

Welcome to [E-XFL.COM](https://www.e-xfl.com)

### What is "[Embedded - Microcontrollers](#)"?

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

### Applications of "[Embedded - Microcontrollers](#)"

#### Details

Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	20MHz
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 ~ 5.5V
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	<a href="https://www.e-xfl.com/product-detail/microchip-technology/pic16f84a-20i-so">https://www.e-xfl.com/product-detail/microchip-technology/pic16f84a-20i-so</a>

# PIC16F84A

## Table of Contents

1.0 Device Overview .....	3
2.0 Memory Organization .....	5
3.0 Data EEPROM Memory .....	13
4.0 I/O Ports .....	15
5.0 Timer0 Module .....	19
6.0 Special Features of the CPU .....	21
7.0 Instruction Set Summary .....	35
8.0 Development Support .....	43
9.0 Electrical Characteristics .....	47
10.0 DC/AC Characteristic Graphs .....	59
11.0 Packaging Information.....	69
Appendix A: Revision History .....	77
Appendix B: Conversion Considerations.....	78
Appendix C: Migration from Baseline to Mid-range Devices80	
INDEX .....	81
The Microchip Web Site .....	85
Customer Change Notification Service .....	85
Customer Support .....	85
Reader Response .....	86
PIC16F84A Product Identification System .....	87

## TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at [docerrors@microchip.com](mailto:docerrors@microchip.com) or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. We welcome your feedback.

### Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

<http://www.microchip.com>

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000A is version A of document DS30000).

### Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; <http://www.microchip.com>
- Your local Microchip sales office (see last page)

When contacting a sales office, please specify which device, revision of silicon and data sheet (include literature number) you are using.

### Customer Notification System

Register on our web site at [www.microchip.com](http://www.microchip.com) to receive the most current information on all of our products.

# PIC16F84A

**TABLE 1-1: PIC16F84A PINOUT DESCRIPTION**

Pin Name	PDIP No.	SOIC No.	SSOP No.	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	16	16	18	I	ST/CMOS <sup>(3)</sup>	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	15	19	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	4	4	4	I/P	ST	Master Clear (Reset) input/programming voltage input. This pin is an active low RESET to the device.
RA0 RA1 RA2 RA3 RA4/T0CKI	17 18 1 2 3	17 18 1 2 3	19 20 1 2 3	I/O I/O I/O I/O I/O	TTL TTL TTL TTL ST	PORTA is a bi-directional I/O port.  Can also be selected to be the clock input to the TMR0 timer/counter. Output is open drain type.
RB0/INT RB1 RB2 RB3 RB4 RB5 RB6 RB7	6 7 8 9 10 11 12 13	6 7 8 9 10 11 12 13	7 8 9 10 11 12 13 14	I/O I/O I/O I/O I/O I/O I/O	TTL/ST <sup>(1)</sup> TTL TTL TTL TTL TTL TTL/ST <sup>(2)</sup> TTL/ST <sup>(2)</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.
VSS	5	5	5,6	P	—	Ground reference for logic and I/O pins.
VDD	14	14	15,16	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.  
**Note 2:** This buffer is a Schmitt Trigger input when used in Serial Programming mode.  
**Note 3:** This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

## 2.0 MEMORY ORGANIZATION

There are two memory blocks in the PIC16F84A. These are the program memory and the data memory. Each block has its own bus, so that access to each block can occur during the same oscillator cycle.

The data memory can further be broken down into the general purpose RAM and the Special Function Registers (SFRs). The operation of the SFRs that control the “core” are described here. The SFRs used to control the peripheral modules are described in the section discussing each individual peripheral module.

The data memory area also contains the data EEPROM memory. This memory is not directly mapped into the data memory, but is indirectly mapped. That is, an indirect address pointer specifies the address of the data EEPROM memory to read/write. The 64 bytes of data EEPROM memory have the address range 0h-3Fh. More details on the EEPROM memory can be found in Section 3.0.

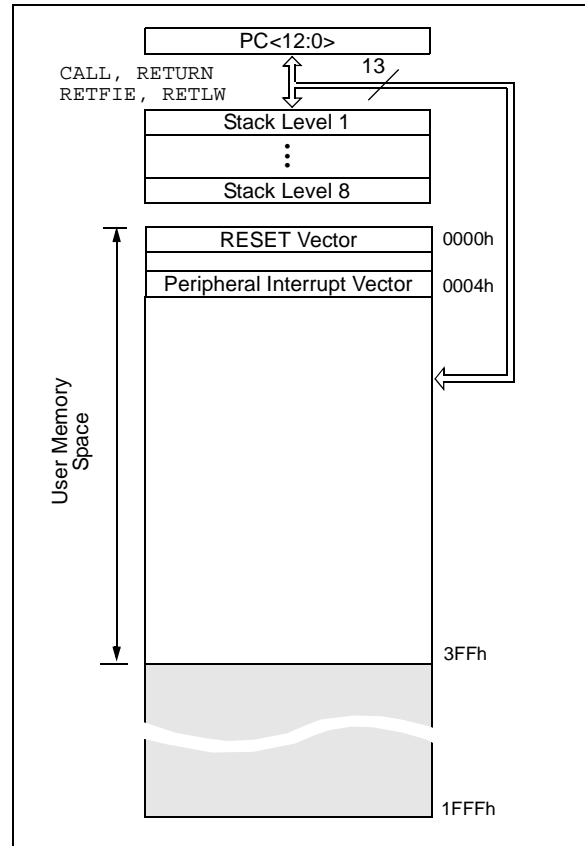
Additional information on device memory may be found in the PIC® Mid-Range Reference Manual, (DS33023).

### 2.1 Program Memory Organization

The PIC16FXX has a 13-bit program counter capable of addressing an 8K x 14 program memory space. For the PIC16F84A, the first 1K x 14 (0000h-03FFh) are physically implemented (Figure 2-1). Accessing a location above the physically implemented address will cause a wraparound. For example, for locations 20h, 420h, 820h, C20h, 1020h, 1420h, 1820h, and 1C20h, the instruction will be the same.

The RESET vector is at 0000h and the interrupt vector is at 0004h.

**FIGURE 2-1: PROGRAM MEMORY MAP AND STACK - PIC16F84A**



# PIC16F84A

## 2.3.1 STATUS REGISTER

The STATUS register contains the arithmetic status of the ALU, the RESET status and the bank select bit for data memory.

