**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

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

## Applications of "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Obsolete |
| Core Processor | PIC |
| Core Size | 8-Bit |
| Speed | 4MHz |
| Connectivity | - |
| Peripherals | Brown-out Detect/Reset, PWM, WDT |
| Number of I/O | 13 |
| Program Memory Size | 1.75KB (1K x 14) |
| Program Memory Type | OTP |
| EEPROM Size | - |
| RAM Size | 68 x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.5V ~ 6V |
| Data Converters | A/D 4x8b |
| Oscillator Type | External |
| Operating Temperature | -40°C ~ 125°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 20-SSOP (0.209", 5.30mm Width) |
| Supplier Device Package | 20-SSOP |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic16lc711t-04e-ss |

## 3.0 ARCHITECTURAL OVERVIEW

The high performance of the PIC16CXX family can be attributed to a number of architectural features commonly found in RISC microprocessors. To begin with, the PIC16CXX uses a Harvard architecture, in which, program and data are accessed from separate memories using separate buses. This improves bandwidth over traditional von Neumann architecture in which program and data are fetched from the same memory using the same bus. Separating program and data buses further allows instructions to be sized differently than the 8-bit wide data word. Instruction opcodes are 14-bits wide making it possible to have all single word instructions. A 14-bit wide program memory access bus fetches a 14-bit instruction in a single cycle. A two-stage pipeline overlaps fetch and execution of instructions (Example 3-1). Consequently, all instructions (35) execute in a single cycle (200 ns @ 20 MHz) except for program branches.

The table below lists program memory (EPROM) and data memory (RAM) for each PIC16C71X device.

| Device | Program Memory | Data Memory |
|---|---|---|
| PIC16C710 | 512 x 14 | 36 x 8 |
| PIC16C71 | 1K x 14 | 36 x 8 |
| PIC16C711 | 1K x 14 | 68 x 8 |
| PIC16C715 | 2K x 14 | 128 x 8 |

The PIC16CXX can directly or indirectly address its register files or data memory. All special function registers, including the program counter, are mapped in the data memory. The PIC16CXX has an orthogonal (symmetrical) instruction set that makes it possible to carry out any operation on any register using any addressing mode. This symmetrical nature and lack of ‘special optimal situations’ make programming with the PIC16CXX simple yet efficient. In addition, the learning curve is reduced significantly.

PIC16CXX devices contain an 8-bit ALU and working register. The ALU is a general purpose arithmetic unit. It performs arithmetic and Boolean functions between the data in the working register and any register file.

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

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

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

## 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 rewritten 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 to an output, 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 5-3 shows the effect of two sequential read-modify-write instructions on an I/O port.

**EXAMPLE 5-3: READ-MODIFY-WRITE INSTRUCTIONS ON AN I/O PORT**

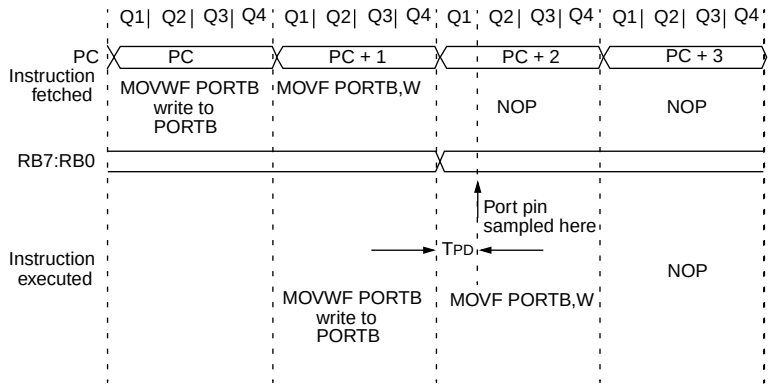
```
;Initial PORT settings: PORTB<7:4> Inputs
;                       PORTB<3:0> Outputs
;PORTB<7:6> have external pull-ups and are
;not connected to other circuitry
;
;                    PORT latch  PORT pins
;                    ----------  ---------
  BCF PORTB, 7      ; 01pp pppp    11pp pppp
  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 ppp. The 2nd BCF
;caused RB7 to be latched as the pin value
;(high).
```

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.

### 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 5-6). 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 5-6: SUCCESSIVE I/O OPERATION**




Note:

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

Note that:

data setup time = ($0.25T_{CY} - T_{PD}$)

where $T_{CY}$ = instruction cycle
$T_{PD}$ = propagation delay

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

### 6.3 Prescaler

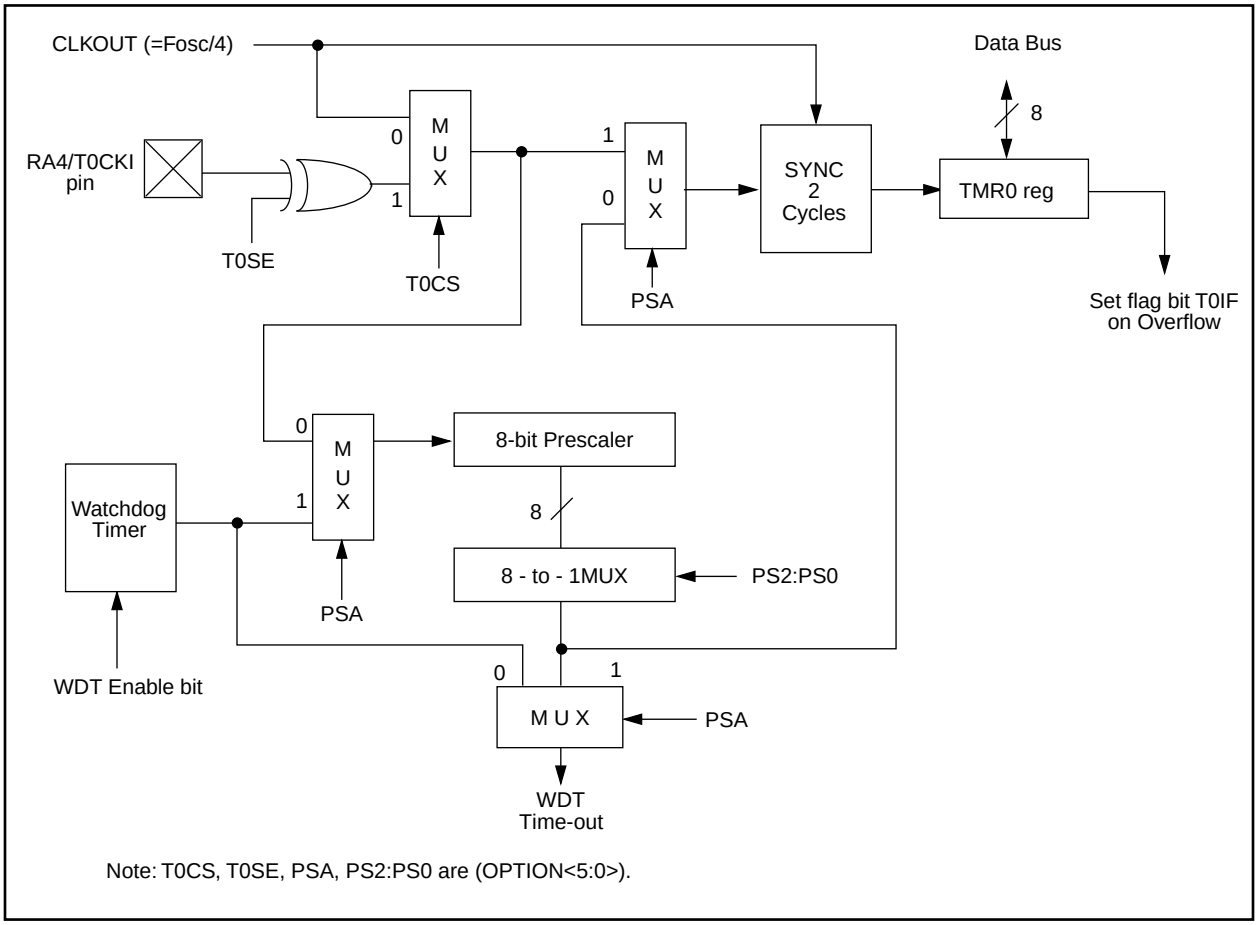
An 8-bit counter is available as a prescaler for the Timer0 module, or as a postscaler for the Watchdog Timer, respectively (Figure 6-6). For simplicity, this counter is being referred to as “prescaler” throughout this data sheet. Note that there is only one prescaler available which is mutually exclusively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

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

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

| **Note:** | Writing to TMR0 when the prescaler is assigned to Timer0 will clear the prescaler count, but will not change the prescaler assignment. |
|---|---|

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




Note: T0CS, T0SE, PSA, PS2:PS0 are (OPTION<5:0>).

**FIGURE 7-2: ADCON0 REGISTER (ADDRESS 1Fh), PIC16C715**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| ADCS1 | ADCS0 | — | CHS1 | CHS0 | GO/$\overline{\text{DONE}}$ | — | ADON |
| bit7 | | | | | | | bit0 |

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

bit 7-6: **ADCS1:ADCS0:** A/D Conversion Clock Select bits
00 = FOSC/2
01 = FOSC/8
10 = FOSC/32
11 = FRC (clock derived from an RC oscillation)

bit 5: **Unused**

bit 6-3: **CHS1:CHS0:** Analog Channel Select bits
000 = channel 0, (RA0/AN0)
001 = channel 1, (RA1/AN1)
010 = channel 2, (RA2/AN2)
011 = channel 3, (RA3/AN3)
100 = channel 0, (RA0/AN0)
101 = channel 1, (RA1/AN1)
110 = channel 2, (RA2/AN2)
111 = channel 3, (RA3/AN3)

bit 2: **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit

If ADON = 1
1 = A/D conversion in progress (setting this bit starts the A/D conversion)
0 = A/D conversion not in progress (This bit is automatically cleared by hardware when the A/D conversion is complete)

bit 1: **Unimplemented:** Read as '0'

bit 0: **ADON:** A/D On bit
1 = A/D converter module is operating
0 = A/D converter module is shutoff and consumes no operating current

**FIGURE 7-3: ADCON1 REGISTER, PIC16C710/71/711 (ADDRESS 88h), PIC16C715 (ADDRESS 9Fh)**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | PCFG1 | PCFG0 |
| bit7 | | | | | | | bit0 |

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

bit 7-2: **Unimplemented:** Read as '0'

bit 1-0: **PCFG1:PCFG0**: A/D Port Configuration Control bits

| PCFG1:PCFG0 | RA1 & RA0 | RA2 | RA3 | VREF |
|---|---|---|---|---|
| 00 | A | A | A | VDD |
| 01 | A | A | VREF | RA3 |
| 10 | A | D | D | VDD |
| 11 | D | D | D | VDD |

A = Analog input

D = Digital I/O

## 8.3 Reset

| Applicable Devices | 710 | 71 | 711 | 715 |
|---|---|---|---|---|

The PIC16CXX differentiates between various kinds of reset:

- Power-on Reset (POR)
- $\overline{MCLR}$ reset during normal operation
- $\overline{MCLR}$ reset during SLEEP
- WDT Reset (normal operation)
- Brown-out Reset (BOR) (PIC16C710/711/715)
- Parity Error Reset (PIC16C715)

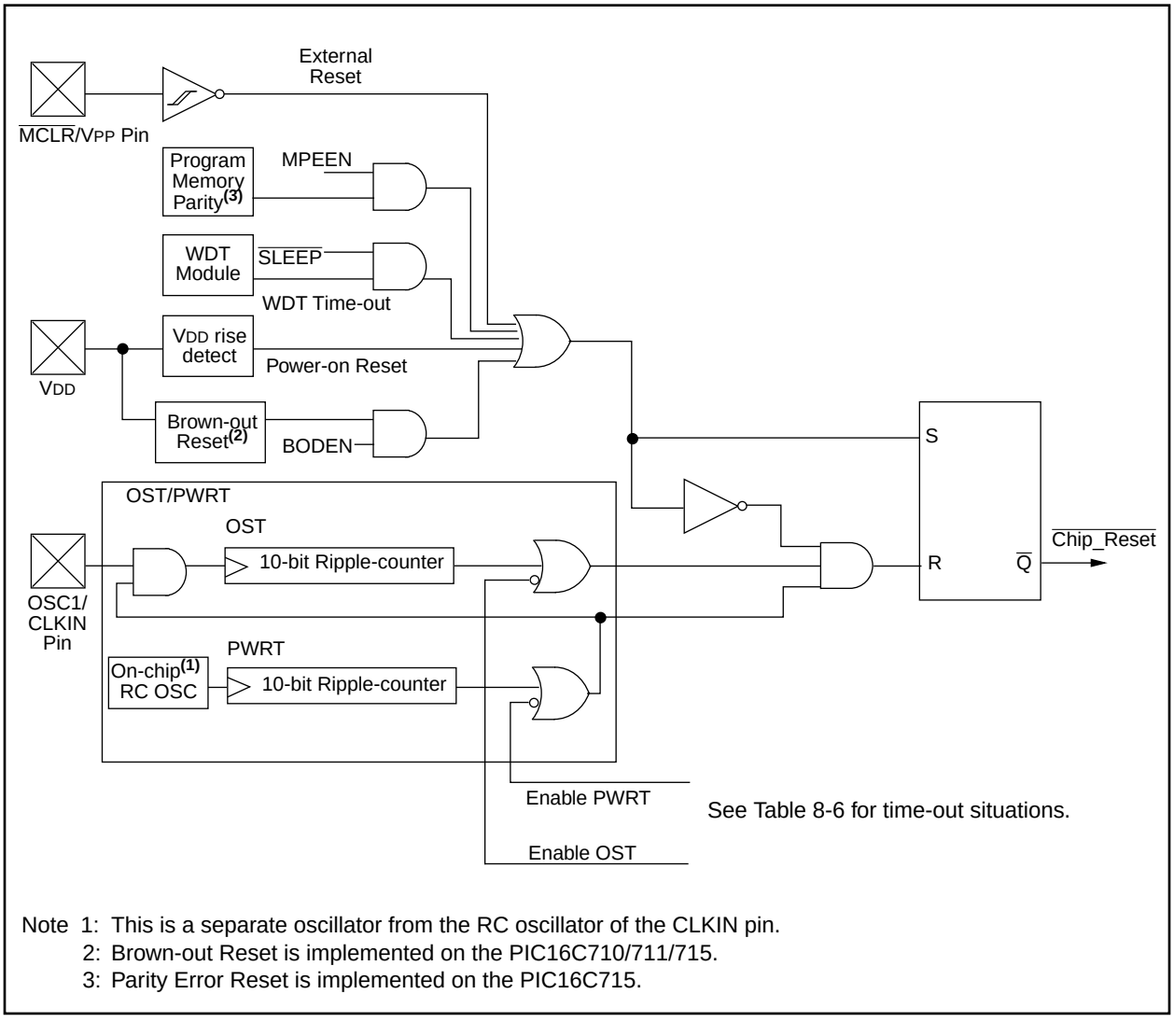
Some registers are not affected in any reset condition; their status is unknown on POR and unchanged in any other reset. Most other registers are reset to a "reset state" on Power-on Reset (POR), on the $\overline{MCLR}$ and WDT Reset, on $\overline{MCLR}$ reset during SLEEP, and Brown-out Reset (BOR). They are not affected by a WDT Wake-up, which is viewed as the resumption of normal operation. The $\overline{TO}$ and $\overline{PD}$ bits are set or cleared differently in different reset situations as indicated in Table 8-7, Table 8-8 and Table 8-9. These bits are used in software to determine the nature of the reset. See Table 8-10 and Table 8-11 for a full description of reset states of all registers.

A simplified block diagram of the on-chip reset circuit is shown in Figure 8-9.

The PIC16C710/711/715 have a $\overline{MCLR}$ noise filter in the $\overline{MCLR}$ reset path. The filter will detect and ignore small pulses.

It should be noted that a WDT Reset does not drive $\overline{MCLR}$ pin low.

**FIGURE 8-9: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT**




Note 1: This is a separate oscillator from the RC oscillator of the CLKIN pin.
2: Brown-out Reset is implemented on the PIC16C710/711/715.
3: Parity Error Reset is implemented on the PIC16C715.

**TABLE 8-13: INITIALIZATION CONDITIONS FOR ALL REGISTERS, PIC16C715**

| Register | Power-on Reset, Brown-out Reset Parity Error Reset | $\overline{\text{MCLR}}$ Resets WDT Reset | Wake-up via WDT or Interrupt |
|---|---|---|---|
| W | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| INDF | N/A | N/A | N/A |
| TMR0 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PCL | 0000 0000 | 0000 0000 | PC + 1(2) |
| STATUS | 0001 1xxx | 000q quuu(3) | uuuq quuu(3) |
| FSR | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTA | ---x 0000 | ---u 0000 | ---u uuuu |
| PORTB | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PCLATH | ---0 0000 | ---0 0000 | ---u uuuu |
| INTCON | 0000 000x | 0000 000u | uuuu uuuu(1) |
| PIR1 | -0-- ---- | -0-- ---- | -u-- ----(1) |
| ADCON0 | 0000 00-0 | 0000 00-0 | uuuu uu-u |
| OPTION | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISA | ---1 1111 | ---1 1111 | ---u uuuu |
| TRISB | 1111 1111 | 1111 1111 | uuuu uuuu |
| PIE1 | -0-- ---- | -0-- ---- | -u-- ---- |
| PCON | ---- -qqq | ---- -1uu | ---- -1uu |
| ADCON1 | ---- --00 | ---- --00 | ---- --uu |

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

Note 1: One or more bits in INTCON and PIR1 will be affected (to cause wake-up).

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

3: See Table 8-11 for reset value for specific condition.

**NOTES:**

## 9.0 INSTRUCTION SET SUMMARY

Each PIC16CXX instruction is a 14-bit word divided into an OPCODE which specifies the instruction type and one or more operands which further specify the operation of the instruction. The PIC16CXX instruction set summary in Table 9-2 lists **byte-oriented**, **bit-oriented**, and **literal and control** operations. Table 9-1 shows the opcode field descriptions.

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

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

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

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

**TABLE 9-1: OPCODE FIELD DESCRIPTIONS**

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

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

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

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

Table 9-2 lists the instructions recognized by the MPASM assembler.

Figure 9-1 shows the general formats that the instructions can have.

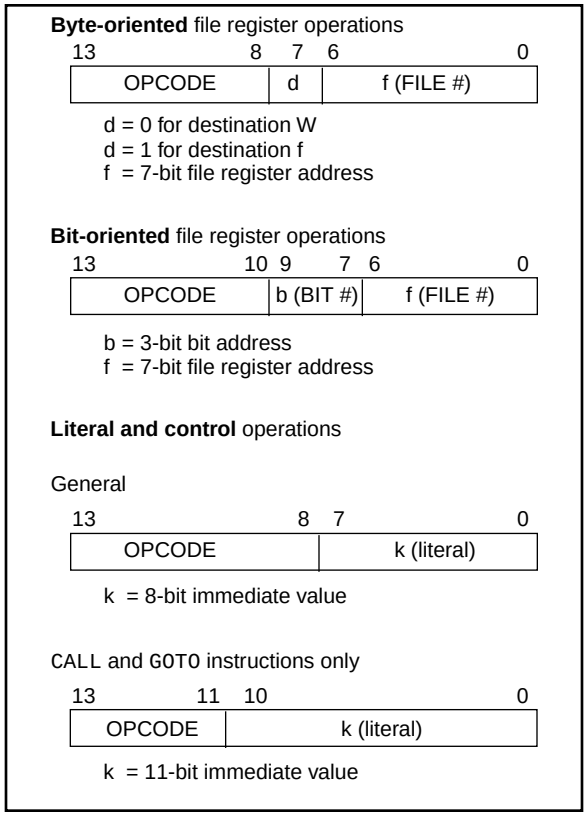
**Note:** To maintain upward compatibility with future PIC16CXX products, do not use the OPTION and TRIS instructions.

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

0xhh

where h signifies a hexadecimal digit.

**FIGURE 9-1: GENERAL FORMAT FOR INSTRUCTIONS**



**BTFSS** **Bit Test f, Skip if Set**

Syntax: [*label*] BTFSS f,b

Operands: 0 ≤ f ≤ 127
0 ≤ 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 | NOP |

If Skip: (2nd Cycle)

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| NOP | NOP | NOP | NOP |

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 ≤ k ≤ 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 | NOP | NOP | NOP | NOP |

Example

```
HERE    CALL    THERE
```

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

**IORWF** **Inclusive OR W with f**

Syntax: [ *label* ] IORWF f,d

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

Operation: (W) .OR. (f) → (dest)

Status Affected: Z

Encoding:

| | | | |
|---|---|---|---|
| 00 | 0100 | dfff | ffff |

Description: Inclusive OR the W register with register 'f'. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process data | Write to dest |

Example
```
IORWF   RESULT, 0
```
Before Instruction
RESULT = 0x13
W = 0x91
After Instruction
RESULT = 0x13
W = 0x93
Z = 1

**MOVF** **Move f**

Syntax: [ *label* ] MOVF f,d

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

Operation: (f) → (dest)

Status Affected: Z

Encoding:

| | | | |
|---|---|---|---|
| 00 | 1000 | dfff | ffff |

Description: The contents of register f is moved to a destination dependant upon the status of d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register since status flag Z is affected.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process data | Write to dest |

Example
```
MOVF    FSR, 0
```
After Instruction
W = value in FSR register
Z = 1

**MOVLW** **Move Literal to W**

Syntax: [ *label* ] MOVLW k

Operands: $0 \le k \le 255$

Operation: k → (W)

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 11 | 00xx | kkkk | kkkk |

Description: The eight bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process data | Write to W |

Example
```
MOVLW   0x5A
```
After Instruction
W = 0x5A

**MOVWF** **Move W to f**

Syntax: [ *label* ] MOVWF f

Operands: $0 \le f \le 127$

Operation: (W) → (f)

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 00 | 0000 | 1fff | ffff |

Description: Move data from W register to register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process data | Write register 'f' |

Example
```
MOVWF   OPTION_REG
```
Before Instruction
OPTION = 0xFF
W = 0x4F
After Instruction
OPTION = 0x4F
W = 0x4F

### 10.6 PICDEM-1 Low-Cost PIC16/17 Demonstration Board

The PICDEM-1 is a simple board which demonstrates the capabilities of several of Microchip's microcontrollers. The microcontrollers supported are: PIC16C5X (PIC16C54 to PIC16C58A), PIC16C61, PIC16C62X, PIC16C71, PIC16C8X, PIC17C42, PIC17C43 and PIC17C44. All necessary hardware and software is included to run basic demo programs. The users can program the sample microcontrollers provided with the PICDEM-1 board, on a PRO MATE II or PICSTART-Plus programmer, and easily test firmware. The user can also connect the PICDEM-1 board to the PICMASTER emulator and download the firmware to the emulator for testing. Additional prototype area is available for the user to build some additional hardware and connect it to the microcontroller socket(s). Some of the features include an RS-232 interface, a potentiometer for simulated analog input, push-button switches and eight LEDs connected to PORTB.

### 10.7 PICDEM-2 Low-Cost PIC16CXX Demonstration Board

The PICDEM-2 is a simple demonstration board that supports the PIC16C62, PIC16C64, PIC16C65, PIC16C73 and PIC16C74 microcontrollers. All the necessary hardware and software is included to run the basic demonstration programs. The user can program the sample microcontrollers provided with the PICDEM-2 board, on a PRO MATE II programmer or PICSTART-Plus, and easily test firmware. The PICMASTER emulator may also be used with the PICDEM-2 board to test firmware. Additional prototype area has been provided to the user for adding additional hardware and connecting it to the microcontroller socket(s). Some of the features include a RS-232 interface, push-button switches, a potentiometer for simulated analog input, a Serial EEPROM to demonstrate usage of the $I^2C$ bus and separate headers for connection to an LCD module and a keypad.

### 10.8 PICDEM-3 Low-Cost PIC16CXXX Demonstration Board

The PICDEM-3 is a simple demonstration board that supports the PIC16C923 and PIC16C924 in the PLCC package. It will also support future 44-pin PLCC microcontrollers with a LCD Module. All the necessary hardware and software is included to run the basic demonstration programs. The user can program the sample microcontrollers provided with the PICDEM-3 board, on a PRO MATE II programmer or PICSTART Plus with an adapter socket, and easily test firmware. The PICMASTER emulator may also be used with the PICDEM-3 board to test firmware. Additional prototype area has been provided to the user for adding hardware and connecting it to the microcontroller socket(s). Some of the features include an RS-232 interface, push-button switches, a potentiometer for simulated analog input, a thermistor and separate headers for connection to an external LCD module and a keypad. Also provided on the PICDEM-3 board is an LCD panel, with 4 commons and 12 segments, that is capable of displaying time, temperature and day of the week. The PICDEM-3 provides an additional RS-232 interface and Windows 3.1 software for showing the demultiplexed LCD signals on a PC. A simple serial interface allows the user to construct a hardware demultiplexer for the LCD signals.

### 10.9 MPLAB Integrated Development Environment Software

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

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

MPLAB allows you to:

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

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

### 10.10 Assembler (MPASM)

The MPASM Universal Macro Assembler is a PC-hosted symbolic assembler. It supports all microcontroller series including the PIC12C5XX, PIC14000, PIC16C5X, PIC16CXXX, and PIC17CXX families.

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

MPASM allows full symbolic debugging from PICMASTER, Microchip's Universal Emulator System.

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

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

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

### 10.11 Software Simulator (MPLAB-SIM)

The MPLAB-SIM Software Simulator allows code development in a PC host environment. It allows the user to simulate the PIC16/17 series microcontrollers on an instruction level. On any given instruction, the user may examine or modify any of the data areas or provide external stimulus to any of the pins. The input/output radix can be set by the user and the execution can be performed in; single step, execute until break, or in a trace mode.

MPLAB-SIM fully supports symbolic debugging using MPLAB-C and MPASM. The Software Simulator offers the low cost flexibility to develop and debug code outside of the laboratory environment making it an excellent multi-project software development tool.

### 10.12 C Compiler (MPLAB-C)

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

For easier source level debugging, the compiler provides symbol information that is compatible with the MPLAB IDE memory display.

### 10.13 Fuzzy Logic Development System (*fuzzy*TECH-MP)

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

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

### 10.14 MP-DriveWay™ – Application Code Generator

MP-DriveWay is an easy-to-use Windows-based Application Code Generator. With MP-DriveWay you can visually configure all the peripherals in a PIC16/17 device and, with a click of the mouse, generate all the initialization and many functional code modules in C language. The output is fully compatible with Microchip's MPLAB-C C compiler. The code produced is highly modular and allows easy integration of your own code. MP-DriveWay is intelligent enough to maintain your code through subsequent code generation.

### 10.15 SEEVAL® Evaluation and Programming System

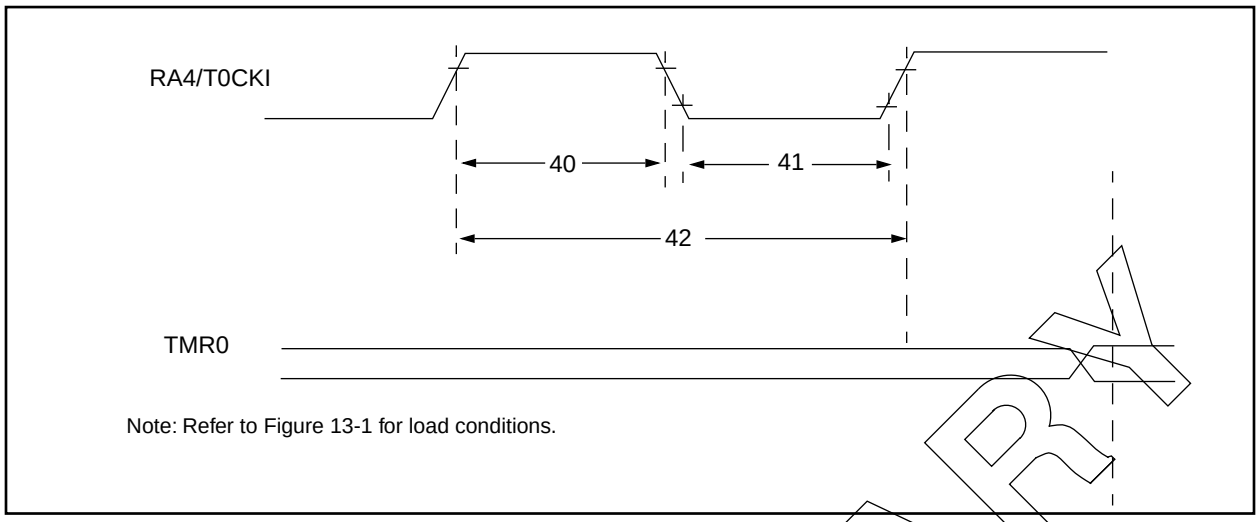
The SEEVAL SEEPROM Designer's Kit supports all Microchip 2-wire and 3-wire Serial EEPROMs. The kit includes everything necessary to read, write, erase or program special features of any Microchip SEEPROM product including Smart Serials™ and secure serials. The Total Endurance™ Disk is included to aid in trade-off analysis and reliability calculations. The total kit can significantly reduce time-to-market and result in an optimized system.

### 10.16 KEELOQ® Evaluation and Programming Tools

KEELOQ evaluation and programming tools support Microchips HCS Secure Data Products. The HCS evaluation kit includes an LCD display to show changing codes, a decoder to decode transmissions, and a programming interface to program test transmitters.

| Applicable Devices | 710 | 71 | 711 | 715 |
|---|---|---|---|---|

**FIGURE 13-6: TIMER0 CLOCK TIMINGS**




Note: Refer to Figure 13-1 for load conditions.

**TABLE 13-5: TIMER0 CLOCK REQUIREMENTS**

| Param No. | Sym | Characteristic | | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 40 | Tt0H | T0CKI High Pulse Width | No Prescaler | 0.5TCY + 20* | — | — | ns | |
| | | | With Prescaler | 10* | — | — | ns | |
| 41 | Tt0L | T0CKI Low Pulse Width | No Prescaler | 0.5TCY + 20* | — | — | ns | |
| | | | With Prescaler | 10* | — | — | ns | |
| 42 | Tt0P | T0CKI Period | | Greater of: 20µs or $\frac{T_{CY} + 40}{N}$* | — | — | ns | N = prescale value (1, 2, 4,..., 256) |
| 48 | Tcke2tmrl | Delay from external clock edge to timer increment | | 2Tosc | — | 7Tosc | — | |

* 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.

| Applicable Devices | 710 | 71 | 711 | 715 |
|---|---|---|---|---|

## 14.0 DC AND AC CHARACTERISTICS GRAPHS AND TABLES FOR PIC16C715

The graphs and tables provided in this section are for design guidance and are not tested or guaranteed.

**In some graphs or tables the data presented are outside specified operating range (i.e., outside specified $V_{DD}$ range). This is for information only and devices are guaranteed to operate properly only within the specified range.**

**Note:** The data presented in this section is a statistical summary of data collected on units from different lots over a period of time and matrix samples. 'Typical' represents the mean of the distribution at, 25°C, while 'max' or 'min' represents (mean +3σ) and (mean -3σ) respectively where σ is standard deviation.

**FIGURE 14-1: TYPICAL $I_{PD}$ vs. $V_{DD}$ (WDT DISABLED, RC MODE)**

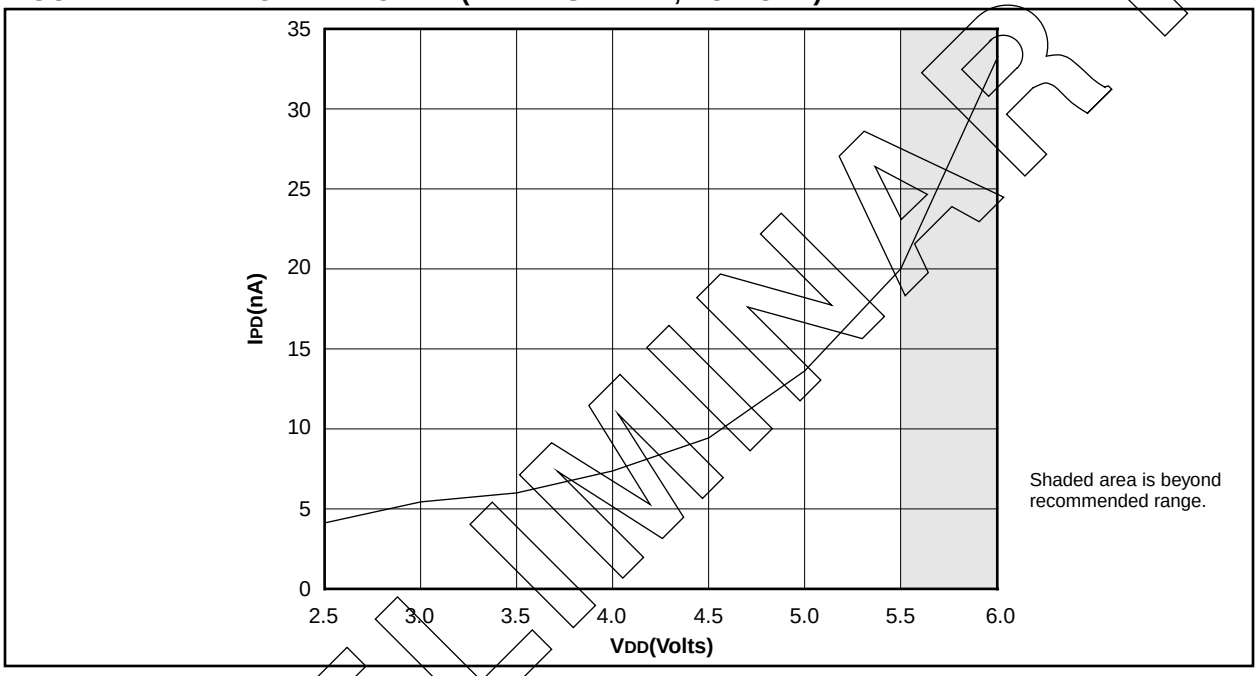
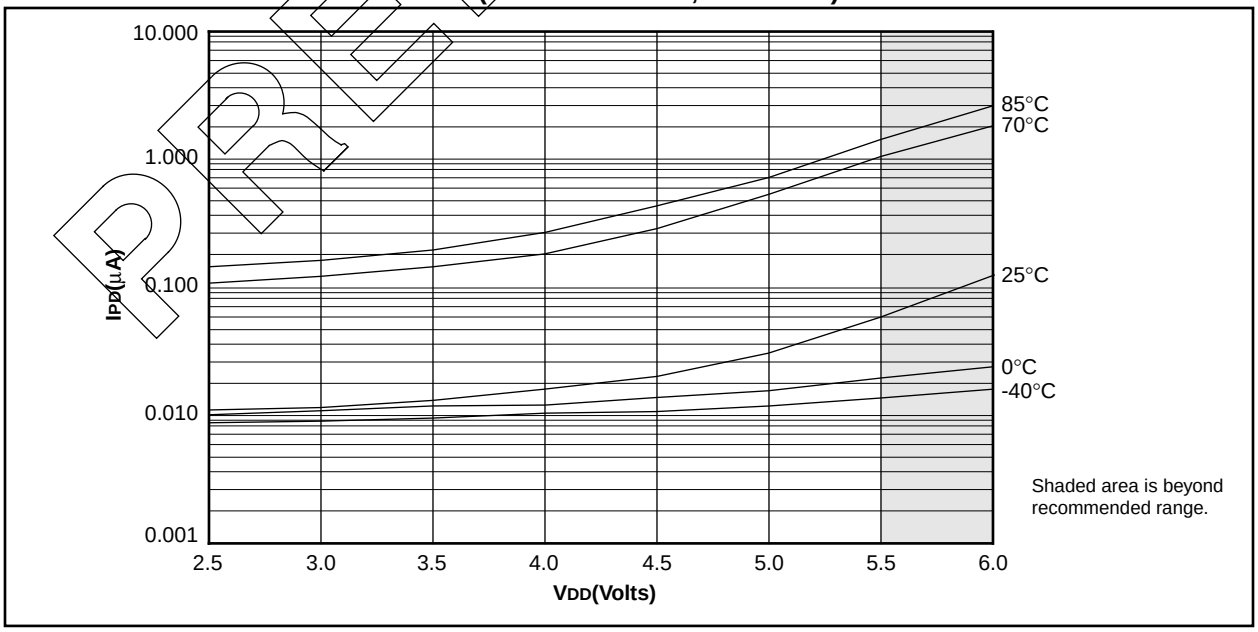



**FIGURE 14-2: MAXIMUM $I_{PD}$ vs. $V_{DD}$ (WDT DISABLED, RC MODE)**



| Applicable Devices | 710 | 71 | 711 | 715 |
|---|---|---|---|---|

## 15.0 ELECTRICAL CHARACTERISTICS FOR PIC16C71

**Absolute Maximum Ratings †**

Ambient temperature under bias ..... -55 to +125˚C

Storage temperature ..... -65˚C to +150˚C

Voltage on any pin with respect to VSS (except VDD, $\overline{MCLR}$, and RA4) ..... -0.3V to (VDD + 0.3V)

Voltage on VDD with respect to VSS ..... -0.3 to +7.5V

Voltage on $\overline{MCLR}$ with respect to VSS (Note 2) ..... 0 to +14V

Voltage on RA4 with respect to Vss ..... 0 to +14V

Total power dissipation (Note 1) ..... 800 mW

Maximum current out of VSS pin ..... 150 mA

Maximum current into VDD pin ..... 100 mA

Input clamp current, IIK (VI < 0 or VI > VDD) ..... ± 20 mA

Output clamp current, IOK (VO < 0 or VO > VDD) ..... ± 20 mA

Maximum output current sunk by any I/O pin ..... 25 mA

Maximum output current sourced by any I/O pin ..... 20 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:** Power dissipation is calculated as follows: Pdis = VDD x {IDD - ∑ IOH} + ∑ {(VDD-VOH) x IOH} + ∑(Vol x IOL)

**Note 2:** Voltage spikes below VSS at the $\overline{MCLR}$ 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 $\overline{MCLR}$ pin rather than pulling this pin directly to VSS.

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

**TABLE 15-1: CROSS REFERENCE OF DEVICE SPECS FOR OSCILLATOR CONFIGURATIONS AND FREQUENCIES OF OPERATION (COMMERCIAL DEVICES)**

| OSC | PIC16C71-04 | PIC16C71-20 | PIC16LC71-04 | JW Devices |
|---|---|---|---|---|
| RC | VDD: 4.0V to 6.0V<br>IDD: 3.3 mA max. at 5.5V<br>IPD: 14 µA max. at 4V<br>Freq:4 MHz max. | VDD: 4.5V to 5.5V<br>IDD: 1.8 mA typ. at 5.5V<br>IPD: 1.0 µA typ. at 4V<br>Freq: 4 MHz max. | VDD: 3.0V to 6.0V<br>IDD: 1.4 mA typ. at 3.0V<br>IPD: 0.6 µA typ. at 3V<br>Freq: 4 MHz max. | VDD: 4.0V to 6.0V<br>IDD: 3.3 mA max. at 5.5V<br>IPD: 14 µA max. at 4V<br>Freq:4 MHz max. |
| XT | VDD: 4.0V to 6.0V<br>IDD: 3.3 mA max. at 5.5V<br>IPD: 14 µA max. at 4V<br>Freq: 4 MHz max. | VDD: 4.5V to 5.5V<br>IDD: 1.8 mA typ. at 5.5V<br>IPD: 1.0 µA typ. at 4V<br>Freq: 4 MHz max. | VDD: 3.0V to 6.0V<br>IDD: 1.4 mA typ. at 3.0V<br>IPD: 0.6 µA typ. at 3V<br>Freq: 4 MHz max. | VDD: 4.0V to 6.0V<br>IDD: 3.3 mA max. at 5.5V<br>IPD: 14 µA max. at 4V<br>Freq: 4 MHz max. |
| HS | VDD: 4.5V to 5.5V<br>IDD: 13.5 mA typ. at 5.5V<br>IPD: 1.0 µA typ. at 4.5V<br>Freq: 4 MHz max. | VDD: 4.5V to 5.5V<br>IDD: 30 mA max. at 5.5V<br>IPD: 1.0 µA typ. at 4.5V<br>Freq: 20 MHz max. | Not recommended for use in HS mode | VDD: 4.5V to 5.5V<br>IDD: 30 mA max. at 5.5V<br>IPD: 1.0 µA typ. at 4.5V<br>Freq: 20 MHz max. |
| LP | VDD: 4.0V to 6.0V<br>IDD: 15 µA typ. at 32 kHz, 4.0V<br>IPD: 0.6 µA typ. at 4.0V<br>Freq: 200 kHz max. | Not recommended for use in LP mode | VDD: 3.0V to 6.0V<br>IDD: 32 µA max. at 32 kHz, 3.0V<br>IPD: 9 µA max. at 3.0V<br>Freq: 200 kHz max. | VDD: 3.0V to 6.0V<br>IDD: 32 µA max. at 32 kHz, 3.0V<br>IPD: 9 µA max. at 3.0V<br>Freq: 200 kHz max. |

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

| Applicable Devices | 710 | 71 | 711 | 715 |
|---|---|---|---|---|

**FIGURE 16-7: MAXIMUM IPD VS. VDD WATCHDOG DISABLED**

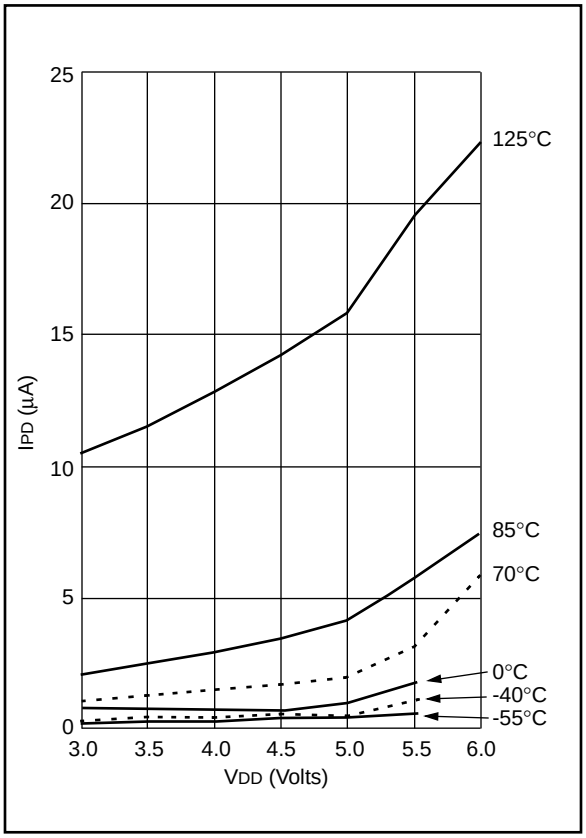
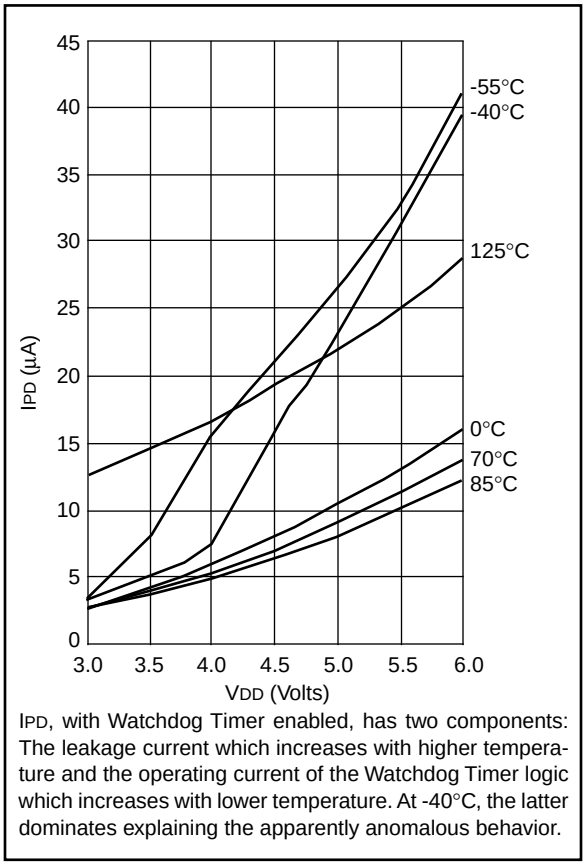
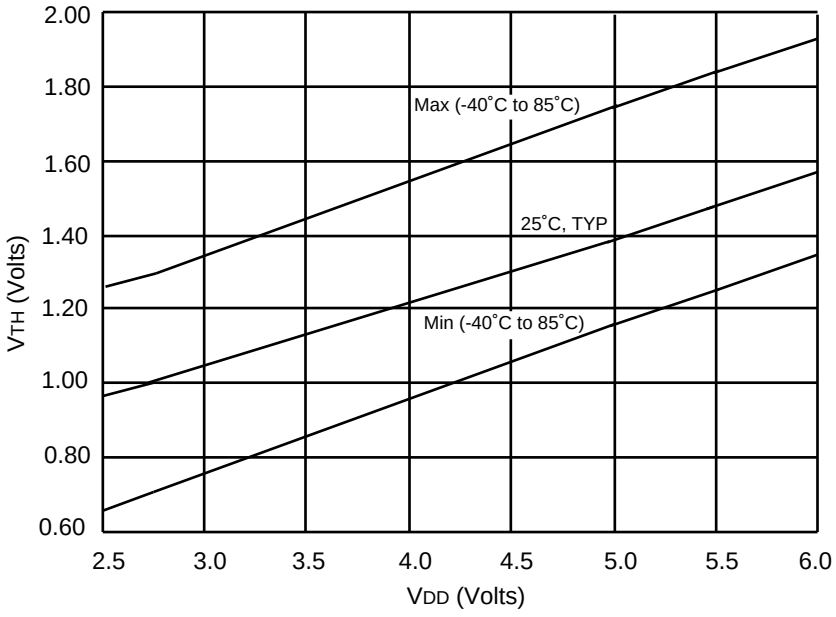



**FIGURE 16-8: MAXIMUM IPD VS. VDD WATCHDOG ENABLED**



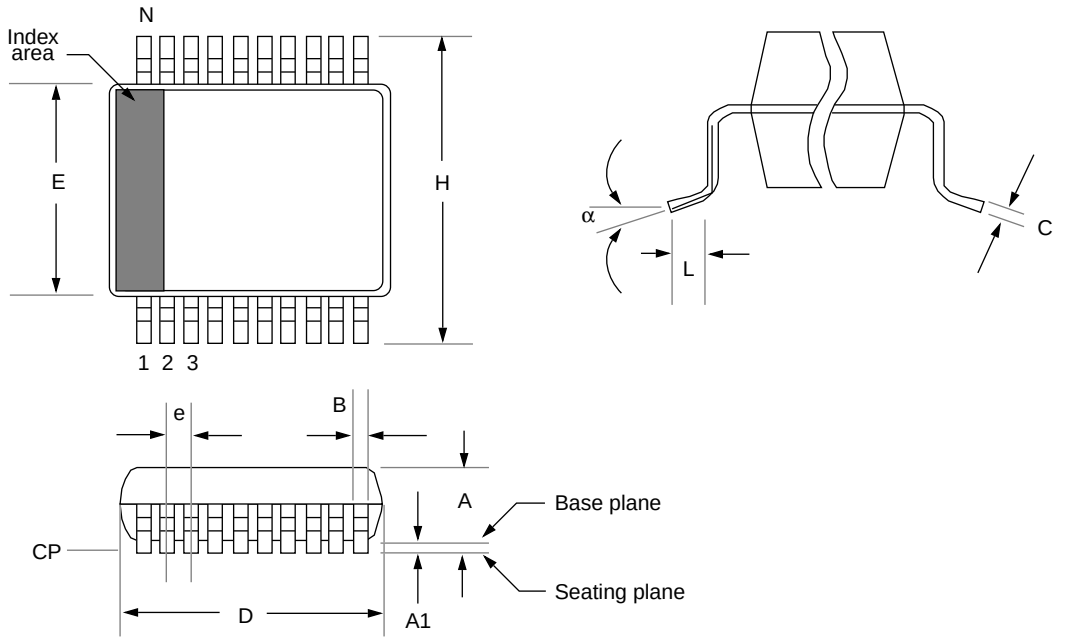

IPD, with Watchdog Timer enabled, has two components: The leakage current which increases with higher temperature and the operating current of the Watchdog Timer logic which increases with lower temperature. At -40°C, the latter dominates explaining the apparently anomalous behavior.

**FIGURE 16-9: VTH (INPUT THRESHOLD VOLTAGE) OF I/O PINS VS. VDD**




Data based on matrix samples. See first page of this section for details.

## 17.4 20-Lead Plastic Surface Mount (SSOP - 209 mil Body 5.30 mm) (SS)




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

Note 1: Dimensions D1 and E1 do not include mold protrusion. Allowable mold protrusion is 0.25m/m (0.010”) per side. D1 and E1 dimensions including mold mismatch.

2: Dimension “b” does not include Dambar protrusion, allowable Dambar protrusion shall be 0.08m/m (0.003”)max.

3: This outline conforms to JEDEC MS-026.

## APPENDIX C: WHAT'S NEW

1. Consolidated all pin compatible 18-pin A/D based devices into one data sheet.

## APPENDIX D: WHAT'S CHANGED

1. Minor changes, spelling and grammatical changes.
2. Low voltage operation on the PIC16LC710/711/715 has been reduced from 3.0V to 2.5V.
3. Part numbers of the PIC16C70 and PIC16C71A have changed to PIC16C710 and PIC16C711, respectively.

## PIC16C71X PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery refer to the factory or the listed sales office.

**PART NO. -XX X /XX XXX**

| | | | |
|---|---|---|---|
| **Pattern:** | QTP, SQTP, Code or Special Requirements | | |
| **Package:** | JW | = | Windowed CERDIP |
| | SO | = | SOIC |
| | SP | = | Skinny plastic dip |
| | P | = | PDIP |
| | SS | = | SSOP |
| **Temperature Range:** | - | = | 0°C to +70°C |
| | I | = | -40°C to +85°C |
| | E | = | -40°C to +125°C |
| **Frequency Range:** | 04 | = | 200 kHz (PIC16C7X-04) |
| | 04 | = | 4 MHz |
| | 10 | = | 10 MHz |
| | 20 | = | 20 MHz |
| **Device** | PIC16C7X | :VDD range 4.0V to 6.0V | |
| | PIC16C7XT | :VDD range 4.0V to 6.0V (Tape/Reel) | |
| | PIC16LC7X | :VDD range 2.5V to 6.0V | |
| | PIC16LC7XT | :VDD range 2.5V to 6.0V (Tape/Reel) | |

**Examples**

a) PIC16C71 - 04/P 301 Commercial Temp., PDIP Package, 4 MHz, normal VDD limits, QTP pattern #301

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

### Sales and Support

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

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

Please specify which device, revision of silicon and Data Sheet (include Literature #) you are using.
For latest version information and upgrade kits for Microchip Development Tools, please call 1-800-755-2345 or 1-602-786-7302.