



# PIC16C62X

## EPROM-Based 8-Bit CMOS Microcontroller

### Devices included in this data sheet:

- PIC16C620
- PIC16C621
- PIC16C622

### High Performance RISC CPU:

- Only 35 instructions to learn
- All single-cycle instructions (200 ns), except for program branches which are two-cycle
- Operating speed:
  - DC - 20 MHz clock input
  - DC - 200 ns instruction cycle

Device	Program Memory	Data Memory
PIC16C620	512	80
PIC16C621	1K	80
PIC16C622	2K	128

- Interrupt capability
- 16 special function hardware registers
- 8-level deep hardware stack
- Direct, Indirect and Relative addressing modes

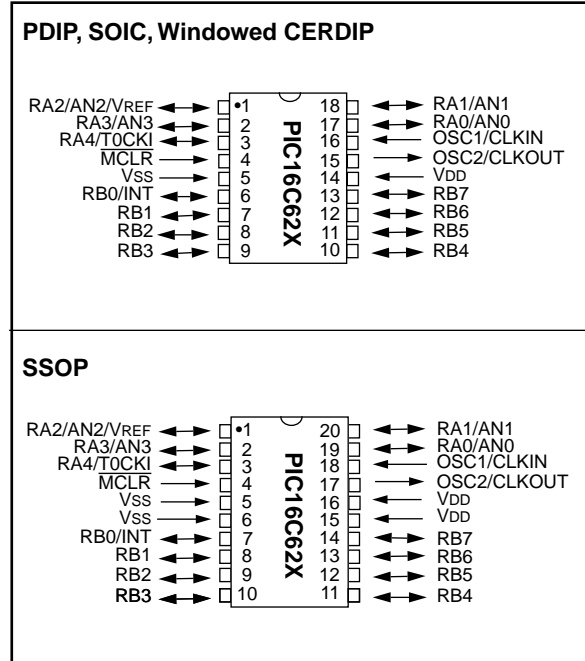
### Peripheral Features:

- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
- Analog comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (VREF) module
  - Programmable input multiplexing from device inputs and internal voltage reference
  - Comparator outputs can be output signals
- Timer0: 8-bit timer/counter with 8-bit programmable prescaler

### Special Microcontroller Features:

- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Brown-out Reset
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation

### Pin Diagram



### Special Microcontroller Features (cont'd)

- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Serial in-circuit programming (via two pins)
- Four user programmable ID locations

### CMOS Technology:

- Low-power, high-speed CMOS EPROM technology
- Fully static design
- Wide operating voltage range
  - 2.5V to 6.0V
- Commercial and industrial temperature range
- Low power consumption
  - < 2.0 mA @ 5.0V, 4.0 MHz
  - 15 μA typical @ 3.0V, 32 kHz
  - < 1.0 μA typical standby current @ 3.0V

# PIC16C62X

---

---

## Table of Contents

1.0	General Description .....	3
2.0	PIC16C62X Device Varieties.....	5
3.0	Architectural Overview.....	7
4.0	Memory Organization .....	11
5.0	I/O Ports.....	23
6.0	Timer0 Module.....	29
7.0	Comparator Module .....	35
8.0	Voltage Reference Module .....	41
9.0	Special Features of the CPU .....	43
10.0	Instruction Set Summary .....	59
11.0	Development Support.....	71
12.0	Electrical Specifications.....	77
13.0	Device Characterization Information.....	89
14.0	Packaging Information.....	91
Appendix A:	Enhancements.....	97
Appendix B:	Compatibility .....	97
Appendix C:	What's New.....	98
Appendix D:	What's Changed .....	98
Appendix E:	PIC16/17 Microcontrollers .....	99
PIC14XXX	Family of Devices.....	99
PIC16C5X	Family of Devices .....	100
PIC16C62X	Family of Devices .....	101
PIC16C6X	Family of Devices .....	102
PIC16C7X	Family of Devices .....	103
PIC16C8X	Family of Devices .....	104
PIC17CXX	Family of Devices.....	105
Pin Compatibility.....		106
Index.....		107
Connecting to Microchip BBS.....		111
Reader Response .....		112

## To Our Valued Customers

We constantly strive to improve the quality of all our products and documentation. We have spent an exceptional amount of time to ensure that these documents are correct. However, we realize that we may have missed a few things. If you find any information that is missing or appears in error from the previous version of this data sheet (PIC16C62X Data Sheet, Literature Number DS30235D), please use the reader response form in the back of this data sheet to inform us. We appreciate your assistance in making this a better document.

## 1.0 GENERAL DESCRIPTION

The PIC16C62X are 18-Pin EPROM-based members of the versatile PIC16CXX family of low-cost, high-performance, CMOS, fully-static, 8-bit microcontrollers.

All PIC16/17 microcontrollers employ an advanced RISC architecture. The PIC16C62X have enhanced core features, eight-level deep stack, and multiple internal and external interrupt sources. The separate instruction and data buses of the Harvard architecture allow a 14-bit wide instruction word with the separate 8-bit wide data. The two-stage instruction pipeline allows all instructions to execute in a single-cycle, except for program branches (which require two cycles). A total of 35 instructions (reduced instruction set) are available. Additionally, a large register set gives some of the architectural innovations used to achieve a very high performance.

PIC16C62X microcontrollers typically achieve a 2:1 code compression and a 4:1 speed improvement over other 8-bit microcontrollers in its class.

The PIC16C620 and PIC16C621 have 80 bytes of RAM. The PIC16C622 has 128 bytes of RAM. Each device has 13 I/O pins and an 8-bit timer/counter with an 8-bit programmable prescaler. In addition, the PIC16C62X add two analog comparators with a programmable on-chip voltage reference module. The comparator module is ideally suited for applications requiring a low-cost analog interface (e.g., battery chargers, threshold detectors, white goods controllers, etc).

PIC16C62X devices have special features to reduce external components, thus reducing cost, enhancing system reliability and reducing power consumption. There are four oscillator options, of which the single pin RC oscillator provides a low-cost solution, the LP oscillator minimizes power consumption, XT is a standard crystal, and the HS is for High Speed crystals. The SLEEP (power-down) mode offers power saving. The user can wake up the chip from SLEEP through several external and internal interrupts and reset.

A highly reliable Watchdog Timer with its own on-chip RC oscillator provides protection against software lock-up.

A UV-erasable CERDIP-packaged version is ideal for code development while the cost-effective One-Time Programmable (OTP) version is suitable for production in any volume.

Table 1-1 shows the features of the PIC16C62X mid-range microcontroller families.

A simplified block diagram of the PIC16C62X is shown in Figure 3-1.

The PIC16C62X series fit perfectly in applications ranging from battery chargers to low-power remote sensors. The EPROM technology makes customization of application programs (detection levels, pulse generation, timers, etc.) extremely fast and convenient. The small footprint packages make this microcontroller series perfect for all applications with space limitations. Low-cost, low-power, high-performance, ease of use and I/O flexibility make the PIC16C62X very versatile.

### 1.1 Family and Upward Compatibility

Those users familiar with the PIC16C5X family of microcontrollers will realize that this is an enhanced version of the PIC16C5X architecture. Please refer to Appendix A for a detailed list of enhancements. Code written for PIC16C5X can be easily ported to PIC16C62X family of devices (Appendix B).

### 1.2 Development Support

The PIC16C62X family is supported by a full-featured macro assembler, a software simulator, an in-circuit emulator, a low-cost development programmer and a full-featured programmer. A "C" compiler and fuzzy logic support tools are also available.

# PIC16C62X

TABLE 1-1: PIC16C62X FAMILY OF DEVICES

	Memory			Clock				Peripherals			Features	
	Maximum Frequency of Operation (MHz)	Program Memory	Data Memory (bytes)	EPROM	Timer Module(s)	Comparator(s)	Internal Reference Voltage	Interrupt Sources	I/O Pins	Voltage Range (Volts)	Brown-out Reset	Packages
PIC16C620	20	512	80	TMR0	2	Yes	4	13	2.5-6.0	Yes	18-pin DIP, SOIC; 20-pin SSOP	
PIC16C621	20	1K	80	TMR0	2	Yes	4	13	2.5-6.0	Yes	18-pin DIP, SOIC; 20-pin SSOP	
PIC16C622	20	2K	128	TMR0	2	Yes	4	13	2.5-6.0	Yes	18-pin DIP, SOIC; 20-pin SSOP	

All PIC16/17 Family devices have Power-on Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability.

All PIC16C62X Family devices use serial programming with clock pin RB6 and data pin RB7.

## 2.0 PIC16C62X DEVICE VARIETIES

A variety of frequency ranges and packaging options are available. Depending on application and production requirements the proper device option can be selected using the information in the PIC16C62X Product Identification System section at the end of this data sheet. When placing orders, please use this page of the data sheet to specify the correct part number.

### 2.1 UV Erasable Devices

The UV erasable version, offered in CERDIP package is optimal for prototype development and pilot programs. This version can be erased and reprogrammed to any of the oscillator modes.

Microchip's PICSTART<sup>®</sup> and PRO MATE<sup>™</sup> programmers both support programming of the PIC16C62X.

### 2.2 One-Time-Programmable (OTP) Devices

The availability of OTP devices is especially useful for customers who need the flexibility for frequent code updates and small volume applications. In addition to the program memory, the configuration bits must also be programmed.

### 2.3 Quick-Turnaround-Production (QTP) Devices

Microchip offers a QTP Programming Service for factory production orders. This service is made available for users who chose not to program a medium to high quantity of units and whose code patterns have stabilized. The devices are identical to the OTP devices but with all EPROM locations and configuration options already programmed by the factory. Certain code and prototype verification procedures apply before production shipments are available. Please contact your Microchip Technology sales office for more details.

### 2.4 Serialized Quick-Turnaround-Production (SQTP<sup>SM</sup>) Devices

Microchip offers a unique programming service where a few user-defined locations in each device are programmed with different serial numbers. The serial numbers may be random, pseudo-random or sequential.

Serial programming allows each device to have a unique number which can serve as an entry-code, password or ID number.

# PIC16C62X

---

NOTES:

## 3.0 ARCHITECTURAL OVERVIEW

The high performance of the PIC16C62X family can be attributed to a number of architectural features commonly found in RISC microprocessors. To begin with, the PIC16C62X uses a Harvard architecture, in which, program and data are accessed from separate memories using separate busses. This improves bandwidth over traditional von Neumann architecture where program and data are fetched from the same memory. Separating program and data memory further allows instructions to be sized differently than 8-bit wide data word. Instruction opcodes are 14-bits wide making it possible to have all single word instructions. A 14-bit wide program memory access bus fetches a 14-bit instruction in a single cycle. A two-stage pipeline overlaps fetch and execution of instructions. Consequently, all instructions (35) execute in a single-cycle (200 ns @ 20 MHz) except for program branches.

The PIC16C620 addresses 512 x 14 on-chip program memory. The PIC16C621 addresses 1K x 14 program memory. The PIC16C622 addresses 2K x 14 program memory. All program memory is internal.

The PIC16C62X can directly or indirectly address its register files or data memory. All special function registers including the program counter are mapped in the data memory. The PIC16C62X have an orthogonal (symmetrical) instruction set that makes it possible to carry out any operation on any register using any addressing mode. This symmetrical nature and lack of 'special optimal situations' make programming with the PIC16C62X simple yet efficient. In addition, the learning curve is reduced significantly.

The PIC16C62X devices contain an 8-bit ALU and working register. The ALU is a general purpose arithmetic unit. It performs arithmetic and Boolean functions between data in the working register and any register file.

The ALU is 8-bit wide and capable of addition, subtraction, shift and logical operations. Unless otherwise mentioned, arithmetic operations are two's complement in nature. In two-operand instructions, typically one operand is the working register (W register). The other operand is a file register or an immediate constant. In single operand instructions, the operand is either the W register or a file register.

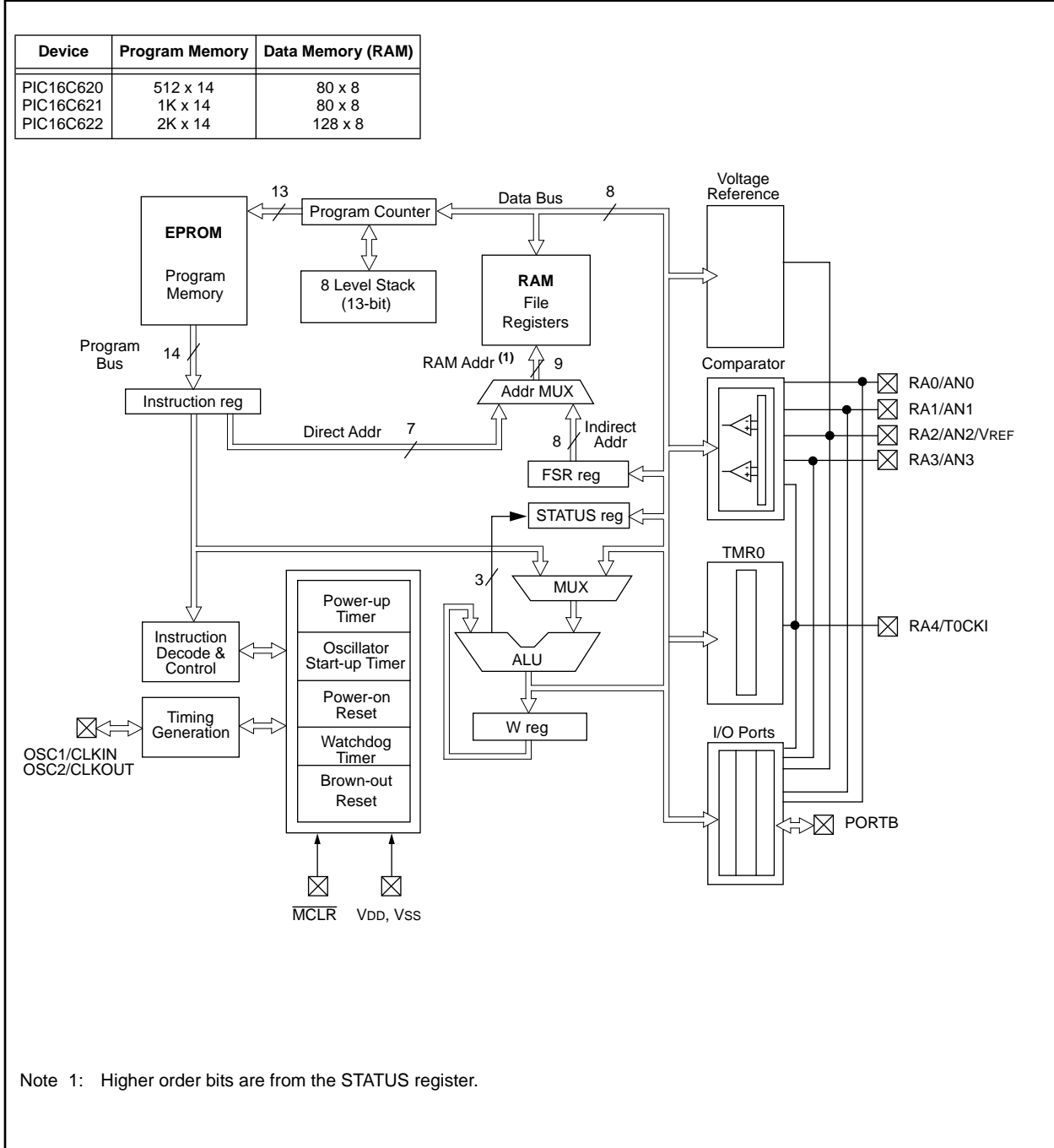
The W register is an 8-bit working register used for ALU operations. It is not an addressable register.

Depending on the instruction executed, the ALU may affect the values of the Carry (C), Digit Carry (DC), and Zero (Z) bits in the STATUS register. The C and DC bits operate as a  $\overline{\text{Borrow}}$  and  $\overline{\text{Digit Borrow}}$  out bit, respectively, bit in subtraction. See the SUBLW and SUBWF instructions for examples.

A simplified block diagram is shown in Figure 3-1, with a description of the device pins in Table 3-1.

# PIC16C62X

**FIGURE 3-1: BLOCK DIAGRAM**





**TABLE 3-1: PIC16C62X PINOUT DESCRIPTION**

Name	DIP SOIC Pin #	SSOP Pin #	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	16	18	I	ST/CMOS	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	17	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
MCLR/VPP	4	4	I/P	ST	Master clear (reset) input/programming voltage input. This pin is an active low reset to the device.
RA0/AN0	17	19	I/O	ST	PORTA is a bi-directional I/O port. Analog comparator input Analog comparator input Analog comparator input or VREF output Analog comparator input /output Can be selected to be the clock input to the Timer0 timer/counter or a comparator output. Output is open drain type.
RA1/AN1	18	20	I/O	ST	
RA2/AN2/VREF	1	1	I/O	ST	
RA3/AN3	2	2	I/O	ST	
RA4/TOCKI	3	3	I/O	ST	
RB0/INT	6	7	I/O	TTL/ST <sup>(1)</sup>	PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs. RB0/INT can also be selected as an external interrupt pin.  Interrupt on change pin. Interrupt on change pin. Interrupt on change pin. Serial programming clock. Interrupt on change pin. Serial programming data.
RB1	7	8	I/O	TTL	
RB2	8	9	I/O	TTL	
RB3	9	10	I/O	TTL	
RB4	10	11	I/O	TTL	
RB5	11	12	I/O	TTL	
RB6	12	13	I/O	TTL/ST <sup>(2)</sup>	
RB7	13	14	I/O	TTL/ST <sup>(2)</sup>	
Vss	5	5,6	P	—	Ground reference for logic and I/O pins.
VDD	14	15,16	P	—	Positive supply for logic and I/O pins.

Legend:            O = output                            I/O = input/output            P = power  
                      — = Not used                        I = Input                        ST = Schmitt Trigger input  
                      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.

# PIC16C62X

## 3.1 Clocking Scheme/Instruction Cycle

The clock input (from OSC1) is internally divided by four to generate four non-overlapping quadrature clocks namely Q1, Q2, Q3 and Q4. Internally, the program counter (PC) is incremented every Q1, the instruction is fetched from the program memory and latched into the instruction register in Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow is shown in Figure 3-2.

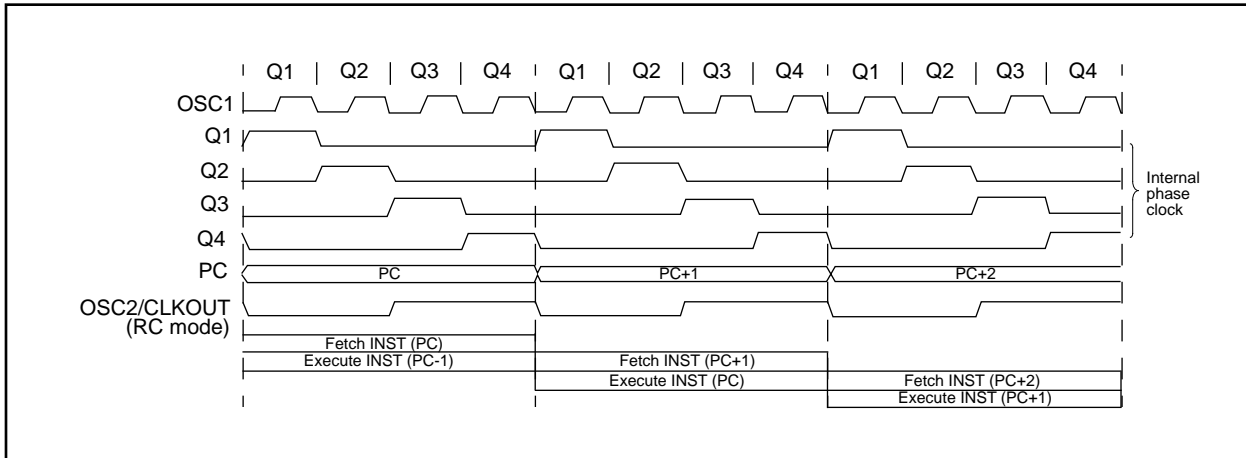
## 3.2 Instruction Flow/Pipelining

An "Instruction Cycle" consists of four Q cycles (Q1, Q2, Q3 and Q4). The instruction fetch and execute are pipelined such that fetch takes one instruction cycle while decode and execute takes another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., GOTO) then two cycles are required to complete the instruction (Example 3-1).

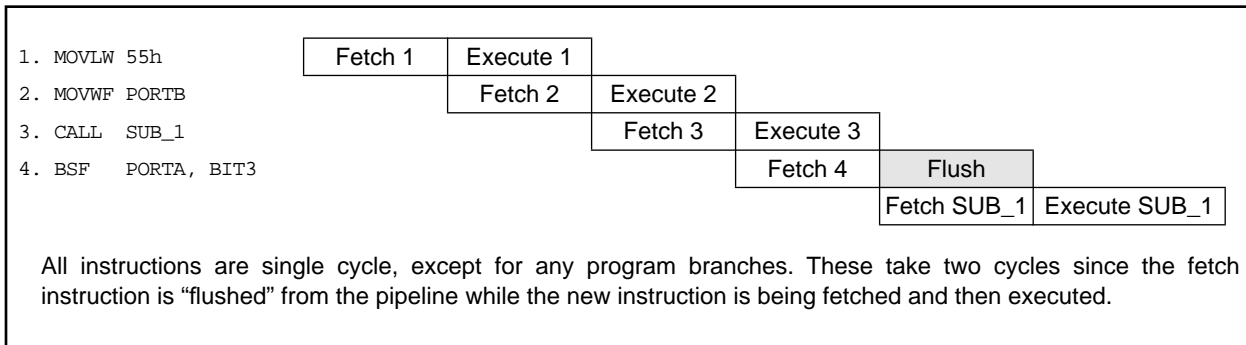
A fetch cycle begins with the program counter (PC) incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the "Instruction Register (IR)" in cycle Q1. This instruction is then decoded and executed during the Q2, Q3, and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

**FIGURE 3-2: CLOCK/INSTRUCTION CYCLE**



**EXAMPLE 3-1: INSTRUCTION PIPELINE FLOW**

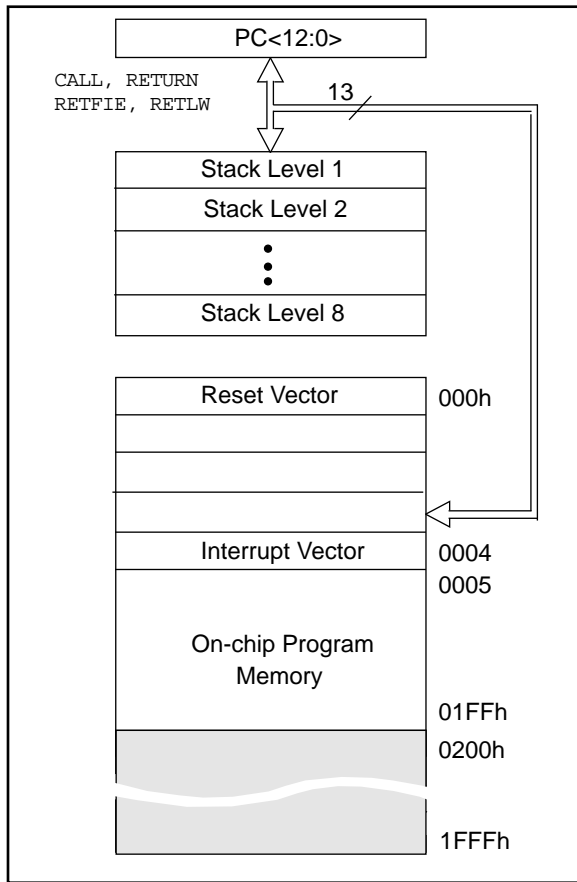


## 4.0 MEMORY ORGANIZATION

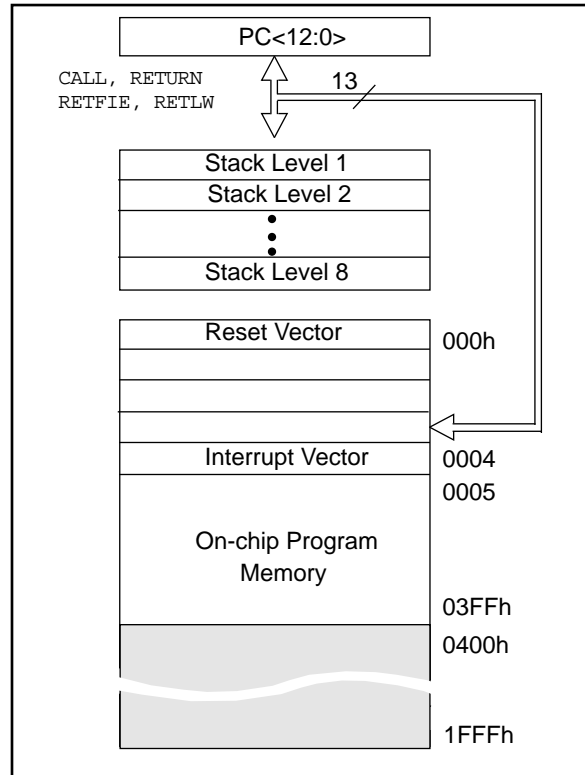
### 4.1 Program Memory Organization

The PIC16C62X has a 13-bit program counter capable of addressing an 8K x 14 program memory space. Only the first 512 x 14 (0000h - 01FFh) for the PIC16C620, 1K x 14 (0000h - 03FFh) for the PIC16C621 and 2K x 14 (0000h - 07FFh) for the PIC16C622 are physically implemented. Accessing a location above these boundaries will cause a wrap-around within the first 512 x 14 space (PIC16C620) or 1K x 14 space (PIC16C621) or 2K x 14 space (PIC16C622). The reset vector is at 0000h and the interrupt vector is at 0004h (Figure 4-1, Figure 4-2, Figure 4-3).

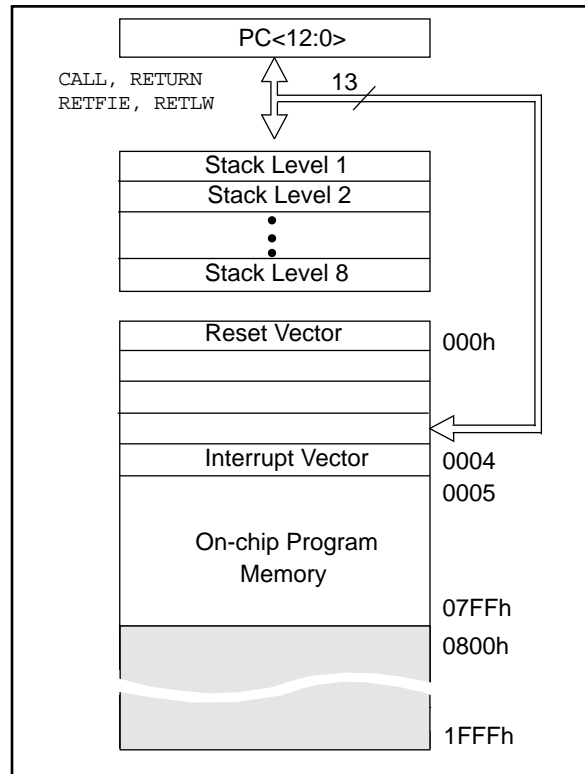
**FIGURE 4-1: PROGRAM MEMORY MAP AND STACK FOR THE PIC16C620**



**FIGURE 4-2: PROGRAM MEMORY MAP AND STACK FOR THE PIC16C621**



**FIGURE 4-3: PROGRAM MEMORY MAP AND STACK FOR THE PIC16C622**



# PIC16C62X

---

## 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 20-6Fh (Bank0) on the PIC16C620/621 and 20-7Fh (Bank0) and A0-BFh (Bank1) on the PIC16C622 are general purpose registers implemented as static RAM. Some special purpose registers are mapped in Bank 1.

### 4.2.1 GENERAL PURPOSE REGISTER FILE

The register file is organized as 80 x 8 in the PIC16C620/621 and 128 x 8 in the PIC16C622. Each is accessed either directly or indirectly through the File Select Register FSR (Section 4.4).

**FIGURE 4-4: DATA MEMORY MAP FOR THE PIC16C620/621**

File Address			File Address
00h	INDF <sup>(1)</sup>	INDF <sup>(1)</sup>	80h
01h	TMR0	OPTION	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h			87h
08h			88h
09h			89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch	PIR1	PIE1	8Ch
0Dh			8Dh
0Eh		PCON	8Eh
0Fh			8Fh
10h			90h
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	VRCON	9Fh
20h	General Purpose Register		A0h
6Fh			
70h			
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 PIC16C622**

File Address			File Address	
00h	INDF <sup>(1)</sup>	INDF <sup>(1)</sup>	80h	
01h	TMR0	OPTION	81h	
02h	PCL	PCL	82h	
03h	STATUS	STATUS	83h	
04h	FSR	FSR	84h	
05h	PORTA	TRISA	85h	
06h	PORTB	TRISB	86h	
07h			87h	
08h			88h	
09h			89h	
0Ah	PCLATH	PCLATH	8Ah	
0Bh	INTCON	INTCON	8Bh	
0Ch	PIR1	PIE1	8Ch	
0Dh			8Dh	
0Eh		PCON	8Eh	
0Fh			8Fh	
10h			90h	
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	VRCON	9Fh	
20h	General Purpose Register	General Purpose Register	A0h	
				BFh
				C0h
7Fh			FFh	

Bank 0                      Bank 1

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

# PIC16C62X

## 4.2.2 SPECIAL FUNCTION REGISTERS

The special function registers are registers used by the CPU and Peripheral functions for controlling the desired operation of the device (Table 4-1). These registers are static RAM.

The special registers can be classified into two sets (core and peripheral). The special function registers associated with the “core” functions are described in this section. Those related to the operation of the peripheral features are described in the section of that peripheral feature.

**TABLE 4-1: SPECIAL REGISTERS FOR THE PIC16C62X**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR/BOR Reset	Value on all other resets <sup>(1)</sup>
<b>Bank 0</b>											
00h	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								xxxx xxxx	xxxx xxxx
01h	TMR0	Timer0 Module's Register								xxxx xxxx	uuuu uuuu
02h	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	0000 0000
03h	STATUS	IRP <sup>(2)</sup>	RP1 <sup>(2)</sup>	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
04h	FSR	Indirect data memory address pointer								xxxx xxxx	uuuu uuuu
05h	PORTA	—	—	—	RA4	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
07h	Unimplemented									—	—
08h	Unimplemented									—	—
09h	Unimplemented									—	—
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of program counter				---0 0000	---0 0000	
0Bh	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBF	0000 000x	0000 000x
0Ch	PIR1	—	CMIF	—	—	—	—	—	—	-0-- ----	-0-- ----
0Dh-1Eh	Unimplemented									—	—
1Fh	CMCON	C2OUT	C1OUT	—	—	CIS	CM2	CM1	CM0	00-- 0000	00-- 0000
<b>Bank 1</b>											
80h	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								xxxx xxxx	xxxx xxxx
81h	OPTION	$\overline{RBPU}$	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
82h	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	0000 0000
83h	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
84h	FSR	Indirect data memory address pointer								xxxx xxxx	uuuu uuuu
85h	TRISA	—	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111
87h	Unimplemented									—	—
88h	Unimplemented									—	—
89h	Unimplemented									—	—
8Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of program counter				---0 0000	---0 0000	
8Bh	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBF	0000 000x	0000 000x
8Ch	PIE1	—	CMIE	—	—	—	—	—	—	-0-- ----	-0-- ----
8Dh	Unimplemented									—	—
8Eh	PCON	—	—	—	—	—	—	POR	$\overline{BOR}$	---- --0x	---- --nq
8Fh-9Eh	Unimplemented									—	—
9Fh	VRCON	VREN	VROE	VRR	—	VR3	VR2	VR1	VR0	000- 0000	000- 0000

Legend: — = Unimplemented locations read as '0', u = unchanged, x = unknown, q = value depends on condition, shaded = unimplemented

**Note 1:** Other (non power-up) resets include  $\overline{MCLR}$  reset, Brown-out Reset and Watchdog Timer Reset during normal operation.

**Note 2:** IRP & RPI bits are reserved, always maintain these bits clear.

## 4.2.2.1 STATUS REGISTER

The STATUS register, shown in Figure 4-6, 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.

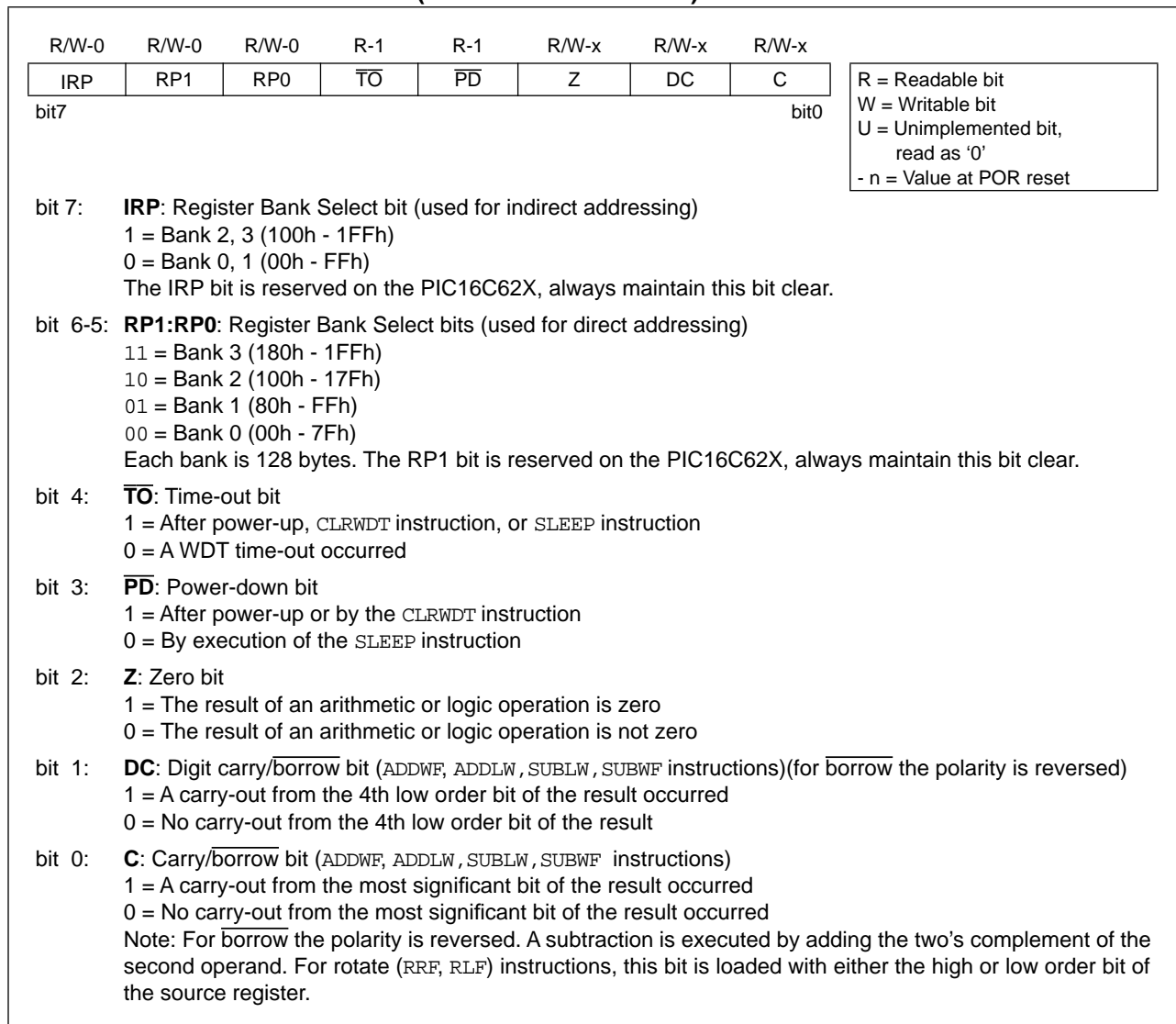
For example, `CLRF STATUS` will clear the upper-three bits and set the Z bit. This leaves the status register as `000uu1uu` (where `u` = unchanged).

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 PIC16C62X 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.

**FIGURE 4-6: STATUS REGISTER (ADDRESS 03H OR 83H)**



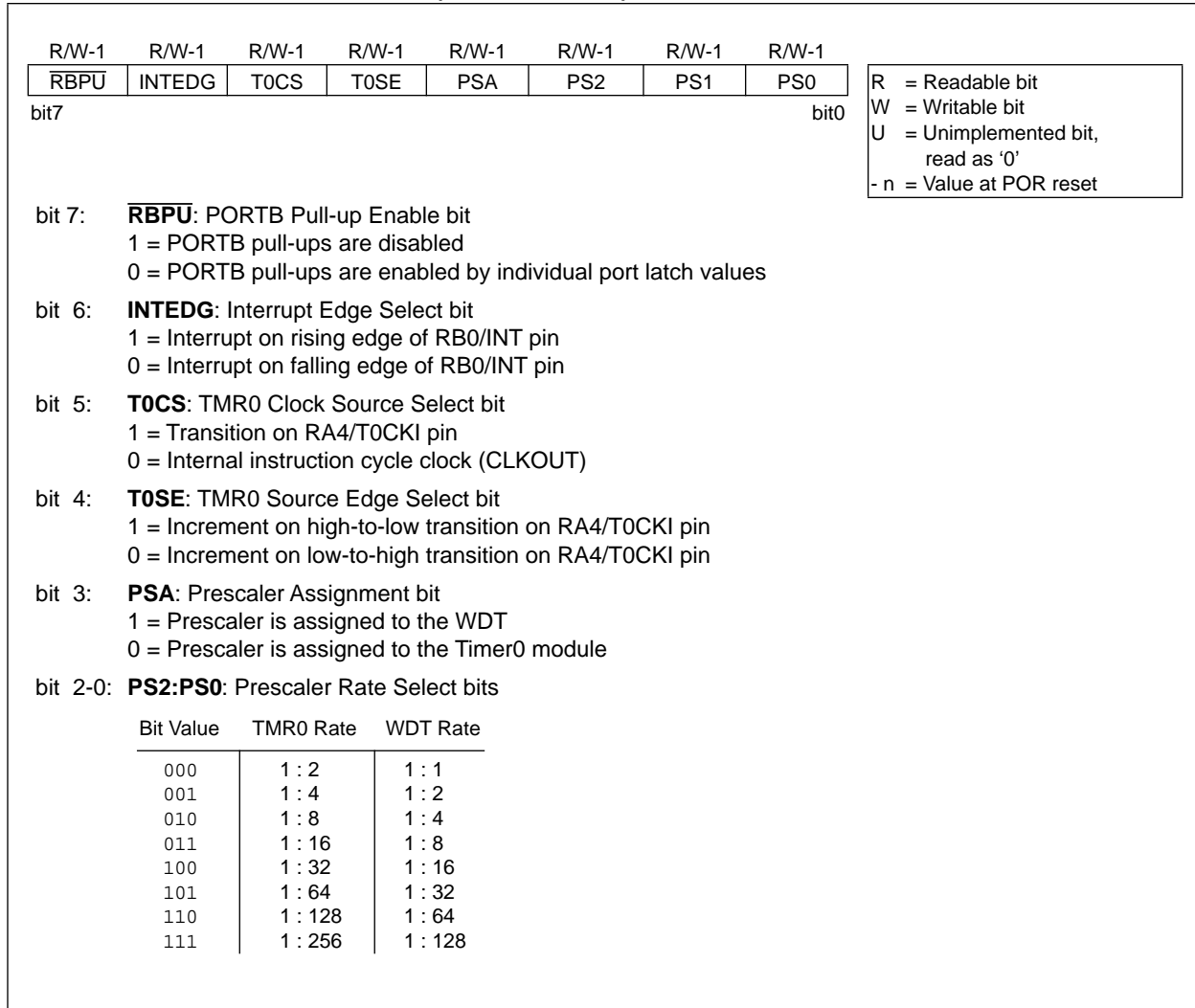
# PIC16C62X

## 4.2.2.2 OPTION REGISTER

The OPTION register is a readable and writable register which contains various control bits to configure the TMR0/WDT prescaler, the external RB0/INT interrupt, TMR0, and the weak pull-ups on PORTB.

