Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	20MHz
Connectivity	-
Peripherals	Brown-out Detect/Reset, POR, WDT
Number of I/O	13
Program Memory Size	3.5KB (2K x 14)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 5.5V
Data Converters	-
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SSOP (0.209", 5.30mm Width)
Supplier Device Package	20-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16c622at-20i-ss

NOTES:

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>).

REGISTER 4-3: INTCON REGISTER (ADDRESS 0BH OR 8BH)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all un-masked interrupts
0 = Disables all interrupts

bit 6 **PEIE**: Peripheral Interrupt Enable bit
1 = Enables all un-masked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **T0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE**: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

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

bit 2 **T0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF**: RB Port Change Interrupt Flag bit
1 = When at least one of the RB<7:4> pins changed state (must be cleared in software)
0 = None of the RB<7:4> pins have changed state

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	’1’ = Bit is set	’0’ = Bit is cleared	x = Bit is unknown

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 bi-directional 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

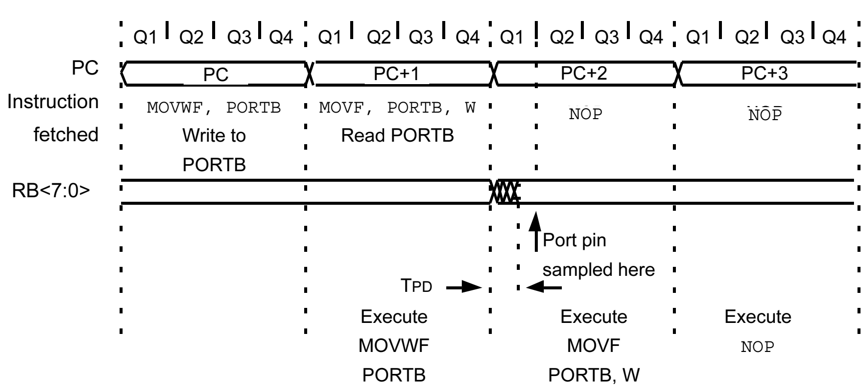
```
; Initial PORT settings:    PORTB<7:4> Inputs
;
;                           PORTB<3:0> Outputs
; PORTB<7:6> have external pull-up and are not
; connected to other circuitry
;
;                           PORT latch   PORT pins
;                           ----------   ---------
                                         -

    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



Note:

This example shows write to PORTB followed by a read from PORTB.

Note that:

data setup time = (0.25 T_{CY} - T_{PD}) where T_{CY} = instruction cycle and TPD = propagation delay of Q1 cycle to output valid.

Therefore, at higher clock frequencies, a write followed by a read may be problematic.

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.

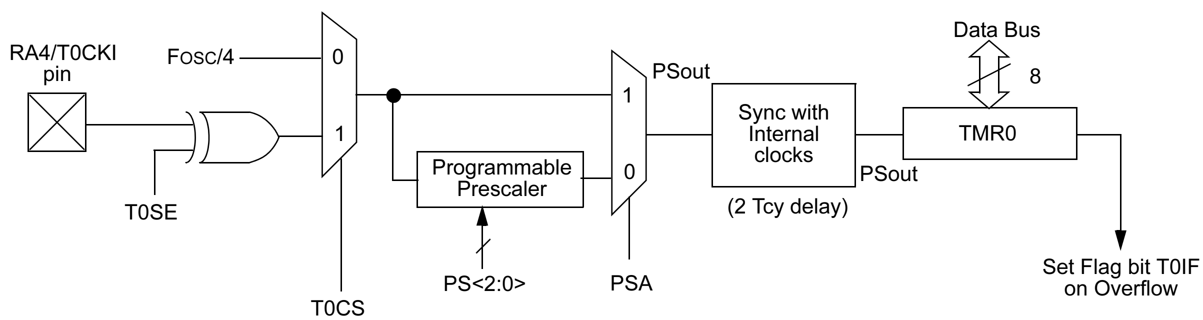
Counter mode is selected by setting the T0CS bit. In this mode, Timer0 will increment either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the source edge (T0SE) control bit (OPTION<4>). Clearing the T0SE bit selects the rising edge. Restrictions on the external clock input are discussed in detail in Section 6.2.

The prescaler is shared between the Timer0 module and the Watchdog Timer. The prescaler assignment is controlled in software by the control bit PSA (OPTION<3>). Clearing the PSA bit will assign the prescaler to Timer0. The prescaler is not readable or writable. When the prescaler is assigned to the Timer0 module, prescale value of 1:2, 1:4, ..., 1:256 are selectable. Section 6.3 details the operation of the prescaler.

