



Welcome to 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	896B (512 x 14)
Program Memory Type	OTP
EEPROM Size	128 x 8
RAM Size	96 x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 5.5V
Data Converters	-
Oscillator Type	External
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SSOP (0.209", 5.30mm Width)
Supplier Device Package	20-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16lce623-04e-ss

4.2 Data Memory Organization

The data memory (Figure 4-4 and Figure 4-5) is partitioned into two Banks which contain the General Purpose Registers and the Special Function Registers. Bank 0 is selected when the RP0 bit is cleared. Bank 1 is selected when the RP0 bit (STATUS <5>) is set. The Special Function Registers are located in the first 32 locations of each Bank. Register locations 20-7Fh (Bank0) on the PIC16CE623/624 and 20-7Fh (Bank0) and A0-BFh (Bank1) on the PIC16CE625 are General Purpose Registers implemented as static RAM. Some special purpose registers are mapped in Bank 1. In all three microcontrollers, address space F0h-FFh (Bank1) is mapped to 70-7Fh (Bank0) as common RAM.

4.2.1 GENERAL PURPOSE REGISTER FILE

The register file is organized as 96 x 8 in the PIC16CE623/624 and 128 x 8 in the PIC16CE625. Each is accessed either directly or indirectly through the File Select Register FSR (Section 4.4).

FIGURE 4-4: DATA MEMORY MAP FOR THE PIC16CE623/624

File Address		File Address	
00h	INDF ⁽¹⁾	80h	INDF ⁽¹⁾
01h	TMR0	81h	OPTION
02h	PCL	82h	PCL
03h	STATUS	83h	STATUS
04h	FSR	84h	FSR
05h	PORTA	85h	TRISA
06h	PORTB	86h	TRISB
07h		87h	
08h		88h	
09h		89h	
0Ah	PCLATH	8Ah	PCLATH
0Bh	INTCON	8Bh	INTCON
0Ch	PIR1	8Ch	PIE1
0Dh		8Dh	
0Eh		8Eh	PCON
0Fh		8Fh	
10h		90h	EEINTF
11h		91h	
12h		92h	
13h		93h	
14h		94h	
15h		95h	
16h		96h	
17h		97h	
18h		98h	
19h		99h	
1Ah		9Ah	
1Bh		9Bh	
1Ch		9Ch	
1Dh		9Dh	
1Eh		9Eh	
1Fh	CMCON	9Fh	VRCON
20h		A0h	
	General Purpose Register		
		EFh	
		F0h	Accesses 70h-7Fh
7Fh		FFh	
	Bank 0		Bank 1

Unimplemented data memory locations, read as '0'.
Note 1: Not a physical register.

FIGURE 4-5: DATA MEMORY MAP FOR THE PIC16CE625

File Address		File Address	
00h	INDF ⁽¹⁾	80h	INDF ⁽¹⁾
01h	TMR0	81h	OPTION
02h	PCL	82h	PCL
03h	STATUS	83h	STATUS
04h	FSR	84h	FSR
05h	PORTA	85h	TRISA
06h	PORTB	86h	TRISB
07h		87h	
08h		88h	
09h		89h	
0Ah	PCLATH	8Ah	PCLATH
0Bh	INTCON	8Bh	INTCON
0Ch	PIR1	8Ch	PIE1
0Dh		8Dh	
0Eh		8Eh	PCON
0Fh		8Fh	
10h		90h	EEINTF
11h		91h	
12h		92h	
13h		93h	
14h		94h	
15h		95h	
16h		96h	
17h		97h	
18h		98h	
19h		99h	
1Ah		9Ah	
1Bh		9Bh	
1Ch		9Ch	
1Dh		9Dh	
1Eh		9Eh	
1Fh	CMCON	9Fh	VRCON
20h		A0h	
	General Purpose Register		General Purpose Register
		BFh	
		C0h	
		F0h	Accesses 70h-7Fh
7Fh		FFh	
	Bank 0		Bank 1

Unimplemented data memory locations, read as '0'.
Note 1: Not a physical register.

4.2.2.1 STATUS REGISTER

The STATUS register, shown in Register 4-1, contains the arithmetic status of the ALU, the RESET status and the bank select bits for data memory.

The STATUS register can be the destination for any instruction, like any other register. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits are not writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS` will clear the upper-three bits and set the Z bit. This leaves the status register as 000uu1uu (where u = unchanged).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions are used to alter the STATUS register, because these instructions do not affect any status bit. For other instructions, not affecting any status bits, see the "Instruction Set Summary".

Note 1: The IRP and RP1 bits (STATUS<7:6>) are not used by the PIC16CE62X and should be programmed as '0'. Use of these bits as general purpose R/W bits is NOT recommended, since this may affect upward compatibility with future products.

Note 2: The C and DC bits operate as a Borrow and Digit Borrow out bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

REGISTER 4-1: STATUS REGISTER (ADDRESS 03H OR 83H)

Reserved	Reserved	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C
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: **IRP:** The IRP bit is reserved on the PIC16CE62X, always maintain this bit clear.

bit 6:5 **RP<1:0>:** Register Bank Select bits (used for direct addressing)
11 = Bank 3 (180h - 1FFh)
10 = Bank 2 (100h - 17Fh)
01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)
Each bank is 128 bytes. The RP1 bit is reserved, always maintain this bit clear.

bit 4: **$\overline{\text{TO}}$:** Time-out bit
1 = After power-up, `CLRWDT` instruction, or `SLEEP` instruction
0 = A WDT time-out occurred

bit 3: **$\overline{\text{PD}}$:** Power-down bit
1 = After power-up or by the `CLRWDT` instruction
0 = By execution of the `SLEEP` instruction

bit 2: **Z:** Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1: **DC:** Digit carry/borrow bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions) (for borrow the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

