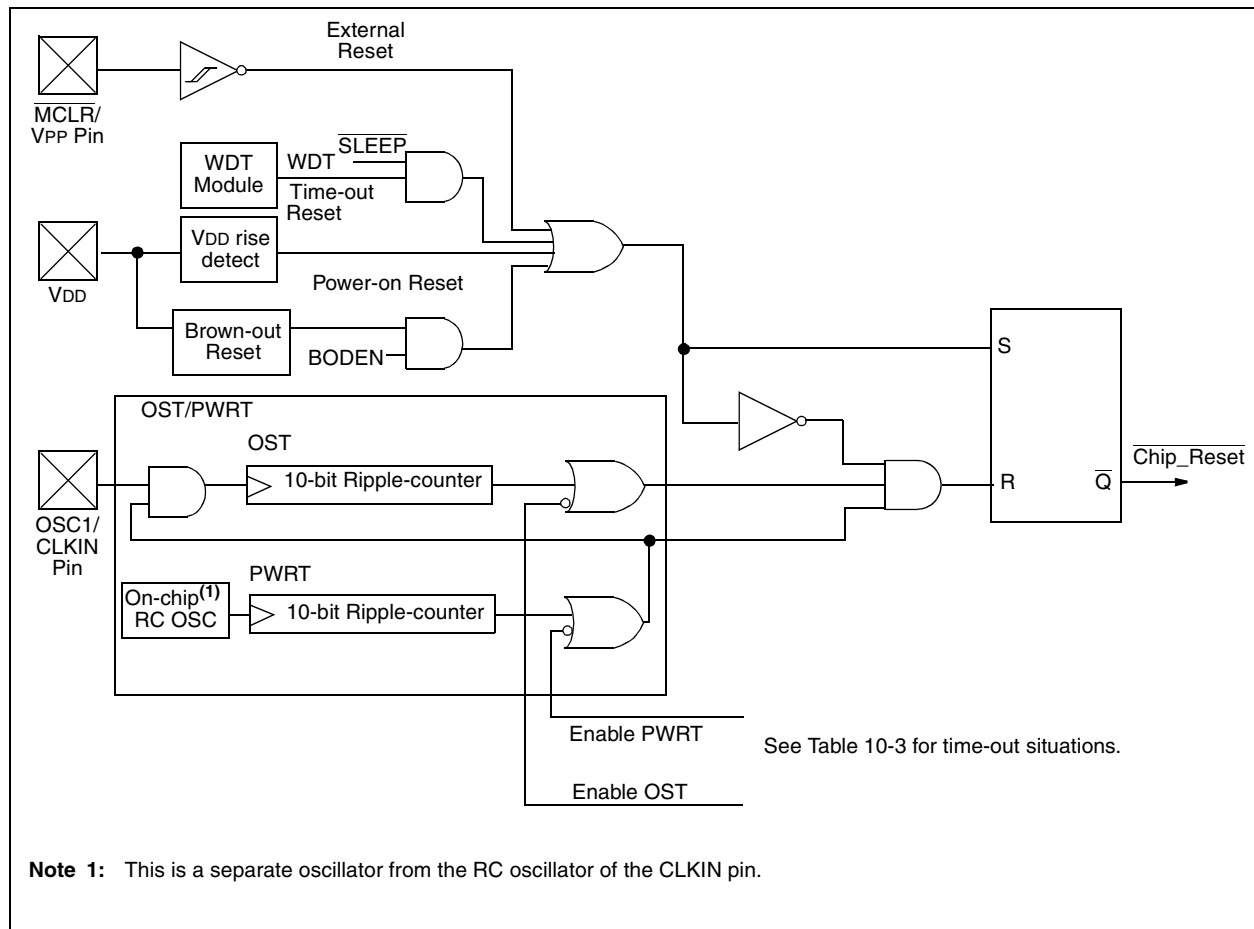
Some registers are not affected in any reset condition. Their status is unknown on POR and unchanged in any other reset. Most other registers are reset to a "reset

state" on Power-on reset,  $\overline{\text{MCLR}}$  reset, WDT reset and  $\overline{\text{MCLR}}$  reset during SLEEP. They are not affected by a WDT wake-up, since this is viewed as the resumption of normal operation.  $\overline{\text{TO}}$  and  $\overline{\text{PD}}$  bits are set or cleared differently in different reset situations as indicated in Table 10-4. These bits are used in software to determine the nature of the reset. See Table 10-6 for a full description of reset states of all registers.

A simplified block diagram of the on-chip reset circuit is shown in Figure 10-6.

The  $\overline{\text{MCLR}}$  reset path has a noise filter to detect and ignore small pulses. See Table 13-5 for pulse width specification.

**FIGURE 10-6: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT**





## 10.4 Power-on Reset (POR), Power-up Timer (PWRT), Oscillator Start-up Timer (OST) and Brown-out Reset (BOD)

### 10.4.1 POWER-ON RESET (POR)

The on-chip POR circuit holds the chip in reset until  $V_{DD}$  has reached a high enough level for proper operation. To take advantage of the POR, just tie the  $\overline{MCLR}$  pin through a resistor to  $V_{DD}$ . This will eliminate external RC components usually needed to create Power-on Reset. A maximum rise time for  $V_{DD}$  is required. See electrical specifications for details.

The POR circuit does not produce an internal reset when  $V_{DD}$  declines.

When the device starts normal operation (exits the reset condition), device operating parameters (voltage, frequency, temperature, etc.) must be met to ensure operation. If these conditions are not met, the device must be held in reset until the operating conditions are met.