6.1 TIMER0 Interrupt

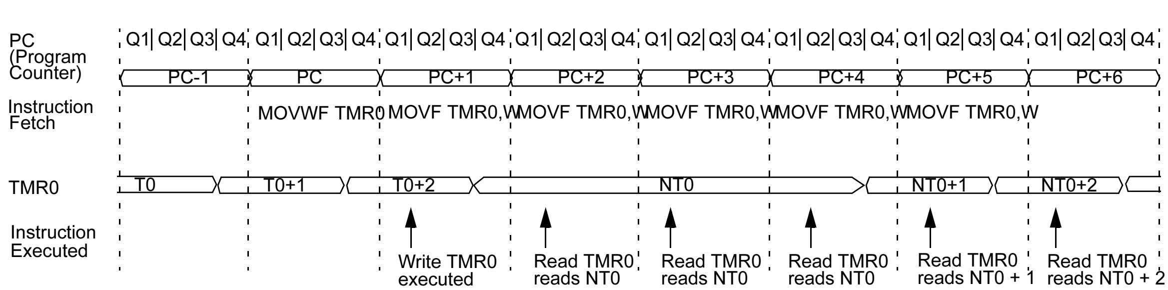
Timer0 interrupt is generated when the TMR0 register timer/counter overflows from FFh to 00h. This overflow sets the T0IF bit. The interrupt can be masked by clearing the T0IE bit (INTCON<5>). The T0IF bit (INTCON<2>) must be cleared in software by the Timer0 module interrupt service routine before re-enabling this interrupt. The Timer0 interrupt cannot wake the processor from SLEEP, since the timer is shut off during SLEEP. See Figure 6-4 for Timer0 interrupt timing.

FIGURE 6-1: TIMER0 BLOCK DIAGRAM



Note 1: Bits T0SE, T0CS, PS2, PS1, PS0 and PSA are located in the OPTION register.
2: The prescaler is shared with Watchdog Timer (Figure 6-6).

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

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

To change prescaler from the WDT to the TMR0 module, use the sequence shown in Example 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'xxxx0xxx'         ;Select TMR0, new
                                ;prescale value and
                                ;clock source
MOVWF       OPTION_REG
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	Value on All Other RESETS
01h	TMR0	Timer0 module register								xxxx xxxx	uuuu uuuu
0Bh/8Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
81h	OPTION	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
85h	TRISA	—	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111

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

Note: Shaded bits are not used by TMR0 module.

