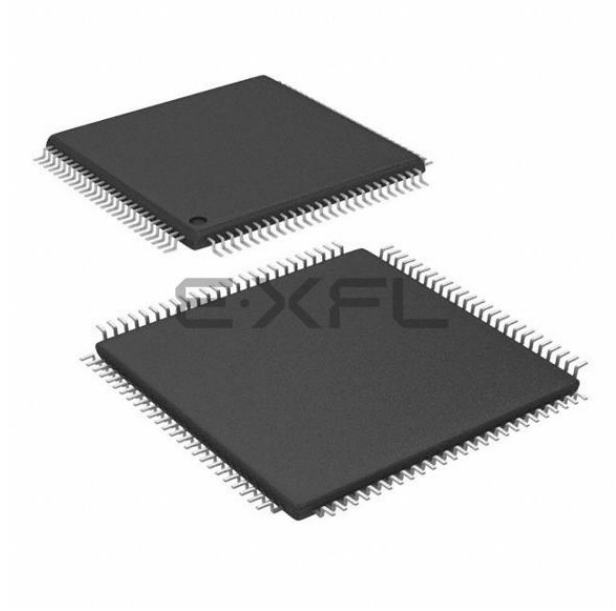


**Welcome to E-XFL.COM**

## 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 | 16-Bit |
| Speed | 32MHz |
| Connectivity | I²C, SPI, UART/USART, USB OTG |
| Peripherals | Brown-out Detect/Reset, LVD, POR, PWM, WDT |
| Number of I/O | 83 |
| Program Memory Size | 192KB (65.5K x 24) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 16K x 8 |
| Voltage - Supply (Vcc/Vdd) | 2V ~ 3.6V |
| Data Converters | A/D 16x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 100-TQFP |
| Supplier Device Package | 100-TQFP (14x14) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic24fj192gb110t-i-pf |

**TABLE 1-4: PIC24FJ256GB110 FAMILY PINOUT DESCRIPTIONS (CONTINUED)**

| Function | Pin Number | | | I/O | Input Buffer | Description |
|---|---|---|---|---|---|---|
| | 64-Pin TQFP, QFN | 80-Pin TQFP | 100-Pin TQFP | | | |
| PMD0 | 60 | 76 | 93 | I/O | ST/TTL | Parallel Master Port Data (Demultiplexed Master mode) or Address/Data (Multiplexed Master modes). |
| PMD1 | 61 | 77 | 94 | I/O | ST/TTL | |
| PMD2 | 62 | 78 | 98 | I/O | ST/TTL | |
| PMD3 | 63 | 79 | 99 | I/O | ST/TTL | |
| PMD4 | 64 | 80 | 100 | I/O | ST/TTL | |
| PMD5 | 1 | 1 | 3 | I/O | ST/TTL | |
| PMD6 | 2 | 2 | 4 | I/O | ST/TTL | |
| PMD7 | 3 | 3 | 5 | I/O | ST/TTL | |
| PMRD | 53 | 67 | 82 | O | — | Parallel Master Port Read Strobe. |
| PMWR | 52 | 66 | 81 | O | — | Parallel Master Port Write Strobe. |
| RA0 | — | — | 17 | I/O | ST | PORTA Digital I/O. |
| RA1 | — | — | 38 | I/O | ST | |
| RA2 | — | — | 58 | I/O | ST | |
| RA3 | — | — | 59 | I/O | ST | |
| RA4 | — | — | 60 | I/O | ST | |
| RA5 | — | — | 61 | I/O | ST | |
| RA6 | — | — | 91 | I/O | ST | |
| RA7 | — | — | 92 | I/O | ST | |
| RA9 | — | 23 | 28 | I/O | ST | |
| RA10 | — | 24 | 29 | I/O | ST | |
| RA14 | — | 52 | 66 | I/O | ST | |
| RA15 | — | 53 | 67 | I/O | ST | |
| RB0 | 16 | 20 | 25 | I/O | ST | PORTB Digital I/O. |
| RB1 | 15 | 19 | 24 | I/O | ST | |
| RB2 | 14 | 18 | 23 | I/O | ST | |
| RB3 | 13 | 17 | 22 | I/O | ST | |
| RB4 | 12 | 16 | 21 | I/O | ST | |
| RB5 | 11 | 15 | 20 | I/O | ST | |
| RB6 | 17 | 21 | 26 | I/O | ST | |
| RB7 | 18 | 22 | 27 | I/O | ST | |
| RB8 | 21 | 27 | 32 | I/O | ST | |
| RB9 | 22 | 28 | 33 | I/O | ST | |
| RB10 | 23 | 29 | 34 | I/O | ST | |
| RB11 | 24 | 30 | 35 | I/O | ST | |
| RB12 | 27 | 33 | 41 | I/O | ST | |
| RB13 | 28 | 34 | 42 | I/O | ST | |
| RB14 | 29 | 35 | 43 | I/O | ST | |
| RB15 | 30 | 36 | 44 | I/O | ST | |

**Legend:** TTL = TTL input buffer ST = Schmitt Trigger input buffer
ANA = Analog level input/output $I^2C$™ = $I^2C$/SMBus input buffer

## 2.0 GUIDELINES FOR GETTING STARTED WITH 16-BIT MICROCONTROLLERS

### 2.1 Basic Connection Requirements

Getting started with the PIC24FJ256GB110 family of 16-bit microcontrollers requires attention to a minimal set of device pin connections before proceeding with development.

The following pins must always be connected:

- All VDD and VSS pins
  (see **Section 2.2 "Power Supply Pins"**)
- All AVDD and AVSS pins, regardless of whether or not the analog device features are used
  (see **Section 2.2 "Power Supply Pins"**)
- $\overline{\text{MCLR}}$ pin
  (see **Section 2.3 "Master Clear (MCLR) Pin"**)
- ENVREG/DISVREG and VCAP/VDDCORE pins (PIC24FJ devices only)
  (see **Section 2.4 "Voltage Regulator Pins (ENVREG/DISVREG and VCAP/VDDCORE)"**)

These pins must also be connected if they are being used in the end application:

- PGECx/PGEDx pins used for In-Circuit Serial Programming™ (ICSP™) and debugging purposes
  (see **Section 2.5 "ICSP Pins"**)
- OSCI and OSCO pins when an external oscillator source is used
  (see **Section 2.6 "External Oscillator Pins"**)

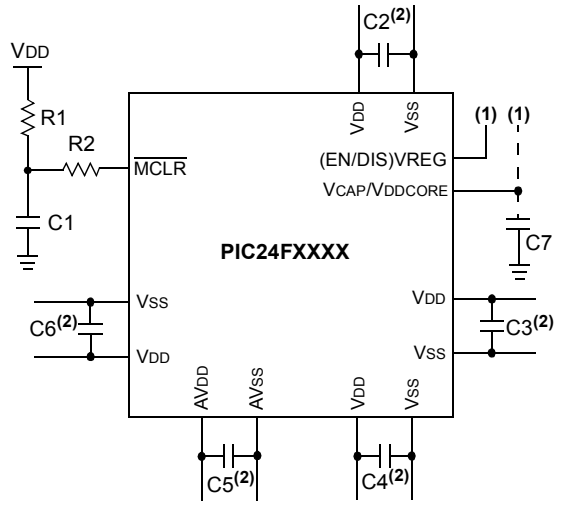
Additionally, the following pins may be required:

- VREF+/VREF- pins used when external voltage reference for analog modules is implemented

| **Note:** | The AVDD and AVSS pins must always be connected, regardless of whether any of the analog modules are being used. |
|---|---|

The minimum mandatory connections are shown in Figure 2-1.

**FIGURE 2-1: RECOMMENDED MINIMUM CONNECTIONS**




**Key (all values are recommendations):**

C1 through C6: 0.1 μF, 20V ceramic

C7: 10 μF, 6.3V or greater, tantalum or ceramic

R1: 10 kΩ

R2: 100Ω to 470Ω

**Note 1:** See **Section 2.4 "Voltage Regulator Pins (ENVREG/DISVREG and VCAP/VDDCORE)"** for explanation of ENVREG/DISVREG pin connections.

**2:** The example shown is for a PIC24F device with five VDD/VSS and AVDD/AVSS pairs. Other devices may have more or less pairs; adjust the number of decoupling capacitors appropriately.

## 2.6 External Oscillator Pins

Many microcontrollers have options for at least two oscillators: a high-frequency primary oscillator and a low-frequency secondary oscillator (refer to **Section 8.0 "Oscillator Configuration"** for details).

The oscillator circuit should be placed on the same side of the board as the device. Place the oscillator circuit close to the respective oscillator pins with no more than 0.5 inch (12 mm) between the circuit components and the pins. The load capacitors should be placed next to the oscillator itself, on the same side of the board.

Use a grounded copper pour around the oscillator circuit to isolate it from surrounding circuits. The grounded copper pour should be routed directly to the MCU ground. Do not run any signal traces or power traces inside the ground pour. Also, if using a two-sided board, avoid any traces on the other side of the board where the crystal is placed.

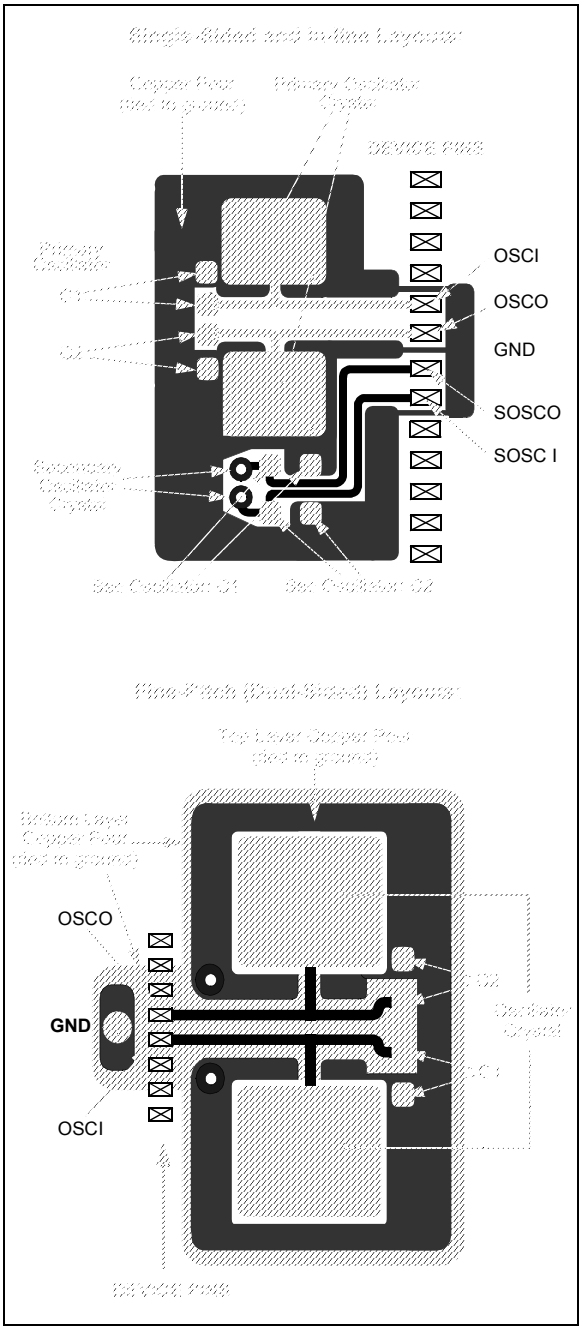
Layout suggestions are shown in Figure 2-4. In-line packages may be handled with a single-sided layout that completely encompasses the oscillator pins. With fine-pitch packages, it is not always possible to completely surround the pins and components. A suitable solution is to tie the broken guard sections to a mirrored ground layer. In all cases, the guard trace(s) must be returned to ground.

In planning the application's routing and I/O assignments, ensure that adjacent port pins and other signals in close proximity to the oscillator are benign (i.e., free of high frequencies, short rise and fall times and other similar noise).

For additional information and design guidance on oscillator circuits, please refer to these Microchip Application Notes, available at the corporate web site (www.microchip.com):

- *AN826, "Crystal Oscillator Basics and Crystal Selection for rfPIC™ and PICmicro® Devices"*
- *AN849, "Basic PICmicro® Oscillator Design"*
- *AN943, "Practical PICmicro® Oscillator Analysis and Design"*
- *AN949, "Making Your Oscillator Work"*

**FIGURE 2-4: SUGGESTED PLACEMENT OF THE OSCILLATOR CIRCUIT**





© 2009 Microchip Technology Inc.

### 5.6.1 PROGRAMMING ALGORITHM FOR FLASH PROGRAM MEMORY

The user can program one row of Flash program memory at a time. To do this, it is necessary to erase the 8-row erase block containing the desired row. The general process is:

1. Read eight rows of program memory (512 instructions) and store in data RAM.
2. Update the program data in RAM with the desired new data.
3. Erase the block (see Example 5-1 for an implementation in assembler):
   a) Set the NVMOP bits (NVMCON<3:0>) to '0010' to configure for block erase. Set the ERASE (NVMCON<6>) and WREN (NVMCON<14>) bits.
   b) Write the starting address of the block to be erased into the TBLPAG and W registers.
   c) Write 55h to NVMKEY.
   d) Write AAh to NVMKEY.
   e) Set the WR bit (NVMCON<15>). The erase cycle begins and the CPU stalls for the duration of the erase cycle. When the erase is done, the WR bit is cleared automatically.
4. Write the first 64 instructions from data RAM into the program memory buffers (see Example 5-3 for the implementation in assembler).
5. Write the program block to Flash memory:
   a) Set the NVMOP bits to '0001' to configure for row programming. Clear the ERASE bit and set the WREN bit.
   b) Write 55h to NVMKEY.
   c) Write AAh to NVMKEY.
   d) Set the WR bit. The programming cycle begins and the CPU stalls for the duration of the write cycle. When the write to Flash memory is done, the WR bit is cleared automatically.
6. Repeat steps 4 and 5, using the next available 64 instructions from the block in data RAM by incrementing the value in TBLPAG, until all 512 instructions are written back to Flash memory.

For protection against accidental operations, the write initiate sequence for NVMKEY must be used to allow any erase or program operation to proceed. After the programming command has been executed, the user must wait for the programming time until programming is complete. The two instructions following the start of the programming sequence should be NOPs, as shown in Example 5-5.

| **Note:** | The equivalent C code for these steps, prepared using Microchip's MPLAB C30 compiler and specific library of built-in hardware functions, is shown in Examples 5-2, 5-4 and 5-6. |
|---|---|

**EXAMPLE 5-1: ERASING A PROGRAM MEMORY BLOCK (ASSEMBLY LANGUAGE CODE)**

```
; Set up NVMCON for block erase operation
       MOV     #0x4042, W0                 ;
       MOV     W0, NVMCON                  ; Initialize NVMCON
; Init pointer to row to be ERASED
       MOV     #tblpage(PROG_ADDR), W0     ;
       MOV     W0, TBLPAG                  ; Initialize PM Page Boundary SFR
       MOV     #tbloffset(PROG_ADDR), W0   ; Initialize in-page EA[15:0] pointer
       TBLWTL  W0, [W0]                    ; Set base address of erase block
       DISI    #5                          ; Block all interrupts with priority <7
                                           ; for next 5 instructions
       MOV     #0x55, W0
       MOV     W0, NVMKEY                  ; Write the 55 key
       MOV     #0xAA, W1                   ;
       MOV     W1, NVMKEY                  ; Write the AA key
       BSET    NVMCON, #WR                 ; Start the erase sequence
       NOP                                 ; Insert two NOPs after the erase
       NOP                                 ; command is asserted
```


© 2009 Microchip Technology Inc.

## 7.0 INTERRUPT CONTROLLER

| | |
|---|---|
| **Note:** | This data sheet summarizes the features of this group of PIC24F devices. It is not intended to be a comprehensive reference source. For more information, refer to the *"PIC24F Family Reference Manual"*, **Section 8. "Interrupts"** (DS39707). |

The PIC24F interrupt controller reduces the numerous peripheral interrupt request signals to a single interrupt request signal to the PIC24F CPU. It has the following features:

- Up to 8 processor exceptions and software traps
- 7 user-selectable priority levels
- Interrupt Vector Table (IVT) with up to 118 vectors
- A unique vector for each interrupt or exception source
- Fixed priority within a specified user priority level
- Alternate Interrupt Vector Table (AIVT) for debug support
- Fixed interrupt entry and return latencies

### 7.1 Interrupt Vector Table

The Interrupt Vector Table (IVT) is shown in Figure 7-1. The IVT resides in program memory, starting at location 000004h. The IVT contains 126 vectors, consisting of 8 non-maskable trap vectors, plus up to 118 sources of interrupt. In general, each interrupt source has its own vector. Each interrupt vector contains a 24-bit wide address. The value programmed into each interrupt vector location is the starting address of the associated Interrupt Service Routine (ISR).

Interrupt vectors are prioritized in terms of their natural priority; this is linked to their position in the vector table. All other things being equal, lower addresses have a higher natural priority. For example, the interrupt associated with vector 0 will take priority over interrupts at any other vector address.

PIC24FJ256GB110 family devices implement non-maskable traps and unique interrupts. These are summarized in Table 7-1 and Table 7-2.

#### 7.1.1 ALTERNATE INTERRUPT VECTOR TABLE

The Alternate Interrupt Vector Table (AIVT) is located after the IVT, as shown in Figure 7-1. Access to the AIVT is provided by the ALTIVT control bit (INTCON2<15>). If the ALTIVT bit is set, all interrupt and exception processes will use the alternate vectors instead of the default vectors. The alternate vectors are organized in the same manner as the default vectors.

The AIVT supports emulation and debugging efforts by providing a means to switch between an application and a support environment without requiring the interrupt vectors to be reprogrammed. This feature also enables switching between applications for evaluation of different software algorithms at run time. If the AIVT is not needed, the AIVT should be programmed with the same addresses used in the IVT.

### 7.2 Reset Sequence

A device Reset is not a true exception because the interrupt controller is not involved in the Reset process. The PIC24F devices clear their registers in response to a Reset which forces the PC to zero. The microcontroller then begins program execution at location 000000h. The user programs a GOTO instruction at the Reset address, which redirects program execution to the appropriate start-up routine.

| | |
|---|---|
| **Note:** | Any unimplemented or unused vector locations in the IVT and AIVT should be programmed with the address of a default interrupt handler routine that contains a RESET instruction. |

**REGISTER 7-12: IEC1: INTERRUPT ENABLE CONTROL REGISTER 1 (CONTINUED)**

bit 1 **MI2C1IE:** Master I2C1 Event Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 0 **SI2C1IE:** Slave I2C1 Event Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

**Note 1:** If an external interrupt is enabled, the interrupt input must also be configured to an available RPn or RPIn pin. See **Section 10.4 "Peripheral Pin Select"** for more information.


© 2009 Microchip Technology Inc.

**REGISTER 7-14: IEC3: INTERRUPT ENABLE CONTROL REGISTER 3**

| U-0 | R/W-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | RTCIE | — | — | — | — | — | — |
| bit 15 | | | | | | | bit 8 |

| U-0 | R/W-0 | R/W-0 | U-0 | U-0 | R/W-0 | R/W-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | INT4IE(1) | INT3IE(1) | — | — | MI2C2IE | SI2C2IE | — |
| bit 7 | | | | | | | bit 0 |

| **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 |

bit 15 **Unimplemented:** Read as '0'

bit 14 **RTCIE:** Real-Time Clock/Calendar Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 13-7 **Unimplemented:** Read as '0'

bit 6 **INT4IE:** External Interrupt 4 Enable bit(1)
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 5 **INT3IE:** External Interrupt 3 Enable bit(1)
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 4-3 **Unimplemented:** Read as '0'

bit 2 **MI2C2IE:** Master I2C2 Event Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 1 **SI2C2IE:** Slave I2C2 Event Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

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

**Note 1:** If an external interrupt is enabled, the interrupt input must also be configured to an available RPn or RPIn pin. See **Section 10.4 "Peripheral Pin Select"** for more information.


© 2009 Microchip Technology Inc.

## 7.4 Interrupt Setup Procedures

### 7.4.1 INITIALIZATION

To configure an interrupt source:

1. Set the NSTDIS Control bit (INTCON1<15>) if nested interrupts are not desired.
2. Select the user-assigned priority level for the interrupt source by writing the control bits in the appropriate IPCx register. The priority level will depend on the specific application and type of interrupt source. If multiple priority levels are not desired, the IPCx register control bits for all enabled interrupt sources may be programmed to the same non-zero value.

| **Note:** | At a device Reset, the IPCx registers are initialized, such that all user interrupt sources are assigned to priority level 4. |
|---|---|

3. Clear the interrupt flag status bit associated with the peripheral in the associated IFSx register.
4. Enable the interrupt source by setting the interrupt enable control bit associated with the source in the appropriate IECx register.

### 7.4.2 INTERRUPT SERVICE ROUTINE

The method that is used to declare an ISR and initialize the IVT with the correct vector address will depend on the programming language (i.e., 'C' or assembler) and the language development toolsuite that is used to develop the application. In general, the user must clear the interrupt flag in the appropriate IFSx register for the source of the interrupt that the ISR handles. Otherwise, the ISR will be re-entered immediately after exiting the routine. If the ISR is coded in assembly language, it must be terminated using a `RETFIE` instruction to unstack the saved PC value, SRL value and old CPU priority level.

### 7.4.3 TRAP SERVICE ROUTINE

A Trap Service Routine (TSR) is coded like an ISR, except that the appropriate trap status flag in the INTCON1 register must be cleared to avoid re-entry into the TSR.

### 7.4.4 INTERRUPT DISABLE

All user interrupts can be disabled using the following procedure:

1. Push the current SR value onto the software stack using the `PUSH` instruction.
2. Force the CPU to priority level 7 by inclusive ORing the value E0h with SRL.

To enable user interrupts, the `POP` instruction may be used to restore the previous SR value.

Note that only user interrupts with a priority level of 7 or less can be disabled. Trap sources (level 8-15) cannot be disabled.

The `DISI` instruction provides a convenient way to disable interrupts of priority levels 1-6 for a fixed period of time. Level 7 interrupt sources are not disabled by the `DISI` instruction.

### 10.1.1 OPEN-DRAIN CONFIGURATION

In addition to the PORT, LAT and TRIS registers for data control, each port pin can also be individually configured for either digital or open-drain output. This is controlled by the Open-Drain Control register, ODCx, associated with each port. Setting any of the bits configures the corresponding pin to act as an open-drain output.

The open-drain feature allows the generation of outputs higher than VDD (e.g., 5V) on any desired digital only pins by using external pull-up resistors. The maximum open-drain voltage allowed is the same as the maximum VIH specification.

## 10.2 Configuring Analog Port Pins

The AD1PCFGL and TRIS registers control the operation of the A/D port pins. Setting a port pin as an analog input also requires that the corresponding TRIS bit be set. If the TRIS bit is cleared (output), the digital output level (VOH or VOL) will be converted.

When reading the PORT register, all pins configured as analog input channels will read as cleared (a low level).

Pins configured as digital inputs will not convert an analog input. Analog levels on any pin that is defined as a digital input (including the ANx pins) may cause the input buffer to consume current that exceeds the device specifications.

### 10.2.1 I/O PORT WRITE/READ TIMING

One instruction cycle is required between a port direction change or port write operation and a read operation of the same port. Typically, this instruction would be a NOP.

### 10.2.2 ANALOG INPUT PINS AND VOLTAGE CONSIDERATIONS

The voltage tolerance of pins used as device inputs is dependent on the pin's input function. Pins that are used as digital only inputs are able to handle DC voltages up to 5.5V, a level typical for digital logic circuits. In contrast, pins that also have analog input functions of any kind can only tolerate voltages up to VDD. Voltage excursions beyond VDD on these pins are always to be avoided. Table 10-1 summarizes the input capabilities. Refer to **Section 29.1 "DC Characteristics"** for more details.

| | |
|---|---|
| **Note:** | For easy identification, the pin diagrams at the beginning of the data sheet also indicate 5.5V tolerant pins with dark grey shading. |

**TABLE 10-1: INPUT VOLTAGE LEVELS(1)**

| Port or Pin | Tolerated Input | Description |
|---|---|---|
| PORTA<10:9> | VDD | Only VDD input levels tolerated. |
| PORTB<15:0> | | |
| PORTC<15:12> | | |
| PORTD<7:6> | | |
| PORTF<0> | | |
| PORTG<9:6>, PORTG<3:2> | | |
| PORTA<15:14>, PORTA<7:0> | 5.5V | Tolerates input levels above VDD, useful for most standard logic. |
| PORTC<4:1> | | |
| PORTD<15:8>, PORTD<5:0> | | |
| PORTE<9:0> | | |
| PORTF<13:12>, PORTF<8>, PORTF<5:1> | | |
| PORTG<15:12>, PORTG<1:0> | | |

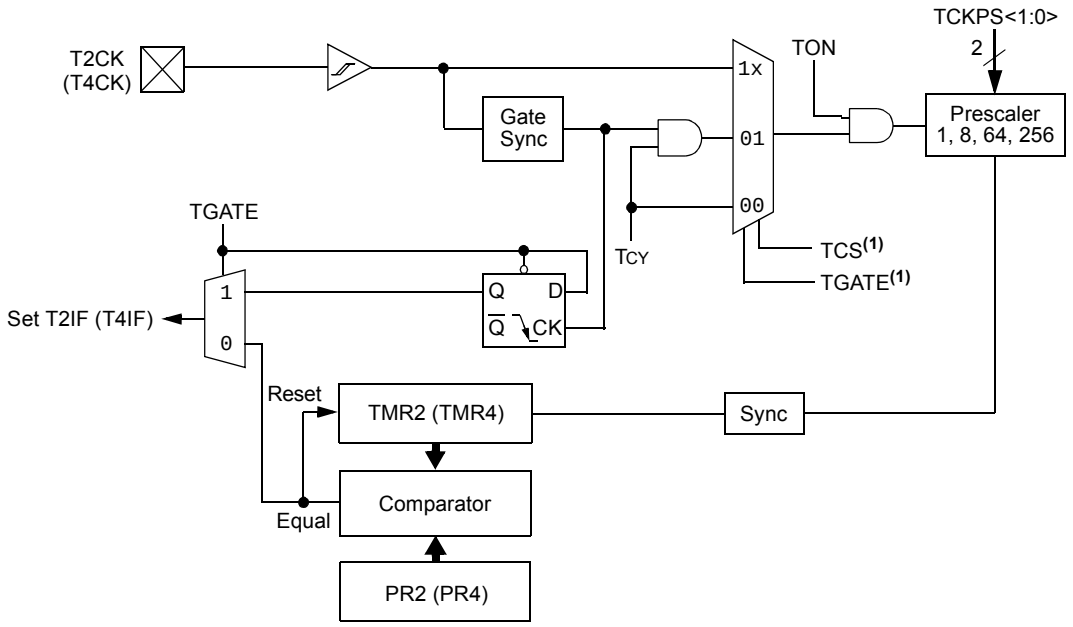
**Note 1:** Not all port pins shown here are implemented on 64-pin and 80-pin devices. Refer to **Section 1.0 "Device Overview"** to confirm which ports are available in specific devices.

**EXAMPLE 10-1: PORT WRITE/READ EXAMPLE**

```
MOV   0xFF00, W0          ; Configure PORTB<15:8> as inputs
MOV   W0, TRISBB          ; and PORTB<7:0> as outputs
NOP                       ; Delay 1 cycle
BTSS  PORTB, #13          ; Next Instruction
```

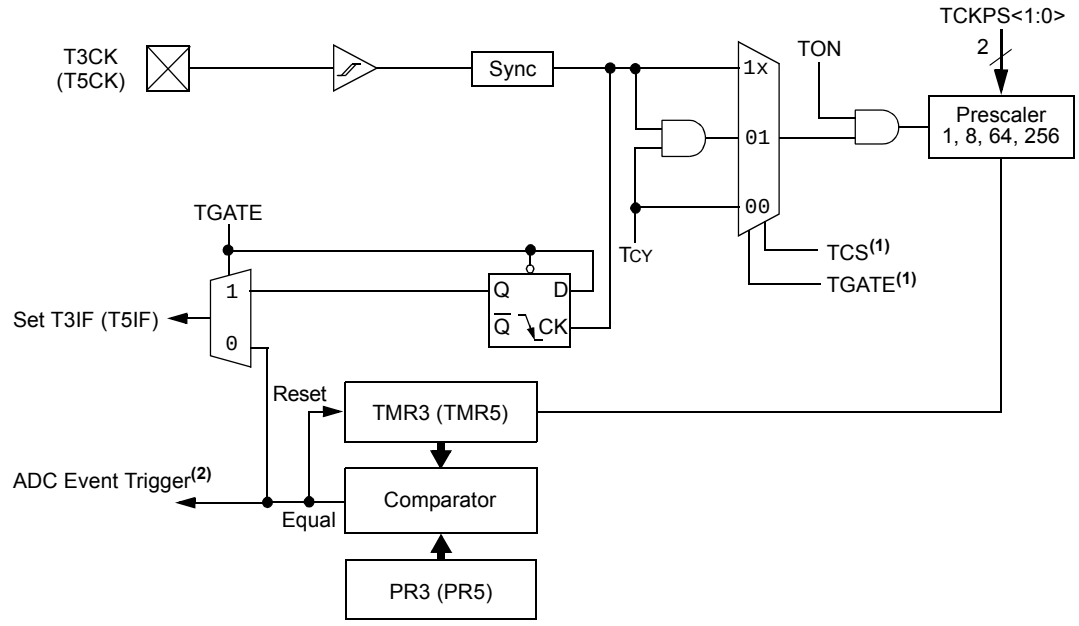

© 2009 Microchip Technology Inc.

**FIGURE 12-2: TIMER2 AND TIMER4 (16-BIT SYNCHRONOUS) BLOCK DIAGRAM**




**Note 1:** The timer clock input must be assigned to an available RPn pin before use. Please see **Section 10.4 "Peripheral Pin Select"** for more information.

**FIGURE 12-3: TIMER3 AND TIMER5 (16-BIT ASYNCHRONOUS) BLOCK DIAGRAM**



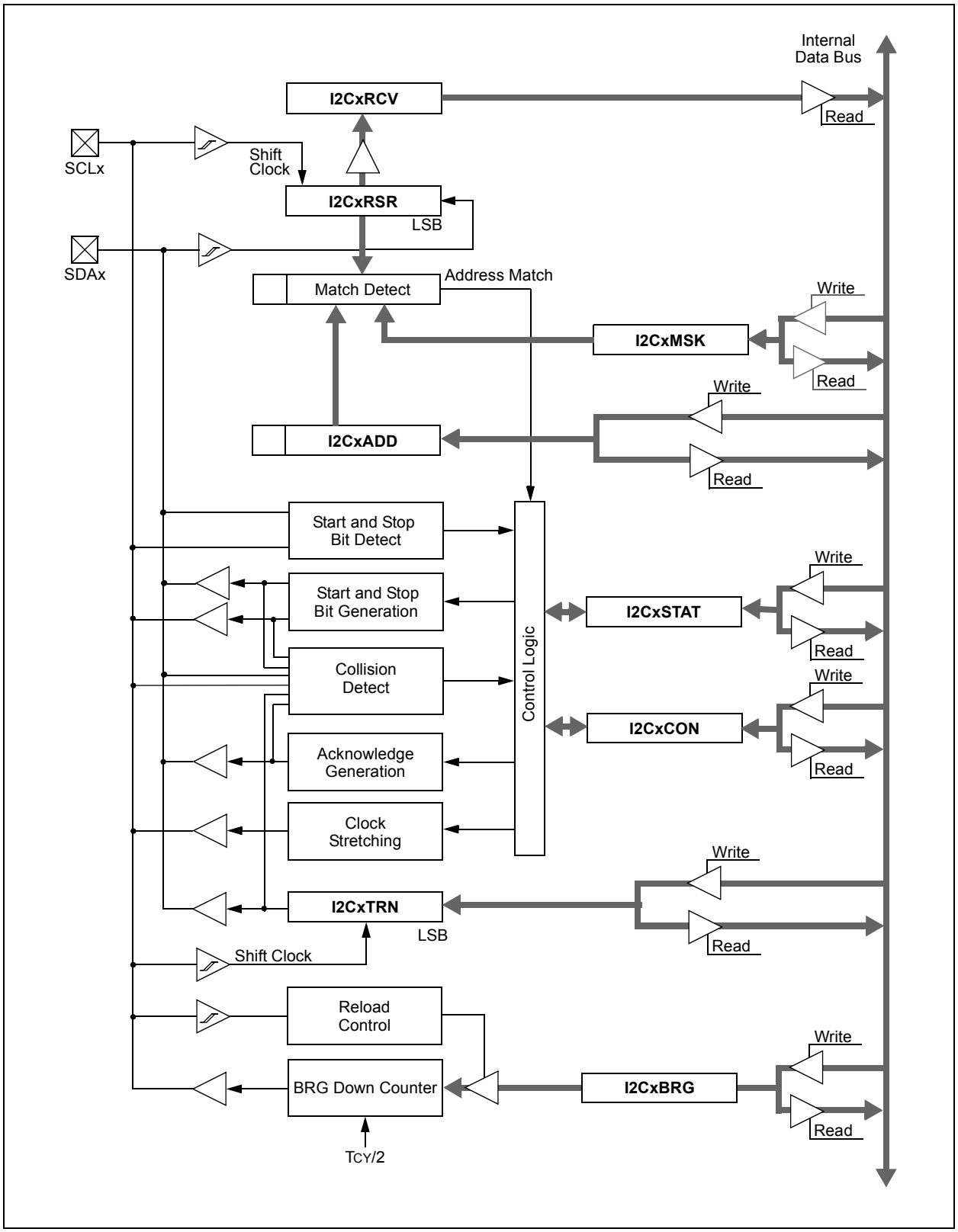

**Note 1:** The timer clock input must be assigned to an available RPn pin before use. Please see **Section 10.4 "Peripheral Pin Select"** for more information.

**2:** The ADC Event Trigger is available only on Timer3.

**NOTES:**


© 2009 Microchip Technology Inc.

**FIGURE 16-1: I²C™ BLOCK DIAGRAM**





© 2009 Microchip Technology Inc.

**NOTES:**


© 2009 Microchip Technology Inc.

**REGISTER 18-9: U1ADDR: USB ADDRESS REGISTER**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 15 | | | | | | | bit 8 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| LSPDEN(1) | ADDR6 | ADDR5 | ADDR4 | ADDR3 | ADDR2 | ADDR1 | ADDR0 |
| bit 7 | | | | | | | bit 0 |

| **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 |

bit 15-8 **Unimplemented:** Read as '0'

bit 7 **LSPDEN:** Low-Speed Enable Indicator bit(1)
1 = USB module operates at low speed
0 = USB module operates at full speed

bit 6-0 **ADDR<6:0>:** USB Device Address bits

**Note 1:** Host mode only. In Device mode, this bit is unimplemented and read as '0'.

**REGISTER 18-10: U1TOK: USB TOKEN REGISTER (HOST MODE ONLY)**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 15 | | | | | | | bit 8 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| PID3 | PID2 | PID1 | PID0 | EP3 | EP2 | EP1 | EP0 |
| bit 7 | | | | | | | bit 0 |

| **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 |

bit 15-8 **Unimplemented:** Read as '0'

bit 7-4 **PID<3:0>:** Token Type Identifier bits
1101 = SETUP (TX) token type transaction(1)
1001 = IN (RX) token type transaction(1)
0001 = OUT (TX) token type transaction(1)

bit 3-0 **EP<3:0>:** Token Command Endpoint Address bits
This value must specify a valid endpoint on the attached device.

**Note 1:** All other combinations are reserved and are not to be used.


© 2009 Microchip Technology Inc.

## 19.0 PARALLEL MASTER PORT (PMP)

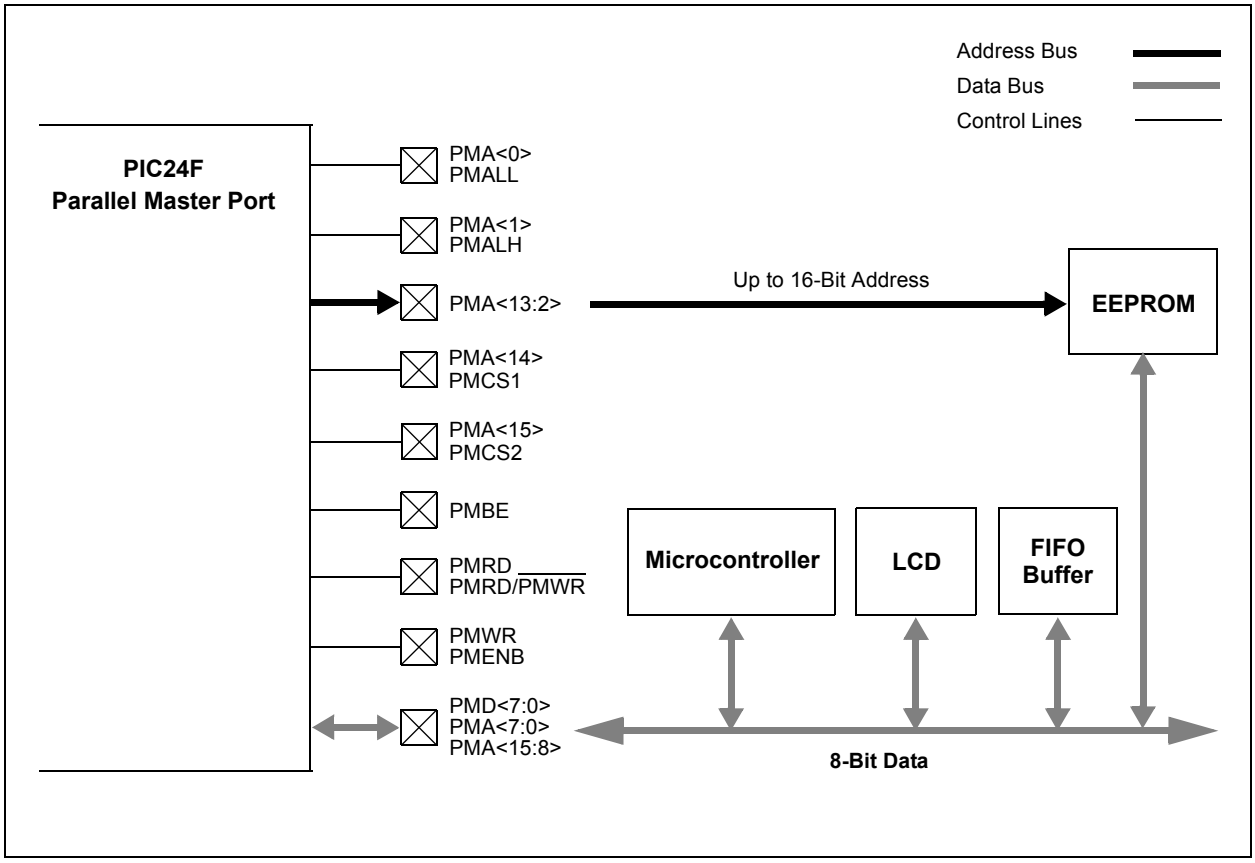
**Note:** This data sheet summarizes the features of this group of PIC24F devices. It is not intended to be a comprehensive reference source. For more information, refer to the *"PIC24F Family Reference Manual"*, **Section 13. "Parallel Master Port (PMP)"** (DS39713).

The Parallel Master Port (PMP) module is a parallel 8-bit I/O module, specifically designed to communicate with a wide variety of parallel devices, such as communication peripherals, LCDs, external memory devices and microcontrollers. Because the interface to parallel peripherals varies significantly, the PMP is highly configurable.

Key features of the PMP module include:

- Up to 16 Programmable Address Lines
- Up to 2 Chip Select Lines
- Programmable Strobe Options:
  - Individual Read and Write Strobes or;
  - Read/$\overline{\text{Write}}$ Strobe with Enable Strobe
- Address Auto-Increment/Auto-Decrement
- Programmable Address/Data Multiplexing
- Programmable Polarity on Control Signals
- Legacy Parallel Slave Port Support
- Enhanced Parallel Slave Support:
  - Address Support
  - 4-Byte Deep Auto-Incrementing Buffer
- Programmable Wait States
- Selectable Input Voltage Levels

**FIGURE 19-1: PMP MODULE OVERVIEW**



**REGISTER 19-1: PMCON: PARALLEL PORT CONTROL REGISTER (CONTINUED)**

bit 2 **BEP:** Byte Enable Polarity bit
1 = Byte enable active-high (PMBE)
0 = Byte enable active-low ($\overline{\text{PMBE}}$)

bit 1 **WRSP:** Write Strobe Polarity bit
For Slave modes and Master mode 2 (PMMODE<9:8> = 00,01,10):
1 = Write strobe active-high (PMWR)
0 = Write strobe active-low ($\overline{\text{PMWR}}$)
For Master mode 1 (PMMODE<9:8> = 11):
1 = Enable strobe active-high (PMENB)
0 = Enable strobe active-low ($\overline{\text{PMENB}}$)

bit 0 **RDSP:** Read Strobe Polarity bit
For Slave modes and Master mode 2 (PMMODE<9:8> = 00,01,10):
1 = Read strobe active-high (PMRD)
0 = Read strobe active-low ($\overline{\text{PMRD}}$)
For Master mode 1 (PMMODE<9:8> = 11):
1 = Read/write strobe active-high (PMRD/$\overline{\text{PMWR}}$)
0 = Read/write strobe active-low ($\overline{\text{PMRD}}$/$\overline{\text{PMWR}}$)

**Note 1:** These bits have no effect when their corresponding pins are used as address lines.

**REGISTER 26-2: CW2: FLASH CONFIGURATION WORD 2 (CONTINUED)**

bit 4 **IOL1WAY:** IOLOCK One-Way Set Enable bit
1 = The IOLOCK bit (OSCCON<6>)can be set once, provided the unlock sequence has been completed. Once set, the Peripheral Pin Select registers cannot be written to a second time.
0 = The IOLOCK bit can be set and cleared as needed, provided the unlock sequence has been completed

bit 3 **DISUVREG:** Internal USB 3.3V Regulator Disable bit
1 = Regulator is disabled
0 = Regulator is enabled

bit 2 **Reserved:** Always maintain as '1'

bit 1-0 **POSCMD<1:0>:** Primary Oscillator Configuration bits
11 = Primary Oscillator disabled
10 = HS Oscillator mode selected
01 = XT Oscillator mode selected
00 = EC Oscillator mode selected

**REGISTER 26-3: CW3: FLASH CONFIGURATION WORD 3**

| U-1 | U-1 | U-1 | U-1 | U-1 | U-1 | U-1 | U-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 23 | | | | | | | bit 16 |

| R/PO-1 | R/PO-1 | R/PO-1 | U-1 | U-1 | U-1 | U-1 | U-1 |
|---|---|---|---|---|---|---|---|
| WPEND | WPCFG | WPDIS | — | — | — | — | — |
| bit 15 | | | | | | | bit 8 |

| R/PO-1 | R/PO-1 | R/PO-1 | R/PO-1 | R/PO-1 | R/PO-1 | R/PO-1 | R/PO-1 |
|---|---|---|---|---|---|---|---|
| WPFP7 | WPFP6 | WPFP5 | WPFP4 | WPFP3 | WPFP2 | WPFP1 | WPFP0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | PO = Program-once bit | U = Unimplemented bit, read as '0' |
| -n = Value when device is unprogrammed | '1' = Bit is set | '0' = Bit is cleared |

bit 23-16 **Unimplemented:** Read as '1'

bit 15 **WPEND:** Segment Write Protection End Page Select bit
1 = Protected code segment lower boundary is at the bottom of program memory (000000h); upper boundary is the code page specified by WPFP<7:0>
0 = Protected code segment upper boundary is at the last page of program memory; lower boundary is the code page specified by WPFP<7:0>

bit 14 **WPCFG:** Configuration Word Code Page Protection Select bit
1 = Last page (at the top of program memory) and Flash Configuration Words are not protected
0 = Last page and Flash Configuration Words are code protected

bit 13 **WPDIS:** Segment Write Protection Disable bit
1 = Segmented code protection disabled
0 = Segmented code protection enabled; protected segment defined by WPEND, WPCFG and WPFPx Configuration bits

bit 12-8 **Unimplemented:** Read as '1'

bit 7-0 **WPFP<7:0>:** Protected Code Segment Boundary Page bits
Designates the 512-word program code page that is the boundary of the protected code segment, starting with Page 0 at the bottom of program memory.
If WPEND = 1:
Last address of designated code page is the upper boundary of the segment.
If WPEND = '0':
First address of designated code page is the lower boundary of the segment.

### 26.2.2 ON-CHIP REGULATOR AND POR

When the voltage regulator is enabled, it takes approximately 10 μs for it to generate output. During this time, designated as TVREG, code execution is disabled. TVREG is applied every time the device resumes operation after any power-down, including Sleep mode. The length of TVREG is determined by the PMSLP bit (RCON<8>), as described in **Section 26.2.5 "Voltage Regulator Standby Mode"**.

If the regulator is disabled, a separate Power-up Timer (PWRT) is automatically enabled. The PWRT adds a fixed delay of 64 ms nominal delay at device start-up (POR or BOR only). When waking up from Sleep with the regulator disabled, the PMSLP bit determines the wake-up time. When operating with the regulator disabled, setting PMSLP can decrease the device wake-up time.

### 26.2.3 ON-CHIP REGULATOR AND BOR

When the on-chip regulator is enabled, PIC24FJ256GB110 family devices also have a simple brown-out capability. If the voltage supplied to the regulator is inadequate to maintain the tracking level, the regulator Reset circuitry will generate a Brown-out Reset. This event is captured by the BOR flag bit (RCON<1>). The brown-out voltage specifications are provided in the *"PIC24FJ Family Reference Manual"*, **Section 7. "Reset"** (DS39712).

### 26.2.4 POWER-UP REQUIREMENTS

The on-chip regulator is designed to meet the power-up requirements for the device. If the application does not use the regulator, then strict power-up conditions must be adhered to. While powering up, VDDCORE must never exceed VDD by 0.3 volts.

| | |
|---|---|
| **Note:** | For more information, see **Section 29.0 "Electrical Characteristics"**. |

### 26.2.5 VOLTAGE REGULATOR STANDBY MODE

When enabled, the on-chip regulator always consumes a small incremental amount of current over IDD/IPD, including when the device is in Sleep mode, even though the core digital logic does not require power. To provide additional savings in applications where power resources are critical, the regulator automatically disables itself whenever the device goes into Sleep mode. This feature is controlled by the PMSLP bit (RCON<8>). By default, the bit is cleared, which removes power from the Flash program memory and thus enables Standby mode. When waking up from Standby mode, the regulator must wait for TVREG to expire before wake-up. This extra time is needed to ensure that the regulator can source enough current to power the Flash memory.

For applications which require a faster wake-up time, it is possible to disable regulator Standby mode. The PMSLP bit can be set to turn off Standby mode so that the Flash stays powered when in Sleep mode and the device can wake-up without waiting for TVREG. When PMSLP is set, the power consumption while in Sleep mode, will be approximately 40 μA higher than power consumption when the regulator is allowed to enter Standby mode.

## 26.3 Watchdog Timer (WDT)

For PIC24FJ256GB110 family devices, the WDT is driven by the LPRC Oscillator. When the WDT is enabled, the clock source is also enabled.

The nominal WDT clock source from LPRC is 31 kHz. This feeds a prescaler that can be configured for either 5-bit (divide-by-32) or 7-bit (divide-by-128) operation. The prescaler is set by the FWPSA Configuration bit. With a 31 kHz input, the prescaler yields a nominal WDT time-out period (TWDT) of 1 ms in 5-bit mode, or 4 ms in 7-bit mode.

A variable postscaler divides down the WDT prescaler output and allows for a wide range of time-out periods. The postscaler is controlled by the WDTPS<3:0> Configuration bits (CW1<3:0>), which allow the selection of a total of 16 settings, from 1:1 to 1:32,768. Using the prescaler and postscaler, time-out periods ranging from 1 ms to 131 seconds can be achieved.

The WDT, prescaler and postscaler are reset:

- On any device Reset
- On the completion of a clock switch, whether invoked by software (i.e., setting the OSWEN bit after changing the NOSC bits) or by hardware (i.e., Fail-Safe Clock Monitor)
- When a PWRSAV instruction is executed (i.e., Sleep or Idle mode is entered)
- When the device exits Sleep or Idle mode to resume normal operation
- By a CLRWDT instruction during normal execution

If the WDT is enabled, it will continue to run during Sleep or Idle modes. When the WDT time-out occurs, the device will wake the device and code execution will continue from where the PWRSAV instruction was executed. The corresponding SLEEP or IDLE bits (RCON<3:2>) will need to be cleared in software after the device wakes up.

The WDT Flag bit, WDTO (RCON<4>), is not automatically cleared following a WDT time-out. To detect subsequent WDT events, the flag must be cleared in software.

| | |
|---|---|
| **Note:** | The CLRWDT and PWRSAV instructions clear the prescaler and postscaler counts when executed. |


© 2009 Microchip Technology Inc.

## 27.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers and dsPIC® digital signal controllers are supported with a full range of software and hardware development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Compilers/Assemblers/Linkers
  - MPLAB C Compiler for Various Device Families
  - HI-TECH C for Various Device Families
  - MPASM™ Assembler
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
  - MPLAB Assembler/Linker/Librarian for Various Device Families
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers
  - MPLAB ICD 3
  - PICkit™ 3 Debug Express
- Device Programmers
  - PICkit™ 2 Programmer
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards, Evaluation Kits, and Starter Kits

### 27.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8/16/32-bit microcontroller market. The MPLAB IDE is a Windows® operating system-based application that contains:

- A single graphical interface to all debugging tools
  - Simulator
  - Programmer (sold separately)
  - In-Circuit Emulator (sold separately)
  - In-Circuit Debugger (sold separately)
- A full-featured editor with color-coded context
- A multiple project manager
- Customizable data windows with direct edit of contents
- High-level source code debugging
- Mouse over variable inspection
- Drag and drop variables from source to watch windows
- Extensive on-line help
- Integration of select third party tools, such as IAR C Compilers

The MPLAB IDE allows you to:

- Edit your source files (either C or assembly)
- One-touch compile or assemble, and download to emulator and simulator tools (automatically updates all project information)
- Debug using:
  - Source files (C or assembly)
  - Mixed C and assembly
  - Machine code

MPLAB IDE supports multiple debugging tools in a single development paradigm, from the cost-effective simulators, through low-cost in-circuit debuggers, to full-featured emulators. This eliminates the learning curve when upgrading to tools with increased flexibility and power.

## 28.0 INSTRUCTION SET SUMMARY

| **Note:** | This chapter is a brief summary of the PIC24F instruction set architecture, and is not intended to be a comprehensive reference source. |
|---|---|

The PIC24F instruction set adds many enhancements to the previous PIC® MCU instruction sets, while maintaining an easy migration from previous PIC MCU instruction sets. Most instructions are a single program memory word. Only three instructions require two program memory locations.

Each single-word instruction is a 24-bit word divided into an 8-bit opcode, which specifies the instruction type and one or more operands, which further specify the operation of the instruction. The instruction set is highly orthogonal and is grouped into four basic categories:

- Word or byte-oriented operations
- Bit-oriented operations
- Literal operations
- Control operations

Table 28-1 shows the general symbols used in describing the instructions. The PIC24F instruction set summary in Table 28-2 lists all the instructions, along with the status flags affected by each instruction.

Most word or byte-oriented W register instructions (including barrel shift instructions) have three operands:

- The first source operand which is typically a register ‘Wb’ without any address modifier
- The second source operand which is typically a register ‘Ws’ with or without an address modifier
- The destination of the result which is typically a register ‘Wd’ with or without an address modifier

However, word or byte-oriented file register instructions have two operands:

- The file register specified by the value, ‘f’
- The destination, which could either be the file register ‘f’ or the W0 register, which is denoted as ‘WREG’

Most bit-oriented instructions (including simple rotate/shift instructions) have two operands:

- The W register (with or without an address modifier) or file register (specified by the value of ‘Ws’ or ‘f’)
- The bit in the W register or file register (specified by a literal value or indirectly by the contents of register, ‘Wb’)

The literal instructions that involve data movement may use some of the following operands:

- A literal value to be loaded into a W register or file register (specified by the value of ‘k’)
- The W register or file register where the literal value is to be loaded (specified by ‘Wb’ or ‘f’)

However, literal instructions that involve arithmetic or logical operations use some of the following operands:

- The first source operand which is a register ‘Wb’ without any address modifier
- The second source operand which is a literal value
- The destination of the result (only if not the same as the first source operand) which is typically a register ‘Wd’ with or without an address modifier

The control instructions may use some of the following operands:

- A program memory address
- The mode of the table read and table write instructions

All instructions are a single word, except for certain double-word instructions, which were made double-word instructions so that all the required information is available in these 48 bits. In the second word, the 8 MSbs are ‘0’s. If this second word is executed as an instruction (by itself), it will execute as a NOP.

Most single-word instructions are executed in a single instruction cycle, unless a conditional test is true or the program counter is changed as a result of the instruction. In these cases, the execution takes two instruction cycles, with the additional instruction cycle(s) executed as a NOP. Notable exceptions are the BRA (unconditional/computed branch), indirect CALL/GOTO, all table reads and writes, and RETURN/RETFIE instructions, which are single-word instructions but take two or three cycles.

Certain instructions that involve skipping over the subsequent instruction require either two or three cycles if the skip is performed, depending on whether the instruction being skipped is a single-word or two-word instruction. Moreover, double-word moves require two cycles. The double-word instructions execute in two instruction cycles.