For additional information, refer to Application Note AN607, "Power-up Trouble Shooting".

### 10.4.2 POWER-UP TIMER (PWRT)

The Power-up Timer provides a fixed 72 ms (nominal) time-out on power-up only, from POR or Brown-out Reset. The Power-up Timer operates on an internal RC oscillator. The chip is kept in reset as long as PWRT is active. The PWRT delay allows the  $V_{DD}$  to rise to an acceptable level. A configuration bit,  $\overline{PWRTE}$ , can disable (if set) or enable (if cleared or programmed) the Power-up Timer. The Power-up Timer should always be enabled when Brown-out Reset is enabled.

The Power-Up Time delay will vary from chip-to-chip and due to  $V_{DD}$ , temperature and process variation. See DC parameters for details.

### 10.4.3 OSCILLATOR START-UP TIMER (OST)

The Oscillator Start-Up Timer (OST) provides a 1024 oscillator cycle (from OSC1 input) delay after the PWRT delay is over. This ensures that the crystal oscillator or resonator has started and stabilized.

The OST time-out is invoked only for XT, LP and HS modes and only on power-on reset or wake-up from SLEEP.

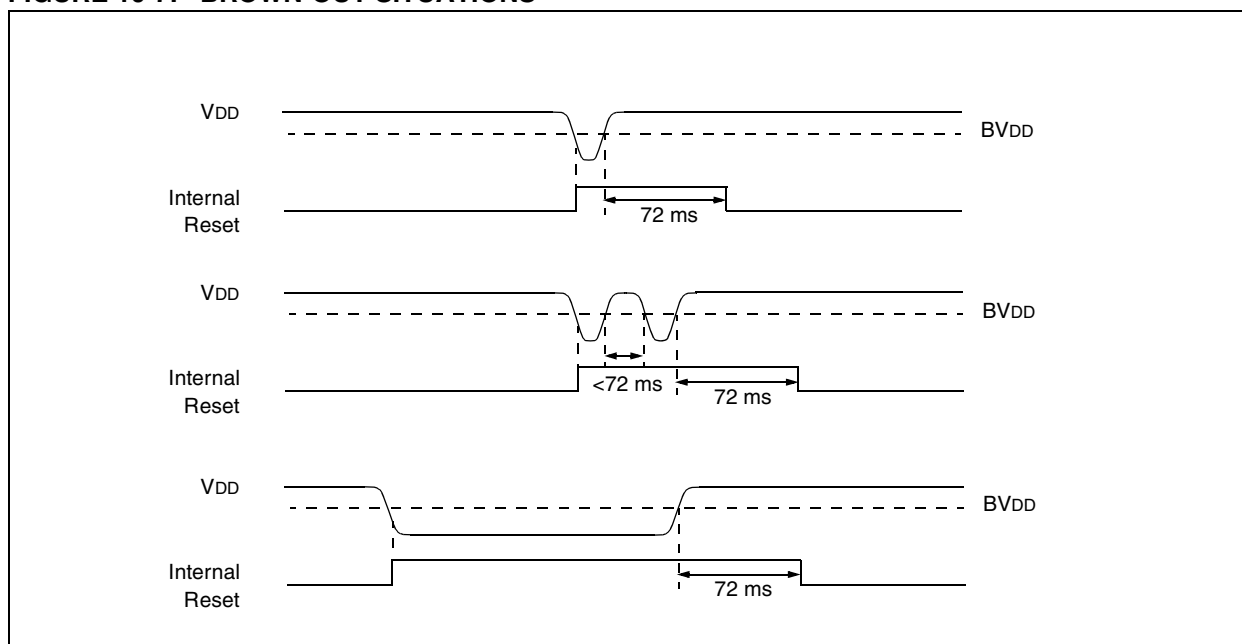
### 10.4.4 BROWN-OUT RESET (BOD)

The PIC16CE62X members have on-chip Brown-out Reset circuitry. A configuration bit, BOREN, can disable (if clear/programmed) or enable (if set) the Brown-out Reset circuitry. If  $V_{DD}$  falls below 4.0V (refer to  $BV_{DD}$  parameter D005) for greater than parameter (TBOR) in Table 13-5, the brown-out situation will reset the chip. A reset won't occur if  $V_{DD}$  falls below 4.0V for less than parameter (TBOR).

On any reset (Power-on, Brown-out, Watch-dog, etc.) the chip will remain in reset until  $V_{DD}$  rises above  $BV_{DD}$ . The Power-up Timer will then be invoked and will keep the chip in reset an additional 72 ms.

If  $V_{DD}$  drops below  $BV_{DD}$  while the Power-up Timer is running, the chip will go back into a Brown-out Reset and the Power-up Timer will be re-initialized. Once  $V_{DD}$  rises above  $BV_{DD}$ , the Power-Up Timer will execute a 72 ms reset. The Power-up Timer should always be enabled when Brown-out Reset is enabled. Figure 10-7 shows typical Brown-out situations.

**FIGURE 10-7: BROWN-OUT SITUATIONS**



## 10.4.5 TIME-OUT SEQUENCE

On power-up, the time-out sequence is as follows: First PWRT time-out is invoked after POR has expired, then OST is activated. The total time-out will vary based on oscillator configuration and  $\overline{\text{PWRT}}\text{E}$  bit status. For example, in RC mode with  $\overline{\text{PWRT}}\text{E}$  bit erased (PWRT disabled), there will be no time-out at all. Figure 10-8, Figure 10-9 and Figure 10-10 depict time-out sequences.