**Note:** To achieve a 1:1 prescaler assignment for TMR0, assign the prescaler to the WDT (PSA = 1).

**FIGURE 4-7: OPTION REGISTER (ADDRESS 81H)**



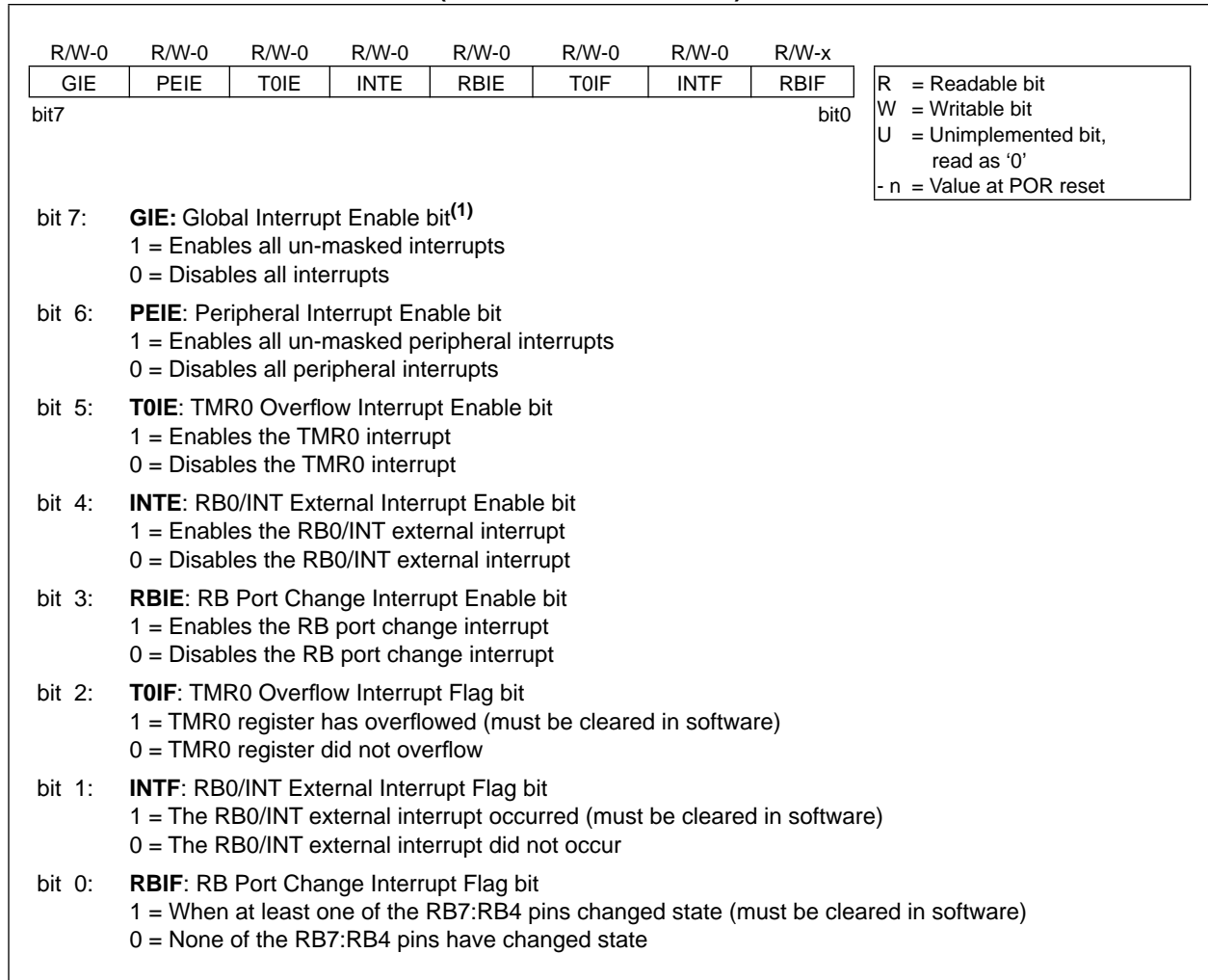


## 4.2.2.3 INTCON REGISTER

The INTCON register is a readable and writable register which contains the various enable and flag bits for all interrupt sources except the comparator module. See Section 4.2.2.4 and Section 4.2.2.5 for a description of the comparator enable and flag bits.

**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>).

**FIGURE 4-8: INTCON REGISTER (ADDRESS 0BH OR 8BH)**

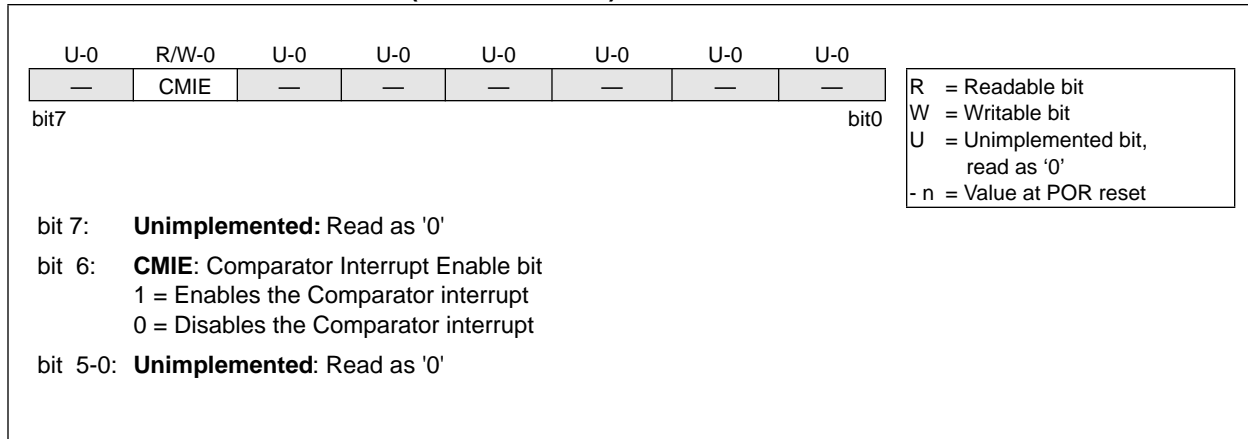


# PIC16C62X

## 4.2.2.4 PIE1 REGISTER

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

**FIGURE 4-9: PIE1 REGISTER (ADDRESS 8CH)**

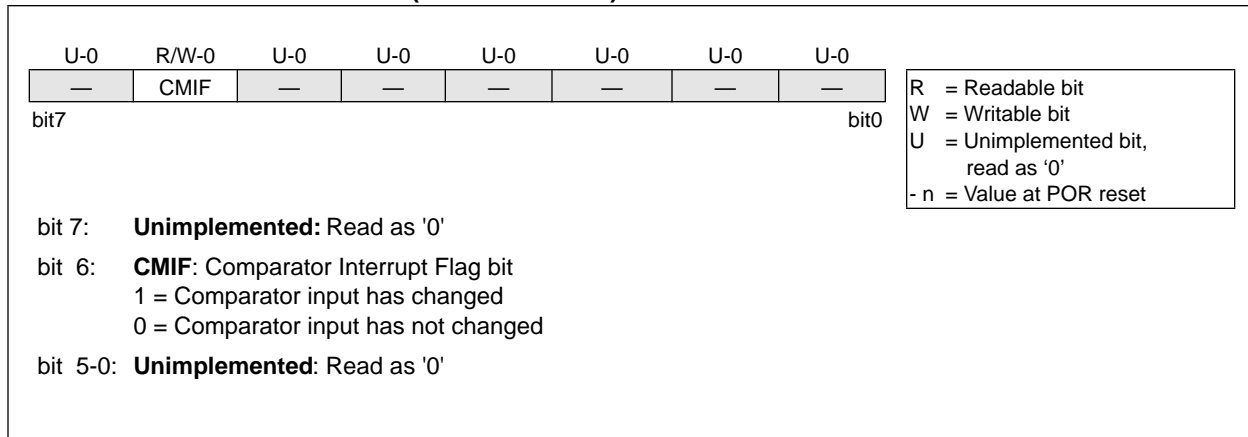


## 4.2.2.5 PIR1 REGISTER

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.

**FIGURE 4-10: PIR1 REGISTER (ADDRESS 0CH)**

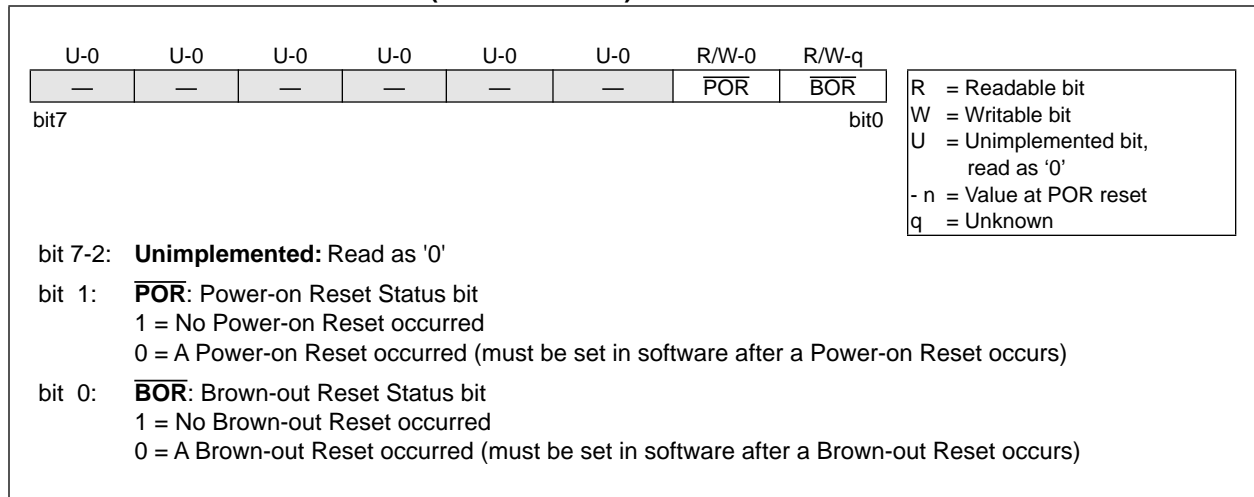


## 4.2.2.6 PCON REGISTER

The PCON register contains flag bits to differentiate between a Power-on Reset, an external  $\overline{MCLR}$  reset, WDT reset or a Brown-out Reset.

**Note:**  $\overline{BOR}$  is unknown on Power-on Reset. It must then be set by the user and checked on subsequent resets to see if  $\overline{BOR}$  is cleared, indicating a brown-out has occurred. The  $\overline{BOR}$  status bit is a "don't care" and is not necessarily predictable if the brown-out circuit is disabled (by programming BODEN bit in the Configuration word).

**FIGURE 4-11: PCON REGISTER (ADDRESS 8Eh)**

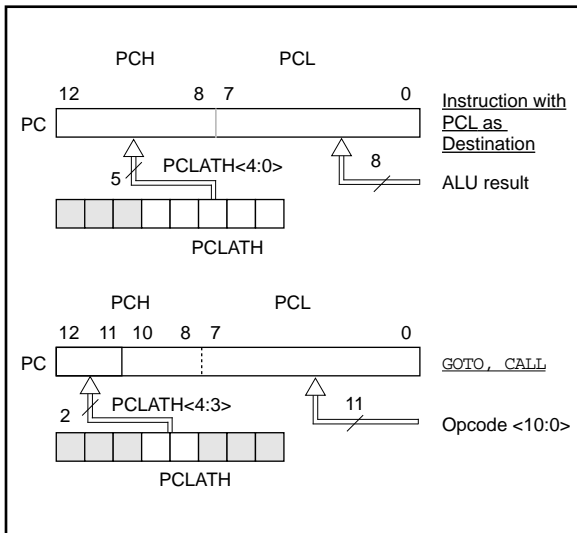


# PIC16C62X

## 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 4-12 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 4-12: 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 PIC16C62X 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 instructions 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.

## 4.4 Indirect Addressing, INDF and FSR Registers

The INDF register is not a physical register. Addressing the INDF register will cause indirect addressing.

Indirect addressing is possible by using the INDF register. Any instruction using the INDF register actually accesses data pointed to by the file select register (FSR). Reading INDF itself indirectly will produce 00h. Writing to the INDF register indirectly results in a no-operation (although status bits may be affected). An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (STATUS<7>), as shown in Figure 4-13. However, IRP is not used in the PIC16C62X.

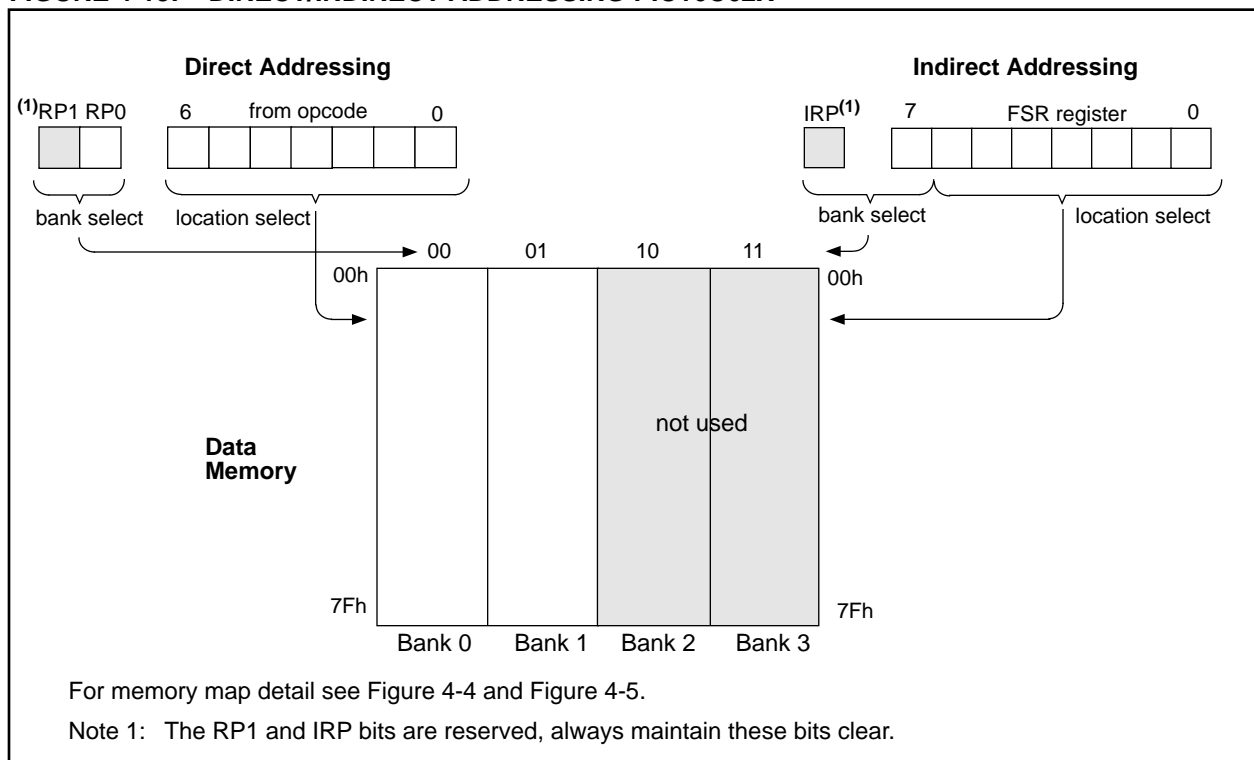
A simple program to clear RAM location 20h-2Fh using indirect addressing is shown in Example 4-1.

### EXAMPLE 4-1: INDIRECT ADDRESSING

```

movlw 0x20 ;initialize pointer
movwf FSR ;to RAM
NEXT   clrf INDF ;clear INDF register
       incf FSR ;inc pointer
       btfss FSR,4 ;all done?
       goto NEXT ;no clear next
                               ;yes continue
CONTINUE:
    
```

**FIGURE 4-13: DIRECT/INDIRECT ADDRESSING PIC16C62X**



# PIC16C62X

---

NOTES:

## 5.0 I/O PORTS

The PIC16C62X have two ports, PORTA and PORTB. Some pins for these I/O ports are multiplexed with an alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

### 5.1 PORTA and TRISA Registers

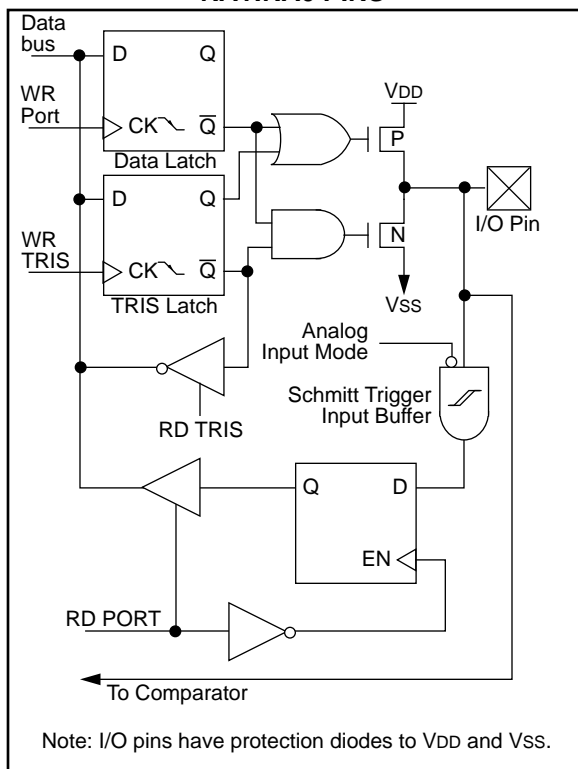
PORTA is a 5-bit wide latch. RA4 is a Schmitt Trigger input and an open drain output. Port RA4 is multiplexed with the T0CKI clock input. All other RA port pins have Schmitt Trigger input levels and full CMOS output drivers. All pins have data direction bits (TRIS registers) which can configure these pins as input or output.

A '1' in the TRISA register puts the corresponding output driver in a hi-impedance mode. A '0' in the TRISA register puts the contents of the output latch on the selected pin(s).

Reading the PORTA register reads the status of the pins whereas writing to it will write to the port latch. All write operations are read-modify-write operations. So a write to a port implies that the port pins are first read, then this value is modified and written to the port data latch.

The PORTA pins are multiplexed with comparator and voltage reference functions. The operation of these pins are selected by control bits in the CMCON (comparator control register) register and the VRCON (voltage reference control register) register. When selected as a comparator input, these pins will read as '0's.

**FIGURE 5-1: BLOCK DIAGRAM OF RA1:RA0 PINS**



**Note:** On reset, the TRISA register is set to all inputs. The digital inputs are disabled and the comparator inputs are forced to ground to reduce excess current consumption.

TRISA controls the direction of the RA pins, even when they are being used as comparator inputs. The user must make sure to keep the pins configured as inputs when using them as comparator inputs.

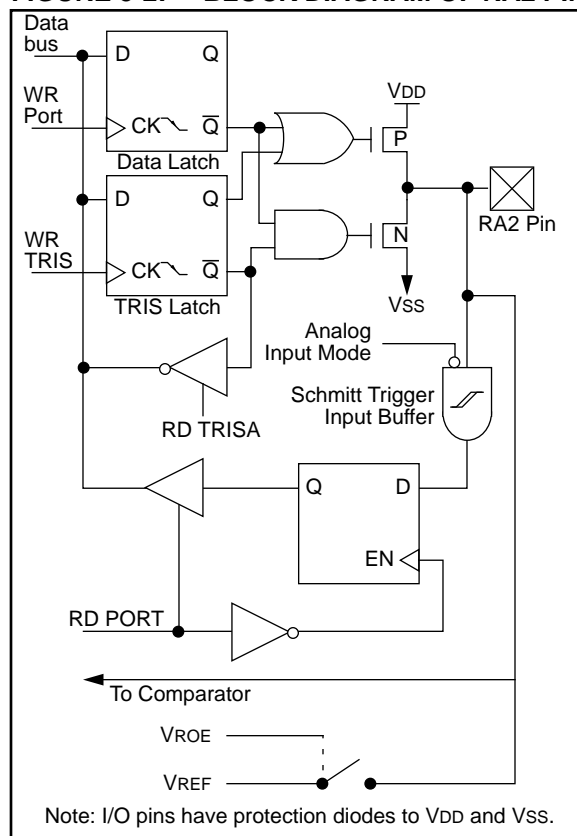
The RA2 pin will also function as the output for the voltage reference. When in this mode, the VREF pin is a very high impedance output. The user must configure TRISA<2> bit as an input and use high impedance loads.

In one of the comparator modes defined by the CMCON register, pins RA3 and RA4 become outputs of the comparators. The TRISA<4:3> bits must be cleared to enable outputs to use this function.

### EXAMPLE 5-1: INITIALIZING PORTA

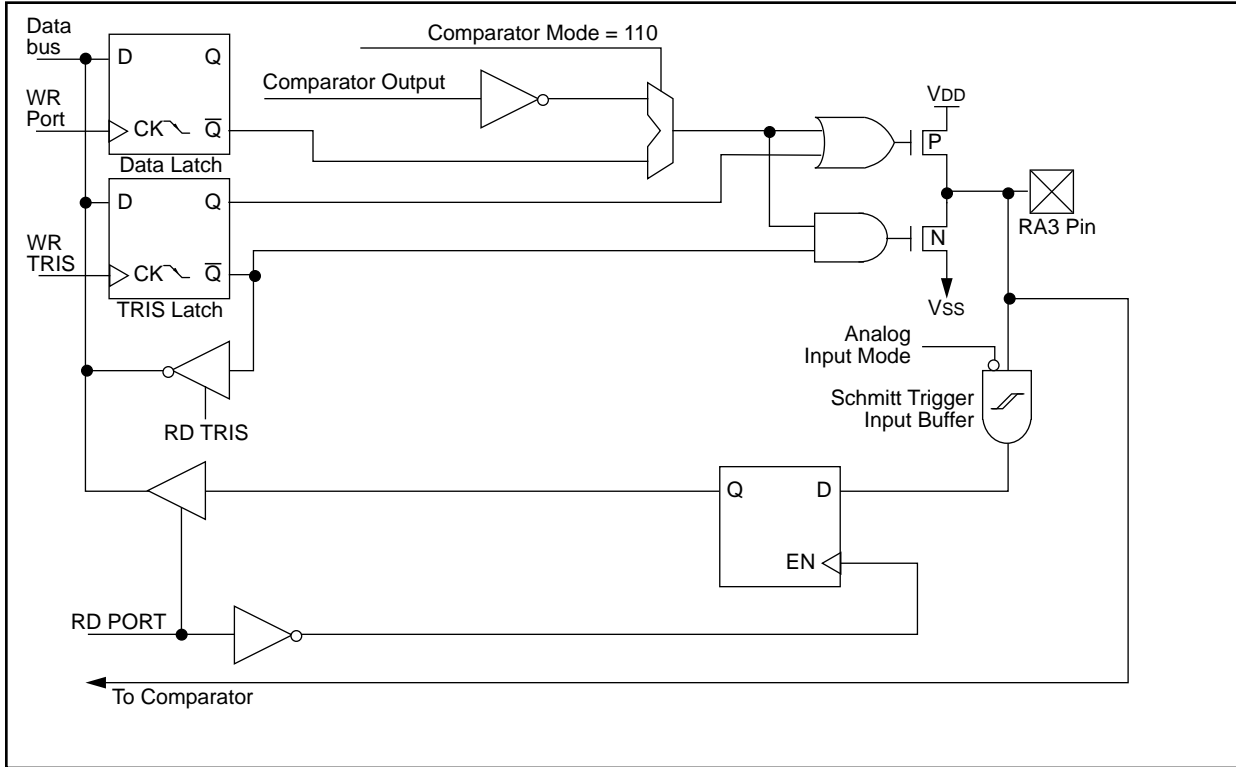
```
CLRF   PORTA           ;Initialize PORTA by setting
                        ;output data latches
MOVLW  0X07           ;Turn comparators off and
MOVWF  CMCON           ;enable pins for I/O
                        ;functions
BSF    STATUS, RP0    ;Select Bank1
MOVLW  0x1F           ;Value used to initialize
                        ;data direction
MOVWF  TRISA          ;Set RA<4:0> as inputs
                        ;TRISA<7:5> are always
                        ;read as '0'.
```

**FIGURE 5-2: BLOCK DIAGRAM OF RA2 PIN**

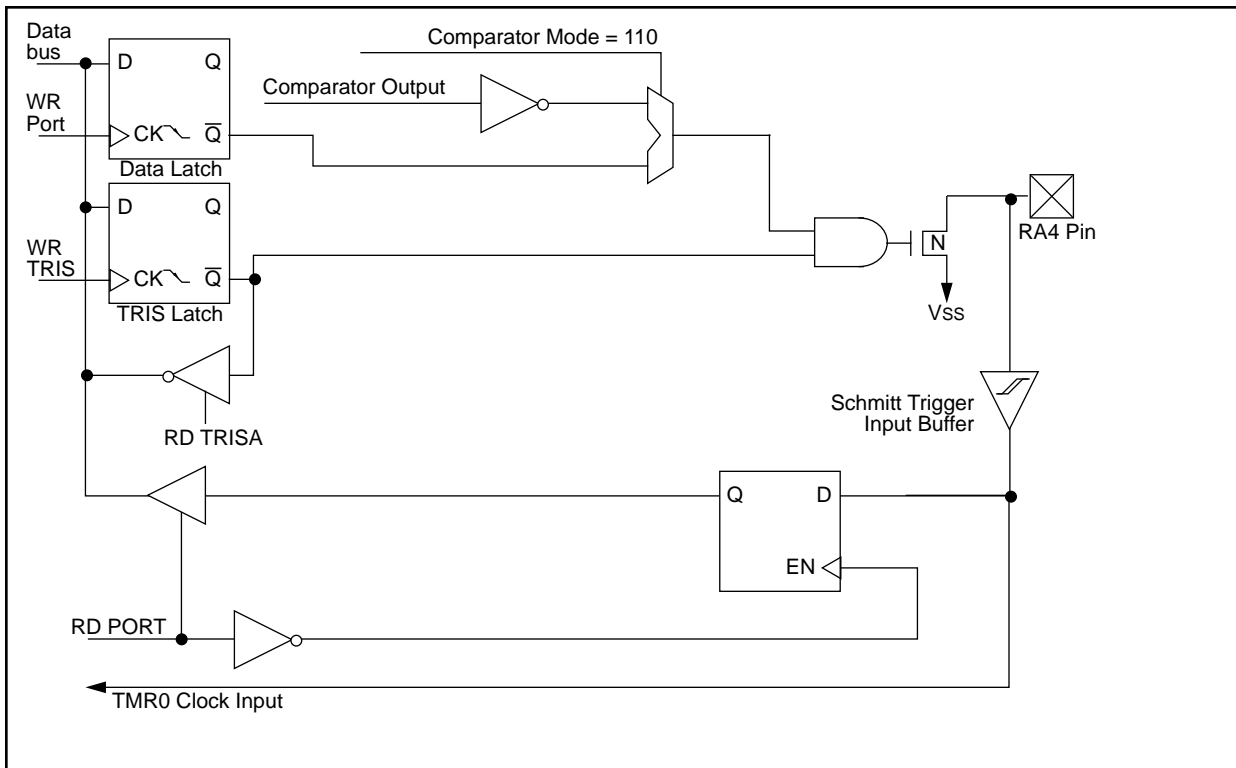


# PIC16C62X

**FIGURE 5-3: BLOCK DIAGRAM OF RA3 PIN**



**FIGURE 5-4: BLOCK DIAGRAM OF RA4 PIN**





**TABLE 5-1: PORTA FUNCTIONS**

Name	Bit #	Buffer Type	Function
RA0/AN0	bit0	ST	Input/output or comparator input
RA1/AN1	bit1	ST	Input/output or comparator input
RA2/AN2/VREF	bit2	ST	Input/output or comparator input or VREF output
RA3/AN3	bit3	ST	Input/output or comparator input/output
RA4/T0CKI	bit4	ST	Input/output or external clock input for TMR0 or comparator output. Output is open drain type.

Legend: ST = Schmitt Trigger input

**TABLE 5-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR / BOR	Value on All Other Resets
05h	PORTA	—	—	—	RA4	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
85h	TRISA	—	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111
1Fh	CMCON	C2OUT	C1OUT	—	—	CIS	CM2	CM1	CM0	00-- 0000	00-- 0000
9Fh	VRCON	VREN	VROE	VRR	—	VR3	VR2	VR1	VR0	000- 0000	000- 0000

Legend: — = Unimplemented locations, read as '0'

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

## 5.2 PORTB and TRISB Registers

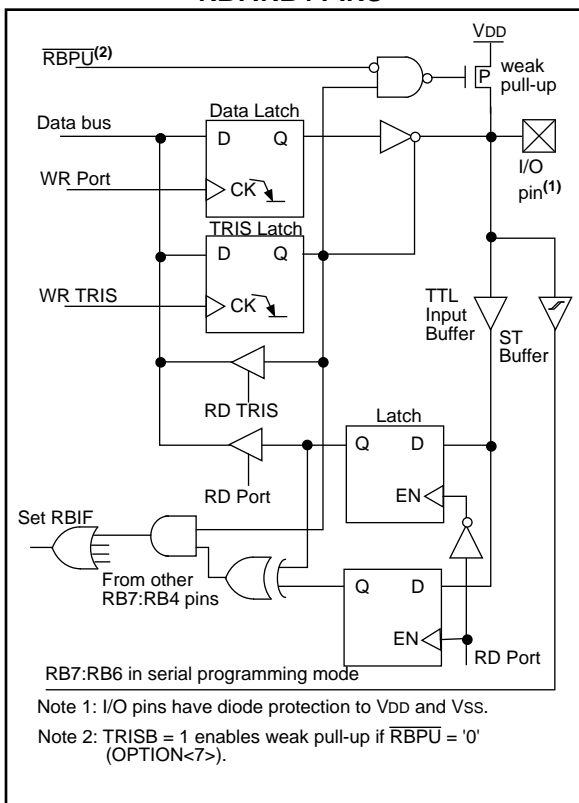
PORTB is an 8-bit wide bi-directional port. The corresponding data direction register is TRISB. A '1' in the TRISB register puts the corresponding output driver in a high impedance mode. A '0' in the TRISB register puts the contents of the output latch on the selected pin(s).

Reading PORTB register reads the status of the pins, whereas writing to it will write to the port latch. All write operations are read-modify-write operations. So a write to a port implies that the port pins are first read, then this value is modified and written to the port data latch.

Each of the PORTB pins has a weak internal pull-up ( $\approx 200 \mu\text{A}$  typical). A single control bit can turn on all the pull-ups. This is done by clearing the  $\overline{\text{RBP}}\text{U}$  (OPTION<7>) bit. The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on Power-on Reset.

Four of PORTB's pins, RB7:RB4, have an interrupt on change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt on change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'ed together to generate the RBIF interrupt (flag latched in INTCON<0>).

**FIGURE 5-5: BLOCK DIAGRAM OF RB7:RB4 PINS**



This interrupt can wake the device from SLEEP. The user, in the interrupt service routine, can clear the interrupt in the following manner:

- Any read or write of PORTB. This will end the mismatch condition.
- Clear flag bit RBIF.

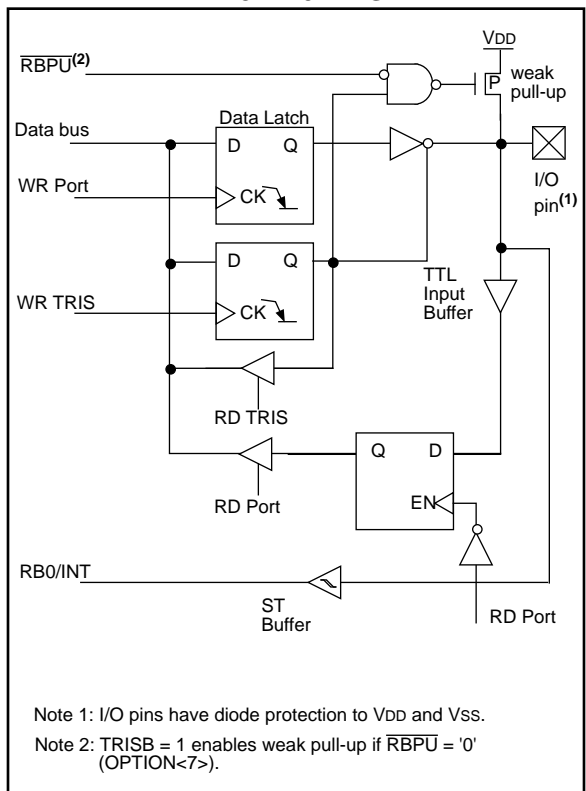
A mismatch condition will continue to set flag bit RBIF. Reading PORTB will end the mismatch condition, and allow flag bit RBIF to be cleared.

This interrupt on mismatch feature, together with software configurable pull-ups on these four pins allow easy interface to a key pad and make it possible for wake-up on key-depression. (See AN552 in the Microchip *Embedded Control Handbook*.)

**Note:** If a change on the I/O pin should occur when the read operation is being executed (start of the Q2 cycle), then the RBIF interrupt flag may not get set.

The interrupt on change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt on change feature. Polling of PORTB is not recommended while using the interrupt on change feature.

**FIGURE 5-6: BLOCK DIAGRAM OF RB3:RB0 PINS**



**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 / BOR	Value on All Other Rests
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	RBPŪ	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

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

# PIC16C62X

## 5.3 I/O Programming Considerations

### 5.3.1 BI-DIRECTIONAL I/O PORTS

Any instruction which writes, operates internally as a read followed by a write operation. The BCF and BSF instructions, for example, read the register into the CPU, execute the bit operation and write the result back to the register. Caution must be used when these instructions are applied to a port with both inputs and outputs defined. For example, a BSF operation on bit5 of PORTB will cause all eight bits of PORTB to be read into the CPU. Then the BSF operation takes place on bit5 and PORTB is written to the output latches. If another bit of PORTB is used as a bidirectional I/O pin (e.g., bit0) and it is defined as an input at this time, the input signal present on the pin itself would be read into the CPU and re-written to the data latch of this particular pin, overwriting the previous content. As long as the pin stays in the input mode, no problem occurs. However, if bit0 is switched into output mode later on, the content of the data latch may now be unknown.

Reading the port register, reads the values of the port pins. Writing to the port register writes the value to the port latch. When using read modify write instructions (ex. BCF, BSF, etc.) on a port, the value of the port pins is read, the desired operation is done to this value, and this value is then written to the port latch.

Example 5-2 shows the effect of two sequential read-modify-write instructions (ex., BCF, BSF, etc.) on an I/O port.

A pin actively outputting a Low or High should not be driven from external devices at the same time in order to change the level on this pin ("wired-or", "wired-and"). The resulting high output currents may damage the chip.

### EXAMPLE 5-2: READ-MODIFY-WRITE INSTRUCTIONS ON AN I/O PORT

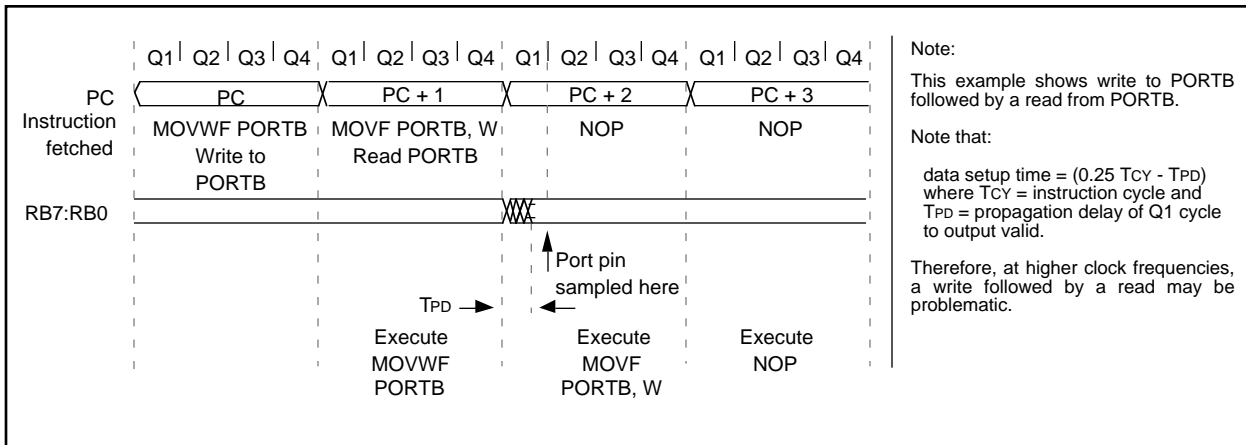
```

; Initial PORT settings:   PORTB<7:4> Inputs
;
;                           PORTB<3:0> Outputs
; PORTB<7:6> have external pull-up and are not
; connected to other circuitry
;
;                           PORT latch  PORT pins
;                           -----  -----
;
BCF PORTB, 7           ; 01pp pppp   11pp pppp
BCF PORTB, 6           ; 10pp pppp   11pp pppp
BSF STATUS,RP0        ;
BCF TRISB, 7          ; 10pp pppp   11pp pppp
BCF TRISB, 6          ; 10pp pppp   10pp pppp
;
; Note that the user may have expected the pin
; values to be 00pp pppp. The 2nd BCF caused
; RB7 to be latched as the pin value (High).
    
```

### 5.3.2 SUCCESSIVE OPERATIONS ON I/O PORTS

The actual write to an I/O port happens at the end of an instruction cycle, whereas for reading, the data must be valid at the beginning of the instruction cycle (Figure 5-7). Therefore, care must be exercised if a write followed by a read operation is carried out on the same I/O port. The sequence of instructions should be such to allow the pin voltage to stabilize (load dependent) before the next instruction which causes that file to be read into the CPU is executed. Otherwise, the previous state of that pin may be read into the CPU rather than the new state. When in doubt, it is better to separate these instructions with a NOP or another instruction not accessing this I/O port.