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

REGISTER 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
bit 7							bit 0

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

bit 6 **C1OUT**: Comparator 1 output
1 = C1 VIN+ > C1 VIN-
0 = 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.

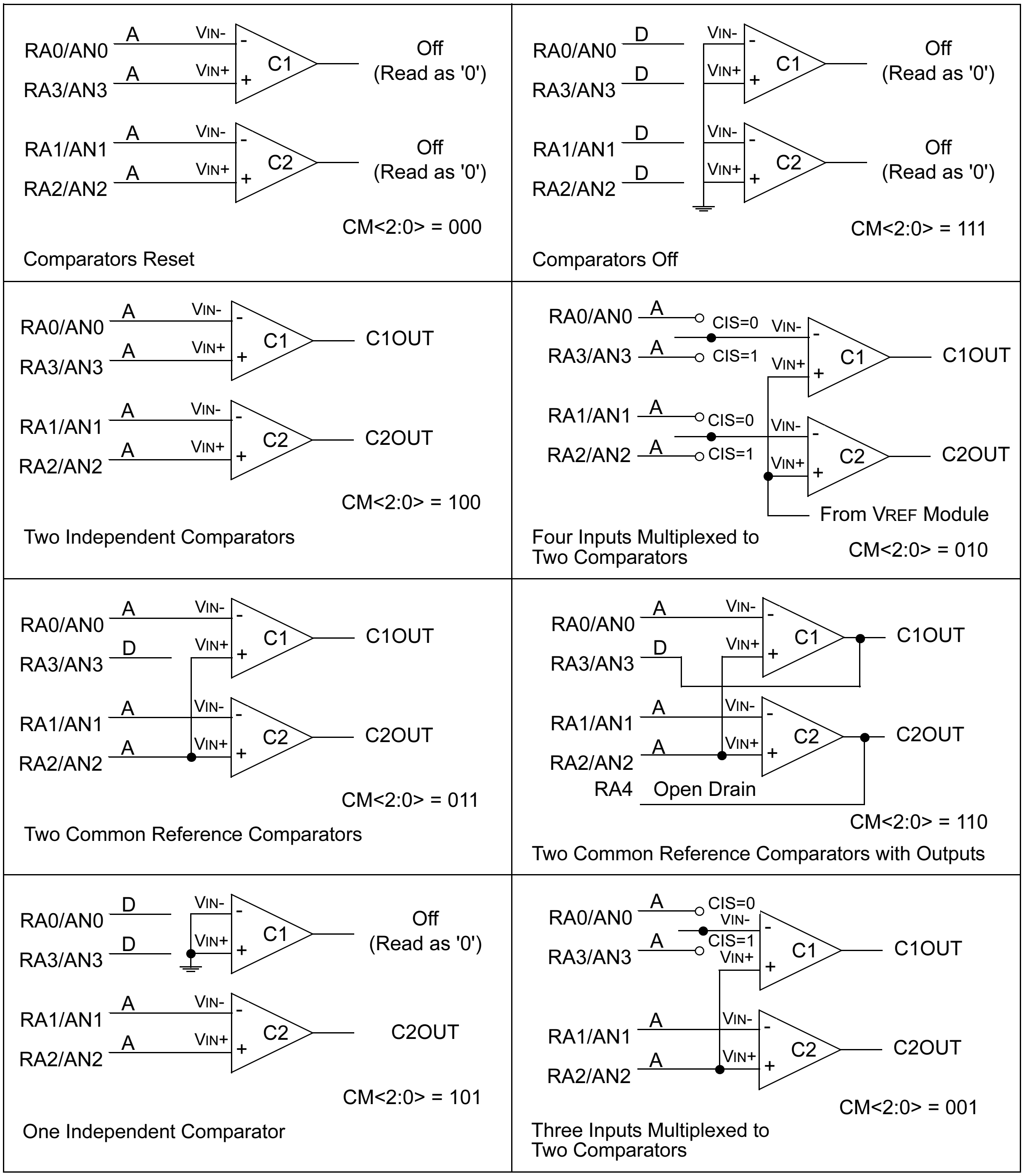
Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	’1’ = Bit is set	’0’ = Bit is cleared	x = Bit is unknown

7.1 Comparator Configuration

There are eight modes of operation for the comparators. The CMCON register is used to select the mode. Figure 7-1 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 7-1: COMPARATOR I/O OPERATING MODES



A = Analog Input, Port Reads Zeros Always
D = Digital Input
CIS = CMCON<3>, Comparator Input Switch

9.5 Interrupts

The PIC16C62X has 4 sources of interrupt:

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

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

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

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

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

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

When an interrupt is responded to, the GIE is cleared to disable any further interrupt, the return address is pushed into the stack and the PC is loaded with 0004h.

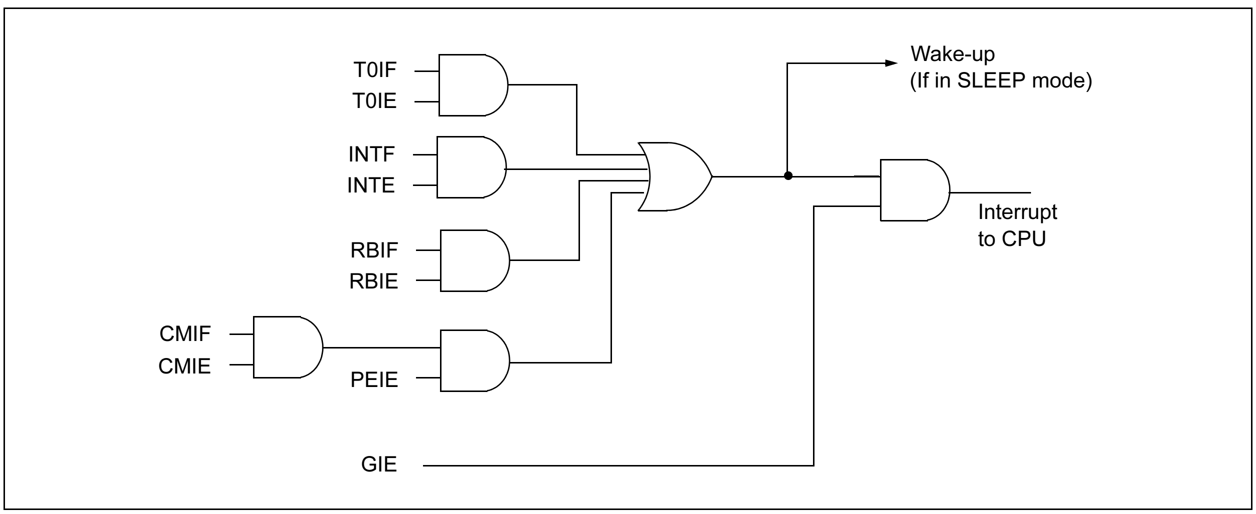
Once in the interrupt service routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid RB0/INT recursive interrupts.