Since the time-outs occur from the POR pulse, if  $\overline{\text{MCLR}}$  is kept low long enough, the time-outs will expire. Then bringing  $\overline{\text{MCLR}}$  high will begin execution immediately (see Figure 10-9). This is useful for testing purposes or to synchronize more than one PIC® device operating in parallel.

Table 10-5 shows the reset conditions for some special registers, while Table 10-6 shows the reset conditions for all the registers.

## 10.4.6 POWER CONTROL (PCON)/STATUS REGISTER

The power control/status register, PCON (address 8Eh) has two bits.

Bit0 is  $\overline{\text{BOR}}$  (Brown-out).  $\overline{\text{BOR}}$  is unknown on power-on-reset. It must then be set by the user and checked on subsequent resets to see if  $\overline{\text{BOR}} = 0$  indicating that a brown-out has occurred. The  $\overline{\text{BOR}}$  status bit is a don't care and is not necessarily predictable if the brown-out circuit is disabled (by setting BODEN bit = 0 in the Configuration word).

Bit1 is  $\overline{\text{POR}}$  (Power-on-reset). It is a '0' on power-on-reset and unaffected otherwise. The user must write a '1' to this bit following a power-on-reset. On a subsequent reset, if  $\overline{\text{POR}}$  is '0', it will indicate that a power-on-reset must have occurred (VDD may have gone too low).

**TABLE 10-3: TIME-OUT IN VARIOUS SITUATIONS**

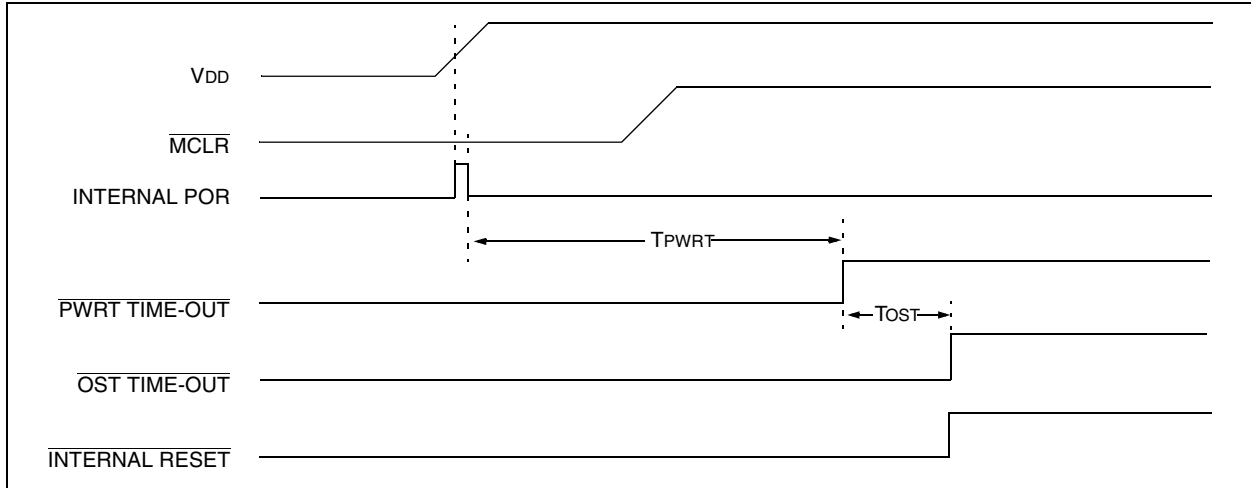
Oscillator Configuration	Power-up		Brown-out Reset	Wake-up from SLEEP
	$\overline{\text{PWRT}}\text{E} = 0$	$\overline{\text{PWRT}}\text{E} = 1$		
XT, HS, LP	72 ms + 1024 TOSC	1024 TOSC	72 ms + 1024 TOSC	1024 TOSC
RC	72 ms	—	72 ms	—

**TABLE 10-4: STATUS/PCON BITS AND THEIR SIGNIFICANCE**

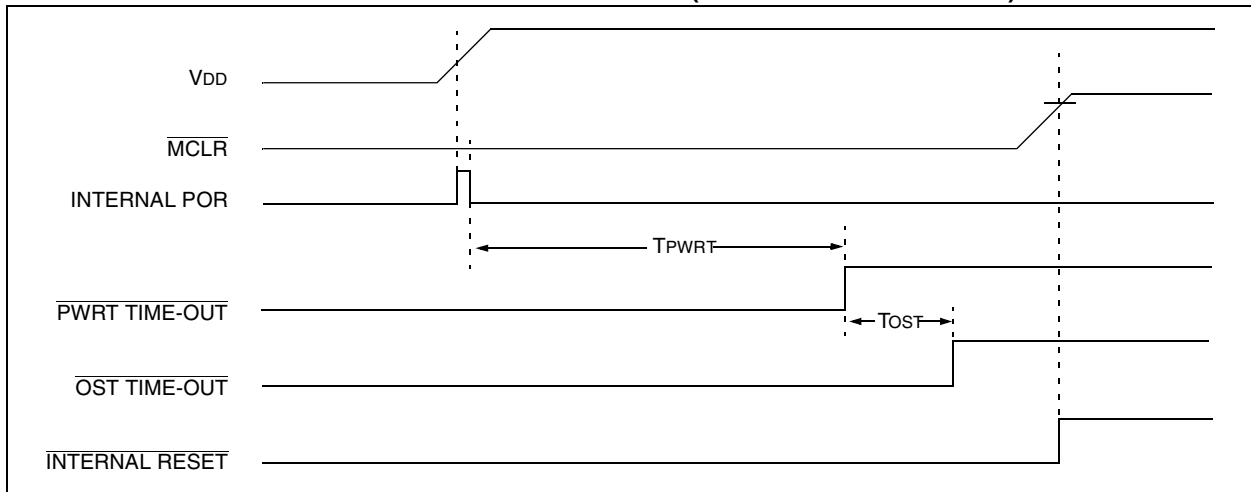
$\overline{\text{POR}}$	$\overline{\text{BOR}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	
0	X	1	1	Power-on-reset
0	X	0	X	Illegal, $\overline{\text{TO}}$ is set on $\overline{\text{POR}}$
0	X	X	0	Illegal, $\overline{\text{PD}}$ is set on $\overline{\text{POR}}$
1	0	X	X	Brown-out Reset
1	1	0	u	WDT Reset
1	1	0	0	WDT Wake-up
1	1	u	u	$\overline{\text{MCLR}}$ reset during normal operation
1	1	1	0	$\overline{\text{MCLR}}$ reset during SLEEP