FIGURE 5-7: SUCCESSIVE I/O OPERATION



## 6.0 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 6-1 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 6-2 and Figure 6-3). 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

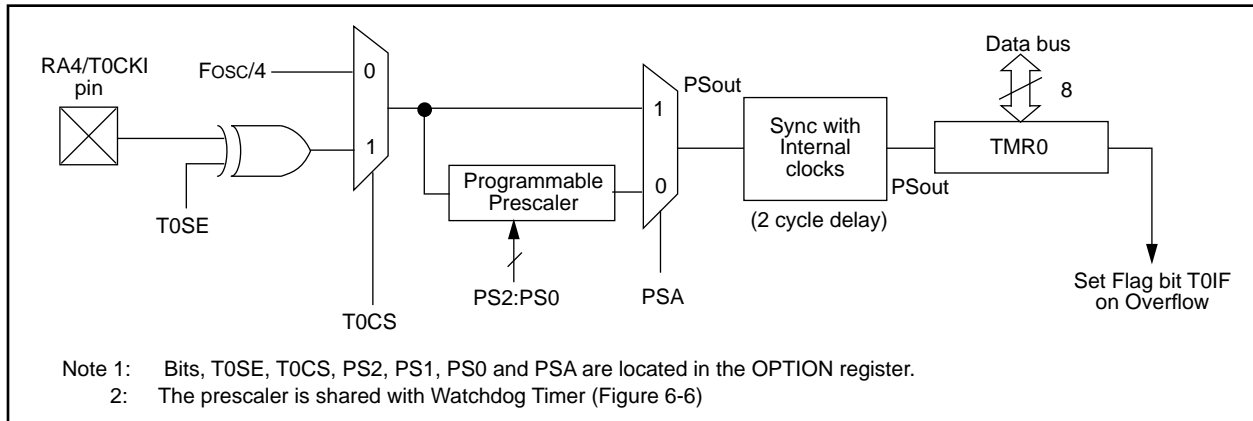
bit (OPTION<4>). Clearing the T0SE bit selects the rising edge. Restrictions on the external clock input are discussed in detail in Section 6.2.

The prescaler is shared between the Timer0 module and the WatchdogTimer. 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 6.3 details the operation of the prescaler.

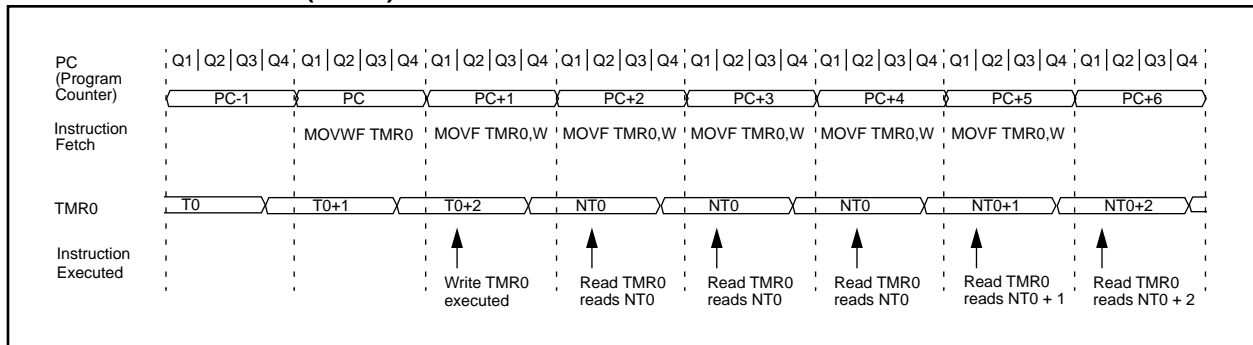
### 6.1 TIMER0 Interrupt

Timer0 interrupt is generated when the TMR0 register timer/counter overflows from FFh to 00h. This overflow sets the TOIF bit. The interrupt can be masked by clearing the TOIE bit (INTCON<5>). The TOIF 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 6-4 for Timer0 interrupt timing.

**FIGURE 6-1: TIMER0 BLOCK DIAGRAM**

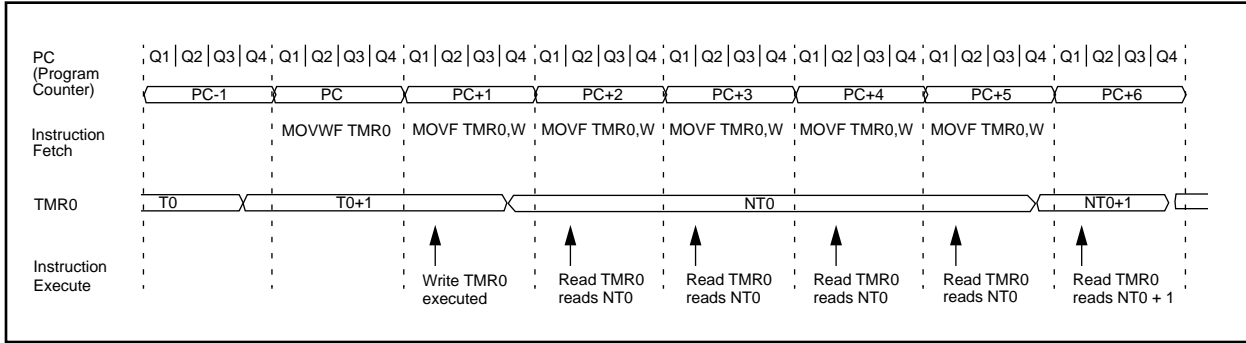


**FIGURE 6-2: TIMER0 (TMR0) TIMING: INTERNAL CLOCK/NO PRESCALER**

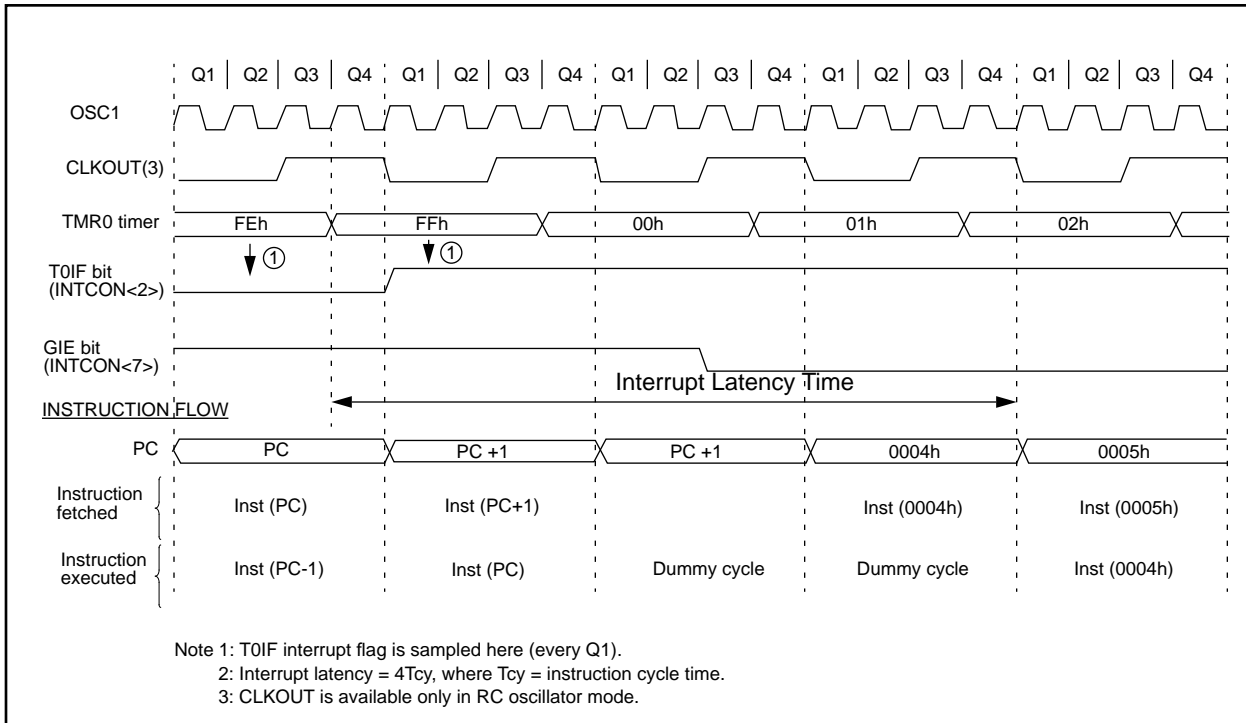


# PIC16C62X

**FIGURE 6-3: TIMER0 TIMING: INTERNAL CLOCK/PRESCALE 1:2**



**FIGURE 6-4: TIMER0 INTERRUPT TIMING**



## 6.2 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.

### 6.2.1 EXTERNAL CLOCK SYNCHRONIZATION

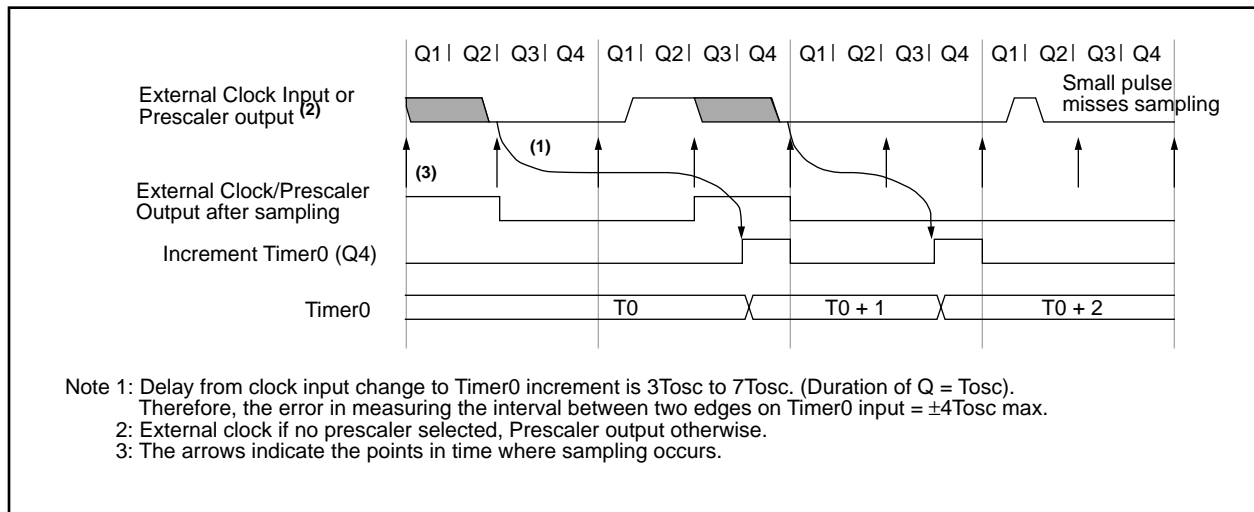
When no prescaler is used, the external clock input is the same as the prescaler output. The synchronization of T0CKI with the internal phase clocks is accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the internal phase clocks (Figure 6-5). Therefore, it is necessary for T0CKI to be high for at least  $2T_{OSC}$  (and a small RC delay of 20 ns) and low for at least  $2T_{OSC}$  (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  $4T_{OSC}$  (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.

### 6.2.2 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 6-5 shows the delay from the external clock edge to the timer incrementing.

**FIGURE 6-5: TIMER0 TIMING WITH EXTERNAL CLOCK**



# PIC16C62X

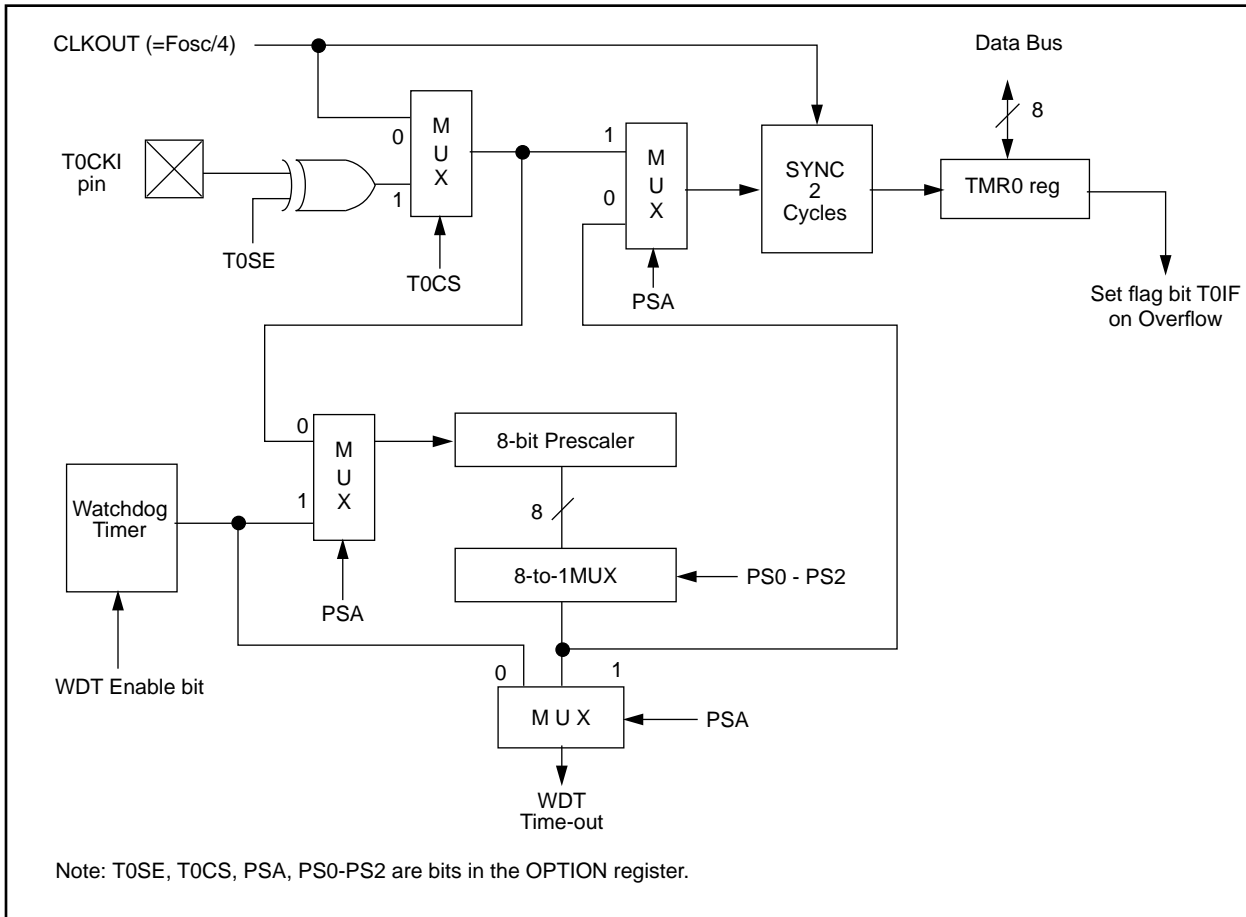
## 6.3 Prescaler

An 8-bit counter is available as a prescaler for the Timer0 module, or as a postscaler for the Watchdog Timer, respectively (Figure 6-6). For simplicity, this counter is being referred to as “prescaler” throughout this data sheet. Note that there is only one prescaler available which is mutually exclusive between the Timer0 module and the Watchdog Timer. Thus, a prescaler assignment for the Timer0 module means that there is no prescaler for the Watchdog Timer, and vice-versa.

The PSA and PS2:PS0 bits (OPTION<3:0>) determine the prescaler assignment and prescale ratio.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g., CLRF 1, MOVWF 1, BSF 1,x,...etc.) will clear the prescaler. When assigned to WDT, a CLRWDT instruction will clear the prescaler along with the Watchdog Timer. The prescaler is not readable or writable.

**FIGURE 6-6: BLOCK DIAGRAM OF THE TIMER0/WDT PRESCALER**





## 6.3.1 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 6-1) must be executed when changing the prescaler assignment from Timer0 to WDT.

### EXAMPLE 6-1: CHANGING PRESCALER (TIMER0→WDT)

```
BCF STATUS, RP0 ;Bank 0
CLRWF TMR0 ;Clear TMR0 & Prescaler
BSF STATUS, RP0 ;Bank 1
CLRWDT ;Clears WDT and
;
MOVLW b'xxxxlxxx' ;Select new prescaler
MOVWF OPTION ;value
BCF STATUS, RP0 ;Bank 0
```

To change prescaler from the WDT to the TMR0 module use the sequence shown in Example 6-2. This precaution must be taken even if the WDT is disabled.

### EXAMPLE 6-2: CHANGING PRESCALER (WDT→TIMER0)

```
CLRWDT ;Clear WDT and
;prescaler
BSF STATUS, RP0
MOVLW b'xxx0xxx' ;Select TMR0, new
;prescale value and
;clock source
MOVWF OPTION
BCF STATUS, RP0
```

**TABLE 6-1: 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 / BOR	Value on All Other Resets
01h	TMR0	Timer0 module's register								xxxx xxxx	uuuu uuuu
0Bh/8Bh	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 000x	0000 000x
81h	OPTION	$\overline{\text{RBPU}}$	INTEDG	TOCS	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'.

**Note:** Shaded bits are not used by TMR0 module.

# PIC16C62X

---

NOTES:

## 7.0 COMPARATOR MODULE

The comparator module contains two analog comparators. The inputs to the comparators are multiplexed with the RA0 through RA3 pins. The on-chip Voltage Reference (Section 8.0) can also be an input to the comparators.

The CMCON register, shown in Figure 7-1, controls the comparator input and output multiplexers. A block diagram of the comparator is shown in Figure 7-2.

**FIGURE 7-1: CMCON REGISTER (ADDRESS 1Fh)**

R-0	R-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
C2OUT	C1OUT	-	-	CIS	CM2	CM1	CM0
bit7				bit0			

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

bit 7: **C2OUT**: Comparator 2 output  
 1 = C2 VIN+ > C2 VIN-  
 2 = C2 VIN+ < C2 VIN-

bit 6: **C1OUT**: Comparator 1 output  
 1 = C1 VIN+ > C1 VIN-  
 2 = C1 VIN+ < C1 VIN-

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

bit 3: **CIS**: Comparator Input Switch  
 When CM<2:0> = 001:  
 1 = C1 VIN- connects to RA3  
 0 = C1 VIN- connects to RA0  
 When CM<2:0> = 010:  
 1 = C1 VIN- connects to RA3  
     C2 VIN- connects to RA2  
 0 = C1 VIN- connects to RA0  
     C2 VIN- connects to RA1

bit 2-0: **CM<2:0>**: Comparator mode  
 Figure 7-2.

# PIC16C62X

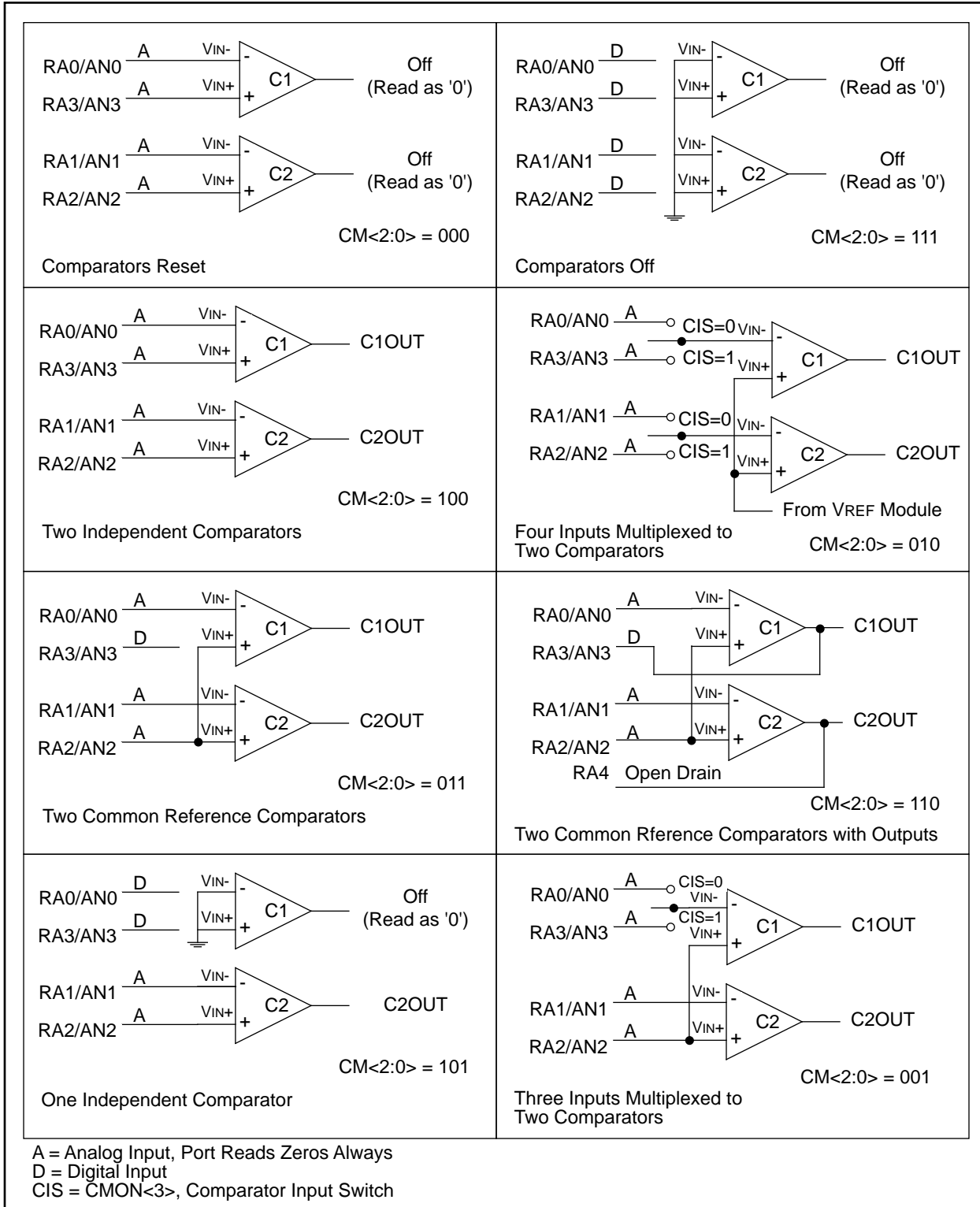
## 7.1 Comparator Configuration

There are eight modes of operation for the comparators. The CMCON register is used to select the mode. Figure 7-2 shows the eight possible modes. The TRISA register controls the data direction of the comparator pins for each mode. If the comparator

mode is changed, the comparator output level may not be valid for the specified mode change delay shown in Table 12-2.

**Note:** Comparator interrupts should be disabled during a comparator mode change otherwise a false interrupt may occur.

**FIGURE 7-2: COMPARATOR I/O OPERATING MODES**



The code example in Example 7-1 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 7-1: INITIALIZING COMPARATOR MODULE

```

FLAG_REG EQU      0X20
CLRF      FLAG_REG ;Init flag register
CLRF      PORTA    ;Init PORTA
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
    
```

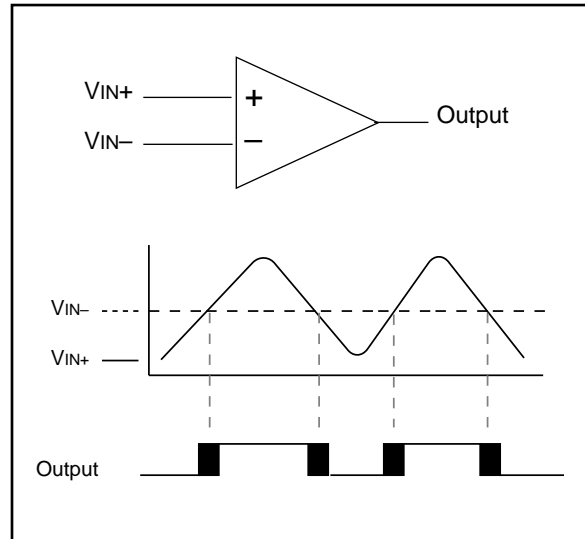
## 7.2 Comparator Operation

A single comparator is shown in Figure 7-3 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 7-3 represent the uncertainty due to input offsets and response time.

## 7.3 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 7-3).

FIGURE 7-3: SINGLE COMPARATOR



### 7.3.1 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).

### 7.3.2 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 7-2). In this mode, the internal voltage reference is applied to the VIN+ pin of both comparators.

# PIC16C62X

## 7.4 Comparator Response Time

Response time is the minimum time, after selecting a new reference voltage or input source, before the comparator output is guaranteed to have a valid level. If the internal reference is changed, the maximum delay of the internal voltage reference must be considered when using the comparator outputs. Otherwise the maximum delay of the comparators should be used (Table 12-4 and Table 12-5).

## 7.5 Comparator Outputs

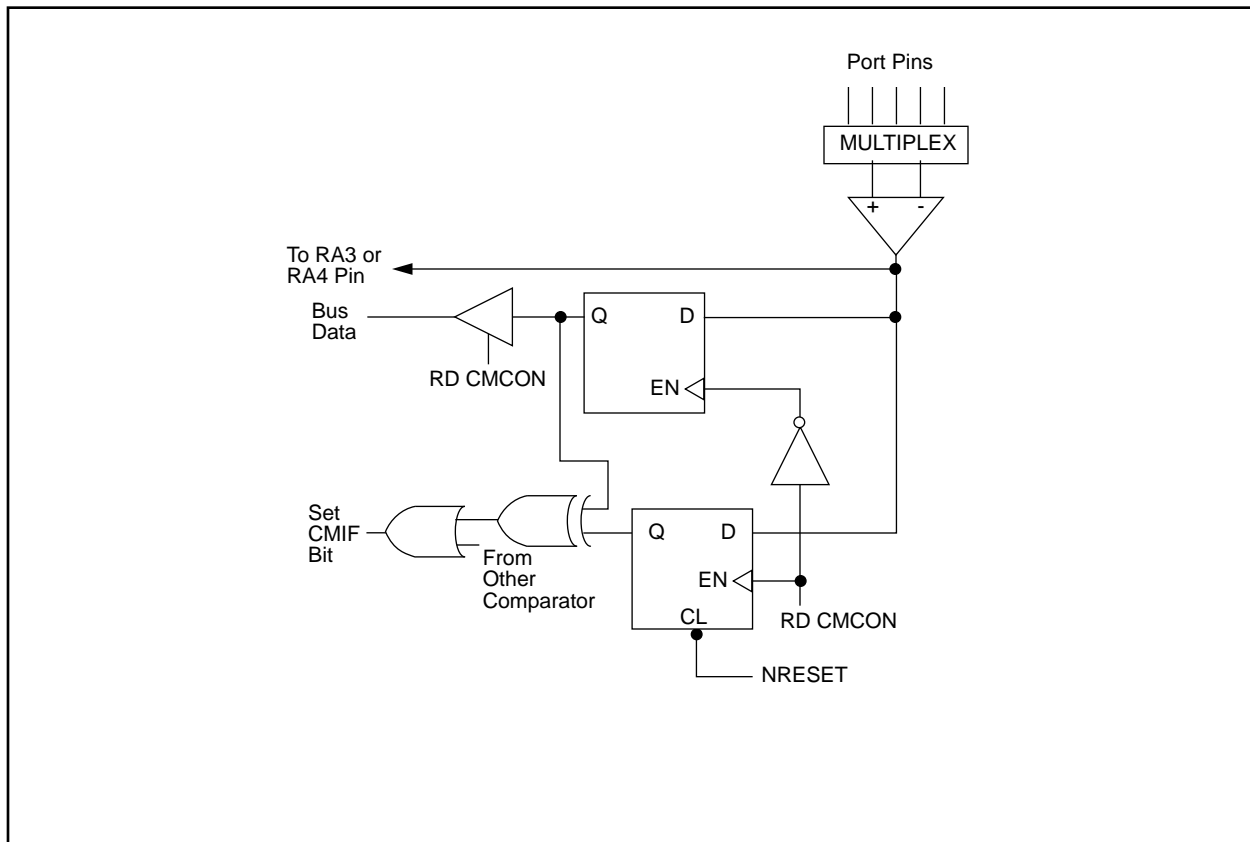
The comparator outputs are read through the CMCON register. These bits are read only. The comparator outputs may also be directly output to the RA3 and RA4 I/O pins. When the CM<2:0> = 110, multiplexers in the output path of the RA3 and RA4 pins will switch and the output of each pin will be the unsynchronized output of the comparator. The uncertainty of each of the comparators is related to the input offset voltage and the response time given in the specifications. Figure 7-4 shows the comparator output block diagram.

The TRISA bits will still function as an output enable/disable for the RA3 and RA4 pins while in this mode.

**Note 1:** When reading the PORT register, all pins configured as analog inputs will read as a '0'. Pins configured as digital inputs will convert an analog input according to the Schmitt trigger input specification.

**2:** Analog levels on any pin that is defined as a digital input may cause the input buffer to consume more current than is specified.

FIGURE 7-4: COMPARATOR OUTPUT BLOCK DIAGRAM



## 7.6 Comparator Interrupts

The comparator interrupt flag is set whenever there is a change in the output value of either comparator. Software will need to maintain information about the status of the output bits, as read from CMCON<7:6>, to determine the actual change that has occurred. The CMIF bit, PIR1<6>, is the comparator interrupt flag. The CMIF bit must be reset by clearing '0'. Since it is also possible to write a '1' to this register, a simulated interrupt may be initiated.

The CMIE bit (PIE1<6>) and the PEIE bit (INTCON<6>) must be set to enable the interrupt. In addition, the GIE bit must also be set. If any of these bits are clear, the interrupt is not enabled, though the CMIF bit will still be set if an interrupt condition occurs.

**Note:** If a change in the CMCON register (C1OUT or C2OUT) should occur when a read operation is being executed (start of the Q2 cycle), then the CMIF (PIR1<6>) interrupt flag may not get set.

The user, in the interrupt service routine, can clear the interrupt in the following manner:

- a) Any read or write of CMCON. This will end the mismatch condition.
- b) Clear flag bit CMIF.

A mismatch condition will continue to set flag bit CMIF. Reading CMCON will end the mismatch condition, and allow flag bit CMIF to be cleared.

## 7.7 Comparator Operation During SLEEP

When a comparator is active and the device is placed in SLEEP mode, the comparator remains active and the interrupt is functional if enabled. This interrupt will

wake up the device from SLEEP mode when enabled. While the comparator is powered-up, higher sleep currents than shown in the power down current specification will occur. Each comparator that is operational will consume additional current as shown in the comparator specifications. To minimize power consumption while in SLEEP mode, turn off the comparators, CM<2:0> = 111, before entering sleep. If the device wakes-up from sleep, the contents of the CMCON register are not affected.

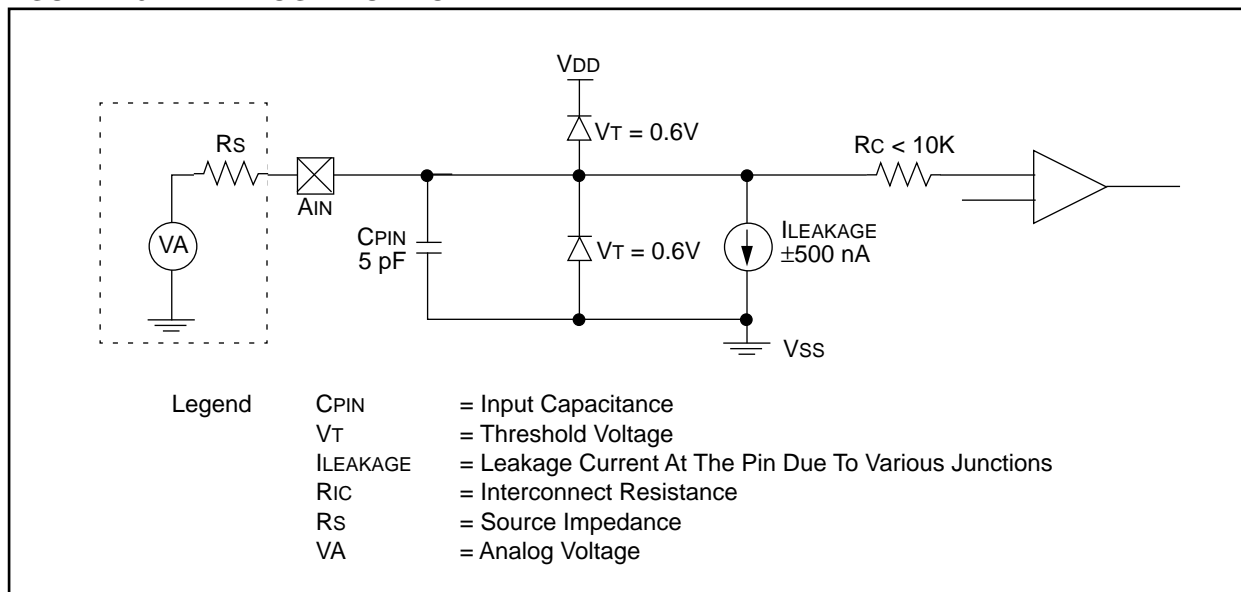
## 7.8 Effects of a RESET

A device reset forces the CMCON register to its reset state. This forces the comparator module to be in the comparator reset mode, CM2:CM0 = 000. This ensures that all potential inputs are analog inputs. Device current is minimized when analog inputs are present at reset time. The comparators will be powered-down during the reset interval.

## 7.9 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 7-5. Since the analog pins are connected to a digital output, they have reverse biased diodes to VDD and VSS. The analog input therefore, must be between VSS and VDD. 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 10 kΩ is recommended for the analog sources. Any external component connected to an analog input pin, such as a capacitor or a Zener diode, should have very little leakage current.

**FIGURE 7-5: ANALOG INPUT MODEL**



# PIC16C62X

**TABLE 7-1: REGISTERS ASSOCIATED WITH COMPARATOR MODULE**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR / BOR	Value on All Other Resets
1Fh	CMCON	C2OUT	C1OUT	—	—	CIS	CM2	CM1	CM0	00-- 0000	00-- 0000
9Fh	VRCON	VREN	VROE	VRR	—	VR3	VR2	VR1	VR0	000- 0000	000- 0000
0Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000x
0Ch	PIR1	—	CMIF	—	—	—	—	—	—	-0-- ----	-0-- ----
8Ch	PIE1	—	CMIE	—	—	—	—	—	—	-0-- ----	-0-- ----
85h	TRISA	—	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111

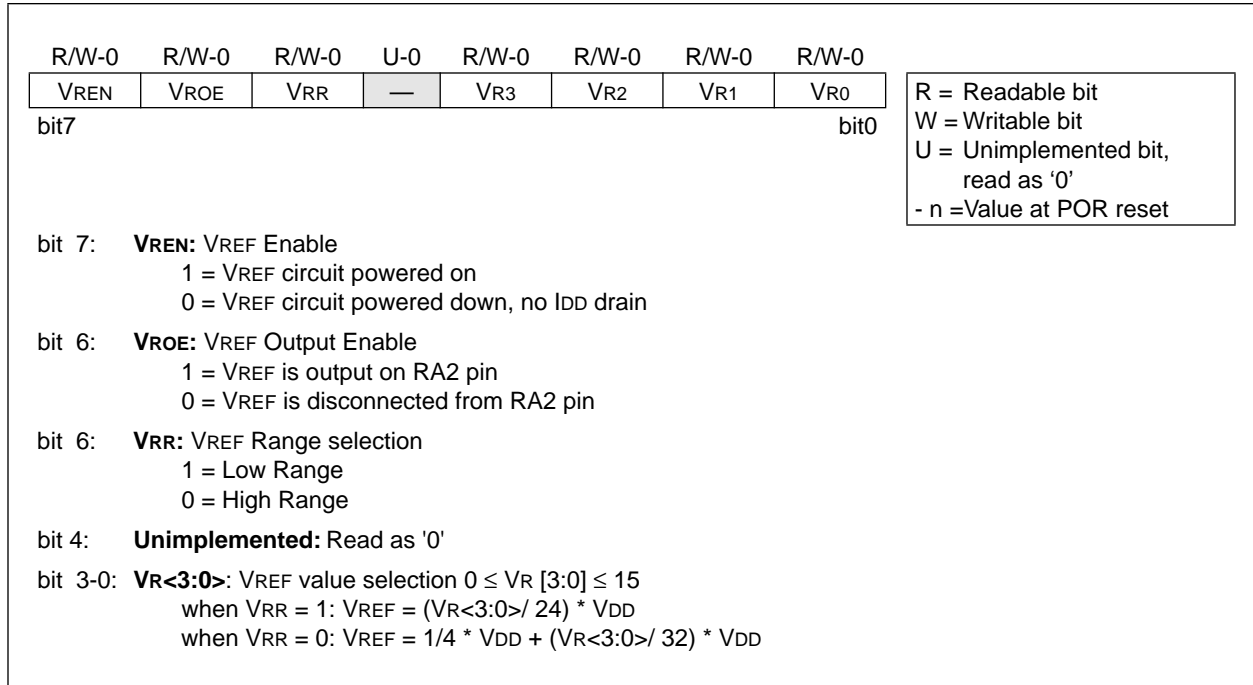


## 8.0 VOLTAGE REFERENCE MODULE

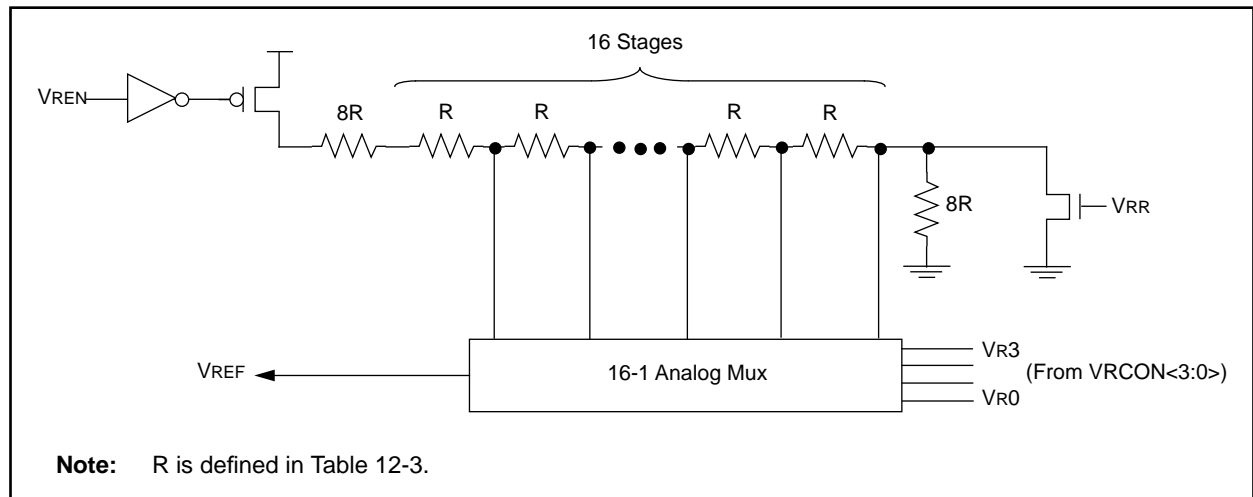
The Voltage Reference is a 16-tap resistor ladder network that provides a selectable voltage reference. The resistor ladder is segmented to provide two ranges

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 Figure 8-1. The block diagram is given in Figure 8-2.