As with any register, the STATUS register can be the destination for any instruction. 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 device logic. Furthermore, the  $\overline{TO}$  and 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 `000u u1uu` (where u = unchanged).

Only the `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions should be used to alter the STATUS register (Table 7-2), because these instructions do not affect any status bit.

**Note 1:** The IRP and RP1 bits (STATUS<7:6>) are not used by the PIC16F84A and should be programmed as cleared. Use of these bits as general purpose R/W bits is NOT recommended, since this may affect upward compatibility with future products.

**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.

**3:** When 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. The specified bit(s) will be updated according to device logic

### REGISTER 2-1: STATUS REGISTER (ADDRESS 03h, 83h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	PD	Z	DC	C
bit 7					bit 0		

- bit 7-6     **Unimplemented:** Maintain as '0'
  - bit 5     **RP0:** Register Bank Select bits (used for direct addressing)  
01 = Bank 1 (80h - FFh)  
00 = Bank 0 (00h - 7Fh)
  - bit 4      **$\overline{TO}$ :** Time-out bit  
1 = After power-up, `CLRWDT` instruction, or `SLEEP` instruction  
0 = A WDT time-out occurred
  - bit 3     **PD:** Power-down bit  
1 = After power-up or by the `CLRWDT` instruction  
0 = By execution of the `SLEEP` instruction
  - bit 2     **Z:** Zero bit  
1 = The result of an arithmetic or logic operation is zero  
0 = The result of an arithmetic or logic operation is not zero
  - bit 1     **DC:** Digit carry/borrow bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions) (for borrow, the polarity is reversed)  
1 = A carry-out from the 4th low order bit of the result occurred  
0 = No carry-out from the 4th low order bit of the result
  - bit 0     **C:** Carry/borrow bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions) (for borrow, the polarity is reversed)  
1 = A carry-out from the Most Significant bit of the result occurred  
0 = No carry-out from the Most Significant bit of the result occurred
- Note:** A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high or low order bit of the source register.

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

## 2.4 PCL and PCLATH

The program counter (PC) specifies the address of the instruction to fetch for execution. The PC is 13 bits wide. The low byte is called the PCL register. This register is readable and writable. The high byte is called the PCH register. This register contains the PC<12:8> bits and is not directly readable or writable. If the program counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a *NOB*. All updates to the PCH register go through the PCLATH register.

### 2.4.1 STACK

The stack allows a combination of up to 8 program calls and interrupts to occur. The stack contains the return address from this branch in program execution.

Mid-range devices have an 8 level deep x 13-bit wide hardware stack. 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 modified when the stack is PUSHed or POPed.

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).

## 2.5 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 2-1: 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 2-2.

### EXAMPLE 2-2: 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 2-3. However, IRP is not used in the PIC16F84A.

# PIC16F84A

**TABLE 4-1: PORTA FUNCTIONS**

Name	Bit0	Buffer Type	Function
RA0	bit0	TTL	Input/output
RA1	bit1	TTL	Input/output
RA2	bit2	TTL	Input/output
RA3	bit3	TTL	Input/output
RA4/T0CKI	bit4	ST	Input/output or external clock input for TMR0. Output is open drain type.

Legend: TTL = TTL input, ST = Schmitt Trigger input

**TABLE 4-2: 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 Power-on Reset	Value on all other RESETS
05h	PORTA	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
85h	TRISA	—	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are unimplemented, read as '0'.

## 6.0 SPECIAL FEATURES OF THE CPU

What sets a microcontroller apart from other processors are special circuits to deal with the needs of real time applications. The PIC16F84A 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 features are:

- OSC Selection
- RESET
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
- Interrupts
- Watchdog Timer (WDT)
- SLEEP
- Code Protection
- ID Locations
- In-Circuit Serial Programming™ (ICSP™)

The PIC16F84A has a Watchdog Timer which can be shut-off only through 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. This design keeps the device in RESET while the power supply stabilizes. With these two timers on-chip, most applications need no external RESET circuitry.

SLEEP mode offers a very low current power-down mode. The user can wake-up from SLEEP through external RESET, Watchdog Timer Time-out or through an interrupt. Several oscillator options are provided 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 the various options.

Additional information on special features is available in the PIC® Mid-Range Reference Manual (DS33023).

### 6.1 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.

Address 2007h is beyond the user program memory space and it belongs to the special test/configuration memory space (2000h - 3FFFh). This space can only be accessed during programming.

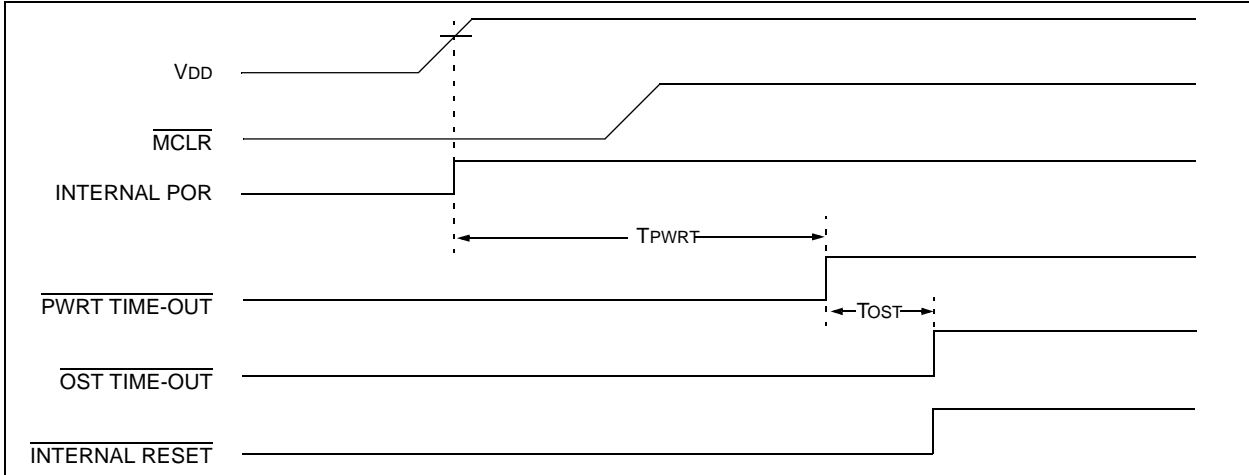
**REGISTER 6-1: PIC16F84A CONFIGURATION WORD**