For external interrupt events, such as the INT pin or PORTB change interrupt, the interrupt latency will be three or four instruction cycles. The exact latency depends 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.

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

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

FIGURE 9-15: INTERRUPT LOGIC



10.1 Instruction Descriptions

ADDLW **Add Literal and W**

Syntax:	[*label*] ADDLW k
Operands:	$0 \le k \le 255$
Operation:	(W) + k → (W)
Status Affected:	C, DC, Z
Encoding:	`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

ADDWF **Add W and f**

Syntax:	[*label*] ADDWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W) + (f) → (dest)
Status Affected:	C, DC, Z
Encoding:	`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

ANDLW **AND Literal with W**

Syntax:	[*label*] ANDLW k
Operands:	$0 \le k \le 255$
Operation:	(W) .AND. (k) → (W)
Status Affected:	Z
Encoding:	`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

ANDWF **AND W with f**

Syntax:	[*label*] ANDWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W) .AND. (f) → (dest)
Status Affected:	Z
Encoding:	`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

MOVF **Move f**

Syntax:	[*label*] MOVF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) → (dest)
Status Affected:	Z
Encoding:	`00` `1000` `dfff` `ffff`
Description:	The contents of register f is moved to a destination dependent 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	`MOVF FSR,` 0 After Instruction W = value in FSR register Z = 1

MOVWF **Move W to f**

Syntax:	[*label*] MOVWF f
Operands:	0 ≤ f ≤ 127
Operation:	(W) → (f)
Status Affected:	None
Encoding:	`00` `0000` `1fff` `ffff`
Description:	Move data from W register to register 'f'.
Words:	1
Cycles:	1
Example	`MOVWF OPTION` Before Instruction OPTION = 0xFF W = 0x4F After Instruction OPTION = 0x4F W = 0x4F

NOP **No Operation**

Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Encoding:	`00` `0000` `0xx0` `0000`
Description:	No operation.
Words:	1
Cycles:	1
Example	`NOP`

OPTION **Load Option Register**

Syntax:	[*label*] OPTION
Operands:	None
Operation:	(W) → OPTION
Status Affected:	None
Encoding:	`00` `0000` `0110` `0010`
Description:	The contents of the W register are loaded in the OPTION register. This instruction is supported for code compatibility with PIC16C5X products. Since OPTION is a readable/writable register, the user can directly address it.
Words:	1
Cycles:	1
Example	**To maintain upward compatibility with future PICmicro® products, do not use this instruction.**

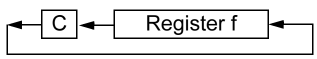
RLF — Rotate Left f through Carry

Syntax:	[*label*] RLF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [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
```

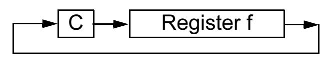
RRF — Rotate Right f through Carry

Syntax:	[*label*] RRF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [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
```

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`

11.3 MPLAB C17 and MPLAB C18 C Compilers

The MPLAB C17 and MPLAB C18 Code Development Systems are complete ANSI C compilers for Microchip's PIC17CXXX and PIC18CXXX family of microcontrollers. These compilers provide powerful integration capabilities, superior code optimization and ease of use not found with other compilers.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

11.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK object linker combines relocatable objects created by the MPASM assembler and the MPLAB C17 and MPLAB C18 C compilers. It can link relocatable objects from pre-compiled libraries, using directives from a linker script.

The MPLIB object librarian manages the creation and modification of library files of pre-compiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

11.5 MPLAB C30 C Compiler

The MPLAB C30 C compiler is a full-featured, ANSI compliant, optimizing compiler that translates standard ANSI C programs into dsPIC30F assembly language source. The compiler also supports many command-line options and language extensions to take full advantage of the dsPIC30F device hardware capabilities, and afford fine control of the compiler code generator.

MPLAB C30 is distributed with a complete ANSI C standard library. All library functions have been validated and conform to the ANSI C library standard. The library includes functions for string manipulation, dynamic memory allocation, data conversion, timekeeping, and math functions (trigonometric, exponential and hyperbolic). The compiler provides symbolic information for high level source debugging with the MPLAB IDE.

11.6 MPLAB ASM30 Assembler, Linker, and Librarian

MPLAB ASM30 assembler produces relocatable machine code from symbolic assembly language for dsPIC30F devices. MPLAB C30 compiler uses the assembler to produce it's object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire dsPIC30F instruction set
- Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- MPLAB IDE compatibility

11.7 MPLAB SIM Software Simulator

The MPLAB SIM software simulator allows code development in a PC hosted environment by simulating the PICmicro series microcontrollers on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a file, or user defined key press, to any pin. The execution can be performed in Single-Step, Execute Until Break, or Trace mode.

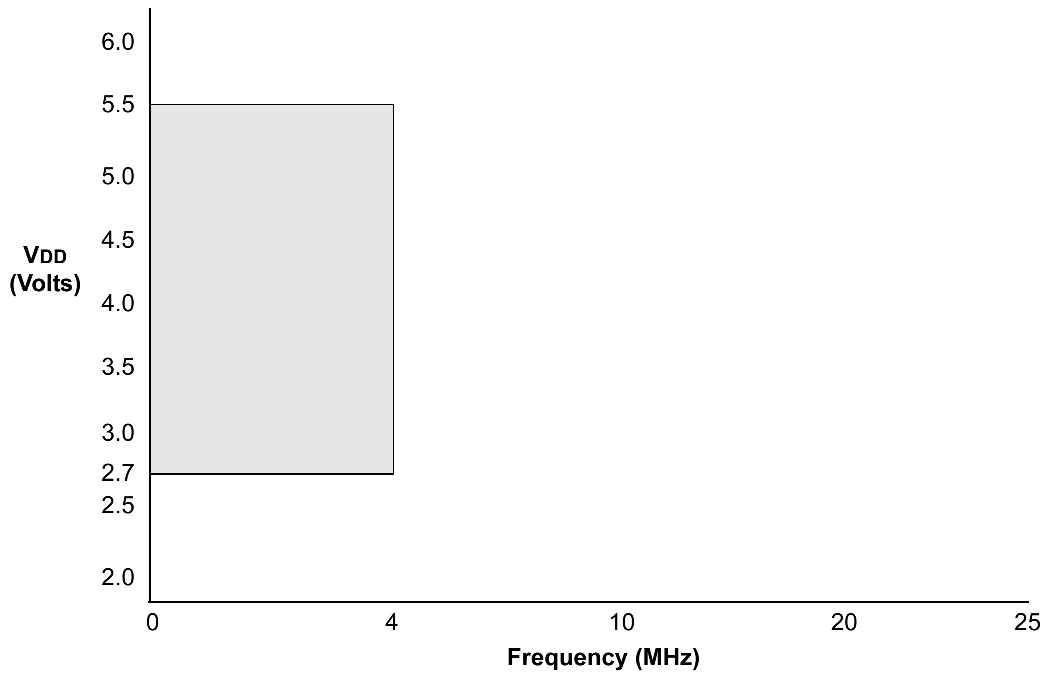
The MPLAB SIM simulator fully supports symbolic debugging using the MPLAB C17 and MPLAB C18 C Compilers, as well as the MPASM assembler. The software simulator offers the flexibility to develop and debug code outside of the laboratory environment, making it an excellent, economical software development tool.

11.8 MPLAB SIM30 Software Simulator

The MPLAB SIM30 software simulator allows code development in a PC hosted environment by simulating the dsPIC30F series microcontrollers on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a file, or user defined key press, to any of the pins.

The MPLAB SIM30 simulator fully supports symbolic debugging using the MPLAB C30 C Compiler and MPLAB ASM30 assembler. The simulator runs in either a Command Line mode for automated tasks, or from MPLAB IDE. This high speed simulator is designed to debug, analyze and optimize time intensive DSP routines.

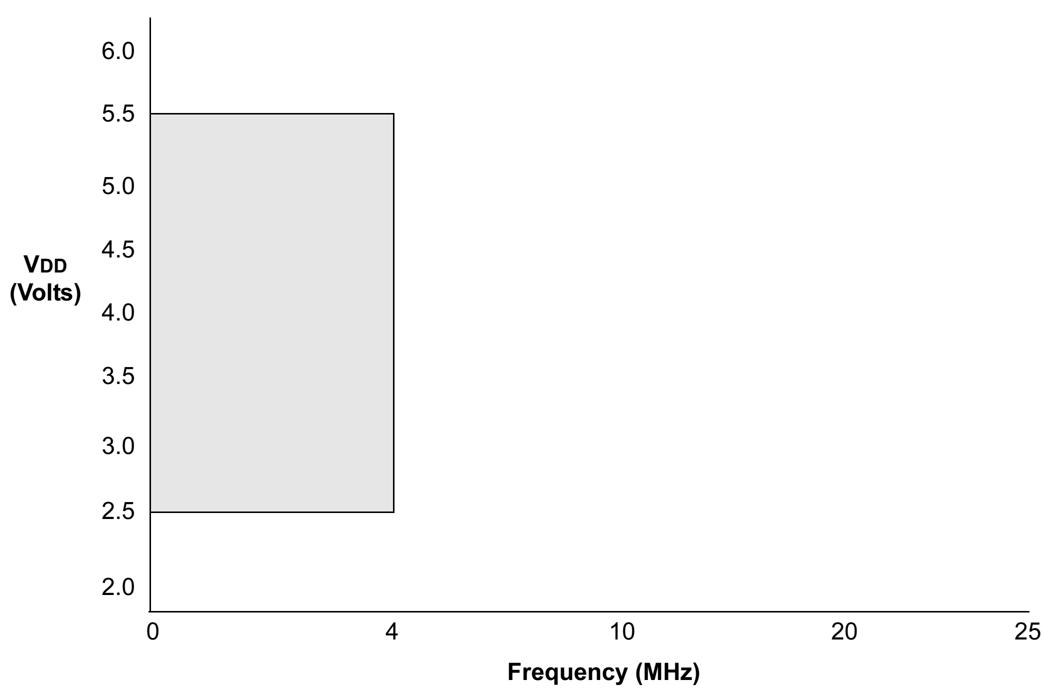
FIGURE 12-5: PIC16LC620A/LC621A/LC622A VOLTAGE-FREQUENCY GRAPH, -40°C ≤ TA ≤ 0°C



Note 1: The shaded region indicates the permissible combinations of voltage and frequency.

2: The maximum rated speed of the part limits the permissible combinations of voltage and frequency. Please reference the Product Identification System section for the maximum rated speed of the parts.

FIGURE 12-6: PIC16LC620A/LC621A/LC622A VOLTAGE-FREQUENCY GRAPH, 0°C ≤ TA ≤ +125°C



Note 1: The shaded region indicates the permissible combinations of voltage and frequency.

2: The maximum rated speed of the part limits the permissible combinations of voltage and frequency. Please reference the Product Identification System section for the maximum rated speed of the parts.

12.3 DC CHARACTERISTICS: PIC16CR62XA-04 (Commercial, Industrial, Extended) PIC16CR62XA-20 (Commercial, Industrial, Extended) PIC16LCR62XA-04 (Commercial, Industrial, Extended)

PIC16CR62XA-04 **PIC16CR62XA-20**			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial and 0°C ≤ TA ≤ +70°C for commercial and -40°C ≤ TA ≤ +125°C for extended				
PIC16LCR62XA-04			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial and 0°C ≤ TA ≤ +70°C for commercial and -40°C ≤ TA ≤ +125°C for extended				
Param. No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
D001	VDD	Supply Voltage	3.0	—	5.5	V	See Figures 12-7, 12-8, 12-9
D001	VDD	Supply Voltage	2.5	—	5.5	V	See Figures 12-7, 12-8, 12-9
D002	VDR	RAM Data Retention Voltage(1)	—	1.5*	—	V	Device in SLEEP mode
D002	VDR	RAM Data Retention Voltage(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
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
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	4.35	V	BOREN configuration bit is cleared
D005	VBOR	Brown-out Detect Voltage	3.7	4.0	4.35	V	BOREN configuration bit is cleared
D010	IDD	Supply Current(2)	— — — — — —	1.2 500 1.0 4.0 3.0 35	1.7 900 2.0 7.0 6.0 70	mA μA mA mA mA μA	FOSC = 4 MHz, VDD = 5.5V, WDT disabled, XT mode, (**Note 4**)* FOSC = 4 MHz, VDD = 3.0V, WDT disabled, XT mode, (**Note 4**) FOSC = 10 MHz, VDD = 3.0V, WDT disabled, HS mode, (**Note 6**) FOSC = 20 MHz, VDD = 5.5V, WDT disabled*, HS mode FOSC = 20 MHz, VDD = 4.5V, WDT disabled, HS mode FOSC = 32 kHz, VDD = 3.0V, WDT disabled, LP mode
D010	IDD	Supply Current(2)	— — —	1.2 400 35	1.7 800 70	mA μA μA	FOSC = 4.0 MHz, VDD = 5.5V, WDT disabled, XT mode, (**Note 4**)* FOSC = 4.0 MHz, VDD = 2.5V, WDT disabled, XT mode (**Note 4**) FOSC = 32 kHz, VDD = 2.5V, WDT disabled, LP mode

12.4 DC Characteristics: PIC16C62X/C62XA/CR62XA (Commercial, Industrial, Extended) PIC16LC62X/LC62XA/LCR62XA (Commercial, Industrial, Extended)

PIC16C62X/C62XA/CR62XA	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C $\leq T_A \leq$ +85°C for industrial and 0°C $\leq T_A \leq$ +70°C for commercial and -40°C $\leq T_A \leq$ +125°C for extended
PIC16LC62X/LC62XA/LCR62XA	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C $\leq T_A \leq$ +85°C for industrial and 0°C $\leq T_A \leq$ +70°C for commercial and -40°C $\leq T_A \leq$ +125°C for extended

Param. No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
	V_{IL}	**Input Low Voltage**					
		I/O ports					
D030		with TTL buffer	V_{SS}	—	0.8V 0.15 V_{DD}	V	V_{DD} = 4.5V to 5.5V otherwise
D031		with Schmitt Trigger input	V_{SS}	—	0.2 V_{DD}	V	
D032		$\overline{MCLR}$, RA4/T0CKI,OSC1 (in RC mode)	V_{SS}	—	0.2 V_{DD}	V	**(Note 1)**
D033		OSC1 (in XT and HS)	V_{SS}	—	0.3 V_{DD}	V	
		OSC1 (in LP)	V_{SS}	—	0.6 V_{DD}-1.0	V	
	V_{IL}	**Input Low Voltage**					
		I/O ports					
D030		with TTL buffer	V_{SS}	—	0.8V 0.15 V_{DD}	V	V_{DD} = 4.5V to 5.5V otherwise
D031		with Schmitt Trigger input	V_{SS}	—	0.2 V_{DD}	V	
D032		$\overline{MCLR}$, RA4/T0CKI,OSC1 (in RC mode)	V_{SS}	—	0.2 V_{DD}	V	**(Note 1)**
D033		OSC1 (in XT and HS)	V_{SS}	—	0.3 V_{DD}	V	
		OSC1 (in LP)	V_{SS}	—	0.6 V_{DD}-1.0	V	
	V_{IH}	Input High Voltage					
		I/O ports					
D040		with TTL buffer	2.0V 0.25 V_{DD} + 0.8V	—	V_{DD} V_{DD}	V	V_{DD} = 4.5V to 5.5V otherwise
D041		with Schmitt Trigger input	0.8 V_{DD}	—	V_{DD}		
D042		$\overline{MCLR}$ RA4/T0CKI	0.8 V_{DD}	—	V_{DD}	V	
D043 D043A		OSC1 (XT, HS and LP) OSC1 (in RC mode)	0.7 V_{DD} 0.9 V_{DD}	—	V_{DD}	V	 **(Note 1)**

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 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(A) be driven with external clock in RC mode.

2: The leakage current on the $\overline{MCLR}$ 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.

12.5 DC CHARACTERISTICS: PIC16C620A/C621A/C622A-40(7) (Commercial) PIC16CR620A-40(7) (Commercial)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C for commercial				
Param No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
D001	VDD	Supply Voltage	3.0	—	5.5	V	FOSC = DC to 20 MHz
D002	VDR	RAM Data Retention Voltage(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	3.65	4.0	4.35	V	BOREN configuration bit is cleared
D010	IDD	Supply Current(2,4)	—	1.2	2.0	mA	FOSC = 4 MHz, VDD = 5.5V, WDT disabled, XT OSC mode, **(Note 4)***
			—	0.4	1.2	mA	FOSC = 4 MHz, VDD = 3.0V, WDT disabled, XT OSC mode, **(Note 4)**
			—	1.0	2.0	mA	FOSC = 10 MHz, VDD = 3.0V, WDT disabled, HS OSC mode, **(Note 6)**
			—	4.0	6.0	mA	FOSC = 20 MHz, VDD = 4.5V, WDT disabled, HS OSC mode
			—	4.0	7.0	mA	FOSC = 20 MHz, VDD = 5.5V, WDT disabled*, HS OSC mode
			—	35	70	μA	FOSC = 32 kHz, VDD = 3.0V, WDT disabled, LP OSC mode
D020	IPD	Power Down Current(3)	—	—	2.2	μA	VDD = 3.0V
			—	—	5.0	μA	VDD = 4.5V*
			—	—	9.0	μA	VDD = 5.5V
			—	—	15	μA	VDD = 5.5V Extended
D022	ΔIWDT	WDT Current(5)	—	6.0	10	μA	VDD = 4.0V
					12	μA	(125°C)
D022A	ΔIBOR	Brown-out Reset Current(5)	—	75	125	μA	$\overline{\text{BOD}}$ enabled, VDD = 5.0V
D023	ΔICOMP	Comparator Current for each Comparator(5)	—	30	60	μA	VDD = 4.0V
D023A	ΔIVREF	VREF Current(5)	—	80	135	μA	VDD = 4.0V
	ΔIEE Write	Operating Current	—		3	mA	VCC = 5.5V, SCL = 400 kHz
	ΔIEE Read	Operating Current	—		1	mA	
	ΔIEE	Standby Current	—		30	μA	VCC = 3.0V, EE VDD = VCC
	ΔIEE	Standby Current	—		100	μA	VCC = 3.0V, EE VDD = VCC
1A	FOSC	LP Oscillator Operating Frequency	0	—	200	kHz	All temperatures
		RC Oscillator Operating Frequency	0	—	4	MHz	All temperatures
		XT Oscillator Operating Frequency	0	—	4	MHz	All temperatures
		HS Oscillator Operating Frequency	0	—	20	MHz	All temperatures

* 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, $\overline{\text{MCLR}}$ = 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 Ir = VDD/2REXT (mA) with REXT in kΩ.

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

6: Commercial temperature range only.

7: See Section 12.1 and Section 12.3 for 16C62X and 16CR62X devices for operation between 20 MHz and 40 MHz for valid modified characteristics.

12.8 Timing Parameter Symbology

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

1. TppS2ppS
2. TppS

T			
F	Frequency	T	Time

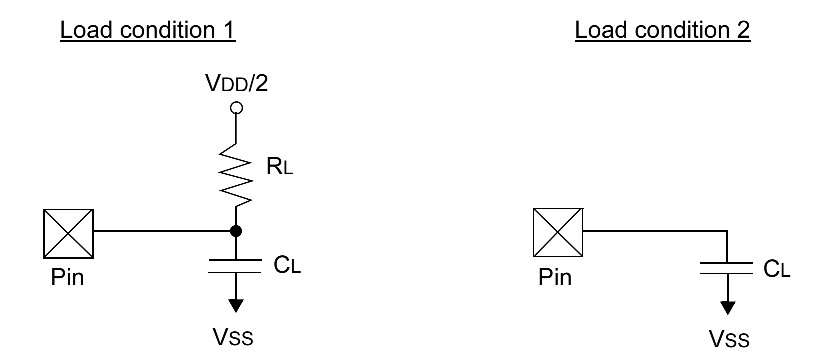
Lowercase subscripts (pp) and their meanings:

pp			
ck	CLKOUT	osc	OSC1
io	I/O port	t0	T0CKI
mc	$\overline{\text{MCLR}}$		

Uppercase letters and their meanings:

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

FIGURE 12-11: LOAD CONDITIONS



R_L = 464Ω

C_L = 50 pF for all pins except OSC2

15 pF for OSC2 output

TABLE 12-4: CLKOUT AND I/O TIMING REQUIREMENTS

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
10*	TosH2ckL	OSC1↑ to CLKOUT↓(1)	— —	75 —	200 400	ns ns	PIC16C62X(A) PIC16LC62X(A) PIC16CR62XA PIC16LCR62XA
11*	TosH2ckH	OSC1↑ to CLKOUT↑(1)	— —	75 —	200 400	ns ns	PIC16C62X(A) PIC16LC62X(A) PIC16CR62XA PIC16LCR62XA
12*	TckR	CLKOUT rise time(1)	— —	35 —	100 200	ns ns	PIC16C62X(A) PIC16LC62X(A) PIC16CR62XA PIC16LCR62XA
13*	TckF	CLKOUT fall time(1)	— —	35 —	100 200	ns ns	PIC16C62X(A) PIC16LC62X(A) PIC16CR62XA PIC16LCR62XA
14*	TckL2ioV	CLKOUT ↓ to Port out valid(1)	—	—	20	ns	
15*	TioV2ckH	Port in valid before CLKOUT ↑(1)	TOSC +200 ns TOSC +400 ns	— —	— —	ns ns	PIC16C62X(A) PIC16LC62X(A) PIC16CR62XA PIC16LCR62XA
16*	TckH2ioI	Port in hold after CLKOUT ↑(1)	0	—	—	ns	
17*	TosH2ioV	OSC1↑ (Q1 cycle) to Port out valid	— —	50	150 300	ns ns	PIC16C62X(A) PIC16LC62X(A) PIC16CR62XA PIC16LCR62XA
18*	TosH2ioI	OSC1↑ (Q2 cycle) to Port input invalid (I/O in hold time)	100 200	— —	— —	ns ns	PIC16C62X(A) PIC16LC62X(A) PIC16CR62XA PIC16LCR62XA
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(A) PIC16LC62X(A) PIC16CR62XA PIC16LCR62XA
21*	TioF	Port output fall time	— —	10 —	40 80	ns ns	PIC16C62X(A) PIC16LC62X(A) PIC16CR62XA PIC16LCR62XA
22*	Tinp	RB0/INT pin high or low time	25 40	— —	— —	ns ns	PIC16C62X(A) PIC16LC62X(A) PIC16CR62XA PIC16LCR62XA
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.

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 (480) 792-4150.

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

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: DS30235J

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 document easy to follow? If not, why?

____

____

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

____

____

5. What deletions from the document 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?

____

____