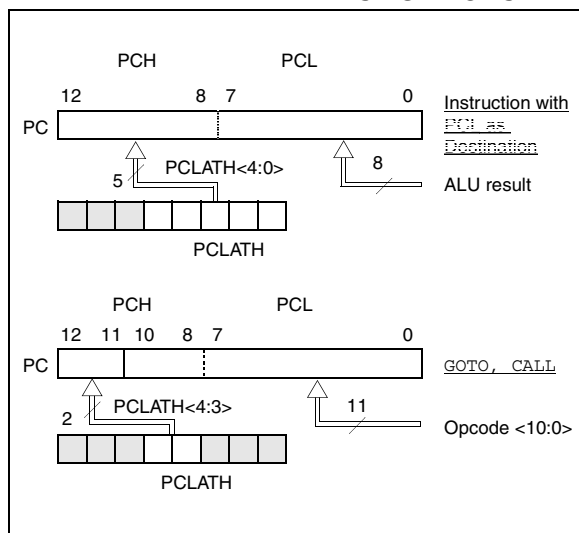
bit 0: **C:** Carry/borrow bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)
1 = A carry-out from the most significant bit of the result occurred
0 = No carry-out from the most significant bit of the result occurred

Note: For borrow the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high or low order bit of the source register.

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.

6.1 Bus Characteristics

In this section, the term “processor” refers to the portion of the PIC16CE62X that interfaces to the EEPROM through software manipulating the EEINTF register. The following **bus protocol** is to be used with the EEPROM data memory.

- Data transfer may be initiated only when the bus is not busy.
- During data transfer, the data line must remain stable whenever the clock line is HIGH. Changes in the data line while the clock line is HIGH will be interpreted by the EEPROM as a START or STOP condition.

Accordingly, the following bus conditions have been defined (Figure 6-1).

6.1.1 BUS NOT BUSY (A)

Both data and clock lines remain HIGH.

6.1.2 START DATA TRANSFER (B)

A HIGH to LOW transition of the SDA line while the clock (SCL) is HIGH determines a START condition. All commands must be preceded by a START condition.

6.1.3 STOP DATA TRANSFER (C)

A LOW to HIGH transition of the SDA line while the clock (SCL) is HIGH determines a STOP condition. All operations must be ended with a STOP condition.

6.1.4 DATA VALID (D)

The state of the data line represents valid data when, after a START condition, the data line is stable for the duration of the HIGH period of the clock signal.

The data on the line must be changed during the LOW period of the clock signal. There is one bit of data per clock pulse.

Each data transfer is initiated with a START condition and terminated with a STOP condition. The number of the data bytes transferred between the START and STOP conditions is determined by the processor and is theoretically unlimited, although only the last sixteen will be stored when doing a write operation. When an overwrite does occur, it will replace data in a first-in, first-out fashion.

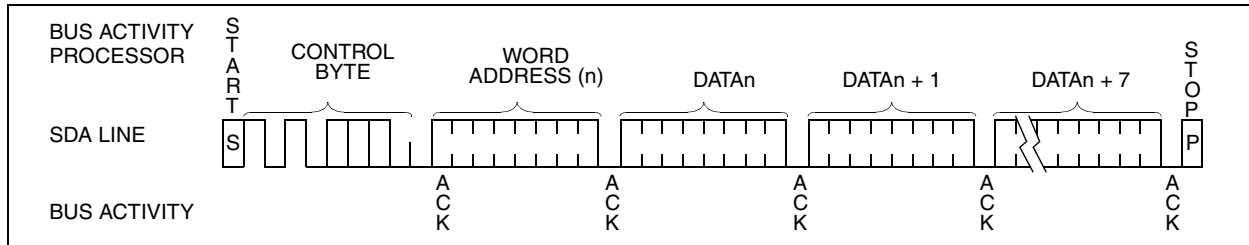
6.1.5 ACKNOWLEDGE

The EEPROM will generate an acknowledge after the reception of each byte. The processor must generate an extra clock pulse which is associated with this acknowledge bit.

Note: Acknowledge bits are not generated if an internal programming cycle is in progress.
--

When the EEPROM acknowledges, it pulls down the SDA line during the acknowledge clock pulse in such a way that the SDA line is stable LOW during the HIGH period of the acknowledge related clock pulse. Of course, setup and hold times must be taken into account. The processor must signal an end of data to the EEPROM by not generating an acknowledge bit on the last byte that has been clocked out of the EEPROM. In this case, the EEPROM must leave the data line HIGH to enable the processor to generate the STOP condition (Figure 6-2).

FIGURE 6-6: PAGE WRITE



6.5 Read Operation

Read operations are initiated in the same way as write operations with the exception that the R/W bit of the EEPROM address is set to one. There are three basic types of read operations: current address read, random read, and sequential read.

6.6 Current Address Read

The EEPROM contains an address counter that maintains the address of the last word accessed, internally incremented by one. Therefore, if the previous access (either a read or write operation) was to address n , the next current address read operation would access data from address $n + 1$. Upon receipt of the EEPROM address with R/W bit set to one, the EEPROM issues an acknowledge and transmits the eight bit data word. The processor will not acknowledge the transfer, but does generate a stop condition and the EEPROM discontinues transmission (Figure 6-7).

6.7 Random Read

Random read operations allow the processor to access any memory location in a random manner. To perform this type of read operation, first the word address must be set. This is done by sending the word address to the EEPROM as part of a write operation. After the word address is sent, the processor generates a start condition following the acknowledge. This terminates the write operation, but not before the internal address pointer is set. Then the processor issues the control byte again, but with the R/W bit set to a one. The EEPROM will then issue an acknowledge and transmits the eight bit data word. The processor will not acknowledge the transfer, but does generate a stop condition and the EEPROM discontinues transmission (Figure 6-8).