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	R/P-u
CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	PWRT $\bar{E}$	WDTE	F0SC1	F0SC0	
bit13											bit0			

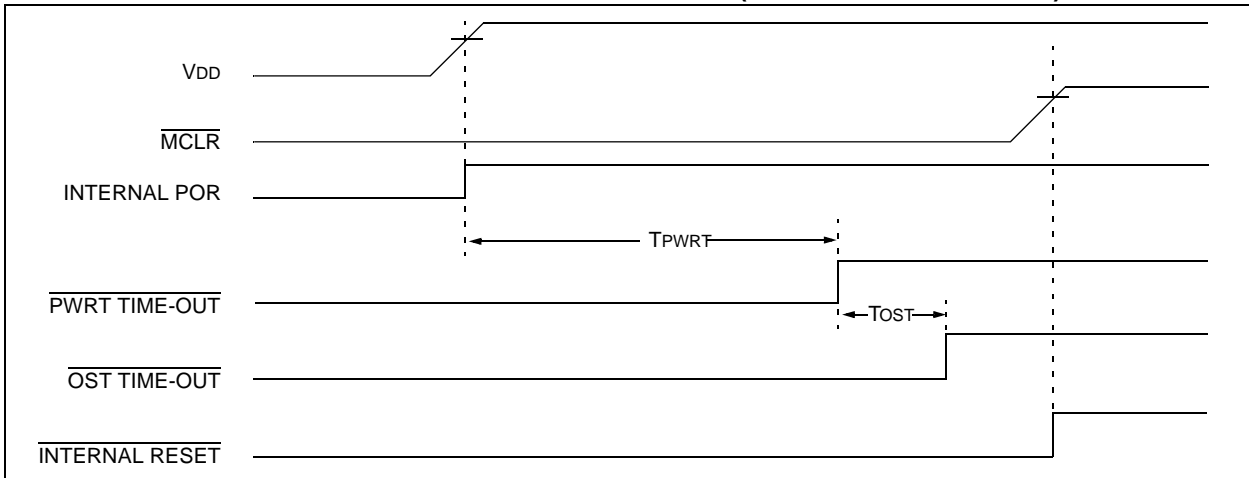
- bit 13-4      **CP:** Code Protection bit  
 1 = Code protection disabled  
 0 = All program memory is code protected
- bit 3        **PWRT $\bar{E}$ :** Power-up Timer Enable bit  
 1 = Power-up Timer is disabled  
 0 = Power-up Timer is enabled
- bit 2        **WDTE:** Watchdog Timer Enable bit  
 1 = WDT enabled  
 0 = WDT disabled
- bit 1-0      **F0SC1:F0SC0:** Oscillator Selection bits  
 11 = RC oscillator  
 10 = HS oscillator  
 01 = XT oscillator  
 00 = LP oscillator



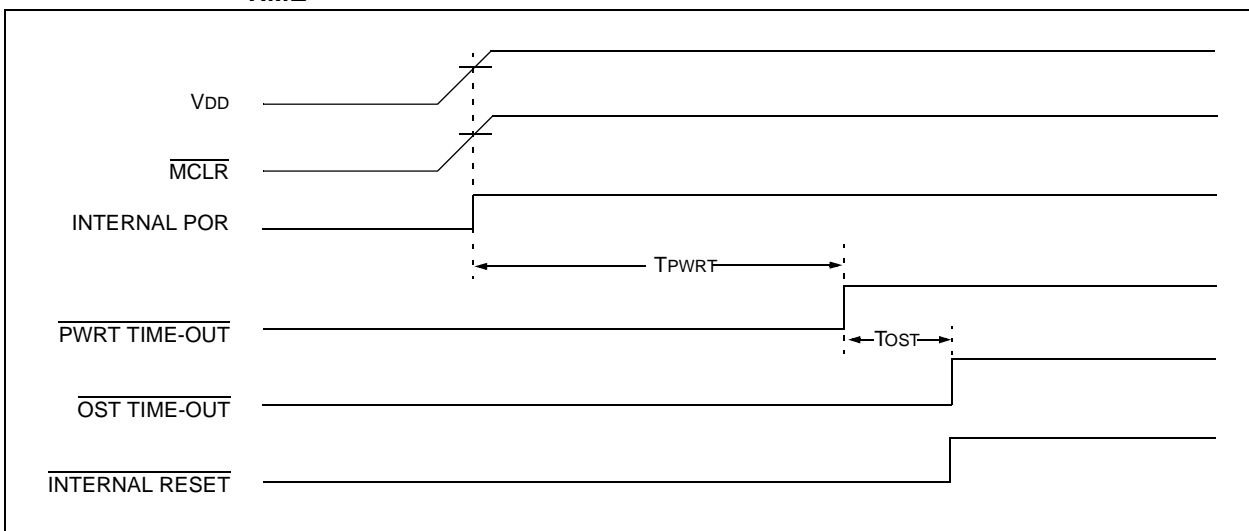
**FIGURE 6-6: TIME-OUT SEQUENCE ON POWER-UP ( $\overline{\text{MCLR}}$  NOT TIED TO  $V_{DD}$ ): CASE 1**



**FIGURE 6-7: TIME-OUT SEQUENCE ON POWER-UP ( $\overline{\text{MCLR}}$  NOT TIED TO  $V_{DD}$ ): CASE 2**



**FIGURE 6-8: TIME-OUT SEQUENCE ON POWER-UP ( $\overline{\text{MCLR}}$  TIED TO  $V_{DD}$ ): FAST  $V_{DD}$  RISE TIME**



## 7.1 Instruction Descriptions

### **ADDLW**      **Add Literal and W**

Syntax:            *[label]* ADDLW    *k*  
 Operands:         $0 \leq k \leq 255$   
 Operation:         $(W) + k \rightarrow (W)$   
 Status Affected:    C, DC, Z  
 Description:      The contents of the W register are added to the eight-bit literal 'k' and the result is placed in the W register.

### **ADDWF**      **Add W and f**

Syntax:            *[label]* ADDWF    *f,d*  
 Operands:         $0 \leq f \leq 127$   
                        $d \in [0,1]$   
 Operation:         $(W) + (f) \rightarrow (\text{destination})$   
 Status Affected:    C, DC, Z  
 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'.