Legend: x = unknown, u = unchanged

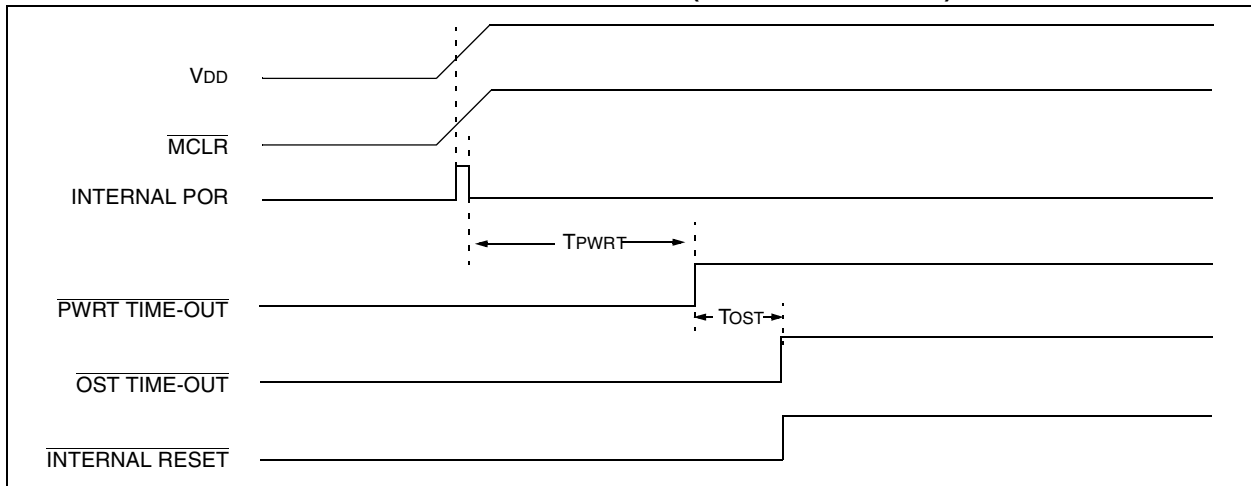
**FIGURE 10-8: TIME-OUT SEQUENCE ON POWER-UP ( $\overline{\text{MCLR}}$  NOT TIED TO  $V_{DD}$ ): CASE 1**



**FIGURE 10-9: TIME-OUT SEQUENCE ON POWER-UP ( $\overline{\text{MCLR}}$  NOT TIED TO  $V_{DD}$ ): CASE 2**



**FIGURE 10-10: TIME-OUT SEQUENCE ON POWER-UP ( $\overline{\text{MCLR}}$  TIED TO  $V_{DD}$ )**



## 10.5 Interrupts

The PIC16CE62X has 4 sources of interrupt:

- External interrupt RB0/INT
- TMR0 overflow interrupt
- PortB change interrupts (pins RB<7:4>)
- Comparator interrupt

The interrupt control register (INTCON) records individual interrupt requests in flag bits. It also has individual and global interrupt enable bits.

A global interrupt enable bit, GIE (INTCON<7>) enables (if set) all un-masked interrupts or disables (if cleared) all interrupts. Individual interrupts can be disabled through their corresponding enable bits in INTCON register. GIE is cleared on reset.

The “return from interrupt” instruction, RETFIE, exits interrupt routine, as well as sets the GIE bit, which re-enable RB0/INT interrupts.

The INT pin interrupt, the RB port change interrupt and the TMR0 overflow interrupt flags are contained in the INTCON register.

The peripheral interrupt flag is contained in the special register PIR1. The corresponding interrupt enable bit is contained in special registers PIE1.

When an interrupt is responded to, the GIE is cleared to disable any further interrupt, the return address is pushed into the stack and the PC is loaded with 0004h. Once in the interrupt service routine, the source(s) of