**FIGURE 8-1: VRCON REGISTER (ADDRESS 9Fh)**



**FIGURE 8-2: VOLTAGE REFERENCE BLOCK DIAGRAM**



# PIC16C62X

## 8.1 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 12-2). Example 8-1 shows an example of how to configure the Voltage Reference for an output voltage of 1.25V with VDD = 5.0V.

### EXAMPLE 8-1: VOLTAGE REFERENCE CONFIGURATION

```

MOVLW    0x02    ; 4 Inputs Muxed
MOVWF    CMCON   ; to 2 comps.
BSF      STATUS,RP0 ; go to Bank 1
MOVLW    0x07    ; RA3-RA0 are
MOVWF    TRISA   ; outputs
MOVLW    0xA6    ; enable VREF
MOVWF    VRCON   ; low range
           ; set Vr<3:0>=6

BCF      STATUS,RP0 ; go to Bank 0
CALL     DELAY10 ; 10µs delay
    
```

## 8.2 Voltage Reference Accuracy/Error

The full range of VSS to VDD cannot be realized due to the construction of the module. The transistors on the top and bottom of the resistor ladder network (Figure 8-2) keep VREF from approaching VSS or VDD. The Voltage Reference is VDD derived and therefore,

the VREF output changes with fluctuations in VDD. The absolute accuracy of the Voltage Reference can be found in Table 12-3.

## 8.3 Operation During Sleep

When the device wakes up from sleep through an interrupt or a Watchdog Timer time-out, the contents of the VRCON register are not affected. To minimize current consumption in SLEEP mode, the Voltage Reference should be disabled.

## 8.4 Effects of a Reset

A device reset disables the Voltage Reference by clearing bit VREN (VRCON<7>). This reset also disconnects the reference from the RA2 pin by clearing bit VROE (VRCON<6>) and selects the high voltage range by clearing bit VRR (VRCON<5>). The VREF value select bits, VRCON<3:0>, are also cleared.

## 8.5 Connection Considerations

The Voltage Reference Module operates independently of the comparator module. The output of the reference generator may be connected to the RA2 pin if the TRISA<2> bit is set and the VROE bit, VRCON<6>, is set. Enabling the Voltage Reference output onto the RA2 pin with an input signal present will increase current consumption. Connecting RA2 as a digital output with VREF enabled will also increase current consumption.

The RA2 pin can be used as a simple D/A output with limited drive capability. Due to the limited drive capability, a buffer must be used in conjunction with the Voltage Reference output for external connections to VREF. Figure 8-3 shows an example buffering technique.

FIGURE 8-3: VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE

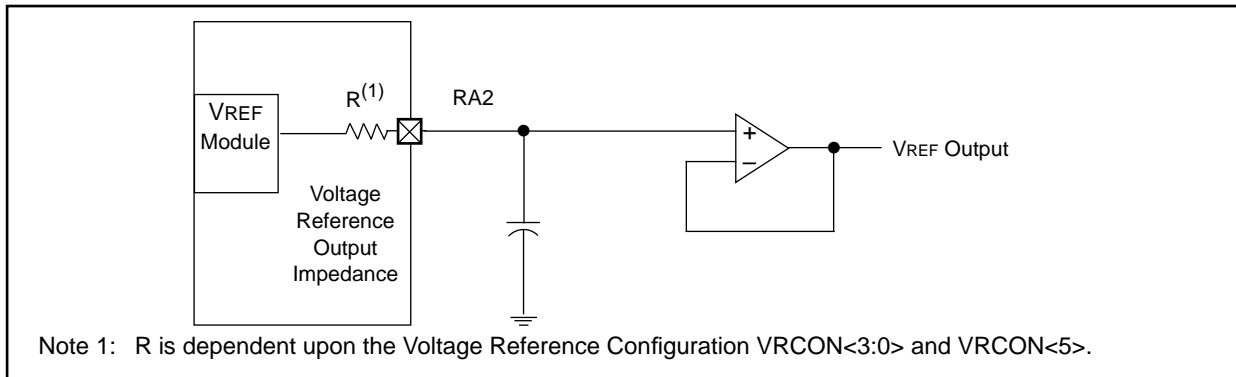


TABLE 8-1: REGISTERS ASSOCIATED WITH VOLTAGE REFERENCE

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value On POR / BOR	Value On All Other Resets
9Fh	VRCON	VREN	VROE	VRR	—	VR3	VR2	VR1	VR0	000- 0000	000- 0000
1Fh	CMCON	C2OUT	C1OUT	—	—	CIS	CM2	CM1	CM0	00-- 0000	00-- 0000
85h	TRISA	—	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111

## 9.0 SPECIAL FEATURES OF THE CPU

What sets apart a microcontroller from other processors are special circuits to deal with the needs of real time applications. The PIC16C62X family has a host of such features intended to maximize system reliability, minimize cost through elimination of external components, provide power saving operating modes and offer code protection.

These are:

1. OSC selection
2. Reset
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-Up Timer (OST)
  - Brown-out Reset (BOR)
3. Interrupts
4. Watchdog Timer (WDT)
5. SLEEP
6. Code protection
7. ID Locations
8. In-circuit serial programming

The PIC16C62X has a Watchdog Timer which is controlled by configuration bits. It runs off its own RC oscillator for added reliability. There are two timers that offer necessary delays on power-up. One is the Oscillator Start-up Timer (OST), intended to keep the chip in reset until the crystal oscillator is stable. The other is the Power-up Timer (PWRT), which provides a fixed delay of 72 ms (nominal) on power-up only, designed to keep the part in reset while the power supply stabilizes. There is also circuitry to reset the device if a brown-out occurs which provides at least a 72 ms reset. With these three functions on-chip, most applications need no external reset circuitry.

The SLEEP mode is designed to offer a very low current power-down mode. The user can wake-up from SLEEP through external reset, Watchdog Timer wake-up or through an interrupt. Several oscillator options are also made available to allow the part to fit the application. The RC oscillator option saves system cost while the LP crystal option saves power. A set of configuration bits are used to select various options.

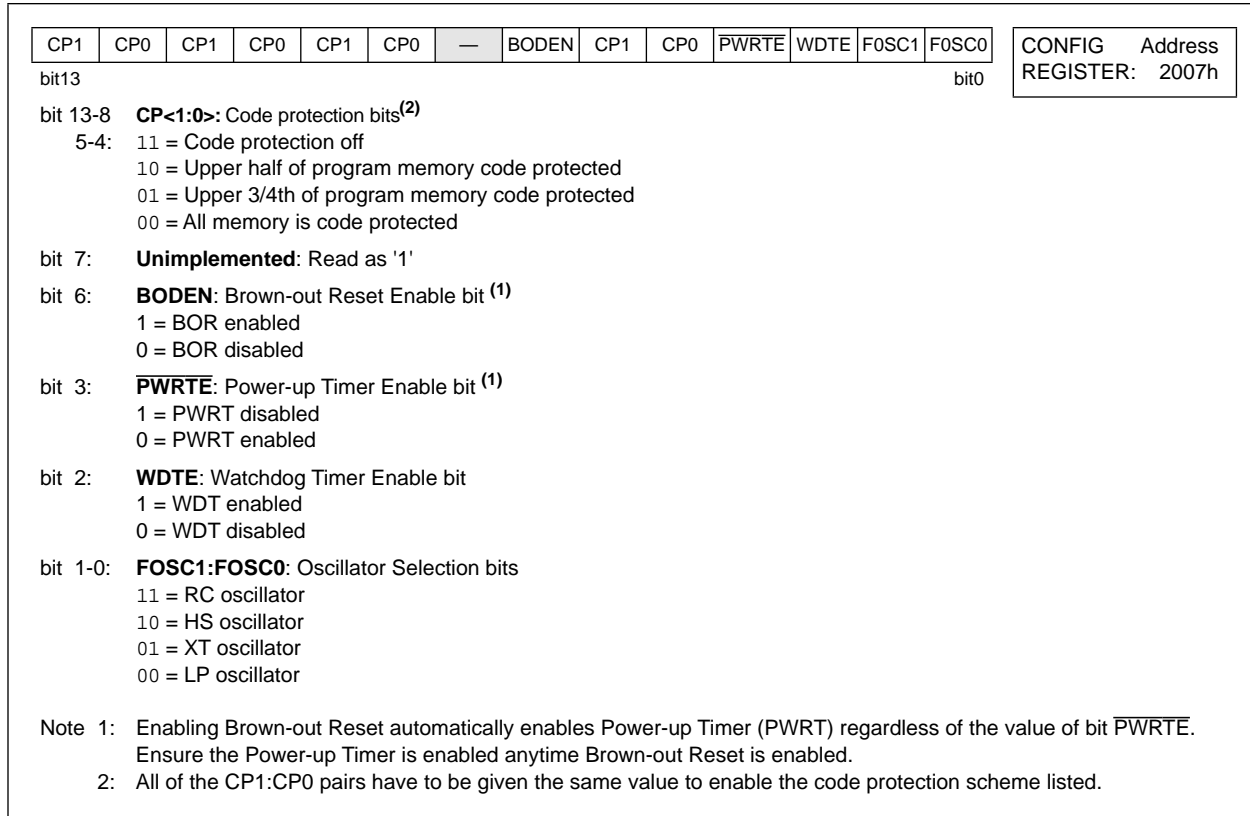
# PIC16C62X

## 9.1 Configuration Bits

The configuration bits can be programmed (read as '0') or left unprogrammed (read as '1') to select various device configurations. These bits are mapped in program memory location 2007h.

The user will note that address 2007h is beyond the user program memory space. In fact, it belongs to the special test/configuration memory space (2000h – 3FFFh), which can be accessed only during programming.

**FIGURE 9-1: CONFIGURATION WORD**



## 9.2 Oscillator Configurations

### 9.2.1 OSCILLATOR TYPES

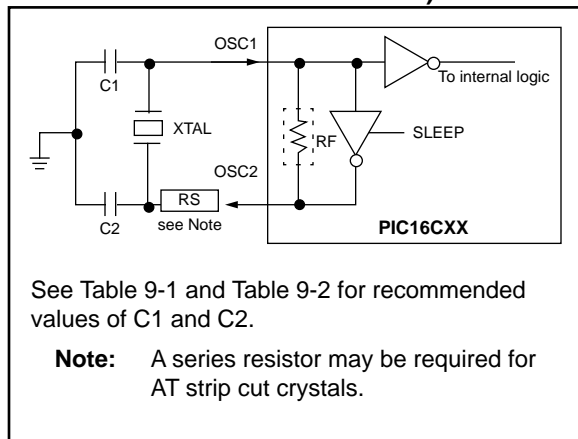
The PIC16CXX can be operated in four different oscillator options. The user can program two configuration bits (FOSC1 and FOSC0) to select one of these four modes:

- LP Low Power Crystal
- XT Crystal/Resonator
- HS High Speed Crystal/Resonator
- RC Resistor/Capacitor

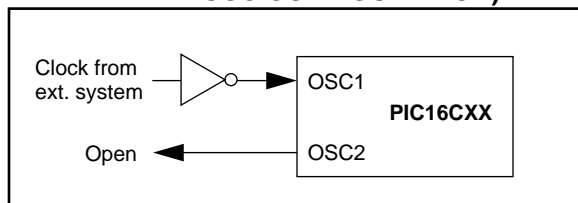
### 9.2.2 CRYSTAL OSCILLATOR / CERAMIC RESONATORS

In XT, LP or HS modes a crystal or ceramic resonator is connected to the OSC1 and OSC2 pins to establish oscillation (Figure 9-2). The PIC16CXX oscillator design requires the use of a parallel cut crystal. Use of a series cut crystal may give a frequency out of the crystal manufacturers specifications. When in XT, LP or HS modes, the device can have an external clock source to drive the OSC1 pin (Figure 9-3).

**FIGURE 9-2: CRYSTAL OPERATION (OR CERAMIC RESONATOR) (HS, XT OR LP OSC CONFIGURATION)**



**FIGURE 9-3: EXTERNAL CLOCK INPUT OPERATION (HS, XT OR LP OSC CONFIGURATION)**



**TABLE 9-1: CAPACITOR SELECTION FOR CERAMIC RESONATORS (PRELIMINARY)**

Ranges Characterized:		
Mode	Freq	OSC1
XT	455 kHz	22 - 100 pF
	2.0 MHz	15 - 68 pF
	4.0 MHz	15 - 68 pF
HS	8.0 MHz	10 - 68 pF
	16.0 MHz	10 - 22 pF

Note: Recommended values of C1 and C2 are identical to the ranges tested table. Higher capacitance increases the stability of the oscillator but also increases the start-up time. These values are for design guidance only. Since each resonator has its own characteristics, the user should consult the resonator manufacturer for appropriate values of external components.

Resonators to be Characterized:		
455 kHz	Panasonic EFO-A455K04B	±0.3%
2.0 MHz	Murata Erie CSA2.00MG	±0.5%
4.0 MHz	Murata Erie CSA4.00MG	±0.5%
8.0 MHz	Murata Erie CSA8.00MT	±0.5%
16.0 MHz	Murata Erie CSA16.00MX	±0.5%

All resonators used did not have built-in capacitors.

**TABLE 9-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR (PRELIMINARY)**

Mode	Freq	OSC1	OSC2
LP	32 kHz	68 - 100 pF	68 - 100 pF
	200 kHz	15 - 30 pF	15 - 30 pF
XT	100 kHz	68 - 150 pF	150 - 200 pF
	2 MHz	15 - 30 pF	15 - 30 pF
	4 MHz	15 - 30 pF	15 - 30 pF
HS	8 MHz	15 - 30 pF	15 - 30 pF
	10 MHz	15 - 30 pF	15 - 30 pF
	20 MHz	15 - 30 pF	15 - 30 pF

Higher capacitance increases the stability of the oscillator but also increases the start-up time. These values are for design guidance only. Rs may be required in HS mode as well as XT mode to avoid overdriving crystals with low drive level specification. Since each crystal has its own characteristics, the user should consult the crystal manufacturer for appropriate values of external components.

**Crystals to be Characterized:**

32.768 kHz	Epson C-001R32.768K-A	± 20 PPM
100 kHz	Epson C-2 100.00 KC-P	± 20 PPM
200 kHz	STD XTL 200.000 kHz	± 20 PPM
2.0 MHz	ECS ECS-20-S-2	± 50 PPM
4.0 MHz	ECS ECS-40-S-4	± 50 PPM
10.0 MHz	ECS ECS-100-S-4	± 50 PPM
20.0 MHz	ECS ECS-200-S-4	± 50 PPM

# PIC16C62X

## 9.2.3 EXTERNAL CRYSTAL OSCILLATOR CIRCUIT

Either a prepackaged oscillator can be used or a simple oscillator circuit with TTL gates can be built. Prepackaged oscillators provide a wide operating range and better stability. A well-designed crystal oscillator will provide good performance with TTL gates. Two types of crystal oscillator circuits can be used; one with series resonance, or one with parallel resonance.

Figure 9-4 shows implementation of a parallel resonant oscillator circuit. The circuit is designed to use the fundamental frequency of the crystal. The 74AS04 inverter performs the 180° phase shift that a parallel oscillator requires. The 4.7 kΩ resistor provides the negative feedback for stability. The 10 kΩ potentiometers bias the 74AS04 in the linear region. This could be used for external oscillator designs.

**FIGURE 9-4: EXTERNAL PARALLEL RESONANT CRYSTAL OSCILLATOR CIRCUIT**

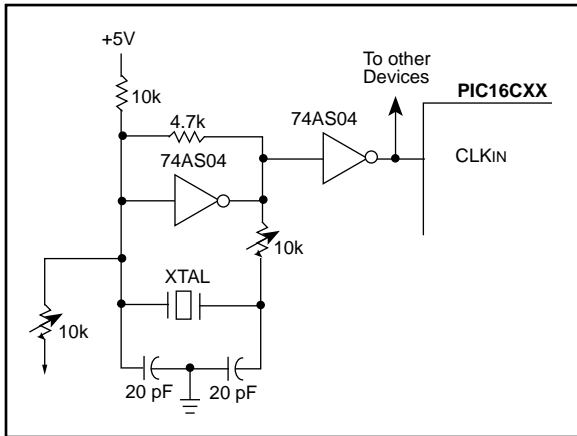
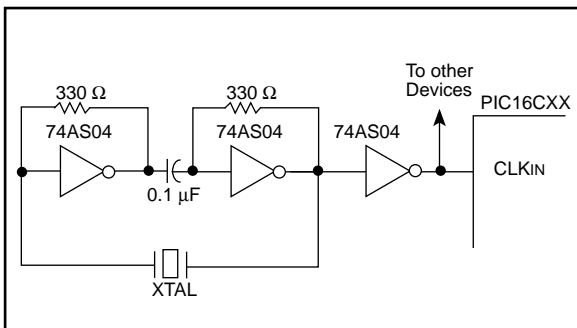


Figure 9-5 shows a series resonant oscillator circuit. This circuit is also designed to use the fundamental frequency of the crystal. The inverter performs a 180° phase shift in a series resonant oscillator circuit. The 330 Ω resistors provide the negative feedback to bias the inverters in their linear region.

**FIGURE 9-5: EXTERNAL SERIES RESONANT CRYSTAL OSCILLATOR CIRCUIT**



## 9.2.4 RC OSCILLATOR

For timing insensitive applications the “RC” device option offers additional cost savings. The RC oscillator frequency is a function of the supply voltage, the resistor ( $R_{ext}$ ) and capacitor ( $C_{ext}$ ) values, and the operating temperature. In addition to this, the oscillator frequency will vary from unit to unit due to normal process parameter variation. Furthermore, the difference in lead frame capacitance between package types will also affect the oscillation frequency, especially for low  $C_{ext}$  values. The user also needs to take into account variation due to tolerance of external R and C components used. Figure 9-6 shows how the R/C combination is connected to the PIC16CXX. For  $R_{ext}$  values below 2.2 kΩ, the oscillator operation may become unstable, or stop completely. For very high  $R_{ext}$  values (e.g., 1 MΩ), the oscillator becomes sensitive to noise, humidity and leakage. Thus, we recommend to keep  $R_{ext}$  between 3 kΩ and 100 kΩ.

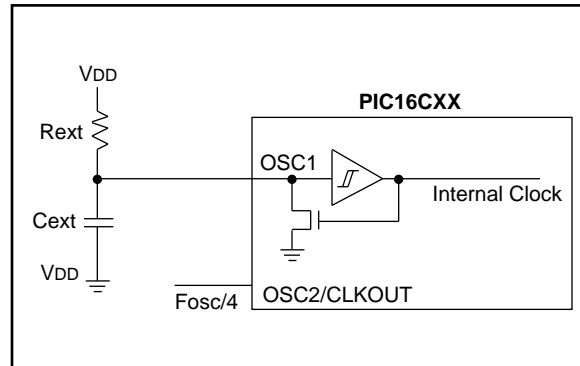
Although the oscillator will operate with no external capacitor ( $C_{ext} = 0$  pF), we recommend using values above 20 pF for noise and stability reasons. With no or small external capacitance, the oscillation frequency can vary dramatically due to changes in external capacitances, such as PCB trace capacitance or package lead frame capacitance.

See Section 13.0 for RC frequency variation from part to part due to normal process variation. The variation is larger for larger R (since leakage current variation will affect RC frequency more for large R) and for smaller C (since variation of input capacitance will affect RC frequency more).

See Section 13.0 for variation of oscillator frequency due to  $V_{DD}$  for given  $R_{ext}/C_{ext}$  values as well as frequency variation due to operating temperature for given R, C, and  $V_{DD}$  values.

The oscillator frequency, divided by 4, is available on the OSC2/CLKOUT pin, and can be used for test purposes or to synchronize other logic (Figure 3-2 for waveform).

**FIGURE 9-6: RC OSCILLATOR MODE**



## 9.3 Reset

The PIC16CXX differentiates between various kinds of reset:

- a) Power-on reset (POR)
- b)  $\overline{\text{MCLR}}$  reset during normal operation
- c)  $\overline{\text{MCLR}}$  reset during SLEEP
- d) WDT reset (normal operation)
- e) WDT wake-up (SLEEP)
- f) Brown-out Reset (BOR)

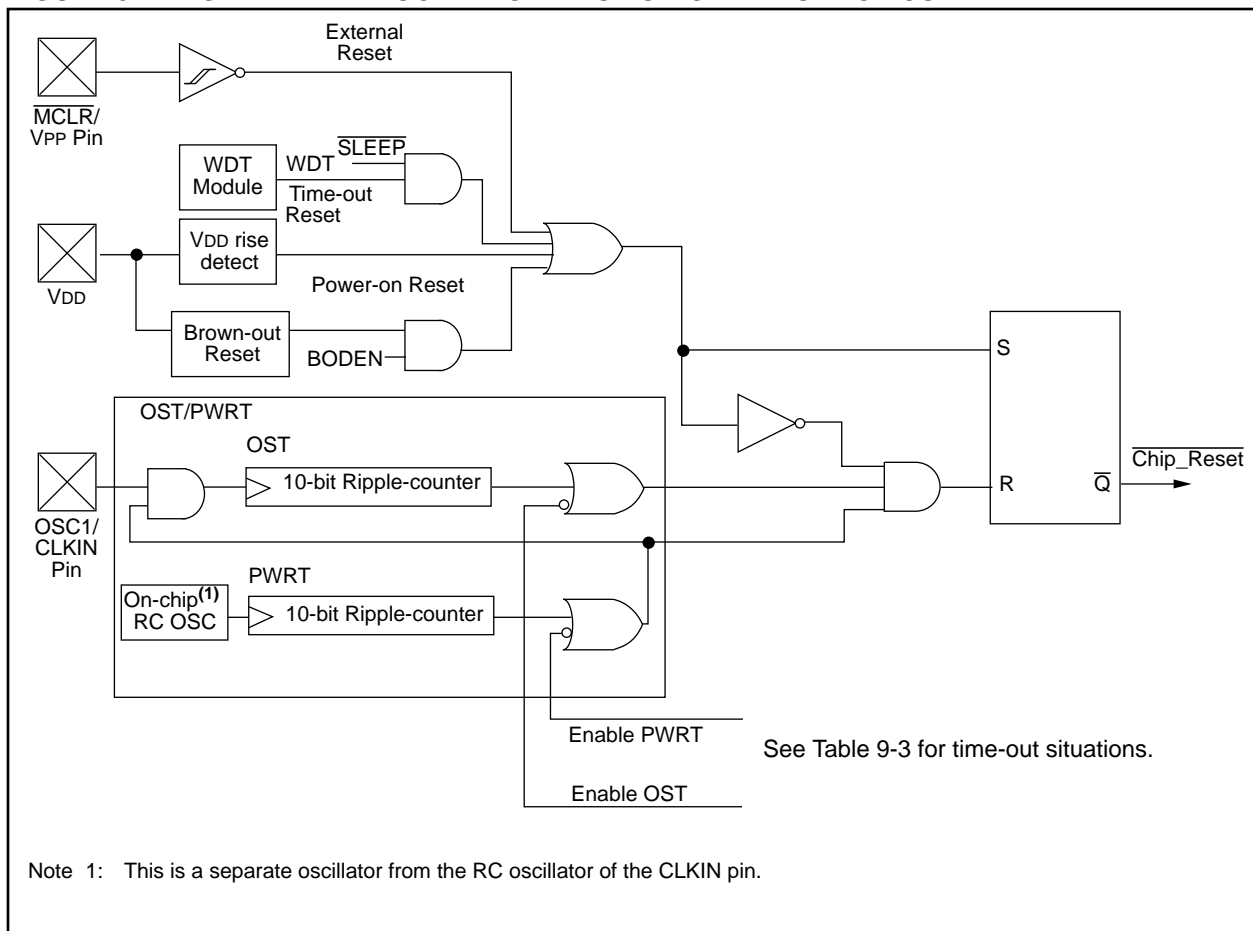
Some registers are not affected in any reset condition; their status is unknown on POR and unchanged in any other reset. Most other registers are reset to a "reset

state" on Power-on reset, on  $\overline{\text{MCLR}}$  or WDT reset and on  $\overline{\text{MCLR}}$  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 9-4. These bits are used in software to determine the nature of the reset. See Table 9-6 for a full description of reset states of all registers.

A simplified block diagram of the on-chip reset circuit is shown in Figure 9-7.

The  $\overline{\text{MCLR}}$  reset path has a noise filter to detect and ignore small pulses. See Table 12-6 for pulse width specification.

**FIGURE 9-7: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT**



# PIC16C62X

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

### 9.4.1 POWER-ON RESET (POR)

A Power-on Reset pulse is generated on-chip when VDD rise is detected (in the range of 1.6 V – 1.8 V). To take advantage of the POR, just tie the  $\overline{\text{MCLR}}$  pin directly (or through a resistor) to VDD. This will eliminate external RC components usually needed to create Power-on Reset. A maximum rise time for VDD is required. See Electrical Specifications for details.

The POR circuit does not produce internal reset when VDD 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".

### 9.4.2 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 VDD to rise to an acceptable level. A configuration bit,  $\overline{\text{PWRTE}}$  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 VDD, temperature and process variation. See DC parameters for details.

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

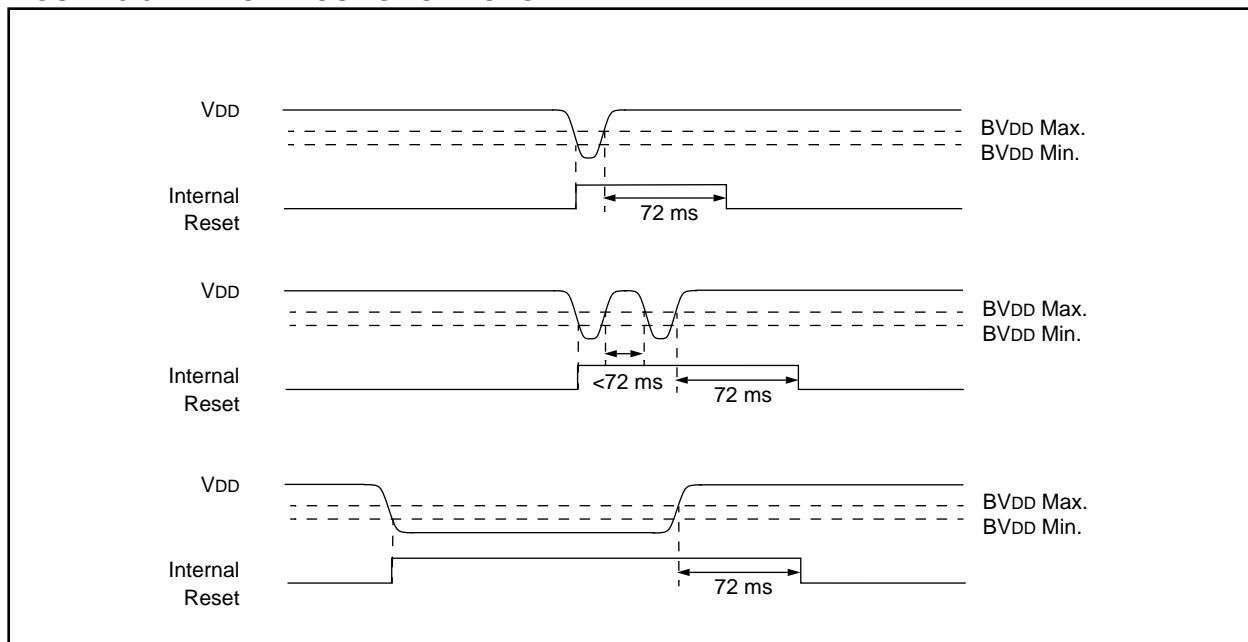
The Oscillator Start-Up Timer (OST) provides a 1024 oscillator cycle (from OSC1 input) delay after the PWRT delay is over. This ensures that the crystal oscillator or resonator has started and stabilized.

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

### 9.4.4 BROWN-OUT RESET (BOR)

The PIC16C62X members have on-chip Brown-out Reset circuitry. A configuration bit, BODEN, can disable (if clear/programmed) or enable (if set) the Brown-out Reset circuitry. If VDD falls below 4.0V (3.8V – 4.2V range) for greater than parameter 35 in Table 12-6, the brown-out situation will reset the chip. A reset is not guaranteed to occur if VDD falls below 4.0V for less than parameter 35. The chip will remain in Brown-out Reset until VDD rises above BVDD. The Power-up Timer will now be invoked and will keep the chip in reset an additional 72 ms. If VDD drops below BVDD while the Power-up Timer is running, the chip will go back into a Brown-out Reset and the Power-up Timer will be initialized. Once VDD rises above BVDD, 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 9-8 shows typical Brown-out situations.

FIGURE 9-8: BROWN-OUT SITUATIONS





## 9.4.5 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  $\overline{\text{PWRTE}}$  bit status. For example, in RC mode with  $\overline{\text{PWRTE}}$  bit erased (PWRT disabled), there will be no time-out at all. Figure 9-9, Figure 9-10 and Figure 9-11 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 9-10). This is useful for testing purposes or to synchronize more than one PIC16C62X device operating in parallel.

Table 9-5 shows the reset conditions for some special registers, while Table 9-6 shows the reset conditions for all the registers.

## 9.4.6 POWER CONTROL/STATUS REGISTER (PCON)

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

Bit0 is  $\overline{\text{BO}}$  (Brown-out).  $\overline{\text{BO}}$  is unknown on power-on-reset. It must then be set by the user and checked on subsequent resets to see if  $\overline{\text{BO}} = 0$  indicating that a brown-out has occurred. The  $\overline{\text{BO}}$  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 9-3: TIME-OUT IN VARIOUS SITUATIONS**

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

**TABLE 9-4: STATUS 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	1	WDT Reset
1	1	0	0	WDT Wake-up
1	1	1	1	$\overline{\text{MCLR}}$ reset during normal operation
1	1	1	0	$\overline{\text{MCLR}}$ reset during SLEEP

# PIC16C62X

**TABLE 9-5: INITIALIZATION CONDITION FOR SPECIAL REGISTERS**

Condition	Program Counter	STATUS Register	PCON Register
Power-on Reset	000h	0001 1xxx	---- --0x
MCLR reset during normal operation	000h	0001 1uuu	---- --uu
MCLR reset during SLEEP	000h	0001 0uuu	---- --uu
WDT reset	000h	0000 1uuu	---- --uu
WDT Wake-up	PC + 1	uuu0 0uuu	---- --uu
Brown-out Reset	000h	0001 1uuu	---- --u0
Interrupt Wake-up from SLEEP	PC + 1 <sup>(1)</sup>	uuu1 0uuu	---- --uu

Legend: u = unchanged, x = unknown, - = unimplemented bit, reads as '0'.

Note 1: When the wake-up is due to an interrupt and global enable bit, GIE is set, the PC is loaded with the interrupt vector (0004h) after execution of PC+1.

**TABLE 9-6: INITIALIZATION CONDITION FOR REGISTERS**

Register	Address	Power-on Reset	<ul style="list-style-type: none"> <li>• MCLR Reset during normal operation</li> <li>• MCLR Reset during SLEEP</li> <li>• WDT Reset</li> <li>• Brown-out Reset <sup>(1)</sup></li> </ul>	<ul style="list-style-type: none"> <li>• Wake up from SLEEP through interrupt</li> <li>• Wake up from SLEEP through WDT time-out</li> </ul>
W	-	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF	00h	-	-	-
TMRO	01h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	02h	0000 0000	0000 0000	PC + 1 <sup>(3)</sup>
STATUS	03h	0001 1xxx	000q quuu <sup>(4)</sup>	uuuq quuu <sup>(4)</sup>
FSR	04h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA	05h	---x xxxx	---u uuuu	---u uuuu
PORTB	06h	xxxx xxxx	uuuu uuuu	uuuu uuuu
CMCON	1Fh	00-- 0000	00-- 0000	uu-- uuuu
PCLATH	0Ah	---0 0000	---0 0000	---u uuuu
INTCON	0Bh	0000 000x	0000 000x	uuuu uuuu <sup>(2)</sup>
PIR1	0Ch	-0-- ----	-0-- ----	-u-- ---- <sup>(2)</sup>
OPTION	81h	1111 1111	1111 1111	uuuu uuuu
TRISA	85h	---1 1111	---1 1111	---u uuuu
TRISB	86h	1111 1111	1111 1111	uuuu uuuu
PIE1	8Ch	-0-- ----	-0-- ----	-u-- ----
PCON	8Eh	---- --0x	---- --uq <sup>(1)</sup>	---- --uu
VRCON	9Fh	000- 0000	000- 0000	uuu- uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, reads as '0', q = value depends on condition.

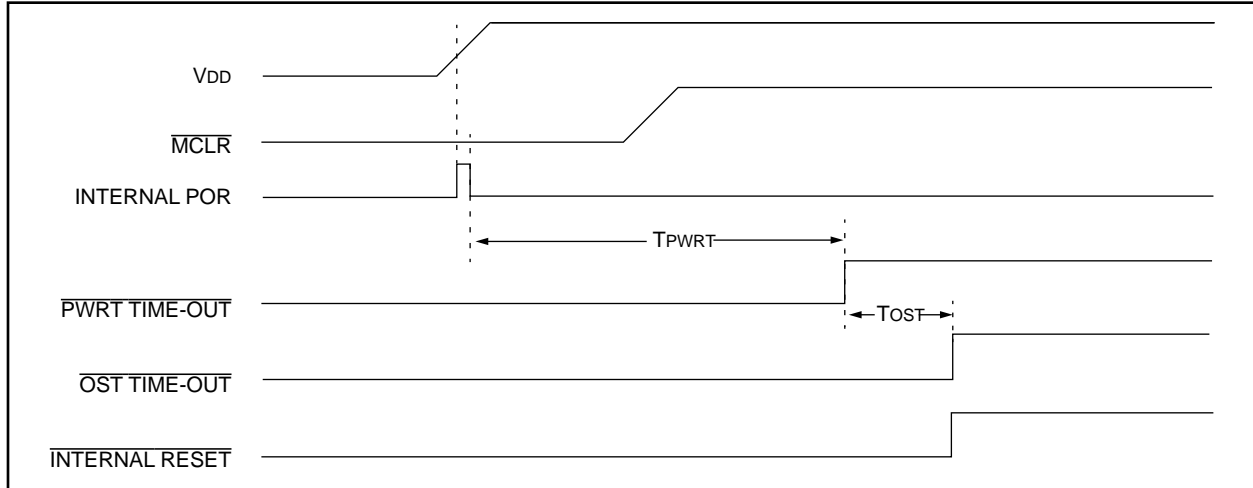
Note 1: If VDD goes too low, Power-on Reset will be activated and registers will be affected differently.

2: One or more bits in INTCON, PIR1 and/or PIR2 will be affected (to cause wake-up).

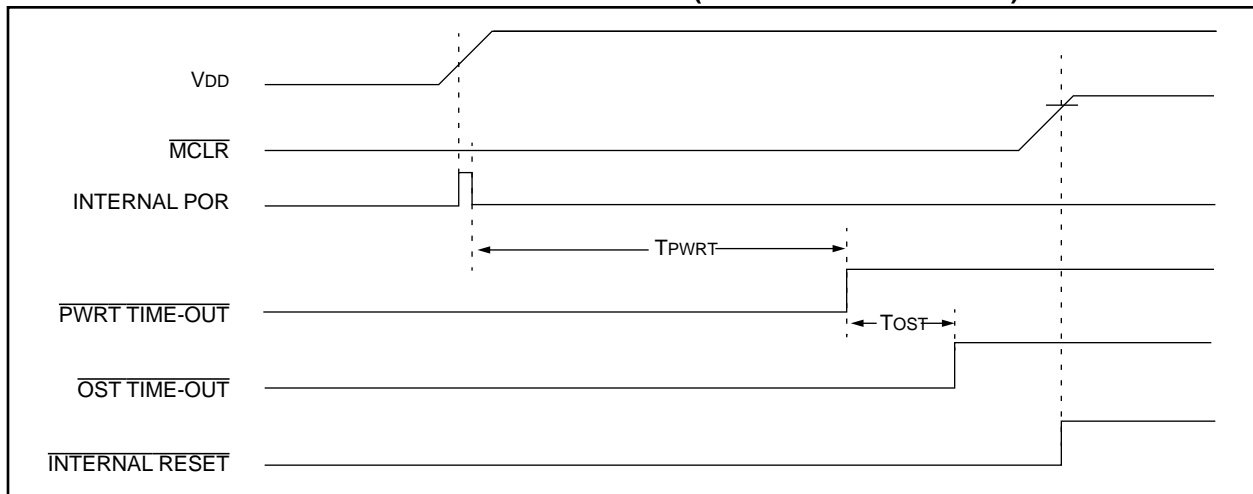
3: When the wake-up is due to an interrupt and the GIE bit is set, the PC is loaded with the interrupt vector (0004h).

4: See Table 9-5 for reset value for specific condition.