### **ANDLW**      **AND Literal with W**

Syntax:            *[label]* ANDLW    *k*  
 Operands:         $0 \leq k \leq 255$   
 Operation:         $(W) .\text{AND.} (k) \rightarrow (W)$   
 Status Affected:    Z  
 Description:      The contents of W register are AND'ed with the eight-bit literal 'k'. The result is placed in the W register.

### **ANDWF**      **AND W with f**

Syntax:            *[label]* ANDWF    *f,d*  
 Operands:         $0 \leq f \leq 127$   
                        $d \in [0,1]$   
 Operation:         $(W) .\text{AND.} (f) \rightarrow (\text{destination})$   
 Status Affected:    Z  
 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'.

### **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  
 Description:      Bit 'b' in register 'f' is cleared.

### **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  
 Description:      Bit 'b' in register 'f' is set.

### **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  
 Description:      If bit 'b' in register 'f' is '0', 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.

---

## DECFSZ      Decrement f, Skip if 0

---

Syntax:            [ *label* ] DECFSZ f,d

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

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

Status Affected: None

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.

---

## INCFSZ      Increment f, Skip if 0

---

Syntax:            [ *label* ] INCFSZ f,d

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

Operation:        (f) + 1 → (destination),  
 skip if result = 0

Status Affected: None

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 1, the next instruction is executed. If the result is 0, a NOP is executed instead, making it a 2TCY instruction.

---

## 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

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.

---

## IORLW      Inclusive OR Literal with W

---

Syntax:            [ *label* ] IORLW k

Operands:         $0 \leq k \leq 255$

Operation:        (W) .OR. k → (W)

Status Affected: Z

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

---

## INCF        Increment f

---

Syntax:            [ *label* ] INCF f,d

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

Operation:        (f) + 1 → (destination)

Status Affected: Z

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'.

---

## IORWF      Inclusive OR W with f

---

Syntax:            [ *label* ] IORWF f,d

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

Operation:        (W) .OR. (f) → (destination)

Status Affected: Z

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'.

**RLF**                      **Rotate Left f through Carry**

---

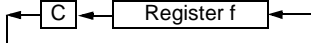
Syntax:                    [ *label* ] RLF f,d

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

Operation:                See description below

Status Affected:        C

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'.



**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

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

**RRF**                      **Rotate Right f through Carry**

---

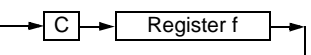
Syntax:                    [ *label* ] RRF f,d

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

Operation:                See description below

Status Affected:        C

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'.



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

---

Syntax:                    [ *label* ] SUBWF f,d

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

Operation:                 $(f) - (W) \rightarrow (\text{destination})$

Status Affected:        C, DC, Z

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'.

**SLEEP**

---

Syntax:                    [ *label* ] SLEEP

Operands:                None

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

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

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

**SWAPF**                   **Swap Nibbles in f**

---

Syntax:                    [ *label* ] SWAPF f,d

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

Operation:                 $(f<3:0>) \rightarrow (\text{destination}<7:4>)$ ,  
 $(f<7:4>) \rightarrow (\text{destination}<3:0>)$

Status Affected:        None

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'.

---

## 8.2 MPLAB C Compilers for Various Device Families

The MPLAB C Compiler code development systems are complete ANSI C compilers for Microchip's PIC18, PIC24 and PIC32 families of microcontrollers and the dsPIC30 and dsPIC33 families of digital signal controllers. These compilers provide powerful integration capabilities, superior code optimization and ease of use.

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

## 8.3 HI-TECH C for Various Device Families

The HI-TECH C Compiler code development systems are complete ANSI C compilers for Microchip's PIC family of microcontrollers and the dsPIC family of digital signal controllers. These compilers provide powerful integration capabilities, omniscient code generation and ease of use.

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

The compilers include a macro assembler, linker, pre-processor, and one-step driver, and can run on multiple platforms.

## 8.4 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multi-purpose source files
- Directives that allow complete control over the assembly process

## 8.5 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler and the MPLAB C18 C Compiler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled 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

## 8.6 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC devices. MPLAB C Compiler uses the assembler to produce its 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 device instruction set
- Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- MPLAB IDE compatibility

---

## 8.7 MPLAB SIM Software Simulator

The MPLAB SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC<sup>®</sup> DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB SIM Software Simulator fully supports symbolic debugging using the MPLAB C Compilers, and the MPASM and MPLAB Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

## 8.8 MPLAB REAL ICE In-Circuit Emulator System

MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs PIC<sup>®</sup> Flash MCUs and dsPIC<sup>®</sup> Flash DSCs with the easy-to-use, powerful graphical user interface of the MPLAB Integrated Development Environment (IDE), included with each kit.

The emulator is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with in-circuit debugger systems (RJ11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

The emulator is field upgradable through future firmware downloads in MPLAB IDE. In upcoming releases of MPLAB IDE, new devices will be supported, and new features will be added. MPLAB REAL ICE offers significant advantages over competitive emulators including low-cost, full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, a ruggedized probe interface and long (up to three meters) interconnection cables.

## 8.9 MPLAB ICD 3 In-Circuit Debugger System

MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost effective high-speed hardware debugger/programmer for Microchip Flash Digital Signal Controller (DSC) and microcontroller (MCU) devices. It debugs and programs PIC<sup>®</sup> Flash microcontrollers and dsPIC<sup>®</sup> DSCs with the powerful, yet easy-to-use graphical user interface of MPLAB Integrated Development Environment (IDE).

The MPLAB ICD 3 In-Circuit Debugger probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with a connector compatible with the MPLAB ICD 2 or MPLAB REAL ICE systems (RJ-11). MPLAB ICD 3 supports all MPLAB ICD 2 headers.

## 8.10 PICkit 3 In-Circuit Debugger/Programmer and PICkit 3 Debug Express

The MPLAB PICkit 3 allows debugging and programming of PIC<sup>®</sup> and dsPIC<sup>®</sup> Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB Integrated Development Environment (IDE). The MPLAB PICkit 3 is connected to the design engineer's PC using a full speed USB interface and can be connected to the target via a Microchip debug (RJ-11) connector (compatible with MPLAB ICD 3 and MPLAB REAL ICE). The connector uses two device I/O pins and the reset line to implement in-circuit debugging and In-Circuit Serial Programming™.