6.8 Sequential Read

Sequential reads are initiated in the same way as a random read except that after the EEPROM transmits the first data byte, the processor issues an acknowledge as opposed to a stop condition in a random read. This directs the EEPROM to transmit the next sequentially addressed 8-bit word (Figure 6-9).

To provide sequential reads, the EEPROM contains an internal address pointer which is incremented by one at the completion of each operation. This address pointer allows the entire memory contents to be serially read during one operation.

6.9 Noise Protection

The EEPROM employs a V_{CC} threshold detector circuit, which disables the internal erase/write logic if the V_{CC} is below 1.5 volts at nominal conditions.

The SCL and SDA inputs have Schmitt trigger and filter circuits, which suppress noise spikes to assure proper device operation even on a noisy bus.

7.2 Using Timer0 with External Clock

When an external clock input is used for Timer0, it must meet certain requirements. The external clock requirement is due to internal phase clock (TOSC) synchronization. Also, there is a delay in the actual incrementing of Timer0 after synchronization.

7.2.1 EXTERNAL CLOCK SYNCHRONIZATION

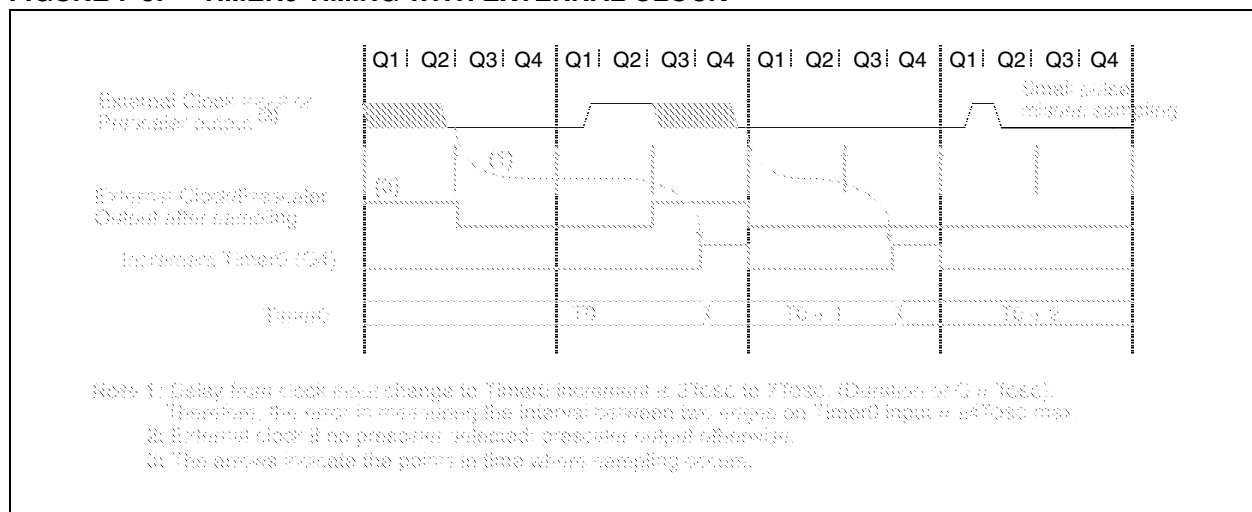
When no prescaler is used, the external clock input is the same as the prescaler output. The synchronization of T0CKI with the internal phase clocks is accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the internal phase clocks (Figure 7-5). Therefore, it is necessary for T0CKI to be high for at least 2TOSC (and a small RC delay of 20 ns) and low for at least 2TOSC (and a small RC delay of 20 ns). Refer to the electrical specification of the desired device.

When a prescaler is used, the external clock input is divided by the asynchronous ripple-counter type prescaler so that the prescaler output is symmetrical. For the external clock to meet the sampling requirement, the ripple-counter must be taken into account. Therefore, it is necessary for T0CKI to have a period of at least 4TOSC (and a small RC delay of 40 ns) divided by the prescaler value. The only requirement on T0CKI high and low time is that they do not violate the minimum pulse width requirement of 10 ns. Refer to parameters 40, 41 and 42 in the electrical specification of the desired device.

7.2.2 TIMER0 INCREMENT DELAY

Since the prescaler output is synchronized with the internal clocks, there is a small delay from the time the external clock edge occurs to the time the TMR0 is actually incremented. Figure 7-5 shows the delay from the external clock edge to the timer incrementing.

FIGURE 7-5: TIMER0 TIMING WITH EXTERNAL CLOCK



7.3.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control (i.e., it can be changed “on-the-fly” during program execution). To avoid an unintended device RESET, the following instruction sequence (Example 7-1) must be executed when changing the prescaler assignment from Timer0 to WDT.

EXAMPLE 7-1: CHANGING PRESCALER (TIMER0→WDT)

```

1.BCF STATUS, RP0 ;Skip if already in
                      ; Bank 0
2.CLRWDT            ;Clear WDT
3.CLRF TMR0         ;Clear TMR0 & Prescaler
4.BSF STATUS, RP0  ;Bank 1
5.MOVLW '00101111'b ;These 3 lines (5, 6, 7)
6.MOVWF OPTION      ; are required only if
                      ; desired PS<2:0> are
7.CLRWDT            ; 000 or 001
8.MOVLW '00101xxx'b ;Set Postscaler to
9.MOVWF OPTION      ; desired WDT rate
10.BCF STATUS, RP0 ;Return to Bank 0

```

To change prescaler from the WDT to the TMR0 module, use the sequence shown in Example 7-2. This precaution must be taken even if the WDT is disabled.