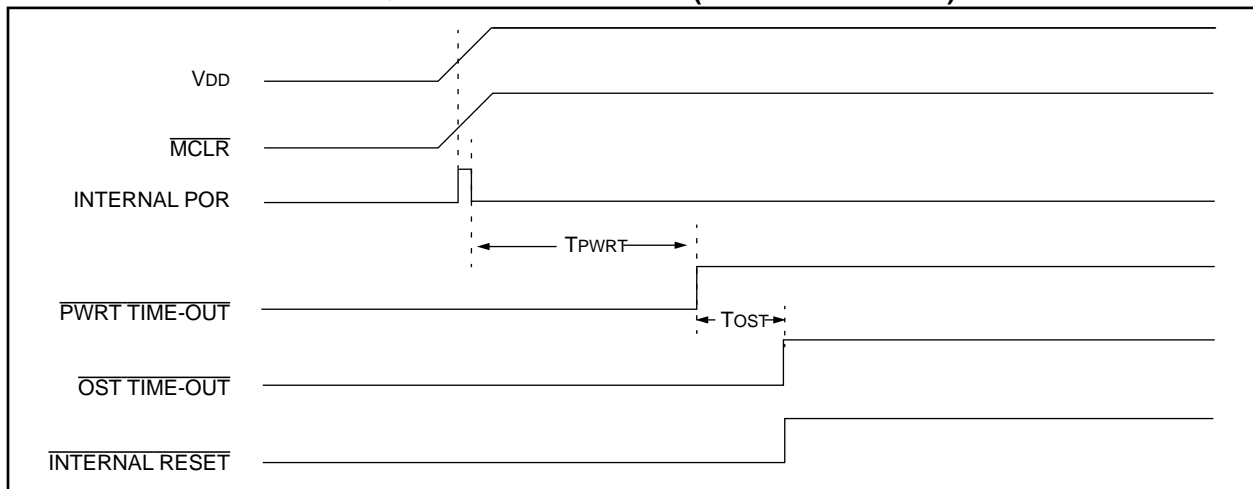
**FIGURE 9-9: TIME-OUT SEQUENCE ON POWER-UP ( $\overline{\text{MCLR}}$  NOT TIED TO  $V_{DD}$ ): CASE 1**



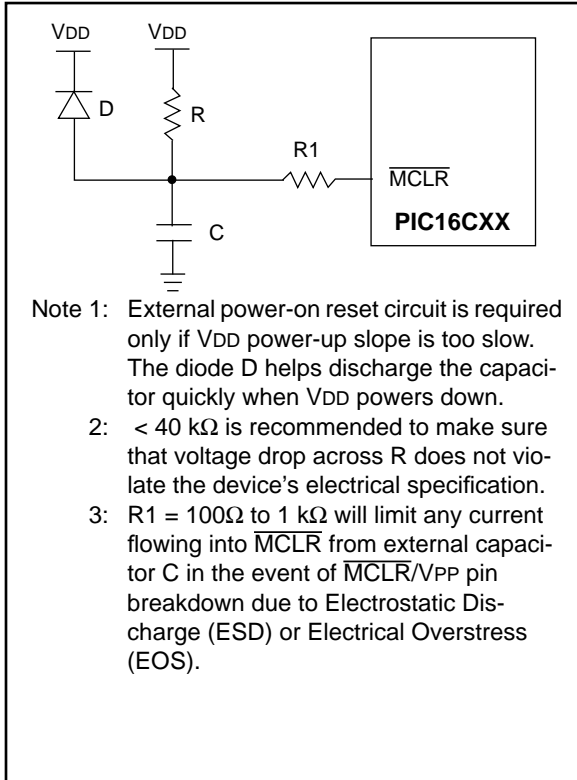
**FIGURE 9-10: TIME-OUT SEQUENCE ON POWER-UP ( $\overline{\text{MCLR}}$  NOT TIED TO  $V_{DD}$ ): CASE 2**



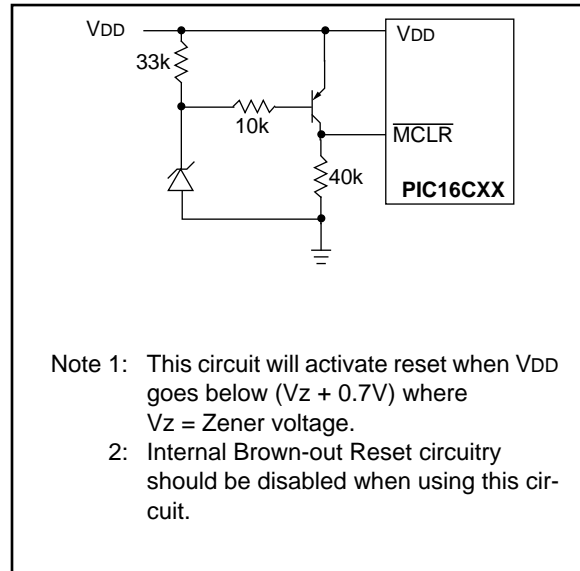
**FIGURE 9-11: TIME-OUT SEQUENCE ON POWER-UP ( $\overline{\text{MCLR}}$  TIED TO  $V_{DD}$ )**



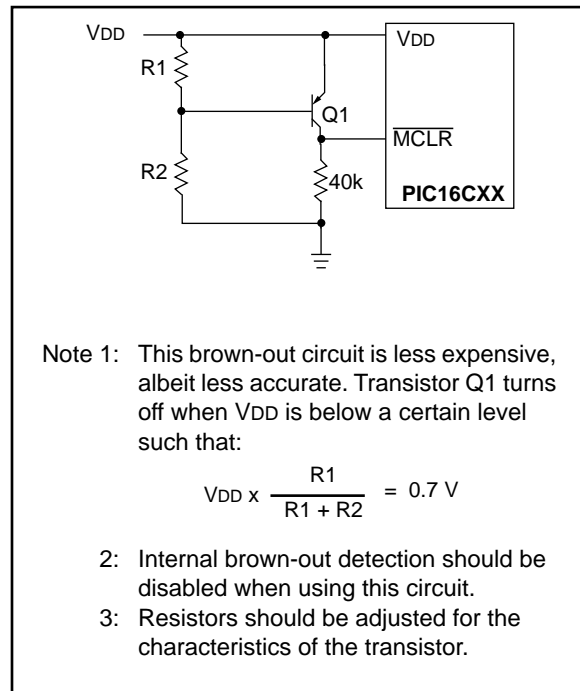
**FIGURE 9-12: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW V<sub>DD</sub> POWER-UP)**



**FIGURE 9-13: EXTERNAL BROWN-OUT PROTECTION CIRCUIT 1**



**FIGURE 9-14: EXTERNAL BROWN-OUT PROTECTION CIRCUIT 2**



## 9.5 Interrupts

The PIC16C62X has 4 sources of interrupt:

- External interrupt RB0/INT
- TMR0 overflow interrupt
- PortB change interrupts (pins RB7:RB4)
- 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

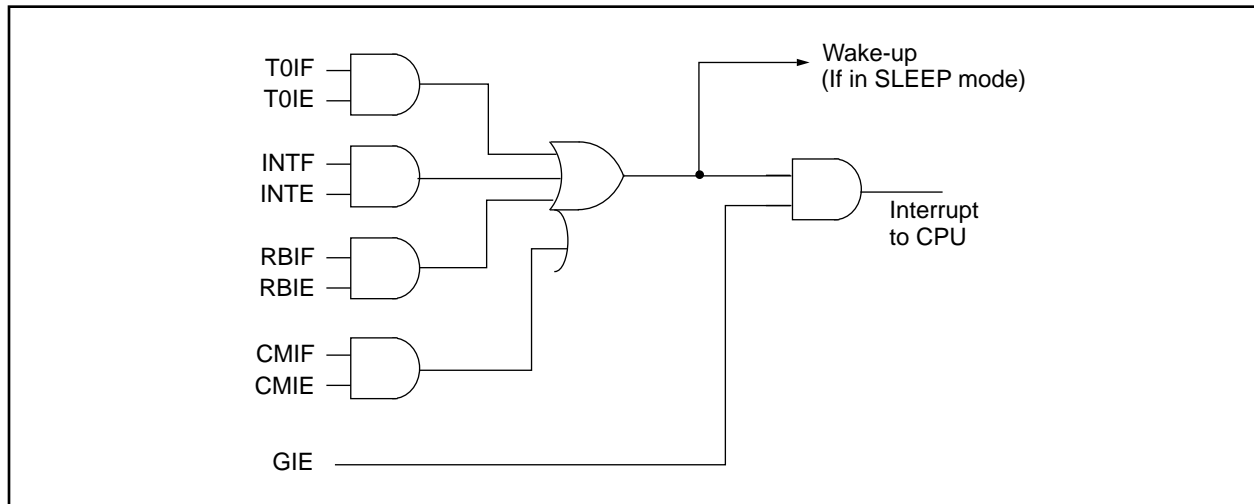
the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid RB0/INT recursive interrupts.

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 when the interrupt event occurs (Figure 9-16). 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. Individual interrupt flag bits are set regardless of the status of their corresponding mask bit or the GIE bit.

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

**2:** In 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 9-15: INTERRUPT LOGIC**



# PIC16C62X

## 9.5.1 RB0/INT INTERRUPT

External interrupt on RB0/INT pin is edge triggered: either rising if INTEDG bit (OPTION<6>) is set, or falling, if INTEDG bit is clear. When a valid edge appears on the RB0/INT pin, the INTF bit (INTCON<1>) is set. This interrupt can be disabled by clearing the INTE control bit (INTCON<4>). The INTF bit must be cleared in software in the interrupt service routine before re-enabling this interrupt. The RB0/INT interrupt can wake-up the processor from SLEEP, if the INTE bit was set prior to going into SLEEP. The status of the GIE bit decides whether or not the processor branches to the interrupt vector following wake-up. See Section 9.8 for details on SLEEP and Figure 9-19 for timing of wake-up from SLEEP through RB0/INT interrupt.

## 9.5.2 TMR0 INTERRUPT

An overflow (FFh → 00h) in the TMR0 register will set the T0IF (INTCON<2>) bit. The interrupt can be enabled/disabled by setting/clearing T0IE (INTCON<5>) bit. For operation of the Timer0 module, see Section 6.0.

## 9.5.3 PORTB INTERRUPT

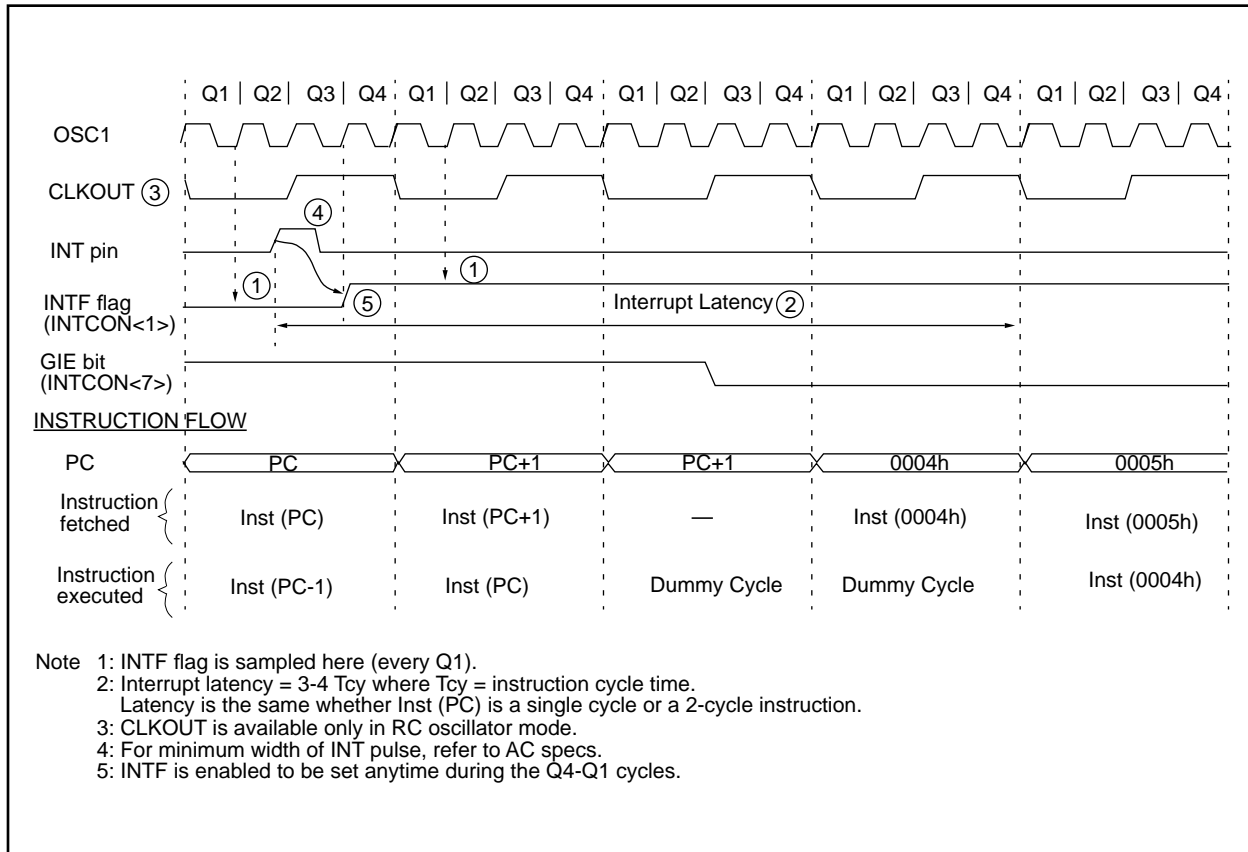
An input change on PORTB <7:4> sets the RBIF (INTCON<0>) bit. The interrupt can be enabled/disabled by setting/clearing the RBIE (INTCON<4>) bit. For operation of PORTB (Section 5.2).

**Note:** If a change on the I/O pin should occur when the read operation is being executed (start of the Q2 cycle), then the RBIF interrupt flag may get set.

## 9.5.4 COMPARATOR INTERRUPT

See Section 7.6 for complete description of comparator interrupts.

**FIGURE 9-16: INT PIN INTERRUPT TIMING**



## 9.6 Context Saving During Interrupts

During an interrupt, only the return PC value is saved on the stack. Typically, users may wish to save key registers during an interrupt e.g. W register and STATUS register. This will have to be implemented in software.

Example 9-1 stores and restores the STATUS and W registers. The user register, W\_TEMP, must be defined in both banks and must be defined at the same offset from the bank base address (i.e., W\_TEMP is defined at 0x20 in Bank 0 and it must also be defined at 0xA0 in Bank 1). The user register, STATUS\_TEMP, must be defined in Bank 0. The Example 9-1:

- Stores the W register
- Stores the STATUS register in Bank 0
- Executes the ISR code
- Restores the STATUS (and bank select bit register)
- Restores the W register

### EXAMPLE 9-1: SAVING THE STATUS AND W REGISTERS IN RAM

```

MOVWF    W_TEMP        ;copy W to temp register,
                       ;could be in either bank

SWAPF    STATUS,W      ;swap status to be saved into W

BCF      STATUS,RP0    ;change to bank 0 regardless
                       ;of current bank

MOVWF    STATUS_TEMP    ;save status to bank 0
                       ;register

:
:   (ISR)
:

SWAPF    STATUS_TEMP,W  ;swap STATUS_TEMP register
                       ;into W, sets bank to original
                       ;state

MOVWF    STATUS         ;move W into STATUS register

SWAPF    W_TEMP,F      ;swap W_TEMP

SWAPF    W_TEMP,W      ;swap W_TEMP into W
    
```

## 9.7 Watchdog Timer (WDT)

The watchdog timer is a free running on-chip RC oscillator which does not require any external components. This RC oscillator is separate from the RC oscillator of the CLKIN pin. That means that the WDT will run, even if the clock on the OSC1 and OSC2 pins of the device has been stopped, for example, by execution of a SLEEP instruction. During normal operation, a WDT time-out generates a device RESET. If the device is in SLEEP mode, a WDT time-out causes the device to wake-up and continue with normal operation. The WDT can be permanently disabled by programming the configuration bit WDTE as clear (Section 9.1).

### 9.7.1 WDT PERIOD

The WDT has a nominal time-out period of 18 ms, (with no prescaler). The time-out periods vary with temperature, VDD and process variations from part to part (see DC specs). If longer time-out periods are desired, a prescaler with a division ratio of up to 1:128 can be assigned to the WDT under software control by writing to the OPTION register. Thus, time-out periods up to 2.3 seconds can be realized.

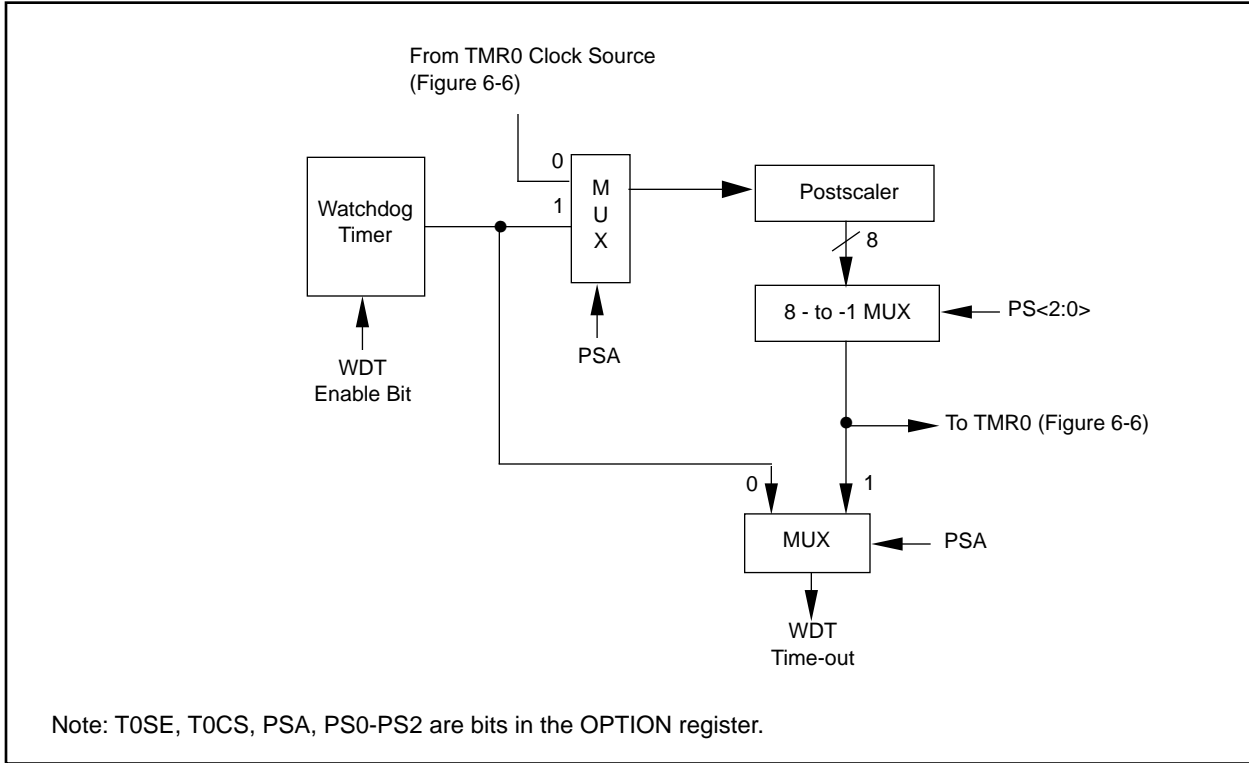
The CLRWDT and SLEEP instructions clear the WDT and the postscaler, if assigned to the WDT, and prevent it from timing out and generating a device RESET.

The  $\overline{TO}$  bit in the STATUS register will be cleared upon a Watchdog Timer time-out.

### 9.7.2 WDT PROGRAMMING CONSIDERATIONS

It should also be taken in account that under worst case conditions (VDD = Min., Temperature = Max., max. WDT prescaler) it may take several seconds before a WDT time-out occurs.

**FIGURE 9-17: WATCHDOG TIMER BLOCK DIAGRAM**



**FIGURE 9-18: SUMMARY OF WATCHDOG TIMER REGISTERS**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
2007h	Config. bits	(1)	BODEN <sup>(1)</sup>	CP1	CP0	PWRTE	WDTE	FOSC1	FOSC0
81h	OPTION	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

Legend: Shaded cells are not used by the Watchdog Timer.



## 9.8 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 (VIHMC).

**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 a 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 wakeup from sleep. The sleep instruction is completely executed.

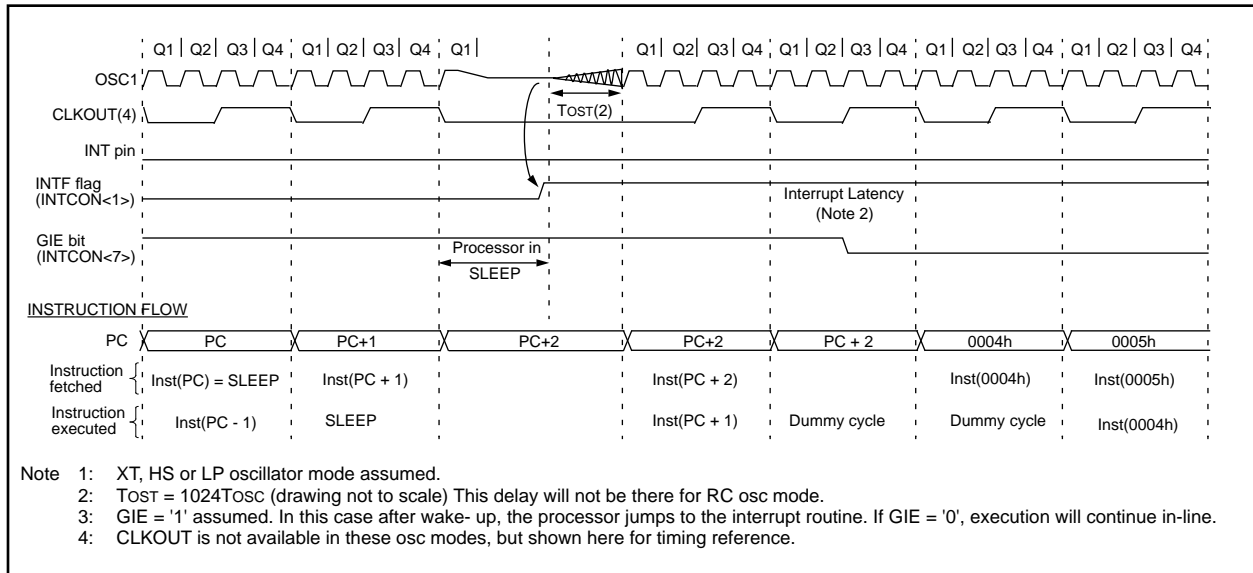
### 9.8.1 WAKE-UP FROM SLEEP

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 9-19: WAKE-UP FROM SLEEP THROUGH INTERRUPT**



# PIC16C62X

## 9.9 Code Protection

If the code protection bit(s) have not been programmed, the on-chip program memory can be read out for verification purposes.

**Note:** Microchip does not recommend code protecting windowed devices.

## 9.10 ID Locations

Four memory locations (2000h-2003h) are designated as ID locations where the user can store checksum or other code-identification numbers. These locations are not accessible during normal execution but are readable and writable during program/verify. Only the least significant 4 bits of the ID locations are used.

## 9.11 In-Circuit Serial Programming

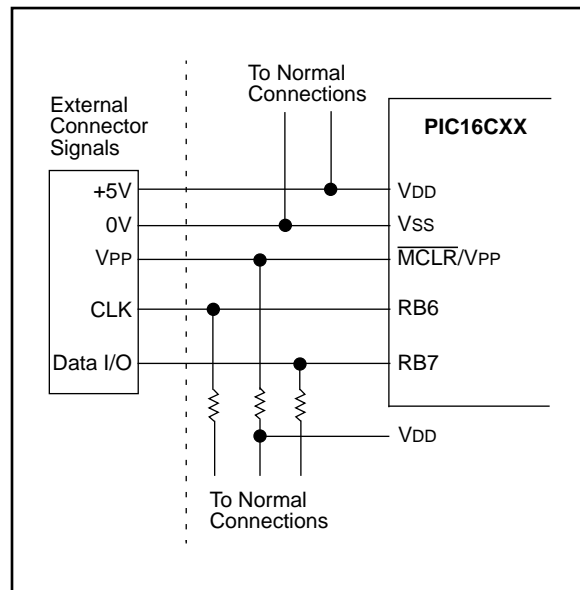
The PIC16CXX microcontrollers can be serially programmed while in the end application circuit. This is simply done with two lines for clock and data, and three other lines for power, ground, and the programming voltage. This allows customers to manufacture boards with unprogrammed devices, and then program the microcontroller just before shipping the product. This also allows the most recent firmware or a custom firmware to be programmed.

The device is placed into a program/verify mode by holding the RB6 and RB7 pins low while raising the  $\overline{\text{MCLR}}$  ( $V_{PP}$ ) pin from  $V_{IL}$  to  $V_{IH}$  (see programming specification). RB6 becomes the programming clock and RB7 becomes the programming data. Both RB6 and RB7 are Schmitt Trigger inputs in this mode.

After reset, to place the device into programming/verify mode, the program counter (PC) is at location 00h. A 6-bit command is then supplied to the device. Depending on the command, 14-bits of program data are then supplied to or from the device, depending if the command was a load or a read. For complete details of serial programming, please refer to the PIC16C6X/7X Programming Specifications (Literature #DS30228).

A typical in-circuit serial programming connection is shown in Figure 9-20.

**FIGURE 9-20: TYPICAL IN-CIRCUIT SERIAL PROGRAMMING CONNECTION**



## 10.0 INSTRUCTION SET SUMMARY

Each PIC16CXX instruction is a 14-bit word divided into an OPCODE which specifies the instruction type and one or more operands which further specify the operation of the instruction. The PIC16CXX instruction set summary in Table 10-2 lists **byte-oriented**, **bit-oriented**, and **literal and control** operations. Table 10-1 shows the opcode field descriptions.

For **byte-oriented** instructions, 'f' represents a file register designator and 'd' represents a destination designator. The file register designator specifies which file register is to be used by the instruction.

The destination designator specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in the W register. If 'd' is one, the result is placed in the file register specified in the instruction.

For **bit-oriented** instructions, 'b' represents a bit field designator which selects the number of the bit affected by the operation, while 'f' represents the number of the file in which the bit is located.

For **literal and control** operations, 'k' represents an eight or eleven bit constant or literal value.

**TABLE 10-1: OPCODE FIELD DESCRIPTIONS**

Field	Description
f	Register file address (0x00 to 0x7F)
w	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1) The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1
label	Label name
TOS	Top of Stack
PC	Program Counter
PCLATH	Program Counter High Latch
GIE	Global Interrupt Enable bit
WDT	Watchdog Timer/Counter
TO	Time-out bit
PD	Power-down bit
dest	Destination either the W register or the specified register file location
[ ]	Options
( )	Contents
→	Assigned to
< >	Register bit field
∈	In the set of
<i>italics</i>	User defined term (font is courier)

The instruction set is highly orthogonal and is grouped into three basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal and control** operations

All instructions are executed within one single instruction cycle, unless a conditional test is true or the program counter is changed as a result of an instruction. In this case, the execution takes two instruction cycles with the second cycle executed as a NOP. One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 μs. If a conditional test is true or the program counter is changed as a result of an instruction, the instruction execution time is 2 μs.

Table 10-1 lists the instructions recognized by the MPASM assembler.

Figure 10-1 shows the three general formats that the instructions can have.

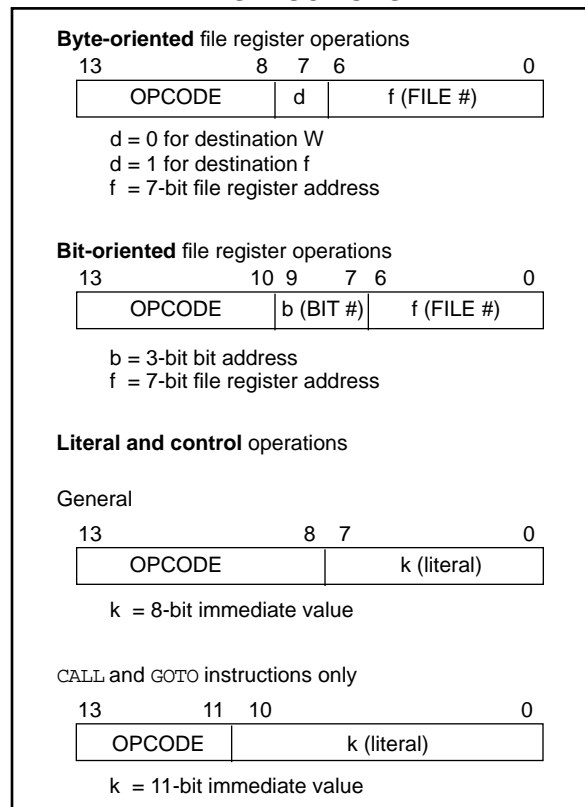
**Note:** To maintain upward compatibility with future PIC16CXX products, do not use the `OPTION` and `TRIS` instructions.

All examples use the following format to represent a hexadecimal number:

0xhh

where h signifies a hexadecimal digit.

**FIGURE 10-1: GENERAL FORMAT FOR INSTRUCTIONS**



# PIC16C62X

**TABLE 10-2: PIC16CXX INSTRUCTION SET**

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

## 10.1 Instruction Descriptions

<b>ADDLW</b>	<b>Add Literal and W</b>				
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	ADDLW 0x15 Before Instruction W = 0x10 After Instruction W = 0x25				

<b>ANDLW</b>	<b>AND Literal with W</b>				
Syntax:	[ <i>label</i> ] ANDLW k				
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	ANDLW 0x5F Before Instruction W = 0xA3 After Instruction W = 0x03				

<b>ADDWF</b>	<b>Add W and f</b>				
Syntax:	[ <i>label</i> ] ADDWF f,d				
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	ADDWF FSR, 0 Before Instruction W = 0x17 FSR = 0xC2 After Instruction W = 0xD9 FSR = 0xC2				

<b>ANDWF</b>	<b>AND W with f</b>				
Syntax:	[ <i>label</i> ] ANDWF f,d				
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	ANDWF FSR, 1 Before Instruction W = 0x17 FSR = 0xC2 After Instruction W = 0x17 FSR = 0x02				

# PIC16C62X

## BCF Bit Clear f

Syntax: [ *label* ] BCF f,b  
Operands:  $0 \leq f \leq 127$   
 $0 \leq b \leq 7$   
Operation:  $0 \rightarrow (f<b>)$   
Status Affected: None  
Encoding: 

01	00bb	bfff	ffff
----	------	------	------

  
Description: Bit 'b' in register 'f' is cleared.  
Words: 1  
Cycles: 1  
Example `BCF FLAG_REG, 7`

Before Instruction  
FLAG\_REG = 0xC7  
After Instruction  
FLAG\_REG = 0x47

## BTFSC Bit Test, Skip if Clear

Syntax: [ *label* ] BTFSC f,b  
Operands:  $0 \leq f \leq 127$   
 $0 \leq b \leq 7$   
Operation: skip if (f<b>) = 0  
Status Affected: None  
Encoding: 

01	10bb	bfff	ffff
----	------	------	------

Description: If bit 'b' in register 'f' is '0' then the next instruction is skipped.  
If bit 'b' is '0' 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 BTFSC FLAG,1`  
`FALSE GOTO PROCESS_CODE`  
`TRUE`  
`•`  
`•`  
`•`

Before Instruction  
PC = address HERE  
After Instruction  
if FLAG<1> = 0,  
PC = address TRUE  
if FLAG<1> = 1,  
PC = address FALSE

## BSF Bit Set f

Syntax: [ *label* ] BSF f,b  
Operands:  $0 \leq f \leq 127$   
 $0 \leq b \leq 7$   
Operation:  $1 \rightarrow (f<b>)$   
Status Affected: None  
Encoding: 

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

  
Description: Bit 'b' in register 'f' is set.  
Words: 1  
Cycles: 1  
Example `BSF FLAG_REG, 7`

Before Instruction  
FLAG\_REG = 0x0A  
After Instruction  
FLAG\_REG = 0x8A

## **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<b>) = 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    BTFSC   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	0xxx	xxxx
----	------	------	------

**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

# PIC16C62X

## 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



**GOTO Unconditional Branch**

Syntax: `[label] GOTO k`

Operands:  $0 \leq k \leq 2047$

Operation:  $k \rightarrow PC<10:0>$   
 $PCLATH<4:3> \rightarrow PC<12:11>$

Status Affected: None

Encoding: 

10	1kkk	kkkk	kkkk
----	------	------	------

Description: GOTO is an unconditional branch. The eleven bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a two-cycle instruction.

Words: 1

Cycles: 2

Example `GOTO THERE`

After Instruction  
 $PC = \text{Address } THERE$

**INCFSZ Increment f, Skip if 0**

Syntax: `[label] INCFSZ f,d`

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

Operation:  $(f) + 1 \rightarrow (\text{dest})$ , skip if result = 0

Status Affected: None

Encoding: 

00	1111	dfff	ffff
----	------	------	------

Description: The contents of register 'f' are incremented. 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      INCFSZ   CNT, 1
          GOTO     LOOP
CONTINUE  .
          .
          .
    