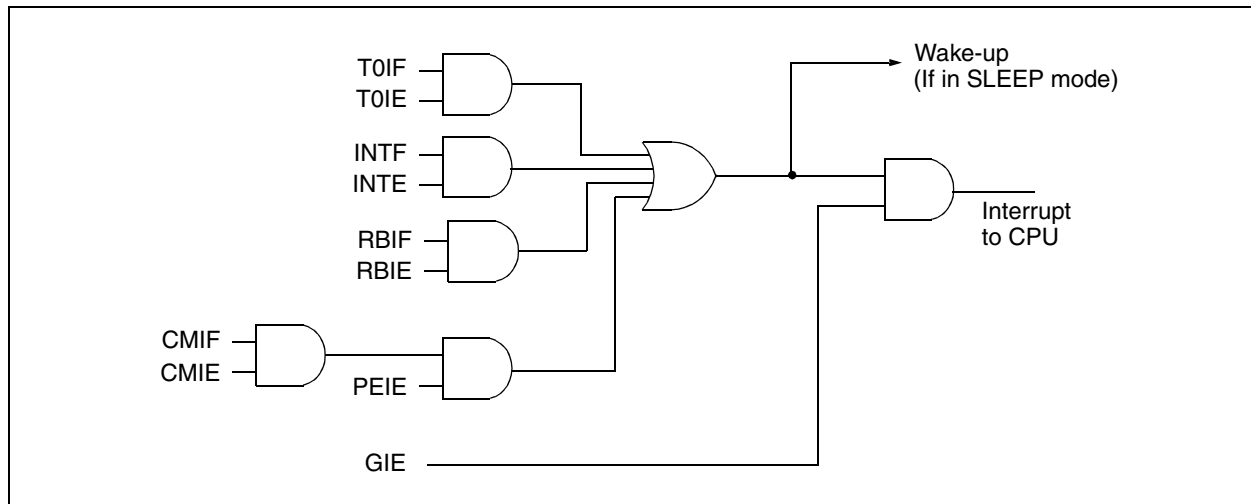
the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid RB0/INT recursive interrupts.

For external interrupt events, such as the INT pin or PORTB change interrupt, the interrupt latency will be three or four instruction cycles. The exact latency depends on when the interrupt event occurs (Figure 10-16). The latency is the same for one or two cycle instructions. Once in the interrupt service routine the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid multiple interrupt requests.

**Note 1:** Individual interrupt flag bits are set, regardless of the status of their corresponding mask bit or the GIE bit.

**2:** When an instruction that clears the GIE bit is executed, any interrupts that were pending for execution in the next cycle are ignored. The CPU will execute a NOP in the cycle immediately following the instruction which clears the GIE bit. The interrupts which were ignored are still pending to be serviced when the GIE bit is set again.

**FIGURE 10-15: INTERRUPT LOGIC**



# PIC16CE62X

TABLE 11-2: PIC16CE62X INSTRUCTION SET

Mnemonic, Operands	Description	Cycles	14-Bit Opcode				Status Affected	Notes	
			MSb		LSb				
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	1fff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0000	0011	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	1fff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWD <sub>T</sub>	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}, \overline{PD}$	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into standby mode	1	00	0000	0110	0011	$\overline{TO}, \overline{PD}$	
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

**Note 1:** When an I/O register is modified as a function of itself ( e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

**2:** If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 Module.

**3:** If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

## 11.1 Instruction Descriptions

### ADDLW Add Literal and W

Syntax:	[ <i>label</i> ] ADDLW    k				
Operands:	$0 \leq k \leq 255$				
Operation:	$(W) + k \rightarrow (W)$				
Status Affected:	C, DC, Z				
Encoding:	<table border="1"><tr><td>11</td><td>111x</td><td>kkkk</td><td>kkkk</td></tr></table>	11	111x	kkkk	kkkk
11	111x	kkkk	kkkk		
Description:	The contents of the W register are added to the eight bit literal 'k' and the result is placed in the W register.				
Words:	1				
Cycles:	1				
Example	<div>ADDLW    0x15</div> <div>Before Instruction</div> <div>W    =    0x10</div> <div>After Instruction</div> <div>W    =    0x25</div>				

### ANDLW AND Literal with W

Syntax:	[ <i>label</i> ] ANDLW <i>k</i>				
Operands:	$0 \leq k \leq 255$				
Operation:	(W) .AND. (k) $\rightarrow$ (W)				
Status Affected:	Z				
Encoding:	<table border="1"><tr><td>11</td><td>1001</td><td>kkkk</td><td>kkkk</td></tr></table>	11	1001	kkkk	kkkk
11	1001	kkkk	kkkk		
Description:	The contents of W register are AND'ed with the eight bit literal 'k'. The result is placed in the W register.				
Words:	1				
Cycles:	1				
Example	<div>ANDLW    0x5F</div> <div>Before Instruction</div> <div>W    =    0xA3</div> <div>After Instruction</div> <div>W    =    0x03</div>				

### ADDWF Add W and f

Syntax:	[ <i>label</i> ] ADDWF <i>f</i> , <i>d</i>				
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$				
Operation:	$(W) + (f) \rightarrow (dest)$				
Status Affected:	C, DC, Z				
Encoding:	<table border="1"><tr><td>00</td><td>0111</td><td>dfff</td><td>ffff</td></tr></table>	00	0111	dfff	ffff
00	0111	dfff	ffff		
Description:	Add the contents of the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.				
Words:	1				
Cycles:	1				
Example	<pre>           ADDWF   FSR, 0  Before Instruction            W = 0x17            FSR = 0xC2  After Instruction            W = 0xD9            FSR = 0xC2</pre>				

### ANDWF AND W with f

Syntax:	[ <i>label</i> ] ANDWF <i>f</i> , <i>d</i>				
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$				
Operation:	(W) .AND. (f) $\rightarrow$ (dest)				
Status Affected:	Z				
Encoding:	<table border="1"><tr><td>00</td><td>0101</td><td>dfff</td><td>ffff</td></tr></table>	00	0101	dfff	ffff
00	0101	dfff	ffff		
Description:	AND the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.				
Words:	1				
Cycles:	1				
Example	<pre>           ANDWF    FSR, 1             Before Instruction                W  =  0x17                FSR = 0xC2             After Instruction                W  =  0x17                FSR = 0x02</pre>				

## CLRWDT Clear Watchdog Timer

Syntax: [ *label* ] CLRWDT

Operands: None

Operation: 00h → WDT  
0 → WDT prescaler,  
1 →  $\overline{TO}$   
1 →  $\overline{PD}$

Status Affected:  $\overline{TO}$ ,  $\overline{PD}$

Encoding: 

00	0000	0110	0100
----	------	------	------

Description: CLRWDT instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. Status bits  $\overline{TO}$  and  $\overline{PD}$  are set.