EXAMPLE 7-2: CHANGING PRESCALER (WDT→TIMER0)

```

CLRWDT            ;Clear WDT and
                  ;prescaler
BSF STATUS, RP0
MOVLW b'xxxx0xxx' ;Select TMR0, new
                  ;prescale value and
                  ;clock source
MOVWF OPTION_REG
BCF STATUS, RP0

```

TABLE 7-1: REGISTERS ASSOCIATED WITH TIMER0

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
01h	TMR0	Timer0 module register								xxxx xxxx	uuuu uuuu
0Bh/8Bh	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 000x	0000 000u
81h	OPTION	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
85h	TRISA	—	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111

Legend: — = Unimplemented locations, read as '0', x = unknown, u = unchanged.

Note: Shaded bits are not used by TMR0 module.

The code example in Example 8-1 depicts the steps required to configure the comparator module. RA3 and RA4 are configured as digital output. RA0 and RA1 are configured as the V- inputs and RA2 as the V+ input to both comparators.

EXAMPLE 8-1: INITIALIZING COMPARATOR MODULE

```
FLAG_REG EQU      0X20
CLRF   FLAG_REG   ;Init flag register
CLRF   PORTA       ;Init PORTA
MOVF   CMCON,W     ;Move comparator contents to W
ANDLW  0xC0        ;Mask comparator bits
IORWF  FLAG_REG,F  ;Store bits in flag register
MOVLW  0x03        ;Init comparator mode
MOVWF  CMCON       ;CM<2:0> = 011
BSF    STATUS,RP0  ;Select Bank1
MOVLW  0x07        ;Initialize data direction
MOVWF  TRISA       ;Set RA<2:0> as inputs
                        ;RA<4:3> as outputs
                        ;TRISA<7:5> always read '0'

BCF    STATUS,RP0  ;Select Bank 0
CALL   DELAY 10    ;10µs delay
MOVF   CMCON,F     ;Read CMCON to end change condition
BCF    PIR1,CMIF   ;Clear pending interrupts
BSF    STATUS,RP0  ;Select Bank 1
BSF    PIE1,CMIE   ;Enable comparator interrupts
BCF    STATUS,RP0  ;Select Bank 0
BSF    INTCON,PEIE ;Enable peripheral interrupts
BSF    INTCON,GIE  ;Global interrupt enable
```

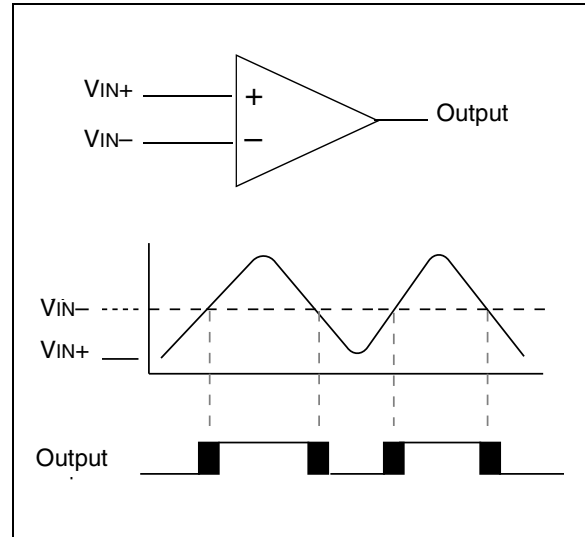
8.2 Comparator Operation

A single comparator is shown in Figure 8-2 along with the relationship between the analog input levels and the digital output. When the analog input at VIN+ is less than the analog input VIN-, the output of the comparator is a digital low level. When the analog input at VIN+ is greater than the analog input VIN-, the output of the comparator is a digital high level. The shaded areas of the output of the comparator in Figure 8-2 represent the uncertainty due to input offsets and response time.

8.3 Comparator Reference

An external or internal reference signal may be used depending on the comparator operating mode. The analog signal that is present at VIN- is compared to the signal at VIN+, and the digital output of the comparator is adjusted accordingly (Figure 8-2).

FIGURE 8-2: SINGLE COMPARATOR



8.3.1 EXTERNAL REFERENCE SIGNAL

When external voltage references are used, the comparator module can be configured to have the comparators operate from the same or different reference sources. However, threshold detector applications may require the same reference. The reference signal must be between VSS and VDD and can be applied to either pin of the comparator(s).

8.3.2 INTERNAL REFERENCE SIGNAL

The comparator module also allows the selection of an internally generated voltage reference for the comparators. Section 13, Instruction Sets, contains a detailed description of the Voltage Reference Module that provides this signal. The internal reference signal is used when the comparators are in mode CM<2:0>=010 (Figure 8-1). In this mode, the internal voltage reference is applied to the VIN+ pin of both comparators.

9.0 VOLTAGE REFERENCE MODULE

The Voltage Reference is a 16-tap resistor ladder network that provides a selectable voltage reference. The resistor ladder is segmented to provide two ranges of VREF values and has a power-down function to conserve power when the reference is not being used. The VRCON register controls the operation of the reference as shown in Register 9-1. The block diagram is given in Figure 9-1.

9.1 Configuring the Voltage Reference

The Voltage Reference can output 16 distinct voltage levels for each range.

The equations used to calculate the output of the Voltage Reference are as follows:

$$\text{if } VRR = 1: VREF = (VR_{<3:0>} / 24) \times VDD$$

$$\text{if } VRR = 0: VREF = (VDD \times 1/4) + (VR_{<3:0>} / 32) \times VDD$$

The setting time of the Voltage Reference must be considered when changing the VREF output (Table 13-1). Example 9-1 shows an example of how to configure the Voltage Reference for an output voltage of 1.25V with VDD = 5.0V.