```

Before Instruction  
 $PC = \text{address } HERE$

After Instruction  
 $CNT = CNT + 1$   
if  $CNT = 0$ ,  
 $PC = \text{address } CONTINUE$   
if  $CNT \neq 0$ ,  
 $PC = \text{address } HERE + 1$

**INCF Increment f**

Syntax: `[label] INCF f,d`

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

Operation:  $(f) + 1 \rightarrow (\text{dest})$

Status Affected: Z

Encoding: 

00	1010	dfff	ffff
----	------	------	------

Description: The contents of register 'f' are incremented. 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 `INCF CNT, 1`

Before Instruction  
 $CNT = 0xFF$   
 $Z = 0$

After Instruction  
 $CNT = 0x00$   
 $Z = 1$

**IORLW Inclusive OR Literal with W**

Syntax: `[label] IORLW k`

Operands:  $0 \leq k \leq 255$

Operation:  $(W) .OR. k \rightarrow (W)$

Status Affected: Z

Encoding: 

11	1000	kkkk	kkkk
----	------	------	------

Description: The contents of the W register is OR'ed with the eight bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Example `IORLW 0x35`

Before Instruction  
 $W = 0x9A$

After Instruction  
 $W = 0xBF$   
 $Z = 1$

# PIC16C62X

<b>IORWF</b>	<b>Inclusive OR W with f</b>				
Syntax:	[ <i>label</i> ] IORWF f,d				
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]				
Operation:	(W) .OR. (f) → (dest)				
Status Affected:	Z				
Encoding:	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="text-align: center;">00</td> <td style="text-align: center;">0100</td> <td style="text-align: center;">dfff</td> <td style="text-align: center;">ffff</td> </tr> </table>	00	0100	dfff	ffff
00	0100	dfff	ffff		
Description:	Inclusive OR the W register with register 'f'. 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	<pre> IORWF      RESULT, 0 Before Instruction     RESULT = 0x13     W      = 0x91 After Instruction     RESULT = 0x13     W      = 0x93     Z      = 1         </pre>				

<b>MOVF</b>	<b>Move f</b>				
Syntax:	[ <i>label</i> ] MOVF f,d				
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]				
Operation:	(f) → (dest)				
Status Affected:	Z				
Encoding:	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="text-align: center;">00</td> <td style="text-align: center;">1000</td> <td style="text-align: center;">dfff</td> <td style="text-align: center;">ffff</td> </tr> </table>	00	1000	dfff	ffff
00	1000	dfff	ffff		
Description:	The contents of register f is moved to a destination dependant upon the status of d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register since status flag Z is affected.				
Words:	1				
Cycles:	1				
Example	<pre> MOVF      FSR, 0 After Instruction     W = value in FSR register     Z = 1         </pre>				

<b>MOVLW</b>	<b>Move Literal to W</b>				
Syntax:	[ <i>label</i> ] MOVLW k				
Operands:	0 ≤ k ≤ 255				
Operation:	k → (W)				
Status Affected:	None				
Encoding:	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="text-align: center;">11</td> <td style="text-align: center;">00xx</td> <td style="text-align: center;">kkkk</td> <td style="text-align: center;">kkkk</td> </tr> </table>	11	00xx	kkkk	kkkk
11	00xx	kkkk	kkkk		
Description:	The eight bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.				
Words:	1				
Cycles:	1				
Example	<pre> MOVLW    0x5A After Instruction     W = 0x5A         </pre>				

<b>MOVWF</b>	<b>Move W to f</b>				
Syntax:	[ <i>label</i> ] MOVWF f				
Operands:	0 ≤ f ≤ 127				
Operation:	(W) → (f)				
Status Affected:	None				
Encoding:	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="text-align: center;">00</td> <td style="text-align: center;">0000</td> <td style="text-align: center;">1fff</td> <td style="text-align: center;">ffff</td> </tr> </table>	00	0000	1fff	ffff
00	0000	1fff	ffff		
Description:	Move data from W register to register 'f'.				
Words:	1				
Cycles:	1				
Example	<pre> MOVWF    OPTION Before Instruction     OPTION = 0xFF     W      = 0x4F After Instruction     OPTION = 0x4F     W      = 0x4F         </pre>				

<b>NOP</b>	<b>No Operation</b>				
Syntax:	[ <i>label</i> ] NOP				
Operands:	None				
Operation:	No operation				
Status Affected:	None				
Encoding:	<table border="1"> <tr> <td>00</td> <td>0000</td> <td>0xxx0</td> <td>0000</td> </tr> </table>	00	0000	0xxx0	0000
00	0000	0xxx0	0000		
Description:	No operation.				
Words:	1				
Cycles:	1				
Example	NOP				

<b>RETFIE</b>	<b>Return from Interrupt</b>				
Syntax:	[ <i>label</i> ] RETFIE				
Operands:	None				
Operation:	TOS → PC, 1 → GIE				
Status Affected:	None				
Encoding:	<table border="1"> <tr> <td>00</td> <td>0000</td> <td>0000</td> <td>1001</td> </tr> </table>	00	0000	0000	1001
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	<pre>RETFIE After Interrupt PC = TOS GIE = 1</pre>				

<b>OPTION</b>	<b>Load Option Register</b>				
Syntax:	[ <i>label</i> ] OPTION				
Operands:	None				
Operation:	(W) → OPTION				
Status Affected:	None				
Encoding:	<table border="1"> <tr> <td>00</td> <td>0000</td> <td>0110</td> <td>0010</td> </tr> </table>	00	0000	0110	0010
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	<div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 0 auto;"> <p><b>To maintain upward compatibility with future PIC16CXX products, do not use this instruction.</b></p> </div>				

<b>RETLW</b>	<b>Return with Literal in W</b>				
Syntax:	[ <i>label</i> ] RETLW k				
Operands:	0 ≤ k ≤ 255				
Operation:	k → (W); TOS → PC				
Status Affected:	None				
Encoding:	<table border="1"> <tr> <td>11</td> <td>01xx</td> <td>kkkk</td> <td>kkkk</td> </tr> </table>	11	01xx	kkkk	kkkk
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	<pre>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</pre> <p><b>Before Instruction</b> W = 0x07</p> <p><b>After Instruction</b> W = value of k8</p>				

# PIC16C62X

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

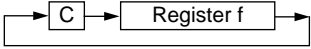
Operation: See description below

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

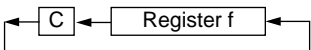
Operation: See description below

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 →  $\overline{TO}$ ,  
0 →  $\overline{PD}$

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

Encoding: 

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

Description: The power-down status bit,  $\overline{PD}$  is cleared. Time-out status bit,  $\overline{TO}$  is set. Watchdog Timer and its prescaler are cleared. The processor is put into SLEEP mode with the oscillator stopped. See Section 9.8 for more details.

Words: 1

Cycles: 1

Example: SLEEP

## **SUBLW**      **Subtract W from Literal**

Syntax:      [ *label* ]    SUBLW    k  
 Operands:     $0 \leq k \leq 255$   
 Operation:     $k - (W) \rightarrow (W)$   
 Status  
 Affected:    C, DC, Z  
 Encoding:    

11	110x	kkkk	kkkk
----	------	------	------

Description:    The W register is subtracted (2's complement method) from the eight bit literal 'k'. The result is placed in the W register.

Words:      1  
 Cycles:      1

Example 1:    SUBLW    0x02  
                   Before Instruction  
                                 W = 1  
                                 C = ?  
                   After Instruction  
                                 W = 1  
                                 C = 1; result is positive

Example 2:    Before Instruction  
                                 W = 2  
                                 C = ?  
                   After Instruction  
                                 W = 0  
                                 C = 1; result is zero

Example 3:    Before Instruction  
                                 W = 3  
                                 C = ?  
                   After Instruction  
                                 W = 0xFF  
                                 C = 0; result is negative

## **SUBWF**      **Subtract W from f**

Syntax:      [ *label* ]    SUBWF    f,d  
 Operands:     $0 \leq f \leq 127$   
                    $d \in [0,1]$   
 Operation:     $(f) - (W) \rightarrow (\text{dest})$   
 Status  
 Affected:    C, DC, Z  
 Encoding:    

00	0010	dfff	ffff
----	------	------	------

Description:    Subtract (2's complement method) W register from 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 1:    SUBWF    REG1,1  
                   Before Instruction  
                                 REG1 = 3  
                                 W = 2  
                                 C = ?  
                   After Instruction  
                                 REG1 = 1  
                                 W = 2  
                                 C = 1; result is positive

Example 2:    Before Instruction  
                                 REG1 = 2  
                                 W = 2  
                                 C = ?  
                   After Instruction  
                                 REG1 = 0  
                                 W = 2  
                                 C = 1; result is zero

Example 3:    Before Instruction  
                                 REG1 = 1  
                                 W = 2  
                                 C = ?  
                   After Instruction  
                                 REG1 = 0xFF  
                                 W = 2  
                                 C = 0; result is negative

# PIC16C62X

SWAPF	Swap Nibbles in f				
Syntax:	[ <i>label</i> ] SWAPF f,d				
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]				
Operation:	(f<3:0>) → (dest<7:4>), (f<7:4>) → (dest<3:0>)				
Status Affected:	None				
Encoding:	<table border="1"> <tr> <td>00</td> <td>1110</td> <td>dfff</td> <td>ffff</td> </tr> </table>	00	1110	dfff	ffff
00	1110	dfff	ffff		
Description:	The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0 the result is placed in W register. If 'd' is 1 the result is placed in register 'f'.				
Words:	1				
Cycles:	1				
Example	<pre>SWAPF REG, 0</pre> <p>Before Instruction</p> <pre>REG1 = 0xA5</pre> <p>After Instruction</p> <pre>REG1 = 0xA5 W     = 0x5A</pre>				

XORLW	Exclusive OR Literal with W				
Syntax:	[ <i>label</i> ] XORLW k				
Operands:	0 ≤ k ≤ 255				
Operation:	(W) .XOR. k → (W)				
Status Affected:	Z				
Encoding:	<table border="1"> <tr> <td>11</td> <td>1010</td> <td>kkkk</td> <td>kkkk</td> </tr> </table>	11	1010	kkkk	kkkk
11	1010	kkkk	kkkk		
Description:	The contents of the W register are XOR'ed with the eight bit literal 'k'. The result is placed in the W register.				
Words:	1				
Cycles:	1				
Example:	<pre>XORLW 0xAF</pre> <p>Before Instruction</p> <pre>W = 0xB5</pre> <p>After Instruction</p> <pre>W = 0x1A</pre>				

TRIS	Load TRIS Register				
Syntax:	[ <i>label</i> ] TRIS f				
Operands:	5 ≤ f ≤ 7				
Operation:	(W) → TRIS register f;				
Status Affected:	None				
Encoding:	<table border="1"> <tr> <td>00</td> <td>0000</td> <td>0110</td> <td>0fff</td> </tr> </table>	00	0000	0110	0fff
00	0000	0110	0fff		
Description:	The instruction is supported for code compatibility with the PIC16C5X products. Since TRIS registers are readable and writable, the user can directly address them.				
Words:	1				
Cycles:	1				
Example	<p><b>To maintain upward compatibility with future PIC16CXX products, do not use this instruction.</b></p>				

XORWF	Exclusive OR W with f				
Syntax:	[ <i>label</i> ] XORWF f,d				
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]				
Operation:	(W) .XOR. (f) → (dest)				
Status Affected:	Z				
Encoding:	<table border="1"> <tr> <td>00</td> <td>0110</td> <td>dfff</td> <td>ffff</td> </tr> </table>	00	0110	dfff	ffff
00	0110	dfff	ffff		
Description:	Exclusive OR 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>XORWF REG 1</pre> <p>Before Instruction</p> <pre>REG = 0xAF W   = 0xB5</pre> <p>After Instruction</p> <pre>REG = 0x1A W   = 0xB5</pre>				

## 11.0 DEVELOPMENT SUPPORT

### 11.1 Development Tools

The PIC16/17 microcontrollers are supported with a full range of hardware and software development tools:

- PICMASTER® Real-Time In-Circuit Emulator
- PRO MATE™ Universal Programmer
- PICSTART® Low-Cost Prototype Programmer
- PICDEM-1 Low-Cost Demonstration Board
- PICDEM-2 Low-Cost Demonstration Board
- MPASM Assembler
- MPSIM Software Simulator
- C Compiler (MP-C)
- Fuzzy Logic Development System (*fuzzyTECH*®-MP)

### 11.2 PICMASTER: High Performance Universal In-Circuit Emulator with MPLAB IDE

The PICMASTER Universal In-Circuit Emulator is intended to provide the product development engineer with a complete microcontroller design tool set for all microcontrollers in the PIC16C5X, PIC16CXX and PIC17CXX families. PICMASTER is supplied with the MPLAB™ Integrated Development Environment (IDE), which allows editing, "make" and download, and source debugging from a single environment. A PICMASTER System configuration is shown in Figure 11-1.

Interchangeable target probes allow the system to be easily reconfigured for emulation of different processors. The universal architecture of the PICMASTER allows expansion to support all new PIC16C5X, PIC16CXX and PIC17CXX microcontrollers.

The PICMASTER Emulator System has been designed as a real-time emulation system with advanced features that are generally found on more expensive development tools. The PC compatible 386 (and better) machine platform and Microsoft® Windows® 3.x environment was chosen to best make these features available to you, the end user.

The PICMASTER Universal Emulator System consists primarily of four major components:

- Host-Interface Card
- Emulator Control Pod
- Target-Specific Emulator Probe
- PC-Host Emulation Control Software

The Windows operating system allows the developer to take full advantage of the many powerful features and functions of the PICMASTER system.

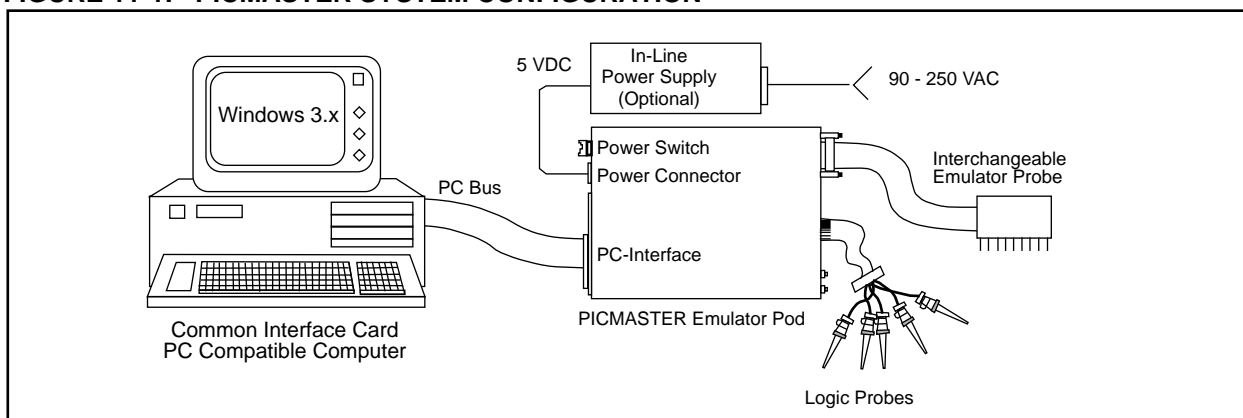
PICMASTER emulation can operate in one window, while a text editor is running in a second window.

PC-Host Emulation Control software takes full advantage of Dynamic Data Exchange (DDE), a feature of Windows. DDE allows data to be dynamically transferred between two or more Windows programs. With this feature, data collected with PICMASTER can be automatically transferred to a spreadsheet or database program for further analysis.

Under Windows, as many as four PICMASTER emulators can be run simultaneously from the same PC making development of multi-microcontroller systems possible (e.g., a system containing a PIC16CXX processor and a PIC17CXX processor).

The PICMASTER probes specifications are shown in Table 11-1.

**FIGURE 11-1: PICMASTER SYSTEM CONFIGURATION**



# PIC16C62X

**TABLE 11-1: PICMASTER PROBE SPECIFICATION**

Devices	PICMASTER PROBE	PROBE	
		Maximum Frequency	Operating Voltage
PIC16C54	PROBE-16D	20 MHz	4.5V - 5.5V
PIC16C54A	PROBE-16D	20 MHz	4.5V - 5.5V
PIC16CR54	PROBE-16D	20 MHz	4.5V - 5.5V
PIC16CR54A	PROBE-16D <sup>(1)</sup>	20 MHz	4.5V - 5.5V
PIC16C55	PROBE-16D	20 MHz	4.5V - 5.5V
PIC16C56	PROBE-16D	20 MHz	4.5V - 5.5V
PIC16C57	PROBE-16D	20 MHz	4.5V - 5.5V
PIC16CR57B	PROBE-16D <sup>(1)</sup>	20 MHz	4.5V - 5.5V
PIC16C58A	PROBE-16D	20 MHz	4.5V - 5.5V
PIC16CR58A	PROBE-16D	20 MHz	4.5V - 5.5V
PIC16C61	PROBE-16G	10 MHz	4.5V - 5.5V
PIC16C62	PROBE-16E	10 MHz	4.5V - 5.5V
PIC16C62A	PROBE-16E <sup>(1)</sup>	10 MHz	4.5V - 5.5V
PIC16CR62	PROBE-16E <sup>(1)</sup>	10 MHz	4.5V - 5.5V
PIC16C63	PROBE-16F <sup>(1)</sup>	10 MHz	4.5V - 5.5V
PIC16C64	PROBE-16E	10 MHz	4.5V - 5.5V
PIC16C64A	PROBE-16E <sup>(1)</sup>	10 MHz	4.5V - 5.5V
PIC16CR64	PROBE-16E <sup>(1)</sup>	10 MHz	4.5V - 5.5V
PIC16C65	PROBE-16F	10 MHz	4.5V - 5.5V
PIC16C65A	PROBE-16F <sup>(1)</sup>	10 MHz	4.5V - 5.5V
PIC16C620	PROBE-16H	10 MHz	4.5V - 5.5V
PIC16C621	PROBE-16H	10 MHz	4.5V - 5.5V

**TABLE 11-1: PICMASTER PROBE SPECIFICATION**

Devices	PICMASTER PROBE	PROBE	
		Maximum Frequency	Operating Voltage
PIC16C622	PROBE-16H	10 MHz	4.5V - 5.5V
PIC16C710	PROBE-16B <sup>(1)</sup>	10 MHz	4.5V - 5.5V
PIC16C71	PROBE-16B	10 MHz	4.5V - 5.5V
PIC16C711	PROBE-16B <sup>(1)</sup>	10 MHz	4.5V - 5.5V
PIC16C72	PROBE-16F <sup>(1)</sup>	10 MHz	4.5V - 5.5V
PIC16C73	PROBE-16F	10 MHz	4.5V - 5.5V
PIC16C73A	PROBE-16F <sup>(1)</sup>	10 MHz	4.5V - 5.5V
PIC16C74	PROBE-16F	10 MHz	4.5V - 5.5V
PIC16C74A	PROBE-16F <sup>(1)</sup>	10 MHz	4.5V - 5.5V
PIC16C83	PROBE-16C	10 MHz	4.5V - 5.5V
PIC16C84	PROBE-16C	10 MHz	4.5V - 5.5V
PIC17C42	PROBE-17B	20 MHz	4.5V - 5.5V
PIC17C43	PROBE-17B	20 MHz	4.5V - 5.5V
PIC17C44	PROBE-17B	20 MHz	4.5V - 5.5V

Note 1: This PICMASTER probe can be used to functionally emulate the device listed in the previous column. Contact your Microchip sales office for details.



## 11.3 PRO MATE: Universal Programmer

The PRO MATE Universal Programmer is a full-featured programmer capable of operating in stand-alone mode as well as PC-hosted mode.

The PRO MATE has programmable VDD and VPP supplies which allows it to verify programmed memory at VDD min and VDD max for maximum reliability. It has an LCD display for displaying error messages, keys to enter commands and a modular detachable socket assembly to support various package types. In stand-alone mode the PRO MATE can read, verify or program PIC16C5X, PIC16CXX and PIC17CXX devices. It can also set configuration and code-protect bits in this mode.

In PC-hosted mode, the PRO MATE connects to the PC via one of the COM (RS-232) ports. PC based user-interface software makes using the programmer simple and efficient. The user interface is full-screen and menu-based. Full screen display and editing of data, easy selection of bit configuration and part type, easy selection of VDD min, VDD max and VPP levels, load and store to and from disk files (Intel<sup>®</sup> hex format) are some of the features of the software. Essential commands such as read, verify, program and blank check can be issued from the screen. Additionally, serial programming support is possible where each part is programmed with a different serial number, sequential or random.

The PRO MATE has a modular "programming socket module". Different socket modules are required for different processor types and/or package types.

PRO MATE supports all PIC16C5X, PIC16CXX and PIC17CXX processors.

## 11.4 PICSTART Low-Cost Development System

The PICSTART programmer is an easy to use, very low-cost prototype programmer. It connects to the PC via one of the COM (RS-232) ports. A PC-based user interface software makes using the programmer simple and efficient. The user interface is full-screen and menu-based. PICSTART is not recommended for production programming.

## 11.5 PICDEM-1 Low-Cost PIC16/17 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 or PICSTART-16B programmer, and easily test firmware. The user can also connect the PICDEM-1 board to the PICMASTER 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.

## 11.6 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 programmer or PICSTART-16C, and easily test firmware. The PICMASTER 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.

## 11.7 MPLAB™ Integrated Development Environment Software.

The MPLAB Software brings an ease of software development previously unseen in the 8-bit microcontroller market. MPLAB is a windows based application which contains:

- A full featured editor
- Three operating modes
  - editor
  - emulator
  - simulator (available soon)
- A project manager
- Customizable tool bar and key mapping
- A status bar with project information
- Extensive on-line help

MPLAB allows you to:

- Edit your source files (either assembly or "C")
- One touch assemble (or compile) and download to PIC16/17 tools (automatically updates all project information)
- Debug using:
  - source files
  - absolute listing file
- Transfer data dynamically via DDE (soon to be replaced by OLE)
- Run up to four emulators on the same PC

The ability to use MPLAB with Microchip's simulator (available soon) allows a consistent platform and the ability to easily switch from the low cost simulator to the full featured emulator with minimal retraining due to development tools.

## 11.8 Assembler (MPASM)

The MPASM Cross Assembler is a PC-hosted symbolic assembler. It supports all microcontroller series including the PIC16C5X, PIC16CXX and PIC17CXX families.

MPASM offers full featured Macro capabilities, conditional assembly, and several source and listing formats. It generates various object code formats to support Microchip's development tools as well as third party programmers.

MPASM allows full symbolic debugging from the Microchip Universal Emulator System (PICMASTER).

MPASM has the following features to assist in developing software for specific use applications.

- Provides translation of Assembler source code to object code for all Microchip microcontrollers.
- Macro assembly capability.
- Produces all the files (Object, Listing, Symbol, and special) required for symbolic debug with Microchip's emulator systems.
- Supports Hex (default), Decimal and Octal source and listing formats.

MPASM provides a rich directive language to support programming of the PIC16/17. Directives are helpful in making the development of your assemble source code shorter and more maintainable.

- **Data Directives** are those that control the allocation of memory and provide a way to refer to data items symbolically (i.e., by meaningful names).
- **Control Directives** control the MPASM listing display. They allow the specification of titles and sub-titles, page ejects and other listing control. This eases the readability of the printed output file.
- **Conditional Directives** permit sections of conditionally assembled code. This is most useful where additional functionality may wished to be added depending on the product (less functionality for the low end product, then for the high end product). Also this is very helpful in the debugging of a program.
- **Macro Directives** control the execution and data allocation within macro body definitions. This makes very simple the re-use of functions in a program as well as between programs.

## 11.9 Software Simulator (MPSIM)

The MPSIM Software Simulator allows code development in a PC host environment. It allows the user to simulate the PIC16/17 series microcontrollers on an instruction level. On any given instruction, the user may examine or modify any of the data areas or provide external stimulus to any of the pins. The input/output radix can be set by the user and the execution can be performed in; single step, execute until break, or in a trace mode. MPSIM fully supports symbolic debugging using MP-C and MPASM. The Software Simulator offers the low cost flexibility to develop and debug code outside of the laboratory environment making it an excellent multi-project software development tool.

## 11.10 C Compiler (MP-C)

The MP-C Code Development System is a complete 'C' compiler and integrated development environment for Microchip's PIC16/17 family of microcontrollers. The compiler provides powerful integration capabilities and ease of use not found with other compilers.

For easier source level debugging, the compiler provides symbol information that is compatible with the PICMASTER Universal Emulator memory display (PICMASTER emulator software versions 1.13 and later).

The MP-C Code Development System is supplied directly by Byte Craft Limited of Waterloo, Ontario, Canada. If you have any questions, please contact your regional Microchip FAE or Microchip technical support personnel at (602) 786-7627.

## 11.11 Fuzzy Logic Development System (fuzzyTECH-MP)

fuzzyTECH-MP fuzzy logic development tool is available in two versions - a low cost introductory version, MP Explorer, for designers to gain a comprehensive working knowledge of fuzzy logic system design; and a full-featured version, fuzzyTECH-MP, edition for implementing more complex systems.

Both versions include Microchip's fuzzyLAB™ demonstration board for hands-on experience with fuzzy logic systems implementation.

## 11.12 Development Systems

For convenience, the development tools are packaged into comprehensive systems as listed in Table 11-2.

**TABLE 11-2: DEVELOPMENT SYSTEM PACKAGES**

Item	Name	System Description
1.	PICMASTER System	PICMASTER In-Circuit Emulator, PRO MATE Programmer, Assembler, Software Simulator, Samples and your choice of Target Probe.
2.	PICSTART System	PICSTART Low-Cost Prototype Programmer, Assembler, Software Simulator and Samples.
3.	PRO MATE System	PRO MATE Universal Programmer, full featured stand-alone or PC-hosted programmer, Assembler, Simulator

# PIC16C62X

---

NOTES:

## 12.0 ELECTRICAL SPECIFICATIONS

### Absolute Maximum Ratings †

Ambient Temperature under bias .....	-40° to +125°C
Storage Temperature .....	-65° to +150°C
Voltage on any pin with respect to VSS (except VDD and $\overline{\text{MCLR}}$ ) .....	-0.6V to VDD +0.6V
Voltage on VDD with respect to VSS .....	0 to +7.5V
Voltage on $\overline{\text{MCLR}}$ with respect to VSS (Note 2) .....	0 to +14V
Total power Dissipation (Note 1) .....	1.0W
Maximum Current out of VSS pin .....	300 mA
Maximum Current into VDD pin .....	250 mA
Input Clamp Current, I <sub>IK</sub> (V <sub>I</sub> < 0 or V <sub>I</sub> > VDD) .....	±20 mA
Output Clamp Current, I <sub>OK</sub> (V <sub>O</sub> < 0 or V <sub>O</sub> > VDD) .....	±20 mA
Maximum Output Current sunk by any I/O pin .....	25 mA
Maximum Output Current sourced by any I/O pin .....	25 mA
Maximum Current sunk by PORTA and PORTB .....	200 mA
Maximum Current sourced by PORTA and PORTB .....	200 mA

**Note 1:** Power dissipation is calculated as follows:  $P_{DIS} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OI} \times I_{OL})$

† **NOTICE:** Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

**TABLE 12-1: CROSS REFERENCE OF DEVICE SPECS FOR OSCILLATOR CONFIGURATIONS AND FREQUENCIES OF OPERATION (COMMERCIAL DEVICES)**

OSC	PIC16C62X-04	PIC16C62X-20	PIC16LC62X-04	JW Devices
RC	VDD: 4.0V to 6.0V IDD: 3.3 mA max. @5.5V IPD: 20 µA max. @4.0V Freq: 4.0 MHz max.	VDD: 4.5V to 5.5V IDD: 1.8 mA typ. @5.5V IPD: 1.0 µA typ. @4.5V Freq: 4.0 MHz max.	VDD: 3.0V to 6.0V IDD: 1.4 mA typ. @3.0V IPD: 0.7 µA typ. @3.0V Freq: 4.0 MHz max.	VDD: 3.0V to 6.0V IDD: 3.3 mA max. @5.5V IPD: 20 µA max. @4.0V Freq: 4.0 MHz Max.
XT	VDD: 4.0V to 6.0V IDD: 3.3 mA max. @5.5V IPD: 20 µA max. @4.0V Freq: 4.0 MHz max.	VDD: 4.5V to 5.5V IDD: 1.8 mA typ. @5.5V IPD: 1.0 µA typ. @4.5V Freq: 4.0 MHz max.	VDD: 3.0V to 6.0V IDD: 1.4 mA typ. @3.0V IPD: 0.7 µA typ. @3.0V Freq: 4.0 MHz max.	VDD: 3.0V to 6.0V IDD: 3.3 mA max. @5.5V IPD: 20 µA max. @4.0V Freq: 4.0 MHz max.
HS	VDD: 4.5V to 5.5V IDD: 9.0 mA typ. @5.5V IPD: 1.0 µA typ. @4.0V Freq: 4.0 MHz max.	VDD: 4.5V to 5.5V IDD: 20 mA max. @5.5V IPD: 1.0 µA typ. @4.5V Freq: 20 MHz max.	Do not use in HS mode	VDD: 4.5V to 5.5V IDD: 20 mA max. @5.5V IPD: 1.0 µA typ. @4.5V Freq: 20 MHz max.
LP	VDD: 4.0V to 6.0V IDD: 35 µA typ. @32 kHz, 3.0V IPD: 1.0 µA typ. @4.0 V Freq: 200 kHz maximum	Do not use in LP mode	VDD: 2.5V to 6.0V IDD: 32 µA max. @32 kHz, 3.0V IPD: 9.0 µA max. @3.0V Freq: 200 kHz max.	VDD: 2.5V to 6.0V IDD: 32 µA max. @32 kHz, 3.0V IPD: 9.0 µA Max. @3.0V Freq: 200 kHz max.

The shaded sections indicate oscillator selections which are tested for functionality, but not for MIN/MAX specifications. It is recommended that the user select the device type that guarantees the specifications required.

# PIC16C62X

## 12.1 DC CHARACTERISTICS: PIC16C62X-04 (Commercial, Industrial, Automotive) PIC16C62X-20 (Commercial, Industrial, Automotive)

Standard Operating Conditions (unless otherwise stated)							
Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial and $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ for commercial and $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for automotive							
Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
D001 D001A	VDD	Supply Voltage	4.0 4.5	- -	6.0 5.5	V V	XT, RC and LP osc configuration HS osc configuration
D002	VDR	RAM Data Retention Voltage (Note 1)	-	1.5*	-	V	Device in SLEEP mode
D003	VPOR	VDD start voltage to ensure Power-on Reset	-	VSS	-	V	See section on power-on reset for details
D004	SVDD	VDD rise rate to ensure Power-on Reset	0.05*	-	-	V/ms	See section on power-on reset for details
D005	VBOR	Brown-out Detect Voltage	3.7 3.7	4.0 4.0	4.3 4.4	V	BODEN configuration bit is cleared (Automotive)
D010 D010A D013	IDD	Supply Current (Note 2)	- - -	1.8 35 9.0	3.3 70 20	mA $\mu\text{A}$ mA	XT and RC osc configuration FOSC = 4 MHz, VDD = 5.5V, WDT disabled (Note 4) LP osc configuration, PIC16C62X-04 only FOSC = 32 kHz, VDD = 4.0V, WDT disabled HS osc configuration FOSC = 20 MHz, VDD = 5.5V, WDT disabled
	$\Delta\text{IWDT}$	WDT Current (Note 5)	-	6.0	20 25	$\mu\text{A}$ $\mu\text{A}$	VDD = 4.0V (+85°C to +125°C)
D015	$\Delta\text{IBOR}$	Brown-out Reset Current (Note 5)	-	350	425	$\mu\text{A}$	$\text{BOR}$ enabled, VDD = 5.0V
	$\Delta\text{ICOMP}$	Comparator Current for each Comparator (Note 5)	-	-	100	$\mu\text{A}$	VDD = 4.0V
	$\Delta\text{IVREF}$	VREF Current (Note 5)	-	-	300	$\mu\text{A}$	VDD = 4.0V

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 5.0V, 25°C, unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: This is the limit to which VDD can be lowered in SLEEP mode without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern, and temperature also have an impact on the current consumption.

The test conditions for all IDD measurements in active operation mode are:

OSC1 = external square wave, from rail to rail; all I/O pins tri-stated™, pulled to VDD,

$\overline{\text{MCLR}} = \text{VDD}$ ; WDT enabled/disabled as specified.

3: The power down current in SLEEP mode does not depend on the oscillator type. Power down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to VDD or VSS.

4: For RC osc configuration, current through Rext is not included. The current through the resistor can be estimated by the formula  $I_r = \text{VDD}/2\text{Rext}$  (mA) with Rext in kΩ.

5: The  $\Delta$  current is the additional current consumed when this peripheral is enabled. This current should be added to the base IDD or IPD measurement.

## 12.1 DC CHARACTERISTICS: PIC16C62X-04 (Commercial, Industrial, Automotive) PIC16C62X-20 (Commercial, Industrial, Automotive) (Cont.)

Standard Operating Conditions (unless otherwise stated)							
Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial and $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ for commercial and $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for automotive							
Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
D020	IPD	Power Down Current (Note 3)	–	1.0	2.5 15	$\mu\text{A}$ $\mu\text{A}$	$V_{DD}=4.0\text{V}$ , WDT disabled ( $+85^{\circ}\text{C}$ to $+125^{\circ}\text{C}$ )
D023	$\Delta I_{WDT}$	WDT Current (Note 5)	–	6.0	20 25	$\mu\text{A}$ $\mu\text{A}$	$V_{DD}=4.0\text{V}$ ( $+85^{\circ}\text{C}$ to $+125^{\circ}\text{C}$ )
	$\Delta I_{BOR}$	Brown-out Reset Current (Note 5)	–	350	425	$\mu\text{A}$	$\overline{BOR}$ enabled, $V_{DD} = 5.0\text{V}$
	$\Delta I_{COMP}$	Comparator Current for each Comparator (Note 5)	–		100	$\mu\text{A}$	$V_{DD} = 4.0\text{V}$
	$\Delta I_{VREF}$	VREF Current (Note 5)	–		300	$\mu\text{A}$	$V_{DD} = 4.0\text{V}$

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 5.0V, 25°C, unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: This is the limit to which VDD can be lowered in SLEEP mode without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern, and temperature also have an impact on the current consumption.

The test conditions for all IDD measurements in active operation mode are:

OSC1 = external square wave, from rail to rail; all I/O pins tri-stated™, pulled to VDD,

$\overline{MCLR} = V_{DD}$ ; WDT enabled/disabled as specified.

3: The power down current in SLEEP mode does not depend on the oscillator type. Power down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to VDD or VSS.

4: For RC osc configuration, current through Rext is not included. The current through the resistor can be estimated by the formula  $I_r = V_{DD}/2R_{ext}$  (mA) with Rext in kΩ.

5: The  $\Delta$  current is the additional current consumed when this peripheral is enabled. This current should be added to the base IDD or IPD measurement.

# PIC16C62X

## 12.2 DC CHARACTERISTICS: PIC16LC62X-04 (Commercial, Industrial)

Standard Operating Conditions (unless otherwise stated)							
Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial and $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ for commercial							
Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
D001	VDD	Supply Voltage	3.0 2.5	-	6.0 6.0	V	XT and RC osc configuration LP osc configuration
D002	VDR	RAM Data Retention Voltage (Note 1)	-	1.5*	-	V	Device in SLEEP mode
D003	VPOR	VDD start voltage to ensure Power-on Reset	-	VSS	-	V	See section on Power-on Reset for details
D004	SVDD	VDD rise rate to ensure Power-on Reset	0.05*	-	-	V/ms	See section on Power-on Reset for details
D005	VBOR	Brown-out Detect Voltage	3.7	4.0	4.3	V	BODEN configuration bit is cleared
D010	IDD	Supply Current (Note 2)	-	1.4	2.5	mA	XT and RC osc configuration FOSC = 2.0 MHz, VDD = 3.0V, WDT disabled (Note 4)
D010A			-	26	53	$\mu\text{A}$	LP osc configuration FOSC = 32 kHz, VDD = 3.0V, WDT disabled
D015	$\Delta\text{IWDT}$	WDT Current (Note 5)	-	6.0	15	$\mu\text{A}$	VDD = 3.0V
	$\Delta\text{IBOR}$	Brown-out Reset Current (Note 5)	-	350	425	$\mu\text{A}$	$\overline{\text{BOR}}$ enabled, VDD = 5.0V
	$\Delta\text{ICOMP}$	Comparator Current for each Comparator (Note 5)	-	-	100	$\mu\text{A}$	VDD = 3.0V
	$\Delta\text{IVREF}$	VREF Current (Note 5)	-	-	300	$\mu\text{A}$	VDD = 3.0V
D020	IPD	Power Down Current (Note 3)	-	0.7	2	$\mu\text{A}$	VDD=3.0V, WDT disabled
D023	$\Delta\text{IWDT}$	WDT Current (Note 5)	-	6.0	15	$\mu\text{A}$	VDD=3.0V
	$\Delta\text{IBOR}$	Brown-out Reset Current (Note 5)	-	350	425	$\mu\text{A}$	$\overline{\text{BOR}}$ enabled, VDD = 5.0V
	$\Delta\text{ICOMP}$	Comparator Current for each Comparator (Note 5)	-	-	100	$\mu\text{A}$	VDD = 3.0V
	$\Delta\text{IVREF}$	VREF Current (Note 5)	-	-	300	$\mu\text{A}$	VDD = 3.0V

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 5.0V, 25°C, unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: This is the limit to which VDD can be lowered in SLEEP mode without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern, and temperature also have an impact on the current consumption.

The test conditions for all IDD measurements in active operation mode are:

OSC1=external square wave, from rail to rail; all I/O pins tristated, pulled to VDD,

$\overline{\text{MCLR}} = \text{VDD}$ ; WDT enabled/disabled as specified.

3: The power down current in SLEEP mode does not depend on the oscillator type. Power down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to VDD to VSS.

4: For RC osc configuration, current through Rext is not included. The current through the resistor can be estimated by the formula  $I_r = \text{VDD}/2\text{Rext}$  (mA) with Rext in k $\Omega$ .

5: The  $\Delta$  current is the additional current consumed when this peripheral is enabled. This current should be added to the base IDD or IPD measurement.



## 12.3 DC CHARACTERISTICS: PIC16C62X (Commercial, Industrial, Automotive) PIC16LC62X (Commercial, Industrial, Automotive)

Standard Operating Conditions (unless otherwise stated)							
Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial and $0^{\circ}\text{C} \leq \text{TA} \leq +70^{\circ}\text{C}$ for commercial and $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for automotive							
Operating voltage $V_{DD}$ range as described in DC spec Table 12-1 and Table 12-2							
Param. No.	Sym	Characteristic	Min	Typ†	Max	Unit	Conditions
D030	$V_{IL}$	<b>Input Low Voltage</b> I/O ports with TTL buffer	$V_{SS}$	-	0.8V 0.15 $V_{DD}$	V	$V_{DD} = 4.5\text{V to } 5.5\text{V}$ otherwise
D031		with Schmitt Trigger input	$V_{SS}$	-	0.2 $V_{DD}$	V	Note1
D032		$\overline{\text{MCLR}}$ , RA4/T0CKI, OSC1 (in RC mode)	$V_{SS}$	-	0.2 $V_{DD}$	V	
D033		OSC1 (in XT and HS) OSC1 (in LP)	$V_{SS}$ $V_{SS}$	- -	0.3 $V_{DD}$ 0.6 $V_{DD}$ -1.0	V V	
D040	$V_{IH}$	<b>Input High Voltage</b> I/O ports with TTL buffer	2.0V	-	$V_{DD}$	V	Note1
D041		with Schmitt Trigger input	0.8 $V_{DD}$	-	$V_{DD}$	V	
D042		$\overline{\text{MCLR}}$ RA4/T0CKI	0.8 $V_{DD}$	-	$V_{DD}$	V	
D043		OSC1 (XT, HS and LP)	0.7 $V_{DD}$	-	$V_{DD}$	V	
D043A		OSC1 (in RC mode)	0.9 $V_{DD}$	-	$V_{DD}$	V	
D070	IPURB	PORTB weak pull-up current	50	200	400	$\mu\text{A}$	$V_{DD} = 5.0\text{V}$ , $V_{PIN} = V_{SS}$
D060	$I_{IL}$	<b>Input Leakage Current</b> (Notes 2, 3) I/O ports (Except PORTA)	-	-	$\pm 1.0$	$\mu\text{A}$	$V_{SS} \leq V_{PIN} \leq V_{DD}$ , pin at hi-impedance
D061		PORTA	-	-	$\pm 0.5$	$\mu\text{A}$	$V_{SS} \leq V_{PIN} \leq V_{DD}$ , pin at hi-impedance
D063		RA4/T0CKI	-	-	$\pm 1.0$	$\mu\text{A}$	$V_{SS} \leq V_{PIN} \leq V_{DD}$
		OSC1, $\overline{\text{MCLR}}$	-	-	$\pm 5.0$	$\mu\text{A}$	$V_{SS} \leq V_{PIN} \leq V_{DD}$ , XT, HS and LP osc configuration
D080	$V_{OL}$	<b>Output Low Voltage</b> I/O ports	-	-	0.6	V	$I_{OL}=8.5\text{ mA}$ , $V_{DD}=4.5\text{V}$ , $-40^{\circ}$ to $+85^{\circ}\text{C}$
D083		OSC2/CLKOUT (RC only)	-	-	0.6	V	$I_{OL}=7.0\text{ mA}$ , $V_{DD}=4.5\text{V}$ , $+125^{\circ}\text{C}$ $I_{OL}=1.6\text{ mA}$ , $V_{DD}=4.5\text{V}$ , $-40^{\circ}$ to $+85^{\circ}\text{C}$ $I_{OL}=1.2\text{ mA}$ , $V_{DD}=4.5\text{V}$ , $+125^{\circ}\text{C}$
D090	$V_{OH}$	<b>Output High Voltage</b> (Note 3) I/O ports (Except RA4)	$V_{DD}-0.7$	-	-	V	$I_{OH}=-3.0\text{ mA}$ , $V_{DD}=4.5\text{V}$ , $-40^{\circ}$ to $+85^{\circ}\text{C}$
D092		OSC2/CLKOUT (RC only)	$V_{DD}-0.7$	-	-	V	$I_{OH}=-2.5\text{ mA}$ , $V_{DD}=4.5\text{V}$ , $+125^{\circ}\text{C}$ $I_{OH}=-1.3\text{ mA}$ , $V_{DD}=4.5\text{V}$ , $-40^{\circ}$ to $+85^{\circ}\text{C}$ $I_{OH}=-1.0\text{ mA}$ , $V_{DD}=4.5\text{V}$ , $+125^{\circ}\text{C}$
*	$V_{OD}$	<b>Open-Drain High Voltage</b>			14*	V	RA4 pin

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: In RC oscillator configuration, the OSC1 pin is a Schmitt Trigger input. It is not recommended that the PIC16C62X be driven with external clock in RC mode.

- The leakage current on the  $\overline{\text{MCLR}}$  pin is strongly dependent on applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.
- Negative current is defined as coming out of the pin.

# PIC16C62X

## 12.3 DC CHARACTERISTICS: PIC16C62X (Commercial, Industrial, Automotive) PIC16LC62X (Commercial, Industrial, Automotive) (Cont.)

Standard Operating Conditions (unless otherwise stated)							
Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial and $0^{\circ}\text{C} \leq \text{TA} \leq +70^{\circ}\text{C}$ for commercial and $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for automotive							
Operating voltage $V_{\text{DD}}$ range as described in DC spec Table 12-1 and Table 12-2							
Param. No.	Sym	Characteristic	Min	Typ†	Max	Unit	Conditions
D100	Cosc2	<b>Capacitive Loading Specs on Output Pins</b> OSC2 pin			15	pF	In XT, HS and LP modes when external clock used to drive OSC1.
D101	Cio	All I/O pins/OSC2 (in RC mode)			50	pF	

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

- Note 1: In RC oscillator configuration, the OSC1 pin is a Schmitt Trigger input. It is not recommended that the PIC16C62X be driven with external clock in RC mode.
- 2: The leakage current on the MCLR pin is strongly dependent on applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.
- 3: Negative current is defined as coming out of the pin.

### TABLE 12-2: COMPARATOR SPECIFICATIONS

Operating Conditions:  $2.5\text{V} < V_{\text{DD}} < 6.0\text{V}$ ,  $-40^{\circ}\text{C} < \text{TA} < +125^{\circ}\text{C}$ , unless otherwise stated. Current consumption is specified in Table 12-1.

Characteristics	Sym	Min	Typ	Max	Units	Comments
Input offset voltage			$\pm 5.0$	$\pm 10$	mV	
Input common mode voltage		0		$V_{\text{DD}} - 1.5$	V	
CMRR		$-35^*$			db	
Response Time <sup>(1)</sup>			150*	400* 600*	ns ns	PIC16C62X PIC16LC62X
Comparator Mode Change to Output Valid				10*	$\mu\text{s}$	

\* These parameters are characterized but not tested.  
Note 1: Response time measured with one comparator input at  $(V_{\text{DD}} - 1.5)/2$  while the other input transitions from  $V_{\text{SS}}$  to  $V_{\text{DD}}$ .

### TABLE 12-3: VOLTAGE REFERENCE SPECIFICATIONS

Operating Conditions:  $2.5\text{V} < V_{\text{DD}} < 6.0\text{V}$ ,  $-40^{\circ}\text{C} < \text{TA} < +125^{\circ}\text{C}$ , unless otherwise stated. Current consumption is specified in Table 12-1.

Characteristics	Sym	Min	Typ	Max	Units	Comments
Resolution		$V_{\text{DD}}/24$		$V_{\text{DD}}/32$	LSB	
Absolute Accuracy				1/4 1/2	LSB LSB	Low Range ( $V_{\text{RR}}=1$ ) High Range ( $V_{\text{RR}}=0$ )
Unit Resistor Value (R)			2K*		$\Omega$	Figure 8-2
Settling Time <sup>(1)</sup>				10*	$\mu\text{s}$	

\* These parameters are characterized but not tested.  
Note 1: Settling time measured while  $V_{\text{RR}} = 1$  and  $V_{\text{R}} < 3:0 >$  transitions from 0000 to 1111.

## 12.4 Timing Parameter Symbology

The timing parameter symbols have been created with one of the following formats:

1. TppS2ppS
2. TppS

<b>T</b>			
F	Frequency	T	Time

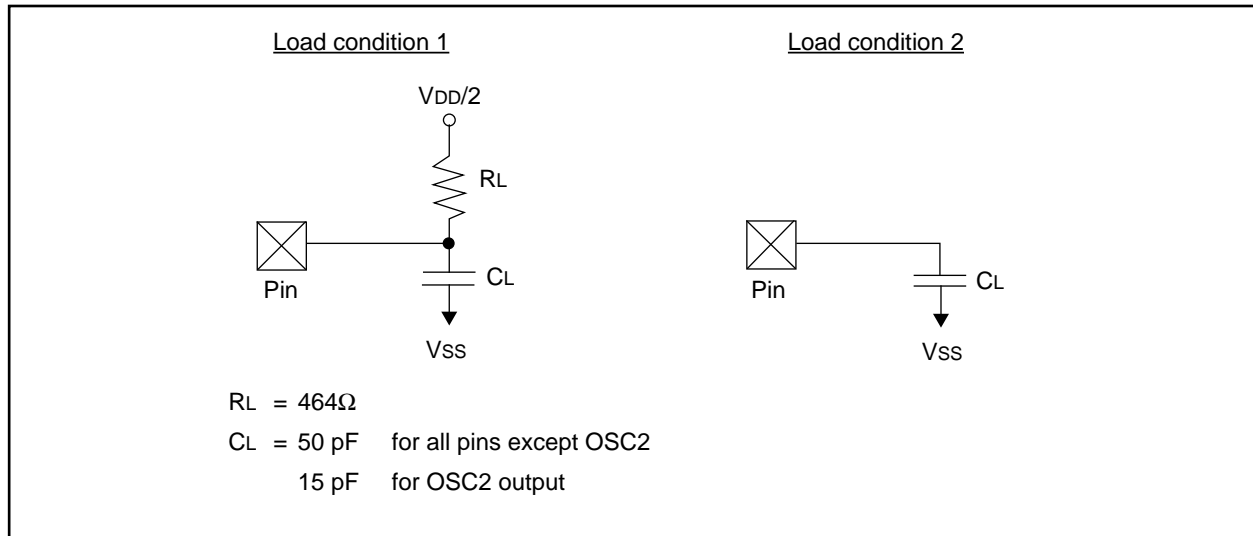
Lowercase subscripts (pp) and their meanings:

<b>pp</b>			
ck	CLKOUT	osc	OSC1
io	I/O port	t0	T0CKI
mc	MCLR		

Uppercase letters and their meanings:

<b>S</b>			
F	Fall	P	Period
H	High	R	Rise
I	Invalid (Hi-impedance)	V	Valid
L	Low	Z	Hi-Impedance

**FIGURE 12-1: LOAD CONDITIONS**



# PIC16C62X

## 12.5 Timing Diagrams and Specifications

FIGURE 12-2: EXTERNAL CLOCK TIMING

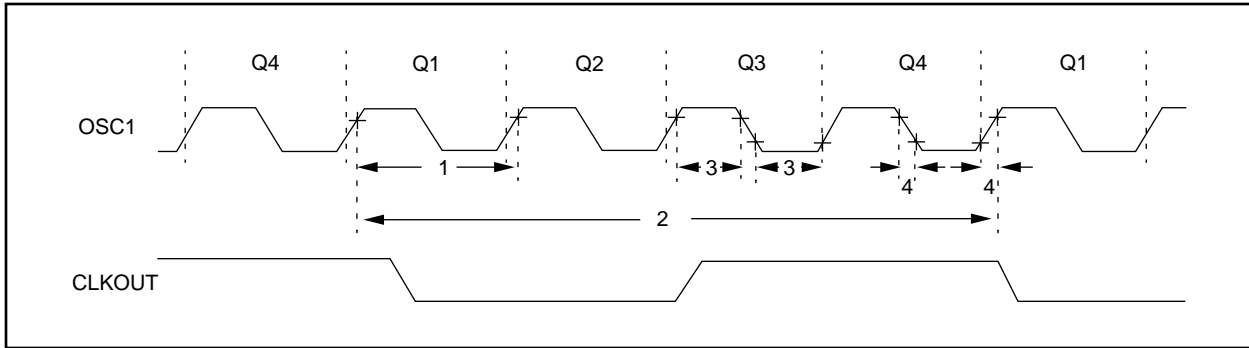


TABLE 12-4: EXTERNAL CLOCK TIMING REQUIREMENTS

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
	Fos	<b>External CLKIN Frequency (Note 1)</b>	DC	—	4	MHz	XT and RC osc mode, VDD=5.0V
			DC	—	20	MHz	HS osc mode
			DC	—	200	kHz	LP osc mode
		<b>Oscillator Frequency (Note 1)</b>	DC	—	4	MHz	RC osc mode, VDD=5.0V
			0.1	—	4	MHz	XT osc mode
			1	—	20	MHz	HS osc mode
1	Tosc	<b>External CLKIN Period (Note 1)</b>	250	—	—	ns	XT and RC osc mode
			50	—	—	ns	HS osc mode
			5	—	—	μs	LP osc mode
		<b>Oscillator Period (Note 1)</b>	250	—	—	ns	RC osc mode
			250	—	10,000	ns	XT osc mode
			50	—	1,000	ns	HS osc mode
2	Tcy	<b>Instruction Cycle Time (Note 1)</b>	1.0	Fosc/4	DC	μs	TCYS=FOSC/4
			5	—	—	μs	LP osc mode
3*	TosL, TosH	External Clock in (OSC1) High or Low Time	100*	—	—	ns	XT osc mode
			2*	—	—	μs	LP osc mode
			20*	—	—	ns	HS osc mode
4*	TosR, TosF	External Clock in (OSC1) Rise or Fall Time	25*	—	—	ns	XT osc mode
			50*	—	—	ns	LP osc mode
			15*	—	—	ns	HS osc mode

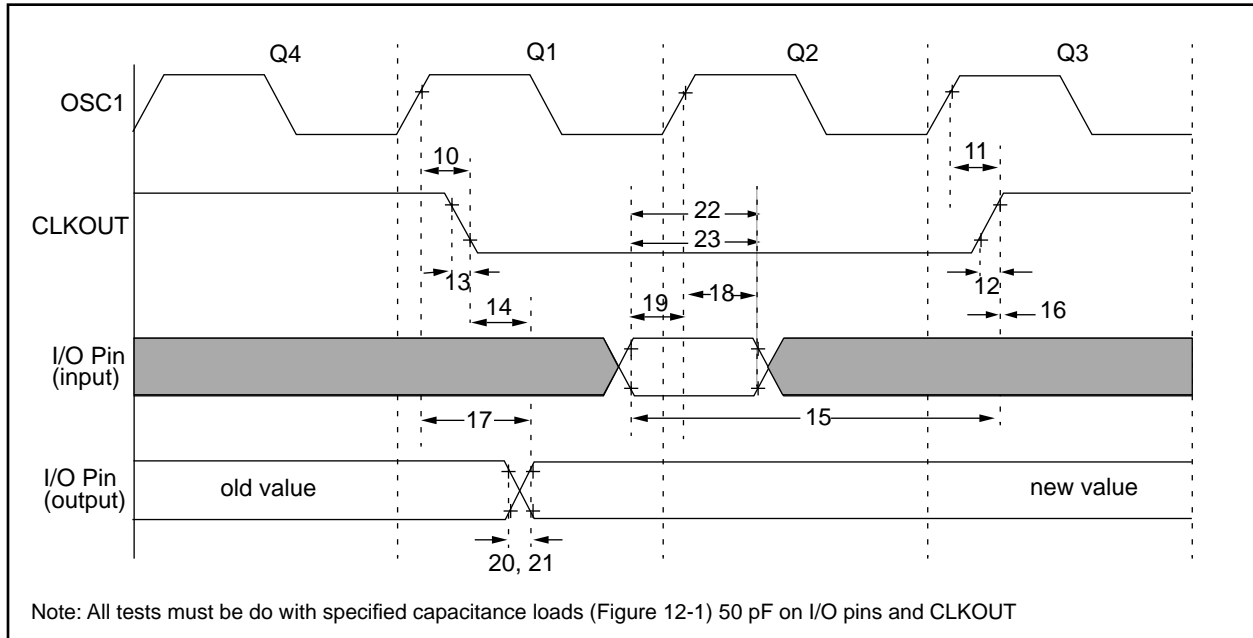
\* These parameters are characterized but not tested.

† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Instruction cycle period (Tcy) equals four times the input oscillator time-base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at "min." values with an external clock applied to the OSC1 pin.

When an external clock input is used, the "Max." cycle time limit is "DC" (no clock) for all devices.

**FIGURE 12-3: CLKOUT AND I/O TIMING**



**TABLE 12-5: CLKOUT AND I/O TIMING REQUIREMENTS**

Parameter #	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
10*	TosH2ckL	OSC1↑ to CLKOUT↓ <sup>(1)</sup>	—	75	200 400	ns ns	PIC16C62X PIC16LC62X
11*	TosH2ckH	OSC1↑ to CLKOUT↑ <sup>(1)</sup>	—	75	200 400	ns ns	PIC16C62X PIC16LC62X
12*	TckR	CLKOUT rise time <sup>(1)</sup>	—	35	100 200	ns ns	PIC16C62X PIC16LC62X
13*	TckF	CLKOUT fall time <sup>(1)</sup>	—	35	100 200	ns ns	PIC16C62X PIC16LC62X
14*	TckL2ioV	CLKOUT ↓ to Port out valid <sup>(1)</sup>	—	—	20	ns	
15*	TioV2ckH	Port in valid before CLKOUT ↑ <sup>(1)</sup>	Tosc +200 ns Tosc +400 ns	—	—	ns ns	PIC16C62X PIC16LC62X
16*	TckH2iol	Port in hold after CLKOUT ↑ <sup>(1)</sup>	0	—	—	ns	
17*	TosH2ioV	OSC1↑ (Q1 cycle) to Port out valid	—	50	150 300	ns ns	PIC16C62X PIC16LC62X
18*	TosH2iol	OSC1↑ (Q2 cycle) to Port input invalid (I/O in hold time)	100 200	—	—	ns ns	PIC16C62X PIC16LC62X
19*	TioV2osH	Port input valid to OSC1↑ (I/O in setup time)	0	—	—	ns	
20*	TioR	Port output rise time	—	10	40 80	ns ns	PIC16C62X PIC16LC62X
21*	TioF	Port output fall time	—	10	40 80	ns ns	PIC16C62X PIC16LC62X
22*	Tinp	RB0/INT pin high or low time	25 40	—	—	ns ns	PIC16C62X PIC16LC62X
23	Trbp	RB<7:4> change interrupt high or low time	Tcy	—	—	ns	

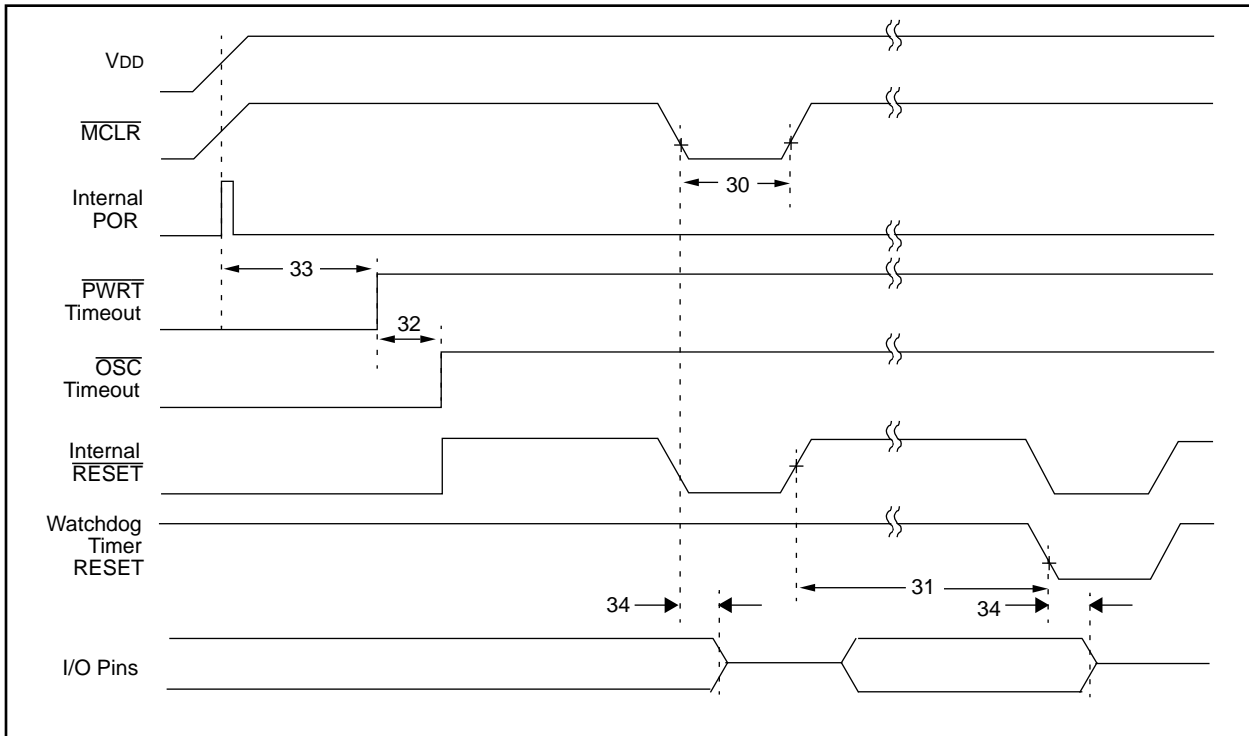
\* These parameters are characterized but not tested

† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

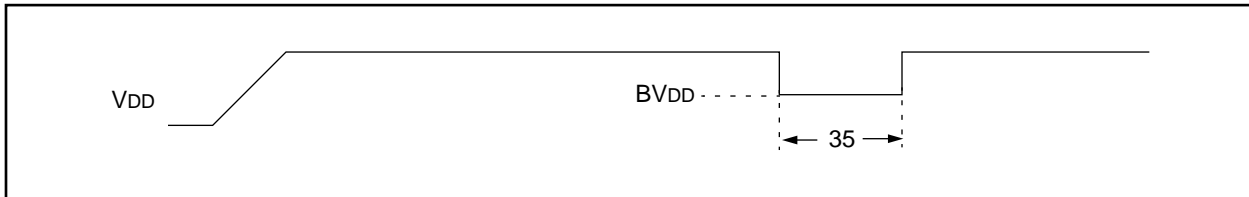
Note 1: Measurements are taken in RC Mode where CLKOUT output is 4 x TOSC

# PIC16C62X

**FIGURE 12-4: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING**



**FIGURE 12-5: BROWN-OUT RESET TIMING**



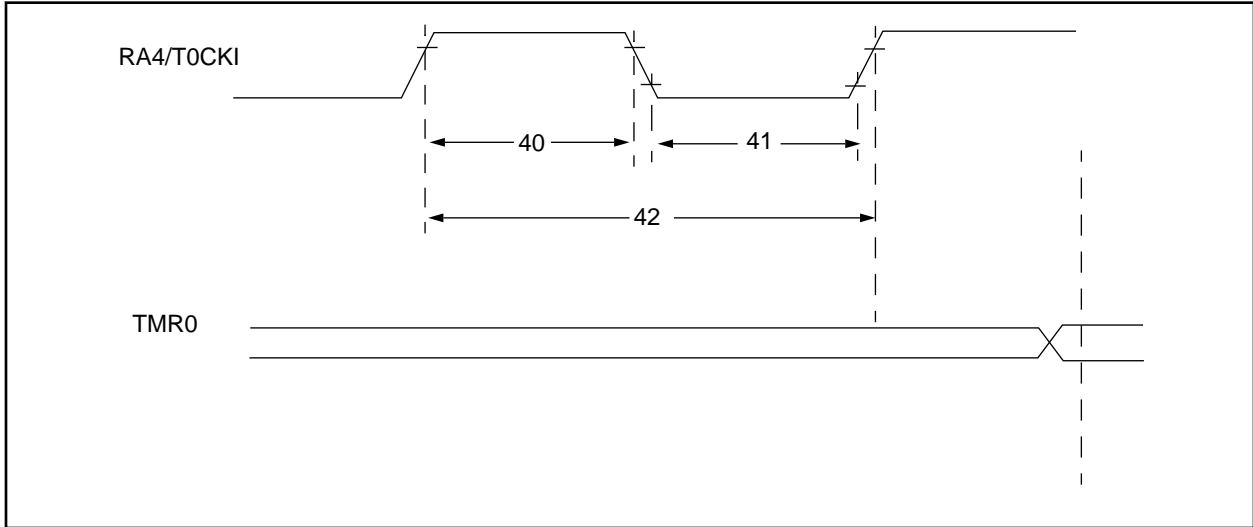
**TABLE 12-6: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER REQUIREMENTS**

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
30	Tmcl	MCLR Pulse Width (low)	2000	—	—	ns	-40° to +85°C
31	Twdt	Watchdog Timer Time-out Period (No Prescaler)	7*	18	33*	ms	VDD = 5.0V, -40° to +85°C
32	Tost	Oscillation Start-up Timer Period	—	1024 T <sub>osc</sub>	—	—	T <sub>osc</sub> = OSC1 period
33	Tpwrt	Power-up Timer Period	28*	72	132*	ms	VDD = 5.0V, -40° to +85°C
34	Tioz	I/O hi-impedance from MCLR low	—	—	2.0	µs	
35	TBOR	Brown-out Reset Pulse Width	100*	—	—	µs	3.8V ≤ VDD ≤ 4.2V

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**FIGURE 12-6: TIMER0 CLOCK TIMING**



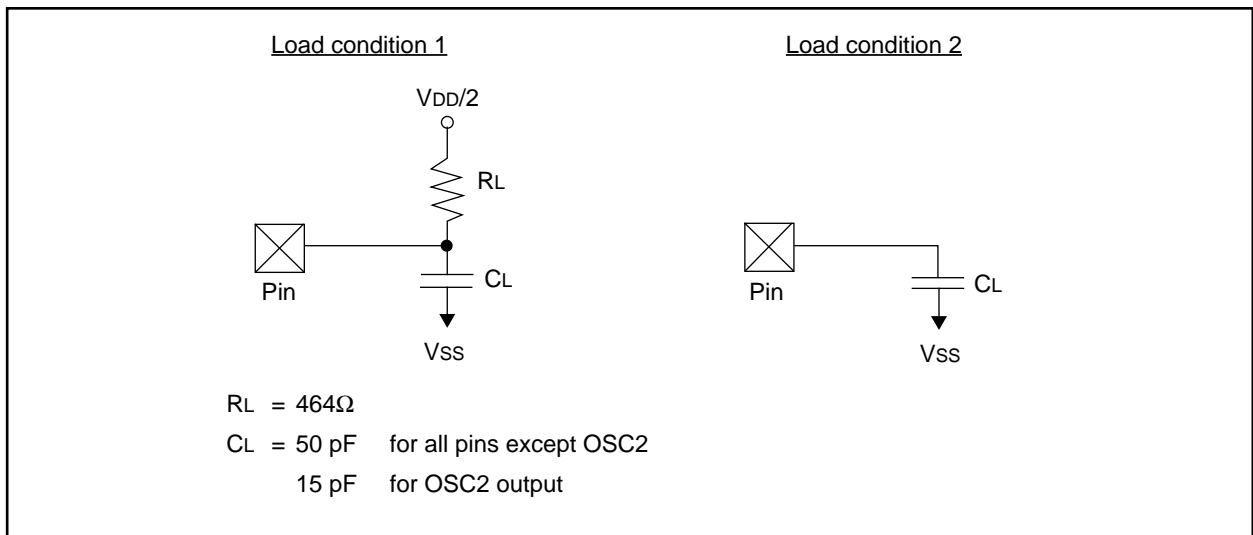
**TABLE 12-7: TIMER0 CLOCK REQUIREMENTS**

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
40	Tt0H	T0CKI High Pulse Width	No Prescaler	$0.5 T_{CY} + 20^*$	—	—	ns
			With Prescaler	10*	—	—	ns
41	Tt0L	T0CKI Low Pulse Width	No Prescaler	$0.5 T_{CY} + 20^*$	—	—	ns
			With Prescaler	10*	—	—	ns
42	Tt0P	T0CKI Period	$\frac{T_{CY} + 40^*}{N}$	—	—	ns	N = prescale value (1, 2, 4, ..., 256)

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**FIGURE 12-7: LOAD CONDITIONS**



# PIC16C62X

---

NOTES:



## 13.0 DEVICE CHARACTERIZATION INFORMATION

Not Available at this time.

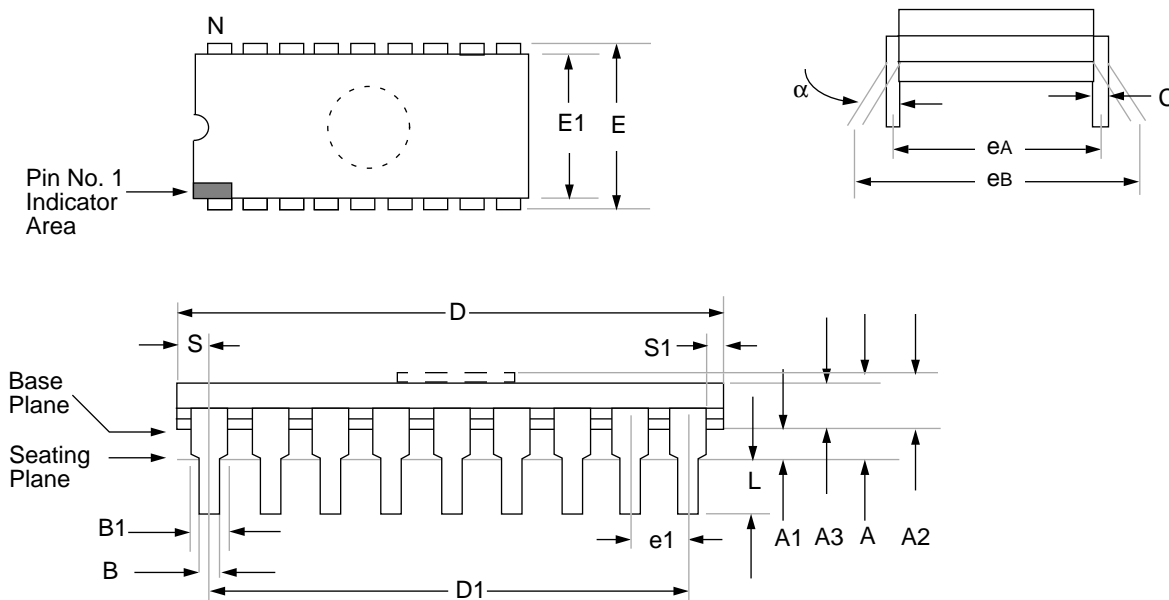
# PIC16C62X

---

NOTES:

## 14.0 PACKAGING INFORMATION

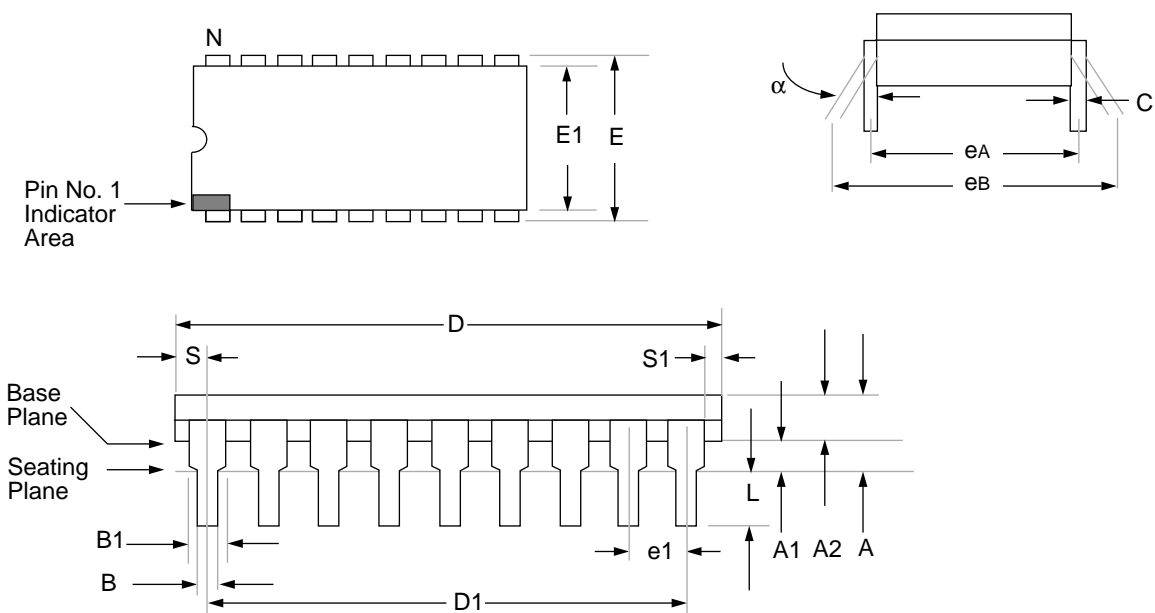
### 14.1 18-Lead Ceramic CERDIP Dual In-line with Window (300 mil)



Package Group: Ceramic CERDIP Dual In-Line (CDP)						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
$\alpha$	0°	10°		0°	10°	
A	—	5.080		—	0.200	
A1	0.381	1.7780		0.015	0.070	
A2	3.810	4.699		0.150	0.185	
A3	3.810	4.445		0.150	0.175	
B	0.355	0.585		0.014	0.023	
B1	1.270	1.651	Typical	0.050	0.065	Typical
C	0.203	0.381	Typical	0.008	0.015	Typical
D	22.352	23.622		0.880	0.930	
D1	20.320	20.320	Reference	0.800	0.800	Reference
E	7.620	8.382		0.300	0.330	
E1	5.588	7.874		0.220	0.310	
e1	2.540	2.540	Reference	0.100	0.100	Reference
eA	7.366	8.128	Typical	0.290	0.320	Typical
eB	7.620	10.160		0.300	0.400	
L	3.175	3.810		0.125	0.150	
N	18	18		18	18	
S	0.508	1.397		0.020	0.055	
S1	0.381	1.270		0.015	0.050	

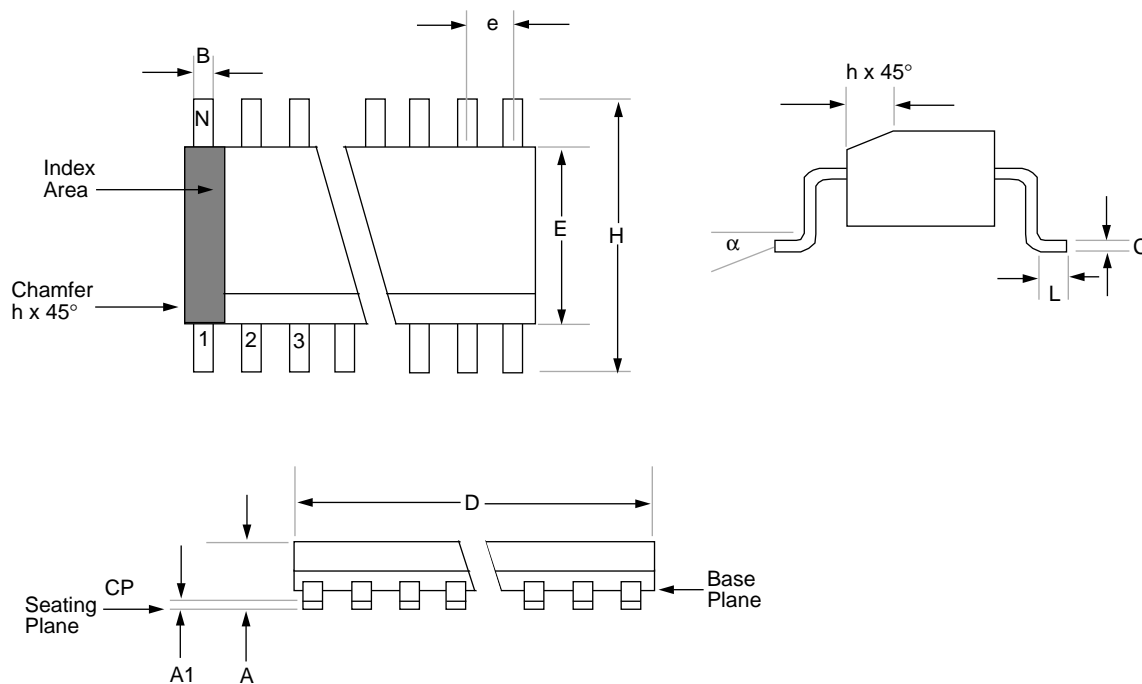
# PIC16C62X

## 14.2 18-Lead Plastic Dual In-line (300 mil)



Package Group: Plastic Dual In-Line (PLA)						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
$\alpha$	0°	10°		0°	10°	
A	–	4.064		–	0.160	
A1	0.381	–		0.015	–	
A2	3.048	3.810		0.120	0.150	
B	0.355	0.559		0.014	0.022	
B1	1.524	1.524	Reference	0.060	0.060	Reference
C	0.203	0.381	Typical	0.008	0.015	Typical
D	22.479	23.495		0.885	0.925	
D1	20.320	20.320	Reference	0.800	0.800	Reference
E	7.620	8.255		0.300	0.325	
E1	6.096	7.112		0.240	0.280	
e1	2.489	2.591	Typical	0.098	0.102	Typical
eA	7.620	7.620	Reference	0.300	0.300	Reference
eB	7.874	9.906		0.310	0.390	
L	3.048	3.556		0.120	0.140	
N	18	18		18	18	
S	0.889	–		0.035	–	
S1	0.127	–		0.005	–	

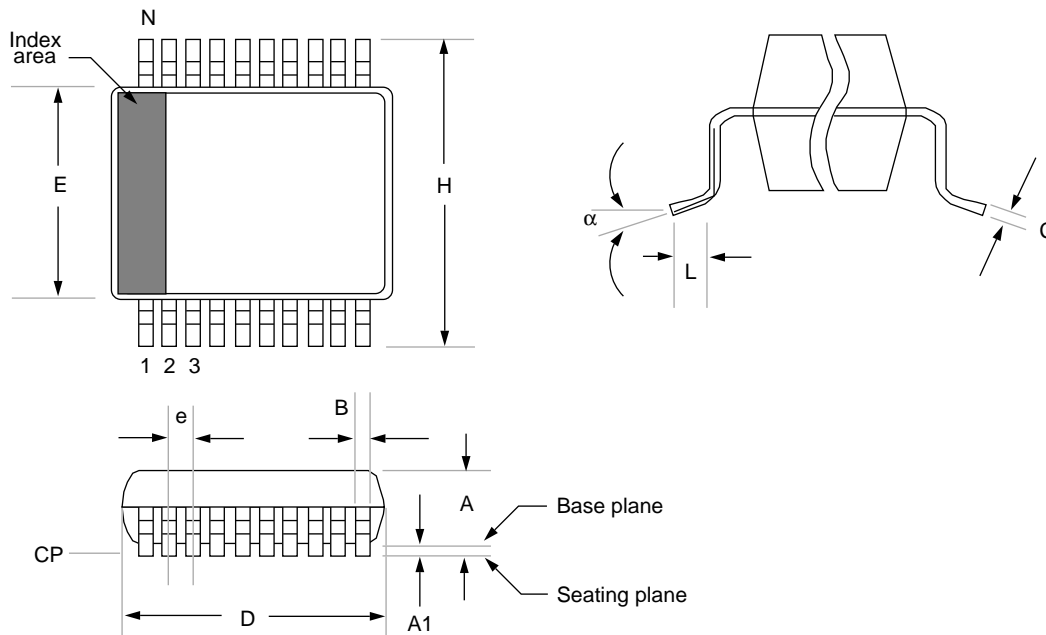
## 14.3 18-Lead Plastic Surface Mount (SOIC - Wide, 300 mil Body)



Package Group: Plastic SOIC (SO)						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
$\alpha$	0°	8°		0°	8°	
A	2.362	2.642		0.093	0.104	
A1	0.101	0.300		0.004	0.012	
B	0.355	0.483		0.014	0.019	
C	0.241	0.318		0.009	0.013	
D	11.353	11.735		0.447	0.462	
E	7.416	7.595		0.292	0.299	
e	1.270	1.270	Reference	0.050	0.050	Reference
H	10.007	10.643		0.394	0.419	
h	0.381	0.762		0.015	0.030	
L	0.406	1.143		0.016	0.045	
N	18	18		18	18	
CP	—	0.102		—	0.004	

# PIC16C62X

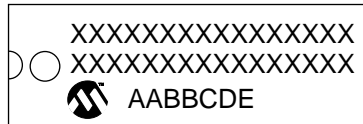
## 14.4 20-Lead Plastic Surface Mount (SSOP - 209 mil Body 5.30 mm)



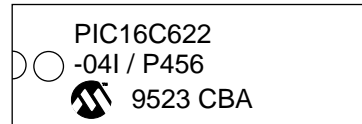
Package Group: Plastic SSOP						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
$\alpha$	0°	8°		0°	8°	
A	1.730	1.990		0.068	0.078	
A1	0.050	0.210		0.002	0.008	
B	0.250	0.380		0.010	0.015	
C	0.130	0.220		0.005	0.009	
D	7.070	7.330		0.278	0.289	
E	5.200	5.380		0.205	0.212	
e	0.650	0.650	Reference	0.026	0.026	Reference
H	7.650	7.900		0.301	0.311	
L	0.550	0.950		0.022	0.037	
N	20	20		20	20	
CP	-	0.102		-	0.004	

## 14.5 Package Marking Information

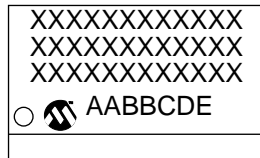
### 18-Lead PDIP



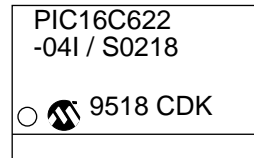
### Example



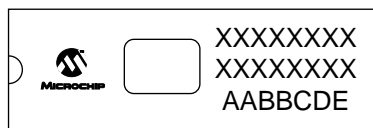
### 18-Lead SOIC (.300")



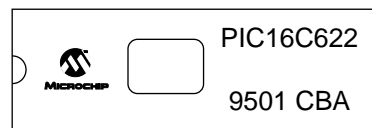
### Example



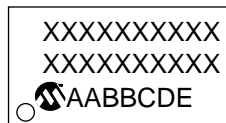
### 18-Lead CERDIP Windowed



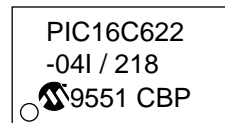
### Example



### 20-Lead SSOP



### Example



<b>Legend:</b>	MM...M	Microchip part number information
	XX...X	Customer specific information*
	AA	Year code (last 2 digits of calendar year)
	BB	Week code (week of January 1 is week '01')
	C	Facility code of the plant at which wafer is manufactured C = Chandler, Arizona, U.S.A.
	D	Mask revision number
	E	Assembly code of the plant or country of origin in which part was assembled

**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.

\* Standard OTP marking consists of Microchip part number, year code, week code, facility code, mask rev#, and assembly code. For OTP marking beyond this, certain price adders apply. Please check with your Microchip Sales Office. For QTP devices, any special marking adders are included in QTP price.

# PIC16C62X

---

NOTES:



## APPENDIX A: ENHANCEMENTS

The following are the list of enhancements over the PIC16C5X microcontroller family:

1. Instruction word length is increased to 14 bits. This allows larger page sizes both in program memory (4K now as opposed to 512 before) and register file (up to 128 bytes now versus 32 bytes before).
2. A PC high latch register (PCLATH) is added to handle program memory paging. PA2, PA1, PA0 bits are removed from STATUS register.
3. Data memory paging is slightly redefined. STATUS register is modified.
4. Four new instructions have been added: RETURN, RETFIE, ADDLW, and SUBLW. Two instructions TRIS and OPTION are being phased out although they are kept for compatibility with PIC16C5X.
5. OPTION and TRIS registers are made addressable.
6. Interrupt capability is added. Interrupt vector is at 0004h.
7. Stack size is increased to 8 deep.
8. Reset vector is changed to 0000h.
9. Reset of all registers is revisited. Five different reset (and wake-up) types are recognized. Registers are reset differently.
10. Wake up from SLEEP through interrupt is added.
11. Two separate timers, Oscillator Start-up Timer (OST) and Power-up Timer (PWRT) are included for more reliable power-up. These timers are invoked selectively to avoid unnecessary delays on power-up and wake-up.
12. PORTB has weak pull-ups and interrupt on change feature.
13. Timer0 clock input, T0CKI pin is also a port pin (RA4/T0CKI) and has a TRIS bit.
14. FSR is made a full 8-bit register.
15. "In-circuit programming" is made possible. The user can program PIC16CXX devices using only five pins: VDD, VSS, /VPP, RB6 (clock) and RB7 (data in/out).
16. PCON status register is added with a Power-on-Reset ( $\overline{\text{POR}}$ ) status bit and a Brown-out Reset status bit ( $\overline{\text{BOR}}$ ).
17. Code protection scheme is enhanced such that portions of the program memory can be protected, while the remainder is unprotected.
18. PORTA inputs are now Schmitt Trigger inputs.
19. Brown-out Reset reset has been added.

## APPENDIX B: COMPATIBILITY

To convert code written for PIC16C5X to PIC16CXX, the user should take the following steps:

1. Remove any program memory page select operations (PA2, PA1, PA0 bits) for CALL, GOTO.
2. Revisit any computed jump operations (write to PC or add to PC, etc.) to make sure page bits are set properly under the new scheme.
3. Eliminate any data memory page switching. Redefine data variables to reallocate them.
4. Verify all writes to STATUS, OPTION, and FSR registers since these have changed.
5. Change reset vector to 0000h.

## APPENDIX C: WHAT'S NEW

The format of certain sections of this data sheet have been changed to be consistent with other product families.

1. PORTB input buffers have changed to TTL from Schmitt Trigger.

## APPENDIX D: WHAT'S CHANGED

1. Table 3-1 was changed to reflect the TTL input buffers on PORTB.
2. Figure 5-5 and Figure 5-6 were updated to reflect the TTL input buffers on PORTB.
3. Figure 9-7 was updated.
4. The orientation of the diode in Figure 9-20 was changed.
5. A device specification for JW devices was added to Table 12-1.
6. Max spec for Brown-out Reset current was changed to 400  $\mu$ A in Section 12.1 and Section 12.2.
7. Information added to support the 2.5V "LC" devices.

## APPENDIX E: PIC16/17 MICROCONTROLLERS

### PIC14XXX FAMILY OF DEVICES

	Clock	Memory	Peripherals	Features
PIC14000	20	4K	192	TMRO
		Maximum Frequency of Operation (MHz)		
		EPROM		
		Data Memory (bytes)		
		Program Memory		
		Timer Modules		
		Capture/Compare/PWM Modules		
		Serial Port(s) (SPI/I <sup>2</sup> C, USART)		
		Parallel Slave Port		
		A/D Converter (16-bit Channels)		
		Interrupt Sources		
		I/O Pins		
		Voltage Range (Volts)		
		In-Circuit Serial Programming		
		Brown-out Reset		
		Packages		
				28-pin SDIP, SOIC, SSOP

# PIC16C62X

## PIC16C5X FAMILY OF DEVICES

Device	Clock		Memory		Peripherals		Features		
	Maximum Frequency of Operation (MHz)	Program Memory (Words)	RAM Data Memory (bytes)	Timer Module(s)	I/O Pins	Voltage Range (Volts)	Number of Instructions	Packages	
PIC16C52	4	384	—	25	TMR0	12	2.5-6.25	33	18-pin DIP, SOIC
PIC16C54	20	512	—	25	TMR0	12	2.5-6.25	33	18-pin DIP, SOIC; 20-pin SSOP
PIC16C54A	20	512	—	25	TMR0	12	2.5-6.25	33	18-pin DIP, SOIC; 20-pin SSOP
PIC16CR54A	20	—	512	25	TMR0	12	2.0-6.25	33	18-pin DIP, SOIC; 20-pin SSOP
PIC16C55	20	512	—	24	TMR0	20	2.5-6.25	33	28-pin DIP, SOIC, SSOP
PIC16C56	20	1K	—	25	TMR0	12	2.5-6.25	33	18-pin DIP, SOIC; 20-pin SSOP
PIC16C57	20	2K	—	72	TMR0	20	2.5-6.25	33	28-pin DIP, SOIC, SSOP
PIC16CR57B	20	—	2K	72	TMR0	20	2.5-6.25	33	28-pin DIP, SOIC, SSOP
PIC16C58A	20	2K	—	73	TMR0	12	2.5-6.25	33	18-pin DIP, SOIC; 20-pin SSOP
PIC16CR58A	20	—	2K	73	TMR0	12	2.5-6.25	33	18-pin DIP, SOIC; 20-pin SSOP

All PIC16/17 Family devices have Power-On Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability.

## PIC16C62X FAMILY OF DEVICES

	Clock		Memory		Peripherals		Features				