Words: 1

Cycles: 1

Example CLRWDT

Before Instruction

WDT counter = ?

After Instruction

WDT counter = 0x00

WDT prescaler = 0

$\overline{TO}$  = 1

$\overline{PD}$  = 1

## COMF Complement f

Syntax: [ *label* ] COMF f,d

Operands:  $0 \leq f \leq 127$   
 $d \in [0,1]$

Operation: ( $\bar{f}$ ) → (dest)

Status Affected: Z

Encoding: 

00	1001	dfff	ffff
----	------	------	------

Description: The contents of register 'f' are complemented. If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored back in register 'f'.

Words: 1

Cycles: 1

Example COMF REG1, 0

Before Instruction

REG1 = 0x13

After Instruction

REG1 = 0x13

W = 0xEC

## DECF Decrement f

Syntax: [ *label* ] DECF f,d

Operands:  $0 \leq f \leq 127$   
 $d \in [0,1]$

Operation: (f) - 1 → (dest)

Status Affected: Z

Encoding: 

00	0011	dfff	ffff
----	------	------	------

Description: Decrement register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

Words: 1

Cycles: 1

Example DECF CNT, 1

Before Instruction

CNT = 0x01

Z = 0

After Instruction

CNT = 0x00

Z = 1

## DECFSZ Decrement f, Skip if 0

Syntax: [ *label* ] DECFSZ f,d

Operands:  $0 \leq f \leq 127$   
 $d \in [0,1]$

Operation: (f) - 1 → (dest); skip if result = 0

Status Affected: None

Encoding: 

00	1011	dfff	ffff
----	------	------	------

Description: The contents of register 'f' are decremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'. If the result is 0, the next instruction, which is already fetched, is discarded. A NOP is executed instead making it a two-cycle instruction.

Words: 1

Cycles: 1(2)

Example  
HERE DECFSZ CNT, 1  
GOTO LOOP

CONTINUE  
•  
•

Before Instruction

PC = address HERE

After Instruction

CNT = CNT - 1

if CNT = 0,

PC = address CONTINUE

if CNT ≠ 0,

PC = address HERE+1

## NOP No Operation

Syntax: [ *label* ] NOP

Operands: None

Operation: No operation

Status Affected: None

Encoding: 

00	0000	0xx0	0000
----	------	------	------

Description: No operation.

Words: 1

Cycles: 1

Example NOP

## RETFIE Return from Interrupt

Syntax: [ *label* ] RETFIE

Operands: None

Operation: TOS → PC,  
1 → GIE

Status Affected: None

Encoding: 

00	0000	0000	1001
----	------	------	------

Description: Return from Interrupt. Stack is POPed and Top of Stack (TOS) is loaded in the PC. Interrupts are enabled by setting Global Interrupt Enable bit, GIE (INTCON<7>). This is a two-cycle instruction.

Words: 1

Cycles: 2

Example RETFIE

After Interrupt

PC = TOS  
GIE = 1

## OPTION Load Option Register

Syntax: [ *label* ] OPTION

Operands: None

Operation: (W) → OPTION

Status Affected: None

Encoding: 

00	0000	0110	0010
----	------	------	------

Description: The contents of the W register are loaded in the OPTION register. This instruction is supported for code compatibility with PIC16C5X products. Since OPTION is a readable/writable register, the user can directly address it.

Words: 1

Cycles: 1

Example

**To maintain upward compatibility with future PIC® MCU products, do not use this instruction.**

## RETLW Return with Literal in W

Syntax: [ *label* ] RETLW k

Operands:  $0 \leq k \leq 255$

Operation: k → (W);  
TOS → PC

Status Affected: None

Encoding: 

11	01xx	kkkk	kkkk
----	------	------	------

Description: The W register is loaded with the eight bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two-cycle instruction.

Words: 1

Cycles: 2

Example

```
CALL TABLE      ;W contains table
                  ;offset value
                  ;W now has table
•
value
•
TABLE
•
ADDWF PC          ;W = offset
RETLW k1          ;Begin table
RETLW k2          ;
•
•
•
RETLW kn          ; End of table
```

Before Instruction

W = 0x07

After Instruction

W = value of k8



stand-alone mode the PRO MATE II can read, verify or program PIC devices. It can also set code-protect bits in this mode.

## **PICSTART Plus Entry Level Development System**

The PICSTART programmer is an easy-to-use, low-cost prototype programmer. It connects to the PC via one of the COM (RS-232) ports. MPLAB Integrated Development Environment software makes using the programmer simple and efficient.

PICSTART Plus supports all PIC devices with up to 40 pins. Larger pin count devices such as the PIC16C92X, and PIC17C76X may be supported with an adapter socket. PICSTART Plus is CE compliant.