REGISTER 9-1: VRCON REGISTER (ADDRESS 9Fh)

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
VREN	VROE	VRR	—	VR3	VR2	VR1	VR0
bit7							bit0

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

bit 7: **VREN:** VREF Enable
1 = VREF circuit powered on
0 = VREF circuit powered down, no IDD drain

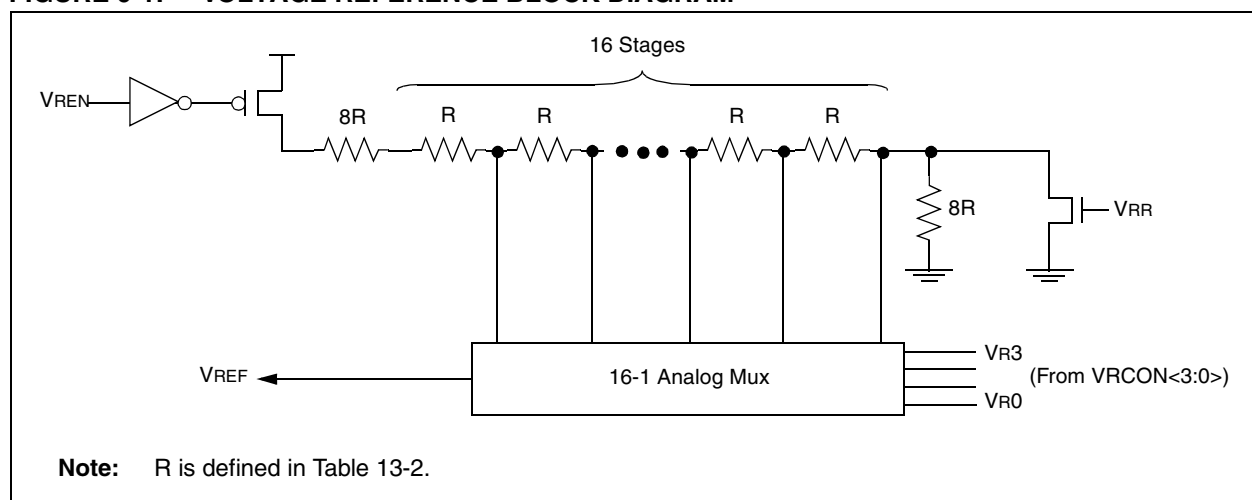
bit 6: **VROE:** VREF Output Enable
1 = VREF is output on RA2 pin
0 = VREF is disconnected from RA2 pin

bit 5: **VRR:** VREF Range selection
1 = Low Range
0 = High Range

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

bit 3-0: **VR<3:0>:** VREF value selection $0 \leq VR_{[3:0]} \leq 15$
when VRR = 1: $VREF = (VR_{<3:0>} / 24) \times VDD$
when VRR = 0: $VREF = 1/4 \times VDD + (VR_{<3:0>} / 32) \times VDD$

FIGURE 9-1: VOLTAGE REFERENCE BLOCK DIAGRAM



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

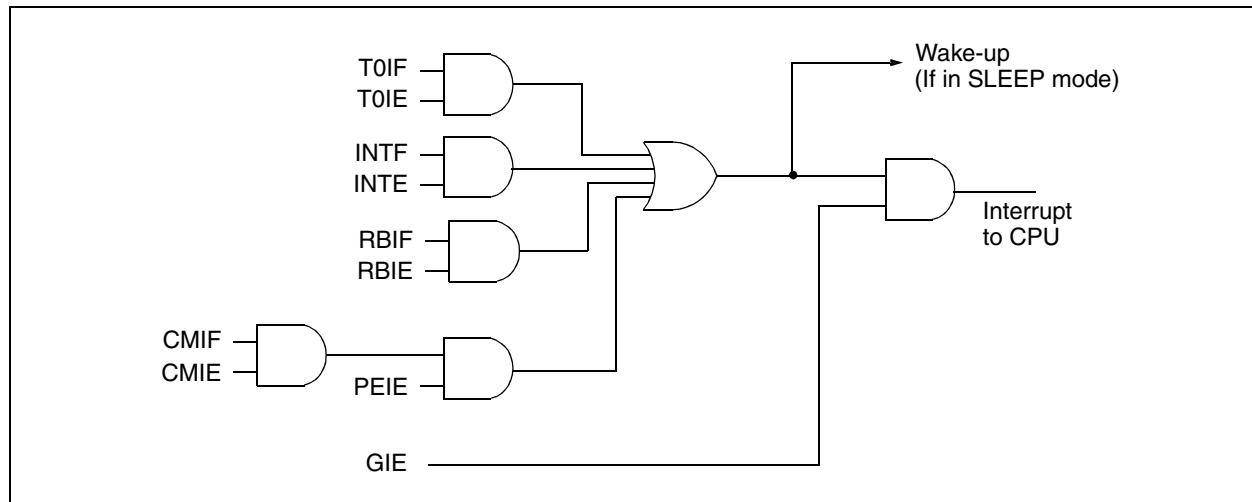
the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid RB0/INT recursive interrupts.

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



10.8 Power-Down Mode (SLEEP)

The Power-down mode is entered by executing a SLEEP instruction.

If enabled, the Watchdog Timer will be cleared but keeps running, the \overline{PD} bit in the STATUS register is cleared, the \overline{TO} bit is set and the oscillator driver is turned off. The I/O ports maintain the status they had before SLEEP was executed (driving high, low, or hi-impedance).

For lowest current consumption in this mode, all I/O pins should be either at VDD or VSS, with no external circuitry drawing current from the I/O pin, and the comparators and VREF should be disabled. I/O pins that are hi-impedance inputs should be pulled high or low externally to avoid switching currents caused by floating inputs. The T0CKI input should also be at VDD or VSS for lowest current consumption. The contribution from on chip pull-ups on PORTB should be considered.

The \overline{MCLR} pin must be at a logic high level (V_{IHMC}).

Note: It should be noted that a RESET generated by a WDT time-out does not drive \overline{MCLR} pin low.