The PICkit 3 Debug Express include the PICkit 3, demo board and microcontroller, hookup cables and CDROM with user's guide, lessons, tutorial, compiler and MPLAB IDE software.

## 9.0 ELECTRICAL CHARACTERISTICS

### Absolute Maximum Ratings †

Ambient temperature under bias .....	-55°C to +125°C
Storage temperature .....	-65°C to +150°C
Voltage on any pin with respect to V <sub>SS</sub> (except V <sub>DD</sub> , $\overline{\text{MCLR}}$ , and RA4) .....	-0.3V to (V <sub>DD</sub> + 0.3V)
Voltage on V <sub>DD</sub> with respect to V <sub>SS</sub> .....	-0.3 to +7.5V
Voltage on $\overline{\text{MCLR}}$ with respect to V <sub>SS</sub> <sup>(1)</sup> .....	-0.3 to +14V
Voltage on RA4 with respect to V <sub>SS</sub> .....	-0.3 to +8.5V
Total power dissipation <sup>(2)</sup> .....	800 mW
Maximum current out of V <sub>SS</sub> pin .....	150 mA
Maximum current into V <sub>DD</sub> pin .....	100 mA
Input clamp current, I <sub>IK</sub> (V <sub>I</sub> < 0 or V <sub>I</sub> > V <sub>DD</sub> ) .....	± 20 mA
Output clamp current, I <sub>OK</sub> (V <sub>O</sub> < 0 or V <sub>O</sub> > V <sub>DD</sub> ) .....	± 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 .....	80 mA
Maximum current sourced by PORTA .....	50 mA
Maximum current sunk by PORTB .....	150 mA
Maximum current sourced by PORTB .....	100 mA

**Note 1:** Voltage spikes below V<sub>SS</sub> at the  $\overline{\text{MCLR}}$  pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a “low” level to the  $\overline{\text{MCLR}}$  pin rather than pulling this pin directly to V<sub>SS</sub>.

**2:** Power dissipation is calculated as follows:  $P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OL} \times I_{OL})$ .

† 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.

# PIC16F84A

## 9.2 DC Characteristics: PIC16F84A-04 (Commercial, Industrial) PIC16F84A-20 (Commercial, Industrial) PIC16LF84A-04 (Commercial, Industrial) (Continued)

DC Characteristics All Pins Except Power Supply Pins			Standard Operating Conditions (unless otherwise stated) Operating temperature $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ (commercial) $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ (industrial) Operating voltage $V_{DD}$ range as described in DC specifications (Section 9.1)				
Param No.	Symbol	Characteristic	Min	Typ†	Max	Units	Conditions
D080	VOL	<b>Output Low Voltage</b> I/O ports	—	—	0.6	V	$I_{OL} = 8.5\text{ mA}$ , $V_{DD} = 4.5\text{V}$
D083		OSC2/CLKOUT	—	—	0.6	V	$I_{OL} = 1.6\text{ mA}$ , $V_{DD} = 4.5\text{V}$ , (RC mode only)
D090	VOH	<b>Output High Voltage</b> I/O ports (Note 3)	$V_{DD}-0.7$	—	—	V	$I_{OH} = -3.0\text{ mA}$ , $V_{DD} = 4.5\text{V}$
D092		OSC2/CLKOUT (Note 3)	$V_{DD}-0.7$	—	—	V	$I_{OH} = -1.3\text{ mA}$ , $V_{DD} = 4.5\text{V}$ (RC mode only)
D150	VOD	<b>Open Drain High Voltage</b> RA4 pin	—	—	8.5	V	
D100	Cosc2	<b>Capacitive Loading Specs on Output Pins</b> OSC2 pin	—	—	15	pF	In XT, HS and LP modes when external clock is used to drive OSC1
D101	Cio	All I/O pins and OSC2 (RC mode)	—	—	50	pF	
D120	ED	<b>Data EEPROM Memory</b> Endurance	1M	10M	—	E/W	$25^{\circ}\text{C}$ at 5V
D121	VDRW	$V_{DD}$ for read/write	$V_{MIN}$	—	5.5	V	$V_{MIN}$ = Minimum operating voltage
D122	TDEW	Erase/Write cycle time	—	4	8	ms	
D130	EP	<b>Program FLASH Memory</b> Endurance	1000	10K	—	E/W	
D131	VPR	$V_{DD}$ for read	$V_{MIN}$	—	5.5	V	$V_{MIN}$ = Minimum operating voltage
D132	VPEW	$V_{DD}$ for erase/write	4.5	—	5.5	V	
D133	TPEW	Erase/Write cycle time	—	4	8	ms	

† Data in "Typ" column is at 5.0V,  $25^{\circ}\text{C}$  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. Do not drive the PIC16F84A with an external clock while the device is in RC mode, or chip damage may result.
- 2:** The leakage current on the MCLR pin is strongly dependent on the 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.
- 4:** The user may choose the better of the two specs.



## 9.3 AC (Timing) Characteristics

### 9.3.1 TIMING PARAMETER SYMBOLOGY

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

1. TppS2ppS
2. TppS

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

Lowercase letters (pp) and their meanings:

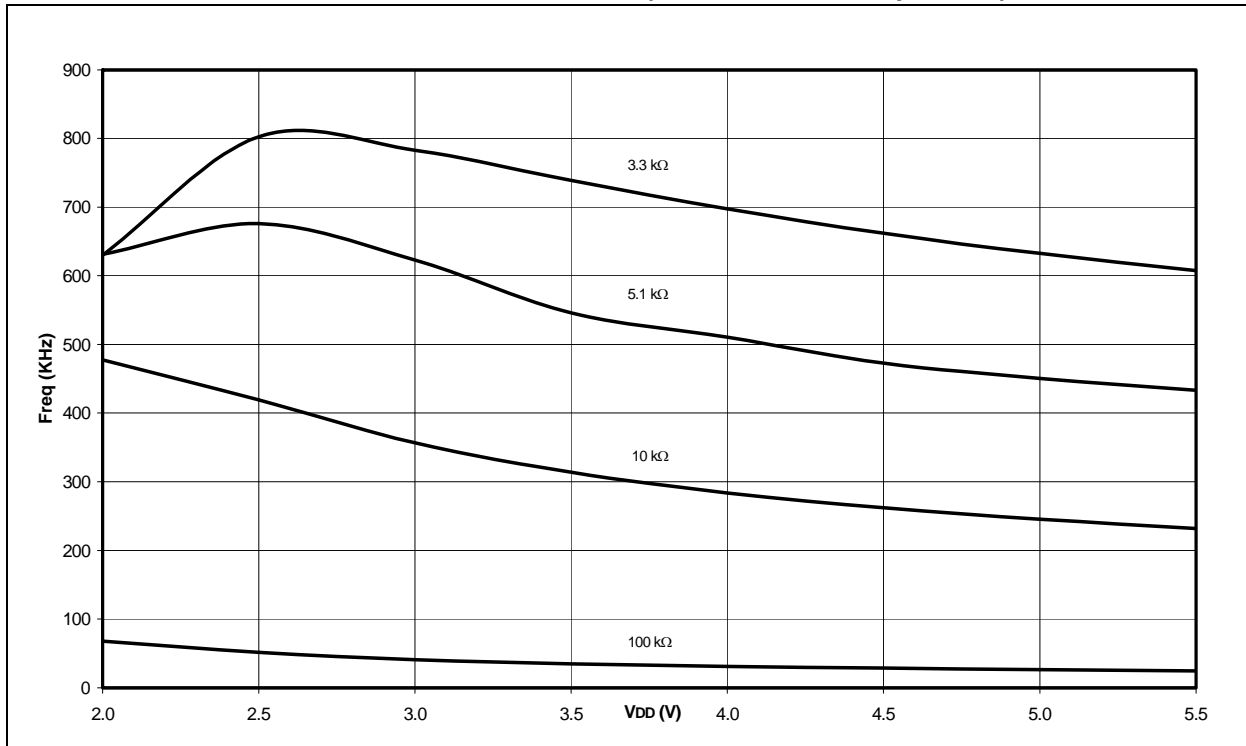
<b>pp</b> 2      to ck      CLKOUT cy      cycle time io      I/O port inp      INT pin mp      MCLR	os, osc      OSC1 ost      oscillator start-up timer pwrt      power-up timer rbt      RBx pins t0      T0CKI wdt      watchdog timer
--	--

Uppercase letters and their meanings:

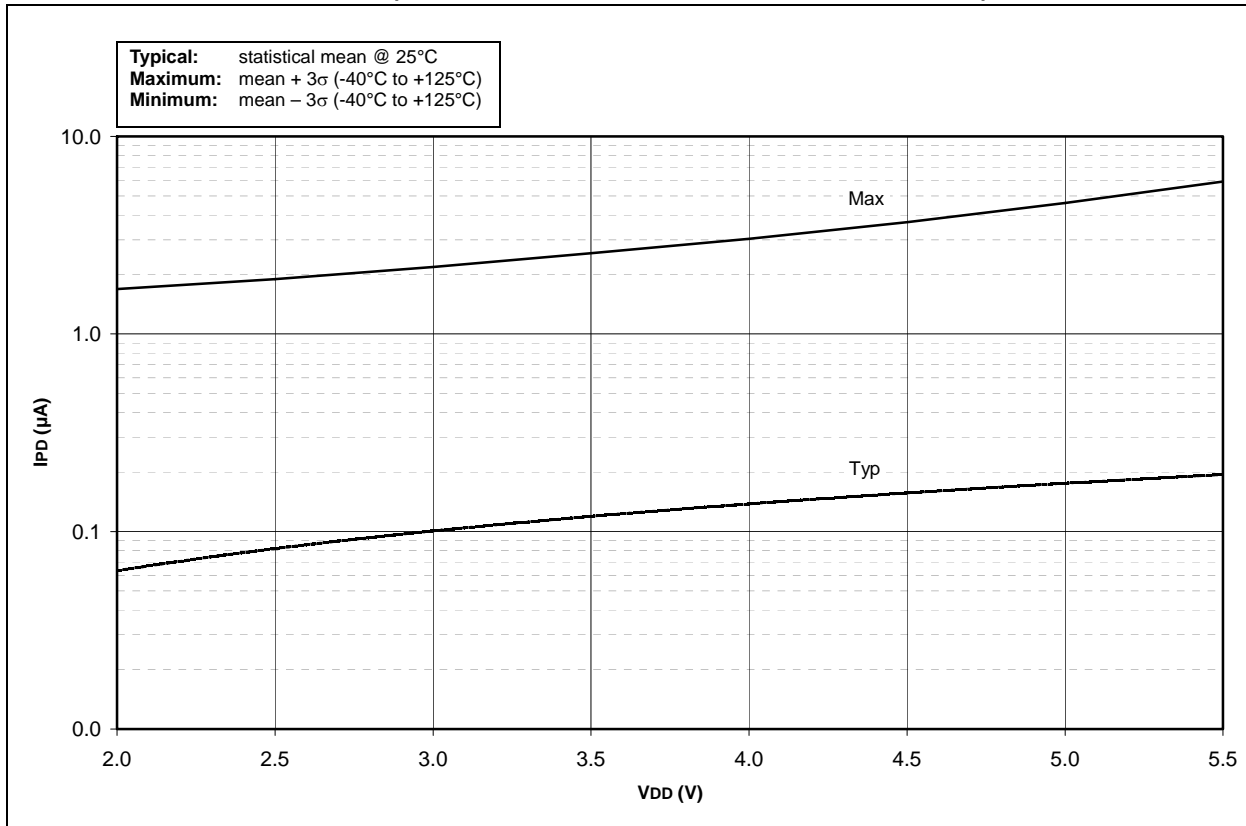
<b>S</b> <b>F</b> Fall <b>H</b> High <b>I</b> Invalid (high impedance) <b>L</b> Low	<b>P</b> Period <b>R</b> Rise <b>V</b> Valid <b>Z</b> High Impedance
---	---

# PIC16F84A

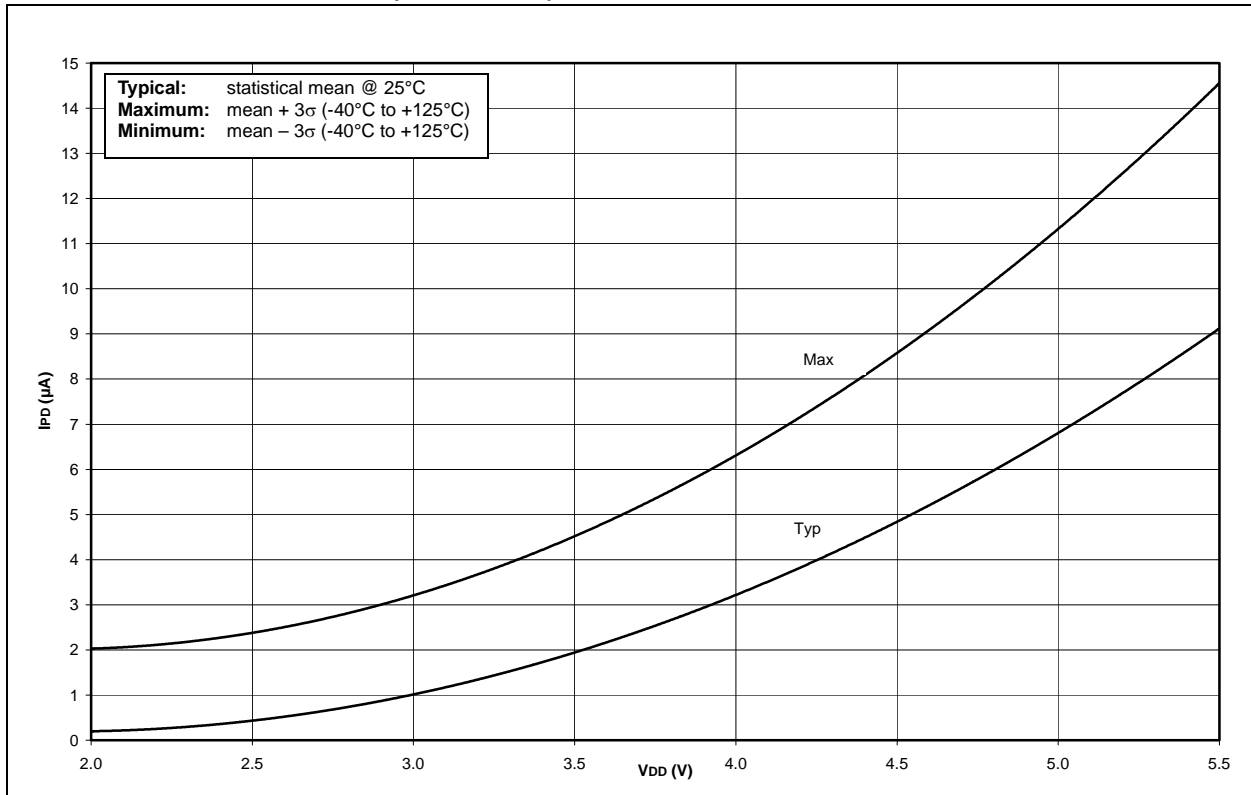
**FIGURE 10-9: AVERAGE Fosc vs. VDD FOR R (RC MODE, C = 300 pF, 25°C)**



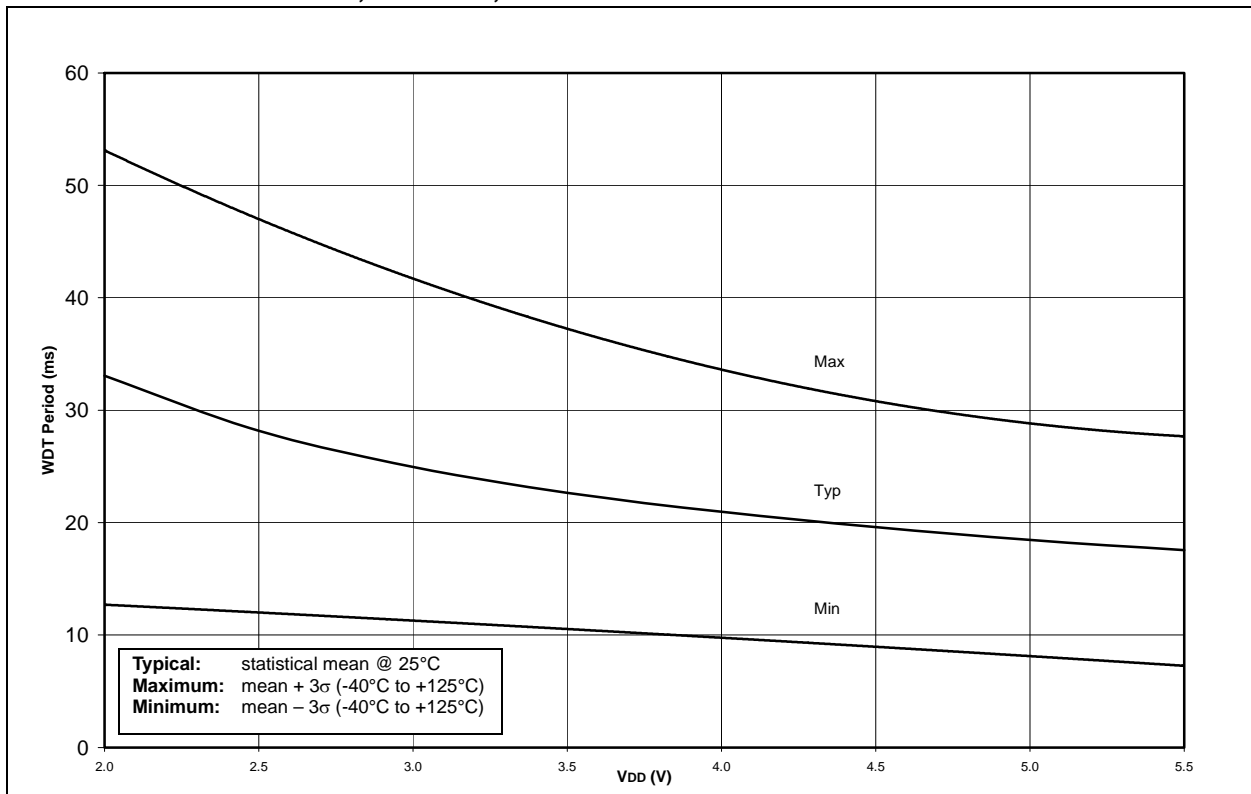
**FIGURE 10-10: IPD vs. VDD (SLEEP MODE, ALL PERIPHERALS DISABLED)**



**FIGURE 10-11: IPD vs. VDD (WDT MODE)**



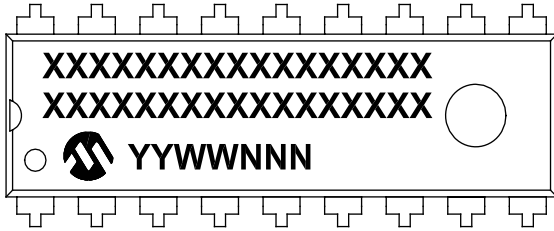
**FIGURE 10-12: TYPICAL, MINIMUM, AND MAXIMUM WDT PERIOD vs. VDD OVER TEMP**



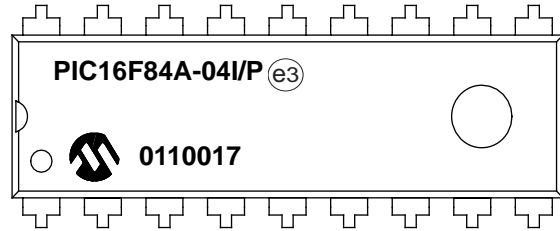
## 11.0 PACKAGING INFORMATION

### 11.1 Package Marking Information

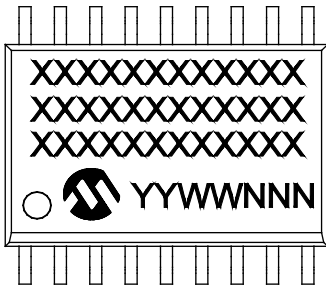
18-Lead PDIP (300 mil)



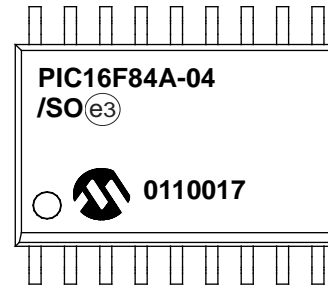
Example



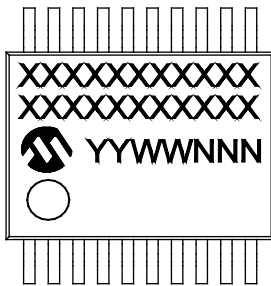
18-Lead SOIC (7.50 mm)



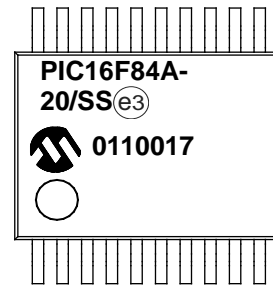
Example



20-Lead SSOP (5.30 mm)



Example



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

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

## INDEX

### A

Absolute Maximum Ratings .....	47
AC (Timing) Characteristics .....	53
Architecture, Block Diagram .....	3
Assembler	
MPASM Assembler .....	44

### B

Banking, Data Memory .....	6
Block Diagrams	
Crystal/Ceramic Resonator Operation .....	22
External Clock Input Operation .....	22
External Power-on Reset Circuit .....	26
Interrupt Logic .....	29
On-chip Reset .....	24
PIC16F84A .....	3
PORTA	
RA3:RA0 Pins .....	15
RA4 Pins .....	15
PORTB	
RB3:RB0 Pins .....	17
RB7:RB4 Pins .....	17
RC Oscillator Mode .....	23
Timer0 .....	19
Timer0/WDT Prescaler .....	20
Watchdog Timer (WDT) .....	31

### C

C (Carry) bit .....	8
C Compilers	
MPLAB C18 .....	44
CLKIN Pin .....	4
CLKOUT Pin .....	4
Code Examples	
Clearing RAM Using Indirect Addressing .....	11
Data EEPROM Write Verify .....	14
Indirect Addressing .....	11
Initializing PORTA .....	15
Initializing PORTB .....	17
Reading Data EEPROM .....	14
Saving STATUS and W Registers in RAM .....	30
Writing to Data EEPROM .....	14
Code Protection .....	21, 33
Configuration Bits .....	21
Configuration Word .....	21
Conversion Considerations .....	78
Customer Change Notification Service .....	85
Customer Notification Service .....	85
Customer Support .....	85

### D

Data EEPROM Memory .....	13
Associated Registers .....	14
EEADR Register .....	7, 13, 25
EECON1 Register .....	7, 13, 25
EECON2 Register .....	7, 13, 25
EEDATA Register .....	7, 13, 25
Write Complete Enable (EEIE Bit) .....	29
Write Complete Flag (EEIF Bit) .....	29
Data EEPROM Write Complete .....	29
Data Memory .....	6
Bank Select (RP0 Bit) .....	6
Banking .....	6
DC bit .....	8

DC Characteristics .....	49, 51
Development Support .....	43
Device Overview .....	3

### E

EECON1 Register	
EEIF Bit .....	29
Electrical Characteristics .....	47
Load Conditions .....	54
Parameter Measurement Information .....	54
PIC16F84A-04 Voltage-Frequency Graph .....	48
PIC16F84A-20 Voltage-Frequency Graph .....	48
PIC16LF84A-04 Voltage-Frequency Graph .....	48
Temperature and Voltage Specifications - AC .....	54
Endurance .....	1
Errata .....	2
External Clock Input (RA4/T0CKI). See Timer0	
External Interrupt Input (RB0/INT). See Interrupt Sources	
External Power-on Reset Circuit .....	26

### F

Firmware Instructions .....	35
-----------------------------	----

### I

I/O Ports .....	15
ID Locations .....	21, 33
In-Circuit Serial Programming (ICSP) .....	21, 33
INDF Register .....	7
Indirect Addressing .....	11
FSR Register .....	6, 7, 11, 25
INDF Register .....	7, 11, 25
Instruction Format .....	35
Instruction Set .....	35
ADDLW .....	37
ADDWF .....	37
ANDLW .....	37
ANDWF .....	37
BCF .....	37
BSF .....	37
BTFSC .....	38
BTFSS .....	37
CALL .....	38
CLRF .....	38
CLRWF .....	38
CLRWDI .....	38
COMF .....	38
DECF .....	38
DECFSZ .....	39
GOTO .....	39
INCF .....	39
INCFSZ .....	39
IORLW .....	39
IORWF .....	39
MOVF .....	40
MOVLW .....	40
MOVWF .....	40
NOP .....	40
RETFIE .....	40
RETLW .....	40
RETURN .....	40
RLF .....	41
RRF .....	41
SLEEP .....	41
SUBLW .....	41