## **SIMICE Entry-Level Hardware Simulator**

SIMICE is an entry-level hardware development system designed to operate in a PC-based environment with Microchip's simulator MPLAB-SIM. Both SIMICE and MPLAB-SIM run under Microchip Technology's MPLAB Integrated Development Environment (IDE) software. Specifically, SIMICE provides hardware simulation for Microchip's PIC12C5XX, PIC12CE5XX, and PIC16C5X families of PIC 8-bit microcontrollers. SIMICE works in conjunction with MPLAB-SIM to provide non-real-time I/O port emulation. SIMICE enables a developer to run simulator code for driving the target system. In addition, the target system can provide input to the simulator code. This capability allows for simple and interactive debugging without having to manually generate MPLAB-SIM stimulus files. SIMICE is a valuable debugging tool for entry-level system development.

## **PICDEM-1 Low-Cost PIC MCU Demonstration Board**

The PICDEM-1 is a simple board which demonstrates the capabilities of several of Microchip's microcontrollers. The microcontrollers supported are: PIC16C5X (PIC16C54 to PIC16C58A), PIC16C61, PIC16C62X, PIC16C71, PIC16C8X, PIC17C42, PIC17C43 and PIC17C44. All necessary hardware and software is included to run basic demo programs. The users can program the sample microcontrollers provided with the PICDEM-1 board, on a PRO MATE II or PICSTART-Plus programmer, and easily test firmware. The user can also connect the PICDEM-1 board to the MPLAB-ICE emulator and download the firmware to the emulator for testing. Additional prototype area is available for the user to build some additional hardware and connect it to the microcontroller socket(s). Some of the features include an RS-232 interface, a potentiometer for simulated analog input, push-button switches and eight LEDs connected to PORTB.

## **PICDEM-2 Low-Cost PIC16CXX Demonstration Board**

The PICDEM-2 is a simple demonstration board that supports the PIC16C62, PIC16C64, PIC16C65, PIC16C73 and PIC16C74 microcontrollers. All the necessary hardware and software is included to run the basic demonstration programs. The user can program the sample microcontrollers provided with the PICDEM-2 board, on a PRO MATE II programmer or PICSTART-Plus, and easily test firmware. The MPLAB-ICE emulator may also be used with the PICDEM-2 board to test firmware. Additional prototype area has been provided to the user for adding additional hardware and connecting it to the microcontroller socket(s). Some of the features include a RS-232 interface, push-button switches, a potentiometer for simulated analog input, a Serial EEPROM to demonstrate usage of the I<sup>2</sup>C bus and separate headers for connection to an LCD module and a keypad.

## **PICDEM-3 Low-Cost PIC16CXXX Demonstration Board**

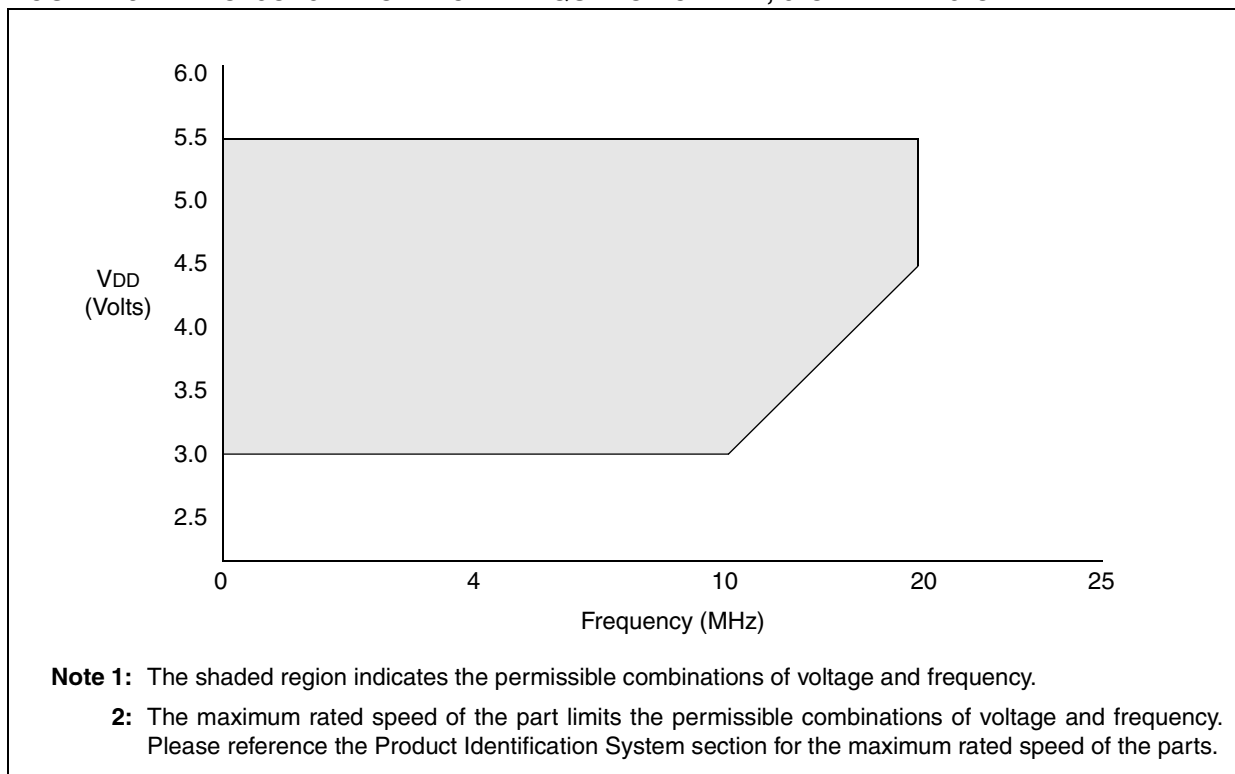
The PICDEM-3 is a simple demonstration board that supports the PIC16C923 and PIC16C924 in the PLCC package. It will also support future 44-pin PLCC microcontrollers with a LCD Module. All the necessary hardware and software is included to run the basic demonstration programs. The user can program the sample microcontrollers provided with the PICDEM-3 board, on a PRO MATE II programmer or PICSTART Plus with an adapter socket, and easily test firmware. The MPLAB-ICE emulator may also be used with the PICDEM-3 board to test firmware. Additional prototype area has been provided to the user for adding hardware and connecting it to the microcontroller socket(s). Some of the features include an RS-232 interface, push-button switches, a potentiometer for simulated analog input, a thermistor and separate headers for connection to an external LCD module and a keypad. Also provided on the PICDEM-3 board is an LCD panel, with 4 commons and 12 segments, that is capable of displaying time, temperature and day of the week. The PICDEM-3 provides an additional RS-232 interface and Windows 3.1 software for showing the demultiplexed LCD signals on a PC. A simple serial interface allows the user to construct a hardware demultiplexer for the LCD signals.