The first event will cause a device reset. The two latter events are considered a continuation of program execution. The \overline{TO} and \overline{PD} bits in the STATUS register can be used to determine the cause of device reset. \overline{PD} bit, which is set on power-up is cleared when SLEEP is invoked. \overline{TO} bit is cleared if WDT wake-up occurred.

When the SLEEP instruction is being executed, the next instruction (PC + 1) is pre-fetched. For the device to wake-up through an interrupt event, the corresponding interrupt enable bit must be set (enabled). Wake-up is regardless of the state of the GIE bit. If the GIE bit is clear (disabled), the device continues execution at the instruction after the SLEEP instruction. If the GIE bit is set (enabled), the device executes the instruction after the SLEEP instruction and then branches to the interrupt address (0004h). In cases where the execution of the instruction following SLEEP is not desirable, the user should have an NOP after the SLEEP instruction.

Note: If the global interrupts are disabled (GIE is cleared), but any interrupt source has both its interrupt enable bit and the corresponding interrupt flag bits set, the device will immediately wake-up from sleep. The sleep instruction is completely executed.

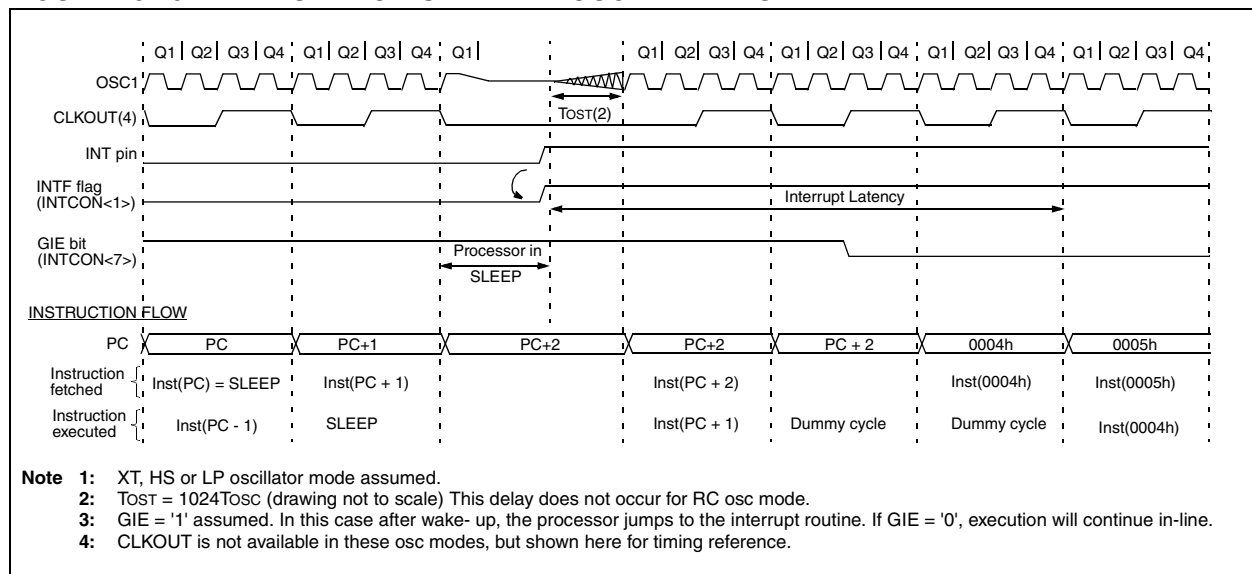
10.8.1 WAKE-UP FROM SLEEP

The device can wake-up from SLEEP through one of the following events:

1. External reset input on \overline{MCLR} pin
2. Watchdog Timer Wake-up (if WDT was enabled)
3. Interrupt from RB0/INT pin, RB Port change, or the Peripheral Interrupt (Comparator).

The WDT is cleared when the device wakes-up from sleep, regardless of the source of wake-up.

FIGURE 10-19: WAKE-UP FROM SLEEP THROUGH INTERRUPT



BTFSS Bit Test f, Skip if Set

Syntax: [*label*] BTFSS f,b

Operands: $0 \leq f \leq 127$
 $0 \leq b < 7$

Operation: skip if (f) = 1

Status Affected: None

Encoding:

01	11bb	bfff	ffff
----	------	------	------

Description: If bit 'b' in register 'f' is '1' then the next instruction is skipped.
 If bit 'b' is '1', then the next instruction fetched during the current instruction execution, is discarded and a NOP is executed instead, making this a two-cycle instruction.

Words: 1

Cycles: 1(2)

Example

```

HERE    BTFSS    FLAG, 1
FALSE   GOTO    PROCESS_CODE
TRUE    •
        •
        •
  
```

Before Instruction

PC = address HERE

After Instruction

```

if FLAG<1> = 0,
PC = address FALSE
if FLAG<1> = 1,
PC = address TRUE
  
```

CALL Call Subroutine

Syntax: [*label*] CALL k

Operands: $0 \leq k \leq 2047$

Operation: (PC)+1 → TOS,
 k → PC<10:0>,
 (PCLATH<4:3>) → PC<12:11>

Status Affected: None

Encoding:

10	0kkk	kkkk	kkkk
----	------	------	------

Description: Call Subroutine. First, return address (PC+1) is pushed onto the stack. The eleven bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a two-cycle instruction.

Words: 1

Cycles: 2

Example

```

HERE    CALL    THERE
  
```

Before Instruction

PC = Address HERE

After Instruction

```

PC = Address THERE
TOS = Address HERE+1
  
```

CLRF Clear f

Syntax: [*label*] CLRF f

Operands: $0 \leq f \leq 127$

Operation: 00h → (f)
 1 → Z

Status Affected: Z

Encoding:

00	0001	1fff	ffff
----	------	------	------

Description: The contents of register 'f' are cleared and the Z bit is set.