	Maximum Frequency of Operation (MHz)	Program Memory	Timer Module(s)	Comparator(s)	Internal Reference Voltage	I/O Pins	Voltage Range (Volts)	Brown-out Reset	Packages		
PIC16C620	20	512	80	TMR0	2	Yes	4	13	2.5-6.0	Yes	18-pin DIP, SOIC; 20-pin SSOP
PIC16C621	20	1K	80	TMR0	2	Yes	4	13	2.5-6.0	Yes	18-pin DIP, SOIC; 20-pin SSOP
PIC16C622	20	2K	128	TMR0	2	Yes	4	13	2.5-6.0	Yes	18-pin DIP, SOIC; 20-pin SSOP

All PIC16/17 Family devices have Power-on Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability.

All PIC16C62X Family devices use serial programming with clock pin RB6 and data pin RB7.

# PIC16C62X

## PIC16C6X FAMILY OF DEVICES

PIC16C62	Clock			Memory				Peripherals				Features		
	Maximum Frequency of Operation (MHz)	Program Memory (Words)	Data Memory (bytes)	ROM	EPROM	Timer Module(s)	Serial Ports (SPI/I <sup>2</sup> C, USART)	Parallel Slave Port	IO Pins	Voltage Range (Volts)	In-Circuit Serial Programming	Brown-out Reset	Packages	
PIC16C62	20	2K	—	128	—	TMR0, TMR1, TMR2	1 SPI/I <sup>2</sup> C	—	7	22	3.0-6.0	Yes	—	28-pin SDIP, SOIC, SSOP
PIC16C62A <sup>(1)</sup>	20	2K	—	128	—	TMR0, TMR1, TMR2	1 SPI/I <sup>2</sup> C	—	7	22	3.0-6.0	Yes	Yes	28-pin SDIP, SOIC, SSOP
PIC16CR62 <sup>(1)</sup>	20	—	2K	128	—	TMR0, TMR1, TMR2	1 SPI/I <sup>2</sup> C	—	7	22	3.0-6.0	Yes	Yes	28-pin SDIP, SOIC, SSOP
PIC16C63	20	4K	—	192	—	TMR0, TMR1, TMR2	2 SPI/I <sup>2</sup> C, USART	—	10	22	3.0-6.0	Yes	Yes	28-pin SDIP, SOIC
PIC16CR63 <sup>(1)</sup>	20	—	4K	192	—	TMR0, TMR1, TMR2	2 SPI/I <sup>2</sup> C, USART	—	10	22	3.0-6.0	Yes	Yes	28-pin SDIP, SOIC
PIC16C64	20	2K	—	128	—	TMR0, TMR1, TMR2	1 SPI/I <sup>2</sup> C	Yes	8	33	3.0-6.0	Yes	—	40-pin DIP; 44-pin PLCC, MQFP
PIC16C64A <sup>(1)</sup>	20	2K	—	128	—	TMR0, TMR1, TMR2	1 SPI/I <sup>2</sup> C	Yes	8	33	3.0-6.0	Yes	Yes	40-pin DIP; 44-pin PLCC, MQFP, TQFP
PIC16CR64 <sup>(1)</sup>	20	—	2K	128	—	TMR0, TMR1, TMR2	1 SPI/I <sup>2</sup> C	Yes	8	33	3.0-6.0	Yes	Yes	40-pin DIP; 44-pin PLCC, MQFP, TQFP
PIC16C65	20	4K	—	192	—	TMR0, TMR1, TMR2	2 SPI/I <sup>2</sup> C, USART	Yes	11	33	3.0-6.0	Yes	—	40-pin DIP; 44-pin PLCC, MQFP
PIC16C65A <sup>(1)</sup>	20	4K	—	192	—	TMR0, TMR1, TMR2	2 SPI/I <sup>2</sup> C, USART	Yes	11	33	3.0-6.0	Yes	Yes	40-pin DIP; 44-pin PLCC, MQFP, TQFP
PIC16CR65 <sup>(1)</sup>	20	—	4K	192	—	TMR0, TMR1, TMR2	2 SPI/I <sup>2</sup> C, USART	Yes	11	33	3.0-6.0	Yes	Yes	40-pin DIP; 44-pin PLCC, MQFP, TQFP

All PIC16/17 family devices have Power-on Reset, selectable Watchdog Timer, selectable code protect, and high I/O current capability.

All PIC16C6X family devices use serial programming with clock pin RB6 and data pin RB7.

Note 1: Please contact your local sales office for availability of these devices.

## PIC16C7X FAMILY OF DEVICES

Device	Clock		Memory		Peripherals				Features			
	Maximum Frequency of Operation (MHz)	Program Memory (Words)	Data Memory (bytes)	Timer Module(s)	Serial Ports (SPI/I <sup>2</sup> C, USART)	Parallel Slave Port	A/D Converter (8-bit Channels)	I/O Pins	Voltage Range (Volts)	In-Circuit Serial Programming	Brown-out Reset	Packages
PIC16C710	20	512	36	TMR0	—	—	4	4	13	3.0-6.0	Yes	18-pin DIP, SOIC; 20-pin SSOP
PIC16C71	20	1K	36	TMR0	—	—	4	4	13	3.0-6.0	Yes	18-pin DIP, SOIC
PIC16C711	20	1K	68	TMR0	—	—	4	4	13	3.0-6.0	Yes	18-pin DIP, SOIC; 20-pin SSOP
PIC16C72	20	2K	128	TMR0, TMR1, TMR2	1	SPI/I <sup>2</sup> C	—	5	8	3.0-6.0	Yes	28-pin SDIP, SOIC, SSOP
PIC16C73	20	4K	192	TMR0, TMR1, TMR2	2	SPI/I <sup>2</sup> C, USART	—	5	11	3.0-6.0	Yes	28-pin SDIP, SOIC
PIC16C73A <sup>(1)</sup>	20	4K	192	TMR0, TMR1, TMR2	2	SPI/I <sup>2</sup> C, USART	—	5	11	3.0-6.0	Yes	28-pin SDIP, SOIC
PIC16C74	20	4K	192	TMR0, TMR1, TMR2	2	SPI/I <sup>2</sup> C, USART	Yes	8	12	3.0-6.0	Yes	40-pin DIP; 44-pin PLCC, MQFP
PIC16C74A <sup>(1)</sup>	20	4K	192	TMR0, TMR1, TMR2	2	SPI/I <sup>2</sup> C, USART	Yes	8	12	3.0-6.0	Yes	40-pin DIP; 44-pin PLCC, MQFP, TQFP

All PIC16/17 Family devices have Power-on Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability.

All PIC16C7X Family devices use serial programming with clock pin RB6 and data pin RB7.  
 Note 1: Please contact your local sales office for availability of these devices.

# PIC16C62X

## PIC16C8X FAMILY OF DEVICES

Device	Clock		Memory		Peripherals		Features			
	Maximum Frequency of Operation (MHz)	Program Memory	Data Memory (bytes)	Data EEPROM (bytes)	Timer Module(s)	Interrupt Sources	I/O Pins	Voltage Range (Volts)		
PIC16C83 <sup>(1)</sup>	10	512	—	36	64	TMR0	4	13	2.0-6.0	18-pin DIP, SOIC
PIC16CR83 <sup>(1)</sup>	10	—	512	36	64	TMR0	4	13	2.0-6.0	18-pin DIP, SOIC
PIC16C84	10	1K	—	36	64	TMR0	4	13	2.0-6.0	18-pin DIP, SOIC
PIC16C84A <sup>(1)</sup>	10	1K	—	68	64	TMR0	4	13	2.0-6.0	18-pin DIP, SOIC
PIC16CR84 <sup>(1)</sup>	10	—	1K	68	64	TMR0	4	13	2.0-6.0	18-pin DIP, SOIC

All PIC16/17 family devices have Power-on Reset, selectable Watchdog Timer, selectable code protect, and high I/O current capability.

All PIC16C8X family devices use serial programming with clock pin RB6 and data pin RB7.

Note 1: Please contact your local sales office for availability of these devices.



## PIC17CXX FAMILY OF DEVICES

Device	Clock		Memory		Peripherals				Features						
	Maximum Frequency of Operation (MHz)	Program Memory (Words)	ROM	RAM Data Memory (bytes)	Timer Modules	Captures PMS	Serial Ports (USART)	Hardware Multiply	External Interrupts	I/O Pins	Voltage Range (Volts)	Number of Instructions	Packages		
PIC17C42	25	2K	—	232	TMR0, TMR1, TMR2, TMR3	2	2	Yes	—	Yes	11	33	4.5-5.5	55	40-pin DIP; 44-pin PLCC, MQFP
PIC17C42A	25	2K	—	232	TMR0, TMR1, TMR2, TMR3	2	2	Yes	Yes	Yes	11	33	4.5-5.5	58	40-pin DIP; 44-pin PLCC, MQFP
PIC17CR42	25	—	2K	232	TMR0, TMR1, TMR2, TMR3	2	2	Yes	Yes	Yes	11	33	4.5-5.5	58	40-pin DIP; 44-pin PLCC, MQFP
PIC17C43	25	4K	—	454	TMR0, TMR1, TMR2, TMR3	2	2	Yes	Yes	Yes	11	33	2.5-6.0	58	40-pin DIP; 44-pin PLCC, TQFP, MQFP
PIC17CR43	25	—	4K	454	TMR0, TMR1, TMR2, TMR3	2	2	Yes	Yes	Yes	11	33	2.5-6.0	58	40-pin DIP; 44-pin PLCC, TQFP, MQFP
PIC17C44	25	8K	—	454	TMR0, TMR1, TMR2, TMR3	2	2	Yes	Yes	Yes	11	33	2.5-6.0	58	40-pin DIP; 44-pin PLCC, TQFP, MQFP

All PIC16/17 Family devices have Power-on Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability.

# PIC16C62X

---

## PIN COMPATIBILITY

Devices that have the same package type and VDD, VSS and MCLR pin locations are said to be pin compatible. This allows these different devices to operate in the same socket. Compatible devices may only require minor software modification to allow proper operation in the application socket (ex., PIC16C56 and PIC16C61 devices). Not all devices in the same package size are pin compatible; for example, the PIC16C62 is compatible with the PIC16C63, but not the PIC16C55.

Pin compatibility does not mean that the devices offer the same features. As an example, the PIC16C54 is pin compatible with the PIC16C71, but does not have an A/D converter, weak pull-ups on PORTB, or interrupts.

**TABLE 1: PIN COMPATIBLE DEVICES**

Pin Compatible Devices	Package
PIC16C54, PIC16C54A, PIC16CR54, PIC16CR54A, PIC16C56, PIC16C58A, PIC16CR58A, PIC16C61, PIC16C620, PIC16C621, PIC16C622, PIC16C710, PIC16C71, PIC16C711, PIC16C83, PIC16CR83, PIC16C84, PIC16C84A, PIC16CR84	18-pin (20-pin)
PIC16C55, PIC16C57, PIC16CR57B	28-pin
PIC16C62, PIC16CR62, PIC16C62A, PIC16C63, PIC16C72, PIC16C73, PIC16C73A	28-pin
PIC16C64, PIC16CR64, PIC16C64A, PIC16C65, PIC16C65A, PIC16C74, PIC16C74A	40-pin
PIC17C42, PIC17C43, PIC17C44	40-pin

## INDEX

### A

ADDLW Instruction .....	61
ADDWF Instruction .....	61
ANDLW Instruction .....	61
ANDWF Instruction .....	61
Architectural Overview .....	7
Assembler .....	74

### B

BCF Instruction .....	62
Block Diagram	
TIMER0 .....	29
TMR0/WDT PRESCALER .....	32
Brown-Out Detect (BOD) .....	48
BSF Instruction .....	62
BTFSC Instruction .....	62
BTFSS Instruction .....	63

### C

C Compiler (MP-C) .....	71, 75
CALL Instruction .....	63
Clocking Scheme/Instruction Cycle .....	10
CLRF Instruction .....	63
CLRW Instruction .....	63
CLRWDW Instruction .....	64
CMCON Register .....	35
Code Protection .....	58
COMF Instruction .....	64
Comparator Configuration .....	36
Comparator Interrupts .....	39
Comparator Module .....	35
Comparator Operation .....	37
Comparator Reference .....	37
Configuration Bits .....	44
Configuring the Voltage Reference .....	42

### D

Data Memory Organization .....	12
DECF Instruction .....	64
DECFSZ Instruction .....	64
Development Support .....	71
Development Systems .....	75
Development Tools .....	71
Device Characterization Information .....	89
Dynamic Data Exchange (DDE) .....	71

### E

External Crystal Oscillator Circuit .....	46
---	----

### F

Family of Devices	
PIC14XXX .....	99
PIC16C5X .....	100
PIC16C62X .....	101
PIC16C6X .....	102
PIC16C7X .....	103
PIC16C8X .....	104
PIC17CXX .....	105
Fuzzy Logic Dev. System ( <i>fuzzyTECH</i> ®-MP) .....	71, 75

### G

General purpose Register File .....	12
GOTO Instruction .....	65

### I

I/O Ports .....	23
I/O Programming Considerations .....	28
ID Locations .....	58
INCF Instruction .....	65
INCFSZ Instruction .....	65
In-Circuit Serial Programming .....	58
Indirect Addressing, INDF and FSR Registers .....	21
Instruction Flow/Pipelining .....	10
Instruction Set	
ADDLW .....	61
ADDWF .....	61
ANDLW .....	61
ANDWF .....	61
BCF .....	62
BSF .....	62
BTFSC .....	62
BTFSS .....	63
CALL .....	63
CLRF .....	63
CLRW .....	63
CLRWDW .....	64
COMF .....	64
DECF .....	64
DECFSZ .....	64
GOTO .....	65
INCF .....	65
INCFSZ .....	65
IORLW .....	65
IORWF .....	66
MOVF .....	66
MOVLW .....	66
MOVWF .....	66
NOP .....	67
OPTION .....	67
RETFIE .....	67
RETLW .....	67
RETURN .....	68
RLF .....	68
RRF .....	68
SLEEP .....	68
SUBLW .....	69
SUBWF .....	69
SWAPF .....	70
TRIS .....	70
XORLW .....	70
XORWF .....	70
Instruction Set Summary .....	59
INT Interrupt .....	54
INTCON Register .....	17
Interrupts .....	53
IORLW Instruction .....	65
IORWF Instruction .....	66

### M

MOVF Instruction .....	66
MOVLW Instruction .....	66
MOVWF Instruction .....	66
MPASM Assembler .....	71, 74
MP-C C Compiler .....	75
MPSIM Software Simulator .....	71, 75

### N

NOP Instruction .....	67
-----------------------	----

# PIC16C62X

---

## O

One-Time-Programmable (OTP) Devices .....	5
OPTION Instruction .....	67
OPTION Register .....	16
Oscillator Configurations .....	45
Oscillator Start-up Timer (OST) .....	48

## P

Package Marking Information .....	95
Packaging Information .....	91
PCL and PCLATH .....	20
PCON Register .....	19
PICDEM-1 Low-Cost PIC16/17 Demo Board .....	71, 73
PICDEM-2 Low-Cost PIC16CXX Demo Board .....	71, 73
PICMASTER Probe .....	72
PICMASTER System Configuration .....	71
PICMASTER™ RT In-Circuit Emulator .....	71
PICSTART™ Low-Cost Development System .....	71, 73
PIE1 Register .....	18
Pin Compatible Devices .....	106
Pinout Description .....	9
PIR1 Register .....	18
Port RB Interrupt .....	54
PORTA .....	23
PORTB .....	26
Power Control/Status Register (PCON) .....	49
Power-Down Mode (SLEEP) .....	57
Power-On Reset (POR) .....	48
Power-up Timer (PWRT) .....	48
Prescaler .....	32
PRO MATE™ Universal Programmer .....	71, 73
Program Memory Organization .....	11

## Q

Quick-Turnaround-Production (QTP) Devices .....	5
---	---

## R

RC Oscillator .....	46
Reset .....	47
RETFIE Instruction .....	67
RETLW Instruction .....	67
RETURN Instruction .....	68
RLF Instruction .....	68
RRF Instruction .....	68

## S

Serialized Quick-Turnaround-Production (SQTP) Devices ..	5
SLEEP Instruction .....	68
Software Simulator (MPSIM) .....	75
Special Features of the CPU .....	43
Special Function Registers .....	14
Stack .....	20
Status Register .....	15
SUBLW Instruction .....	69
SUBWF Instruction .....	69
SWAPF Instruction .....	70

## T

Timer0	
TIMER0 .....	29
TIMER0 (TMR0) Interrupt .....	29
TIMER0 (TMR0) Module .....	29
TMR0 with External Clock .....	31
Timer1	
Switching Prescaler Assignment .....	33
Timing Diagrams and Specifications .....	84

TMR0 Interrupt .....	54
TRIS Instruction .....	70
TRISA .....	23
TRISB .....	26

## V

Voltage Reference Module .....	41
VRCON Register .....	41

## W

Watchdog Timer (WDT) .....	55
----------------------------	----

## X

XORLW Instruction .....	70
XORWF Instruction .....	70

## List of Examples

Example 3-1:	Instruction Pipeline Flow .....	10
Example 4-1:	Indirect Addressing .....	21
Example 5-1:	Initializing PORTA .....	23
Example 5-2:	Read-Modify-Write Instructions on an I/O Port.....	28
Example 6-1:	Changing Prescaler (Timer0→WDT) .....	33
Example 6-2:	Changing Prescaler (WDT→Timer0) .....	33
Example 7-1:	Initializing Comparator Module.....	37
Example 8-1:	Voltage Reference Configuration.....	42
Example 9-1:	Saving the Status and W Registers in RAM .....	55

## List of Figures

Figure 3-1:	PIC16C62X Block Diagram .....	8
Figure 3-2:	Clock/Instruction Cycle .....	10
Figure 4-1:	Program Memory Map and Stack for the PIC16C620 .....	11
Figure 4-2:	Program Memory Map and Stack for the PIC16C621 .....	11
Figure 4-3:	Program Memory Map and Stack for the PIC16C622 .....	11
Figure 4-4:	Data Memory Map for the PIC16C620/621 .....	13
Figure 4-5:	Data Memory Map for the PIC16C622 .....	13
Figure 4-6:	STATUS Register (Address 03h or 83h) ...	15
Figure 4-7:	OPTION Register (Address 81h) .....	16
Figure 4-8:	INTCON Register (Address 0Bh or 8Bh) ...	17
Figure 4-9:	PIE1 Register (Address 8Ch) .....	18
Figure 4-10:	PIR1 Register (Address 0Ch) .....	18
Figure 4-11:	PCON Register (Address 8Eh) .....	19
Figure 4-12:	Loading Of PC In Different Situations.....	20
Figure 4-13:	Direct/Indirect Addressing PIC16C62X.....	21
Figure 5-1:	Block Diagram of RA1:RA0 Pins .....	23
Figure 5-2:	Block Diagram of RA2 Pin .....	23
Figure 5-3:	Block Diagram of RA3 Pin .....	24
Figure 5-4:	Block Diagram of RA4 Pin .....	24
Figure 5-5:	Block Diagram of RB7:RB4 Pins .....	26
Figure 5-6:	Block Diagram of RB3:RB0 Pins .....	26
Figure 5-7:	Successive I/O Operation .....	28
Figure 6-1:	TIMER0 Block Diagram .....	29
Figure 6-2:	TIMER0 (TMR0) Timing: Internal Clock/No Prescaler .....	29
Figure 6-3:	TIMER0 Timing: Internal Clock/Prescale 1:2.....	30
Figure 6-4:	TIMER0 Interrupt Timing .....	30
Figure 6-5:	TIMER0 Timing With External Clock .....	31
Figure 6-6:	Block Diagram of the Timer0/WDT Prescaler.....	32
Figure 7-1:	CMCON Register (Address 1Fh) .....	35
Figure 7-2:	Comparator I/O Operating Modes .....	36
Figure 7-3:	Single Comparator.....	37
Figure 7-4:	Comparator Output Block Diagram.....	38
Figure 7-5:	Analog Input Model.....	39
Figure 8-1:	VRCON Register(Address 9Fh).....	41
Figure 8-2:	Voltage Reference Block Diagram.....	41
Figure 8-3:	Voltage Reference Output Buffer Example.....	42
Figure 9-1:	Configuration Word.....	44
Figure 9-2:	Crystal Operation (or Ceramic Resonator) (HS, XT or LP Osc Configuration) .....	45
Figure 9-3:	External Clock Input Operation (HS, XT or LP Osc Configuration) .....	45
Figure 9-4:	External Parallel Resonant Crystal Oscillator Circuit.....	46
Figure 9-5:	External Series Resonant Crystal Oscillator Circuit.....	46
Figure 9-6:	RC Oscillator Mode.....	46
Figure 9-7:	Simplified Block Diagram of On-chip Reset Circuit .....	47
Figure 9-8:	Brown-out Situations.....	48
Figure 9-9:	Time-out Sequence on Power-up (MCLR not tied to V <sub>DD</sub> ): Case 1 .....	51
Figure 9-10:	Time-out Sequence on Power-up (MCLR not tied to V <sub>DD</sub> ): Case 2 .....	51
Figure 9-11:	Time-out Sequence on Power-up (MCLR tied to V <sub>DD</sub> ).....	51
Figure 9-12:	External Power-on Reset Circuit (For Slow V <sub>DD</sub> Power-up) .....	52

# PIC16C62X

Figure 9-13:	External Brown-out Protection Circuit 1 ....	52
Figure 9-14:	External Brown-out Protection Circuit 2 ....	52
Figure 9-15:	Interrupt Logic .....	53
Figure 9-16:	INT Pin Interrupt Timing .....	54
Figure 9-17:	Watchdog Timer Block Diagram .....	56
Figure 9-18:	Summary of Watchdog Timer Registers ...	56
Figure 9-19:	Wake-up from Sleep Through Interrupt.....	57
Figure 9-20:	Typical In-Circuit Serial Programming Connection .....	58
Figure 10-1:	General Format for Instructions .....	59
Figure 11-1:	PICMASTER System Configuration.....	71
Figure 12-1:	Load Conditions .....	82
Figure 12-2:	External Clock Timing .....	83
Figure 12-3:	CLKOUT and I/O Timing .....	84
Figure 12-4:	Reset, Watchdog Timer, Oscillator Start-Up Timer and Power-Up Timer Timing .....	85
Figure 12-5:	Brown-out Reset Timing.....	85
Figure 12-6:	TIMER0 Clock Timing .....	86
Figure 12-7:	Load Conditions .....	86

## List of Tables

Table 1-1:	PIC16C62X Family of Devices .....	4
Table 3-1:	PIC16C62X Pinout Description.....	9
Table 4-1:	Special Registers for the PIC16C62X.....	14
Table 5-1:	PORTA Functions.....	25
Table 5-2:	Summary of Registers Associated with PORTA .....	25
Table 5-3:	PORTB Functions.....	27
Table 5-4:	Summary of Registers Associated with PORTB .....	27
Table 6-1:	Registers Associated with Timer0 .....	33
Table 7-1:	Registers Associated with Comparator Module .....	40
Table 8-1:	Registers Associated with Voltage Reference .....	42
Table 9-1:	Capacitor Selection for Ceramic Resonators (Preliminary) .....	45
Table 9-2:	Capacitor Selection for Crystal Oscillator (Preliminary) .....	45
Table 9-3:	Time-out in Various Situations.....	49
Table 9-4:	Status Bits and Their Significance .....	49
Table 9-5:	Initialization Condition for Special Registers.....	50
Table 9-6:	Initialization Condition for Registers .....	50
Table 10-1:	OPCODE Field Descriptions.....	59
Table 10-2:	PIC16CXX Instruction Set .....	60
Table 11-1:	PICMASTER Probe Specification .....	72
Table 11-2:	Development System Packages.....	75
Table 12-1:	Cross Reference of Device Specs for Oscillator Configurations and Frequencies of Operation (Commercial Devices).....	77
Table 12-2:	Comparator Specifications.....	81
Table 12-3:	Voltage Reference Specifications .....	81
Table 12-4:	External Clock Timing Requirements .....	83
Table 12-5:	CLKOUT and I/O Timing Requirements ...	84
Table 12-6:	Reset, Watchdog Timer, Oscillator Start-up Timer and Power-up Timer Requirements .....	85
Table 12-7:	TIMER0 Clock Requirements .....	86

## CONNECTING TO MICROCHIP BBS

Connect worldwide to the Microchip BBS using the CompuServe® communications network. In most cases a local call is your only expense. The Microchip BBS connection does not use CompuServe membership services, therefore **you do not need CompuServe membership to join Microchip's BBS.**

There is **no charge** for connecting to the BBS, except for a toll charge to the CompuServe access number, where applicable. You do not need to be a CompuServe member to take advantage of this connection (you never actually log in to CompuServe).

The procedure to connect will vary slightly from country to country. Please check with your local CompuServe agent for details if you have a problem. CompuServe service allows multiple users at baud rates up to 14400 bps.

The following connect procedure applies in most locations:

1. Set your modem to 8-bit, No parity, and One stop (8N1). This is not the normal CompuServe setting which is 7E1.
2. Dial your local CompuServe access number.
3. Depress **<ENTER>** and a garbage string will appear because CompuServe is expecting a 7E1 setting.
4. Type **+**, depress **<ENTER>** and Host Name: will appear.
5. Type **MCHIPBBS**, depress **< ENTER>** and you will be connected to the Microchip BBS.

In the United States, to find CompuServe's phone number closest to you, set your modem to 7E1 and dial (800) 848-4480 for 300-2400 baud or (800) 331-7166 for 9600-14400 baud connection. After the system responds with Host Name:

Type **NETWORK**, depress **< ENTER>** and follow CompuServe's directions.

For voice information (or calling from overseas), you may call (614) 723-1550 for your local CompuServe number.

### Trademarks:

The Microchip name, logo and PIC are registered trademarks in the USA and other countries. PICMASTER and PICSTART are registered trademarks of Microchip Technology Incorporated.

PIC is a registered trademark of Microchip Technology Incorporated in the U.S.A.

MPLAB, PRO MATE and fuzzyLAB are trademarks, and SQTP is a service mark of Microchip Technology Incorporated.

fuzzyTECH is a registered trademark of Inform Software Corporation.

I<sup>2</sup>C is a trademark of Philips Corporation.

IBM, IBM PC-AT are registered trademarks of International Business Machines Corp.

Pentium is a trademark of Intel Corporation.

MS-DOS and Microsoft Windows are registered trademarks of Microsoft Corporation. Windows is a trademark of Microsoft Corporation.

CompuServe is a registered trademark of CompuServe Incorporated.

All other trademarks mentioned herein are the property of their respective companies.

# PIC16C62X

---

---

## READER RESPONSE

It is our intention to provide you with the best documentation possible to ensure successful use of your Microchip product. If you wish to provide your comments on organization, clarity, subject matter, and ways in which our documentation can better serve you, please FAX your comments to the Technical Publications Manager at (602) 786-7578.

Please list the following information, and use this outline to provide us with your comments about this Data Sheet.

To: Technical Publications Manager Total Pages Sent \_\_\_\_\_

RE: Reader Response

From: Name \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

City / State / ZIP / Country \_\_\_\_\_

Telephone: (\_\_\_\_\_) \_\_\_\_\_ - \_\_\_\_\_ FAX: (\_\_\_\_\_) \_\_\_\_\_ - \_\_\_\_\_

Application (optional):

Would you like a reply? \_\_\_Y \_\_\_N

Device: **PIC16C62X**

Literature Number: **DS30235D**

Questions:

1. What are the best features of this document?

\_\_\_\_\_  
\_\_\_\_\_

2. How does this document meet your hardware and software development needs?

\_\_\_\_\_  
\_\_\_\_\_

3. Do you find the organization of this data sheet easy to follow? If not, why?

\_\_\_\_\_  
\_\_\_\_\_

4. What additions to the data sheet do you think would enhance the structure and subject?

\_\_\_\_\_  
\_\_\_\_\_

5. What deletions from the data sheet could be made without affecting the overall usefulness?

\_\_\_\_\_  
\_\_\_\_\_

6. Is there any incorrect or misleading information (what and where)?

\_\_\_\_\_  
\_\_\_\_\_

7. How would you improve this document?

\_\_\_\_\_  
\_\_\_\_\_

8. How would you improve our software, systems, and silicon products?

\_\_\_\_\_  
\_\_\_\_\_



## PIC16C62X Product Identification System

To order or to obtain information, e.g., on pricing or delivery, please use the listed part numbers, and refer to the factory or the listed sales offices.

PART NO.	-XX	X	/XX	XXX	
					<b>Pattern:</b> 3-Digit Pattern Code for QTP (blank otherwise)
					<b>Package:</b> P = PDIP SO = SOIC (Gull Wing, 300 mil body) SS = SSOP (209 mil) JW* = Windowed CERDIP
					<b>Temperature Range:</b> - = 0°C to +70°C I = -40°C to +85°C E = -40°C to +125°C
					<b>Frequency Range:</b> 04 = 200kHz (LP osc) 04 = 4 MHz (XT and RC osc) 20 = 20 MHz (HS osc)
					<b>Device:</b> PIC16C62X :VDD range 4.0V to 6.0V PIC16C62XT:VDD range 4.0V to 6.0V (Tape and Reel) PIC16LC62X:VDD range 2.5V to 6.0V PIC16LC62XT:VDD range 2.5V to 6.0V (Tape and Reel)

**Examples:**

a) PIC16C62X - 04/P 301 = Commercial temp., PDIP package, 4 MHz, normal VDD limits, QTP pattern #301.

b) PIC16LC62X- 04I/SO = Industrial temp., SOIC package, 200kHz, extended VDD limits.

\* JW Devices are UV erasable and can be programmed to any device configuration. JW Devices meet the electrical requirement of each oscillator type (including LC devices).

## Sales and Support

Products supported by a preliminary Data Sheet may possibly have an errata sheet describing minor operational differences and recommended workarounds. To determine if an errata sheet exists for a particular device, please contact one of the following:

1. Your local Microchip sales office.
2. The Microchip Corporate Literature Center U.S. FAX: (602) 786-7277
3. The Microchip's Bulletin Board, via your local CompuServe number (CompuServe membership NOT required).

Please specify which device, revision of silicon and Data Sheet (include Literature #) you are using.

For latest version information and upgrade kits for Microchip Development Tools, please call 1-800-755-2345 or 1-602-786-7302.

# PIC16C62X

---

NOTES:

NOTES:

---

---

# WORLDWIDE SALES & SERVICE

---

---

## AMERICAS

### Corporate Office

Microchip Technology Inc.  
2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 602 786-7200 Fax: 602 786-7277  
Technical Support: 602 786-7627  
Web: <http://www.mchip.com/microchip>

### Atlanta

Microchip Technology Inc.  
500 Sugar Mill Road, Suite 200B  
Atlanta, GA 30350  
Tel: 770 640-0034 Fax: 770 640-0307

### Boston

Microchip Technology Inc.  
5 Mount Royal Avenue  
Marlborough, MA 01752  
Tel: 508 480-9990 Fax: 508 480-8575

### Chicago

Microchip Technology Inc.  
333 Pierce Road, Suite 180  
Itasca, IL 60143  
Tel: 708 285-0071 Fax: 708 285-0075

### Dallas

Microchip Technology Inc.  
14651 Dallas Parkway, Suite 816  
Dallas, TX 75240-8809  
Tel: 214 991-7177 Fax: 214 991-8588

### Dayton

Microchip Technology Inc.  
Suite 150  
Two Prestige Place  
Miamisburg, OH 45342  
Tel: 513 291-1654 Fax: 513 291-9175

### Los Angeles

Microchip Technology Inc.  
18201 Von Karman, Suite 1090  
Irvine, CA 92715  
Tel: 714 263-1888 Fax: 714 263-1338

## AMERICAS (continued)

### New York

Microchip Technology Inc.  
150 Motor Parkway, Suite 416  
Hauppauge, NY 11788  
Tel: 516 273-5305 Fax: 516 273-5335

### San Jose

Microchip Technology Inc.  
2107 North First Street, Suite 590  
San Jose, CA 95131  
Tel: 408 436-7950 Fax: 408 436-7955

## CANADA

Microchip Technology Inc.  
5925 Airport Road, Suite 200  
Mississauga, Ontario L4V 1W1, Canada  
Tel: 905 405-6279 Fax: 905 405-6253

## ASIA/PACIFIC

### Hong Kong

Microchip Technology  
Unit No. 3002-3004, Tower 1  
Metroplaza  
223 Hing Fong Road  
Kwai Fong, N.T. Hong Kong  
Tel: 852 2 401 1200 Fax: 852 2 401 3431

### Korea

Microchip Technology  
168-1, Youngbo Bldg. 3 Floor  
Samsung-Dong, Kangnam-Ku,  
Seoul, Korea  
Tel: 82 2 554 7200 Fax: 82 2 558 5934

### Singapore

Microchip Technology  
200 Middle Road  
#10-03 Prime Centre  
Singapore 188980  
Tel: 65 334 8870 Fax: 65 334 8850

### Taiwan

Microchip Technology  
10F-1C 207  
Tung Hua North Road  
Taipei, Taiwan, ROC  
Tel: 886 2 717 7175 Fax: 886 2 545 0139

## EUROPE

### United Kingdom

Arizona Microchip Technology Ltd.  
Unit 6, The Courtyard  
Meadow Bank, Furlong Road  
Bourne End, Buckinghamshire SL8 5AJ  
Tel: 44 1 628 850303 Fax: 44 1 628 850178

### France

Arizona Microchip Technology SARL  
Zone Industrielle de la Bonde  
2 Rue du Buisson aux Fraises  
91300 Massy - France  
Tel: 33 1 69 53 63 20 Fax: 33 1 69 30 90 79

### Germany

Arizona Microchip Technology GmbH  
Gustav-Heinemann-Ring 125  
D-81739 Muenchen, Germany  
Tel: 49 89 627 144 0 Fax: 49 89 627 144 44

### Italy

Arizona Microchip Technology SRL  
Centro Direzionale Colleoni  
Palazzo Taurus 1 V. Le Colleoni 1  
20041 Agrate Brianza, Milan Italy  
Tel: 39 39 689 9939 Fax: 39 39 689 9883

## JAPAN

Microchip Technology Intl. Inc.  
Benex S-1 6F  
3-18-20, Shin Yokohama  
Kohoku-Ku, Yokohama  
Kanagawa 222 Japan  
Tel: 81 45 471 6166 Fax: 81 45 471 6122

4/08/96



**MICROCHIP**

All rights reserved. © 1996, Microchip Technology Incorporated, USA.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights. The Microchip logo and name are registered trademarks of Microchip Technology Inc. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.