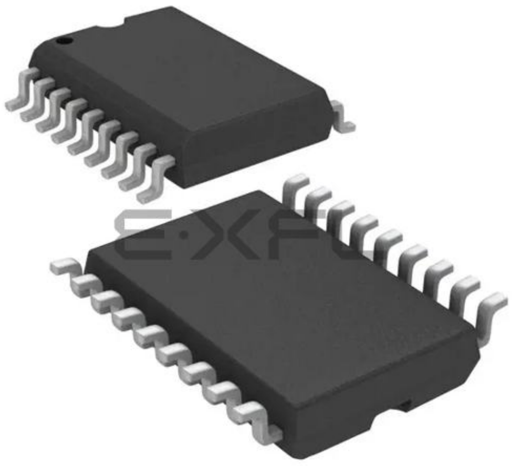
Microchip Technology - PIC16F84-04I/SO Datasheet



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	4MHz
Connectivity	-
Peripherals	POR, WDT
Number of I/O	13
Program Memory Size	1.75KB (1K x 14)
Program Memory Type	FLASH
EEPROM Size	64 x 8
RAM Size	68 x 8
Voltage - Supply (Vcc/Vdd)	4V ~ 6V
Data Converters	-
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	18-SOIC (0.295", 7.50mm Width)
Supplier Device Package	18-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16f84-04i-so

TABLE 3-1 PIC16F8X PINOUT DESCRIPTION

Pin Name	DIP No.	SOIC No.	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	16	16	I	ST/CMOS [(3)]	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	15	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.
$\overline{\text{MCLR}}$	4	4	I/P	ST	Master clear (reset) input/programming voltage input. This pin is an active low reset to the device.
					PORTA is a bi-directional I/O port.
RA0	17	17	I/O	TTL	
RA1	18	18	I/O	TTL	
RA2	1	1	I/O	TTL	
RA3	2	2	I/O	TTL	
RA4/T0CKI	3	3	I/O	ST	Can also be selected to be the clock input to the TMR0 timer/counter. Output is open drain type.
					PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	6	6	I/O	TTL/ST [(1)]	RB0/INT can also be selected as an external interrupt pin.
RB1	7	7	I/O	TTL	
RB2	8	8	I/O	TTL	
RB3	9	9	I/O	TTL	
RB4	10	10	I/O	TTL	Interrupt on change pin.
RB5	11	11	I/O	TTL	Interrupt on change pin.
RB6	12	12	I/O	TTL/ST [(2)]	Interrupt on change pin. Serial programming clock.
RB7	13	13	I/O	TTL/ST [(2)]	Interrupt on change pin. Serial programming data.
V_{SS}	5	5	P	—	Ground reference for logic and I/O pins.
V_{DD}	14	14	P	—	Positive supply for logic and I/O pins.

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

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in serial programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

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.

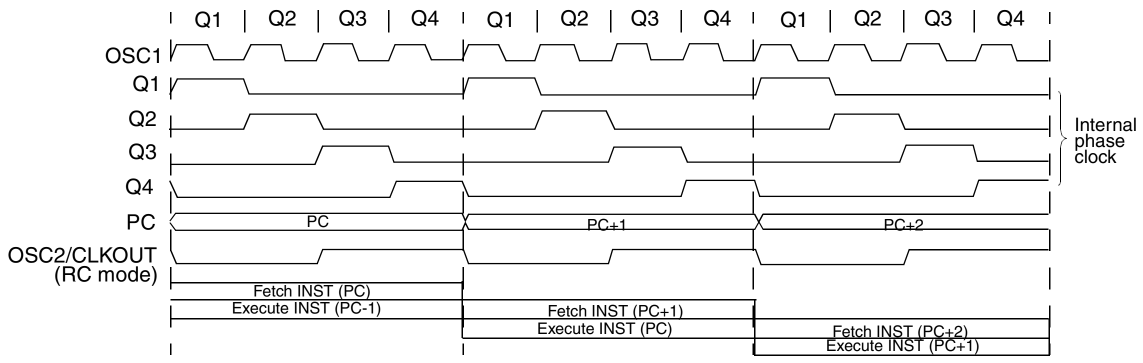
3.2 Instruction Flow/Pipelining

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" 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

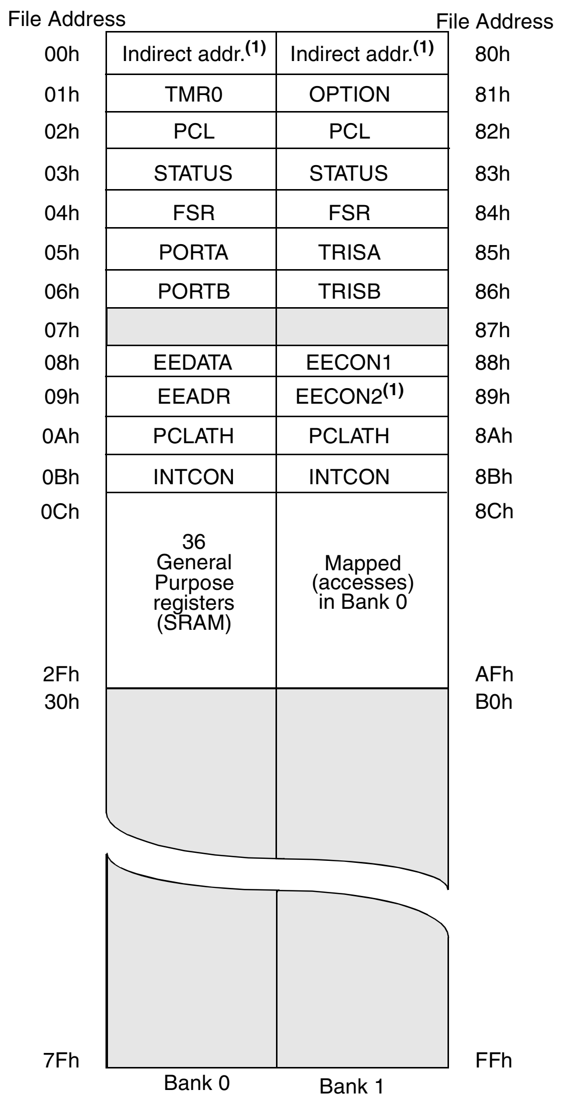


EXAMPLE 3-1: INSTRUCTION PIPELINE FLOW

`1. MOVLW 55h`	Fetch 1	Execute 1				
`2. MOVWF PORTB`		Fetch 2	Execute 2			
`3. CALL SUB_1`			Fetch 3	Execute 3		
`4. BSF   PORTA, BIT3`				Fetch 4	Flush	
					Fetch SUB_1	Execute SUB_1

All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is "flushed" from the pipeline while the new instruction is being fetched and then executed.

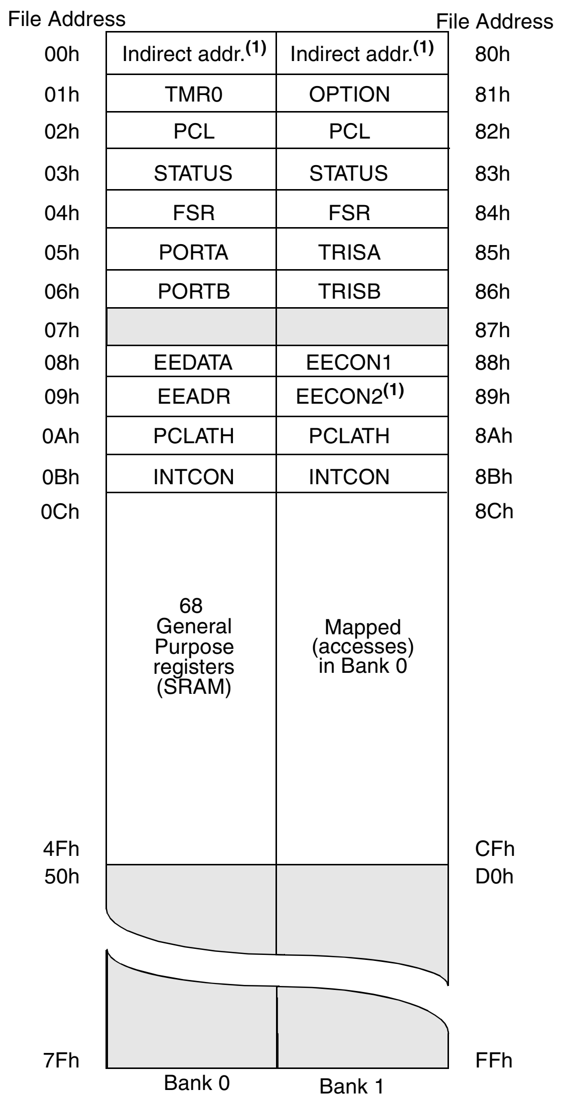
FIGURE 4-1: REGISTER FILE MAP - PIC16F83/CR83



▨ Unimplemented data memory location; read as '0'.

Note 1: Not a physical register.

FIGURE 4-2: REGISTER FILE MAP - PIC16F84/CR84



▨ Unimplemented data memory location; read as '0'.

Note 1: Not a physical register.

TABLE 4-1 REGISTER FILE SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets (Note3)
Bank 0											
00h	INDF	Uses contents of FSR to address data memory (not a physical register)								`---- ----`	`---- ----`
01h	TMR0	8-bit real-time clock/counter								`xxxx xxxx`	`uuuu uuuu`
02h	PCL	Low order 8 bits of the Program Counter (PC)								`0000 0000`	`0000 0000`
03h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	`0001 1xxx`	`000q quuu`
04h	FSR	Indirect data memory address pointer 0								`xxxx xxxx`	`uuuu uuuu`
05h	PORTA	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	`---x xxxx`	`---u uuuu`
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	`xxxx xxxx`	`uuuu uuuu`
07h		Unimplemented location, read as '0'								`---- ----`	`---- ----`
08h	EEDATA	EEPROM data register								`xxxx xxxx`	`uuuu uuuu`
09h	EEADR	EEPROM address register								`xxxx xxxx`	`uuuu uuuu`
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC (1)					`---0 0000`	`---0 0000`
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	`0000 000x`	`0000 000u`
Bank 1											
80h	INDF	Uses contents of FSR to address data memory (not a physical register)								`---- ----`	`---- ----`
81h	OPTION_ REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	`1111 1111`	`1111 1111`
82h	PCL	Low order 8 bits of Program Counter (PC)								`0000 0000`	`0000 0000`
83h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	`0001 1xxx`	`000q quuu`
84h	FSR	Indirect data memory address pointer 0								`xxxx xxxx`	`uuuu uuuu`
85h	TRISA	—	—	—	PORTA data direction register					`---1 1111`	`---1 1111`
86h	TRISB	PORTB data direction register								`1111 1111`	`1111 1111`
87h		Unimplemented location, read as '0'								`---- ----`	`---- ----`
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	`---0 x000`	`---0 q000`
89h	EECON2	EEPROM control register 2 (not a physical register)								`---- ----`	`---- ----`
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC (1)					`---0 0000`	`---0 0000`
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	`0000 000x`	`0000 000u`

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

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> is never transferred to PCLATH.

2: The $\overline{TO}$ and $\overline{PD}$ status bits in the STATUS register are not affected by a $\overline{MCLR}$ reset.

3: Other (non power-up) resets include: external reset through $\overline{MCLR}$ and the Watchdog Timer Reset.

4.2.2.3 INTCON REGISTER

The INTCON register is a readable and writable register which contains the various enable bits for all interrupt sources.

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 4-1: INTCON REGISTER (ADDRESS 0Bh, 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	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit7							bit0

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

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

Note: For the operation of the interrupt structure, please refer to Section 8.5.

bit 6: **EEIE**: EE Write Complete Interrupt Enable bit
1 = Enables the EE write complete interrupt
0 = Disables the EE write complete interrupt

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

bit 4: **INTE**: RB0/INT Interrupt Enable bit
1 = Enables the RB0/INT interrupt
0 = Disables the RB0/INT 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 has overflowed (must be cleared in software)
0 = TMR0 did not overflow

bit 1: **INTF**: RB0/INT Interrupt Flag bit
1 = The RB0/INT interrupt occurred
0 = The RB0/INT interrupt did not occur

bit 0: **RBIF**: RB Port Change Interrupt Flag bit
1 = When at least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

4.5 Indirect Addressing; INDF and FSR Registers

The INDF register is not a physical register. Addressing INDF actually addresses the register whose address is contained in the FSR register (FSR is a *pointer*). This is indirect addressing.

EXAMPLE 4-1: INDIRECT ADDRESSING

- Register file 05 contains the value 10h
- Register file 06 contains the value 0Ah
- Load the value 05 into the FSR register
- A read of the INDF register will return the value of 10h
- Increment the value of the FSR register by one (FSR = 06)
- A read of the INDF register now will return the value of 0Ah.

Reading INDF itself indirectly (FSR = 0) will produce 00h. Writing to the INDF register indirectly results in a no-operation (although STATUS bits may be affected).

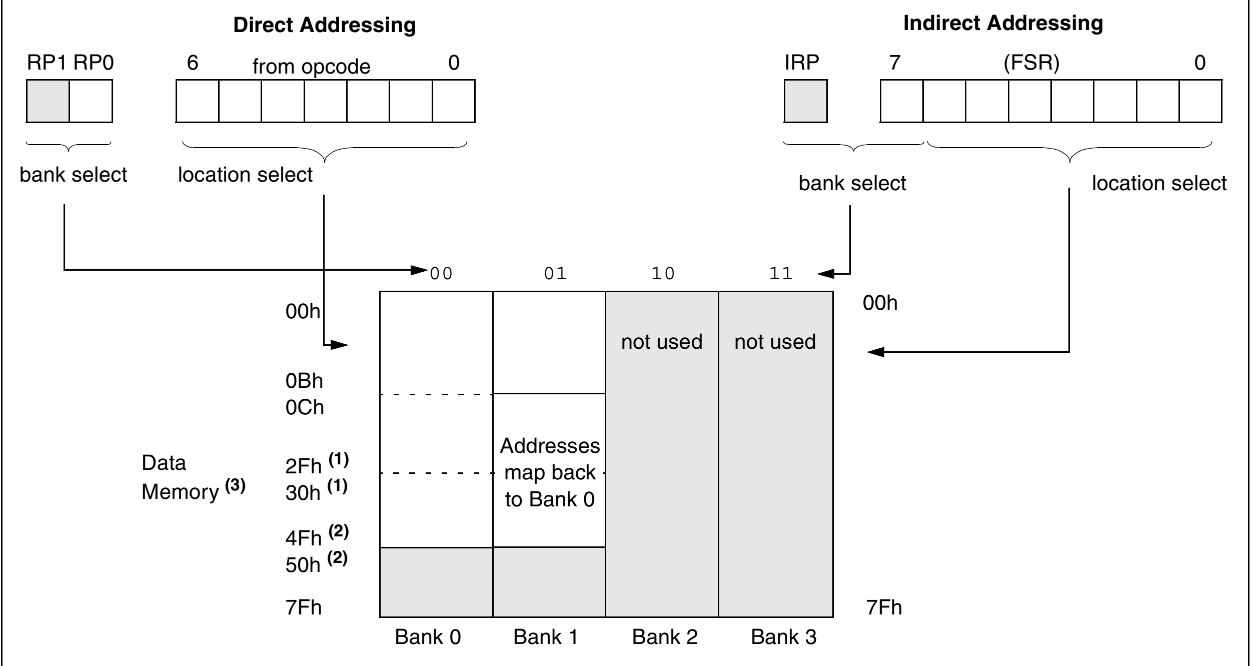
A simple program to clear RAM locations 20h-2Fh using indirect addressing is shown in Example 4-2.

EXAMPLE 4-2: HOW TO CLEAR RAM USING 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
CONTINUE
          :             ;YES, continue
```

An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (STATUS<7>), as shown in Figure 4-1. However, IRP is not used in the PIC16F8X.

FIGURE 4-1: DIRECT/INDIRECT ADDRESSING



Note 1: PIC16F83 and PIC16CR83 devices.
2: PIC16F84 and PIC16CR84 devices
3: For memory map detail see Figure 4-1.

EXAMPLE 5-1: INITIALIZING PORTB

```
CLRF   PORTB        ; Initialize PORTB by
                    ; setting output
                    ; data latches
BSF    STATUS, RP0  ; Select Bank 1
MOVLW  0xCF         ; Value used to
                    ; initialize data
                    ; direction
MOVWF  TRISB        ; Set RB<3:0> as inputs
                    ; RB<5:4> as outputs
                    ; RB<7:6> as inputs
```

TABLE 5-3 PORTB FUNCTIONS

Name	Bit	Buffer Type	I/O Consistency Function
RB0/INT	bit0	TTL/ST(1)	Input/output pin 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(2)	Input/output pin (with interrupt on change). Internal software programmable weak pull-up. Serial programming clock.
RB7	bit7	TTL/ST(2)	Input/output pin (with interrupt on change). Internal software programmable weak pull-up. Serial programming data.

Legend: TTL = TTL input, ST = Schmitt Trigger.
Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
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 Power-on Reset	Value on all other resets
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111
81h	OPTION_REG	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

FIGURE 6-3: TMR0 TIMING: INTERNAL CLOCK/PRESCALE 1:2

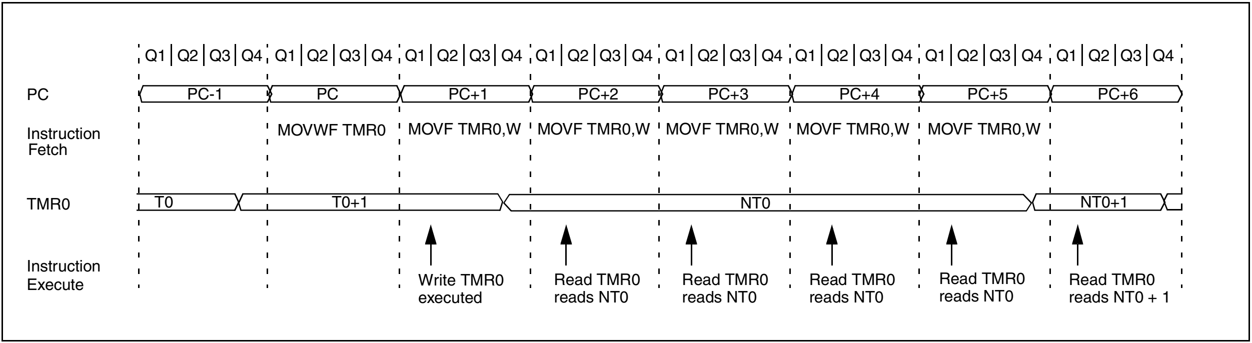
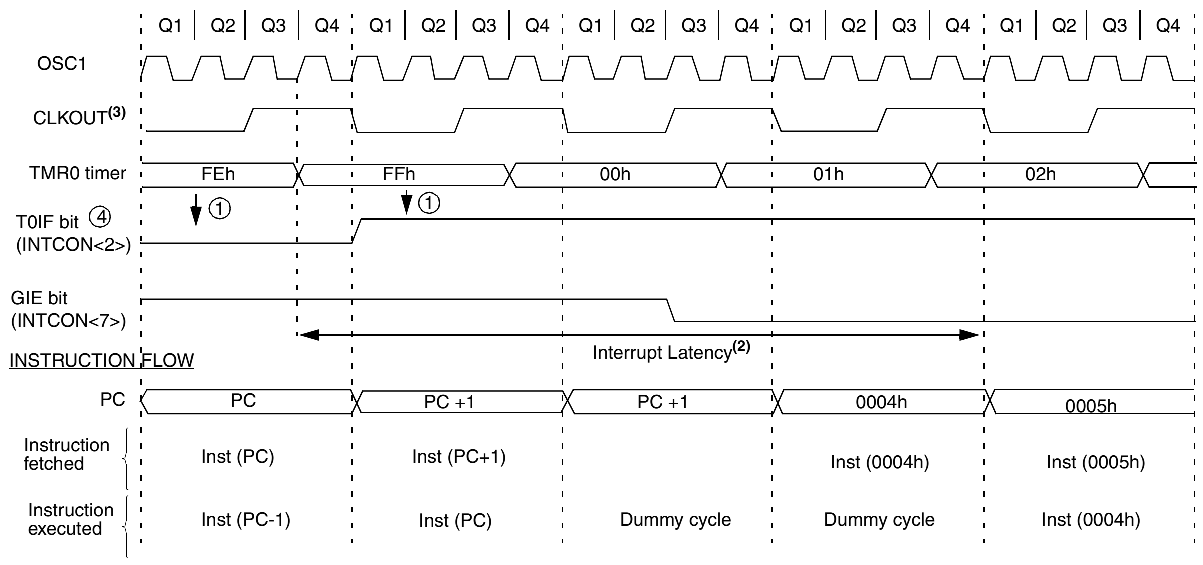


FIGURE 6-4: TMR0 INTERRUPT TIMING



Note 1: T0IF interrupt flag is sampled here (every Q1).
2: Interrupt latency = 3.25Tcy, where Tcy = instruction cycle time.
3: CLKOUT is available only in RC oscillator mode.
4: The timer clock (after the synchronizer circuit) which increments the timer from FFh to 00h immediately sets the T0IF bit. The TMR0 register will roll over 3 Tosc cycles later.

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).

> **Note:** To avoid an unintended device RESET, the following instruction sequence (Example 6-1) must be executed when changing the prescaler assignment from Timer0 to the WDT. This sequence must be taken even if the WDT is disabled. To change prescaler from the WDT to the Timer0 module use the sequence shown in Example 6-2.

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

```
BCF      STATUS, RP0  ;Bank 0
CLRF     TMR0         ;Clear TMR0
                      ; and Prescaler
BSF      STATUS, RP0  ;Bank 1
CLRWDT                ;Clears WDT
MOVLW    b'xxxx1xxx'  ;Select new
MOVWF    OPTION_REG       ; prescale value
BCF      STATUS, RP0  ;Bank 0
```

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

```
CLRWDT                ;Clear WDT and
                      ; prescaler
BSF      STATUS, RP0  ;Bank 1
MOVLW    b'xxxx0xxx'  ;Select TMR0, new
                      ; prescale value
                      ' and clock source
MOVWF    OPTION_REG       ;
BCF      STATUS, RP0  ;Bank 0
```

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 Power-on Reset	Value on all other resets
01h	TMR0	Timer0 module’s register								xxxx xxxx	uuuu uuuu
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 0000
81h	OPTION_REG	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
85h	TRISA	—	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111

Legend: x = unknown, u = unchanged. - = unimplemented read as '0'. Shaded cells are not associated with Timer0.

7.2 EECON1 and EECON2 Registers

EECON1 is the control register with five low order bits physically implemented. The upper-three bits are non-existent and read as '0's.

Control bits RD and WR initiate read and write, respectively. These bits cannot be cleared, only set, in software. They are cleared in hardware at completion of the read or write operation. The inability to clear the WR bit in software prevents the accidental, premature termination of a write operation.

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear. The WRERR bit is set when a write operation is interrupted by a $\overline{\text{MCLR}}$ reset or a WDT time-out reset during normal operation. In these situations, following reset, the user can check the WRERR bit and rewrite the location. The data and address will be unchanged in the EEDATA and EEADR registers.

Interrupt flag bit EEIF is set when write is complete. It must be cleared in software.

EECON2 is not a physical register. Reading EECON2 will read all '0's. The EECON2 register is used exclusively in the Data EEPROM write sequence.

7.3 Reading the EEPROM Data Memory

To read a data memory location, the user must write the address to the EEADR register and then set control bit RD (EECON1<0>). The data is available, in the very next cycle, in the EEDATA register; therefore it can be read in the next instruction. EEDATA will hold this value until another read or until it is written to by the user (during a write operation).

EXAMPLE 7-1: DATA EEPROM READ

```
BCF     STATUS, RP0  ; Bank 0
MOVLW   CONFIG_ADDR  ;
MOVWF   EEADR        ; Address to read
BSF     STATUS, RP0  ; Bank 1
BSF     EECON1, RD   ; EE Read
BCF     STATUS, RP0  ; Bank 0
MOVF    EEDATA, W    ; W = EEDATA
```

7.4 Writing to the EEPROM Data Memory

To write an EEPROM data location, the user must first write the address to the EEADR register and the data to the EEDATA register. Then the user must follow a specific sequence to initiate the write for each byte.

EXAMPLE 7-1: DATA EEPROM WRITE

```
          BSF     STATUS, RP0  ; Bank 1
          BCF     INTCON, GIE  ; Disable INTs.
          BSF     EECON1, WREN ; Enable Write
          MOVLW   55h          ;
Required  MOVWF   EECON2       ; Write 55h
Sequence  MOVLW   AAh          ;
          MOVWF   EECON2       ; Write AAh
          BSF     EECON1,WR    ; Set WR bit
                               ;   begin write
          BSF     INTCON, GIE  ; Enable INTs.
```

The write will not initiate if the above sequence is not exactly followed (write 55h to EECON2, write AAh to EECON2, then set WR bit) for each byte. We strongly recommend that interrupts be disabled during this code segment.

Additionally, the WREN bit in EECON1 must be set to enable write. This mechanism prevents accidental writes to data EEPROM due to errant (unexpected) code execution (i.e., lost programs). The user should keep the WREN bit clear at all times, except when updating EEPROM. The WREN bit is not cleared by hardware

After a write sequence has been initiated, clearing the WREN bit will not affect this write cycle. The WR bit will be inhibited from being set unless the WREN bit is set.

At the completion of the write cycle, the WR bit is cleared in hardware and the EE Write Complete Interrupt Flag bit (EEIF) is set. The user can either enable this interrupt or poll this bit. EEIF must be cleared by software.

FIGURE 8-2: CONFIGURATION WORD - PIC16F83 AND PIC16F84

R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u
CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	$\overline{\text{PWRTE}}$	WDTE	FOSC1	FOSC0
bit13													bit0

R = Readable bit
P = Programmable bit
- n = Value at POR reset
u = unchanged

bit 13:4 **CP**: Code Protection bit
1 = Code protection off
0 = All memory is code protected

bit 3 **$\overline{\text{PWRTE}}$**: Power-up Timer Enable bit
1 = Power-up timer is disabled
0 = Power-up timer is enabled

bit 2 **WDTE**: Watchdog Timer Enable bit
1 = WDT enabled
0 = WDT disabled

bit 1:0 **FOSC1:FOSC0**: Oscillator Selection bits
`11` = RC oscillator
`10` = HS oscillator
`01` = XT oscillator
`00` = LP oscillator

8.2 Oscillator Configurations

8.2.1 OSCILLATOR TYPES

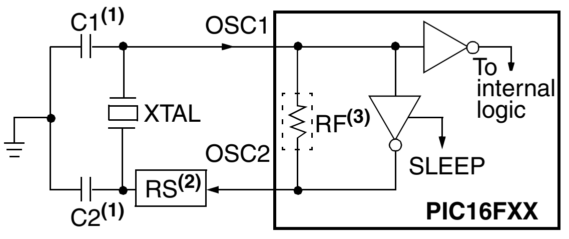
The PIC16F8X can be operated in four different oscillator modes. The user can program two configuration bits (FOSC1 and FOSC0) to select one of these four modes:

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

8.2.2 CRYSTAL OSCILLATOR / CERAMIC RESONATORS

In XT, LP or HS modes a crystal or ceramic resonator is connected to the OSC1/CLKIN and OSC2/CLKOUT pins to establish oscillation (Figure 8-3).

FIGURE 8-3: CRYSTAL/CERAMIC RESONATOR OPERATION (HS, XT OR LP OSC CONFIGURATION)



Note1: See Table 8-1 for recommended values of C1 and C2.
2: A series resistor (RS) may be required for AT strip cut crystals.
3: RF varies with the crystal chosen.

The PIC16F8X 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/CLKIN pin (Figure 8-4).

8.11 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 OSC1/CLKIN pin. That means that the WDT will run even if the clock on the OSC1/CLKIN and OSC2/CLKOUT 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 Wake-up causes the device to wake-up and continue with normal operation. The WDT can be permanently disabled by programming configuration bit WDTE as a '0' (Section 8.1).

8.11.1 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_REG register. Thus, time-out periods up to 2.3 seconds can be realized.

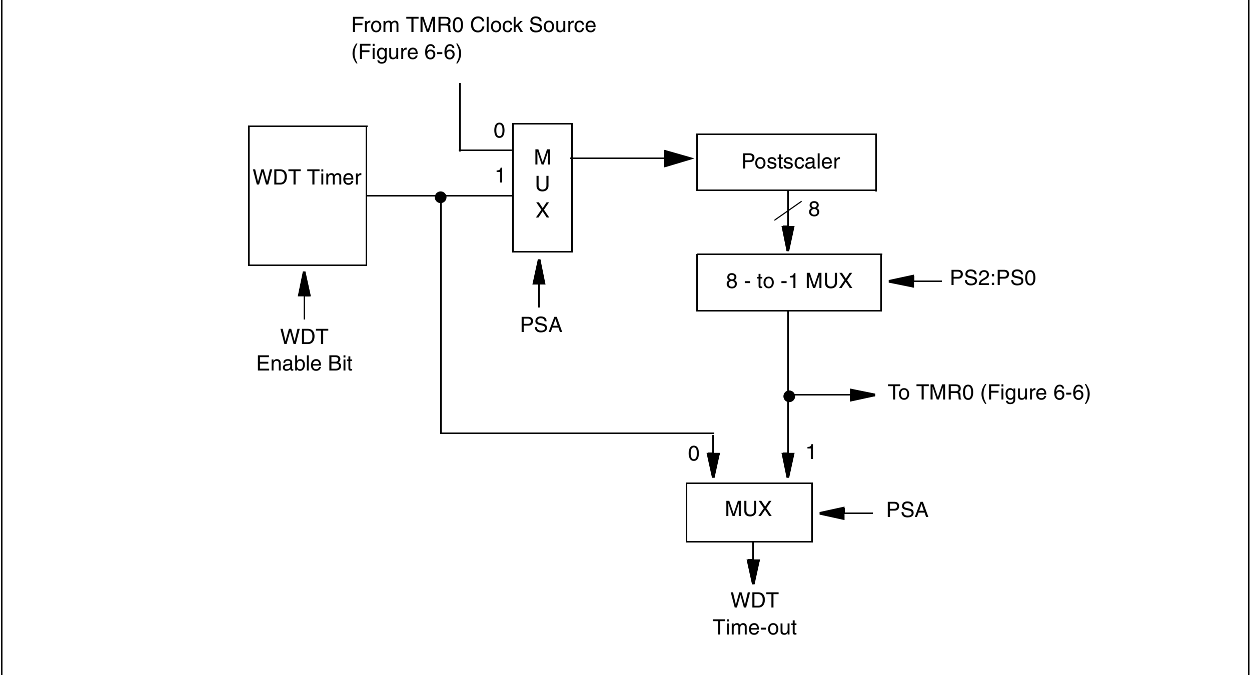
The `CLRWDT` and `SLEEP` instructions clear the WDT and the postscaler (if assigned to the WDT) and prevent it from timing out and generating a device RESET condition.

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

8.11.2 WDT PROGRAMMING CONSIDERATIONS

It should also be taken into 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 8-18: WATCHDOG TIMER BLOCK DIAGRAM



Note: PSA and PS2:PS0 are bits in the OPTION_REG register.

TABLE 8-7 SUMMARY OF REGISTERS ASSOCIATED WITH THE WATCHDOG TIMER

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets
2007h	Config. bits	(2)	(2)	(2)	(2)	PWRTE(1)	WDTE	FOSC1	FOSC0	(2)	
81h	OPTION_ REG	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown. Shaded cells are not used by the WDT.
Note 1: See Figure 8-1 and Figure 8-2 for operation of the PWRTE bit.
2: See Figure 8-1, Figure 8-2 and Section 8.13 for operation of the Code and Data protection bits.

BTFSS Bit Test f, Skip if Set

Syntax:	[*label*] BTFSS f,b
Operands:	$0 \le f \le 127$ $0 \le b < 7$
Operation:	skip if (f<b>) = 1
Status Affected:	None

Encoding:

`01`	`11bb`	`bfff`	`ffff`

Description: If bit 'b' in register 'f' is '0' then the next instruction is executed.
If bit 'b' is '1', then the next instruction is discarded and a NOP is executed instead, making this a 2TCY instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	No-Operation

If Skip: (2nd Cycle)

Q1	Q2	Q3	Q4
No-Operation	No-Operation	No-Operation	No-Operation

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:	$0 \le k \le 2047$
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

Q Cycle Activity:

	Q1	Q2	Q3	Q4
1st Cycle	Decode	Read literal 'k', Push PC to Stack	Process data	Write to PC
2nd Cycle	No-Operation	No-Operation	No-Operation	No-Operation

Example

```
HERE    CALL   THERE
```

Before Instruction
PC = Address `HERE`
After Instruction
PC = Address `THERE`
TOS = Address `HERE+1`

COMF — Complement f

Syntax:	[*label*] COMF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	$(\bar{f}) \rightarrow$ (destination)
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

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example

```
COMF        REG1,0
```

Before Instruction
REG1 = 0x13
After Instruction
REG1 = 0x13
W = 0xEC

DECF — Decrement f

Syntax:	[*label*] DECF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) - 1 $\rightarrow$ (destination)
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

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

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:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) - 1 $\rightarrow$ (destination); 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 1, the next instruction, is executed. If the result is 0, then a NOP is executed instead making it a 2TCY instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

If Skip: (2nd Cycle)

Q1	Q2	Q3	Q4
No-Operation	No-Operation	No-Operation	No-Operation

Example

```
HERE      DECFSZ   CNT, 1
          GOTO     LOOP
CONTINUE  •
          •
          •
```

Before Instruction
PC = address `HERE`
After Instruction
CNT = CNT - 1
if CNT = 0,
PC = address `CONTINUE`
if CNT ≠ 0,
PC = address `HERE+1`

NOP — No Operation

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

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No-Operation	No-Operation	No-Operation

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 PIC16CXX products, do not use this instruction.

RETFIE — Return from Interrupt

Syntax:	[*label*] RETFIE
Operands:	None
Operation:	TOS → PC, 1 → GIE
Status Affected:	None
Encoding:	`00 0000 0000 1001`
Description:	Return from Interrupt. Stack is POPed and Top of Stack (TOS) is loaded in the PC. Interrupts are enabled by setting Global Interrupt Enable bit, GIE (INTCON<7>). This is a two cycle instruction.
Words:	1
Cycles:	2

Q Cycle Activity:

	Q1	Q2	Q3	Q4
1st Cycle	Decode	No-Operation	Set the GIE bit	Pop from the Stack
2nd Cycle	No-Operation	No-Operation	No-Operation	No-Operation

Example `RETFIE`

After Interrupt
PC = TOS
GIE = 1

FIGURE 12-5: TYPICAL IPD vs. VDD, WATCHDOG DISABLED

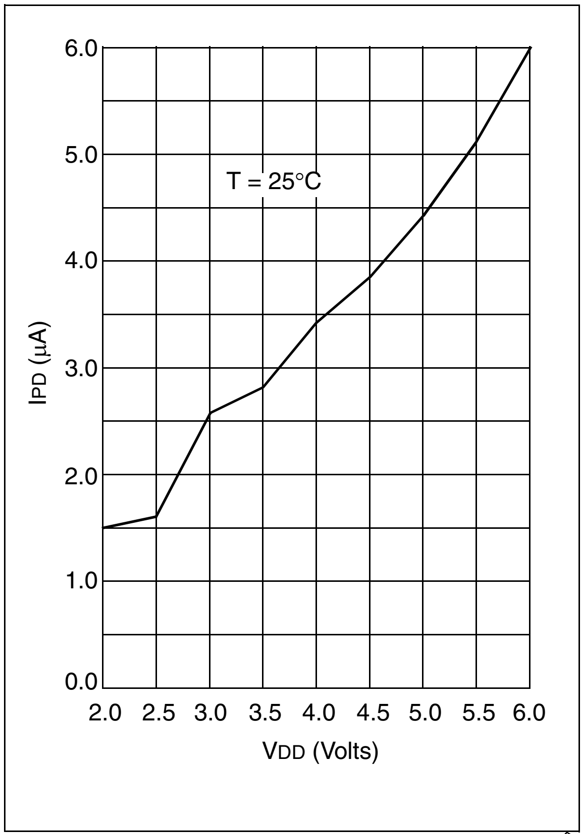


FIGURE 12-6: TYPICAL IPD vs. VDD, WATCHDOG ENABLED

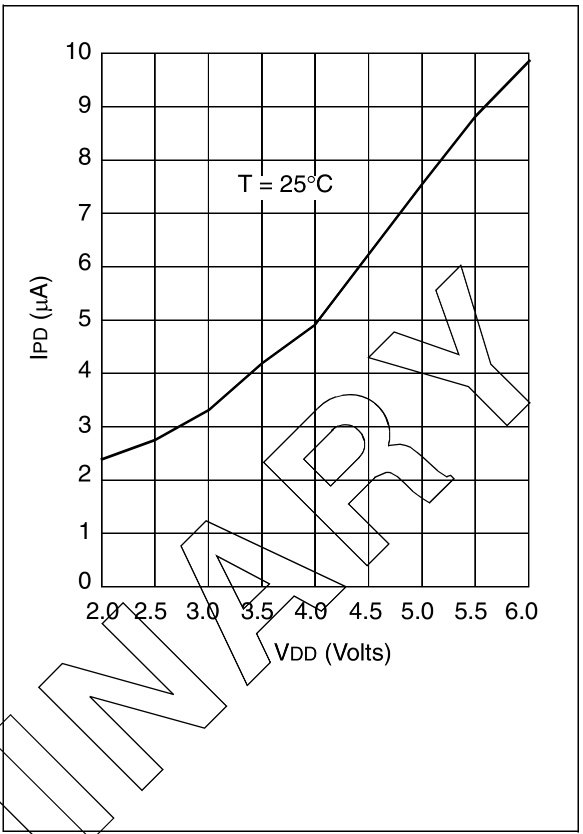
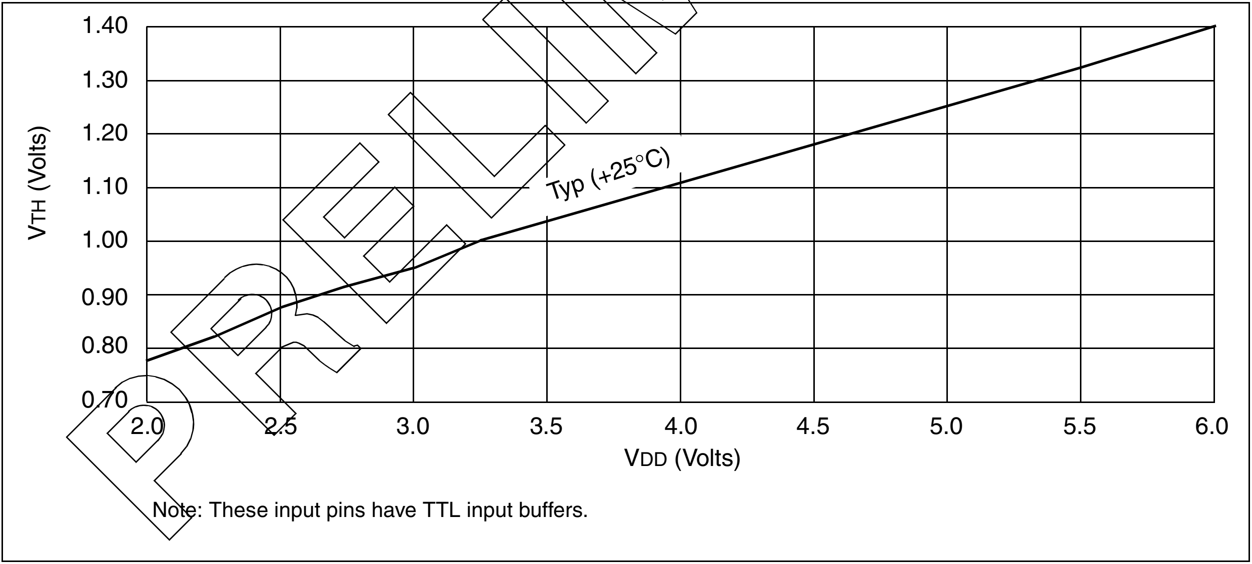


FIGURE 12-7: VTH (INPUT THRESHOLD VOLTAGE) OF I/O PINS vs. VDD



NOTES:

APPENDIX E: CONVERSION CONSIDERATIONS - PIC16C84 TO PIC16F83/F84 AND PIC16CR83/CR84

Considerations for converting from the PIC16C84 to the PIC16F84 are listed in the table below. These considerations apply to converting from the PIC16C84 to the PIC16F83 (same as PIC16F84 except for program and data RAM memory sizes) and the PIC16CR84 and PIC16CR83 (ROM versions of Flash devices). Development Systems support is available for all of the PIC16X8X devices.

Difference	PIC16C84	PIC16F84
The polarity of the PWRTE bit has been reversed. Ensure that the programmer has this bit correctly set before programming.	PWRTE	$\overline{\text{PWRTE}}$
The PIC16F84 (and PIC16CR84) have larger RAM sizes. Ensure that this does not cause an issue with your program.	RAM = 36 bytes	RAM = 68 bytes
The $\overline{\text{MCLR}}$ pin now has an on-chip filter. The input signal on the $\overline{\text{MCLR}}$ pin will require a longer low pulse to generate an interrupt.	$\overline{\text{MCLR}}$ pulse width (low) = 350ns; 2.0V ≤ VDD ≤ 3.0V = 150ns; 3.0V ≤ VDD ≤ 6.0V	$\overline{\text{MCLR}}$ pulse width (low) = 1000ns; 2.0V ≤ VDD ≤ 6.0V
Some electrical specifications have been improved (see IPD example). Compare the electrical specifications of the two devices to ensure that this will not cause a compatibility issue.	IPD (typ @ 2V) = 26μA IPD (max @ 4V, WDT disabled) =100μA (PIC16C84) =100μA (PIC16LC84)	IPD (typ @ 2V) < 1μA IPD (max @ 4V, WDT disabled) =14μA (PIC16F84) =7μA (PIC16LF84)
PORTA and crystal oscillator values less than 500kHz	For crystal oscillator configurations operating below 500kHz, the device may generate a spurious internal Q-clock when PORTA<0> switches state.	N/A
RB0/INT pin	TTL	TTL/ST* (* This buffer is a Schmitt Trigger input when configured as the external interrupt.)
EEADR<7:6> and IDD	It is recommended that the EEADR<7:6> bits be cleared. When either of these bits is set, the maximum IDD for the device is higher than when both are cleared.	N/A
Code Protect	1 CP bit	9 CP bits
Recommended value of REXT for RC oscillator circuits	REXT = 3kΩ - 100kΩ	REXT = 5kΩ - 100kΩ
GIE bit unintentional enable	If an interrupt occurs while the Global Interrupt Enable (GIE) bit is being cleared, the GIE bit may unintentionally be re-enabled by the user's Interrupt Service Routine (the `RETFIE` instruction).	N/A

NOTES:

ON-LINE SUPPORT

Microchip provides on-line support on the Microchip World Wide Web (WWW) site.

The web site is used by Microchip as a means to make files and information easily available to customers. To view the site, the user must have access to the Internet and a web browser, such as Netscape or Microsoft Explorer. Files are also available for FTP download from our FTP site.

Connecting to the Microchip Internet Web Site

The Microchip web site is available by using your favorite Internet browser to attach to:

www.microchip.com

The file transfer site is available by using an FTP service to connect to:

ftp://ftp.futureone.com/pub/microchip

The web site and file transfer site provide a variety of services. Users may download files for the latest Development Tools, Data Sheets, Application Notes, User's Guides, Articles and Sample Programs. A variety of Microchip specific business information is also available, including listings of Microchip sales offices, distributors and factory representatives. Other data available for consideration is:

- Latest Microchip Press Releases
- Technical Support Section with Frequently Asked Questions
- Design Tips
- Device Errata
- Job Postings
- Microchip Consultant Program Member Listing
- Links to other useful web sites related to Microchip Products
- Conferences for products, Development Systems, technical information and more
- Listing of seminars and events