Words: 1

Cycles: 1

Example

```

CLRF    FLAG_REG
  
```

Before Instruction

FLAG_REG = 0x5A

After Instruction

```

FLAG_REG = 0x00
Z         = 1
  
```

CLRW Clear W

Syntax: [*label*] CLRW

Operands: None

Operation: 00h → (W)
 1 → Z

Status Affected: Z

Encoding:

00	0001	0000	0011
----	------	------	------

Description: W register is cleared. Zero bit (Z) is set.

Words: 1

Cycles: 1

Example

```

CLRW
  
```

Before Instruction

W = 0x5A

After Instruction

```

W = 0x00
Z = 1
  
```

RETURN Return from Subroutine

Syntax: [*label*] RETURN

Operands: None

Operation: TOS → PC

Status Affected: None

Encoding:

00	0000	0000	1000
----	------	------	------

Description: Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two cycle instruction.

Words: 1

Cycles: 2

Example
RETURN
After Interrupt
PC = TOS

RRF Rotate Right f through Carry

Syntax: [*label*] RRF f,d

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

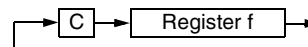
Operation: See description below

Status Affected: C

Encoding:

00	1100	dfff	ffff
----	------	------	------

Description: The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.



Words: 1

Cycles: 1

Example RRF REG1, 0

Before Instruction

REG1 = 1110 0110
C = 0

After Instruction

REG1 = 1110 0110
W = 0111 0011
C = 0

RLF Rotate Left f through Carry

Syntax: [*label*] RLF f,d

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

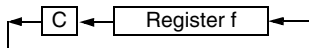
Operation: See description below

Status Affected: C

Encoding:

00	1101	dfff	ffff
----	------	------	------

Description: The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is stored back in register 'f'.



Words: 1

Cycles: 1

Example RLF REG1, 0
Before Instruction
REG1 = 1110 0110
C = 0
After Instruction
REG1 = 1110 0110
W = 1100 1100
C = 1

SLEEP

Syntax: [*label*] SLEEP

Operands: None

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

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

Encoding:

00	0000	0110	0011
----	------	------	------

Description: The power-down status bit, $\overline{\text{PD}}$ is cleared. Time-out status bit, $\overline{\text{TO}}$ is set. Watchdog Timer and its prescaler are cleared. The processor is put into SLEEP mode with the oscillator stopped. See Section 10.8 for more details.

Words: 1

Cycles: 1

Example: SLEEP

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 I²C bus and separate headers for connection to an LCD module and a keypad.

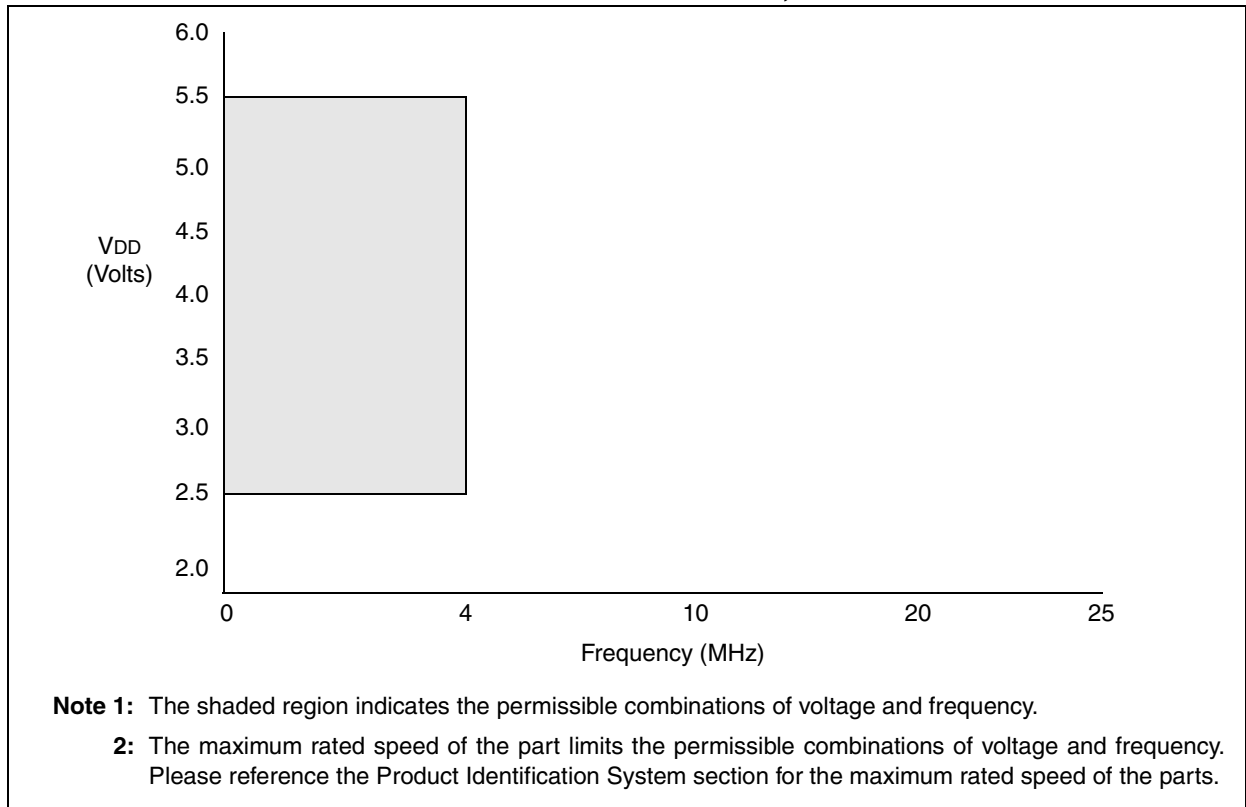
PICDEM-3 Low-Cost PIC16CXXX Demonstration Board

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

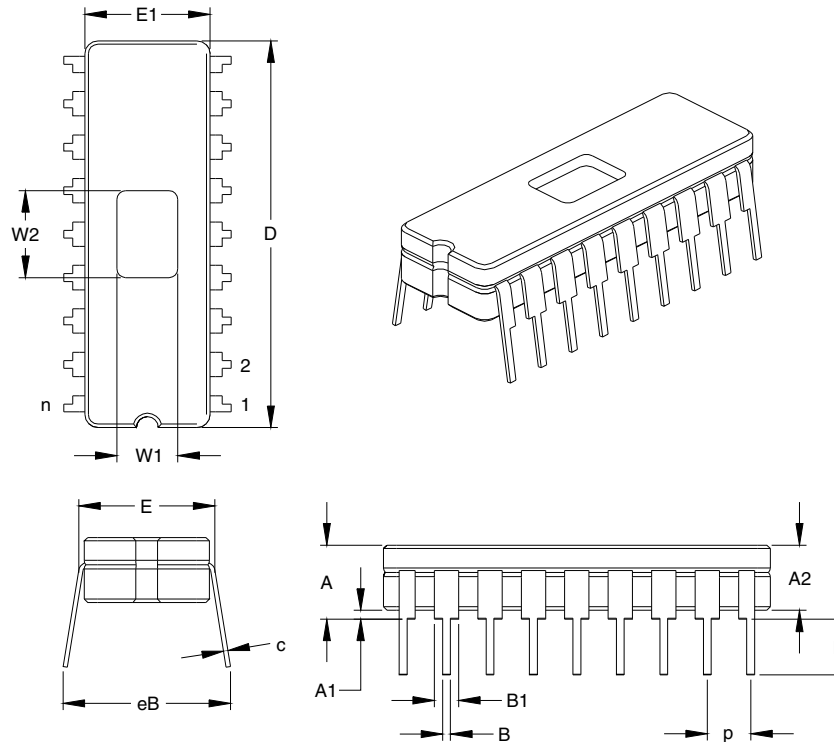
FIGURE 13-3: PIC16LCE62X VOLTAGE-FREQUENCY GRAPH, $-40^{\circ}\text{C} \leq T_A < +125^{\circ}\text{C}$



14.0 PACKAGING INFORMATION

18-Lead Ceramic Dual In-line with Window (JW) – 300 mil (CERDIP)

Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>

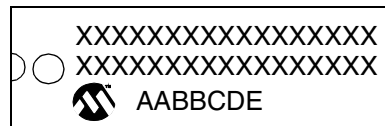


Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		18			18	
Pitch	p		.100			2.54	
Top to Seating Plane	A	.170	.183	.195	4.32	4.64	4.95
Ceramic Package Height	A2	.155	.160	.165	3.94	4.06	4.19
Standoff	A1	.015	.023	.030	0.38	0.57	0.76
Shoulder to Shoulder Width	E	.300	.313	.325	7.62	7.94	8.26
Ceramic Pkg. Width	E1	.285	.290	.295	7.24	7.37	7.49
Overall Length	D	.880	.900	.920	22.35	22.86	23.37
Tip to Seating Plane	L	.125	.138	.150	3.18	3.49	3.81
Lead Thickness	c	.008	.010	.012	0.20	0.25	0.30
Upper Lead Width	B1	.050	.055	.060	1.27	1.40	1.52
Lower Lead Width	B	.016	.019	.021	0.41	0.47	0.53
Overall Row Spacing	eB	.345	.385	.425	8.76	9.78	10.80
Window Width	W1	.130	.140	.150	3.30	3.56	3.81
Window Length	W2	.190	.200	.210	4.83	5.08	5.33

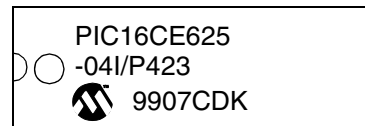
*Controlling Parameter
JEDEC Equivalent: MO-036
Drawing No. C04-010

14.1 Package Marking Information

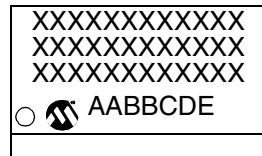
18-Lead PDIP



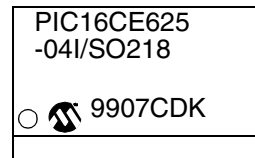
Example



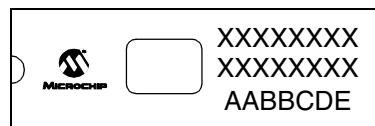
18-Lead SOIC (.300")



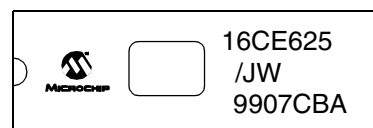
Example



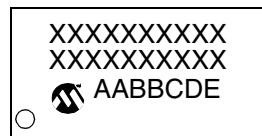
18-Lead CERDIP Windowed



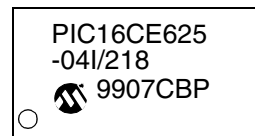
Example



20-Lead SSOP



Example



Legend: XX...X Customer-specific information
Y Year code (last digit of calendar year)
YY Year code (last 2 digits of calendar year)
WW Week code (week of January 1 is week '01')
NNN Alphanumeric traceability code
(e3) Pb-free JEDEC designator for Matte Tin (Sn)
* This package is Pb-free. The Pb-free JEDEC designator (e3) can be found on the outer packaging for this package.

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

PIC16CE62X

NOTES:

APPENDIX A: CODE FOR ACCESSING EEPROM DATA MEMORY

Please check our web site at www.microchip.com for code availability.

APPENDIX B: REVISION HISTORY

Revision D (January 2013)

Added a note to each package outline drawing.