## **PICDEM-17**

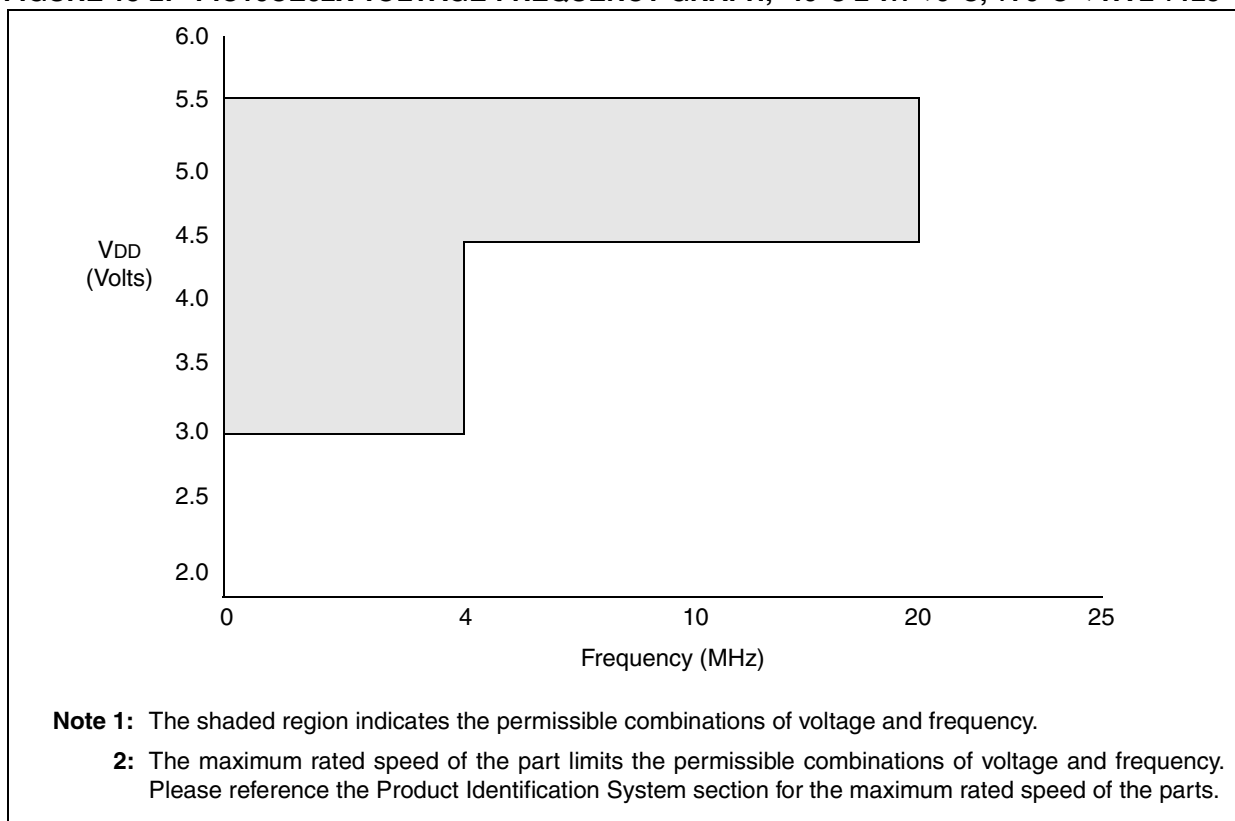
The PICDEM-17 is an evaluation board that demonstrates the capabilities of several Microchip microcontrollers, including PIC17C752, PIC17C756, PIC17C762, and PIC17C766. All necessary hardware is included to run basic demo programs, which are supplied on a 3.5-inch disk. A programmed sample is included, and the user may erase it and program it with the other sample programs using the PRO MATE II or PICSTART Plus device programmers and easily debug

# PIC16CE62X

**FIGURE 13-1: PIC16CE62X VOLTAGE-FREQUENCY GRAPH,  $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$**



**FIGURE 13-2: PIC16CE62X VOLTAGE-FREQUENCY GRAPH,  $-40^{\circ}\text{C} \leq T_A < 0^{\circ}\text{C}$ ,  $+70^{\circ}\text{C} < T_A \leq +125^{\circ}\text{C}$**



# PIC16CE62X

---

NOTES:

# PIC16CE62X

---

NOTES:

NOTES: