# PIC18F1220/1320

## 18/20/28-Pin High-Performance, Enhanced Flash MCUs with 10-bit A/D

### Low-Power Features

- Power Managed modes:
  - Run: CPU on, peripherals on
  - Idle: CPU off, peripherals on
  - Sleep: CPU off, peripherals off
- Power Consumption modes:
  - PRI_RUN: 150 μA, 1 MHz, 2V
  - PRI_IDLE: 37 μA, 1 MHz, 2V
  - SEC_RUN: 14 μA, 32 kHz, 2V
  - SEC_IDLE: 5.8 μA, 32 kHz, 2V
  - RC_RUN: 110 μA, 1 MHz, 2V
  - RC_IDLE: 52 μA, 1 MHz, 2V
  - Sleep: 0.1 μA, 1 MHz, 2V
- Timer1 Oscillator: 1.1 μA, 32 kHz, 2V
- Watchdog Timer: 2.1 μA
- Two-Speed Oscillator Start-up

### Oscillators

- Four Crystal modes:
  - LP, XT, HS: up to 25 MHz
  - HSPLL: 4-10 MHz (16-40 MHz internal)
- Two External RC modes, up to 4 MHz
- Two External Clock modes, up to 40 MHz
- Internal Oscillator Block:
  - 8 user-selectable frequencies: 31 kHz, 125 kHz, 250 kHz, 500 kHz, 1 MHz, 2 MHz, 4 MHz, 8 MHz
  - 125 kHz to 8 MHz calibrated to 1%
  - Two modes select one or two I/O pins
  - OSCTUNE – Allows user to shift frequency
- Secondary Oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor
  - Allows for safe shutdown if peripheral clock stops

### Peripheral Highlights

- High Current Sink/Source 25 mA/25 mA
- Three External Interrupts
- Enhanced Capture/Compare/PWM (ECCP) module:
  - One, two or four PWM outputs
  - Selectable polarity
  - Programmable dead time
  - Auto-Shutdown and Auto-Restart
  - Capture is 16-bit, max resolution 6.25 ns (TCY/16)
  - Compare is 16-bit, max resolution 100 ns (TCY)
- Compatible 10-bit, up to 13-Channel Analog-to-Digital Converter module (A/D) with Programmable Acquisition Time
- Enhanced USART module:
  - Supports RS-485, RS-232 and LIN 1.2
  - Auto-Wake-up on Start bit
  - Auto-Baud Detect

### Special Microcontroller Features

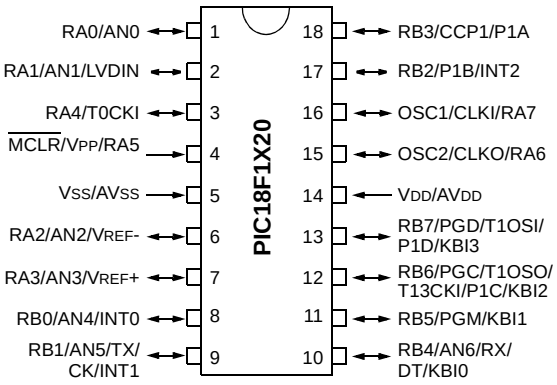
- 100,000 Erase/Write Cycle Enhanced Flash Program Memory, typical
- 1,000,000 Erase/Write Cycle Data EEPROM Memory, typical
- Flash/Data EEPROM Retention: > 40 years
- Self-Programmable under Software Control
- Priority Levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
  - 2% stability over VDD and Temperature
- Single-Supply 5V In-Circuit Serial Programming™ (ICSP™) via Two Pins
- In-Circuit Debug (ICD) via Two Pins
- Wide Operating Voltage Range: 2.0V to 5.5V

| Device | Program Memory | | Data Memory | | I/O | 10-bit A/D (ch) | ECCP (PWM) | EUSART | Timers 8/16-bit |
|---|---|---|---|---|---|---|---|---|---|
| | Flash (bytes) | # Single-Word Instructions | SRAM (bytes) | EEPROM (bytes) | | | | | |
| PIC18F1220 | 4K | 2048 | 256 | 256 | 16 | 7 | 1 | Y | 1/3 |
| PIC18F1320 | 8K | 4096 | 256 | 256 | 16 | 7 | 1 | Y | 1/3 |

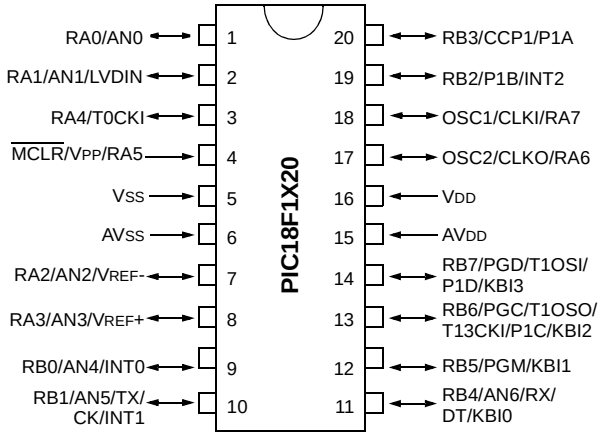
© 2002-2015 Microchip Technology Inc.

## Pin Diagrams

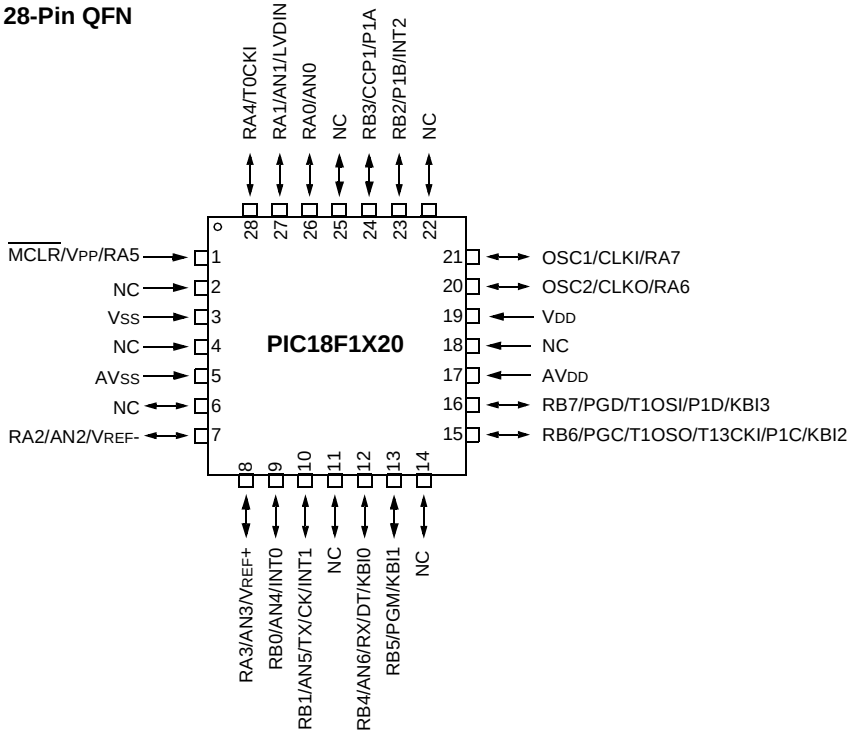
### 18-Pin PDIP, SOIC




### 20-Pin SSOP




### 28-Pin QFN





© 2002-2015 Microchip Technology Inc.

## Table of Contents

1.0 Device Overview ..... 5
2.0 Oscillator Configurations ..... 10
3.0 Power Managed Modes ..... 18
4.0 Reset ..... 31
5.0 Memory Organization ..... 39
6.0 Flash Program Memory ..... 55
7.0 Data EEPROM Memory ..... 64
8.0 8 x 8 Hardware Multiplier ..... 68
9.0 Interrupts ..... 70
10.0 I/O Ports ..... 83
11.0 Timer0 Module ..... 95
12.0 Timer1 Module ..... 98
13.0 Timer2 Module ..... 104
14.0 Timer3 Module ..... 106
15.0 Enhanced Capture/Compare/PWM (ECCP) Module ..... 109
16.0 Enhanced Addressable Universal Synchronous Asynchronous Receiver Transmitter (EUSART) ..... 126
17.0 10-Bit Analog-to-Digital Converter (A/D) Module ..... 149
18.0 Low-Voltage Detect ..... 160
19.0 Special Features of the CPU ..... 165
20.0 Instruction Set Summary ..... 184
21.0 Development Support ..... 226
22.0 Electrical Characteristics ..... 230
23.0 DC and AC Characteristics Graphs and Tables ..... 262
24.0 Packaging Information ..... 280
Appendix A: Revision History ..... 290
Appendix B: Device Differences ..... 290
Appendix C: Conversion Considerations ..... 291
Appendix D: Migration from Baseline to Enhanced Devices ..... 291
Appendix E: Migration from Mid-Range to Enhanced Devices ..... 292
Appendix F: Migration from High-End to Enhanced Devices ..... 292
The Microchip Web Site ..... 293
Customer Change Notification Service ..... 293
Customer Support ..... 293
PIC18F1220/1320 Product Identification System ..... 294

© 2002-2015 Microchip Technology Inc. 

## 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**. 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., DS30000000A is version A of document DS30000000).

### 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** to receive the most current information on all of our products.


© 2002-2015 Microchip Technology Inc.

# 1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC18F1220
- PIC18F1320

This family offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price – with the addition of high endurance Enhanced Flash program memory. On top of these features, the PIC18F1220/1320 family introduces design enhancements that make these microcontrollers a logical choice for many high-performance, power sensitive applications.

## 1.1 New Core Features

### 1.1.1 POWER MODES

All of the devices in the PIC18F1220/1320 family incorporate a range of features that can significantly reduce power consumption during operation. Key items include:

- **Alternate Run Modes:** By clocking the controller from the Timer1 source or the internal oscillator block, power consumption during code execution can be reduced by as much as 90%.
- **Multiple Idle Modes:** The controller can also run with its CPU core disabled, but the peripherals are still active. In these states, power consumption can be reduced even further, to as little as 4% of normal operation requirements.
- **On-the-fly Mode Switching:** The power managed modes are invoked by user code during operation, allowing the user to incorporate power-saving ideas into their application's software design.
- **Lower Consumption in Key Modules:** The power requirements for both Timer1 and the Watchdog Timer have been reduced by up to 80%, with typical values of 1.1 and 2.1 μA, respectively.

### 1.1.2 MULTIPLE OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F1220/1320 family offer nine different oscillator options, allowing users a wide range of choices in developing application hardware. These include:

- Four Crystal modes, using crystals or ceramic resonators.
- Two External Clock modes, offering the option of using two pins (oscillator input and a divide-by-4 clock output), or one pin (oscillator input, with the second pin reassigned as general I/O).
- Two External RC Oscillator modes, with the same pin options as the External Clock modes.
- An internal oscillator block, which provides an 8 MHz clock (±2% accuracy) and an INTRC source (approximately 31 kHz, stable over temperature and $V_{DD}$), as well as a range of six user-selectable clock frequencies (from 125 kHz to 4 MHz) for a total of 8 clock frequencies.

Besides its availability as a clock source, the internal oscillator block provides a stable reference source that gives the family additional features for robust operation:

- **Fail-Safe Clock Monitor:** This option constantly monitors the main clock source against a reference signal provided by the internal oscillator. If a clock failure occurs, the controller is switched to the internal oscillator block, allowing for continued low-speed operation, or a safe application shutdown.
- **Two-Speed Start-up:** This option allows the internal oscillator to serve as the clock source from Power-on Reset, or wake-up from Sleep mode, until the primary clock source is available. This allows for code execution during what would otherwise be the clock start-up interval and can even allow an application to perform routine background activities and return to Sleep without returning to full-power operation.

## 1.2 Other Special Features

- **Memory Endurance:** The Enhanced Flash cells for both program memory and data EEPROM are rated to last for many thousands of erase/write cycles – up to 100,000 for program memory and 1,000,000 for EEPROM. Data retention without refresh is conservatively estimated to be greater than 40 years.
- **Self-programmability:** These devices can write to their own program memory spaces under internal software control. By using a bootloader routine located in the protected Boot Block at the top of program memory, it becomes possible to create an application that can update itself in the field.
- **Enhanced CCP module:** In PWM mode, this module provides 1, 2 or 4 modulated outputs for controlling half-bridge and full-bridge drivers. Other features include auto-shutdown, for disabling PWM outputs on interrupt or other select conditions and auto-restart, to reactivate outputs once the condition has cleared.
- **Enhanced USART:** This serial communication module features automatic wake-up on Start bit and automatic baud rate detection and supports RS-232, RS-485 and LIN 1.2 protocols, making it ideally suited for use in Local Interconnect Network (LIN) bus applications.
- **10-bit A/D Converter:** This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated without waiting for a sampling period and thus, reduce code overhead.
- **Extended Watchdog Timer (WDT):** This enhanced version incorporates a 16-bit prescaler, allowing a time-out range from 4 ms to over two minutes that is stable across operating voltage and temperature.

© 2002-2015 Microchip Technology Inc.

## 1.3 Details on Individual Family Members

Devices in the PIC18F1220/1320 family are available in 18-pin, 20-pin and 28-pin packages. A block diagram for this device family is shown in Figure 1-1.

The devices are differentiated from each other only in the amount of on-chip Flash program memory (4 Kbytes for the PIC18F1220 device, 8 Kbytes for the PIC18F1320 device). These and other features are summarized in Table 1-1.

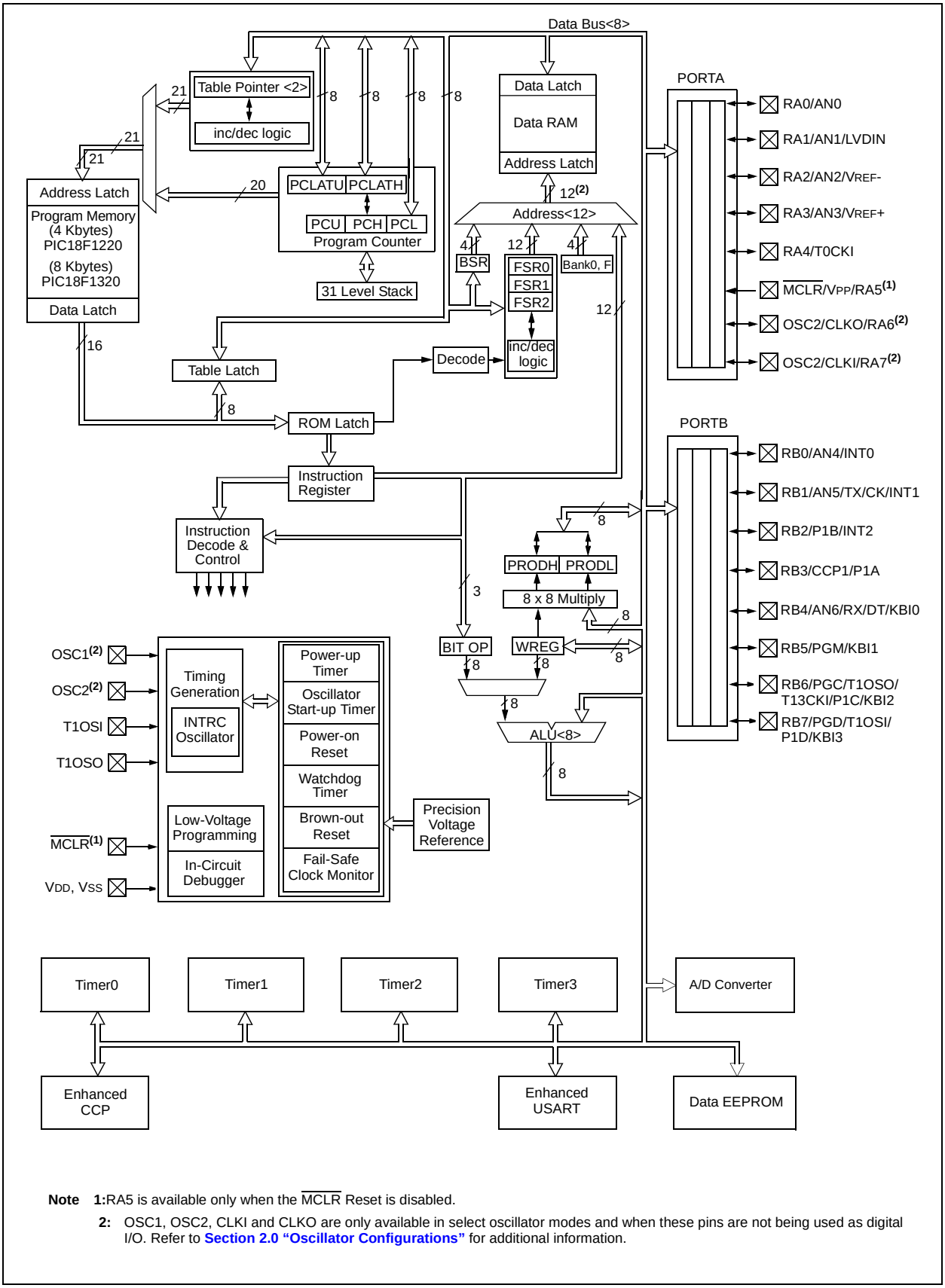
A block diagram of the PIC18F1220/1320 device architecture is provided in Figure 1-1. The pinouts for this device family are listed in Table 1-2.

**TABLE 1-1: DEVICE FEATURES**

| Features | PIC18F1220 | PIC18F1320 |
|---|---|---|
| Operating Frequency | DC – 40 MHz | DC – 40 MHz |
| Program Memory (Bytes) | 4096 | 8192 |
| Program Memory (Instructions) | 2048 | 4096 |
| Data Memory (Bytes) | 256 | 256 |
| Data EEPROM Memory (Bytes) | 256 | 256 |
| Interrupt Sources | 15 | 15 |
| I/O Ports | Ports A, B | Ports A, B |
| Timers | 4 | 4 |
| Enhanced Capture/Compare/PWM Modules | 1 | 1 |
| Serial Communications | Enhanced USART | Enhanced USART |
| 10-bit Analog-to-Digital Module | 7 input channels | 7 input channels |
| Resets (and Delays) | POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT | POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT |
| Programmable Low-Voltage Detect | Yes | Yes |
| Programmable Brown-out Reset | Yes | Yes |
| Instruction Set | 75 Instructions | 75 Instructions |
| Packages | 18-pin SDIP<br>18-pin SOIC<br>20-pin SSOP<br>28-pin QFN | 18-pin SDIP<br>18-pin SOIC<br>20-pin SSOP<br>28-pin QFN |


© 2002-2015 Microchip Technology Inc.

**FIGURE 1-1: PIC18F1220/1320 BLOCK DIAGRAM**




**Note 1:** RA5 is available only when the $\overline{\text{MCLR}}$ Reset is disabled.

**2:** OSC1, OSC2, CLKI and CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 2.0 "Oscillator Configurations"** for additional information.

**TABLE 1-2: PIC18F1220/1320 PINOUT I/O DESCRIPTIONS**

| Pin Name | Pin Number | | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|---|
| | PDIP/ SOIC | SSOP | QFN | | | |
| MCLR/VPP/RA5 | 4 | 4 | 1 | | | Master Clear (input) or programming voltage (input). |
| MCLR | | | | I | ST | Master Clear (Reset) input. This pin is an active-low Reset to the device. |
| VPP | | | | P | — | Programming voltage input. |
| RA5 | | | | I | ST | Digital input. |
| OSC1/CLKI/RA7 | 16 | 18 | 21 | | | Oscillator crystal or external clock input. |
| OSC1 | | | | I | ST | Oscillator crystal input or external clock source input. ST buffer when configured in RC mode, CMOS otherwise. |
| CLKI | | | | I | CMOS | External clock source input. Always associated with pin function OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.) |
| RA7 | | | | I/O | ST | General purpose I/O pin. |
| OSC2/CLKO/RA6 | 15 | 17 | 20 | | | Oscillator crystal or clock output. |
| OSC2 | | | | O | — | Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. |
| CLKO | | | | O | — | In RC, EC and INTRC modes, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes instruction cycle rate. |
| RA6 | | | | I/O | ST | General purpose I/O pin. |
| | | | | | | PORTA is a bidirectional I/O port. |
| RA0/AN0 | 1 | 1 | 26 | | | |
| RA0 | | | | I/O | ST | Digital I/O. |
| AN0 | | | | I | Analog | Analog input 0. |
| RA1/AN1/LVDIN | 2 | 2 | 27 | | | |
| RA1 | | | | I/O | ST | Digital I/O. |
| AN1 | | | | I | Analog | Analog input 1. |
| LVDIN | | | | I | Analog | Low-Voltage Detect input. |
| RA2/AN2/VREF- | 6 | 7 | 7 | | | |
| RA2 | | | | I/O | ST | Digital I/O. |
| AN2 | | | | I | Analog | Analog input 2. |
| VREF- | | | | I | Analog | A/D reference voltage (low) input. |
| RA3/AN3/VREF+ | 7 | 8 | 8 | | | |
| RA3 | | | | I/O | ST | Digital I/O. |
| AN3 | | | | I | Analog | Analog input 3. |
| VREF+ | | | | I | Analog | A/D reference voltage (high) input. |
| RA4/T0CKI | 3 | 3 | 28 | | | |
| RA4 | | | | I/O | ST/OD | Digital I/O. Open-drain when configured as output. |
| T0CKI | | | | I | ST | Timer0 external clock input. |
| RA5 | | | | | | See the MCLR/VPP/RA5 pin. |
| RA6 | | | | | | See the OSC2/CLKO/RA6 pin. |
| RA7 | | | | | | See the OSC1/CLKI/RA7 pin. |

**Legend:** TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
O = Output
OD = Open-drain (no P diode to VDD)
CMOS = CMOS compatible input or output
I = Input
P = Power


© 2002-2015 Microchip Technology Inc.

**TABLE 1-2: PIC18F1220/1320 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|---|
| | PDIP/ SOIC | SSOP | QFN | | | |
| | | | | | | PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs. |
| RB0/AN4/INT0 | 8 | 9 | 9 | | | |
| RB0 | | | | I/O | TTL | Digital I/O. |
| AN4 | | | | I | Analog | Analog input 4. |
| INT0 | | | | I | ST | External interrupt 0. |
| RB1/AN5/TX/CK/INT1 | 9 | 10 | 10 | | | |
| RB1 | | | | I/O | TTL | Digital I/O. |
| AN5 | | | | I | Analog | Analog input 5. |
| TX | | | | O | — | EUSART asynchronous transmit. |
| CK | | | | I/O | ST | EUSART synchronous clock (see related RX/DT). |
| INT1 | | | | I | ST | External interrupt 1. |
| RB2/P1B/INT2 | 17 | 19 | 23 | | | |
| RB2 | | | | I/O | TTL | Digital I/O. |
| P1B | | | | O | — | Enhanced CCP1/PWM output. |
| INT2 | | | | I | ST | External interrupt 2. |
| RB3/CCP1/P1A | 18 | 20 | 24 | | | |
| RB3 | | | | I/O | TTL | Digital I/O. |
| CCP1 | | | | I/O | ST | Capture 1 input/Compare 1 output/PWM 1 output. |
| P1A | | | | O | — | Enhanced CCP1/PWM output. |
| RB4/AN6/RX/DT/KBI0 | 10 | 11 | 12 | | | |
| RB4 | | | | I/O | TTL | Digital I/O. |
| AN6 | | | | I | Analog | Analog input 6. |
| RX | | | | I | ST | EUSART asynchronous receive. |
| DT | | | | I/O | ST | EUSART synchronous data (see related TX/CK). |
| KBI0 | | | | I | TTL | Interrupt-on-change pin. |
| RB5/PGM/KBI1 | 11 | 12 | 13 | | | |
| RB5 | | | | I/O | TTL | Digital I/O. |
| PGM | | | | I/O | ST | Low-Voltage ICSP™ Programming enable pin. |
| KBI1 | | | | I | TTL | Interrupt-on-change pin. |
| RB6/PGC/T1OSO/ T13CKI/P1C/KBI2 | 12 | 13 | 15 | | | |
| RB6 | | | | I/O | TTL | Digital I/O. |
| PGC | | | | I/O | ST | In-Circuit Debugger and ICSP programming clock pin. |
| T1OSO | | | | O | — | Timer1 oscillator output. |
| T13CKI | | | | I | ST | Timer1/Timer3 external clock output. |
| P1C | | | | O | — | Enhanced CCP1/PWM output. |
| KBI2 | | | | I | TTL | Interrupt-on-change pin. |
| RB7/PGD/T1OSI/ P1D/KBI3 | 13 | 14 | 16 | | | |
| RB7 | | | | I/O | TTL | Digital I/O. |
| PGD | | | | I/O | ST | In-Circuit Debugger and ICSP programming data pin. |
| T1OSI | | | | I | CMOS | Timer1 oscillator input. |
| P1D | | | | O | — | Enhanced CCP1/PWM output. |
| KBI3 | | | | I | TTL | Interrupt-on-change pin. |
| VSS | 5 | 5, 6 | 3, 5 | P | — | Ground reference for logic and I/O pins. |
| VDD | 14 | 15, 16 | 17, 19 | P | — | Positive supply for logic and I/O pins. |
| NC | — | — | 18 | — | — | No connect. |

**Legend:** TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
O = Output
OD = Open-drain (no P diode to VDD)
CMOS = CMOS compatible input or output
I = Input
P = Power

## 2.0 OSCILLATOR CONFIGURATIONS

### 2.1 Oscillator Types

The PIC18F1220 and PIC18F1320 devices can be operated in ten different oscillator modes. The user can program the Configuration bits, FOSC3:FOSC0, in Configuration Register 1H to select one of these ten modes:

1. LP — Low-Power Crystal
2. XT — Crystal/Resonator
3. HS — High-Speed Crystal/Resonator
4. HSPLL — High-Speed Crystal/Resonator with PLL enabled
5. RC — External Resistor/Capacitor with FOSC/4 output on RA6
6. RCIO — External Resistor/Capacitor with I/O on RA6
7. INTIO1 — Internal Oscillator with FOSC/4 output on RA6 and I/O on RA7
8. INTIO2 — Internal Oscillator with I/O on RA6 and RA7
9. EC — External Clock with FOSC/4 output
10. ECIO — External Clock with I/O on RA6

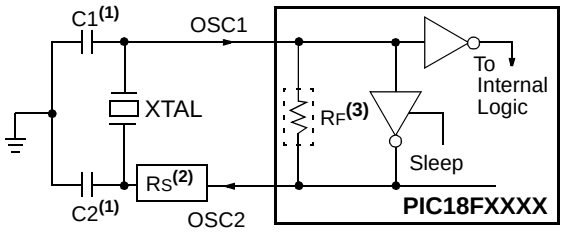
### 2.2 Crystal Oscillator/Ceramic Resonators

In XT, LP, HS or HSPLL Oscillator modes, a crystal or ceramic resonator is connected to the OSC1 and OSC2 pins to establish oscillation. Figure 2-1 shows the pin connections.

The oscillator design requires the use of a parallel cut crystal.

| **Note:** | Use of a series cut crystal may give a frequency out of the crystal manufacturer's specifications. |
|---|---|

**FIGURE 2-1: CRYSTAL/CERAMIC RESONATOR OPERATION (XT, LP, HS OR HSPLL CONFIGURATION)**




**Note 1:** See Table 2-1 and Table 2-2 for initial values of C1 and C2.

**2:** A series resistor (RS) may be required for AT strip cut crystals.

**3:** RF varies with the oscillator mode chosen.

**TABLE 2-1: CAPACITOR SELECTION FOR CERAMIC RESONATORS**

| Typical Capacitor Values Used: | | | |
|---|---|---|---|
| **Mode** | **Freq.** | **OSC1** | **OSC2** |
| XT | 455 kHz<br>2.0 MHz<br>4.0 MHz | 56 pF<br>47 pF<br>33 pF | 56 pF<br>47 pF<br>33 pF |
| HS | 8.0 MHz<br>16.0 MHz | 27 pF<br>22 pF | 27 pF<br>22 pF |

**Capacitor values are for design guidance only.**

These capacitors were tested with the resonators listed below for basic start-up and operation. **These values are not optimized**.

Different capacitor values may be required to produce acceptable oscillator operation. The user should test the performance of the oscillator over the expected VDD and temperature range for the application.

See the notes following Table 2-2 for additional information.

| Resonators Used: | |
|---|---|
| 455 kHz | 4.0 MHz |
| 2.0 MHz | 8.0 MHz |
| 16.0 MHz | |


© 2002-2015 Microchip Technology Inc.

**TABLE 2-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR**

| Osc Type | Crystal Freq. | Typical Capacitor Values Tested: | |
|---|---|---|---|
| | | C1 | C2 |
| LP | 32 kHz | 33 pF | 33 pF |
| | 200 kHz | 15 pF | 15 pF |
| XT | 1 MHz | 33 pF | 33 pF |
| | 4 MHz | 27 pF | 27 pF |
| HS | 4 MHz | 27 pF | 27 pF |
| | 8 MHz | 22 pF | 22 pF |
| | 20 MHz | 15 pF | 15 pF |

**Capacitor values are for design guidance only.**

These capacitors were tested with the crystals listed below for basic start-up and operation. **These values are not optimized.**

Different capacitor values may be required to produce acceptable oscillator operation. The user should test the performance of the oscillator over the expected VDD and temperature range for the application.

See the notes following this table for additional information.

| Crystals Used: | |
|---|---|
| 32 kHz | 4 MHz |
| 200 kHz | 8 MHz |
| 1 MHz | 20 MHz |

**Note 1:** Higher capacitance increases the stability of oscillator, but also increases the start-up time.

**2:** When operating below 3V VDD, or when using certain ceramic resonators at any voltage, it may be necessary to use the HS mode or switch to a crystal oscillator.

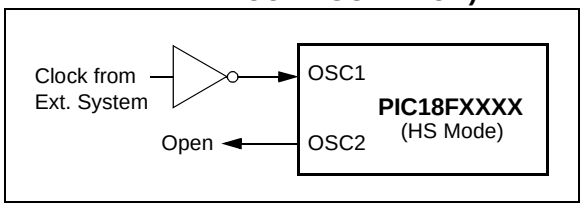
**3:** Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

**4:** RS may be required to avoid overdriving crystals with low drive level specification.

**5:** Always verify oscillator performance over the VDD and temperature range that is expected for the application.

An external clock source may also be connected to the OSC1 pin in the HS mode, as shown in Figure 2-2.

**FIGURE 2-2: EXTERNAL CLOCK INPUT OPERATION (HS OSC CONFIGURATION)**



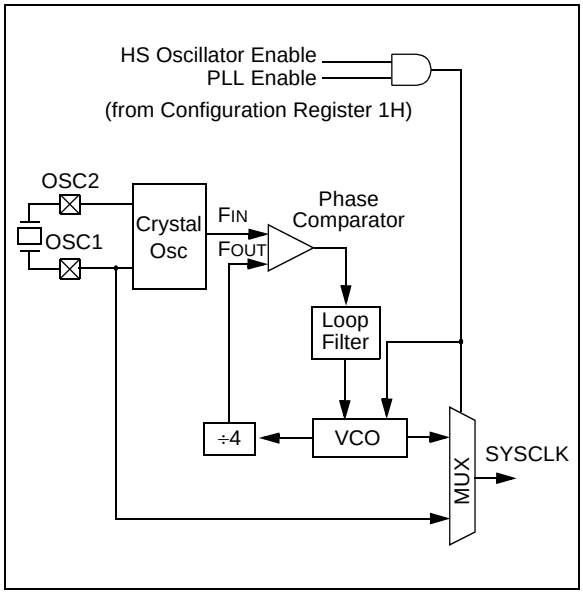

## 2.3 HSPLL

A Phase Locked Loop (PLL) circuit is provided as an option for users who wish to use a lower frequency crystal oscillator circuit, or to clock the device up to its highest rated frequency from a crystal oscillator. This may be useful for customers who are concerned with EMI due to high-frequency crystals.

The HSPLL mode makes use of the HS mode oscillator for frequencies up to 10 MHz. A PLL then multiplies the oscillator output frequency by 4 to produce an internal clock frequency up to 40 MHz.

The PLL is enabled only when the oscillator Configuration bits are programmed for HSPLL mode. If programmed for any other mode, the PLL is not enabled.

**FIGURE 2-3: PLL BLOCK DIAGRAM**

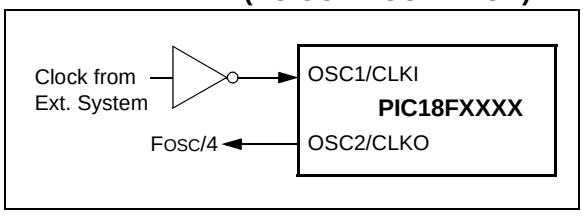


## 2.4 External Clock Input

The EC and ECIO Oscillator modes require an external clock source to be connected to the OSC1 pin. There is no oscillator start-up time required after a Power-on Reset, or after an exit from Sleep mode.

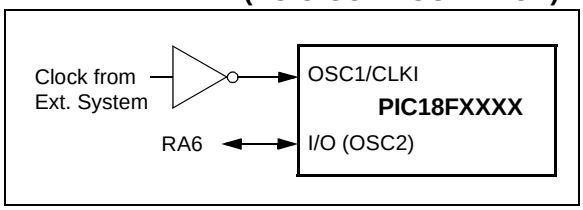
In the EC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes, or to synchronize other logic. Figure 2-4 shows the pin connections for the EC Oscillator mode.

**FIGURE 2-4: EXTERNAL CLOCK INPUT OPERATION (EC CONFIGURATION)**




The ECIO Oscillator mode functions like the EC mode, except that the OSC2 pin becomes an additional general purpose I/O pin. The I/O pin becomes bit 6 of PORTA (RA6). Figure 2-5 shows the pin connections for the ECIO Oscillator mode.

**FIGURE 2-5: EXTERNAL CLOCK INPUT OPERATION (ECIO CONFIGURATION)**

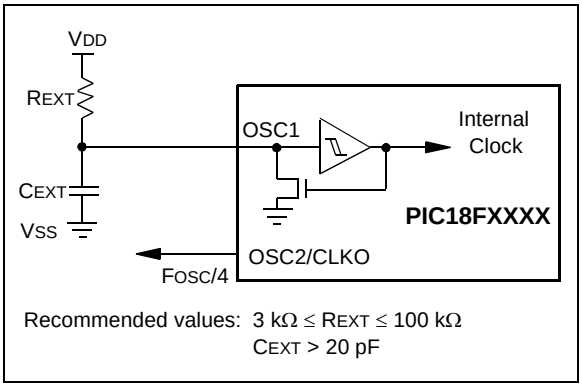



## 2.5 RC Oscillator

For timing insensitive applications, the "RC" and "RCIO" device options offer additional cost savings. The RC oscillator frequency is a function of the supply voltage, the resistor (REXT) and capacitor (CEXT) values and the operating temperature. In addition to this, the oscillator frequency will vary from unit to unit due to normal manufacturing variation. Furthermore, the difference in lead frame capacitance between package types will also affect the oscillation frequency, especially for low CEXT values. The user also needs to take into account variation, due to tolerance of external R and C components used. Figure 2-6 shows how the R/C combination is connected.

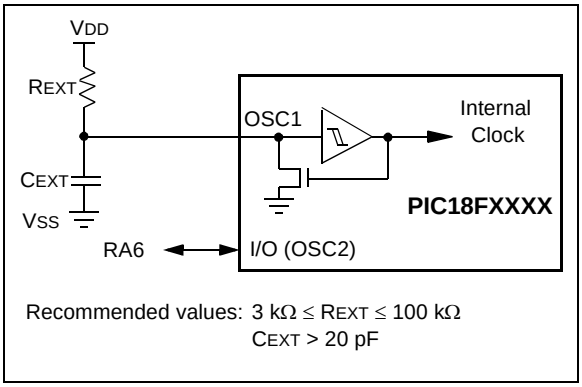
In the RC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes, or to synchronize other logic.

**FIGURE 2-6: RC OSCILLATOR MODE**




The RCIO Oscillator mode (Figure 2-7) functions like the RC mode, except that the OSC2 pin becomes an additional general purpose I/O pin. The I/O pin becomes bit 6 of PORTA (RA6).

**FIGURE 2-7: RCIO OSCILLATOR MODE**





© 2002-2015 Microchip Technology Inc.

## 2.6 Internal Oscillator Block

The PIC18F1220/1320 devices include an internal oscillator block, which generates two different clock signals; either can be used as the system's clock source. This can eliminate the need for external oscillator circuits on the OSC1 and/or OSC2 pins.

The main output (INTOSC) is an 8 MHz clock source, which can be used to directly drive the system clock. It also drives a postscaler, which can provide a range of clock frequencies from 125 kHz to 4 MHz. The INTOSC output is enabled when a system clock frequency from 125 kHz to 8 MHz is selected.

The other clock source is the internal RC oscillator (INTRC), which provides a 31 kHz output. The INTRC oscillator is enabled by selecting the internal oscillator block as the system clock source, or when any of the following are enabled:

- Power-up Timer
- Fail-Safe Clock Monitor
- Watchdog Timer
- Two-Speed Start-up

These features are discussed in greater detail in **Section 19.0 "Special Features of the CPU"**.

The clock source frequency (INTOSC direct, INTRC direct or INTOSC postscaler) is selected by configuring the IRCF bits of the OSCCON register (Register 2-2).

### 2.6.1 INTIO MODES

Using the internal oscillator as the clock source can eliminate the need for up to two external oscillator pins, which can then be used for digital I/O. Two distinct configurations are available:

- In INTIO1 mode, the OSC2 pin outputs FOSC/4, while OSC1 functions as RA7 for digital input and output.
- In INTIO2 mode, OSC1 functions as RA7 and OSC2 functions as RA6, both for digital input and output.

### 2.6.2 INTRC OUTPUT FREQUENCY

The internal oscillator block is calibrated at the factory to produce an INTOSC output frequency of 8.0 MHz (see Table 22-6). This changes the frequency of the INTRC source from its nominal 31.25 kHz. Peripherals and features that depend on the INTRC source will be affected by this shift in frequency.

Once set during factory calibration, the INTRC frequency will remain within ±2% as temperature and VDD change across their full specified operating ranges.

### 2.6.3 OSCTUNE REGISTER

The internal oscillator's output has been calibrated at the factory, but can be adjusted in the user's application. This is done by writing to the OSCTUNE register (Register 2-1). The tuning sensitivity is constant throughout the tuning range.

When the OSCTUNE register is modified, the INTOSC and INTRC frequencies will begin shifting to the new frequency. The INTRC clock will reach the new frequency within 8 clock cycles (approximately 8 * 32 μs = 256 μs). The INTOSC clock will stabilize within 1 ms. Code execution continues during this shift. There is no indication that the shift has occurred. Operation of features that depend on the INTRC clock source frequency, such as the WDT, Fail-Safe Clock Monitor and peripherals, will also be affected by the change in frequency.

© 2002-2015 Microchip Technology Inc.

**REGISTER 2-1: OSCTUNE: OSCILLATOR TUNING REGISTER**

| U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | — | TUN<5:0> | | | | | |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7-6 **Unimplemented:** Read as ‘0’

bit 5-0 **TUN<5:0>:** Frequency Tuning bits

100000 = Minimum frequency
•
•
•
111111 =
000000 = Oscillator module is running at the factory-calibrated frequency
000001 =
•
•
•
011110 =
011111 = Maximum frequency

## 2.7 Clock Sources and Oscillator Switching

Like previous PIC18 devices, the PIC18F1220/1320 devices include a feature that allows the system clock source to be switched from the main oscillator to an alternate low-frequency clock source. PIC18F1220/1320 devices offer two alternate clock sources. When enabled, these give additional options for switching to the various power managed operating modes.

Essentially, there are three clock sources for these devices:

- Primary oscillators
- Secondary oscillators
- Internal oscillator block

The **primary oscillators** include the External Crystal and Resonator modes, the External RC modes, the External Clock modes and the internal oscillator block. The particular mode is defined on POR by the contents of Configuration Register 1H. The details of these modes are covered earlier in this chapter.

The **secondary oscillators** are those external sources not connected to the OSC1 or OSC2 pins. These sources may continue to operate even after the controller is placed in a power managed mode.

PIC18F1220/1320 devices offer only the Timer1 oscillator as a secondary oscillator. This oscillator, in all power managed modes, is often the time base for functions such as a real-time clock.

Most often, a 32.768 kHz watch crystal is connected between the RB6/T1OSO and RB7/T1OSI pins. Like the LP mode oscillator circuit, loading capacitors are also connected from each pin to ground. These pins are also used during ICSP operations.

The Timer1 oscillator is discussed in greater detail in **Section 12.2 “Timer1 Oscillator”**.

In addition to being a primary clock source, the **internal oscillator block** is available as a power managed mode clock source. The INTRC source is also used as the clock source for several special features, such as the WDT and Fail-Safe Clock Monitor.

The clock sources for the PIC18F1220/1320 devices are shown in Figure 2-8. See **Section 12.0 “Timer1 Module”** for further details of the Timer1 oscillator. See **Section 19.1 “Configuration Bits”** for Configuration register details.


© 2002-2015 Microchip Technology Inc.

### 2.7.1 OSCILLATOR CONTROL REGISTER

The OSCCON register (Register 2-2) controls several aspects of the system clock's operation, both in full-power operation and in power managed modes.

The System Clock Select bits, SCS1:SCS0, select the clock source that is used when the device is operating in power managed modes. The available clock sources are the primary clock (defined in Configuration Register 1H), the secondary clock (Timer1 oscillator) and the internal oscillator block. The clock selection has no effect until a SLEEP instruction is executed and the device enters a power managed mode of operation. The SCS bits are cleared on all forms of Reset.

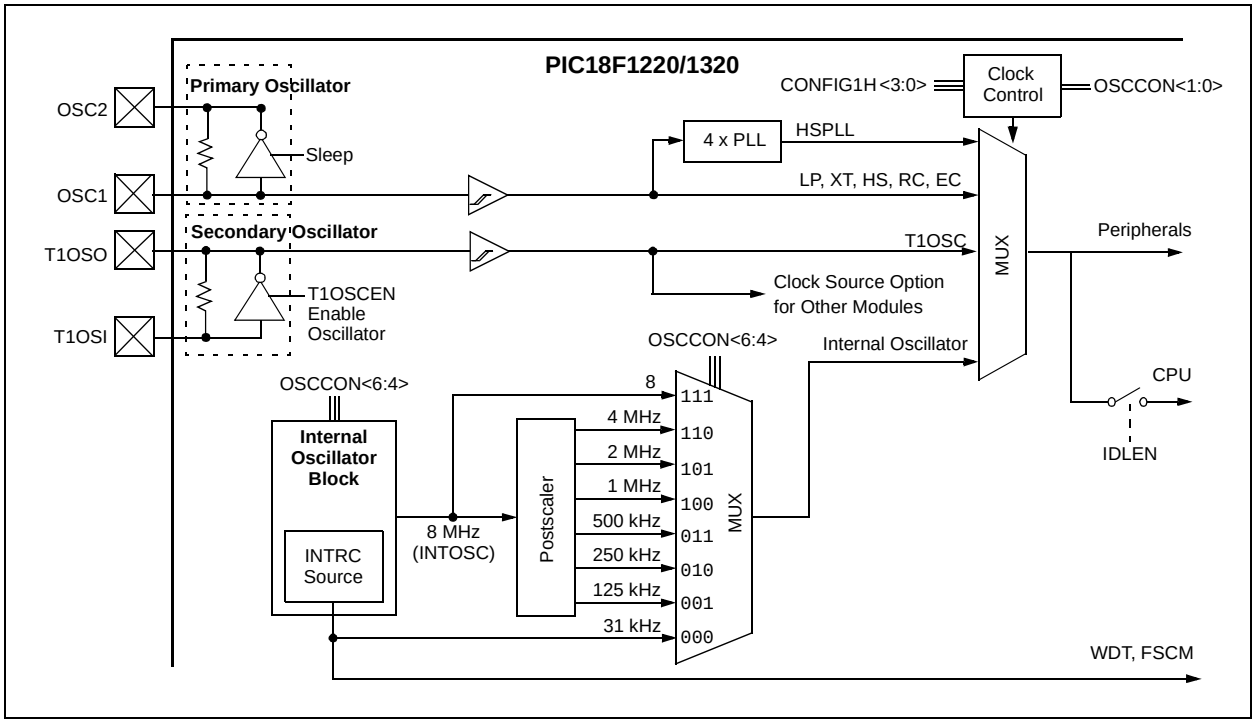
The Internal Oscillator Select bits, IRCF2:IRCF0, select the frequency output of the internal oscillator block that is used to drive the system clock. The choices are the INTRC source, the INTOSC source (8 MHz), or one of the six frequencies derived from the INTOSC postscaler (125 kHz to 4 MHz). If the internal oscillator block is supplying the system clock, changing the states of these bits will have an immediate change on the internal oscillator's output.

The OSTS, IOFS and T1RUN bits indicate which clock source is currently providing the system clock. The OSTS indicates that the Oscillator Start-up Timer has timed out and the primary clock is providing the system clock in Primary Clock modes. The IOFS bit indicates when the internal oscillator block has stabilized and is providing the system clock in RC Clock modes or during Two-Speed Start-ups. The T1RUN bit (T1CON<6>) indicates when the Timer1 oscillator is providing the system clock in Secondary Clock modes. In power managed modes, only one of these three bits will be set at any time. If none of these bits are set, the INTRC is providing the system clock, or the internal oscillator block has just started and is not yet stable.

The IDLEN bit controls the selective shutdown of the controller's CPU in power managed modes. The uses of these bits are discussed in more detail in **Section 3.0 "Power Managed Modes"**.

| | |
|---|---|
| **Note 1:** | The Timer1 oscillator must be enabled to select the secondary clock source. The Timer1 oscillator is enabled by setting the T1OSCEN bit in the Timer1 Control register (T1CON<3>). If the Timer1 oscillator is not enabled, then any attempt to select a secondary clock source when executing a SLEEP instruction will be ignored. |
| **2:** | It is recommended that the Timer1 oscillator be operating and stable before executing the SLEEP instruction or a very long delay may occur while the Timer1 oscillator starts. |

**FIGURE 2-8: PIC18F1220/1320 CLOCK DIAGRAM**



**REGISTER 2-2: OSCCON: OSCILLATOR CONTROL REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-1/1 | R/W-1/1 | R[(1)] | R-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| IDLEN | IRCF<2:0> | | | OSTS | IOFS | SCS<1:0> | |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **IDLEN:** Idle Enable bits
1 = Idle mode enabled; CPU core is not clocked in power managed modes
0 = Run mode enabled; CPU core is clocked in Run modes, but not Sleep mode

bit 6-4 **IRCF<2:0>:** Internal Oscillator Frequency Select bits
111 = 8 MHz (8 MHz source drives clock directly)
110 = 4 MHz
101 = 2 MHz
100 = 1 MHz
011 = 500 kHz
010 = 250 kHz
001 = 125 kHz
000 = 31 kHz (INTRC source drives clock directly)

bit 3 **OSTS:** Oscillator Start-up Time-out Status bit
1 = Oscillator Start-up Timer time-out has expired; primary oscillator is running
0 = Oscillator Start-up Timer time-out is running; primary oscillator is not ready

bit 2 **IOFS:** INTOSC Frequency Stable bit
1 = INTOSC frequency is stable
0 = INTOSC frequency is not stable

bit 1-0 **SCS<1:0>:** System Clock Select bits[(1)]
1x = Internal oscillator block (RC modes)
01 =Timer1 oscillator (Secondary modes)
00 = Primary oscillator (Sleep and PRI_IDLE modes)

**Note 1:** Depends on state of the IESO bit in Configuration Register 1H.


© 2002-2015 Microchip Technology Inc.

### 2.7.2 OSCILLATOR TRANSITIONS

The PIC18F1220/1320 devices contain circuitry to prevent clocking “glitches” when switching between clock sources. A short pause in the system clock occurs during the clock switch. The length of this pause is between eight and nine clock periods of the new clock source. This ensures that the new clock source is stable and that its pulse width will not be less than the shortest pulse width of the two clock sources.

Clock transitions are discussed in greater detail in **Section 3.1.2 “Entering Power Managed Modes”**.

## 2.8 Effects of Power Managed Modes on the Various Clock Sources

When the device executes a SLEEP instruction, the system is switched to one of the power managed modes, depending on the state of the IDLEN and SCS1:SCS0 bits of the OSCCON register. See **Section 3.0 “Power Managed Modes”** for details.

When PRI_IDLE mode is selected, the designated primary oscillator continues to run without interruption. For all other power managed modes, the oscillator using the OSC1 pin is disabled. The OSC1 pin (and OSC2 pin, if used by the oscillator) will stop oscillating.

In Secondary Clock modes (SEC_RUN and SEC_IDLE), the Timer1 oscillator is operating and providing the system clock. The Timer1 oscillator may also run in all power managed modes if required to clock Timer1 or Timer3.

In Internal Oscillator modes (RC_RUN and RC_IDLE), the internal oscillator block provides the system clock source. The INTRC output can be used directly to provide the system clock and may be enabled to support various special features, regardless of the power managed mode (see **Section 19.2 “Watchdog Timer (WDT)”** through **Section 19.4 “Fail-Safe Clock Monitor”**). The INTOSC output at 8 MHz may be used directly to clock the system, or may be divided down first. The INTOSC output is disabled if the system clock is provided directly from the INTRC output.

If the Sleep mode is selected, all clock sources are stopped. Since all the transistor switching currents have been stopped, Sleep mode achieves the lowest current consumption of the device (only leakage currents).

Enabling any on-chip feature that will operate during Sleep will increase the current consumed during Sleep. The INTRC is required to support WDT operation. The Timer1 oscillator may be operating to support a real-time clock. Other features may be operating that do not require a system clock source (i.e., INTn pins, A/D conversions and others).

## 2.9 Power-up Delays

Power-up delays are controlled by two timers, so that no external Reset circuitry is required for most applications. The delays ensure that the device is kept in Reset until the device power supply is stable under normal circumstances and the primary clock is operating and stable. For additional information on power-up delays, see Sections 4.1 through 4.5.

The first timer is the Power-up Timer (PWRT), which provides a fixed delay on power-up (parameter 33, Table 22-8) if enabled in Configuration Register 2L. The second timer is the Oscillator Start-up Timer (OST), intended to keep the chip in Reset until the crystal oscillator is stable (LP, XT and HS modes). The OST does this by counting 1024 oscillator cycles before allowing the oscillator to clock the device.

When the HSPLL Oscillator mode is selected, the device is kept in Reset for an additional 2 ms following the HS mode OST delay, so the PLL can lock to the incoming clock frequency.

There is a delay of 5 to 10 μs following POR while the controller becomes ready to execute instructions. This delay runs concurrently with any other delays. This may be the only delay that occurs when any of the EC, RC or INTIO modes are used as the primary clock source.

**TABLE 2-3: OSC1 AND OSC2 PIN STATES IN SLEEP MODE**

| Oscillator Mode | OSC1 Pin | OSC2 Pin |
|---|---|---|
| RC, INTIO1 | Floating, external resistor should pull high | At logic low (clock/4 output) |
| RCIO, INTIO2 | Floating, external resistor should pull high | Configured as PORTA, bit 6 |
| ECIO | Floating, pulled by external clock | Configured as PORTA, bit 6 |
| EC | Floating, pulled by external clock | At logic low (clock/4 output) |
| LP, XT and HS | Feedback inverter disabled at quiescent voltage level | Feedback inverter disabled at quiescent voltage level |

**Note:** See Table 4-1 in **Section 4.0 “Reset”** for time-outs due to Sleep and $\overline{\text{MCLR}}$ Reset.

## 3.0 POWER MANAGED MODES

The PIC18F1220/1320 devices offer a total of six operating modes for more efficient power management (see Table 3-1). These provide a variety of options for selective power conservation in applications where resources may be limited (i.e., battery powered devices).

There are three categories of power managed modes:

- Sleep mode
- Idle modes
- Run modes

These categories define which portions of the device are clocked and sometimes, what speed. The Run and Idle modes may use any of the three available clock sources (primary, secondary or INTOSC multiplexer); the Sleep mode does not use a clock source.

The clock switching feature offered in other PIC18 devices (i.e., using the Timer1 oscillator in place of the primary oscillator) and the Sleep mode offered by all PIC® devices (where all system clocks are stopped) are both offered in the PIC18F1220/1320 devices (SEC_RUN and Sleep modes, respectively). However, additional power managed modes are available that allow the user greater flexibility in determining what portions of the device are operating. The power managed modes are event driven; that is, some specific event must occur for the device to enter or (more particularly) exit these operating modes.

For PIC18F1220/1320 devices, the power managed modes are invoked by using the existing SLEEP instruction. All modes exit to PRI_RUN mode when triggered by an interrupt, a Reset or a WDT time-out (PRI_RUN mode is the normal full-power execution mode; the CPU and peripherals are clocked by the primary oscillator source). In addition, power managed Run modes may also exit to Sleep mode, or their corresponding Idle mode.

### 3.1 Selecting Power Managed Modes

Selecting a power managed mode requires deciding if the CPU is to be clocked or not and selecting a clock source. The IDLEN bit controls CPU clocking, while the SCS1:SCS0 bits select a clock source. The individual modes, bit settings, clock sources and affected modules are summarized in Table 3-1.

#### 3.1.1 CLOCK SOURCES

The clock source is selected by setting the SCS bits of the OSCCON register (Register 2-2). Three clock sources are available for use in power managed Idle modes: the primary clock (as configured in Configuration Register 1H), the secondary clock (Timer1 oscillator) and the internal oscillator block. The secondary and internal oscillator block sources are available for the power managed modes (PRI_RUN mode is the normal full-power execution mode; the CPU and peripherals are clocked by the primary oscillator source).

**TABLE 3-1: POWER MANAGED MODES**

| Mode | OSCCON Bits | | Module Clocking | | Available Clock and Oscillator Source |
|---|---|---|---|---|---|
| | IDLEN <7> | SCS1:SCS0 <1:0> | CPU | Peripherals | |
| Sleep | 0 | 00 | Off | Off | None – All clocks are disabled |
| PRI_RUN | 0 | 00 | Clocked | Clocked | Primary – LP, XT, HS, HSPLL, RC, EC, INTRC(1) This is the normal full-power execution mode. |
| SEC_RUN | 0 | 01 | Clocked | Clocked | Secondary – Timer1 Oscillator |
| RC_RUN | 0 | 1x | Clocked | Clocked | Internal Oscillator Block(1) |
| PRI_IDLE | 1 | 00 | Off | Clocked | Primary – LP, XT, HS, HSPLL, RC, EC |
| SEC_IDLE | 1 | 01 | Off | Clocked | Secondary – Timer1 Oscillator |
| RC_IDLE | 1 | 1x | Off | Clocked | Internal Oscillator Block(1) |

**Note 1:** Includes INTOSC and INTOSC postscaler, as well as the INTRC source.


© 2002-2015 Microchip Technology Inc.

### 3.1.2 ENTERING POWER MANAGED MODES

In general, entry, exit and switching between power managed clock sources requires clock source switching. In each case, the sequence of events is the same.

Any change in the power managed mode begins with loading the OSCCON register and executing a SLEEP instruction. The SCS1:SCS0 bits select one of three power managed clock sources; the primary clock (as defined in Configuration Register 1H), the secondary clock (the Timer1 oscillator) and the internal oscillator block (used in RC modes). Modifying the SCS bits will have no effect until a SLEEP instruction is executed. Entry to the power managed mode is triggered by the execution of a SLEEP instruction.

Figure 3-5 shows how the system is clocked while switching from the primary clock to the Timer1 oscillator. When the SLEEP instruction is executed, clocks to the device are stopped at the beginning of the next instruction cycle. Eight clock cycles from the new clock source are counted to synchronize with the new clock source. After eight clock pulses from the new clock source are counted, clocks from the new clock source resume clocking the system. The actual length of the pause is between eight and nine clock periods from the new clock source. This ensures that the new clock source is stable and that its pulse width will not be less than the shortest pulse width of the two clock sources.

Three bits indicate the current clock source: OSTS and IOFS in the OSCCON register and T1RUN in the T1CON register. Only one of these bits will be set while in a power managed mode. When the OSTS bit is set, the primary clock is providing the system clock. When the IOFS bit is set, the INTOSC output is providing a stable 8 MHz clock source and is providing the system clock. When the T1RUN bit is set, the Timer1 oscillator is providing the system clock. If none of these bits are set, then either the INTRC clock source is clocking the system, or the INTOSC source is not yet stable.

If the internal oscillator block is configured as the primary clock source in Configuration Register 1H, then both the OSTS and IOFS bits may be set when in PRI_RUN or PRI_IDLE modes. This indicates that the primary clock (INTOSC output) is generating a stable 8 MHz output. Entering an RC power managed mode (same frequency) would clear the OSTS bit.

**Note 1:** Caution should be used when modifying a single IRCF bit. If VDD is less than 3V, it is possible to select a higher clock speed than is supported by the low VDD. Improper device operation may result if the VDD/FOSC specifications are violated.

**2:** Executing a SLEEP instruction does not necessarily place the device into Sleep mode; executing a SLEEP instruction is simply a trigger to place the controller into a power managed mode selected by the OSCCON register, one of which is Sleep mode.

### 3.1.3 MULTIPLE SLEEP COMMANDS

The power managed mode that is invoked with the SLEEP instruction is determined by the settings of the IDLEN and SCS bits at the time the instruction is executed. If another SLEEP instruction is executed, the device will enter the power managed mode specified by these same bits at that time. If the bits have changed, the device will enter the new power managed mode specified by the new bit settings.

### 3.1.4 COMPARISONS BETWEEN RUN AND IDLE MODES

Clock source selection for the Run modes is identical to the corresponding Idle modes. When a SLEEP instruction is executed, the SCS bits in the OSCCON register are used to switch to a different clock source. As a result, if there is a change of clock source at the time a SLEEP instruction is executed, a clock switch will occur.

In Idle modes, the CPU is not clocked and is not running. In Run modes, the CPU is clocked and executing code. This difference modifies the operation of the WDT when it times out. In Idle modes, a WDT time-out results in a wake from power managed modes. In Run modes, a WDT time-out results in a WDT Reset (see Table 3-2).

During a wake-up from an Idle mode, the CPU starts executing code by entering the corresponding Run mode until the primary clock becomes ready. When the primary clock becomes ready, the clock source is automatically switched to the primary clock. The IDLEN and SCS bits are unchanged during and after the wake-up.

Figure 3-2 shows how the system is clocked during the clock source switch. The example assumes the device was in SEC_IDLE or SEC_RUN mode when a wake is triggered (the primary clock was configured in HSPLL mode).

**TABLE 3-2: COMPARISON BETWEEN POWER MANAGED MODES**

| Power Managed Mode | CPU is Clocked by ... | WDT Time-out causes a ... | Peripherals are Clocked by ... | Clock during Wake-up (while primary becomes ready) |
|---|---|---|---|---|
| Sleep | Not clocked (not running) | Wake-up | Not clocked | None or INTOSC multiplexer if Two-Speed Start-up or Fail-Safe Clock Monitor are enabled |
| Any Idle mode | Not clocked (not running) | Wake-up | Primary, Secondary or INTOSC multiplexer | Unchanged from Idle mode (CPU operates as in corresponding Run mode) |
| Any Run mode | Primary or secondary clocks or INTOSC multiplexer | Reset | Primary or secondary clocks or INTOSC multiplexer | Unchanged from Run mode |

## 3.2 Sleep Mode

The power managed Sleep mode in the PIC18F1220/1320 devices is identical to that offered in all other PIC microcontrollers. It is entered by clearing the IDLEN and SCS1:SCS0 bits (this is the Reset state) and executing the SLEEP instruction. This shuts down the primary oscillator and the OSTS bit is cleared (see Figure 3-1).

When a wake event occurs in Sleep mode (by interrupt, Reset or WDT time-out), the system will not be clocked until the primary clock source becomes ready (see Figure 3-2), or it will be clocked from the internal oscillator block if either the Two-Speed Start-up or the Fail-Safe Clock Monitor are enabled (see **Section 19.0 "Special Features of the CPU"**). In either case, the OSTS bit is set when the primary clock is providing the system clocks. The IDLEN and SCS bits are not affected by the wake-up.

## 3.3 Idle Modes

The IDLEN bit allows the microcontroller's CPU to be selectively shut down while the peripherals continue to operate. Clearing IDLEN allows the CPU to be clocked. Setting IDLEN disables clocks to the CPU, effectively stopping program execution (see Register 2-2). The peripherals continue to be clocked regardless of the setting of the IDLEN bit.

There is one exception to how the IDLEN bit functions. When all the low-power OSCCON bits are cleared (IDLEN:SCS1:SCS0 = 000), the device enters Sleep mode upon the execution of the SLEEP instruction. This is both the Reset state of the OSCCON register and the setting that selects Sleep mode. This maintains compatibility with other PIC devices that do not offer power managed modes.

If the Idle Enable bit, IDLEN (OSCCON<7>), is set to a '1' when a SLEEP instruction is executed, the peripherals will be clocked from the clock source selected using the SCS1:SCS0 bits; however, the CPU will not be clocked. Since the CPU is not executing instructions, the only exits from any of the Idle modes are by interrupt, WDT time-out or a Reset.

When a wake event occurs, CPU execution is delayed approximately 10 μs while it becomes ready to execute code. When the CPU begins executing code, it is clocked by the same clock source as was selected in the power managed mode (i.e., when waking from RC_IDLE mode, the internal oscillator block will clock the CPU and peripherals until the primary clock source becomes ready – this is essentially RC_RUN mode). This continues until the primary clock source becomes ready. When the primary clock becomes ready, the OSTS bit is set and the system clock source is switched to the primary clock (see Figure 3-4). The IDLEN and SCS bits are not affected by the wake-up.

While in any Idle mode or the Sleep mode, a WDT time-out will result in a WDT wake-up to full-power operation.


© 2002-2015 Microchip Technology Inc.

**FIGURE 3-1: TIMING TRANSITION FOR ENTRY TO SLEEP MODE**

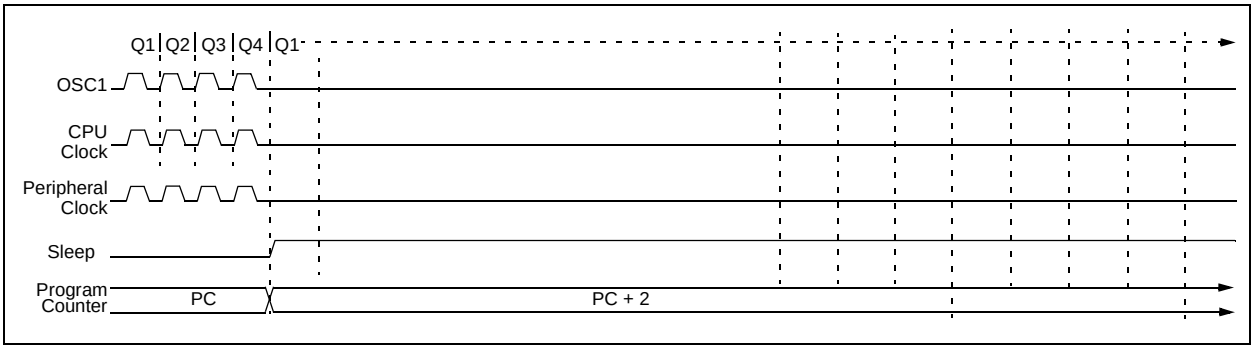
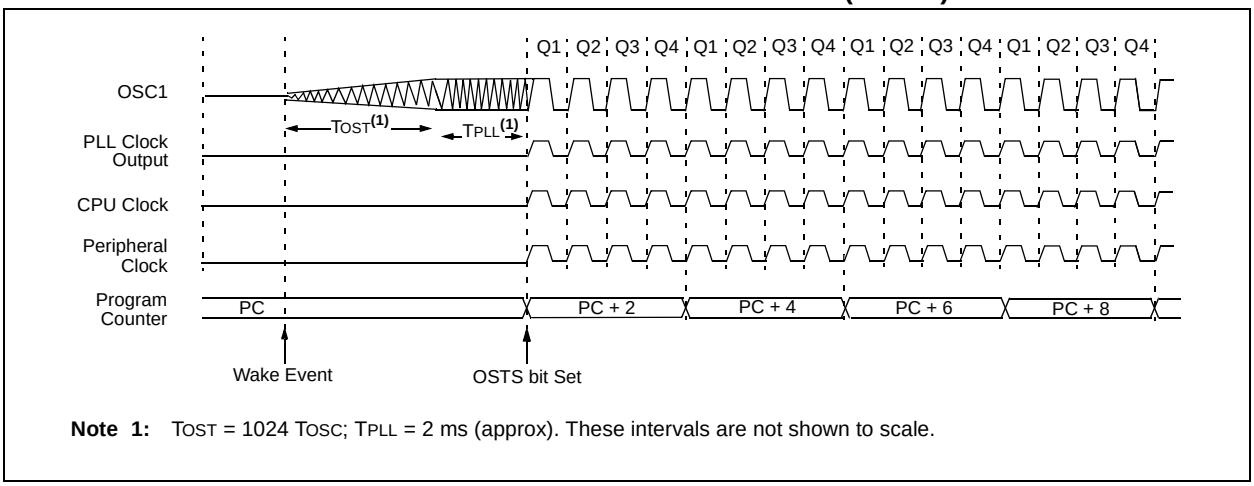



**FIGURE 3-2: TRANSITION TIMING FOR WAKE FROM SLEEP (HSPLL)**




**Note 1:** TOST = 1024 TOSC; TPLL = 2 ms (approx). These intervals are not shown to scale.

### 3.3.1 PRI_IDLE MODE

This mode is unique among the three Low-Power Idle modes, in that it does not disable the primary system clock. For timing sensitive applications, this allows for the fastest resumption of device operation with its more accurate primary clock source, since the clock source does not have to “warm up” or transition from another oscillator.

PRI_IDLE mode is entered by setting the IDLEN bit, clearing the SCS bits and executing a SLEEP instruction. Although the CPU is disabled, the peripherals continue to be clocked from the primary clock source specified in Configuration Register 1H. The OSTS bit remains set in PRI_IDLE mode (see Figure 3-3).

When a wake event occurs, the CPU is clocked from the primary clock source. A delay of approximately 10 μs is required between the wake event and code execution starts. This is required to allow the CPU to become ready to execute instructions. After the wake-up, the OSTS bit remains set. The IDLEN and SCS bits are not affected by the wake-up (see Figure 3-4).

**FIGURE 3-3: TRANSITION TIMING TO PRI_IDLE MODE**

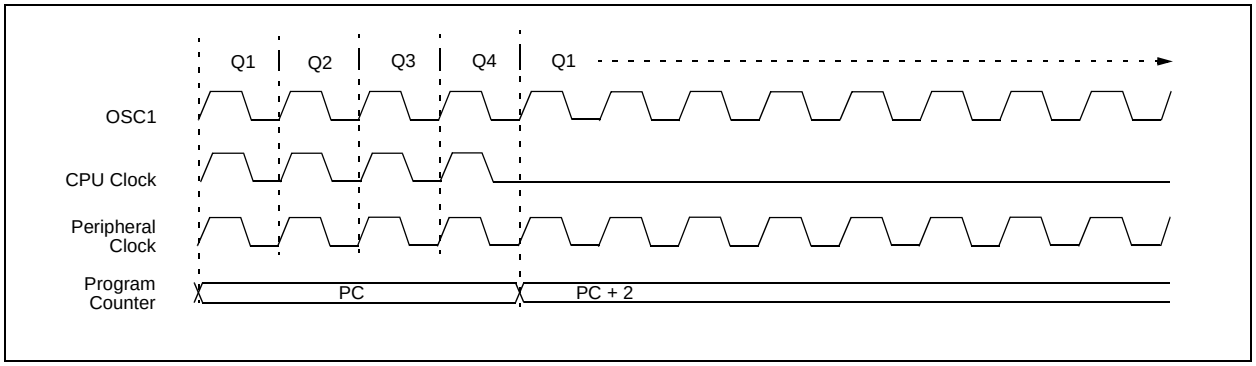
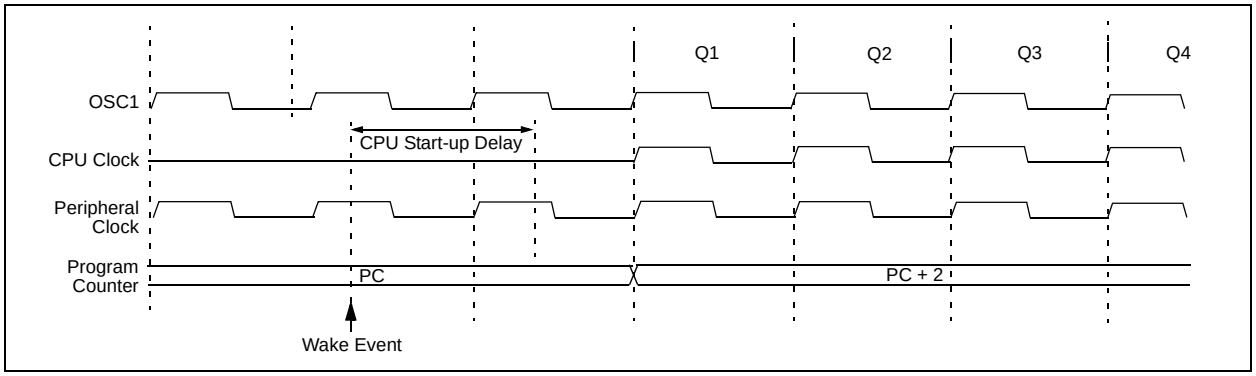



**FIGURE 3-4: TRANSITION TIMING FOR WAKE FROM PRI_IDLE MODE**





© 2002-2015 Microchip Technology Inc.

### 3.3.2 SEC_IDLE MODE

In SEC_IDLE mode, the CPU is disabled, but the peripherals continue to be clocked from the Timer1 oscillator. This mode is entered by setting the Idle bit, modifying bits, SCS1:SCS0 = 01 and executing a SLEEP instruction. When the clock source is switched (see Figure 3-5) to the Timer1 oscillator, the primary oscillator is shut down, the OSTS bit is cleared and the T1RUN bit is set.

**Note:** The Timer1 oscillator should already be running prior to entering SEC_IDLE mode. If the T1OSCEN bit is not set when the SLEEP instruction is executed, the SLEEP instruction will be ignored and entry to SEC_IDLE mode will not occur. If the Timer1 oscillator is enabled, but not yet running, peripheral clocks will be delayed until the oscillator has started; in such situations, initial oscillator operation is far from stable and unpredictable operation may result.

When a wake event occurs, the peripherals continue to be clocked from the Timer1 oscillator. After a 10 μs delay following the wake event, the CPU begins executing code, being clocked by the Timer1 oscillator. The microcontroller operates in SEC_RUN mode until the primary clock becomes ready. When the primary clock becomes ready, a clock switchback to the primary clock occurs (see Figure 3-6). When the clock switch is complete, the T1RUN bit is cleared, the OSTS bit is set and the primary clock is providing the system clock. The IDLEN and SCS bits are not affected by the wake-up. The Timer1 oscillator continues to run.

**FIGURE 3-5: TIMING TRANSITION FOR ENTRY TO SEC_IDLE MODE**

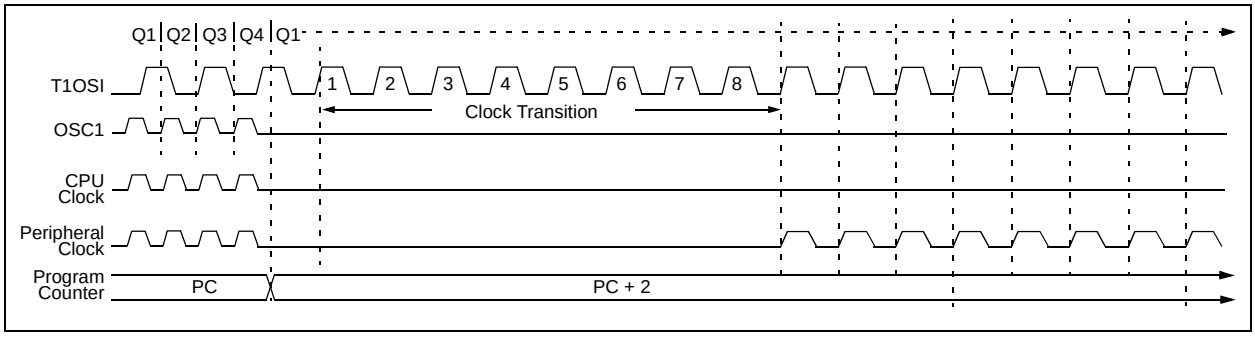
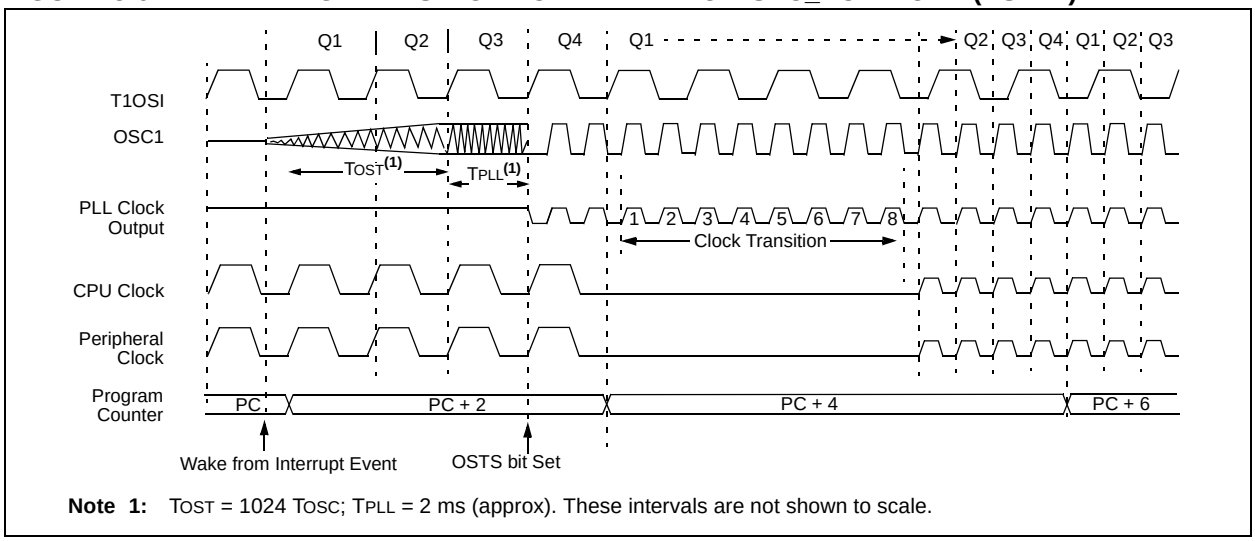



**FIGURE 3-6: TIMING TRANSITION FOR WAKE FROM SEC_RUN MODE (HSPLL)**




**Note 1:** TOST = 1024 TOSC; TPLL = 2 ms (approx). These intervals are not shown to scale.

### 3.3.3 RC_IDLE MODE

In RC_IDLE mode, the CPU is disabled, but the peripherals continue to be clocked from the internal oscillator block using the INTOSC multiplexer. This mode allows for controllable power conservation during Idle periods.

This mode is entered by setting the IDLEN bit, setting SCS1 (SCS0 is ignored) and executing a SLEEP instruction. The INTOSC multiplexer may be used to select a higher clock frequency by modifying the IRCF bits before executing the SLEEP instruction. When the clock source is switched to the INTOSC multiplexer (see Figure 3-7), the primary oscillator is shut down and the OSTS bit is cleared.

If the IRCF bits are set to a non-zero value (thus, enabling the INTOSC output), the IOFS bit becomes set after the INTOSC output becomes stable, in about 1 ms. Clocks to the peripherals continue while the INTOSC source stabilizes. If the IRCF bits were previously at a non-zero value before the SLEEP instruction was executed and the INTOSC source was already stable, the IOFS bit will remain set. If the IRCF bits are all clear, the INTOSC output is not enabled and the IOFS bit will remain clear; there will be no indication of the current clock source.

When a wake event occurs, the peripherals continue to be clocked from the INTOSC multiplexer. After a 10 μs delay following the wake event, the CPU begins executing code, being clocked by the INTOSC multiplexer. The microcontroller operates in RC_RUN mode until the primary clock becomes ready. When the primary clock becomes ready, a clock switchback to the primary clock occurs (see Figure 3-8). When the clock switch is complete, the IOFS bit is cleared, the OSTS bit is set and the primary clock is providing the system clock. The IDLEN and SCS bits are not affected by the wake-up. The INTRC source will continue to run if either the WDT or the Fail-Safe Clock Monitor is enabled.

**FIGURE 3-7: TIMING TRANSITION TO RC_IDLE MODE**

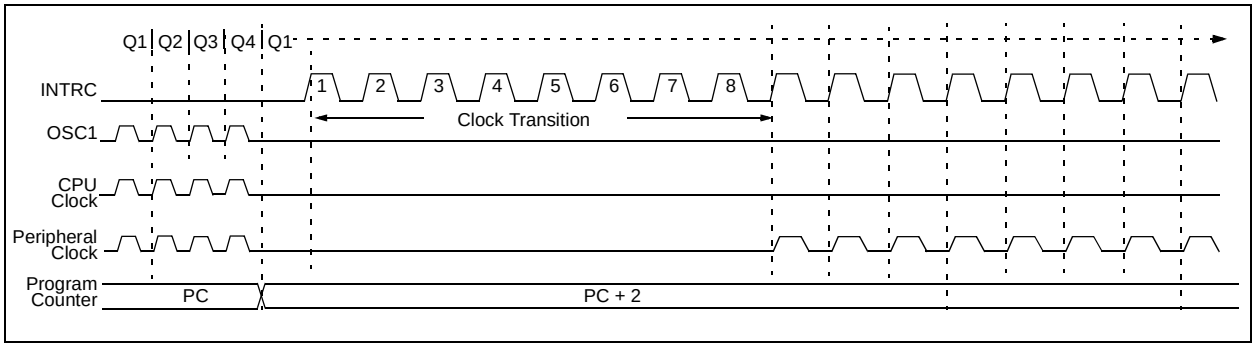
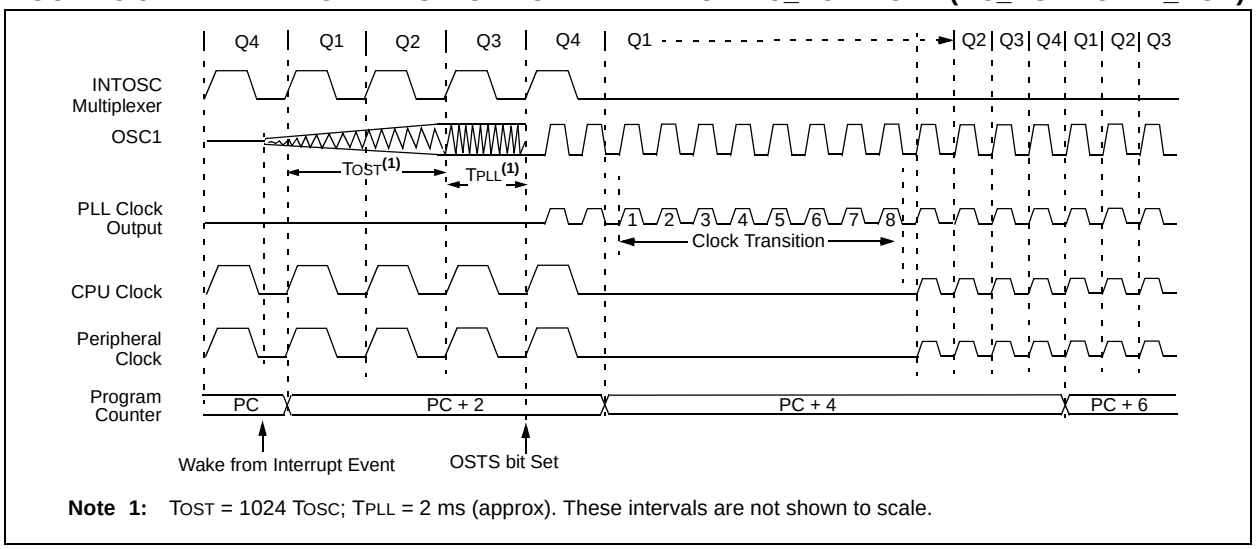



**FIGURE 3-8: TIMING TRANSITION FOR WAKE FROM RC_RUN MODE (RC_RUN TO PRI_RUN)**




**Note 1:** TOST = 1024 TOSC; TPLL = 2 ms (approx). These intervals are not shown to scale.


© 2002-2015 Microchip Technology Inc.

## 3.4 Run Modes

If the IDLEN bit is clear when a SLEEP instruction is executed, the CPU and peripherals are both clocked from the source selected using the SCS1:SCS0 bits. While these operating modes may not afford the power conservation of Idle or Sleep modes, they do allow the device to continue executing instructions by using a lower frequency clock source. RC_RUN mode also offers the possibility of executing code at a frequency greater than the primary clock.

Wake-up from a power managed Run mode can be triggered by an interrupt, or any Reset, to return to full-power operation. As the CPU is executing code in Run modes, several additional exits from Run modes are possible. They include exit to Sleep mode, exit to a corresponding Idle mode and exit by executing a RESET instruction. While the device is in any of the power managed Run modes, a WDT time-out will result in a WDT Reset.

### 3.4.1 PRI_RUN MODE

The PRI_RUN mode is the normal full-power execution mode. If the SLEEP instruction is never executed, the microcontroller operates in this mode (a SLEEP instruction is executed to enter all other power managed modes). All other power managed modes exit to PRI_RUN mode when an interrupt or WDT time-out occur.

There is no entry to PRI_RUN mode. The OSTS bit is set. The IOFS bit may be set if the internal oscillator block is the primary clock source (see **Section 2.7.1 "Oscillator Control Register"**).

### 3.4.2 SEC_RUN MODE

The SEC_RUN mode is the compatible mode to the "clock switching" feature offered in other PIC18 devices. In this mode, the CPU and peripherals are clocked from the Timer1 oscillator. This gives users the option of lower power consumption while still using a high accuracy clock source.

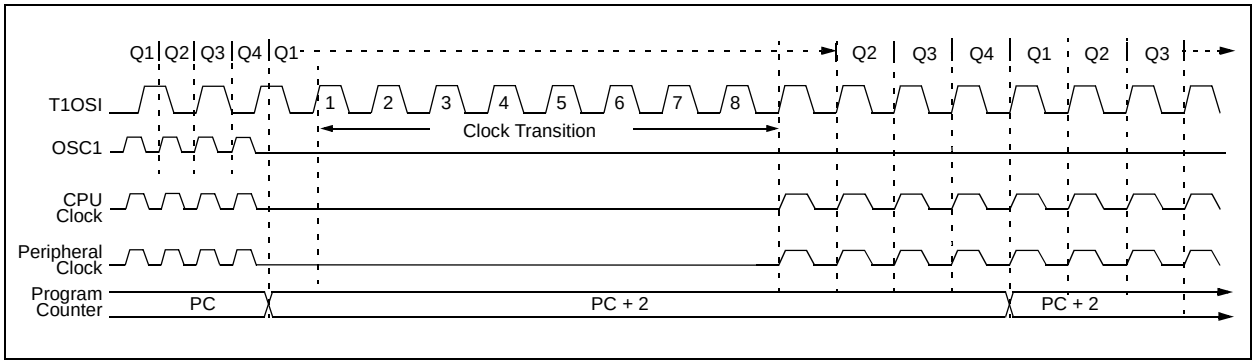
SEC_RUN mode is entered by clearing the IDLEN bit, setting SCS1:SCS0 = 01 and executing a SLEEP instruction. The system clock source is switched to the Timer1 oscillator (see Figure 3-9), the primary oscillator is shut down, the T1RUN bit (T1CON<6>) is set and the OSTS bit is cleared.

| | |
|---|---|
| **Note:** | The Timer1 oscillator should already be running prior to entering SEC_RUN mode. If the T1OSCEN bit is not set when the SLEEP instruction is executed, the SLEEP instruction will be ignored and entry to SEC_RUN mode will not occur. If the Timer1 oscillator is enabled, but not yet running, system clocks will be delayed until the oscillator has started; in such situations, initial oscillator operation is far from stable and unpredictable operation may result. |

When a wake event occurs, the peripherals and CPU continue to be clocked from the Timer1 oscillator while the primary clock is started. When the primary clock becomes ready, a clock switchback to the primary clock occurs (see Figure 3-6). When the clock switch is complete, the T1RUN bit is cleared, the OSTS bit is set and the primary clock is providing the system clock. The IDLEN and SCS bits are not affected by the wake-up. The Timer1 oscillator continues to run.

Firmware can force an exit from SEC_RUN mode. By clearing the T1OSCEN bit (T1CON<3>), an exit from SEC_RUN back to normal full-power operation is triggered. The Timer1 oscillator will continue to run and provide the system clock, even though the T1OSCEN bit is cleared. The primary clock is started. When the primary clock becomes ready, a clock switchback to the primary clock occurs (see Figure 3-6). When the clock switch is complete, the Timer1 oscillator is disabled, the T1RUN bit is cleared, the OSTS bit is set and the primary clock is providing the system clock. The IDLEN and SCS bits are not affected by the wake-up.

**FIGURE 3-9: TIMING TRANSITION FOR ENTRY TO SEC_RUN MODE**



### 3.4.3 RC_RUN MODE

In RC_RUN mode, the CPU and peripherals are clocked from the internal oscillator block using the INTOSC multiplexer and the primary clock is shut down. When using the INTRC source, this mode provides the best power conservation of all the Run modes, while still executing code. It works well for user applications which are not highly timing sensitive, or do not require high-speed clocks at all times.

If the primary clock source is the internal oscillator block (either of the INTIO1 or INTIO2 oscillators), there are no distinguishable differences between PRI_RUN and RC_RUN modes during execution. However, a clock switch delay will occur during entry to and exit from RC_RUN mode. Therefore, if the primary clock source is the internal oscillator block, the use of RC_RUN mode is not recommended.

This mode is entered by clearing the IDLEN bit, setting SCS1 (SCS0 is ignored) and executing a SLEEP instruction. The IRCF bits may select the clock frequency before the SLEEP instruction is executed. When the clock source is switched to the INTOSC multiplexer (see Figure 3-10), the primary oscillator is shut down and the OSTS bit is cleared.

The IRCF bits may be modified at any time to immediately change the system clock speed. Executing a SLEEP instruction is not required to select a new clock frequency from the INTOSC multiplexer.

| **Note:** | Caution should be used when modifying a single IRCF bit. If VDD is less than 3V, it is possible to select a higher clock speed than is supported by the low VDD. Improper device operation may result if the VDD/FOSC specifications are violated. |
|---|---|

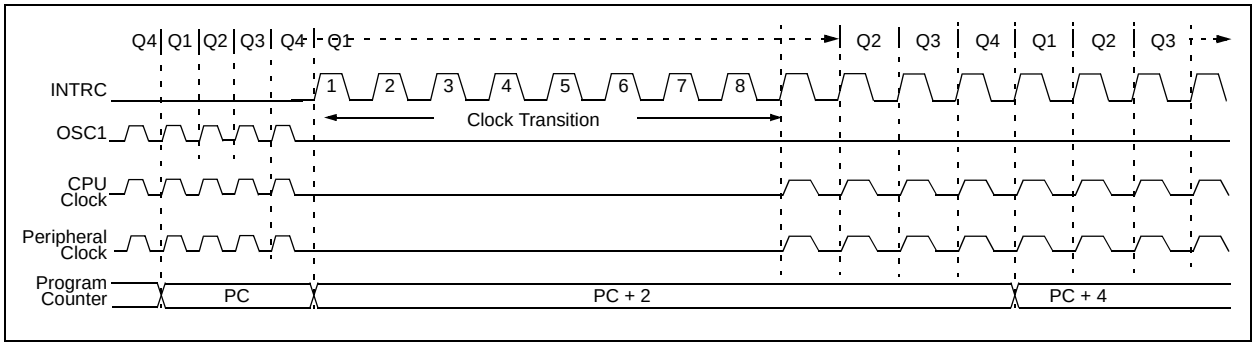
If the IRCF bits are all clear, the INTOSC output is not enabled and the IOFS bit will remain clear; there will be no indication of the current clock source. The INTRC source is providing the system clocks.

If the IRCF bits are changed from all clear (thus, enabling the INTOSC output), the IOFS bit becomes set after the INTOSC output becomes stable. Clocks to the system continue while the INTOSC source stabilizes, in approximately 1 ms.

If the IRCF bits were previously at a non-zero value before the SLEEP instruction was executed and the INTOSC source was already stable, the IOFS bit will remain set.

When a wake event occurs, the system continues to be clocked from the INTOSC multiplexer while the primary clock is started. When the primary clock becomes ready, a clock switch to the primary clock occurs (see Figure 3-8). When the clock switch is complete, the IOFS bit is cleared, the OSTS bit is set and the primary clock is providing the system clock. The IDLEN and SCS bits are not affected by the wake-up. The INTRC source will continue to run if either the WDT or the Fail-Safe Clock Monitor is enabled.

**FIGURE 3-10: TIMING TRANSITION TO RC_RUN MODE**





© 2002-2015 Microchip Technology Inc.

### 3.4.4 EXIT TO IDLE MODE

An exit from a power managed Run mode to its corresponding Idle mode is executed by setting the IDLEN bit and executing a SLEEP instruction. The CPU is halted at the beginning of the instruction following the SLEEP instruction. There are no changes to any of the clock source status bits (OSTS, IOFS or T1RUN). While the CPU is halted, the peripherals continue to be clocked from the previously selected clock source.

### 3.4.5 EXIT TO SLEEP MODE

An exit from a power managed Run mode to Sleep mode is executed by clearing the IDLEN and SCS1:SCS0 bits and executing a SLEEP instruction. The code is no different than the method used to invoke Sleep mode from the normal operating (full-power) mode.

The primary clock and internal oscillator block are disabled. The INTRC will continue to operate if the WDT is enabled. The Timer1 oscillator will continue to run, if enabled in the T1CON register (Register 12-1). All clock source Status bits are cleared (OSTS, IOFS and T1RUN).

## 3.5 Wake from Power Managed Modes

An exit from any of the power managed modes is triggered by an interrupt, a Reset or a WDT time-out. This section discusses the triggers that cause exits from power managed modes. The clocking subsystem actions are discussed in each of the power managed modes (see Sections 3.2 through 3.4).

| | |
|---|---|
| **Note:** | If application code is timing sensitive, it should wait for the OSTS bit to become set before continuing. Use the interval during the low-power exit sequence (before OSTS is set) to perform timing insensitive “housekeeping” tasks. |

Device behavior during Low-Power mode exits is summarized in Table 3-3.

### 3.5.1 EXIT BY INTERRUPT

Any of the available interrupt sources can cause the device to exit a power managed mode and resume full-power operation. To enable this functionality, an interrupt source must be enabled by setting its enable bit in one of the INTCON or PIE registers. The exit sequence is initiated when the corresponding interrupt flag bit is set. On all exits from Low-Power mode by interrupt, code execution branches to the interrupt vector if the GIE/GIEH bit (INTCON<7>) is set. Otherwise, code execution continues or resumes without branching (see **Section 9.0 “Interrupts”**).

**TABLE 3-3: ACTIVITY AND EXIT DELAY ON WAKE FROM SLEEP MODE OR ANY IDLE MODE (BY CLOCK SOURCES)**

| Clock in Power Managed Mode | Primary System Clock | Power Managed Mode Exit Delay | Clock Ready Status Bit (OSCCON) | Activity during Wake-up from Power Managed Mode | |
|---|---|---|---|---|---|
| | | | | Exit by Interrupt | Exit by Reset |
| Primary System Clock (PRI_IDLE mode) | LP, XT, HS | 5-10 μs(5) | OSTS | CPU and peripherals clocked by primary clock and executing instructions. | Not clocked or Two-Speed Start-up (if enabled)(3). |
| | HSPLL | | | | |
| | EC, RC, INTRC(1) | | — | | |
| | INTOSC(2) | | IOFS | | |
| T1OSC or INTRC(1) | LP, XT, HS | OST | OSTS | CPU and peripherals clocked by selected power managed mode clock and executing instructions until primary clock source becomes ready. | |
| | HSPLL | OST + 2 ms | | | |
| | EC, RC, INTRC(1) | 5-10 μs(5) | — | | |
| | INTOSC(2) | 1 ms(4) | IOFS | | |
| INTOSC(2) | LP, XT, HS | OST | OSTS | | |
| | HSPLL | OST + 2 ms | | | |
| | EC, RC, INTRC(1) | 5-10 μs(5) | — | | |
| | INTOSC(2) | None | IOFS | | |
| Sleep mode | LP, XT, HS | OST | OSTS | Not clocked or Two-Speed Start-up (if enabled) until primary clock source becomes ready(3). | |
| | HSPLL | OST + 2 ms | | | |
| | EC, RC, INTRC(1) | 5-10 μs(5) | — | | |
| | INTOSC(2) | 1 ms(4) | IOFS | | |

**Note 1:** In this instance, refers specifically to the INTRC clock source.

**2:** Includes both the INTOSC 8 MHz source and postscaler derived frequencies.

**3:** Two-Speed Start-up is covered in greater detail in **Section 19.3 “Two-Speed Start-up”**.

**4:** Execution continues during the INTOSC stabilization period.

**5:** Required delay when waking from Sleep and all Idle modes. This delay runs concurrently with any other required delays (see **Section 3.3 “Idle Modes”**).


© 2002-2015 Microchip Technology Inc.

### 3.5.2 EXIT BY RESET

Normally, the device is held in Reset by the Oscillator Start-up Timer (OST) until the primary clock (defined in Configuration Register 1H) becomes ready. At that time, the OSTS bit is set and the device begins executing code.

Code execution can begin before the primary clock becomes ready. If either the Two-Speed Start-up (see **Section 19.3 "Two-Speed Start-up"**) or Fail-Safe Clock Monitor (see **Section 19.4 "Fail-Safe Clock Monitor"**) are enabled in Configuration Register 1H, the device may begin execution as soon as the Reset source has cleared. Execution is clocked by the INTOSC multiplexer driven by the internal oscillator block. Since the OSCCON register is cleared following all Resets, the INTRC clock source is selected. A higher speed clock may be selected by modifying the IRCF bits in the OSCCON register. Execution is clocked by the internal oscillator block until either the primary clock becomes ready, or a power managed mode is entered before the primary clock becomes ready; the primary clock is then shut down.

### 3.5.3 EXIT BY WDT TIME-OUT

A WDT time-out will cause different actions, depending on which power managed mode the device is in when the time-out occurs.

If the device is not executing code (all Idle modes and Sleep mode), the time-out will result in a wake from the power managed mode (see Sections 3.2 through 3.4).

If the device is executing code (all Run modes), the time-out will result in a WDT Reset (see **Section 19.2 "Watchdog Timer (WDT)"**).

The WDT timer and postscaler are cleared by executing a SLEEP or CLRWDT instruction, the loss of a currently selected clock source (if the Fail-Safe Clock Monitor is enabled) and modifying the IRCF bits in the OSCCON register if the internal oscillator block is the system clock source.

### 3.5.4 EXIT WITHOUT AN OSCILLATOR START-UP DELAY

Certain exits from power managed modes do not invoke the OST at all. These are:

- PRI_IDLE mode, where the primary clock source is not stopped; or
- the primary clock source is not any of LP, XT, HS or HSPLL modes.

In these cases, the primary clock source either does not require an oscillator start-up delay, since it is already running (PRI_IDLE), or normally does not require an oscillator start-up delay (RC, EC and INTIO Oscillator modes).

However, a fixed delay (approximately 10 μs) following the wake event is required when leaving Sleep and Idle modes. This delay is required for the CPU to prepare for execution. Instruction execution resumes on the first clock cycle following this delay.

## 3.6 INTOSC Frequency Drift

The factory calibrates the internal oscillator block output (INTOSC) for 8 MHz (see Table 22-6). However, this frequency may drift as VDD or temperature changes, which can affect the controller operation in a variety of ways.

It is possible to adjust the INTOSC frequency by modifying the value in the OSCTUNE register (Register 2-1). This has the side effect that the INTRC clock source frequency is also affected. However, the features that use the INTRC source often do not require an exact frequency. These features include the Fail-Safe Clock Monitor, the Watchdog Timer and the RC_RUN/RC_IDLE modes when the INTRC clock source is selected.

Being able to adjust the INTOSC requires knowing when an adjustment is required, in which direction it should be made and in some cases, how large a change is needed. Three examples follow but other techniques may be used.

© 2002-2015 Microchip Technology Inc.

### 3.6.1 EXAMPLE – EUSART

An adjustment may be indicated when the EUSART begins to generate framing errors, or receives data with errors while in Asynchronous mode. Framing errors indicate that the system clock frequency is too high – try decrementing the value in the OSCTUNE register to reduce the system clock frequency. Errors in data may suggest that the system clock speed is too low – increment OSCTUNE.

### 3.6.2 EXAMPLE – TIMERS

This technique compares system clock speed to some reference clock. Two timers may be used; one timer is clocked by the peripheral clock, while the other is clocked by a fixed reference source, such as the Timer1 oscillator.

Both timers are cleared, but the timer clocked by the reference generates interrupts. When an interrupt occurs, the internally clocked timer is read and both timers are cleared. If the internally clocked timer value is greater than expected, then the internal oscillator block is running too fast – decrement OSCTUNE.

### 3.6.3 EXAMPLE – CCP IN CAPTURE MODE

A CCP module can use free running Timer1 (or Timer3), clocked by the internal oscillator block and an external event with a known period (i.e., AC power frequency). The time of the first event is captured in the CCPRxH:CCPRxL registers and is recorded for use later. When the second event causes a capture, the time of the first event is subtracted from the time of the second event. Since the period of the external event is known, the time difference between events can be calculated.

If the measured time is much greater than the calculated time, the internal oscillator block is running too fast – decrement OSCTUNE. If the measured time is much less than the calculated time, the internal oscillator block is running too slow–increment OSCTUNE.


© 2002-2015 Microchip Technology Inc.

## 4.0 RESET

The PIC18F1220/1320 devices differentiate between various kinds of Reset:

a) Power-on Reset (POR)
b) MCLR Reset during normal operation
c) MCLR Reset during Sleep
d) Watchdog Timer (WDT) Reset (during execution)
e) Programmable Brown-out Reset (BOR)
f) RESET Instruction
g) Stack Full Reset
h) Stack Underflow Reset

Most registers are unaffected by a Reset. Their status is unknown on POR and unchanged by all other Resets. The other registers are forced to a "Reset state", depending on the type of Reset that occurred.

Most registers are not affected by a WDT wake-up, since this is viewed as the resumption of normal operation. Status bits from the RCON register (Register 5-2), RI, TO, PD, POR and BOR, are set or cleared differently in different Reset situations, as indicated in Table 4-2. These bits are used in software to determine the nature of the Reset. See Table 4-3 for a full description of the Reset states of all registers.

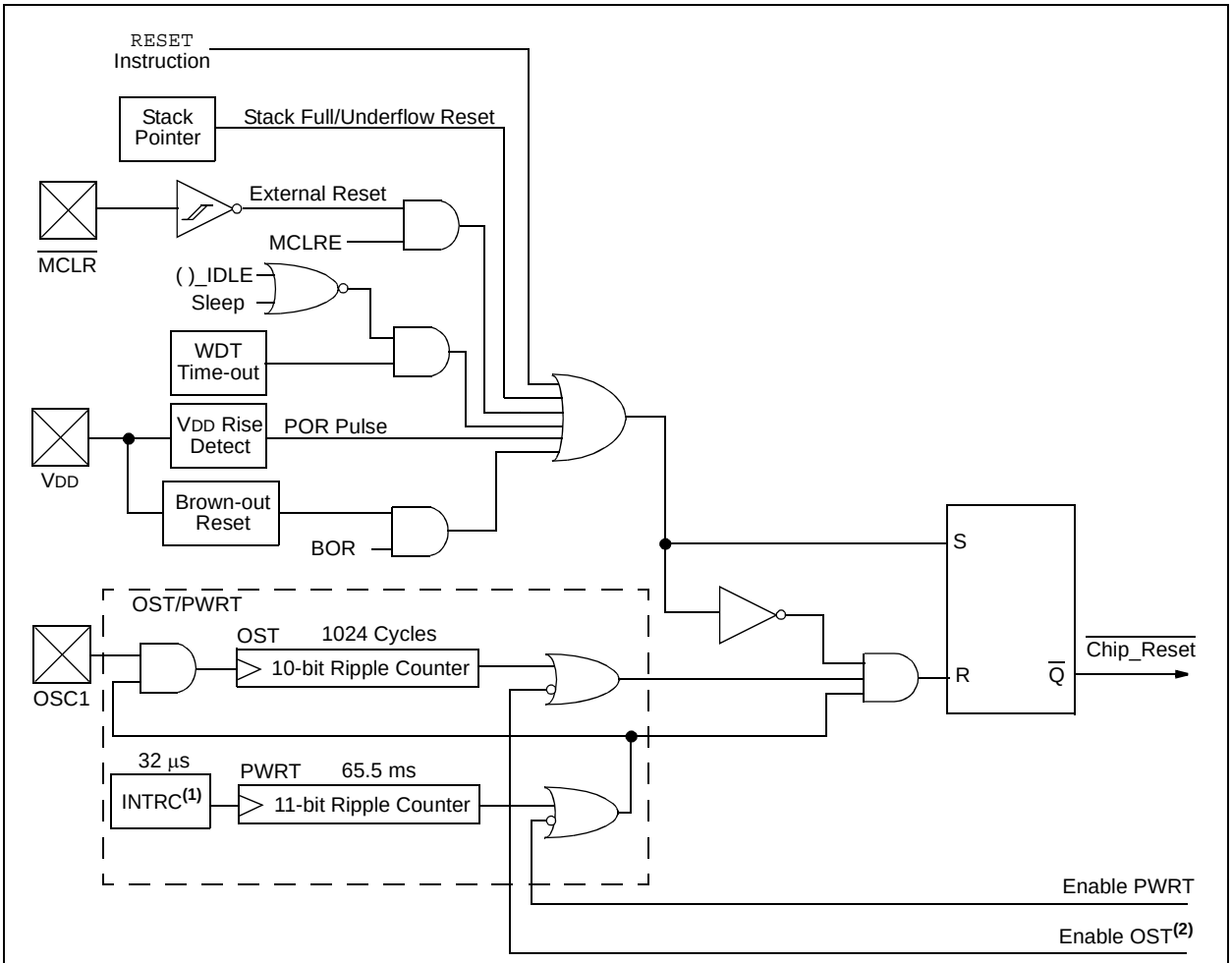
A simplified block diagram of the On-Chip Reset Circuit is shown in Figure 4-1.

The Enhanced MCU devices have a MCLR noise filter in the MCLR Reset path. The filter will detect and ignore small pulses.

The MCLR pin is not driven low by any internal Resets, including the WDT.

The MCLR input provided by the MCLR pin can be disabled with the MCLRE bit in Configuration Register 3H (CONFIG3H<7>).

**FIGURE 4-1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT**



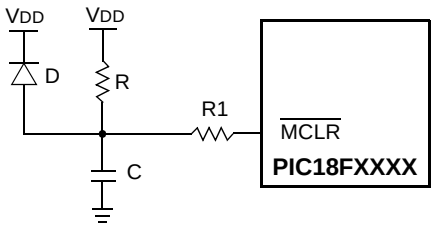

**Note 1:** This is a separate oscillator from the RC oscillator of the CLKI pin.
**2:** See Table 4-1 for time-out situations.

## 4.1 Power-on Reset (POR)

A Power-on Reset pulse is generated on-chip when VDD rise is detected. To take advantage of the POR circuitry, just tie the MCLR pin through a resistor (1k to 10 kΩ) to VDD. This will eliminate external RC components usually needed to create a Power-on Reset delay. A minimum rise rate for VDD is specified (parameter D004). For a slow rise time, see Figure 4-2.

When the device starts normal operation (i.e., exits the Reset condition), device operating parameters (voltage, frequency, temperature, etc.) must be met to ensure operation. If these conditions are not met, the device must be held in Reset until the operating conditions are met.

**FIGURE 4-2: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW VDD POWER-UP)**




**Note 1:** External Power-on Reset circuit is required only if the VDD power-up slope is too slow. The diode D helps discharge the capacitor quickly when VDD powers down.

**2:** R < 40 kΩ is recommended to make sure that the voltage drop across R does not violate the device's electrical specification.

**3:** R1 ≥ 1 kΩ will limit any current flowing into MCLR from external capacitor C, in the event of MCLR/VPP pin breakdown due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS).

## 4.2 Power-up Timer (PWRT)

The Power-up Timer (PWRT) of the PIC18F1220/1320 is an 11-bit counter, which uses the INTRC source as the clock input. This yields a count of 2048 x 32 µs = 65.6 ms. While the PWRT is counting, the device is held in Reset.

The power-up time delay will vary from chip-to-chip due to VDD, temperature and process variation. See DC parameter 33 for details.

The PWRT is enabled by clearing Configuration bit, PWRTEN.

## 4.3 Oscillator Start-up Timer (OST)

The Oscillator Start-up Timer (OST) provides a 1024 oscillator cycle (from OSC1 input) delay after the PWRT delay is over (parameter 33). This ensures that the crystal oscillator or resonator has started and stabilized.

The OST time-out is invoked only for XT, LP, HS and HSPLL modes and only on Power-on Reset, or on exit from most low-power modes.

## 4.4 PLL Lock Time-out

With the PLL enabled in its PLL mode, the time-out sequence following a Power-on Reset is slightly different from other oscillator modes. A portion of the Power-up Timer is used to provide a fixed time-out that is sufficient for the PLL to lock to the main oscillator frequency. This PLL lock time-out (TPLL) is typically 2 ms and follows the Oscillator Start-up Time-out.

## 4.5 Brown-out Reset (BOR)

A Configuration bit, BOR, can disable (if clear/programmed), or enable (if set) the Brown-out Reset circuitry. If VDD falls below VBOR (parameter D005) for greater than TBOR (parameter 35), the brown-out situation will reset the chip. A Reset may not occur if VDD falls below VBOR for less than TBOR. The chip will remain in Brown-out Reset until VDD rises above VBOR. If the Power-up Timer is enabled, it will be invoked after VDD rises above VBOR; it then will keep the chip in Reset for an additional time delay, TPWRT (parameter 33). If VDD drops below VBOR while the Power-up Timer is running, the chip will go back into a Brown-out Reset and the Power-up Timer will be initialized. Once VDD rises above VBOR, the Power-up Timer will execute the additional time delay. Enabling BOR Reset does not automatically enable the PWRT.

## 4.6 Time-out Sequence

On power-up, the time-out sequence is as follows: First, after the POR pulse has cleared, PWRT time-out is invoked (if enabled). Then, the OST is activated. The total time-out will vary based on oscillator configuration and the status of the PWRT. For example, in RC mode with the PWRT disabled, there will be no time-out at all. Figure 4-3, Figure 4-4, Figure 4-5, Figure 4-6 and Figure 4-7 depict time-out sequences on power-up.

Since the time-outs occur from the POR pulse, if MCLR is kept low long enough, all time-outs will expire. Bringing MCLR high will begin execution immediately (Figure 4-5). This is useful for testing purposes or to synchronize more than one PIC18FXXXX device operating in parallel.

Table 4-2 shows the Reset conditions for some Special Function Registers, while Table 4-3 shows the Reset conditions for all the registers.


© 2002-2015 Microchip Technology Inc.

**TABLE 4-1: TIME-OUT IN VARIOUS SITUATIONS**

| Oscillator Configuration | Power-up[2] and Brown-out | | Exit from Low-Power Mode |
|---|---|---|---|
| | $\overline{PWRTEN}$ = 0 | $\overline{PWRTEN}$ = 1 | |
| HSPLL | 66 ms[1] + 1024 TOSC + 2 ms[2] | 1024 TOSC + 2 ms[2] | 1024 TOSC + 2 ms[2] |
| HS, XT, LP | 66 ms[1] + 1024 TOSC | 1024 TOSC | 1024 TOSC |
| EC, ECIO | 66 ms[1] | 5-10 µs[3] | 5-10 µs[3] |
| RC, RCIO | 66 ms[1] | 5-10 µs[3] | 5-10 µs[3] |
| INTIO1, INTIO2 | 66 ms[1] | 5-10 µs[3] | 5-10 µs[3] |

**Note 1:** 66 ms (65.5 ms) is the nominal Power-up Timer (PWRT) delay.
**2:** 2 ms is the nominal time required for the 4x PLL to lock.
**3:** The program memory bias start-up time is always invoked on POR, wake-up from Sleep, or on any exit from power managed mode that disables the CPU and instruction execution.

**REGISTER 4-1: OSCCON: OSCILLATOR CONTROL REGISTER**

| R/W-0/0 | U-0 | U-0 | R/W-1/1 | R/W-1/1 | U-0 | R/W-1/1 | R/W-1/1 |
|---|---|---|---|---|---|---|---|
| IPEN | — | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

**Note:** Refer to **Section 5.14 “RCON Register”** for bit definitions.

**TABLE 4-2: STATUS BITS, THEIR SIGNIFICANCE AND THE INITIALIZATION CONDITION FOR RCON REGISTER**

| Condition | Program Counter | RCON Register | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ | STKFUL | STKUNF |
|---|---|---|---|---|---|---|---|---|---|
| Power-on Reset | 0000h | 0--1 1100 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| RESET Instruction | 0000h | 0--0 uuuu | 0 | u | u | u | u | u | u |
| Brown-out | 0000h | 0--1 11u- | 1 | 1 | 1 | u | 0 | u | u |
| $\overline{MCLR}$ during Power Managed Run modes | 0000h | 0--u 1uuu | u | 1 | u | u | u | u | u |
| $\overline{MCLR}$ during Power Managed Idle modes and Sleep | 0000h | 0--u 10uu | u | 1 | 0 | u | u | u | u |
| WDT Time-out during Full-Power or Power Managed Run | 0000h | 0--u 0uuu | u | 0 | u | u | u | u | u |
| $\overline{MCLR}$ during Full-Power Execution | 0000h | 0--u uuuu | u | u | u | u | u | u | u |
| Stack Full Reset (STVR = 1) | | | | | | | | 1 | u |
| Stack Underflow Reset (STVR = 1) | | | | | | | | u | 1 |
| Stack Underflow Error (not an actual Reset, STVR = 0) | 0000h | u--u uuuu | u | u | u | u | u | u | 1 |
| WDT Time-out during Power Managed Idle or Sleep | PC + 2 | u--u 00uu | u | 0 | 0 | u | u | u | u |
| Interrupt Exit from Power Managed modes | PC + 2 | u--u u0uu | u | u | 0 | u | u | u | u |

**Legend:** u = unchanged, x = unknown, – = unimplemented bit, read as ‘0’
**Note 1:** When the wake-up is due to an interrupt and the GIEH or GIEL bits are set, the PC is loaded with the interrupt vector (0x000008h or 0x000018h).

**TABLE 4-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS**

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | $\overline{\text{MCLR}}$ Resets WDT Reset RESET Instruction Stack Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| TOSU | 1220 | 1320 | ---0 0000 | ---0 0000 | ---0 uuuu(3) |
| TOSH | 1220 | 1320 | 0000 0000 | 0000 0000 | uuuu uuuu(3) |
| TOSL | 1220 | 1320 | 0000 0000 | 0000 0000 | uuuu uuuu(3) |
| STKPTR | 1220 | 1320 | 00-0 0000 | 00-0 0000 | uu-u uuuu(3) |
| PCLATU | 1220 | 1320 | ---0 0000 | ---0 0000 | ---u uuuu |
| PCLATH | 1220 | 1320 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PCL | 1220 | 1320 | 0000 0000 | 0000 0000 | PC + 2(2) |
| TBLPTRU | 1220 | 1320 | --00 0000 | --00 0000 | --uu uuuu |
| TBLPTRH | 1220 | 1320 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TBLPTRL | 1220 | 1320 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TABLAT | 1220 | 1320 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PRODH | 1220 | 1320 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PRODL | 1220 | 1320 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| INTCON | 1220 | 1320 | 0000 000x | 0000 000u | uuuu uuuu(1) |
| INTCON2 | 1220 | 1320 | 1111 -1-1 | 1111 -1-1 | uuuu -u-u(1) |
| INTCON3 | 1220 | 1320 | 11-0 0-00 | 11-0 0-00 | uu-u u-uu(1) |
| INDF0 | 1220 | 1320 | N/A | N/A | N/A |
| POSTINC0 | 1220 | 1320 | N/A | N/A | N/A |
| POSTDEC0 | 1220 | 1320 | N/A | N/A | N/A |
| PREINC0 | 1220 | 1320 | N/A | N/A | N/A |
| PLUSW0 | 1220 | 1320 | N/A | N/A | N/A |
| FSR0H | 1220 | 1320 | ---- 0000 | ---- 0000 | ---- uuuu |
| FSR0L | 1220 | 1320 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| WREG | 1220 | 1320 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| INDF1 | 1220 | 1320 | N/A | N/A | N/A |
| POSTINC1 | 1220 | 1320 | N/A | N/A | N/A |
| POSTDEC1 | 1220 | 1320 | N/A | N/A | N/A |
| PREINC1 | 1220 | 1320 | N/A | N/A | N/A |
| PLUSW1 | 1220 | 1320 | N/A | N/A | N/A |
| FSR1H | 1220 | 1320 | ---- 0000 | ---- 0000 | ---- uuuu |
| FSR1L | 1220 | 1320 | xxxx xxxx | uuuu uuuu | uuuu uuuu |

**Legend:** u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

**Note 1:** One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

**2:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

**3:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

**4:** See Table 4-2 for Reset value for specific condition.

**5:** Bits 6 and 7 of PORTA, LATA and TRISA are enabled, depending on the Oscillator mode selected. When not enabled as PORTA pins, they are disabled and read '0'.

**6:** Bit 5 of PORTA is enabled if $\overline{\text{MCLR}}$ is disabled.


© 2002-2015 Microchip Technology Inc.

**TABLE 4-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)**

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | $\overline{\text{MCLR}}$ Resets WDT Reset RESET Instruction Stack Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| BSR | 1220 | 1320 | ---- 0000 | ---- 0000 | ---- uuuu |
| INDF2 | 1220 | 1320 | N/A | N/A | N/A |
| POSTINC2 | 1220 | 1320 | N/A | N/A | N/A |
| POSTDEC2 | 1220 | 1320 | N/A | N/A | N/A |
| PREINC2 | 1220 | 1320 | N/A | N/A | N/A |
| PLUSW2 | 1220 | 1320 | N/A | N/A | N/A |
| FSR2H | 1220 | 1320 | ---- 0000 | ---- 0000 | ---- uuuu |
| FSR2L | 1220 | 1320 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| STATUS | 1220 | 1320 | ---x xxxx | ---u uuuu | ---u uuuu |
| TMR0H | 1220 | 1320 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TMR0L | 1220 | 1320 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| T0CON | 1220 | 1320 | 1111 1111 | 1111 1111 | uuuu uuuu |
| OSCCON | 1220 | 1320 | 0000 q000 | 0000 q000 | uuuu qquu |
| LVDCON | 1220 | 1320 | --00 0101 | --00 0101 | --uu uuuu |
| WDTCON | 1220 | 1320 | ---- ---0 | ---- ---0 | ---- ---u |
| RCON(4) | 1220 | 1320 | 0--1 11q0 | 0--q qquu | u--u qquu |
| TMR1H | 1220 | 1320 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TMR1L | 1220 | 1320 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| T1CON | 1220 | 1320 | 0000 0000 | u0uu uuuu | uuuu uuuu |
| TMR2 | 1220 | 1320 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PR2 | 1220 | 1320 | 1111 1111 | 1111 1111 | 1111 1111 |
| T2CON | 1220 | 1320 | -000 0000 | -000 0000 | -uuu uuuu |
| ADRESH | 1220 | 1320 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ADRESL | 1220 | 1320 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ADCON0 | 1220 | 1320 | 00-0 0000 | 00-0 0000 | uu-u uuuu |
| ADCON1 | 1220 | 1320 | -000 0000 | -000 0000 | -uuu uuuu |
| ADCON2 | 1220 | 1320 | 0-00 0000 | 0-00 0000 | u-uu uuuu |
| CCPR1H | 1220 | 1320 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR1L | 1220 | 1320 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCP1CON | 1220 | 1320 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PWM1CON | 1220 | 1320 | 0000 0000 | 0000 0000 | uuuu uuuu |
| ECCPAS | 1220 | 1320 | 0000 0000 | 0000 0000 | uuuu uuuu |

**Legend:** u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

**Note 1:** One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

**2:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

**3:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

**4:** See Table 4-2 for Reset value for specific condition.

**5:** Bits 6 and 7 of PORTA, LATA and TRISA are enabled, depending on the Oscillator mode selected. When not enabled as PORTA pins, they are disabled and read '0'.

**6:** Bit 5 of PORTA is enabled if $\overline{\text{MCLR}}$ is disabled.

© 2002-2015 Microchip Technology Inc.

**TABLE 4-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)**

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | $\overline{\text{MCLR}}$ Resets WDT Reset RESET Instruction Stack Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| TMR3H | 1220 | 1320 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TMR3L | 1220 | 1320 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| T3CON | 1220 | 1320 | 0-00 0000 | u-uu uuuu | u-uu uuuu |
| SPBRGH | 1220 | 1320 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SPBRG | 1220 | 1320 | 0000 0000 | 0000 0000 | uuuu uuuu |
| RCREG | 1220 | 1320 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TXREG | 1220 | 1320 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TXSTA | 1220 | 1320 | 0000 0010 | 0000 0010 | uuuu uuuu |
| RCSTA | 1220 | 1320 | 0000 000x | 0000 000x | uuuu uuuu |
| BAUDCTL | 1220 | 1320 | -1-1 0-00 | -1-1 0-00 | -u-u u-uu |
| EEADR | 1220 | 1320 | 0000 0000 | 0000 0000 | uuuu uuuu |
| EEDATA | 1220 | 1320 | 0000 0000 | 0000 0000 | uuuu uuuu |
| EECON2 | 1220 | 1320 | 0000 0000 | 0000 0000 | 0000 0000 |
| EECON1 | 1220 | 1320 | xx-0 x000 | uu-0 u000 | uu-0 u000 |
| IPR2 | 1220 | 1320 | 1--1 -11- | 1--1 -11- | u--u -uu- |
| PIR2 | 1220 | 1320 | 0--0 -00- | 0--0 -00- | u--u -uu-(1) |
| PIE2 | 1220 | 1320 | 0--0 -00- | 0--0 -00- | u--u -uu- |
| IPR1 | 1220 | 1320 | -111 -111 | -111 -111 | -uuu -uuu |
| PIR1 | 1220 | 1320 | -000 -000 | -000 -000 | -uuu -uuu(1) |
| PIE1 | 1220 | 1320 | -000 -000 | -000 -000 | -uuu -uuu |
| OSCTUNE | 1220 | 1320 | --00 0000 | --00 0000 | --uu uuuu |
| TRISB | 1220 | 1320 | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISA(5) | 1220 | 1320 | 11-1 1111(5) | 11-1 1111(5) | uu-u uuuu(5) |
| LATB | 1220 | 1320 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATA(5) | 1220 | 1320 | xx-x xxxx(5) | uu-u uuuu(5) | uu-u uuuu(5) |
| PORTB | 1220 | 1320 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTA(5,6) | 1220 | 1320 | xx0x 0000(5,6) | uu0u 0000(5,6) | uuuu uuuu(5,6) |

**Legend:** u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

**Note 1:** One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

**2:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

**3:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

**4:** See Table 4-2 for Reset value for specific condition.

**5:** Bits 6 and 7 of PORTA, LATA and TRISA are enabled, depending on the Oscillator mode selected. When not enabled as PORTA pins, they are disabled and read '0'.

**6:** Bit 5 of PORTA is enabled if $\overline{\text{MCLR}}$ is disabled.


© 2002-2015 Microchip Technology Inc.

**FIGURE 4-3: TIME-OUT SEQUENCE ON POWER-UP (MCLR TIED TO VDD, VDD RISE < TPWRT)**

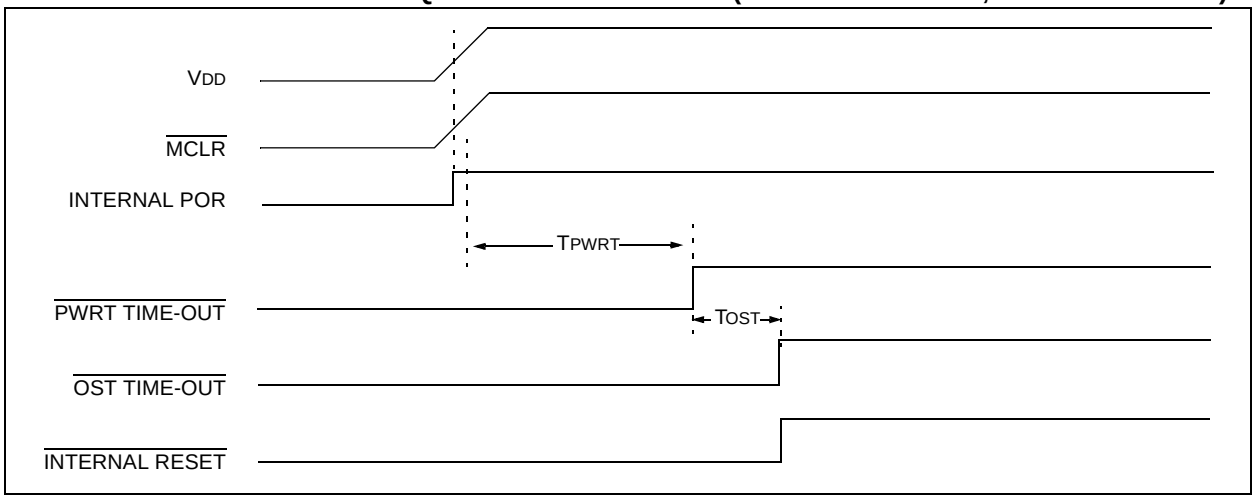



**FIGURE 4-4: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 1**

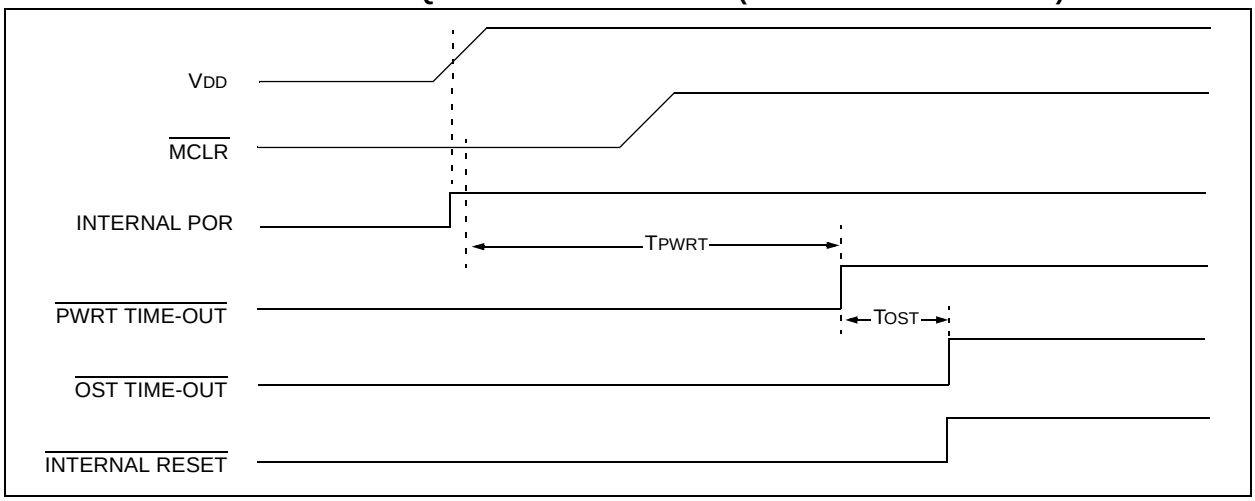



**FIGURE 4-5: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 2**

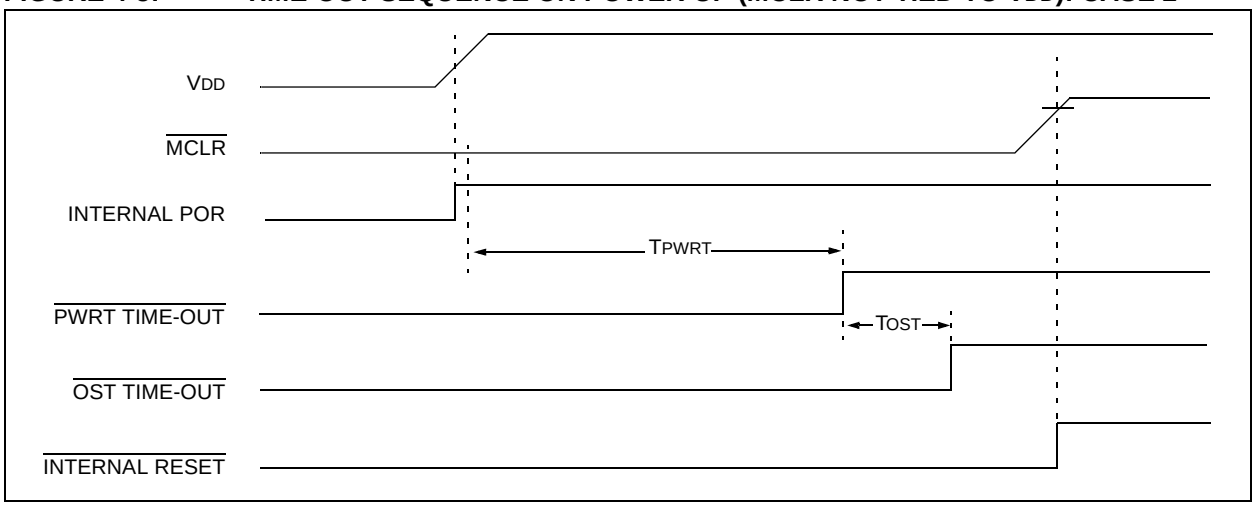


**FIGURE 4-6: SLOW RISE TIME ($\overline{MCLR}$ TIED TO $V_{DD}$, $V_{DD}$ RISE > $T_{PWRT}$)**

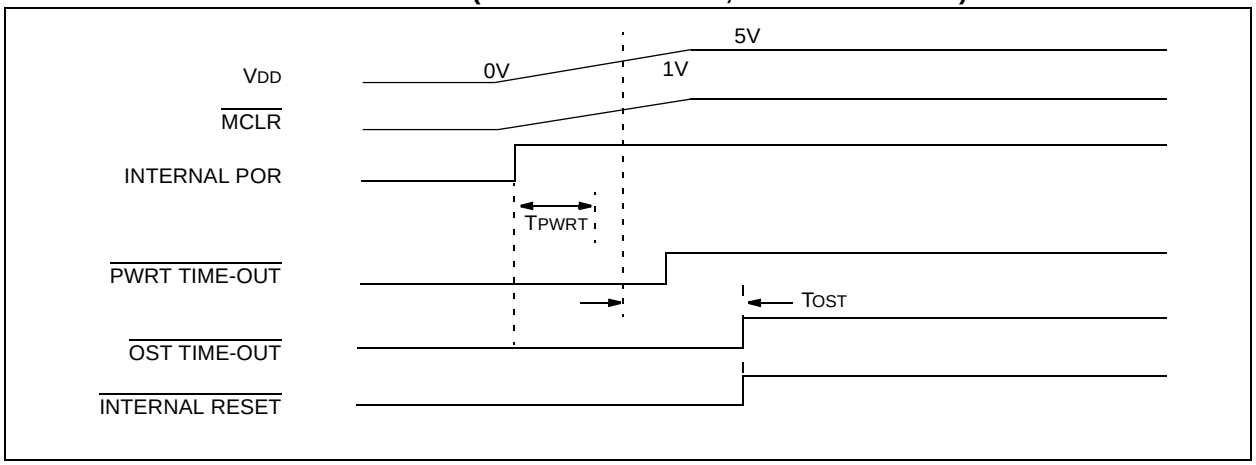
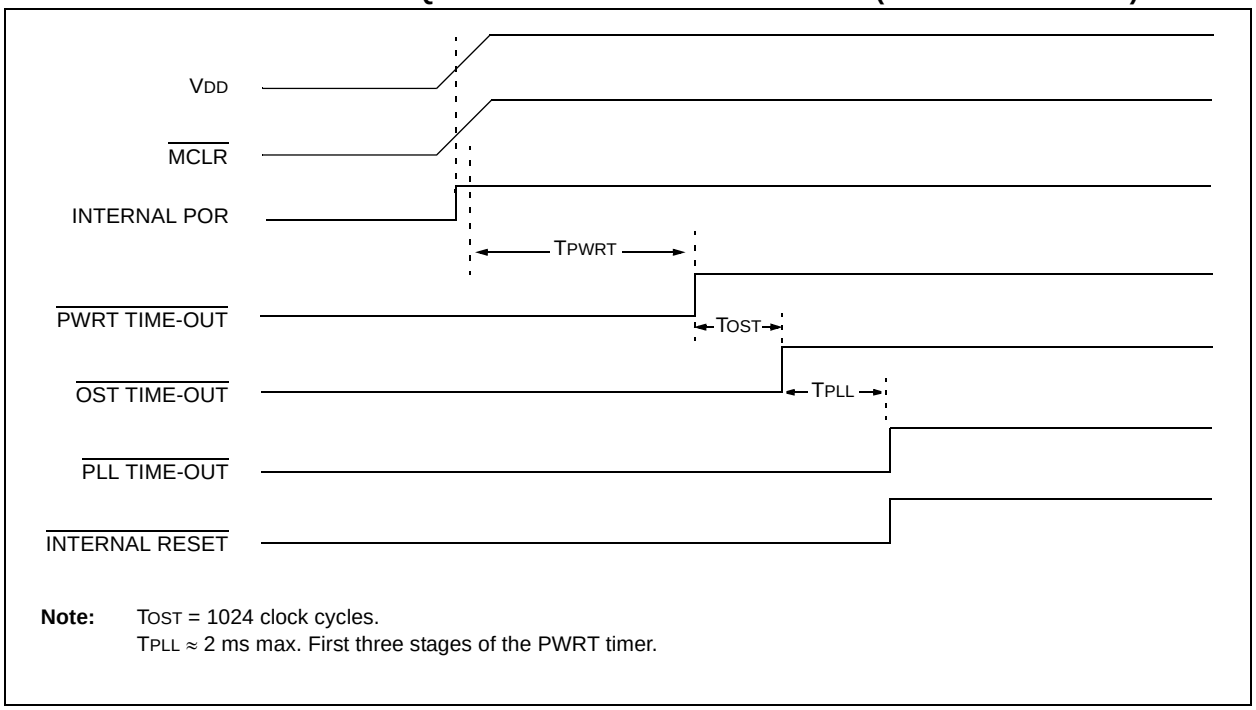



**FIGURE 4-7: TIME-OUT SEQUENCE ON POR W/PLL ENABLED ($\overline{MCLR}$ TIED TO $V_{DD}$)**




**Note:** $T_{OST}$ = 1024 clock cycles.
$T_{PLL} \approx$ 2 ms max. First three stages of the PWRT timer.


© 2002-2015 Microchip Technology Inc.

## 5.0 MEMORY ORGANIZATION

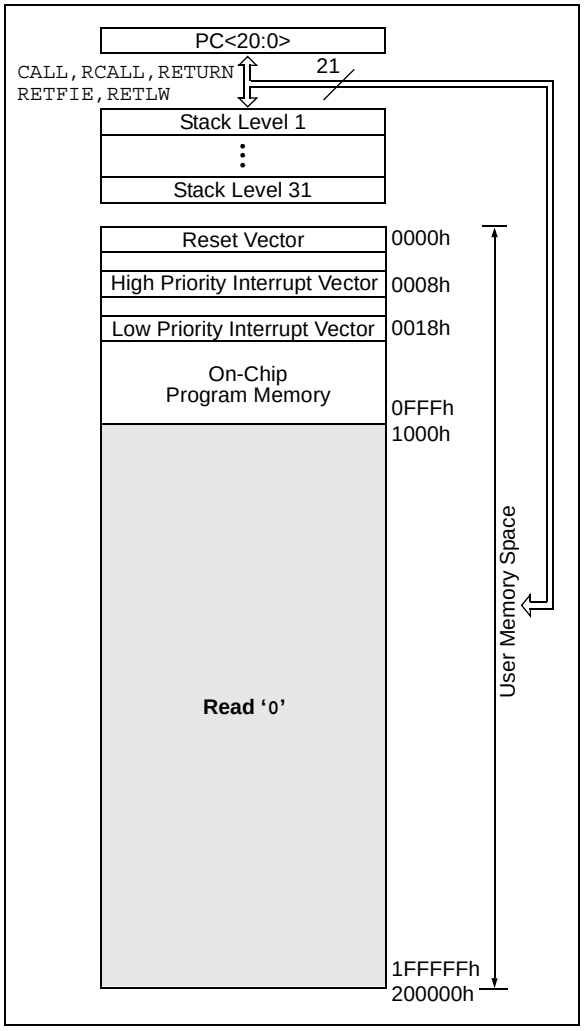
There are three memory types in Enhanced MCU devices. These memory types are:

- Program Memory
- Data RAM
- Data EEPROM

Data and program memory use separate busses, which allows for concurrent access of these types.

Additional detailed information for Flash program memory and data EEPROM is provided in **Section 6.0 "Flash Program Memory"** and **Section 7.0 "Data EEPROM Memory"**, respectively.

**FIGURE 5-1: PROGRAM MEMORY MAP AND STACK FOR PIC18F1220**




## 5.1 Program Memory Organization

A 21-bit program counter is capable of addressing the 2-Mbyte program memory space. Accessing a location between the physically implemented memory and the 2-Mbyte address will cause a read of all '0's (a NOP instruction).

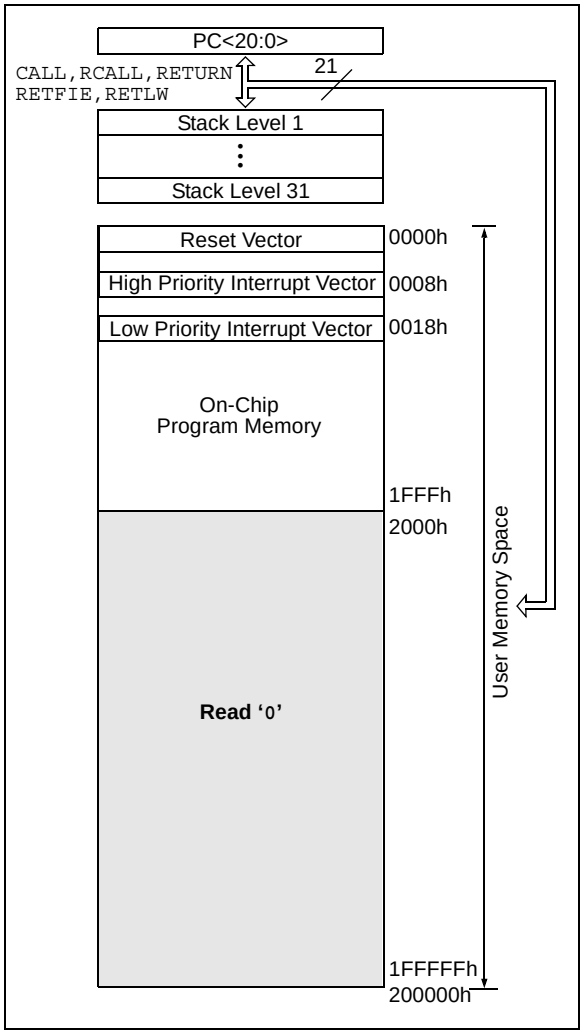
The PIC18F1220 has 4 Kbytes of Flash memory and can store up to 2,048 single-word instructions.

The PIC18F1320 has 8 Kbytes of Flash memory and can store up to 4,096 single-word instructions.

The Reset vector address is at 0000h and the interrupt vector addresses are at 0008h and 0018h.

The program memory maps for the PIC18F1220 and PIC18F1320 devices are shown in Figure 5-1 and Figure 5-2, respectively.

**FIGURE 5-2: PROGRAM MEMORY MAP AND STACK FOR PIC18F1320**



## 5.2 Return Address Stack

The return address stack allows any combination of up to 31 program calls and interrupts to occur. The PC (Program Counter) is pushed onto the stack when a CALL or RCALL instruction is executed, or an interrupt is Acknowledged. The PC value is pulled off the stack on a RETURN, RETLW or a RETFIE instruction. PCLATU and PCLATH are not affected by any of the RETURN or CALL instructions.

The stack operates as a 31-word by 21-bit RAM and a 5-bit Stack Pointer, with the Stack Pointer initialized to 00000B after all Resets. There is no RAM associated with Stack Pointer, 00000B. This is only a Reset value. During a CALL type instruction, causing a push onto the stack, the Stack Pointer is first incremented and the RAM location pointed to by the Stack Pointer (STKPTR) register is written with the contents of the PC (already pointing to the instruction following the CALL). During a RETURN type instruction, causing a pop from the stack, the contents of the RAM location pointed to by the STKPTR are transferred to the PC and then the Stack Pointer is decremented.

The stack space is not part of either program or data space. The Stack Pointer is readable and writable and the address on the top of the stack is readable and writable through the top-of-stack Special File Registers. Data can also be pushed to or popped from the stack using the top-of-stack SFRs. Status bits indicate if the stack is full, has overflowed or underflowed.

### 5.2.1 TOP-OF-STACK ACCESS

The top of the stack is readable and writable. Three register locations, TOSU, TOSH and TOSL, hold the contents of the stack location pointed to by the STKPTR register (Figure 5-3). This allows users to implement a software stack if necessary. After a CALL, RCALL or interrupt, the software can read the pushed value by reading the TOSU, TOSH and TOSL registers. These values can be placed on a user defined software stack. At return time, the software can replace the TOSU, TOSH and TOSL and do a return.

The user must disable the Global Interrupt Enable bits while accessing the stack to prevent inadvertent stack corruption.

### 5.2.2 RETURN STACK POINTER (STKPTR)

The STKPTR register (Register 5-1) contains the Stack Pointer value, the STKFUL (Stack Full) Status bit and the STKUNF (Stack Underflow) Status bits. The value of the Stack Pointer can be 0 through 31. The Stack Pointer increments before values are pushed onto the stack and decrements after values are popped off the stack. At Reset, the Stack Pointer value will be zero. The user may read and write the Stack Pointer value. This feature can be used by a Real-Time Operating System for return stack maintenance.

After the PC is pushed onto the stack 31 times (without popping any values off the stack), the STKFUL bit is set. The STKFUL bit is cleared by software or by a POR.

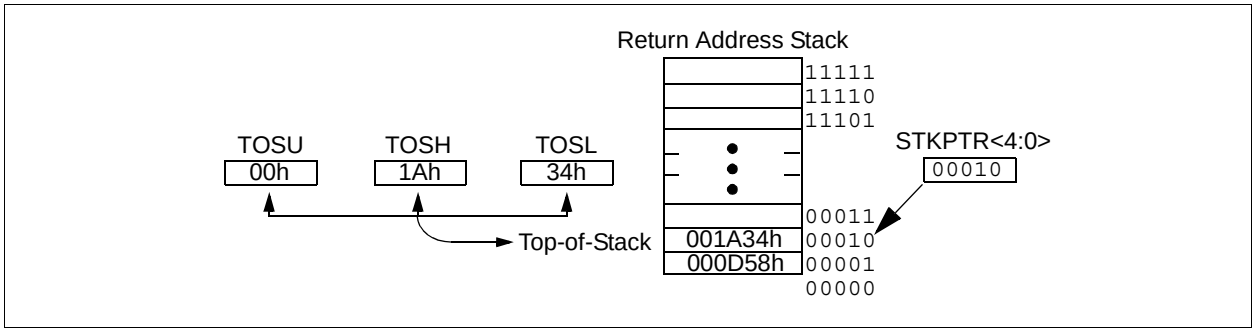
The action that takes place when the stack becomes full depends on the state of the STVR (Stack Overflow Reset Enable) Configuration bit. (Refer to **Section 19.1 "Configuration Bits"** for a description of the device Configuration bits.) If STVR is set (default), the 31st push will push the (PC + 2) value onto the stack, set the STKFUL bit and reset the device. The STKFUL bit will remain set and the Stack Pointer will be set to zero.

If STVR is cleared, the STKFUL bit will be set on the 31st push and the Stack Pointer will increment to 31. Any additional pushes will not overwrite the 31st push and STKPTR will remain at 31.

When the stack has been popped enough times to unload the stack, the next pop will return a value of zero to the PC and sets the STKUNF bit, while the Stack Pointer remains at zero. The STKUNF bit will remain set until cleared by software or a POR occurs.

| **Note:** | Returning a value of zero to the PC on an underflow has the effect of vectoring the program to the Reset vector, where the stack conditions can be verified and appropriate actions can be taken. This is not the same as a Reset, as the contents of the SFRs are not affected. |
|---|---|

**FIGURE 5-3: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS**





© 2002-2015 Microchip Technology Inc.

**REGISTER 5-1: STKPTR: STACK POINTER REGISTER**

| R/C-0/0 | R/C-0/0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| STKFUL(1) | STKUNF(1) | — | SP<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | C = Clearable only bit |

bit 7 **STKFUL:** Stack Full Flag bit(1)
1 = Stack became full or overflowed
0 = Stack has not become full or overflowed

bit 6 **STKUNF:** Stack Underflow Flag bit(1)
1 = Stack Underflow occurred
0 = Stack Underflow did not occur

bit 5 **Unimplemented:** Read as ‘0’

bit 4-0 **SP<4:0>:** Stack Pointer Location bits

**Note 1:** Bit 7 and bit 6 are cleared by user software or by a POR.

### 5.2.3 PUSH AND POP INSTRUCTIONS

Since the Top-of-Stack (TOS) is readable and writable, the ability to push values onto the stack and pull values off the stack, without disturbing normal program execution, is a desirable option. To push the current PC value onto the stack, a PUSH instruction can be executed. This will increment the Stack Pointer and load the current PC value onto the stack. TOSU, TOSH and TOSL can then be modified to place data or a return address on the stack.

The ability to pull the TOS value off of the stack and replace it with the value that was previously pushed onto the stack, without disturbing normal execution, is achieved by using the POP instruction. The POP instruction discards the current TOS by decrementing the Stack Pointer. The previous value pushed onto the stack then becomes the TOS value.

### 5.2.4 STACK FULL/UNDERFLOW RESETS

These Resets are enabled by programming the STVR bit in Configuration Register 4L. When the STVR bit is cleared, a full or underflow condition will set the appropriate STKFUL or STKUNF bit, but not cause a device Reset. When the STVR bit is set, a full or underflow condition will set the appropriate STKFUL or STKUNF bit and then cause a device Reset. The STKFUL or STKUNF bits are cleared by the user software or a Power-on Reset.

## 5.3 Fast Register Stack

A “fast return” option is available for interrupts. A fast register stack is provided for the Status, WREG and BSR registers and is only one in depth. The stack is not readable or writable and is loaded with the current value of the corresponding register when the processor vectors for an interrupt. The values in the registers are then loaded back into the working registers, if the RETFIE, FAST instruction is used to return from the interrupt.

All interrupt sources will push values into the stack registers. If both low and high priority interrupts are enabled, the stack registers cannot be used reliably to return from low priority interrupts. If a high priority interrupt occurs while servicing a low priority interrupt, the stack register values stored by the low priority interrupt will be overwritten. Users must save the key registers in software during a low priority interrupt.

If interrupt priority is not used, all interrupts may use the fast register stack for returns from interrupt.

If no interrupts are used, the fast register stack can be used to restore the Status, WREG and BSR registers at the end of a subroutine call. To use the fast register stack for a subroutine call, a CALL LABEL, FAST instruction must be executed to save the Status, WREG and BSR registers to the fast register stack. A RETURN, FAST instruction is then executed to restore these registers from the fast register stack.

Example 5-1 shows a source code example that uses the fast register stack during a subroutine call and return.

**EXAMPLE 5-1: FAST REGISTER STACK CODE EXAMPLE**

```
CALL SUB1, FAST    ;STATUS, WREG, BSR
                   ;SAVED IN FAST REGISTER
                   ;STACK
       •
       •

SUB1   •
       •
   RETURN, FAST    ;RESTORE VALUES SAVED
                   ;IN FAST REGISTER STACK
```

## 5.4 PCL, PCLATH and PCLATU

The Program Counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21 bits wide. The low byte, known as the PCL register, is both readable and writable. The high byte, or PCH register, contains the PC<15:8> bits and is not directly readable or writable. Updates to the PCH register may be performed through the PCLATH register. The upper byte is called PCU. This register contains the PC<20:16> bits and is not directly readable or writable. Updates to the PCU register may be performed through the PCLATU register.

The contents of PCLATH and PCLATU will be transferred to the program counter by any operation that writes PCL. Similarly, the upper two bytes of the program counter will be transferred to PCLATH and PCLATU by an operation that reads PCL. This is useful for computed offsets to the PC (see **Section 5.8.1 “Computed GOTO”**).

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the LSB of PCL is fixed to a value of ‘0’. The PC increments by two to address sequential instructions in the program memory.

The CALL, RCALL, GOTO and program branch instructions write to the program counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the program counter.


© 2002-2015 Microchip Technology Inc.

## 5.5 Clocking Scheme/Instruction Cycle

The clock input (from OSC1) is internally divided by four to generate four non-overlapping quadrature clocks, namely Q1, Q2, Q3 and Q4. Internally, the Program Counter (PC) is incremented every Q1, the instruction is fetched from the program memory and latched into the instruction register in Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in Figure 5-4.

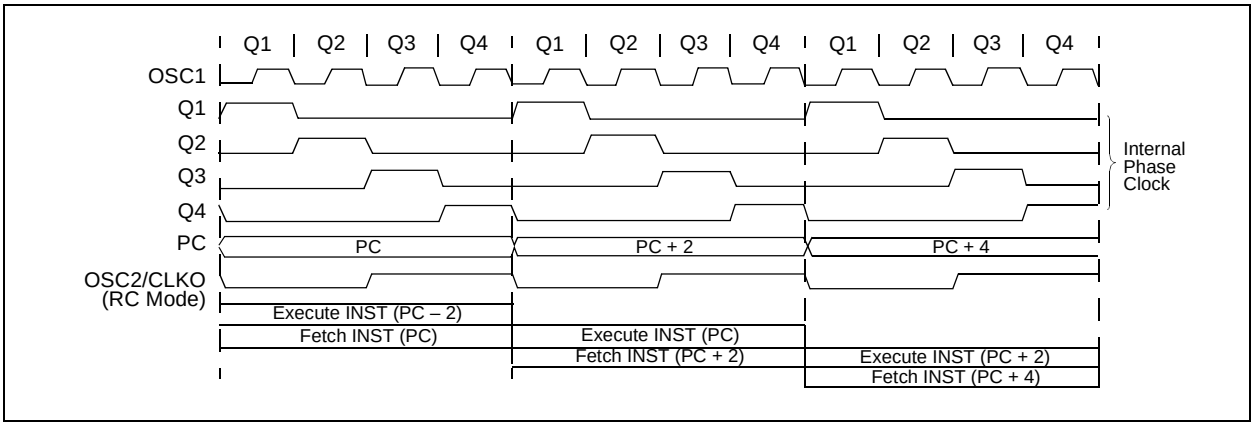
## 5.6 Instruction Flow/Pipelining

An “Instruction Cycle” consists of four Q cycles (Q1, Q2, Q3 and Q4). The instruction fetch and execute are pipelined such that fetch takes one instruction cycle, while decode and execute takes another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., `GOTO`), then two cycles are required to complete the instruction (Example 5-2).

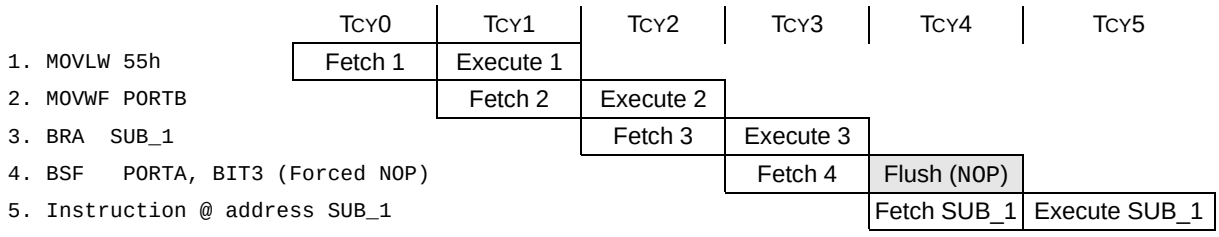
A fetch cycle begins with the Program Counter (PC) incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the “Instruction Register” (IR) in cycle Q1. This instruction is then decoded and executed during the Q2, Q3 and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

**FIGURE 5-4: CLOCK/INSTRUCTION CYCLE**




**EXAMPLE 5-2: INSTRUCTION PIPELINE FLOW**




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

## 5.7 Instructions in Program Memory

The program memory is addressed in bytes. Instructions are stored as two bytes or four bytes in program memory. The Least Significant Byte of an instruction word is always stored in a program memory location with an even address (LSB = 0). Figure 5-5 shows an example of how instruction words are stored in the program memory. To maintain alignment with instruction boundaries, the PC increments in steps of 2 and the LSB will always read '0' (see **Section 5.4 "PCL, PCLATH and PCLATU"**).

The CALL and GOTO instructions have the absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC<20:1>, which accesses the desired byte address in program memory. Instruction #2 in Figure 5-5 shows how the instruction 'GOTO 000006h' is encoded in the program memory. Program branch instructions, which encode a relative address offset, operate in the same manner. The offset value stored in a branch instruction represents the number of single-word instructions that the PC will be offset by. **Section 20.0 "Instruction Set Summary"** provides further details of the instruction set.

**FIGURE 5-5: INSTRUCTIONS IN PROGRAM MEMORY**

Program Memory Byte Locations →

Instruction 1: MOVLW 055h
Instruction 2: GOTO 000006h
Instruction 3: MOVFF 123h, 456h

| LSB = 1 | LSB = 0 | Word Address ↓ |
|---|---|---|
| | | 000000h |
| | | 000002h |
| | | 000004h |
| | | 000006h |
| 0Fh | 55h | 000008h |
| EFh | 03h | 00000Ah |
| F0h | 00h | 00000Ch |
| C1h | 23h | 00000Eh |
| F4h | 56h | 000010h |
| | | 000012h |
| | | 000014h |

### 5.7.1 TWO-WORD INSTRUCTIONS

PIC18F1220/1320 devices have four two-word instructions: MOVFF, CALL, GOTO and LFSR. The second word of these instructions has the 4 MSBs set to '1's and is decoded as a NOP instruction. The lower 12 bits of the second word contain data to be used by the instruction. If the first word of the instruction is executed, the data in the second word is accessed. If the second word of the instruction is executed by itself (first word was skipped), it will execute as a NOP. This action is necessary when the two-word instruction is preceded by a conditional instruction that results in a skip operation. A program example that demonstrates this concept is shown in Example 5-3. Refer to **Section 20.0 "Instruction Set Summary"** for further details of the instruction set.

**EXAMPLE 5-3: TWO-WORD INSTRUCTIONS**

**CASE 1:**

| Object Code | Source Code | | |
|---|---|---|---|
| 0110 0110 0000 0000 | TSTFSZ | REG1 | ; is RAM location 0? |
| 1100 0001 0010 0011 | MOVFF | REG1, REG2 | ; No, skip this word |
| 1111 0100 0101 0110 | | | ; Execute this word as a NOP |
| 0010 0100 0000 0000 | ADDWF | REG3 | ; continue code |

**CASE 2:**

| Object Code | Source Code | | |
|---|---|---|---|
| 0110 0110 0000 0000 | TSTFSZ | REG1 | ; is RAM location 0? |
| 1100 0001 0010 0011 | MOVFF | REG1, REG2 | ; Yes, execute this word |
| 1111 0100 0101 0110 | | | ; 2nd word of instruction |
| 0010 0100 0000 0000 | ADDWF | REG3 | ; continue code |


© 2002-2015 Microchip Technology Inc.

## 5.8 Look-up Tables

Look-up tables are implemented two ways:

- Computed GOTO
- Table Reads

### 5.8.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (see Example 5-4).

A look-up table can be formed with an ADDWF PCL instruction and a group of RETLW 0xnn instructions. WREG is loaded with an offset into the table before executing a call to that table. The first instruction of the called routine is the ADDWF PCL instruction. The next instruction executed will be one of the RETLW 0xnn instructions, that returns the value 0xnn to the calling function.

The offset value (in WREG) specifies the number of bytes that the program counter should advance and should be multiples of 2 (LSB = 0).

In this method, only one data byte may be stored in each instruction location and room on the return address stack is required.

**EXAMPLE 5-4: COMPUTED GOTO USING AN OFFSET VALUE**

```
        MOVFW   OFFSET
        CALL    TABLE
ORG     0xnn00
TABLE   ADDWF   PCL
        RETLW   0xnn
        RETLW   0xnn
        RETLW   0xnn
        .
        .
        .
```

### 5.8.2 TABLE READS/TABLE WRITES

A better method of storing data in program memory allows two bytes of data to be stored in each instruction location.

Look-up table data may be stored two bytes per program word by using table reads and writes. The Table Pointer (TBLPTR) register specifies the byte address and the Table Latch (TABLAT) register contains the data that is read from or written to program memory. Data is transferred to/from program memory, one byte at a time.

The table read/table write operation is discussed further in **Section 6.1 "Table Reads and Table Writes"**.

## 5.9 Data Memory Organization

The data memory is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes of data memory. Figure 5-6 shows the data memory organization for the PIC18F1220/1320 devices.

The data memory map is divided into as many as 16 banks that contain 256 bytes each. The lower four bits of the Bank Select Register (BSR<3:0>) select which bank will be accessed. The upper four bits for the BSR are not implemented.

The data memory contains Special Function Registers (SFR) and General Purpose Registers (GPR). The SFRs are used for control and status of the controller and peripheral functions, while GPRs are used for data storage and scratch pad operations in the user's application. The SFRs start at the last location of Bank 15 (FFFh) and extend towards F80h. Any remaining space beyond the SFRs in the Bank may be implemented as GPRs. GPRs start at the first location of Bank 0 and grow upwards. Any read of an unimplemented location will read as '0's.

The entire data memory may be accessed directly or indirectly. Direct addressing may require the use of the BSR register. Indirect addressing requires the use of a File Select Register (FSRn) and a corresponding Indirect File Operand (INDFn). Each FSR holds a 12-bit address value that can be used to access any location in the Data Memory map without banking. See **Section 5.12 "Indirect Addressing, INDF and FSR Registers"** for indirect addressing details.

The instruction set and architecture allow operations across all banks. This may be accomplished by indirect addressing or by the use of the MOVFF instruction. The MOVFF instruction is a two-word/two-cycle instruction that moves a value from one register to another.

To ensure that commonly used registers (SFRs and select GPRs) can be accessed in a single cycle, regardless of the current BSR values, an Access Bank is implemented. A segment of Bank 0 and a segment of Bank 15 comprise the Access RAM. **Section 5.10 "Access Bank"** provides a detailed description of the Access RAM.

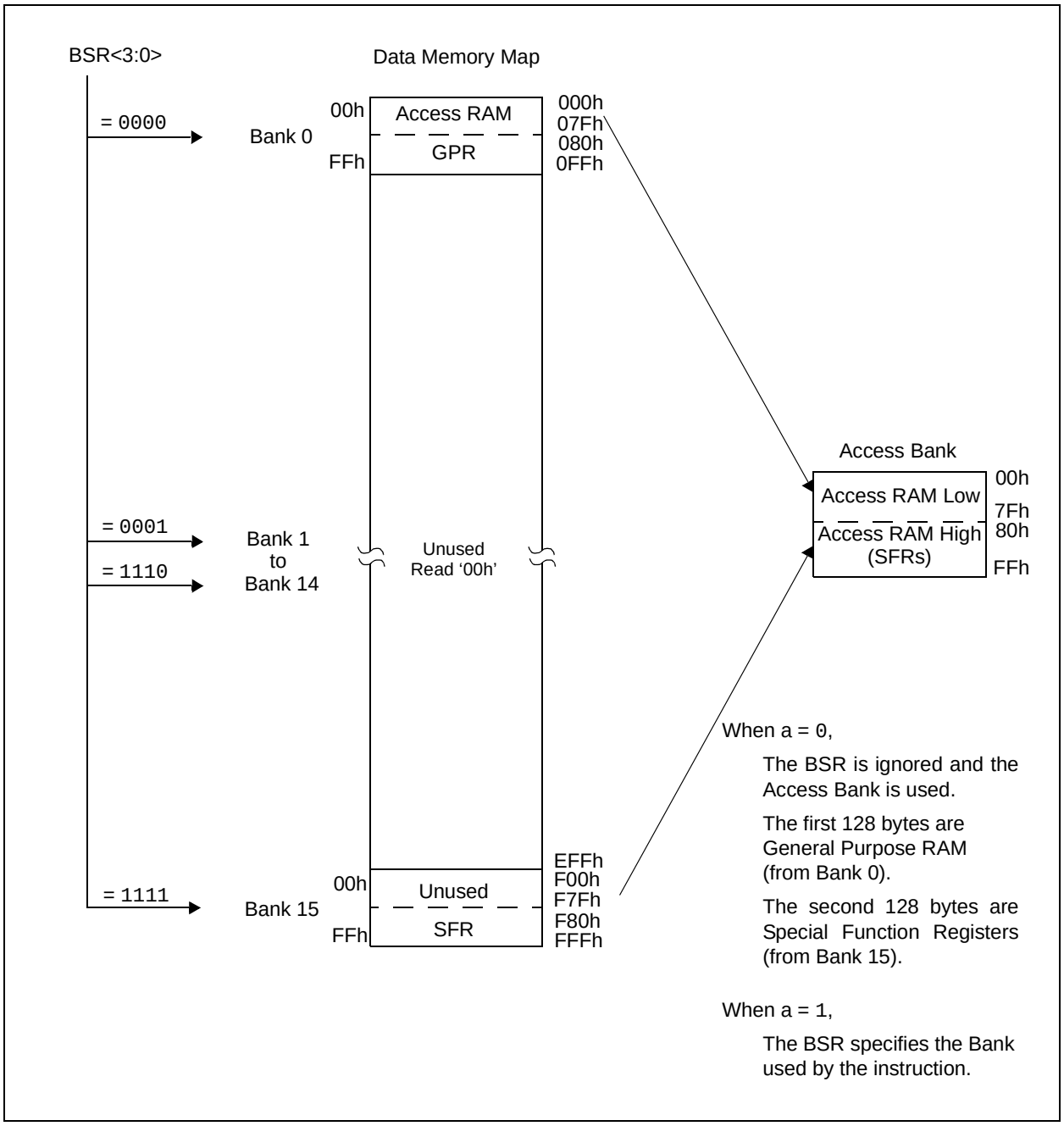
### 5.9.1 GENERAL PURPOSE REGISTER FILE

Enhanced MCU devices may have banked memory in the GPR area. GPRs are not initialized by a Power-on Reset and are unchanged on all other Resets.

Data RAM is available for use as GPR registers by all instructions. The second half of Bank 15 (F80h to FFFh) contains SFRs. All other banks of data memory contain GPRs, starting with Bank 0.

© 2002-2015 Microchip Technology Inc.

**FIGURE 5-6: DATA MEMORY MAP FOR PIC18F1220/1320 DEVICES**





© 2002-2015 Microchip Technology Inc.

### 5.9.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 5-1 and Table 5-2.

The SFRs can be classified into two sets: those associated with the “core” function and those related to the peripheral functions. Those registers related to the “core” are described in this section, while those related to the operation of the peripheral features are described in the section of that peripheral feature.

The SFRs are typically distributed among the peripherals whose functions they control.

The unused SFR locations will be unimplemented and read as ‘0’s.

**TABLE 5-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F1220/1320 DEVICES**

| Address | Name | Address | Name | Address | Name | Address | Name |
|---|---|---|---|---|---|---|---|
| FFFh | TOSU | FDFh | INDF2(2) | FBFh | CCPR1H | F9Fh | IPR1 |
| FFEh | TOSH | FDEh | POSTINC2(2) | FBEh | CCPR1L | F9Eh | PIR1 |
| FFDh | TOSL | FDDh | POSTDEC2(2) | FBDh | CCP1CON | F9Dh | PIE1 |
| FFCh | STKPTR | FDCh | PREINC2(2) | FBCh | — | F9Ch | — |
| FFBh | PCLATU | FDBh | PLUSW2(2) | FBBh | — | F9Bh | OSCTUNE |
| FFAh | PCLATH | FDAh | FSR2H | FBAh | — | F9Ah | — |
| FF9h | PCL | FD9h | FSR2L | FB9h | — | F99h | — |
| FF8h | TBLPTRU | FD8h | STATUS | FB8h | — | F98h | — |
| FF7h | TBLPTRH | FD7h | TMR0H | FB7h | PWM1CON | F97h | — |
| FF6h | TBLPTRL | FD6h | TMR0L | FB6h | ECCPAS | F96h | — |
| FF5h | TABLAT | FD5h | T0CON | FB5h | — | F95h | — |
| FF4h | PRODH | FD4h | — | FB4h | — | F94h | — |
| FF3h | PRODL | FD3h | OSCCON | FB3h | TMR3H | F93h | TRISB |
| FF2h | INTCON | FD2h | LVDCON | FB2h | TMR3L | F92h | TRISA |
| FF1h | INTCON2 | FD1h | WDTCON | FB1h | T3CON | F91h | — |
| FF0h | INTCON3 | FD0h | RCON | FB0h | SPBRGH | F90h | — |
| FEFh | INDF0(2) | FCFh | TMR1H | FAFh | SPBRG | F8Fh | — |
| FEEh | POSTINC0(2) | FCEh | TMR1L | FAEh | RCREG | F8Eh | — |
| FEDh | POSTDEC0(2) | FCDh | T1CON | FADh | TXREG | F8Dh | — |
| FECh | PREINC0(2) | FCCh | TMR2 | FACh | TXSTA | F8Ch | — |
| FEBh | PLUSW0(2) | FCBh | PR2 | FABh | RCSTA | F8Bh | — |
| FEAh | FSR0H | FCAh | T2CON | FAAh | BAUDCTL | F8Ah | LATB |
| FE9h | FSR0L | FC9h | — | FA9h | EEADR | F89h | LATA |
| FE8h | WREG | FC8h | — | FA8h | EEDATA | F88h | — |
| FE7h | INDF1(2) | FC7h | — | FA7h | EECON2 | F87h | — |
| FE6h | POSTINC1(2) | FC6h | — | FA6h | EECON1 | F86h | — |
| FE5h | POSTDEC1(2) | FC5h | — | FA5h | — | F85h | — |
| FE4h | PREINC1(2) | FC4h | ADRESH | FA4h | — | F84h | — |
| FE3h | PLUSW1(2) | FC3h | ADRESL | FA3h | — | F83h | — |
| FE2h | FSR1H | FC2h | ADCON0 | FA2h | IPR2 | F82h | — |
| FE1h | FSR1L | FC1h | ADCON1 | FA1h | PIR2 | F81h | PORTB |
| FE0h | BSR | FC0h | ADCON2 | FA0h | PIE2 | F80h | PORTA |

**Note 1:** Unimplemented registers are read as ‘0’.

**2:** This is not a physical register.

**TABLE 5-2: REGISTER FILE SUMMARY (PIC18F1220/1320)**

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Details on Page: |
|---|---|---|---|---|---|---|---|---|---|---|
| TOSU | — | — | — | Top-of-Stack Upper Byte (TOS<20:16>) | | | | | ---0 0000 | 34, 40 |
| TOSH | Top-of-Stack High Byte (TOS<15:8>) | | | | | | | | 0000 0000 | 34, 40 |
| TOSL | Top-of-Stack Low Byte (TOS<7:0>) | | | | | | | | 0000 0000 | 34, 40 |
| STKPTR | STKFUL | STKUNF | — | Return Stack Pointer | | | | | 00-0 0000 | 34, 41 |
| PCLATU | — | — | bit 21(3) | Holding Register for PC<20:16> | | | | | ---0 0000 | 34, 42 |
| PCLATH | Holding Register for PC<15:8> | | | | | | | | 0000 0000 | 34, 42 |
| PCL | PC Low Byte (PC<7:0>) | | | | | | | | 0000 0000 | 34, 42 |
| TBLPTRU | — | — | bit 21 | Program Memory Table Pointer Upper Byte (TBLPTR<20:16>) | | | | | --00 0000 | 34, 58 |
| TBLPTRH | Program Memory Table Pointer High Byte (TBLPTR<15:8>) | | | | | | | | 0000 0000 | 34, 58 |
| TBLPTRL | Program Memory Table Pointer Low Byte (TBLPTR<7:0>) | | | | | | | | 0000 0000 | 34, 58 |
| TABLAT | Program Memory Table Latch | | | | | | | | 0000 0000 | 34, 58 |
| PRODH | Product Register High Byte | | | | | | | | xxxx xxxx | 34, 68 |
| PRODL | Product Register Low Byte | | | | | | | | xxxx xxxx | 34, 68 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 34, 72 |
| INTCON2 | $\overline{\text{RBPU}}$ | INTEDG0 | INTEDG1 | INTEDG2 | — | TMR0IP | — | RBIP | 1111 -1-1 | 34, 73 |
| INTCON3 | INT2IP | INT1IP | — | INT2IE | INT1IE | — | INT2IF | INT1IF | 11-0 0-00 | 34, 74 |
| INDF0 | Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register) | | | | | | | | N/A | 34, 51 |
| POSTINC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register) | | | | | | | | N/A | 34, 51 |
| POSTDEC0 | Uses contents of FSR0 to address data memory– value of FSR0 post-decremented (not a physical register) | | | | | | | | N/A | 34, 51 |
| PREINC0 | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) | | | | | | | | N/A | 34, 51 |
| PLUSW0 | Uses contents of FSR0 to address data memory – value of FSR0 offset by W (not a physical register) | | | | | | | | N/A | 34, 51 |
| FSR0H | — | — | — | — | Indirect Data Memory Address Pointer 0 High | | | | ---- 0000 | 34, 51 |
| FSR0L | Indirect Data Memory Address Pointer 0 Low Byte | | | | | | | | xxxx xxxx | 34, 51 |
| WREG | Working Register | | | | | | | | xxxx xxxx | 34 |
| INDF1 | Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register) | | | | | | | | **N/A** | 34, 51 |
| POSTINC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register) | | | | | | | | **N/A** | 34, 51 |
| POSTDEC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register) | | | | | | | | **N/A** | 34, 51 |
| PREINC1 | Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) | | | | | | | | **N/A** | 34, 51 |
| PLUSW1 | Uses contents of FSR1 to address data memory – value of FSR1 offset by W (not a physical register) | | | | | | | | **N/A** | 34, 51 |
| FSR1H | — | — | — | — | Indirect Data Memory Address Pointer 1 High | | | | ---- 0000 | 34, 51 |
| FSR1L | Indirect Data Memory Address Pointer 1 Low Byte | | | | | | | | xxxx xxxx | 34, 51 |
| BSR | — | — | — | — | Bank Select Register | | | | ---- 0000 | 35, 50 |
| INDF2 | Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register) | | | | | | | | **N/A** | 35, 51 |
| POSTINC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register) | | | | | | | | **N/A** | 35, 51 |
| POSTDEC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register) | | | | | | | | **N/A** | 35, 51 |
| PREINC2 | Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) | | | | | | | | **N/A** | 35, 51 |
| PLUSW2 | Uses contents of FSR2 to address data memory – value of FSR2 offset by W (not a physical register) | | | | | | | | **N/A** | 35, 51 |
| FSR2H | — | — | — | — | Indirect Data Memory Address Pointer 2 High | | | | ---- 0000 | 35, 51 |
| FSR2L | Indirect Data Memory Address Pointer 2 Low Byte | | | | | | | | xxxx xxxx | 35, 51 |
| STATUS | — | — | — | N | OV | Z | DC | C | ---x xxxx | 35, 53 |
| TMR0H | Timer0 Register High Byte | | | | | | | | 0000 0000 | 35, 97 |
| TMR0L | Timer0 Register Low Byte | | | | | | | | xxxx xxxx | 35, 97 |
| T0CON | TMR0ON | T08BIT | T0CS | T0SE | PSA | T0PS2 | T0PS1 | T0PS0 | 1111 1111 | 35, 95 |
| OSCCON | IDLEN | IRCF2 | IRCF1 | IRCF0 | OSTS | IOFS | SCS1 | SCS0 | 0000 q000 | 35, 16 |
| LVDCON | — | — | IVRST | LVDEN | LVDL3 | LVDL2 | LVDL1 | LVDL0 | --00 0101 | 35, 162 |
| WDTCON | — | — | — | — | — | — | — | SWDTEN | --- ---0 | 35, 174 |
| RCON | IPEN | — | — | $\overline{\text{RI}}$ | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | 0--1 11q0 | 33, 54, 81 |

**Legend:** x = unknown, u = unchanged, – = unimplemented, q = value depends on condition

**Note 1:** RA6 and associated bits are configured as port pins in RCIO, ECIO and INTIO2 (with port function on RA6) Oscillator mode only and read '0' in all other oscillator modes.

**2:** RA7 and associated bits are configured as port pins in INTIO2 Oscillator mode only and read '0' in all other modes.

**3:** Bit 21 of the PC is only available in Test mode and Serial Programming modes.

**4:** The RA5 port bit is only available when MCLRE fuse (CONFIG3H<7>) is programmed to '0'. Otherwise, RA5 reads '0'. This bit is read-only.


© 2002-2015 Microchip Technology Inc.

**TABLE 5-2: REGISTER FILE SUMMARY (PIC18F1220/1320) (CONTINUED)**

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Details on Page: |
|---|---|---|---|---|---|---|---|---|---|---|
| TMR1H | Timer1 Register High Byte | | | | | | | | xxxx xxxx | 35, 103 |
| TMR1L | Timer1 Register Low Byte | | | | | | | | xxxx xxxx | 35, 103 |
| T1CON | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON | 0000 0000 | 35, 98 |
| TMR2 | Timer2 Register | | | | | | | | 0000 0000 | 35, 104 |
| PR2 | Timer2 Period Register | | | | | | | | 1111 1111 | 35, 104 |
| T2CON | — | TOUTPS3 | TOUTPS2 | TOUTPS1 | TOUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | -000 0000 | 35, 104 |
| ADRESH | A/D Result Register High Byte | | | | | | | | xxxx xxxx | 35, 159 |
| ADRESL | A/D Result Register Low Byte | | | | | | | | xxxx xxxx | 35, 159 |
| ADCON0 | VCFG1 | VCFG0 | — | CHS2 | CHS1 | CHS0 | GO/$\overline{\text{DONE}}$ | ADON | 00-0 0000 | 35, 150 |
| ADCON1 | — | PCFG6 | PCFG5 | PCFG4 | PCFG3 | PCFG2 | PCFG1 | PCFG0 | -000 0000 | 35, 151 |
| ADCON2 | ADFM | — | ACQT2 | ACQT1 | ACQT0 | ADCS2 | ADCS1 | ADCS0 | 0-00 0000 | 35, 152 |
| CCPR1H | Capture/Compare/PWM Register 1 High Byte | | | | | | | | xxxx xxxx | 35. 110 |
| CCPR1L | Capture/Compare/PWM Register 1 Low Byte | | | | | | | | xxxx xxxx | 35, 110 |
| CCP1CON | P1M1 | P1M0 | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 | 0000 0000 | 35, 109 |
| PWM1CON | PRSEN | PDC6 | PDC5 | PDC4 | PDC3 | PDC2 | PDC1 | PDC0 | 0000 0000 | 35, 121 |
| ECCPAS | ECCPASE | ECCPAS2 | ECCPAS1 | ECCPAS0 | PSSAC1 | PSSAC0 | PSSBD1 | PSSBD0 | 0000 0000 | 35, 122 |
| TMR3H | Timer3 Register High Byte | | | | | | | | xxxx xxxx | 36, 108 |
| TMR3L | Timer3 Register Low Byte | | | | | | | | xxxx xxxx | 36, 108 |
| T3CON | RD16 | — | T3CKPS1 | T3CKPS0 | T3CCP1 | $\overline{\text{T3SYNC}}$ | TMR3CS | TMR3ON | 0-00 0000 | 36, 106 |
| SPBRGH | EUSART Baud Rate Generator High Byte | | | | | | | | 0000 0000 | 36 |
| SPBRG | EUSART Baud Rate Generator Low Byte | | | | | | | | 0000 0000 | 36, 130 |
| RCREG | EUSART Receive Register | | | | | | | | 0000 0000 | 36, 138, 137 |
| TXREG | EUSART Transmit Register | | | | | | | | 0000 0000 | 36, 135, 137 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 0000 0010 | 36, 127 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x | 36, 128 |
| BAUDCTL | — | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | -1-1 0-00 | 36, 129 |
| EEADR | EEPROM Address Register | | | | | | | | 0000 0000 | 36, 64 |
| EEDATA | EEPROM Data Register | | | | | | | | 0000 0000 | 36, 67 |
| EECON2 | EEPROM Control Register 2 (not a physical register) | | | | | | | | 0000 0000 | 36, 56, 64 |
| EECON1 | EEPGD | CFGS | — | FREE | WRERR | WREN | WR | RD | xx-0 x000 | 36, 57, 65 |
| IPR2 | OSCFIP | — | — | EEIP | — | LVDIP | TMR3IP | — | 1--1 -11- | 36, 80 |
| PIR2 | OSCFIF | — | — | EEIF | — | LVDIF | TMR3IF | — | 0--0 -00- | 36, 76 |
| PIE2 | OSCFIE | — | — | EEIE | — | LVDIE | TMR3IE | — | 0--0 -00- | 36, 78 |
| IPR1 | — | ADIP | RCIP | TXIP | — | CCP1IP | TMR2IP | TMR1IP | -111 -111 | 36, 79 |
| PIR1 | — | ADIF | RCIF | TXIF | — | CCP1IF | TMR2IF | TMR1IF | -000 -000 | 36, 75 |
| PIE1 | — | ADIE | RCIE | TXIE | — | CCP1IE | TMR2IE | TMR1IE | -000 -000 | 36, 77 |
| OSCTUNE | — | — | TUN5 | TUN4 | TUN3 | TUN2 | TUN1 | TUN0 | --00 0000 | 36, 14 |
| TRISB | Data Direction Control Register for PORTB | | | | | | | | 1111 1111 | 36, 94 |
| TRISA | TRISA7(2) | TRISA6(1) | — | Data Direction Control Register for PORTA | | | | | 11-1 1111 | 36, 85 |
| LATB | Read/Write PORTB Data Latch | | | | | | | | xxxx xxxx | 36, 94 |
| LATA | LATA<7>(2) | LATA<6>(1) | — | Read/Write PORTA Data Latch | | | | | xx-x xxxx | 36, 85 |
| PORTB | Read PORTB pins, Write PORTB Data Latch | | | | | | | | xxxx xxxx | 36, 94 |
| PORTA | RA7(2) | RA6(1) | RA5(4) | Read PORTA pins, Write PORTA Data Latch | | | | | xx0x 0000 | 36, 85 |

**Legend:** x = unknown, u = unchanged, – = unimplemented, q = value depends on condition

**Note 1:** RA6 and associated bits are configured as port pins in RCIO, ECIO and INTIO2 (with port function on RA6) Oscillator mode only and read '0' in all other oscillator modes.

**2:** RA7 and associated bits are configured as port pins in INTIO2 Oscillator mode only and read '0' in all other modes.

**3:** Bit 21 of the PC is only available in Test mode and Serial Programming modes.

**4:** The RA5 port bit is only available when MCLRE fuse (CONFIG3H<7>) is programmed to '0'. Otherwise, RA5 reads '0'. This bit is read-only.

## 5.10 Access Bank

The Access Bank is an architectural enhancement which is very useful for C compiler code optimization. The techniques used by the C compiler may also be useful for programs written in assembly.

This data memory region can be used for:

- Intermediate computational values
- Local variables of subroutines
- Faster context saving/switching of variables
- Common variables
- Faster evaluation/control of SFRs (no banking)

The Access Bank is comprised of the last 128 bytes in Bank 15 (SFRs) and the first 128 bytes in Bank 0. These two sections will be referred to as Access RAM High and Access RAM Low, respectively. Figure 5-6 indicates the Access RAM areas.

A bit in the instruction word specifies if the operation is to occur in the bank specified by the BSR register or in the Access Bank. This bit is denoted as the ‘a’ bit (for access bit).

When forced in the Access Bank (a = 0), the last address in Access RAM Low is followed by the first address in Access RAM High. Access RAM High maps the Special Function Registers, so these registers can be accessed without any software overhead. This is useful for testing status flags and modifying control bits.

## 5.11 Bank Select Register (BSR)

The need for a large general purpose memory space dictates a RAM banking scheme. The data memory is partitioned into as many as sixteen banks. When using direct addressing, the BSR should be configured for the desired bank.

BSR<3:0> holds the upper four bits of the 12-bit RAM address. The BSR<7:4> bits will always read ‘0’s and writes will have no effect (see Figure 5-7).

A MOVLB instruction has been provided in the instruction set to assist in selecting banks.

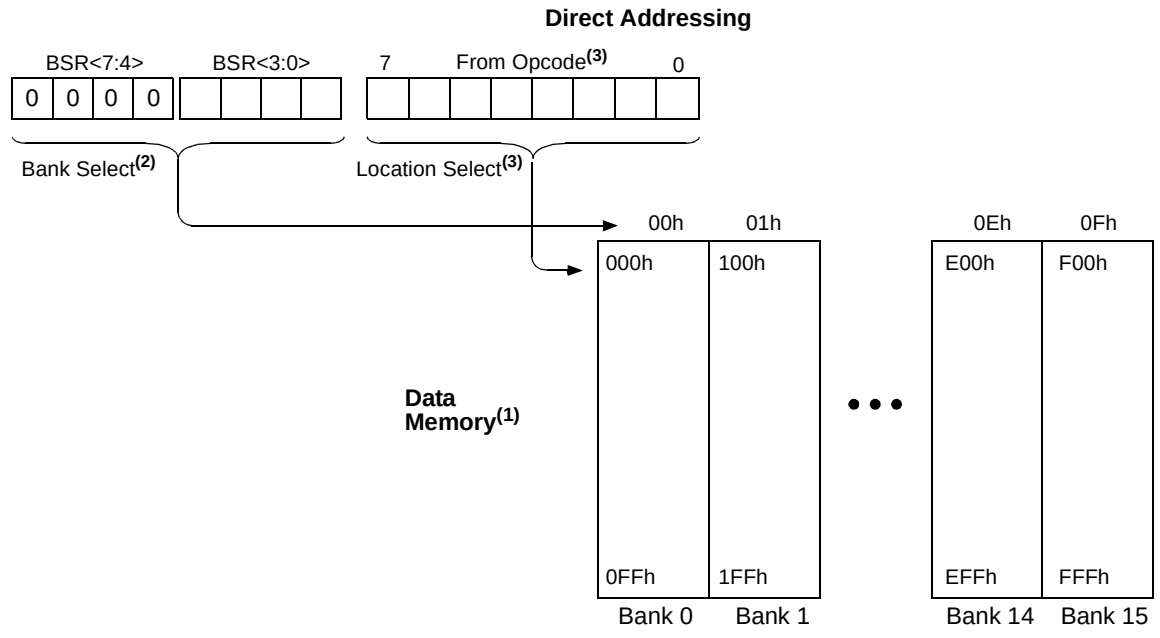
If the currently selected bank is not implemented, any read will return all ‘0’s and all writes are ignored. The Status register bits will be set/cleared as appropriate for the instruction performed.

Each Bank extends up to FFh (256 bytes). All data memory is implemented as static RAM.

A MOVFF instruction ignores the BSR, since the 12-bit addresses are embedded into the instruction word.

**Section 5.12 “Indirect Addressing, INDF and FSR Registers”** provides a description of indirect addressing, which allows linear addressing of the entire RAM space.

**FIGURE 5-7: DIRECT ADDRESSING**




**Note 1:** For register file map detail, see Table 5-1.
**2:** The access bit of the instruction can be used to force an override of the selected bank (BSR<3:0>) to the registers of the Access Bank.
**3:** The MOVFF instruction embeds the entire 12-bit address in the instruction.


© 2002-2015 Microchip Technology Inc.

## 5.12 Indirect Addressing, INDF and FSR Registers

Indirect addressing is a mode of addressing data memory, where the data memory address in the instruction is not fixed. An FSR register is used as a pointer to the data memory location that is to be read or written. Since this pointer is in RAM, the contents can be modified by the program. This can be useful for data tables in the data memory and for software stacks. Figure 5-8 shows how the fetched instruction is modified prior to being executed.

Indirect addressing is possible by using one of the INDF registers. Any instruction, using the INDF register, actually accesses the register pointed to by the File Select Register, FSR. Reading the INDF register itself, indirectly (FSR = 0), will read 00h. Writing to the INDF register indirectly, results in a no operation (NOP). The FSR register contains a 12-bit address, which is shown in Figure 5-9.

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

Example 5-5 shows a simple use of indirect addressing to clear the RAM in Bank 1 (locations 100h-1FFh) in a minimum number of instructions.

**EXAMPLE 5-5: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING**

```
        LFSR   FSR0,0x100  ;
NEXT    CLRF   POSTINC0    ; Clear INDF
                           ; register then
                           ; inc pointer
        BTFSS  FSR0H, 1    ; All done with
                           ; Bank1?
        GOTO   NEXT        ; NO, clear next
CONTINUE                   ; YES, continue
```

There are three indirect addressing registers. To address the entire data memory space (4096 bytes), these registers are 12-bit wide. To store the 12 bits of addressing information, two 8-bit registers are required:

1. FSR0: composed of FSR0H:FSR0L
2. FSR1: composed of FSR1H:FSR1L
3. FSR2: composed of FSR2H:FSR2L

In addition, there are registers INDF0, INDF1 and INDF2, which are not physically implemented. Reading or writing to these registers activates indirect addressing, with the value in the corresponding FSR register being the address of the data. If an instruction writes a value to INDF0, the value will be written to the address pointed to by FSR0H:FSR0L. A read from INDF1 reads the data from the address pointed to by FSR1H:FSR1L. INDFn can be used in code anywhere an operand can be used.

If INDF0, INDF1 or INDF2 are read indirectly via an FSR, all '0's are read (zero bit is set). Similarly, if INDF0, INDF1 or INDF2 are written to indirectly, the operation will be equivalent to a NOP instruction and the Status bits are not affected.

### 5.12.1 INDIRECT ADDRESSING OPERATION

Each FSR register has an INDF register associated with it, plus four additional register addresses. Performing an operation using one of these five registers determines how the FSR will be modified during indirect addressing.

When data access is performed using one of the five INDFn locations, the address selected will configure the FSRn register to:

- Do nothing to FSRn after an indirect access (no change) – INDFn
- Auto-decrement FSRn after an indirect access (post-decrement) – POSTDECn
- Auto-increment FSRn after an indirect access (post-increment) – POSTINCn
- Auto-increment FSRn before an indirect access (pre-increment) – PREINCn
- Use the value in the WREG register as an offset to FSRn. Do not modify the value of the WREG or the FSRn register after an indirect access (no change) – PLUSWn

When using the auto-increment or auto-decrement features, the effect on the FSR is not reflected in the Status register. For example, if the indirect address causes the FSR to equal '0', the Z bit will not be set.

Auto-incrementing or auto-decrementing an FSR affects all 12 bits. That is, when FSRnL overflows from an increment, FSRnH will be incremented automatically.

Adding these features allows the FSRn to be used as a stack pointer, in addition to its uses for table operations in data memory.

Each FSR has an address associated with it that performs an indexed indirect access. When a data access to this INDFn location (PLUSWn) occurs, the FSRn is configured to add the signed value in the WREG register and the value in FSR to form the address before an indirect access. The FSR value is not changed. The WREG offset range is -128 to +127.

If an FSR register contains a value that points to one of the INDFn, an indirect read will read 00h (zero bit is set), while an indirect write will be equivalent to a NOP (Status bits are not affected).

If an indirect addressing write is performed when the target address is an FSRnH or FSRnL register, the data is written to the FSR register, but no pre- or post-increment/decrement is performed.

© 2002-2015 Microchip Technology Inc.

**FIGURE 5-8: INDIRECT ADDRESSING OPERATION**

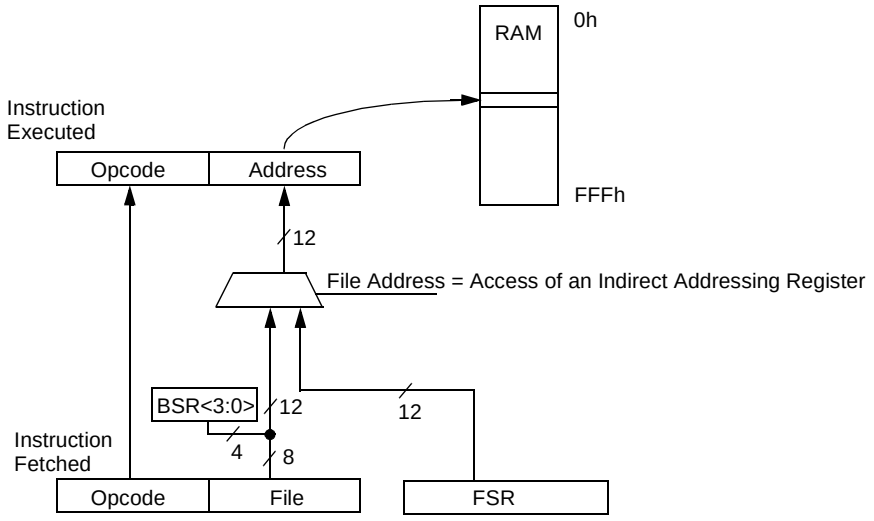
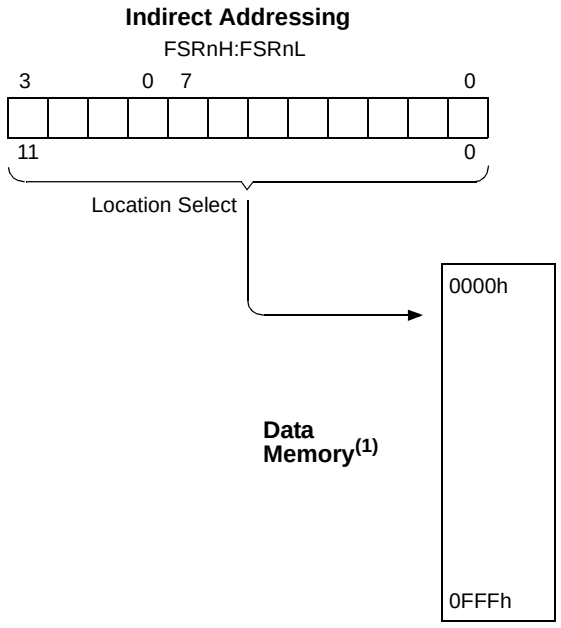



**FIGURE 5-9: INDIRECT ADDRESSING**




**Note 1:** For register file map detail, see Table 5-1.


© 2002-2015 Microchip Technology Inc.

## 5.13 Status Register

The STATUS register, shown in Register 5-2, contains the arithmetic status of the ALU. As with any other SFR, it can be the operand for any instruction.

If the STATUS register is the destination for an instruction that affects the Z, DC, C, OV or N bits, the results of the instruction are not written; instead, the status is updated according to the instruction performed. Therefore, the result of an instruction with the STATUS register as its destination may be different than intended. As an example, CLRF STATUS will set the Z bit and leave the remaining Status bits unchanged ('000u u1uu').

It is recommended that only BCF, BSF, SWAPF, MOVFF and MOVWF instructions are used to alter the STATUS register, because these instructions do not affect the Z, C, DC, OV or N bits in the STATUS register.

For other instructions that do not affect Status bits, see the instruction set summaries in Table 20-1.

**Note:** The C and DC bits operate as the borrow and digit borrow bits, respectively, in subtraction.

**REGISTER 5-2: STATUS: STATUS REGISTER**

| U-0 | U-0 | U-0 | R-1/q | R-1/q | R/W-0/u | R/W-0/u | R/W-0/u |
|---|---|---|---|---|---|---|---|
| — | — | — | N | OV | Z | DC | C |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Value depends on condition |

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **N:** Negative bit
This bit is used for signed arithmetic (2's complement). It indicates whether the result was negative (ALU MSB = 1).
1 = Result was negative
0 = Result was positive

bit 3 **OV:** Overflow bit
This bit is used for signed arithmetic (2's complement). It indicates an overflow of the 7-bit magnitude, which causes the sign bit (bit 7) to change state.
1 = Overflow occurred for signed arithmetic (in this arithmetic operation)
0 = No overflow occurred

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/Digit Borrow bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)(1)
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(1) (ADDWF, ADDLW, SUBLW, SUBWF instructions)(1)
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 1:** For Borrow, the polarity is reversed. 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-order or low-order bit of the source register.

## 5.14 RCON Register

The Reset Control (RCON) register contains flag bits that allow differentiation between the sources of a device Reset. These flags include the $\overline{TO}$, $\overline{PD}$, $\overline{POR}$, $\overline{BOR}$ and $\overline{RI}$ bits. This register is readable and writable.

**Note 1:** If the BOR Configuration bit is set (Brown-out Reset enabled), the $\overline{BOR}$ bit is '1' on a Power-on Reset. After a Brown-out Reset has occurred, the BOR bit will be cleared and must be set by firmware to indicate the occurrence of the next Brown-out Reset.

**2:** It is recommended that the $\overline{POR}$ bit be set after a Power-on Reset has been detected, so that subsequent Power-on Resets may be detected.

**REGISTER 5-3: RCON: RESET CONTROL REGISTER**

| R/W-0 | U-0 | U-0 | R/W-1 | R-1 | R-1 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| IPEN | — | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Value depends on condition |

bit 7 **IPEN:** Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)

bit 6-5 **Unimplemented:** Read as '0'

bit 4 **$\overline{RI}$:** RESET Instruction Flag bit
1 = The RESET instruction was not executed (set by firmware only)
0 = The RESET instruction was executed causing a device Reset (must be set in software after a Brown-out Reset occurs)

bit 3 **$\overline{TO}$:** Watchdog Time-out Flag bit
1 = Set by power-up, CLRWDT instruction or SLEEP instruction
0 = A WDT time-out occurred

bit 2 **$\overline{PD}$:** Power-down Detection Flag bit
1 = Set by power-up or by the CLRWDT instruction
0 = Cleared by execution of the SLEEP instruction

bit 1 **$\overline{POR}$:** Power-on Reset Status bit
1 = A Power-on Reset has not occurred (set by firmware only)
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

bit 0 **$\overline{BOR}$:** Brown-out Reset Status bit
1 = A Brown-out Reset has not occurred (set by firmware only)
0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset occurs)

**Note 1:** For $\overline{Borrow}$, the polarity is reversed. 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-order or low-order bit of the source register.


© 2002-2015 Microchip Technology Inc.

## 6.0 FLASH PROGRAM MEMORY

The Flash program memory is readable, writable and erasable during normal operation over the entire VDD range.

A read from program memory is executed on one byte at a time. A write to program memory is executed on blocks of 8 bytes at a time. Program memory is erased in blocks of 64 bytes at a time. A “Bulk Erase” operation may not be issued from user code.

While writing or erasing program memory, instruction fetches cease until the operation is complete. The program memory cannot be accessed during the write or erase, therefore, code cannot execute. An internal programming timer terminates program memory writes and erases.

A value written to program memory does not need to be a valid instruction. Executing a program memory location that forms an invalid instruction results in a NOP.

### 6.1 Table Reads and Table Writes

In order to read and write program memory, there are two operations that allow the processor to move bytes between the program memory space and the data RAM:

- Table Read (TBLRD)
- Table Write (TBLWT)

The program memory space is 16 bits wide, while the data RAM space is eight bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

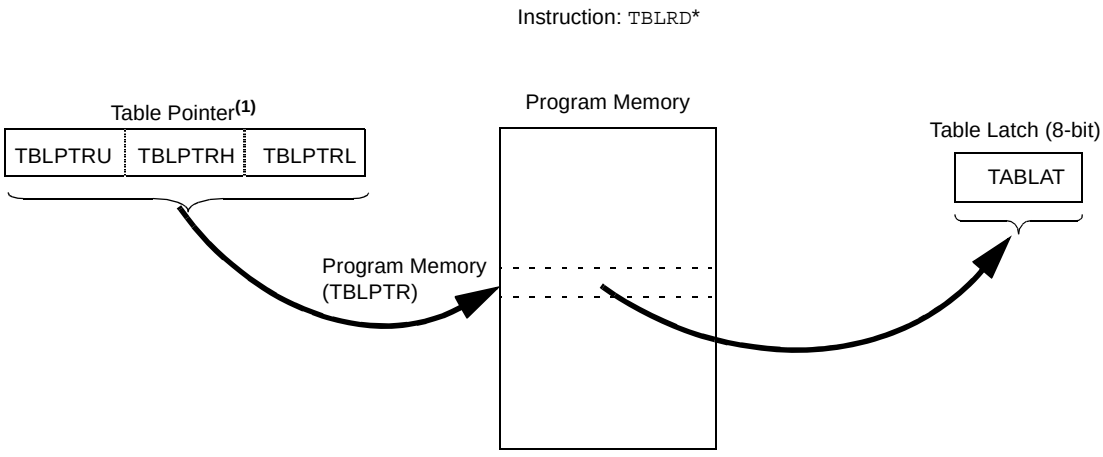
Table read operations retrieve data from program memory and place it into TABLAT in the data RAM space. Figure 6-1 shows the operation of a table read with program memory and data RAM.

Table write operations store data from TABLAT in the data memory space into holding registers in program memory. The procedure to write the contents of the holding registers into program memory is detailed in **Section 6.5 “Writing to Flash Program Memory”**. Figure 6-2 shows the operation of a table write with program memory and data RAM.

Table operations work with byte entities. A table block containing data, rather than program instructions, is not required to be word aligned. Therefore, a table block can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions will need to be word aligned (TBLPTRL<0> = 0).

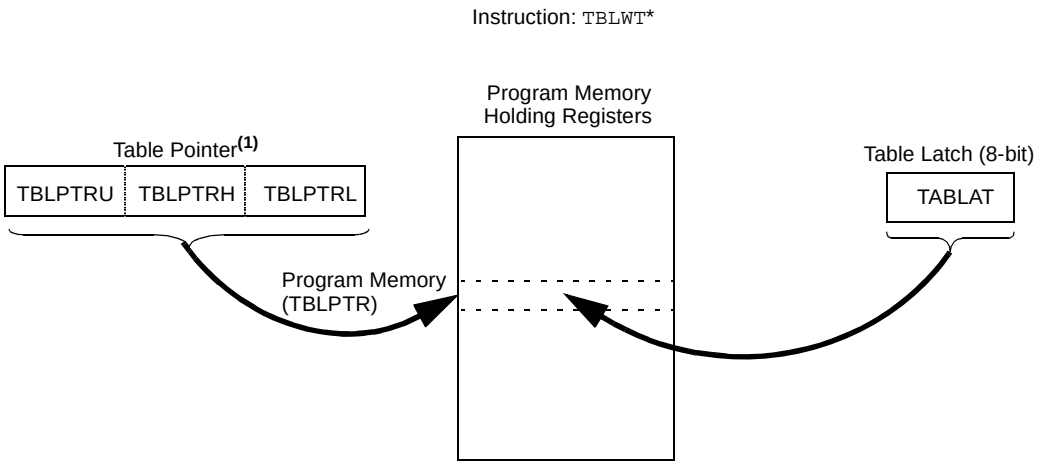
The EEPROM on-chip timer controls the write and erase times. The write and erase voltages are generated by an on-chip charge pump rated to operate over the voltage range of the device for byte or word operations.

**FIGURE 6-1: TABLE READ OPERATION**




**Note 1:** Table Pointer points to a byte in program memory.

**FIGURE 6-2: TABLE WRITE OPERATION**




**Note 1:** Table Pointer actually points to one of eight holding registers, the address of which is determined by TBLPTRL<2:0>. The process for physically writing data to the program memory array is discussed in **Section 6.5 "Writing to Flash Program Memory"**.

## 6.2 Control Registers

Several control registers are used in conjunction with the TBLRD and TBLWT instructions. These include the:

- EECON1 register
- EECON2 register
- TABLAT register
- TBLPTR registers

### 6.2.1 EECON1 AND EECON2 REGISTERS

EECON1 is the control register for memory accesses.

EECON2 is not a physical register. Reading EECON2 will read all '0's. The EECON2 register is used exclusively in the memory write and erase sequences.

Control bit, EEPGD, determines if the access will be to program or data EEPROM memory. When clear, operations will access the data EEPROM memory. When set, program memory is accessed.

Control bit, CFGS, determines if the access will be to the Configuration registers, or to program memory/data EEPROM memory. When set, subsequent operations access Configuration registers. When CFGS is clear, the EEPGD bit selects either program Flash or data EEPROM memory.

The FREE bit controls program memory erase operations. When the FREE bit is set, the erase operation is initiated on the next WR command. When FREE is clear, only writes are enabled.

The WREN bit enables and disables erase and write operations. When set, erase and write operations are allowed. When clear, erase and write operations are disabled – the WR bit cannot be set while the WREN bit is clear. This process helps to prevent accidental writes to memory due to errant (unexpected) code execution.

Firmware should keep the WREN bit clear at all times, except when starting erase or write operations. Once firmware has set the WR bit, the WREN bit may be cleared. Clearing the WREN bit will not affect the operation in progress.

The WRERR bit is set when a write operation is interrupted by a Reset. In these situations, the user can check the WRERR bit and rewrite the location. It will be necessary to reload the data and address registers (EEDATA and EEADR) as these registers have cleared as a result of the Reset.

Control bits, RD and WR, start read and erase/write operations, respectively. These bits are set by firmware and cleared by hardware at the completion of the operation.

The RD bit cannot be set when accessing program memory (EEPGD = 1). Program memory is read using table read instructions. See **Section 6.3 "Reading the Flash Program Memory"** regarding table reads.

| **Note:** | Interrupt flag bit, EEIF in the PIR2 register, is set when the write is complete. It must be cleared in software. |
|---|---|


© 2002-2015 Microchip Technology Inc.

**REGISTER 6-1: EECON1: EEPROM CONTROL 1 REGISTER**

| R/W-x | R/W-x | U-0 | R/W-0 | R/W-x | R/W-0 | R/S-0 | R/S-0 |
|---|---|---|---|---|---|---|---|
| EEPGD | CFGS | — | FREE | WRERR(1) | WREN | WR | RD |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| S = Bit can only be set | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | HC = Bit is cleared by hardware |

bit 7 **EEPGD:** Flash Program or Data EEPROM Memory Select bit
- 1 = Access program Flash memory
- 0 = Access data EEPROM memory

bit 6 **CFGS:** Flash Program/Data EEPROM or Configuration Select bit
- 1 = Accesses Configuration, User ID and Device ID Registers
- 0 = Accesses Flash Program or data EEPROM Memory

bit 5 **Unimplemented:** Read as ‘0’

bit 4 **FREE:** Flash Row Erase Enable bit
- 1 = Erase the program memory row addressed by TBLPTR on the next WR command (cleared by completion of erase operation – TBLPTR<5:0> are ignored)
- 0 = Perform write only

bit 3 **WRERR:** EEPROM Error Flag bit(1)
- 1 = A write operation was prematurely terminated (any Reset during self-timed programming)
- 0 = The write operation completed normally

bit 2 **WREN:** Program/Erase Enable bit
- 1 = Allows program/erase cycles
- 0 = Inhibits programming/erasing of program Flash and data EEPROM

bit 1 **WR:** Write Control bit
- 1 = Initiates a data EEPROM erase/write cycle or a program memory erase cycle or write cycle. (The operation is self-timed and the bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.)
- 0 = Write cycle completed

bit 0 **RD:** Read Control bit
- 1 = Initiates a memory read (Read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software. RD bit cannot be set when EEPGD = 1.)
- 0 = Read completed

**Note 1:** When a WRERR occurs, the EEPGD and CFGS bits are not cleared. This allows tracing of the error condition.

### 6.2.2 TABLAT – TABLE LATCH REGISTER

The Table Latch (TABLAT) is an 8-bit register mapped into the SFR space. The table latch is used to hold 8-bit data during data transfers between program memory and data RAM.

### 6.2.3 TBLPTR – TABLE POINTER REGISTER

The Table Pointer (TBLPTR) addresses a byte within the program memory. The TBLPTR is comprised of three SFR registers: Table Pointer Upper Byte, Table Pointer High Byte and Table Pointer Low Byte (TBLPTRU:TBLPTRH:TBLPTRL). These three registers join to form a 22-bit wide pointer. The low-order 21 bits allow the device to address up to 2 Mbytes of program memory space. Setting the 22nd bit allows access to the device ID, the user ID and the configuration bits.

The Table Pointer (TBLPTR) register is used by the `TBLRD` and `TBLWT` instructions. These instructions can update the TBLPTR in one of four ways based on the table operation. These operations are shown in Table 6-1. These operations on the TBLPTR only affect the low-order 21 bits.

### 6.2.4 TABLE POINTER BOUNDARIES

TBLPTR is used in reads, writes and erases of the Flash program memory.

When a `TBLRD` is executed, all 22 bits of the Table Pointer determine which byte is read from program or configuration memory into TABLAT.

When a `TBLWT` is executed, the three LSbs of the Table Pointer (TBLPTR<2:0>) determine which of the eight program memory holding registers is written to. When the timed write to program memory (long write) begins, the 19 MSbs of the Table Pointer (TBLPTR<21:3>) will determine which program memory block of 8 bytes is written to (TBLPTR<2:0> are ignored). For more detail, see **Section 6.5 "Writing to Flash Program Memory"**.

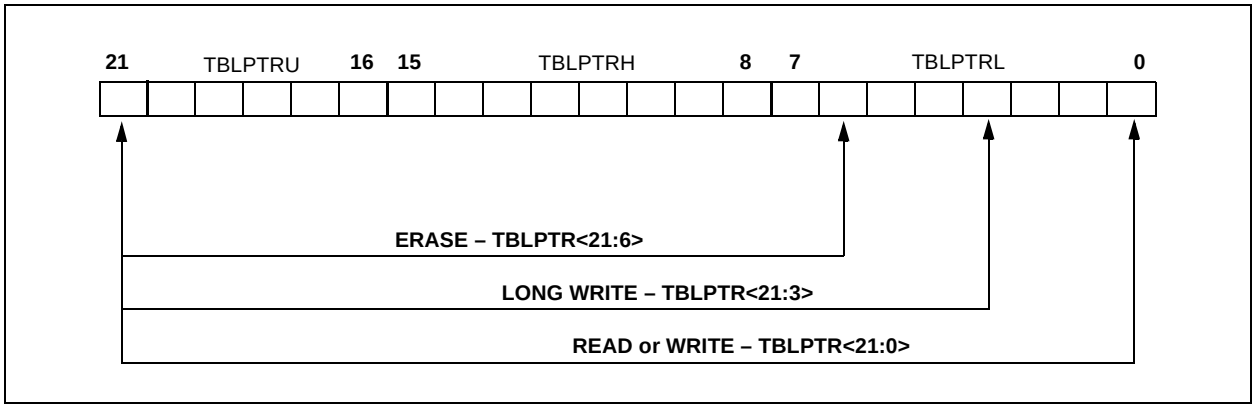
When an erase of program memory is executed, the 16 MSbs of the Table Pointer (TBLPTR<21:6>) point to the 64-byte block that will be erased. The Least Significant bits (TBLPTR<5:0>) are ignored.

Figure 6-3 describes the relevant boundaries of TBLPTR based on Flash program memory operations.

**TABLE 6-1: TABLE POINTER OPERATIONS WITH TBLRD AND TBLWT INSTRUCTIONS**

| Example | Operation on Table Pointer |
|---|---|
| `TBLRD*`<br>`TBLWT*` | TBLPTR is not modified |
| `TBLRD*+`<br>`TBLWT*+` | TBLPTR is incremented after the read/write |
| `TBLRD*-`<br>`TBLWT*-` | TBLPTR is decremented after the read/write |
| `TBLRD+*`<br>`TBLWT+*` | TBLPTR is incremented before the read/write |

**FIGURE 6-3: TABLE POINTER BOUNDARIES BASED ON OPERATION**





© 2002-2015 Microchip Technology Inc.

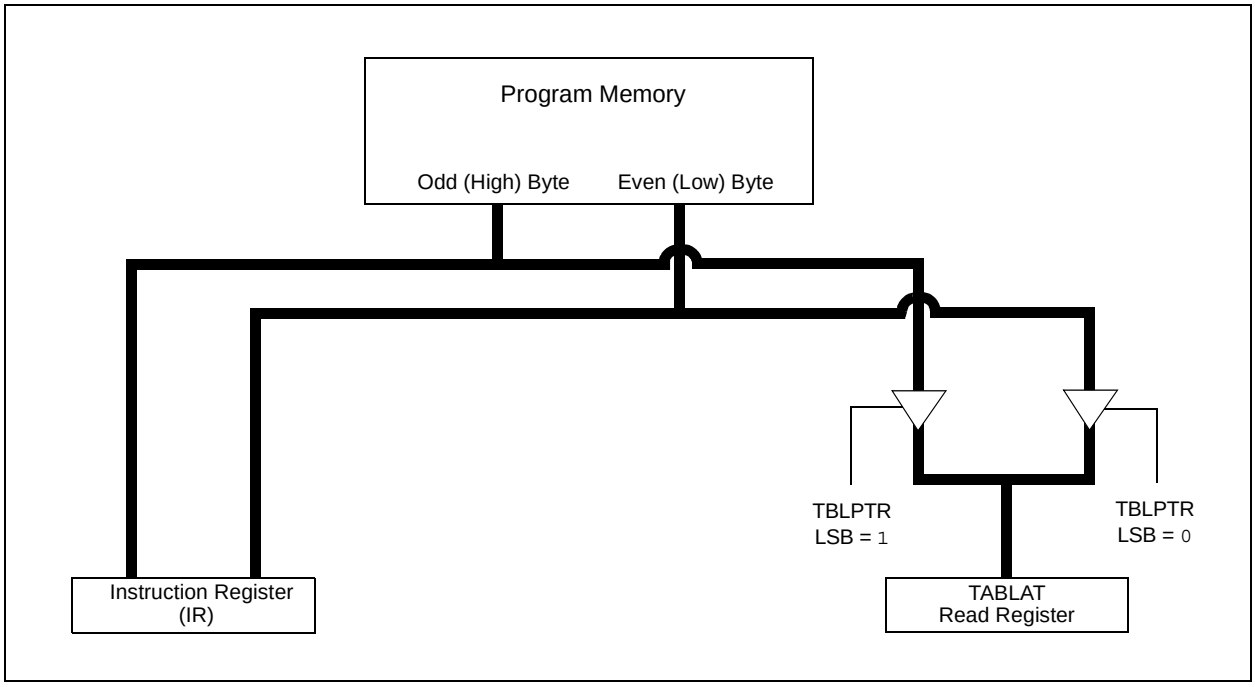
## 6.3 Reading the Flash Program Memory

The TBLRD instruction is used to retrieve data from program memory and place it into data RAM. Table reads from program memory are performed one byte at a time.

TBLPTR points to a byte address in program space. Executing a TBLRD instruction places the byte pointed to into TABLAT. In addition, TBLPTR can be modified automatically for the next table read operation.

The internal program memory is typically organized by words. The Least Significant bit of the address selects between the high and low bytes of the word. Figure 6-4 shows the interface between the internal program memory and the TABLAT.

**FIGURE 6-4: READS FROM FLASH PROGRAM MEMORY**




**EXAMPLE 6-1: READING A FLASH PROGRAM MEMORY WORD**

```
            MOVLW   CODE_ADDR_UPPER       ; Load TBLPTR with the base
            MOVWF   TBLPTRU               ; address of the word
            MOVLW   CODE_ADDR_HIGH
            MOVWF   TBLPTRH
            MOVLW   CODE_ADDR_LOW
            MOVWF   TBLPTRL
READ_WORD
            TBLRD*+                       ; read into TABLAT and increment TBLPTR
            MOVFW   TABLAT                ; get data
            MOVWF   WORD_EVEN
            TBLRD*+                       ; read into TABLAT and increment TBLPTR
            MOVFW   TABLAT                ; get data
            MOVWF   WORD_ODD
```

## 6.4 Erasing Flash Program Memory

The minimum erase block size is 32 words or 64 bytes under firmware control. Only through the use of an external programmer, or through ICSP control, can larger blocks of program memory be bulk erased. Word erase in Flash memory is not supported.

When initiating an erase sequence from the micro-controller itself, a block of 64 bytes of program memory is erased. The Most Significant 16 bits of the TBLPTR<21:6> point to the block being erased. TBLPTR<5:0> are ignored.

The EECON1 register commands the erase operation. The EEPGD bit must be set to point to the Flash program memory. The CFGS bit must be clear to access program Flash and data EEPROM memory. The WREN bit must be set to enable write operations. The FREE bit is set to select an erase operation. The WR bit is set as part of the required instruction sequence (as shown in Example 6-2) and starts the actual erase operation. It is not necessary to load the TABLAT register with any data as it is ignored.

For protection, the write initiate sequence using EECON2 must be used.

A long write is necessary for erasing the internal Flash. Instruction execution is halted while in a long write cycle. The long write will be terminated by the internal programming timer.

### 6.4.1 FLASH PROGRAM MEMORY ERASE SEQUENCE

The sequence of events for erasing a block of internal program memory location is:

1. Load Table Pointer with address of row being erased.
2. Set the EECON1 register for the erase operation:
   - •set EEPGD bit to point to program memory;
   - •clear the CFGS bit to access program memory;
   - •set WREN bit to enable writes;
   - •set FREE bit to enable the erase.
3. Disable interrupts.
4. Write 55h to EECON2.
5. Write AAh to EECON2.
6. Set the WR bit. This will begin the row erase cycle.
7. The CPU will stall for duration of the erase (about 2 ms using internal timer).
8. Execute a NOP.
9. Re-enable interrupts.

**EXAMPLE 6-2: ERASING A FLASH PROGRAM MEMORY ROW**

```
                    MOVLW   CODE_ADDR_UPPER       ; load TBLPTR with the base
                    MOVWF   TBLPTRU               ; address of the memory block
                    MOVLW   CODE_ADDR_HIGH
                    MOVWF   TBLPTRH
                    MOVLW   CODE_ADDR_LOW
                    MOVWF   TBLPTRL
        ERASE_ROW
                    BSF     EECON1, EEPGD         ; point to FLASH program memory
                    BSF     EECON1, WREN          ; enable write to memory
                    BSF     EECON1, FREE          ; enable Row Erase operation
                    BCF     INTCON, GIE           ; disable interrupts
                    MOVLW   55h
                    MOVWF   EECON2                ; write 55H
    Required        MOVLW   AAh
    Sequence        MOVWF   EECON2                ; write AAH
                    BSF     EECON1, WR            ; start erase (CPU stall)
                    NOP
                    BSF     INTCON, GIE           ; re-enable interrupts
```


© 2002-2015 Microchip Technology Inc.

## 6.5 Writing to Flash Program Memory

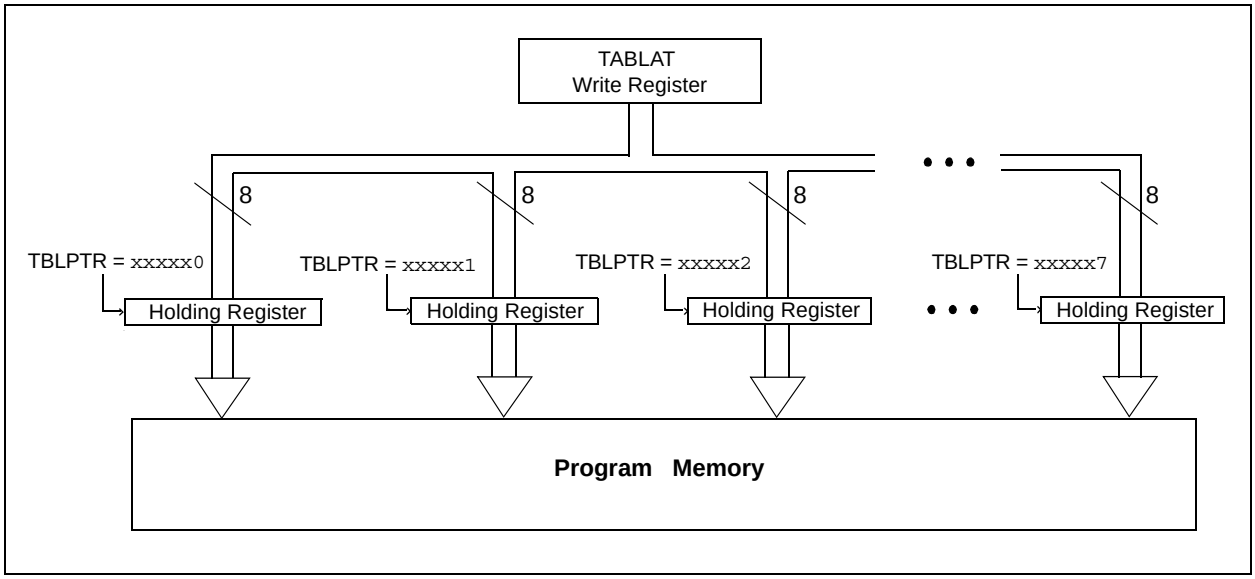
The programming block size is 4 words or 8 bytes. Word or byte programming is not supported.

Table writes are used internally to load the holding registers needed to program the Flash memory. There are eight holding registers used by the table writes for programming.

Since the Table Latch (TABLAT) is only a single byte, the TBLWT instruction must be executed eight times for each programming operation. All of the table write operations will essentially be short writes, because only the holding registers are written. At the end of updating eight registers, the EECON1 register must be written to, to start the programming operation with a long write.

The long write is necessary for programming the internal Flash. Instruction execution is halted while in a long write cycle. The long write will be terminated by the internal programming timer.

**FIGURE 6-5: TABLE WRITES TO FLASH PROGRAM MEMORY**




### 6.5.1 FLASH PROGRAM MEMORY WRITE SEQUENCE

The sequence of events for programming an internal program memory location should be:

1. Read 64 bytes into RAM.
2. Update data values in RAM as necessary.
3. Load Table Pointer with address being erased.
4. Do the row erase procedure (see **Section 6.4.1 "Flash Program Memory Erase Sequence"**).
5. Load Table Pointer with address of first byte being written.
6. Write the first 8 bytes into the holding registers with auto-increment.
7. Set the EECON1 register for the write operation:
   - •set EEPGD bit to point to program memory;
   - •clear the CFGS bit to access program memory;
   - •set WREN bit to enable byte writes.
8. Disable interrupts.
9. Write 55h to EECON2.
10. Write AAh to EECON2.
11. Set the WR bit. This will begin the write cycle.
12. The CPU will stall for duration of the write (about 2 ms using internal timer).
13. Execute a NOP.
14. Re-enable interrupts.
15. Repeat steps 6-14 seven times to write 64 bytes.
16. Verify the memory (table read).

This procedure will require about 18 ms to update one row of 64 bytes of memory. An example of the required code is given in Example 6-3.

**EXAMPLE 6-3: WRITING TO FLASH PROGRAM MEMORY**

```
            MOVLW   D'64                    ; number of bytes in erase block
            MOVWF   COUNTER
            MOVLW   BUFFER_ADDR_HIGH        ; point to buffer
            MOVWF   FSR0H
            MOVLW   BUFFER_ADDR_LOW
            MOVWF   FSR0L
            MOVLW   CODE_ADDR_UPPER         ; Load TBLPTR with the base
            MOVWF   TBLPTRU                 ; address of the memory block
            MOVLW   CODE_ADDR_HIGH
            MOVWF   TBLPTRH
            MOVLW   CODE_ADDR_LOW           ; 6 LSB = 0
            MOVWF   TBLPTRL
READ_BLOCK
            TBLRD*+                         ; read into TABLAT, and inc
            MOVF    TABLAT, W               ; get data
            MOVWF   POSTINC0                ; store data and increment FSR0
            DECFSZ  COUNTER                 ; done?
            GOTO    READ_BLOCK              ; repeat
MODIFY_WORD
            MOVLW   DATA_ADDR_HIGH          ; point to buffer
            MOVWF   FSR0H
            MOVLW   DATA_ADDR_LOW
            MOVWF   FSR0L
            MOVLW   NEW_DATA_LOW            ; update buffer word and increment FSR0
            MOVWF   POSTINC0
            MOVLW   NEW_DATA_HIGH           ; update buffer word
            MOVWF   INDF0
ERASE_BLOCK
            MOVLW   CODE_ADDR_UPPER         ; load TBLPTR with the base
            MOVWF   TBLPTRU                 ; address of the memory block
            MOVLW   CODE_ADDR_HIGH
            MOVWF   TBLPTRH
            MOVLW   CODE_ADDR_LOW           ; 6 LSB = 0
            MOVWF   TBLPTRL
            BCF     EECON1, CFGS            ; point to PROG/EEPROM memory
            BSF     EECON1, EEPGD           ; point to FLASH program memory
            BSF     EECON1, WREN            ; enable write to memory
            BSF     EECON1, FREE            ; enable Row Erase operation
            BCF     INTCON, GIE             ; disable interrupts
            MOVLW   55h                     ; Required sequence
            MOVWF   EECON2                  ; write 55H
            MOVLW   AAh
            MOVWF   EECON2                  ; write AAH
            BSF     EECON1, WR              ; start erase (CPU stall)
            NOP
            BSF     INTCON, GIE             ; re-enable interrupts
WRITE_BUFFER_BACK
            MOVLW   8                       ; number of write buffer groups of 8 bytes
            MOVWF   COUNTER_HI
            MOVLW   BUFFER_ADDR_HIGH        ; point to buffer
            MOVWF   FSR0H
            MOVLW   BUFFER_ADDR_LOW
            MOVWF   FSR0L
PROGRAM_LOOP
            MOVLW   8                       ; number of bytes in holding register
            MOVWF   COUNTER
```


© 2002-2015 Microchip Technology Inc.

**EXAMPLE 6-3: WRITING TO FLASH PROGRAM MEMORY (CONTINUED)**

```
WRITE_WORD_TO_HREGS
            MOVF    POSTINC0, W             ; get low byte of buffer data and increment FSR0
            MOVWF   TABLAT                  ; present data to table latch
            TBLWT+*                         ; short write
                                            ; to internal TBLWT holding register, increment
                                              TBLPTR
            DECFSZ  COUNTER                 ; loop until buffers are full
            GOTO    WRITE_WORD_TO_HREGS
PROGRAM_MEMORY
            BCF     INTCON, GIE             ; disable interrupts
            MOVLW   55h                     ; required sequence
            MOVWF   EECON2                  ; write 55H
            MOVLW   AAh
            MOVWF   EECON2                  ; write AAH
            BSF     EECON1, WR              ; start program (CPU stall)
            NOP
            BSF     INTCON, GIE             ; re-enable interrupts
            DECFSZ  COUNTER_HI              ; loop until done
            GOTO PROGRAM_LOOP
            BCF     EECON1, WREN            ; disable write to memory
```

### 6.5.2 WRITE VERIFY

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit.

### 6.5.3 UNEXPECTED TERMINATION OF WRITE OPERATION

If a write is terminated by an unplanned event, such as loss of power or an unexpected Reset, the memory location just programmed should be verified and reprogrammed if needed. The WRERR bit is set when a write operation is interrupted by a MCLR Reset, or a WDT Time-out Reset during normal operation. In these situations, users can check the WRERR bit and rewrite the location.

## 6.6 Flash Program Operation During Code Protection

See **Section 19.0 “Special Features of the CPU”** for details on code protection of Flash program memory.

**TABLE 6-2: REGISTERS ASSOCIATED WITH PROGRAM FLASH MEMORY**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on: POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| TBLPTRU | — | — | bit 21 | Program Memory Table Pointer Upper Byte (TBLPTR<20:16>) | | | | | --00 0000 | --00 0000 |
| TBPLTRH | Program Memory Table Pointer High Byte (TBLPTR<15:8>) | | | | | | | | 0000 0000 | 0000 0000 |
| TBLPTRL | Program Memory Table Pointer High Byte (TBLPTR<7:0>) | | | | | | | | 0000 0000 | 0000 0000 |
| TABLAT | Program Memory Table Latch | | | | | | | | 0000 0000 | 0000 0000 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INTE | RBIE | TMR0IF | INTF | RBIF | 0000 000x | 0000 000u |
| EECON2 | EEPROM Control Register 2 (not a physical register) | | | | | | | | — | — |
| EECON1 | EEPGD | CFGS | — | FREE | WRERR | WREN | WR | RD | xx-0 x000 | uu-0 u000 |
| IPR2 | OSCFIP | — | — | EEIP | — | LVDIP | TMR3IP | — | 1--1 -11- | 1--1 -11- |
| PIR2 | OSCFIF | — | — | EEIF | — | LVDIF | TMR3IF | — | 0--0 -00- | 0--0 -00- |
| PIE2 | OSCFIE | — | — | EEIE | — | LVDIE | TMR3IE | — | 0--0 -00- | 0--0 -00- |

**Legend:** x = unknown, u = unchanged, – = unimplemented, read as ‘0’.
Shaded cells are not used during Flash/EEPROM access.

## 7.0 DATA EEPROM MEMORY

The data EEPROM is readable and writable during normal operation over the entire VDD range. The data memory is not directly mapped in the register file space. Instead, it is indirectly addressed through the Special Function Registers (SFR).

There are four SFRs used to read and write the program and data EEPROM memory. These registers are:

- EECON1
- EECON2
- EEDATA
- EEADR

The EEPROM data memory allows byte read and write. When interfacing to the data memory block, EEDATA holds the 8-bit data for read/write and EEADR holds the address of the EEPROM location being accessed. These devices have 256 bytes of data EEPROM with an address range from 00h to FFh.

The EEPROM data memory is rated for high erase/write cycle endurance. A byte write automatically erases the location and writes the new data (erase-before-write). The write time is controlled by an on-chip timer. The write time will vary with voltage and temperature, as well as from chip to chip. Please refer to parameter D122 (Table 22-1 in **Section 22.0 "Electrical Characteristics"**) for exact limits.

### 7.1 EEADR

The address register can address 256 bytes of data EEPROM.

### 7.2 EECON1 and EECON2 Registers

EECON1 is the control register for memory accesses.

EECON2 is not a physical register. Reading EECON2 will read all '0's. The EECON2 register is used exclusively in the memory write and erase sequences.

Control bit, EEPGD, determines if the access will be to program or data EEPROM memory. When clear, operations will access the data EEPROM memory. When set, program memory is accessed.

Control bit, CFGS, determines if the access will be to the Configuration registers or to program memory/data EEPROM memory. When set, subsequent operations access Configuration registers. When CFGS is clear, the EEPGD bit selects either program Flash or data EEPROM memory.

The WREN bit enables and disables erase and write operations. When set, erase and write operations are allowed. When clear, erase and write operations are disabled – the WR bit cannot be set while the WREN bit is clear. This mechanism helps to prevent accidental writes to memory due to errant (unexpected) code execution.

Firmware should keep the WREN bit clear at all times, except when starting erase or write operations. Once firmware has set the WR bit, the WREN bit may be cleared. Clearing the WREN bit will not affect the operation in progress.

The WRERR bit is set when a write operation is interrupted by a Reset. In these situations, the user can check the WRERR bit and rewrite the location. It is necessary to reload the data and address registers (EEDATA and EEADR), as these registers have cleared as a result of the Reset.

Control bits, RD and WR, start read and erase/write operations, respectively. These bits are set by firmware and cleared by hardware at the completion of the operation.

The RD bit cannot be set when accessing program memory (EEPGD = 1). Program memory is read using table read instructions. See **Section 6.1 "Table Reads and Table Writes"** regarding table reads.

| **Note:** | Interrupt flag bit, EEIF in the PIR2 register, is set when write is complete. It must be cleared in software. |
|---|---|


© 2002-2015 Microchip Technology Inc.

**REGISTER 7-1: EECON1: EEPROM CONTROL 1 REGISTER**

| R/W-x | R/W-x | U-0 | R/W-0 | R/W-x | R/W-0 | R/S-0 | R/S-0 |
|---|---|---|---|---|---|---|---|
| EEPGD | CFGS | — | FREE | WRERR(1) | WREN | WR | RD |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| S = Bit can only be set | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | HC = Bit is cleared by hardware |

bit 7 **EEPGD:** Flash Program or Data EEPROM Memory Select bit
- 1 = Access program Flash memory
- 0 = Access data EEPROM memory

bit 6 **CFGS:** Flash Program/Data EEPROM or Configuration Select bit
- 1 = Accesses Configuration, User ID and Device ID Registers
- 0 = Accesses Flash Program or data EEPROM Memory

bit 5 **Unimplemented:** Read as ‘0’

bit 4 **FREE:** Flash Row Erase Enable bit
- 1 = Erase the program memory row addressed by TBLPTR on the next WR command (cleared by completion of erase operation – TBLPTR<5:0> are ignored)
- 0 = Perform write only

bit 3 **WRERR:** EEPROM Error Flag bit(1)
- 1 = A write operation was prematurely terminated (any Reset during self-timed programming)
- 0 = The write operation completed normally

bit 2 **WREN:** Program/Erase Enable bit
- 1 = Allows program/erase cycles
- 0 = Inhibits programming/erasing of program Flash and data EEPROM

bit 1 **WR:** Write Control bit
- 1 = Initiates a data EEPROM erase/write cycle or a program memory erase cycle or write cycle. (The operation is self-timed and the bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.)
- 0 = Write cycle completed

bit 0 **RD:** Read Control bit
- 1 = Initiates a memory read (Read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software. RD bit cannot be set when EEPGD = 1.)
- 0 = Read completed

**Note 1:** When a WRERR occurs, the EEPGD and CFGS bits are not cleared. This allows tracing of the error condition.

## 7.3 Reading the Data EEPROM Memory

To read a data memory location, the user must write the address to the EEADR register, clear the EEPGD control bit (EECON1<7>) and then set control bit, RD (EECON1<0>). The data is available for the very next instruction cycle; therefore, the EEDATA register can be read by the next instruction. EEDATA will hold this value until another read operation, or until it is written to by the user (during a write operation).

## 7.4 Writing to the Data EEPROM Memory

To write an EEPROM data location, the address must first be written to the EEADR register and the data written to the EEDATA register. The sequence in Example 7-2 must be followed to initiate the write cycle.

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

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

After a write sequence has been initiated, EECON1, EEADR and EEDATA cannot be modified. The WR bit will be inhibited from being set unless the WREN bit is set. The WREN bit must be set on a previous instruction. Both WR and WREN cannot be set with the same instruction.

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

## 7.5 Write Verify

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit.

## 7.6 Protection Against Spurious Write

There are conditions when the device may not want to write to the data EEPROM memory. To protect against spurious EEPROM writes, various mechanisms have been built-in. On power-up, the WREN bit is cleared. Also, the Power-up Timer (72 ms duration) prevents EEPROM write.

The write initiate sequence and the WREN bit together help prevent an accidental write during brown-out, power glitch or software malfunction.

**EXAMPLE 7-1: DATA EEPROM READ**

```
        MOVLW   DATA_EE_ADDR;
        MOVWF   EEADR       ; Data Memory Address to read
        BCF     EECON1, EEPGD; Point to DATA memory
        BSF     EECON1, RD  ; EEPROM Read
        MOVF    EEDATA, W   ; W = EEDATA
```

**EXAMPLE 7-2: DATA EEPROM WRITE**

```
            MOVLW   DATA_EE_ADDR;
            MOVWF   EEADR       ; Data Memory Address to write
            MOVLW   DATA_EE_DATA;
            MOVWF   EEDATA      ; Data Memory Value to write
            BCF     EECON1, EEPGD; Point to DATA memory
            BSF     EECON1, WREN; Enable writes
            BCF     INTCON, GIE; Disable Interrupts
            MOVLW   55h         ;
Required    MOVWF   EECON2      ; Write 55h
Sequence    MOVLW   AAh         ;
            MOVWF   EECON2      ; Write AAh
            BSF     EECON1, WR  ; Set WR bit to begin write
            BSF     INTCON, GIE; Enable Interrupts

            SLEEP               ; Wait for interrupt to signal write complete
            BCF     EECON1, WREN; Disable writes
```


© 2002-2015 Microchip Technology Inc.

## 7.7 Operation During Code-Protect

Data EEPROM memory has its own code-protect bits in Configuration Words. External read and write operations are disabled if either of these mechanisms are enabled.

The microcontroller itself can both read and write to the internal data EEPROM, regardless of the state of the code-protect Configuration bit. Refer to **Section 19.0 "Special Features of the CPU"** for additional information.

## 7.8 Using the Data EEPROM

The data EEPROM is a high endurance, byte addressable array that has been optimized for the storage of frequently changing information (e.g., program variables or other data that are updated often). Frequently changing values will typically be updated more often than specification D124. If this is not the case, an array refresh must be performed. For this reason, variables that change infrequently (such as constants, IDs, calibration, etc.) should be stored in Flash program memory.

A simple data EEPROM refresh routine is shown in Example 7-3.

| **Note:** | If data EEPROM is only used to store constants and/or data that changes rarely, an array refresh is likely not required. See specification D124. |
|---|---|

**EXAMPLE 7-3: DATA EEPROM REFRESH ROUTINE**

```
        CLRF    EEADR               ; Start at address 0
        BCF     EECON1, CFGS        ; Set for memory
        BCF     EECON1, EEPGD       ; Set for Data EEPROM
        BCF     INTCON, GIE         ; Disable interrupts
        BSF     EECON1, WREN        ; Enable writes
Loop                                ; Loop to refresh array
        BSF     EECON1, RD          ; Read current address
        MOVLW   55h                 ;
        MOVWF   EECON2              ; Write 55h
        MOVLW   AAh                 ;
        MOVWF   EECON2              ; Write AAh
        BSF     EECON1, WR          ; Set WR bit to begin write
        BTFSC   EECON1, WR          ; Wait for write to complete
        BRA     $-2
        INCFSZ  EEADR, F            ; Increment address
        BRA     Loop                ; Not zero, do it again

        BCF     EECON1, WREN        ; Disable writes
        BSF     INTCON, GIE         ; Enable interrupts
```

**TABLE 7-1: REGISTERS ASSOCIATED WITH DATA EEPROM MEMORY**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on: POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INTE | RBIE | TMR0IF | INTF | RBIF | 0000 000x | 0000 000u |
| EEADR | EEPROM Address Register | | | | | | | | 0000 0000 | 0000 0000 |
| EEDATA | EEPROM Data Register | | | | | | | | 0000 0000 | 0000 0000 |
| EECON2 | EEPROM Control Register 2 (not a physical register) | | | | | | | | — | — |
| EECON1 | EEPGD | CFGS | — | FREE | WRERR | WREN | WR | RD | xx-0 x000 | uu-0 u000 |
| IPR2 | OSCFIP | — | — | EEIP | — | LVDIP | TMR3IP | — | 1--1 -11- | 1--1 -11- |
| PIR2 | OSCFIF | — | — | EEIF | — | LVDIF | TMR3IF | — | 0--0 -00- | 0--0 -00- |
| PIE2 | OSCFIE | — | — | EEIE | — | LVDIE | TMR3IE | — | 0--0 -00- | 0--0 -00- |

**Legend:** x = unknown, u = unchanged, – = unimplemented, read as '0'. Shaded cells are not used during Flash/EEPROM access.

# 8.0 8 x 8 HARDWARE MULTIPLIER

## 8.1 Introduction

An 8 x 8 hardware multiplier is included in the ALU of the PIC18F1220/1320 devices. By making the multiply a hardware operation, it completes in a single instruction cycle. This is an unsigned multiply that gives a 16-bit result. The result is stored into the 16-bit product register pair (PRODH:PRODL). The multiplier does not affect any flags in the Status register.

Making the 8 x 8 multiplier execute in a single cycle gives the following advantages:

- Higher computational throughput
- Reduces code size requirements for multiply algorithms

The performance increase allows the device to be used in applications previously reserved for Digital Signal Processors.

Table 8-1 shows a performance comparison between Enhanced devices using the single-cycle hardware multiply and performing the same function without the hardware multiply.

**TABLE 8-1: PERFORMANCE COMPARISON**

| Routine | Multiply Method | Program Memory (Words) | Cycles (Max) | Time | | |
|---|---|---|---|---|---|---|
| | | | | @ 40 MHz | @ 10 MHz | @ 4 MHz |
| 8 x 8 unsigned | Without hardware multiply | 13 | 69 | 6.9 µs | 27.6 µs | 69 µs |
| | Hardware multiply | 1 | 1 | 100 ns | 400 ns | 1 µs |
| 8 x 8 signed | Without hardware multiply | 33 | 91 | 9.1 µs | 36.4 µs | 91 µs |
| | Hardware multiply | 6 | 6 | 600 ns | 2.4 µs | 6 µs |
| 16 x 16 unsigned | Without hardware multiply | 21 | 242 | 24.2 µs | 96.8 µs | 242 µs |
| | Hardware multiply | 28 | 28 | 2.8 µs | 11.2 µs | 28 µs |
| 16 x 16 signed | Without hardware multiply | 52 | 254 | 25.4 µs | 102.6 µs | 254 µs |
| | Hardware multiply | 35 | 40 | 4 µs | 16 µs | 40 µs |

## 8.2 Operation

Example 8-1 shows the sequence to do an 8 x 8 unsigned multiply. Only one instruction is required when one argument of the multiply is already loaded in the WREG register.

Example 8-2 shows the sequence to do an 8 x 8 signed multiply. To account for the sign bits of the arguments, each argument's Most Significant bit (MSb) is tested and the appropriate subtractions are done.

**EXAMPLE 8-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE**

```
MOVF    ARG1, W     ;
MULWF   ARG2        ; ARG1 * ARG2 ->
                    ; PRODH:PRODL
```

**EXAMPLE 8-2: 8 x 8 SIGNED MULTIPLY ROUTINE**

```
MOVF    ARG1, W
MULWF   ARG2        ; ARG1 * ARG2 ->
                    ; PRODH:PRODL
BTFSC   ARG2, SB    ; Test Sign Bit
SUBWF   PRODH, F    ; PRODH = PRODH
                    ;         - ARG1
MOVF    ARG2, W
BTFSC   ARG1, SB    ; Test Sign Bit
SUBWF   PRODH, F    ; PRODH = PRODH
                    ;         - ARG2
```


© 2002-2015 Microchip Technology Inc.

Example 8-3 shows the sequence to do a 16 x 16 unsigned multiply. Equation 8-1 shows the algorithm that is used. The 32-bit result is stored in four registers, RES3:RES0.

**EQUATION 8-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM**

RES3:RES0
= ARG1H:ARG1L • ARG2H:ARG2L
= (ARG1H • ARG2H • $2^{16}$) +
(ARG1H • ARG2L • $2^{8}$) +
(ARG1L • ARG2H • $2^{8}$) +
(ARG1L • ARG2L)

**EXAMPLE 8-3: 16 x 16 UNSIGNED MULTIPLY ROUTINE**

```
    MOVFARG1L, W
    MULWFARG2L          ; ARG1L * ARG2L ->
                        ; PRODH:PRODL
    MOVFFPRODH, RES1    ;
    MOVFFPRODL, RES0    ;
;
    MOVFARG1H, W
    MULWFARG2H          ; ARG1H * ARG2H ->
                        ; PRODH:PRODL
    MOVFFPRODH, RES3    ;
    MOVFFPRODL, RES2    ;
;
    MOVFARG1L, W
    MULWFARG2H          ; ARG1L * ARG2H ->
                        ; PRODH:PRODL
    MOVFPRODL, W        ;
    ADDWFRES1, F        ; Add cross
    MOVFPRODH, W        ; products
    ADDWFCRES2, F       ;
    CLRFWREG            ;
    ADDWFCRES3,F        ;
;
    MOVFARG1H, W        ;
    MULWFARG2L          ; ARG1H * ARG2L ->
                        ; PRODH:PRODL
    MOVFPRODL, W        ;
    ADDWFRES1, F        ; Add cross
    MOVFPRODH, W        ; products
    ADDWFCRES2, F       ;
    CLRFWREG            ;
    ADDWFCRES3, F       ;
```

Example 8-4 shows the sequence to do a 16 x 16 signed multiply. Equation 8-2 shows the algorithm used. The 32-bit result is stored in four registers, RES3:RES0. To account for the sign bits of the arguments, each argument pairs' Most Significant bit (MSb) is tested and the appropriate subtractions are done.

**EQUATION 8-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM**

RES3:RES0
= ARG1H:ARG1L • ARG2H:ARG2L
= (ARG1H • ARG2H • $2^{16}$) +
(ARG1H • ARG2L • $2^{8}$) +
(ARG1L • ARG2H • $2^{8}$) +
(ARG1L • ARG2L) +
(-1 • ARG2H<7> • ARG1H:ARG1L • $2^{16}$) +
(-1 • ARG1H<7> • ARG2H:ARG2L • $2^{16}$)

**EXAMPLE 8-4: 16 x 16 SIGNED MULTIPLY ROUTINE**

```
    MOVF    ARG1L, W
    MULWF   ARG2L           ; ARG1L * ARG2L ->
                            ; PRODH:PRODL
    MOVFF   PRODH, RES1     ;
    MOVFF   PRODL, RES0     ;
;
    MOVF    ARG1H, W
    MULWF   ARG2H           ; ARG1H * ARG2H ->
                            ; PRODH:PRODL
    MOVFF   PRODH, RES3     ;
    MOVFF   PRODL, RES2     ;
;
    MOVF    ARG1L, W
    MULWF   ARG2H           ; ARG1L * ARG2H ->
                            ; PRODH:PRODL
    MOVF    PRODL, W        ;
    ADDWF   RES1, F         ; Add cross
    MOVF    PRODH, W        ; products
    ADDWFC  RES2, F         ;
    CLRF    WREG            ;
    ADDWFC  RES3, F         ;
;
    MOVF    ARG1H, W        ;
    MULWF   ARG2L           ; ARG1H * ARG2L ->
                            ; PRODH:PRODL
    MOVF    PRODL, W        ;
    ADDWF   RES1, F         ; Add cross
    MOVF    PRODH, W        ; products
    ADDWFC  RES2, F         ;
    CLRF    WREG            ;
    ADDWFC  RES3, F         ;
;
    BTFSS   ARG2H, 7        ; ARG2H:ARG2L neg?
    BRA     SIGN_ARG1       ; no, check ARG1
    MOVF    ARG1L, W        ;
    SUBWF   RES2            ;
    MOVF    ARG1H, W        ;
    SUBWFB  RES3
;
SIGN_ARG1
    BTFSS   ARG1H, 7        ; ARG1H:ARG1L neg?
    BRA     CONT_CODE       ; no, done
    MOVF    ARG2L, W        ;
    SUBWF   RES2            ;
    MOVF    ARG2H, W        ;
    SUBWFB  RES3
;
CONT_CODE
   :
```

## 9.0 INTERRUPTS

The PIC18F1220/1320 devices have multiple interrupt sources and an interrupt priority feature that allows each interrupt source to be assigned a high priority level or a low priority level. The high priority interrupt vector is at 000008h and the low priority interrupt vector is at 000018h. High priority interrupt events will interrupt any low priority interrupts that may be in progress.

There are ten registers which are used to control interrupt operation. These registers are:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2
- PIE1, PIE2
- IPR1, IPR2

It is recommended that the Microchip header files supplied with MPLAB® IDE be used for the symbolic bit names in these registers. This allows the assembler/compiler to automatically take care of the placement of these bits within the specified register.

In general, each interrupt source has three bits to control its operation. The functions of these bits are:

- Flag bit to indicate that an interrupt event occurred
- Enable bit that allows program execution to branch to the interrupt vector address when the flag bit is set
- Priority bit to select high priority or low priority (INT0 has no priority bit and is always high priority)

The interrupt priority feature is enabled by setting the IPEN bit (RCON<7>). When interrupt priority is enabled, there are two bits which enable interrupts globally. Setting the GIEH bit (INTCON<7>) enables all interrupts that have the priority bit set (high priority). Setting the GIEL bit (INTCON<6>) enables all interrupts that have the priority bit cleared (low priority). When the interrupt flag, enable bit and appropriate Global Interrupt Enable bit are set, the interrupt will vector immediately to address 000008h or 000018h, depending on the priority bit setting. Individual interrupts can be disabled through their corresponding enable bits.

When the IPEN bit is cleared (default state), the interrupt priority feature is disabled and interrupts are compatible with PIC mid-range devices. In Compatibility mode, the interrupt priority bits for each source have no effect. INTCON<6> is the PEIE bit, which enables/disables all peripheral interrupt sources. INTCON<7> is the GIE bit, which enables/disables all interrupt sources. All interrupts branch to address 000008h in Compatibility mode.

When an interrupt is responded to, the Global Interrupt Enable bit is cleared to disable further interrupts. If the IPEN bit is cleared, this is the GIE bit. If interrupt priority levels are used, this will be either the GIEH or GIEL bit. High priority interrupt sources can interrupt a low priority interrupt. Low priority interrupts are not processed while high priority interrupts are in progress.

The return address is pushed onto the stack and the PC is loaded with the interrupt vector address (000008h or 000018h). Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bits must be cleared in software before re-enabling interrupts to avoid recursive interrupts.

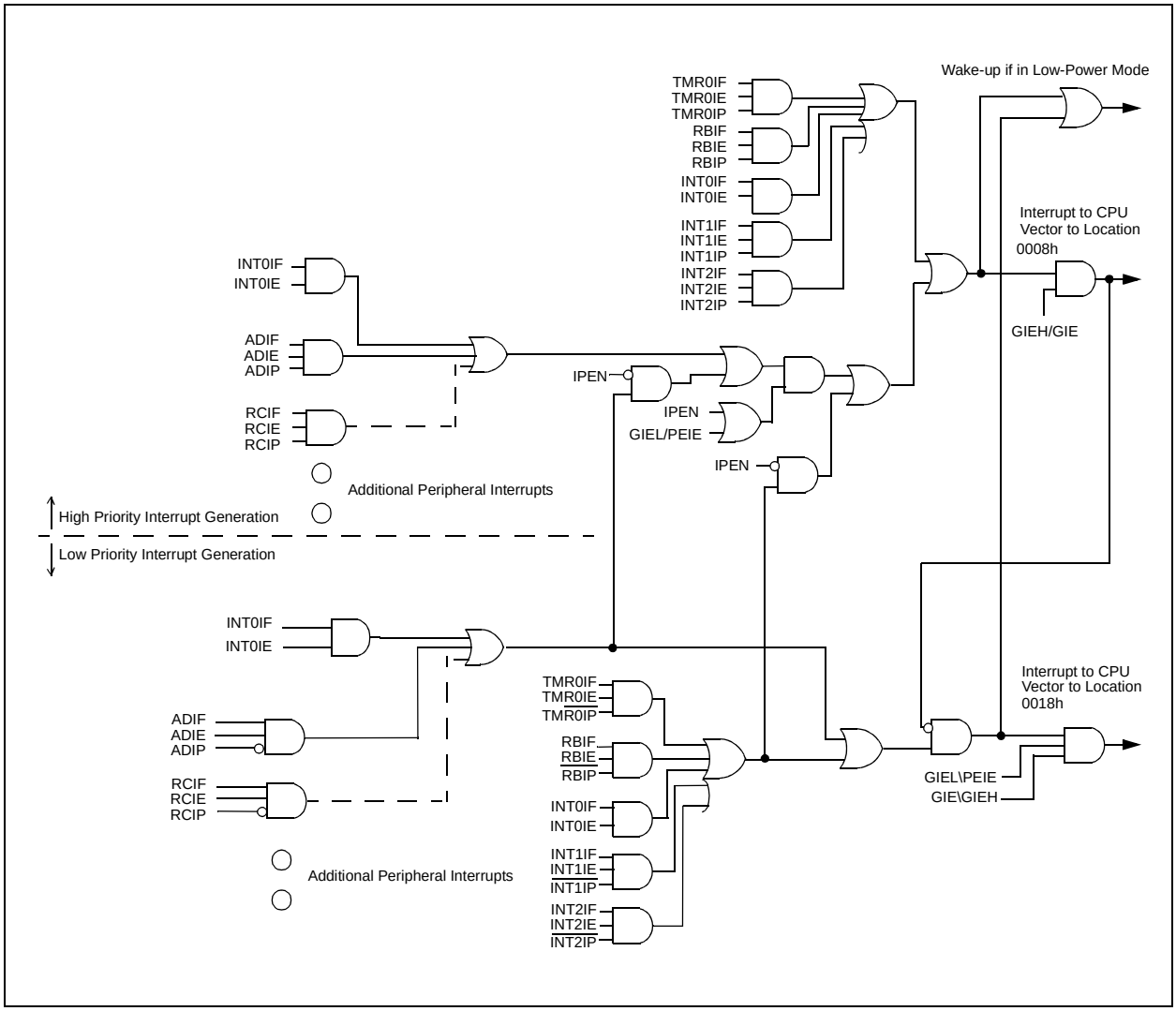
The “return from interrupt” instruction, `RETFIE`, exits the interrupt routine and sets the GIE bit (GIEH or GIEL, if priority levels are used), which re-enables interrupts.

For external interrupt events, such as the INT pins or the PORTB input change interrupt, the interrupt latency will be three to four instruction cycles. The exact latency is the same for one or 2-cycle instructions. Individual interrupt flag bits are set, regardless of the status of their corresponding enable bit or the GIE bit.

| **Note:** | Do not use the `MOVFF` instruction to modify any of the interrupt control registers while **any** interrupt is enabled. Doing so may cause erratic microcontroller behavior. |
|---|---|


© 2002-2015 Microchip Technology Inc.

**FIGURE 9-1: INTERRUPT LOGIC**



## 9.1 INTCON Registers

The INTCON registers are readable and writable registers, which contain various enable, priority and flag bits.

**Note:** Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the Global Interrupt Enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

**REGISTER 9-1: INTCON: INTERRUPT CONTROL REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R-0/0 |
|---|---|---|---|---|---|---|---|
| GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF(1) |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7 **GIE/GIEH:** Global Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked interrupts
0 = Disables all interrupts
When IPEN = 1:
1 = Enables all high priority interrupts
0 = Disables all interrupts

bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts
When IPEN = 1:
1 = Enables all low priority peripheral interrupts
0 = Disables all low priority peripheral interrupts

bit 5 **TMR0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 overflow interrupt
0 = Disables the TMR0 overflow interrupt

bit 4 **INT0IE:** INT0 External Interrupt Enable bit
1 = Enables the INT0 external interrupt
0 = Disables the INT0 external interrupt

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

bit 2 **TMR0IF:** Timer0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed
0 = TMR0 register did not overflow

bit 1 **INT0IF:** INT0 External Interrupt Flag bit
1 = The INT0 external interrupt occurred (must be cleared in software)
0 = The INT0 external interrupt did not occur

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

**Note 1:** A mismatch condition will continue to set this bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared.


© 2002-2015 Microchip Technology Inc.

**REGISTER 9-2: INTCON2: INTERRUPT CONTROL REGISTER 2**

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | U-0 | R/W-1/1 | U-0 | R/W-1/1 |
|---|---|---|---|---|---|---|---|
| $\overline{\text{RBPU}}$ | INTEDG0 | INTEDG1 | INTEDG2 | — | TMR0IP | — | RBIP |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = All PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG0:** External Interrupt 0 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 5 **INTEDG1:** External Interrupt 1 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 4 **INTEDG2:** External Interrupt 2 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 3 **Unimplemented:** Read as ‘0’

bit 2 **TMR0IP:** TMR0 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **Unimplemented:** Read as ‘0’

bit 0 **RBIP:** RB Port Change Interrupt Priority bit
1 = High priority
0 = Low priority

**Note:** Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the Global Interrupt Enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

**REGISTER 9-3: INTCON3: INTERRUPT CONTROL REGISTER 3**

| R/W-1/1 | R/W-1/1 | U-0 | R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| INT2IP | INT1IP | — | INT2IE | INT1IE | — | INT2IF | INT1IF |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **INT2IP:** INT2 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **INT1IP:** INT1 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **Unimplemented:** Read as ‘0’

bit 4 **INT2IE:** INT2 External Interrupt Enable bit
1 = Enables the INT2 external interrupt
0 = Disables the INT2 external interrupt

bit 3 **INT1IE:** INT1 External Interrupt Enable bit
1 = Enables the INT1 external interrupt
0 = Disables the INT1 external interrupt

bit 2 **Unimplemented:** Read as ‘0’

bit 1 **INT2IF:** INT2 External Interrupt Flag bit
1 = The INT2 external interrupt occurred (must be cleared in software)
0 = The INT2 external interrupt did not occur

bit 0 **INT1IF:** INT1 External Interrupt Flag bit
1 = The INT1 external interrupt occurred (must be cleared in software)
0 = The INT1 external interrupt did not occur

**Note:** Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the Global Interrupt Enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.


© 2002-2015 Microchip Technology Inc.

## 9.2 PIR Registers

The PIR registers contain the individual flag bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are two Peripheral Interrupt Request (Flag) registers (PIR1, PIR2).

**Note 1:** Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the Global Interrupt Enable bit, GIE (INTCON<7>).

**2:** User software should ensure the appropriate interrupt flag bits are cleared prior to enabling an interrupt and after servicing that interrupt.

**REGISTER 9-4: PIR1: PERIPHERAL INTERRUPT REQUEST REGISTER 1**

| U-0 | R/W-0/0 | R-0/0 | R-0/0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | ADIF | RCIF | TXIF | — | CCP1IF | TMR2IF | TMR1IF |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **Unimplemented:** Read as ‘0’

bit 6 **ADIF:** A/D Converter Interrupt Flag bit
1 = An A/D conversion completed (must be cleared in software)
0 =The A/D conversion is not complete

bit 5 **RCIF:** EUSART Receive Interrupt Flag bit
1 = The EUSART receive buffer, RCREG, is full (cleared when RCREG is read)
0 = The EUSART receive buffer is empty

bit 4 **TXIF:** EUSART Transmit Interrupt Flag bit
1 = The EUSART transmit buffer, TXREG, is empty (cleared when TXREG is written)
0 = The EUSART transmit buffer is full

bit 3 **Unimplemented:** Read as ‘0’

bit 2 **CCP1IF:** CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode.

bit 1 **TMR2IF:** TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF:** TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

**REGISTER 9-5: PIR2: PERIPHERAL INTERRUPT REQUEST REGISTER 2**

| R/W-0/0 | U-0 | U-0 | R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 | U-0 |
|---|---|---|---|---|---|---|---|
| OSCFIF | — | — | EEIF | — | LVDIF | TMR3IF | — |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **OSCFIF:** Oscillator Fail Interrupt Flag bit
1 = System oscillator failed, clock input has changed to INTOSC (must be cleared in software)
0 = System clock operating

bit 6-5 **Unimplemented:** Read as ‘0’

bit 4 **EEIF:** Data EEPROM/Flash Write Operation Interrupt Flag bit
1 = The write operation is complete (must be cleared in software)
0 = The write operation is not complete or has not been started

bit 3 **Unimplemented:** Read as ‘0’

bit 2 **LVDIF:** Low-Voltage Detect Interrupt Flag bit
1 = A low-voltage condition occurred (must be cleared in software)
0 = The device voltage is above the Low-Voltage Detect trip point

bit 1 **TMR3IF:** TMR3 Overflow Interrupt Flag bit
1 = TMR3 register overflowed (must be cleared in software)
0 = TMR3 register did not overflow

bit 0 **Unimplemented:** Read as ‘0’


© 2002-2015 Microchip Technology Inc.

## 9.3 PIE Registers

The PIE registers contain the individual enable bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are two Peripheral Interrupt Enable registers (PIE1, PIE2). When IPEN = 0, the PEIE bit must be set to enable any of these peripheral interrupts.

**REGISTER 9-6: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1**

| U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | ADIE | RCIE | TXIE | — | CCP1IE | TMR2IE | TMR1IE |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **Unimplemented:** Read as ‘0’

bit 6 **ADIE:** A/D Converter Interrupt Enable bit
1 = Enables the A/D interrupt
0 = Disables the A/D interrupt

bit 5 **RCIE:** EUSART Receive Interrupt Enable bit
1 = Enables the EUSART receive interrupt
0 = Disables the EUSART receive interrupt

bit 4 **TXIE:** EUSART Transmit Interrupt Enable bit
1 = Enables the EUSART transmit interrupt
0 = Disables the EUSART transmit interrupt

bit 3 **Unimplemented:** Read as ‘0’

bit 2 **CCP1IE:** CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE:** TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE:** TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

**REGISTER 9-7: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2**

| R/W-0/0 | U-0 | U-0 | R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 | U-0 |
|---|---|---|---|---|---|---|---|
| OSCFIE | — | — | EEIE | — | LVDIE | TMR3IE | — |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **OSCFIE:** Oscillator Fail Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6-5 **Unimplemented:** Read as ‘0’

bit 4 **EEIE:** Data EEPROM/Flash Write Operation Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 3 **Unimplemented:** Read as ‘0’

bit 2 **LVDIE:** Low-Voltage Detect Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **TMR3IE:** TMR3 Overflow Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **Unimplemented:** Read as ‘0’


© 2002-2015 Microchip Technology Inc.

## 9.4 IPR Registers

The IPR registers contain the individual priority bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are two Peripheral Interrupt Priority registers (IPR1, IPR2). Using the priority bits requires that the Interrupt Priority Enable (IPEN) bit be set.

**REGISTER 9-8: IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1**

| U-0 | R/W-1/1 | R/W-1/1 | R/W-1/1 | U-0 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---|---|---|---|---|---|---|---|
| — | ADIP | RCIP | TXIP | — | CCP1IP | TMR2IP | TMR1IP |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **Unimplemented:** Read as ‘0’

bit 6 **ADIP:** A/D Converter Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **RCIP:** EUSART Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **TXIP:** EUSART Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **Unimplemented:** Read as ‘0’

bit 2 **CCP1IP:** CCP1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR2IP:** TMR2 to PR2 Match Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR1IP:** TMR1 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

**REGISTER 9-9: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2**

| R/W-1/1 | U-0 | U-0 | R/W-1/1 | U-0 | R/W-1/1 | R/W-1/1 | U-0 |
|---|---|---|---|---|---|---|---|
| OSCFIP | — | — | EEIP | — | LVDIP | TMR3IP | — |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **OSCFIP:** Oscillator Fail Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6-5 **Unimplemented:** Read as ‘0’

bit 4 **EEIP:** Data EEPROM/Flash Write Operation Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **Unimplemented:** Read as ‘0’

bit 2 **LVDIP:** Low-Voltage Detect Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR3IP:** TMR3 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **Unimplemented:** Read as ‘0’


© 2002-2015 Microchip Technology Inc.

## 9.5 RCON Register

The RCON register contains bits used to determine the cause of the last Reset or wake-up from a low-power mode. RCON also contains the bit that enables interrupt priorities (IPEN).

**REGISTER 9-10: RCON: RESET CONTROL REGISTER**

| R/W-0 | U-0 | U-0 | R/W-1 | R-1 | R-1 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| IPEN | — | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **IPEN:** Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)

bit 6-5 **Unimplemented:** Read as ‘0’

bit 4 **RI:** RESET Instruction Flag bit
For details of bit operation, see Register 5-3.

bit 3 **TO:** Watchdog Time-out Flag bit
For details of bit operation, see Register 5-3.

bit 2 **PD:** Power-down Detection Flag bit
For details of bit operation, see Register 5-3.

bit 1 **POR:** Power-on Reset Status bit
For details of bit operation, see Register 5-3.

bit 0 **BOR:** Brown-out Reset Status bit
For details of bit operation, see Register 5-3.

© 2002-2015 Microchip Technology Inc.

## 9.6 INTn Pin Interrupts

External interrupts on the RB0/INT0, RB1/INT1 and RB2/INT2 pins are edge-triggered: either rising if the corresponding INTEDGx bit is set in the INTCON2 register, or falling if the INTEDGx bit is clear. When a valid edge appears on the RBx/INTx pin, the corresponding flag bit, INTxF, is set. This interrupt can be disabled by clearing the corresponding enable bit, INTxE. Flag bit, INTxF, must be cleared in software in the Interrupt Service Routine before re-enabling the interrupt. All external interrupts (INT0, INT1 and INT2) can wake-up the processor from low-power modes if bit INTxE was set prior to going into low-power modes. If the Global Interrupt Enable bit, GIE, is set, the processor will branch to the interrupt vector following wake-up.

Interrupt priority for INT1 and INT2 is determined by the value contained in the interrupt priority bits, INT1IP (INTCON3<6>) and INT2IP (INTCON3<7>). There is no priority bit associated with INT0. It is always a high priority interrupt source.

## 9.7 TMR0 Interrupt

In 8-bit mode (which is the default), an overflow (FFh → 00h) in the TMR0 register will set flag bit, TMR0IF. In 16-bit mode, an overflow (FFFFh → 0000h) in the TMR0H:TMR0L registers will set flag bit, TMR0IF. The interrupt can be enabled/disabled by setting/clearing enable bit, TMR0IE (INTCON<5>). Interrupt priority for Timer0 is determined by the value contained in the interrupt priority bit, TMR0IP (INTCON2<2>). See **Section 11.0 "Timer0 Module"** for further details on the Timer0 module.

## 9.8 PORTB Interrupt-on-Change

An input change on PORTB<7:4> sets flag bit, RBIF (INTCON<0>). The interrupt can be enabled/disabled by setting/clearing enable bit, RBIE (INTCON<3>). Interrupt priority for PORTB interrupt-on-change is determined by the value contained in the interrupt priority bit, RBIP (INTCON2<0>).

## 9.9 Context Saving During Interrupts

During interrupts, the return PC address is saved on the stack. Additionally, the WREG, Status and BSR registers are saved on the fast return stack. If a fast return from interrupt is not used (see **Section 5.3 "Fast Register Stack"**), the user may need to save the WREG, Status and BSR registers on entry to the Interrupt Service Routine. Depending on the user's application, other registers may also need to be saved. Example 9-1 saves and restores the WREG, Status and BSR registers during an Interrupt Service Routine.

**EXAMPLE 9-1: SAVING STATUS, WREG AND BSR REGISTERS IN RAM**

```
MOVWF   W_TEMP                  ; W_TEMP is in virtual bank
MOVFF   STATUS, STATUS_TEMP     ; STATUS_TEMP located anywhere
MOVFF   BSR, BSR_TEMP           ; BSR_TMEP located anywhere
;
; USER ISR CODE
;
MOVFF   BSR_TEMP, BSR           ; Restore BSR
MOVF    W_TEMP, W               ; Restore WREG
MOVFF   STATUS_TEMP, STATUS     ; Restore STATUS
```


© 2002-2015 Microchip Technology Inc.

## 10.0 I/O PORTS

Depending on the device selected and features enabled, there are up to five ports available. Some pins of the I/O ports are multiplexed with an alternate function from the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

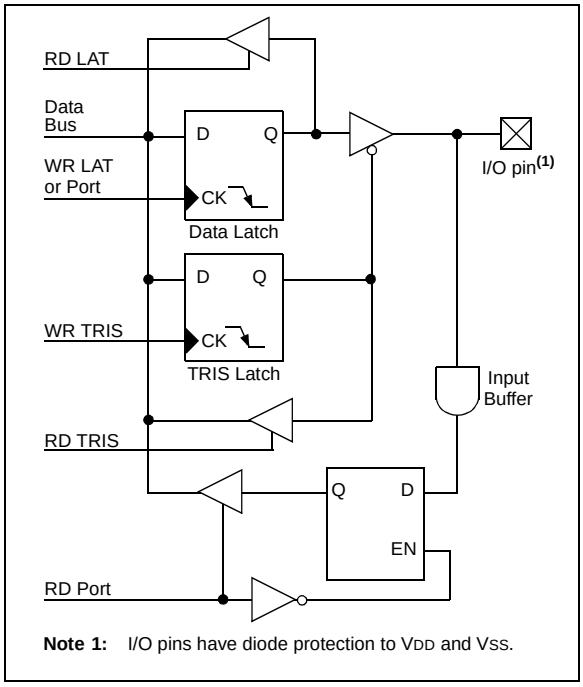
Each port has three registers for its operation. These registers are:

- TRIS register (data direction register)
- PORT register (reads the levels on the pins of the device)
- LAT register (output latch)

The Data Latch (LATA) register is useful for read-modify-write operations on the value that the I/O pins are driving.

A simplified model of a generic I/O port without the interfaces to other peripherals is shown in Figure 10-1.

**FIGURE 10-1: GENERIC I/O PORT OPERATION**




**Note 1:** I/O pins have diode protection to VDD and VSS.

## 10.1 PORTA, TRISA and LATA Registers

PORTA is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISA. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., put the contents of the output latch on the selected pin).

Reading the PORTA register reads the status of the pins, whereas writing to it will write to the port latch.

The Data Latch register (LATA) is also memory mapped. Read-modify-write operations on the LATA register read and write the latched output value for PORTA.

The RA4 pin is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin.

The sixth pin of PORTA ($\overline{\text{MCLR}}$/VPP/RA5) is an input only pin. Its operation is controlled by the MCLRE configuration bit in Configuration Register 3H (CONFIG3H<7>). When selected as a port pin (MCLRE = 0), it functions as a digital input only pin; as such, it does not have TRIS or LAT bits associated with its operation. Otherwise, it functions as the device's Master Clear input. In either configuration, RA5 also functions as the programming voltage input during programming.

| **Note:** | On a Power-on Reset, RA5 is enabled as a digital input only if Master Clear functionality is disabled. |
|---|---|

Pins RA6 and RA7 are multiplexed with the main oscillator pins; they are enabled as oscillator or I/O pins by the selection of the main oscillator in Configuration Register 1H (see **Section 19.1 "Configuration Bits"** for details). When they are not used as port pins, RA6 and RA7 and their associated TRIS and LAT bits are read as '0'.

The other PORTA pins are multiplexed with analog inputs, the analog VREF+ and VREF- inputs and the LVD input. The operation of pins RA3:RA0 as A/D converter inputs is selected by clearing/setting the control bits in the ADCON1 register (A/D Control Register 1).

| **Note:** | On a Power-on Reset, RA3:RA0 are configured as analog inputs and read as '0'. RA4 is always a digital pin. |
|---|---|

The RA4/T0CKI pin is a Schmitt Trigger input and an open-drain output. All other PORTA pins have TTL input levels and full CMOS output drivers.

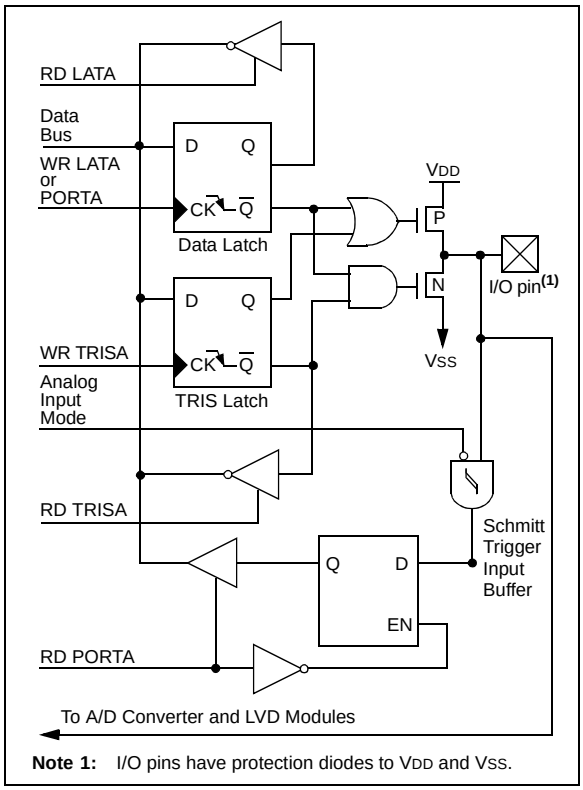
The TRISA register controls the direction of the RA pins, even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

**EXAMPLE 10-1: INITIALIZING PORTA**

```
CLRF    PORTA   ; Initialize PORTA by
                ; clearing output
                ; data latches
CLRF    LATA    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0x7F    ; Configure A/D
MOVWF   ADCON1  ; for digital inputs
MOVLW   0xD0    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISA   ; Set RA<3:0> as outputs
                ; RA<7:4> as inputs
```

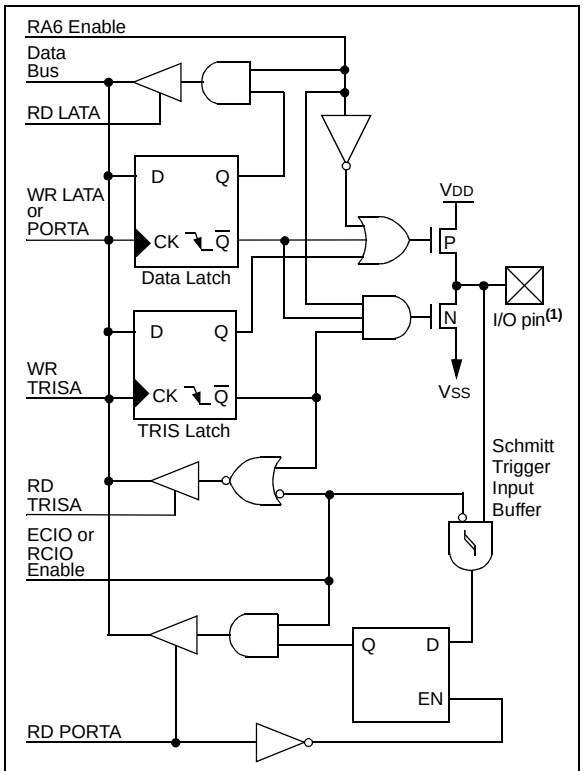
© 2002-2015 Microchip Technology Inc.

**FIGURE 10-2: BLOCK DIAGRAM OF RA3:RA0 PINS**



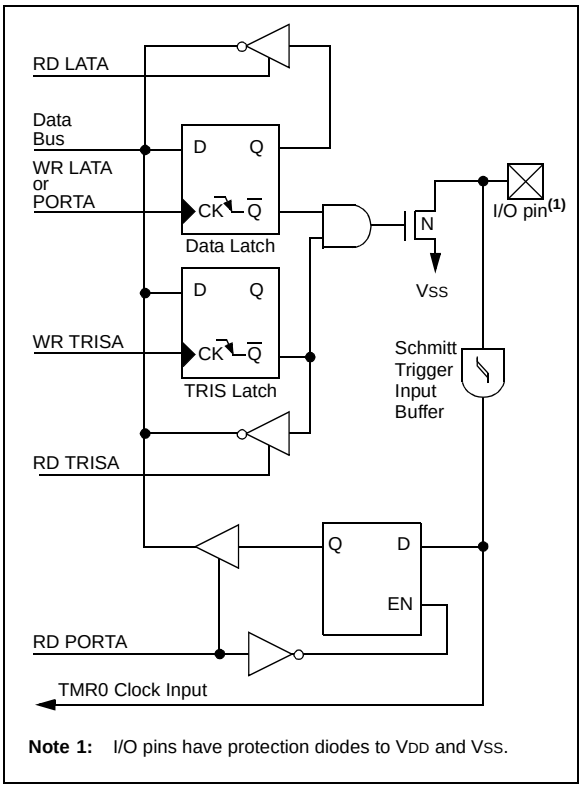

**Note 1:** I/O pins have protection diodes to VDD and VSS.

**FIGURE 10-3: BLOCK DIAGRAM OF OSC2/CLKO/RA6 PIN**



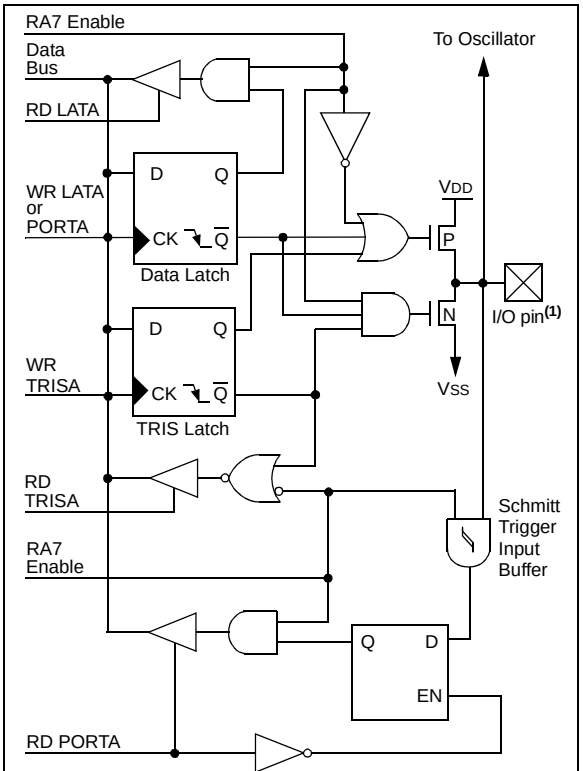

**Note 1:** I/O pins have protection diodes to VDD and VSS.

**FIGURE 10-4: BLOCK DIAGRAM OF RA4/T0CKI PIN**




**Note 1:** I/O pins have protection diodes to VDD and VSS.

**FIGURE 10-5: BLOCK DIAGRAM OF OSC1/CLKI/RA7 PIN**




**Note 1:** I/O pins have protection diodes to VDD and VSS.


© 2002-2015 Microchip Technology Inc.

**FIGURE 10-6: $\overline{\text{MCLR}}$/VPP/RA5 PIN BLOCK DIAGRAM**

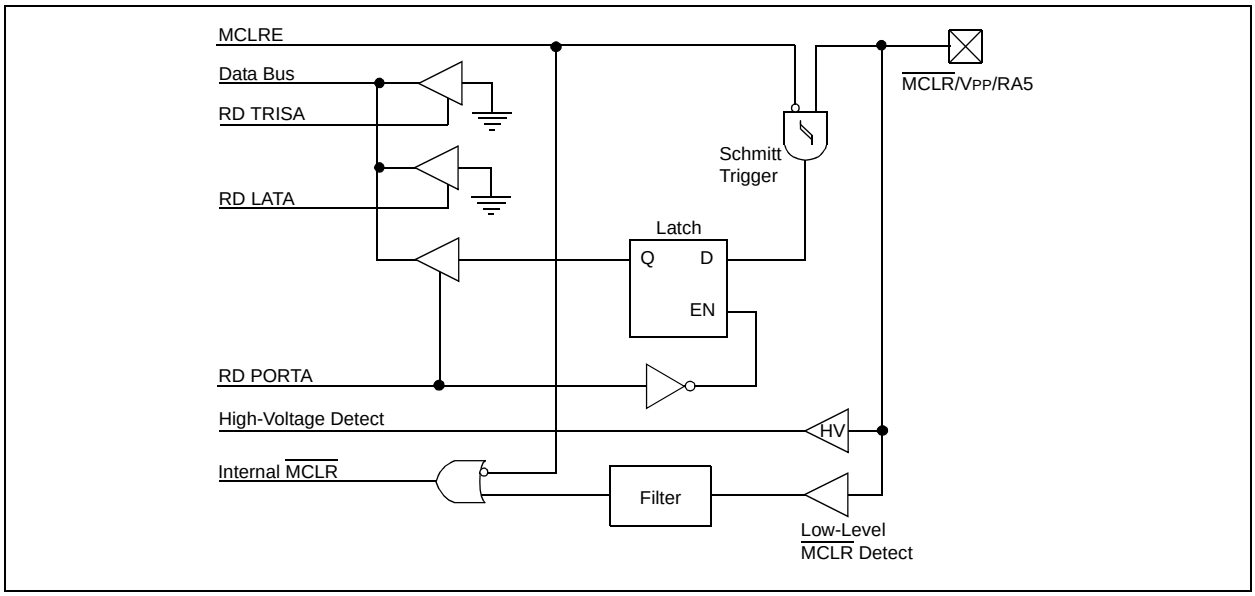



**TABLE 10-1: PORTA FUNCTIONS**

| Name | Bit# | Buffer | Function |
|---|---|---|---|
| RA0/AN0 | bit 0 | ST | Input/output port pin or analog input. |
| RA1/AN1/LVDIN | bit 1 | ST | Input/output port pin, analog input or Low-Voltage Detect input. |
| RA2/AN2/VREF- | bit 2 | ST | Input/output port pin, analog input or VREF-. |
| RA3/AN3/VREF+ | bit 3 | ST | Input/output port pin, analog input or VREF+. |
| RA4/T0CKI | bit 4 | ST | Input/output port pin or external clock input for Timer0. Output is open-drain type. |
| $\overline{\text{MCLR}}$/VPP/RA5 | bit 5 | ST | Master Clear input or programming voltage input (if $\overline{\text{MCLR}}$ is enabled); input only port pin or programming voltage input (if $\overline{\text{MCLR}}$ is disabled). |
| OSC2/CLKO/RA6 | bit 6 | ST | OSC2, clock output or I/O pin. |
| OSC1/CLKI/RA7 | bit 7 | ST | OSC1, clock input or I/O pin. |

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

**TABLE 10-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| PORTA | RA7(1) | RA6(1) | RA5(2) | RA4 | RA3 | RA2 | RA1 | RA0 | xx0x 0000 | uu0u 0000 |
| LATA | LATA7(1) | LATA6(1) | — | LATA Data Output Register | | | | | xx-x xxxx | uu-u uuuu |
| TRISA | TRISA7(1) | TRISA6(1) | — | PORTA Data Direction Register | | | | | 11-1 1111 | 11-1 1111 |
| ADCON1 | — | PCFG6 | PCFG5 | PCFG4 | PCFG3 | PCFG2 | PCFG1 | PCFG0 | -000 0000 | -000 0000 |

**Legend:** x = unknown, u = unchanged, – = unimplemented locations read as '0'. Shaded cells are not used by PORTA.

**Note 1:** RA7:RA6 and their associated latch and data direction bits are enabled as I/O pins based on oscillator configuration; otherwise, they are read as '0'.

**2:** RA5 is an input only if $\overline{\text{MCLR}}$ is disabled.

## 10.2 PORTB, TRISB and LATB Registers

PORTB is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISB. Setting a TRISB bit (= 1) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a High-impedance mode). Clearing a TRISB bit (= 0) will make the corresponding PORTB pin an output (i.e., put the contents of the output latch on the selected pin).

The Data Latch register (LATB) is also memory mapped. Read-modify-write operations on the LATB register read and write the latched output value for PORTB.

**EXAMPLE 10-2: INITIALIZING PORTB**

```
CLRF    PORTB   ; Initialize PORTB by
                ; clearing output
                ; data latches
CLRF    LATB    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0x70    ; Set RB0, RB1, RB4 as
MOVWF   ADCON1  ; digital I/O pins
MOVLW   0xCF    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISB   ; Set RB<3:0> as inputs
                ; RB<5:4> as outputs
                ; RB<7:6> as inputs
```

Pins RB0-RB2 are multiplexed with INT0-INT2; pins RB0, RB1 and RB4 are multiplexed with A/D inputs; pins RB1 and RB4 are multiplexed with EUSART; and pins RB2, RB3, RB6 and RB7 are multiplexed with ECCP.

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit, $\overline{\text{RBPU}}$ (INTCON2<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

| | |
|---|---|
| **Note:** | On a Power-on Reset, RB4:RB0 are configured as analog inputs by default and read as '0'; RB7:RB5 are configured as digital inputs. |

Four of the PORTB pins (RB7:RB4) have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'ed together to generate the RB Port Change Interrupt with Flag bit, RBIF (INTCON<0>).

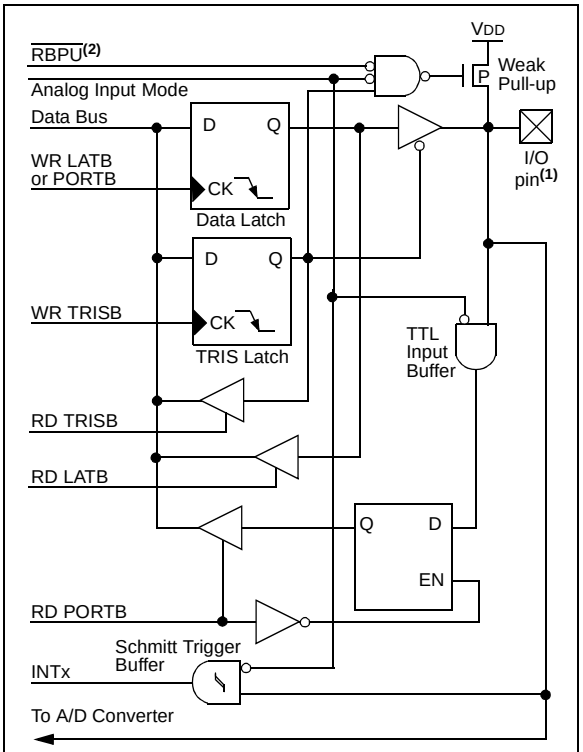
This interrupt can wake the device from Sleep. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of PORTB (except with the MOVFF instruction). This will end the mismatch condition.
b) Clear flag bit, RBIF.

A mismatch condition will continue to set flag bit, RBIF. Reading PORTB will end the mismatch condition and allow flag bit, RBIF, to be cleared.

The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt-on-change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

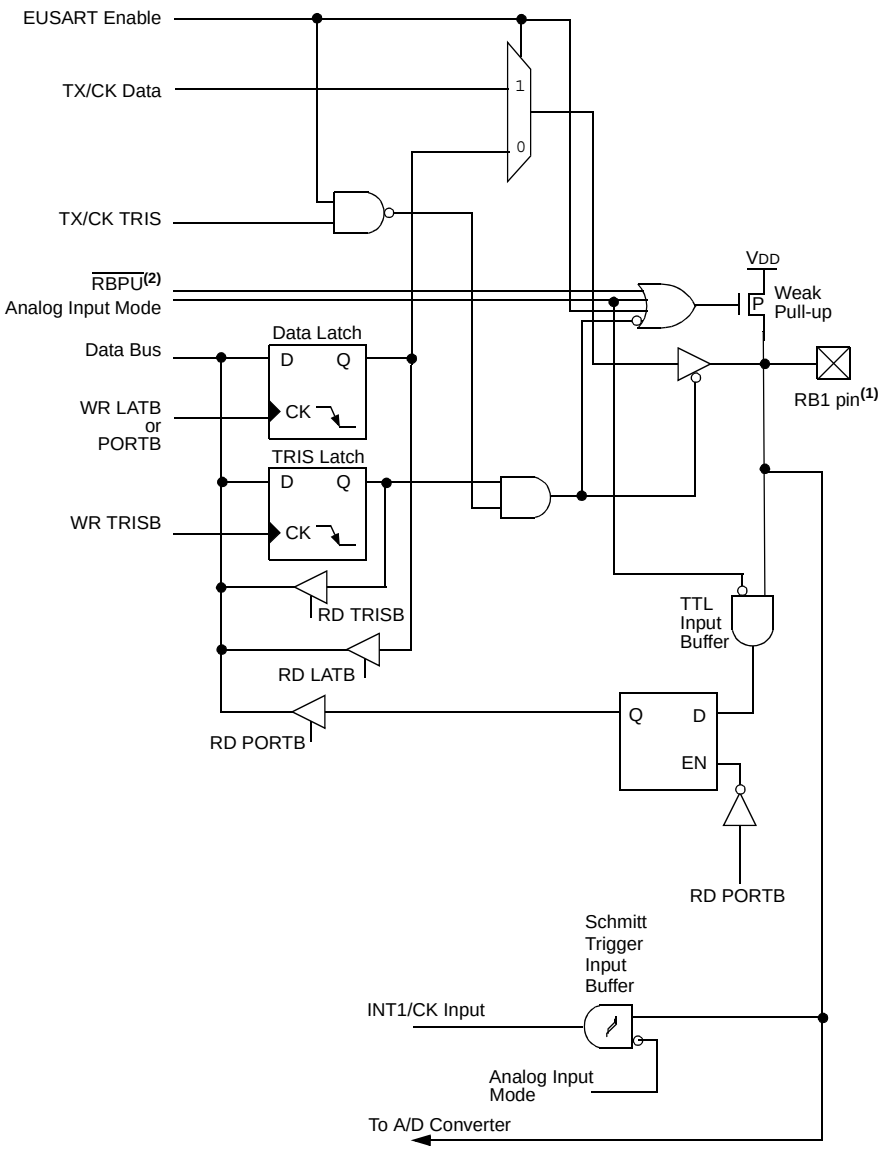
**FIGURE 10-7: BLOCK DIAGRAM OF RB0/AN4/INT0 PIN**




**Note 1:** I/O pins have diode protection to VDD and VSS.
**2:** To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (INTCON2<7>).


© 2002-2015 Microchip Technology Inc.

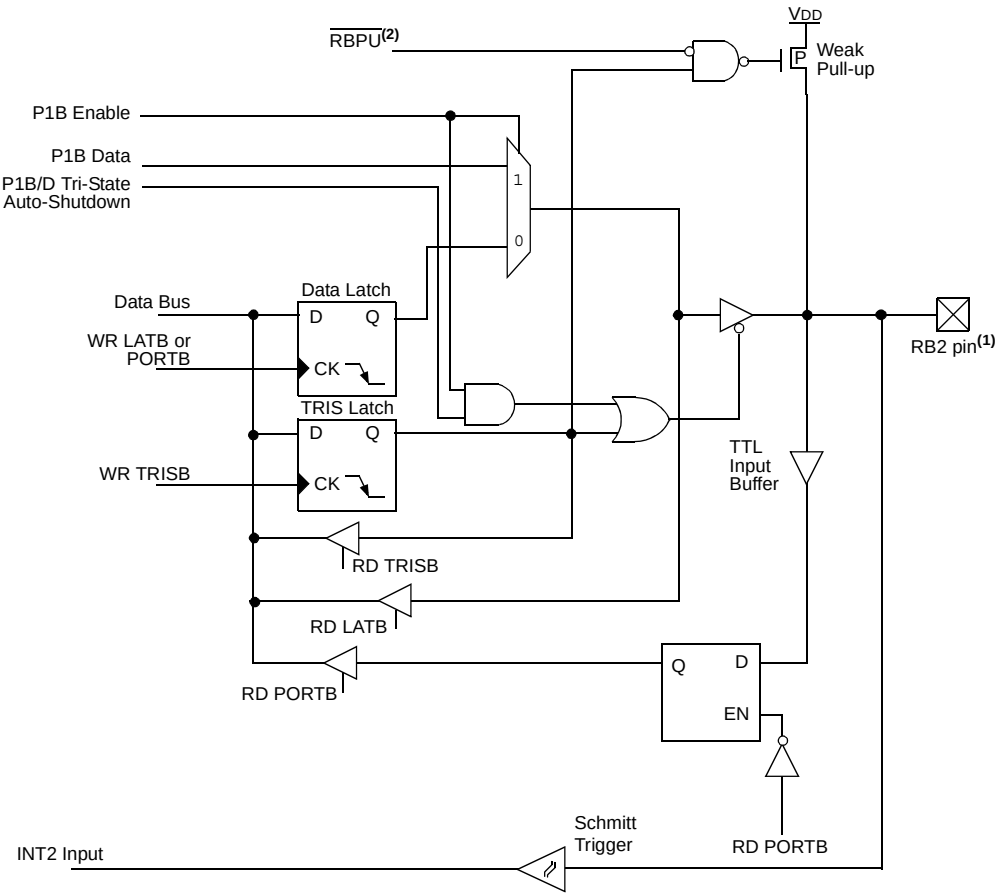
**FIGURE 10-8: BLOCK DIAGRAM OF RB1/AN5/TX/CK/INT1 PIN**




**Note 1:** I/O pins have diode protection to VDD and VSS.

**2:** To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (INTCON2<7>).

**FIGURE 10-9: BLOCK DIAGRAM OF RB2/P1B/INT2 PIN**

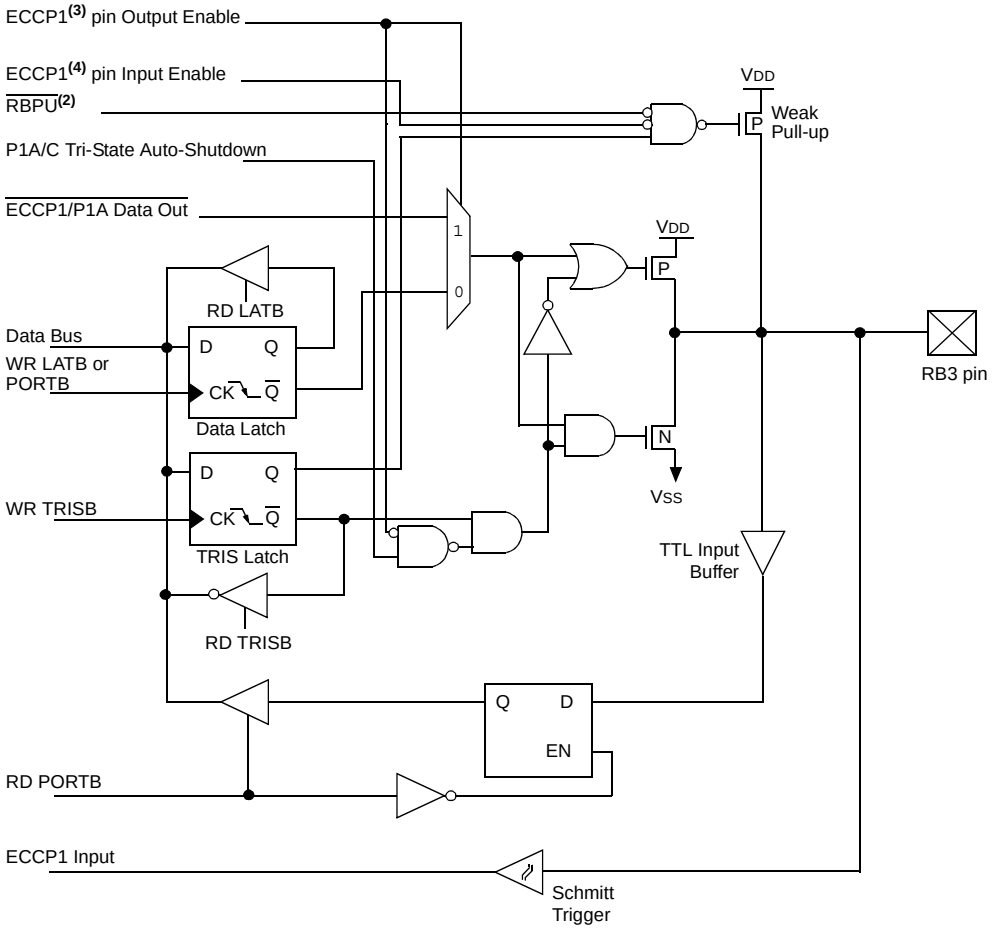



**Note 1:** I/O pins have diode protection to VDD and VSS.

**2:** To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (INTCON2<7>).


© 2002-2015 Microchip Technology Inc.

**FIGURE 10-10: BLOCK DIAGRAM OF RB3/CCP1/P1A PIN**



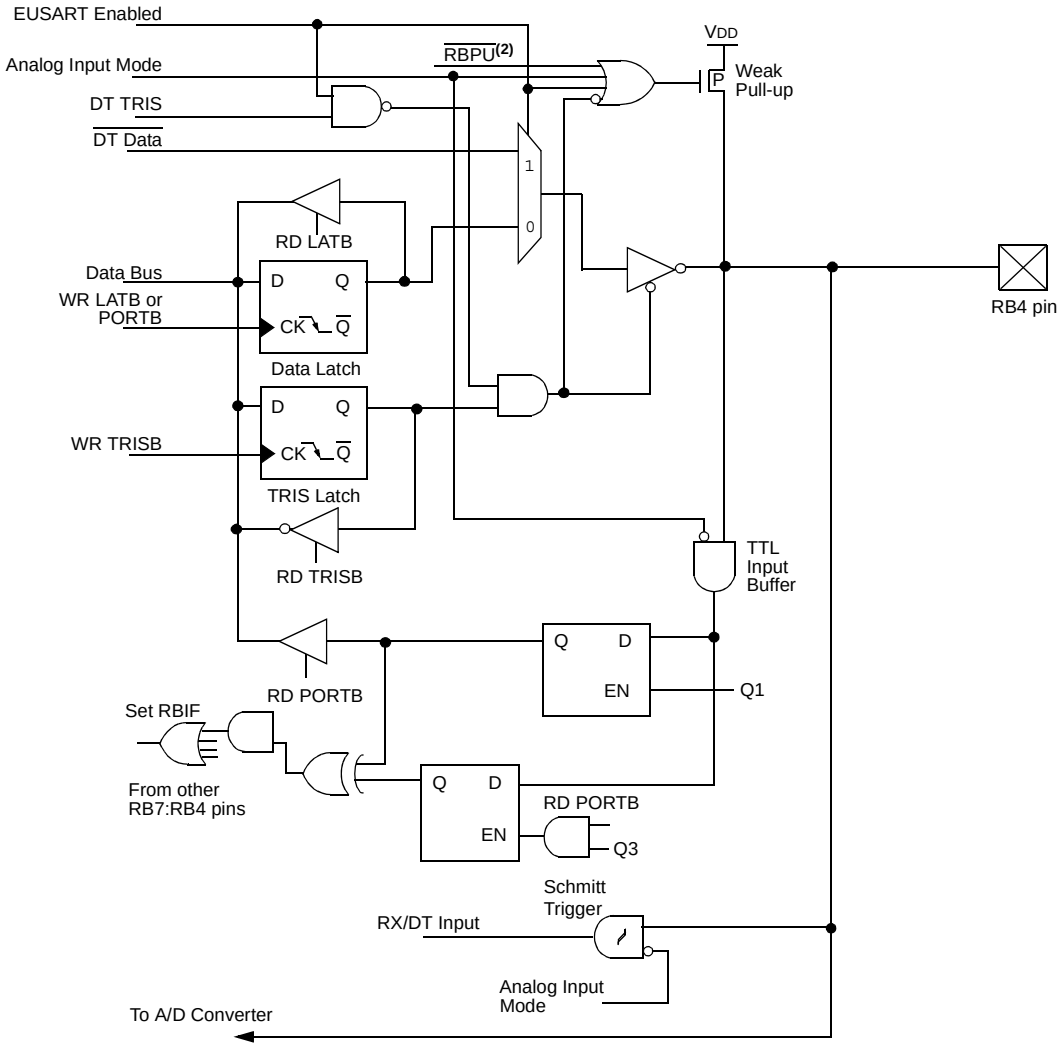

**Note 1:** I/O pins have diode protection to VDD and VSS.

**2:** To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the RBPU bit (INTCON2<7>).

**3:** ECCP1 pin output enable active for any PWM mode and Compare mode, where CCP1M<3:0> = 1000 or 1001.

**4:** ECCP1 pin input enable active for Capture mode only.

**FIGURE 10-11: BLOCK DIAGRAM OF RB4/AN6/RX/DT/KBI0 PIN**

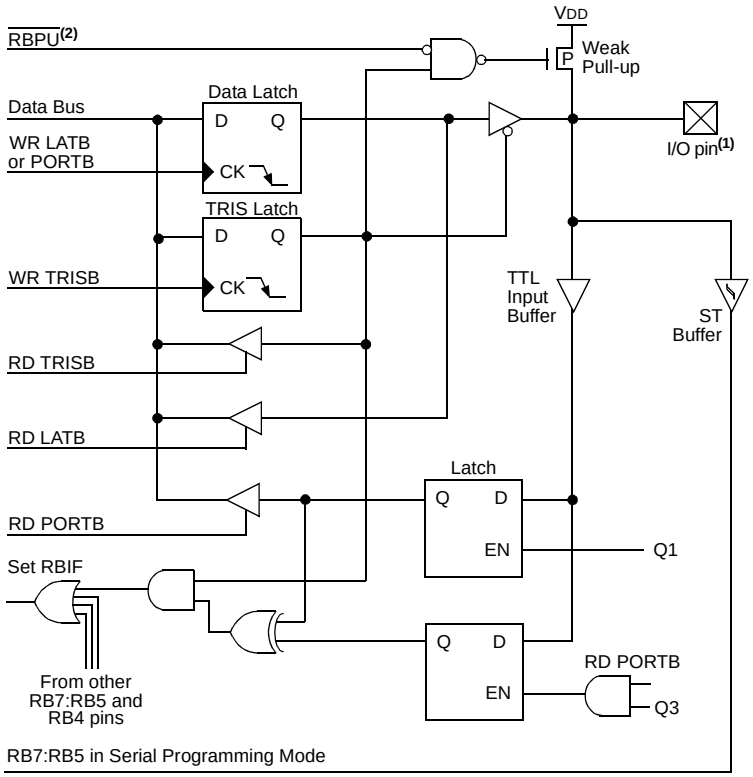



**Note 1:** I/O pins have diode protection to VDD and VSS.

**2:** To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (INTCON2<7>).


© 2002-2015 Microchip Technology Inc.

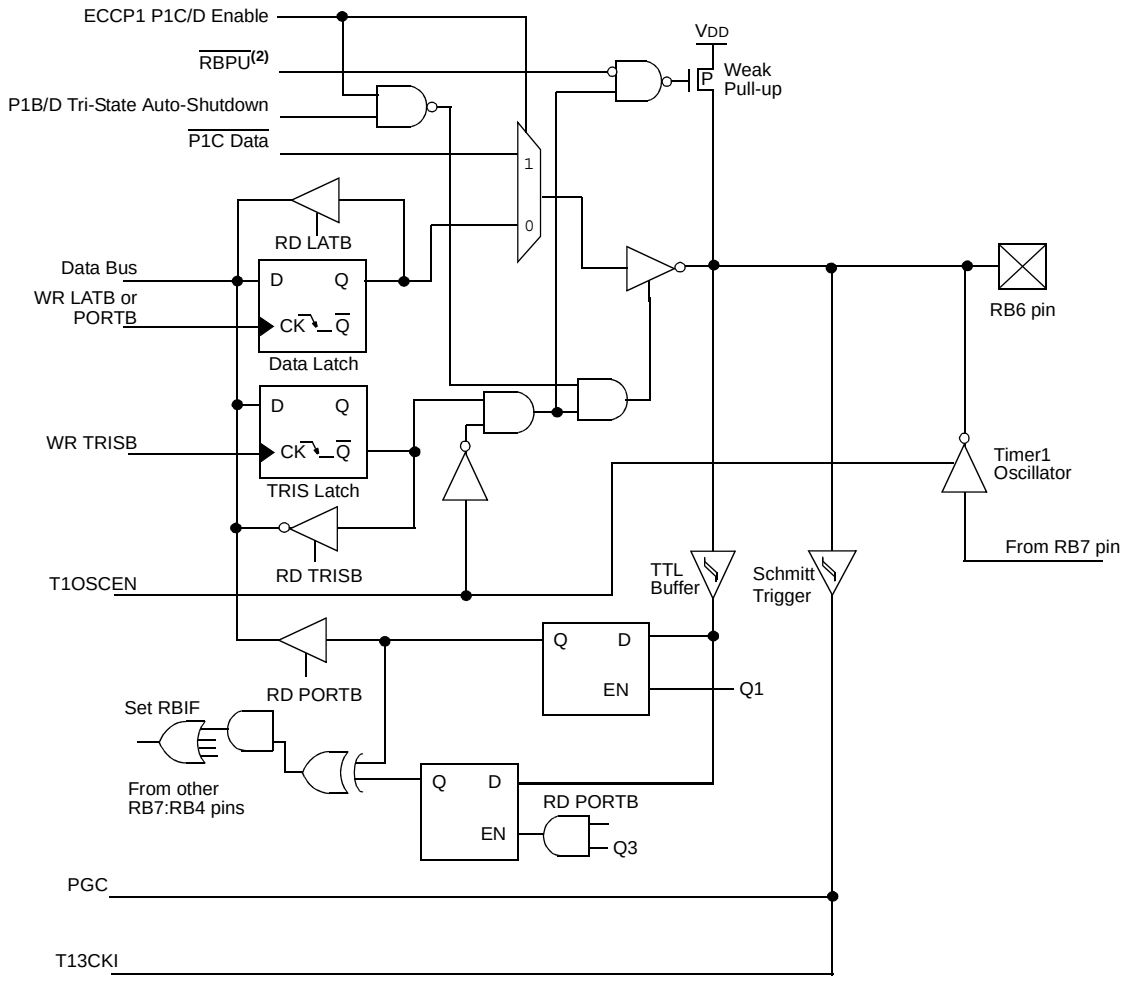
**FIGURE 10-12: BLOCK DIAGRAM OF RB5/PGM/KBI1 PIN**




**Note 1:** I/O pins have diode protection to VDD and VSS.

**2:** To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (INTCON2<7>).

**FIGURE 10-13: BLOCK DIAGRAM OF RB6/PGC/T1OSO/T13CKI/P1C/KBI2 PIN**

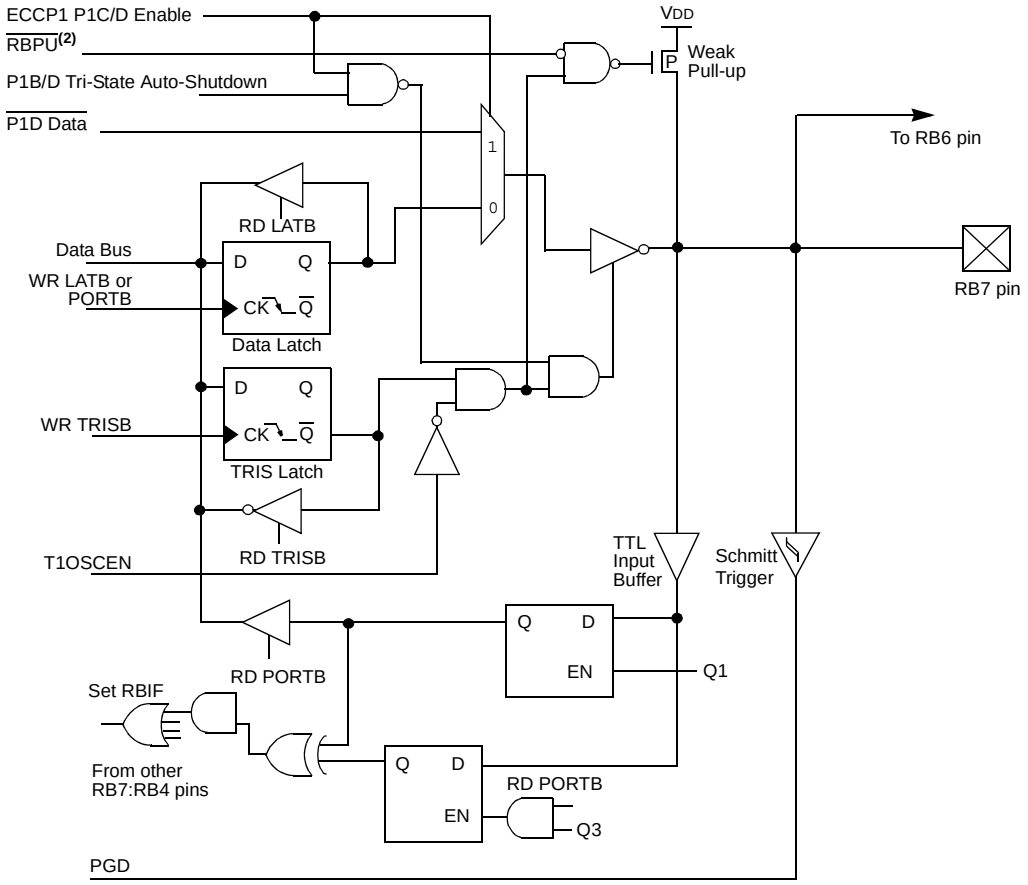



**Note 1:** I/O pins have diode protection to VDD and VSS.

**2:** To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (INTCON2<7>).


© 2002-2015 Microchip Technology Inc.

**FIGURE 10-14: BLOCK DIAGRAM OF RB7/PGD/T1OSI/P1D/KBI3 PIN**




**Note 1:** I/O pins have diode protection to VDD and VSS.

**2:** To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (INTCON2<7>).

**TABLE 10-3: PORTB FUNCTIONS**

| Name | Bit# | Buffer | Function |
|---|---|---|---|
| RB0/AN4/INT0 | bit 0 | TTL(1)/ST(2) | Input/output port pin, analog input or external interrupt input 0. |
| RB1/AN5/TX/CK/INT1 | bit 1 | TTL(1)/ST(2) | Input/output port pin, analog input, Enhanced USART Asynchronous Transmit, Addressable USART Synchronous Clock or external interrupt input 1. |
| RB2/P1B/INT2 | bit 2 | TTL(1)/ST(2) | Input/output port pin or external interrupt input 2. Internal software programmable weak pull-up. |
| RB3/CCP1/P1A | bit 3 | TTL(1)/ST(3) | Input/output port pin or Capture1 input/Compare1 output/ PWM output. Internal software programmable weak pull-up. |
| RB4/AN6/RX/DT/KBI0 | bit 4 | TTL(1)/ST(4) | Input/output port pin (with interrupt-on-change), analog input, Enhanced USART Asynchronous Receive or Addressable USART Synchronous Data. |
| RB5/PGM/KBI1 | bit 5 | TTL(1)/ST(5) | Input/output port pin (with interrupt-on-change). Internal software programmable weak pull-up. Low-Voltage ICSP™ enable pin. |
| RB6/PGC/T1OSO/T13CKI/ P1C/KBI2 | bit 6 | TTL(1)/ST(5,6) | Input/output port pin (with interrupt-on-change), Timer1/ Timer3 clock input or Timer1oscillator output. Internal software programmable weak pull-up. Serial programming clock. |
| RB7/PGD/T1OSI/P1D/KBI3 | bit 7 | TTL(1)/ST(5) | Input/output port pin (with interrupt-on-change) or Timer1 oscillator input. Internal software programmable weak pull-up. Serial programming data. |

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

**Note 1:** This buffer is a TTL input when configured as a port input pin.

**2:** This buffer is a Schmitt Trigger input when configured as the external interrupt.

**3:** This buffer is a Schmitt Trigger input when configured as the CCP1 input.

**4:** This buffer is a Schmitt Trigger input when used as EUSART receive input.

**5:** This buffer is a Schmitt Trigger input when used in Serial Programming mode.

**6:** This buffer is a TTL input when used as the T13CKI input.

**TABLE 10-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| PORTB | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 | xxxq qqqq | uuuu uuuu |
| LATB | LATB Data Output Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| TRISB | PORTB Data Direction Register | | | | | | | | 1111 1111 | 1111 1111 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 0000 000u |
| INTCON2 | $\overline{\text{RBPU}}$ | INTEDG0 | INTEDG1 | INTEDG2 | — | TMR0IP | — | RBIP | 1111 -1-1 | 1111 -1-1 |
| INTCON3 | INT2IP | INT1IP | — | INT2IE | INT1IE | — | INT2IF | INT1IF | 11-0 0-00 | 11-0 0-00 |
| ADCON1 | — | PCFG6 | PCFG5 | PCFG4 | PCFG3 | PCFG2 | PCFG1 | PCFG0 | -000 0000 | -000 0000 |

**Legend:** x = unknown, u = unchanged, q = value depends on condition. Shaded cells are not used by PORTB.


© 2002-2015 Microchip Technology Inc.

## 11.0 TIMER0 MODULE

The Timer0 module has the following features:

- Software selectable as an 8-bit or 16-bit timer/counter
- Readable and writable
- Dedicated 8-bit software programmable prescaler
- Clock source selectable to be external or internal
- Interrupt-on-overflow from FFh to 00h in 8-bit mode and FFFFh to 0000h in 16-bit mode
- Edge select for external clock

Figure 11-1 shows a simplified block diagram of the Timer0 module in 8-bit mode and Figure 11-2 shows a simplified block diagram of the Timer0 module in 16-bit mode.

The T0CON register (Register 11-1) is a readable and writable register that controls all the aspects of Timer0, including the prescale selection.

**REGISTER 11-1: T0CON: TIMER0 CONTROL REGISTER**

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---|---|---|---|---|---|---|---|
| TMR0ON | T08BIT | T0CS | T0SE | PSA | T0PS2 | T0PS1 | T0PS0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT:** Timer0 8-bit/16-bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

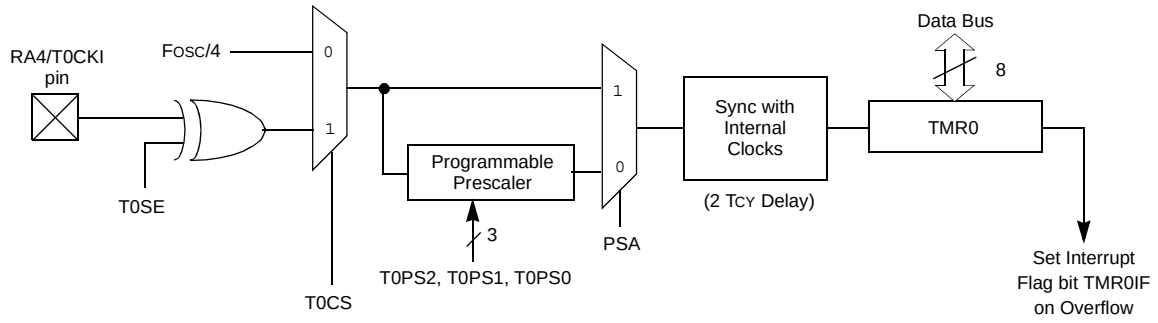
bit 5 **T0CS:** Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE:** Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA:** Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

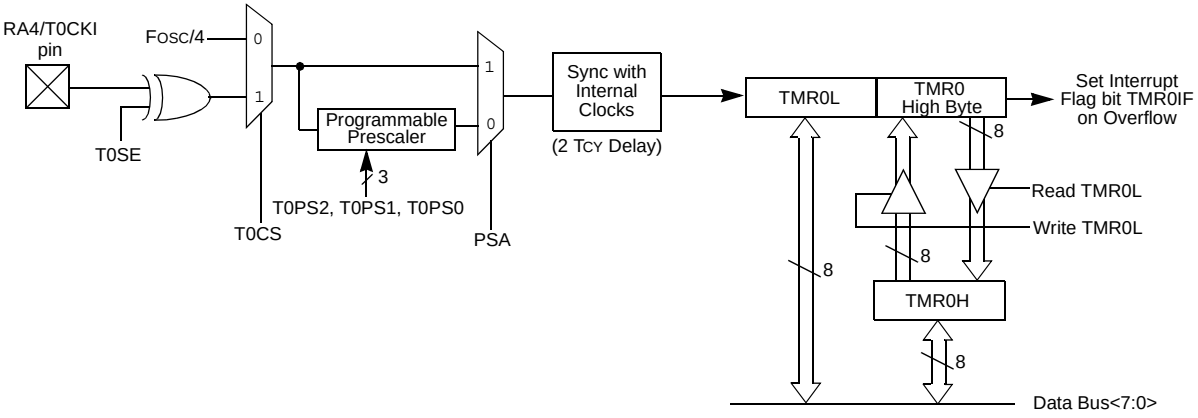
bit 2-0 **T0PS<2:0>:** Timer0 Prescaler Select bits
111 = 1:256 Prescale value
110 = 1:128 Prescale value
101 = 1:64 Prescale value
100 = 1:32 Prescale value
011 = 1:16 Prescale value
010 = 1:8 Prescale value
001 = 1:4 Prescale value
000 = 1:2 Prescale value

**FIGURE 11-1: TIMER0 BLOCK DIAGRAM IN 8-BIT MODE**




**Note:** Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI maximum prescale.

**FIGURE 11-2: TIMER0 BLOCK DIAGRAM IN 16-BIT MODE**




**Note:** Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI maximum prescale.


© 2002-2015 Microchip Technology Inc.

## 11.1 Timer0 Operation

Timer0 can operate as a timer or as a counter.

Timer mode is selected by clearing the T0CS bit. In Timer mode, the Timer0 module will increment every instruction cycle (without prescaler). If the TMR0 register is written, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0 register.

Counter mode is selected by setting the T0CS bit. In Counter mode, Timer0 will increment either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit (T0SE). Clearing the T0SE bit selects the rising edge.

When an external clock input is used for Timer0, it must meet certain requirements. The requirements ensure the external clock can be synchronized with the internal phase clock (TOSC). Also, there is a delay in the actual incrementing of Timer0 after synchronization.

## 11.2 Prescaler

An 8-bit counter is available as a prescaler for the Timer0 module. The prescaler is not readable or writable.

The PSA and T0PS2:T0PS0 bits determine the prescaler assignment and prescale ratio.

Clearing bit PSA will assign the prescaler to the Timer0 module. When the prescaler is assigned to the Timer0 module, prescale values of 1:2, 1:4, ..., 1:256 are selectable.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g., `CLRF TMR0`, `MOVWF TMR0`, `BSF TMR0, x`, ..., etc.) will clear the prescaler count.

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

### 11.2.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control (i.e., it can be changed “on-the-fly” during program execution).

## 11.3 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h in 8-bit mode, or FFFFh to 0000h in 16-bit mode. This overflow sets the TMR0IF bit. The interrupt can be masked by clearing the TMR0IE bit. The TMR0IF bit must be cleared in software by the Timer0 module Interrupt Service Routine before re-enabling this interrupt. The TMR0 interrupt cannot awaken the processor from Low-Power Sleep mode, since the timer requires clock cycles even when T0CS is set.

## 11.4 16-Bit Mode Timer Reads and Writes

TMR0H is not the high byte of the timer/counter in 16-bit mode, but is actually a buffered version of the high byte of Timer0 (refer to Figure 11-2). The high byte of the Timer0 counter/timer is not directly readable nor writable. TMR0H is updated with the contents of the high byte of Timer0 during a read of TMR0L. This provides the ability to read all 16 bits of Timer0, without having to verify that the read of the high and low byte were valid due to a rollover between successive reads of the high and low byte.

A write to the high byte of Timer0 must also take place through the TMR0H Buffer register. Timer0 high byte is updated with the contents of TMR0H when a write occurs to TMR0L. This allows all 16 bits of Timer0 to be updated at once.

**TABLE 11-1: REGISTERS ASSOCIATED WITH TIMER0**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| TMR0L | Timer0 Module Low Byte Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| TMR0H | Timer0 Module High Byte Register | | | | | | | | 0000 0000 | 0000 0000 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 0000 000u |
| T0CON | TMR0ON | T08BIT | T0CS | T0SE | PSA | T0PS2 | T0PS1 | T0PS0 | 1111 1111 | 1111 1111 |
| TRISA | RA7(1) | RA6(1) | — | PORTA Data Direction Register | | | | | 11-1 1111 | 11-1 1111 |

**Legend:** x = unknown, u = unchanged, – = unimplemented locations read as ‘0’. Shaded cells are not used by Timer0.

**Note 1:** RA6 and RA7 are enabled as I/O pins, depending on the oscillator mode selected in Configuration Word 1H.

## 12.0 TIMER1 MODULE

The Timer1 module timer/counter has the following features:

- 16-bit timer/counter
  (two 8-bit registers: TMR1H and TMR1L)
- Readable and writable (both registers)
- Internal or external clock select
- Interrupt-on-overflow from FFFFh to 0000h
- Reset from CCP module special event trigger
- Status of system clock operation

Figure 12-1 is a simplified block diagram of the Timer1 module.

Register 12-1 details the Timer1 Control register. This register controls the operating mode of the Timer1 module and contains the Timer1 Oscillator Enable bit (T1OSCEN). Timer1 can be enabled or disabled by setting or clearing control bit, TMR1ON (T1CON<0>).

The Timer1 oscillator can be used as a secondary clock source in power managed modes. When the T1RUN bit is set, the Timer1 oscillator is providing the system clock. If the Fail-Safe Clock Monitor is enabled and the Timer1 oscillator fails while providing the system clock, polling the T1RUN bit will indicate whether the clock is being provided by the Timer1 oscillator or another source.

Timer1 can also be used to provide Real-Time Clock (RTC) functionality to applications, with only a minimal addition of external components and code overhead.

**REGISTER 12-1: T1CON: TIMER1 CONTROL REGISTER**

| R/W-0/0 | R-1/1 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **RD16:** 16-bit Read/Write Mode Enable bit
1 = Enables register read/write of TImer1 in one 16-bit operation
0 = Enables register read/write of Timer1 in two 8-bit operations

bit 6 **T1RUN:** Timer1 System Clock Status bit
1 = System clock is derived from Timer1 oscillator
0 = System clock is derived from another source

bit 5-4 **T1CKPS<1:0>:** Timer1 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T1OSCEN:** Timer1 Oscillator Enable bit
1 = Timer1 oscillator is enabled
0 = Timer1 oscillator is shut off
The oscillator inverter and feedback resistor are turned off to eliminate power drain.

bit 2 **$\overline{\text{T1SYNC}}$:** Timer1 External Clock Input Synchronization Select bit
When TMR1CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS:** Timer1 Clock Source Select bit
1 = External clock from pin RB6/PGC/T1OSO/T13CKI/P1C/KBI2 (on the rising edge)
0 = Internal clock (Fosc/4)

bit 0 **TMR1ON:** Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1


© 2002-2015 Microchip Technology Inc.

## 12.1 Timer1 Operation

Timer1 can operate in one of these modes:

- As a timer
- As a synchronous counter
- As an asynchronous counter

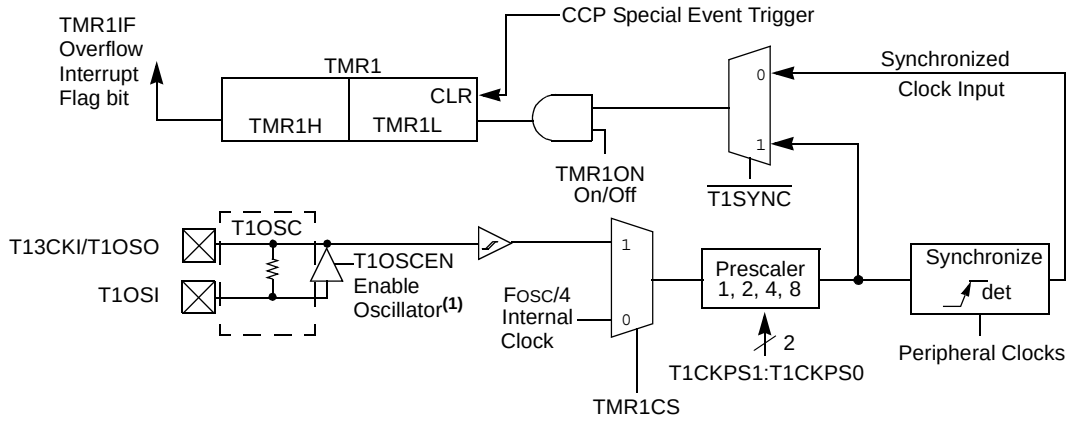
The operating mode is determined by the clock select bit, TMR1CS (T1CON<1>).

When TMR1CS = 0, Timer1 increments every instruction cycle. When TMR1CS = 1, Timer1 increments on every rising edge of the external clock input, or the Timer1 oscillator, if enabled.

When the Timer1 oscillator is enabled (T1OSCEN is set), the RB7/PGD/T1OSI/P1D/KBI3 and RB6/T1OSO/T13CKI/P1C/KBI2 pins become inputs. That is, the TRISB7:TRISB6 values are ignored and the pins read as '0'.

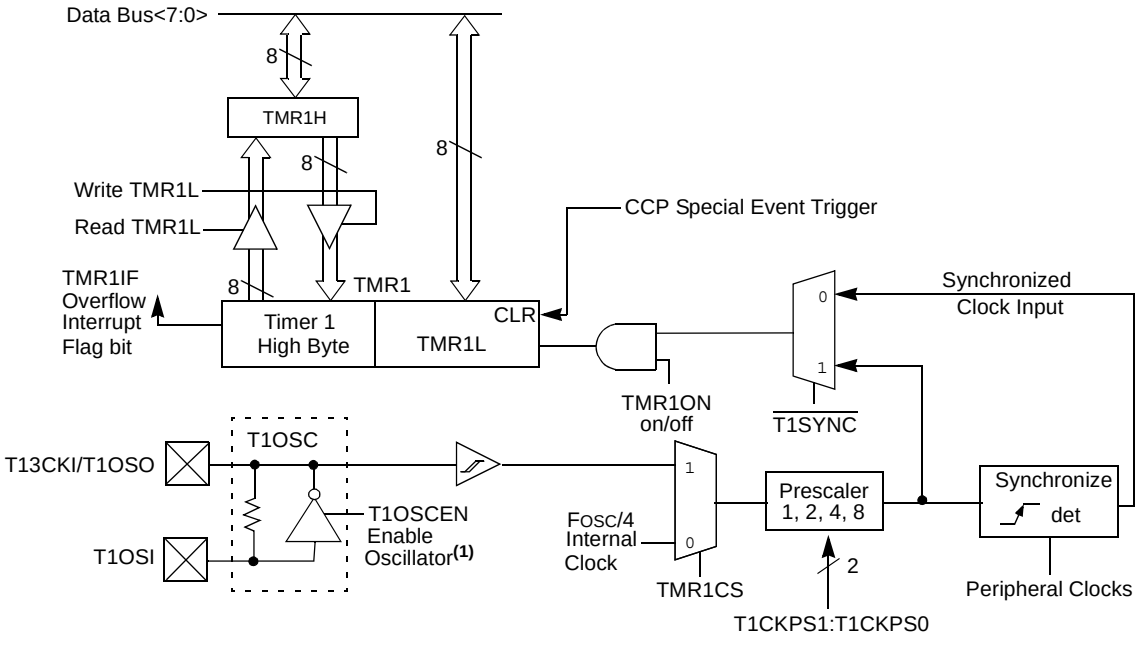
Timer1 also has an internal "Reset input". This Reset can be generated by the CCP module (see Section 15.4.4 "Special Event Trigger").

**FIGURE 12-1: TIMER1 BLOCK DIAGRAM**




**Note 1:** When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.

**FIGURE 12-2: TIMER1 BLOCK DIAGRAM: 16-BIT READ/WRITE MODE**




**Note 1:** When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.

## 12.2 Timer1 Oscillator

A crystal oscillator circuit is built-in between pins T1OSI (input) and T1OSO (amplifier output). It is enabled by setting control bit, T1OSCEN (T1CON<3>). The oscillator is a low-power oscillator rated for 32 kHz crystals. It will continue to run during all power managed modes. The circuit for a typical LP oscillator is shown in Figure 12-3. Table 12-1 shows the capacitor selection for the Timer1 oscillator.

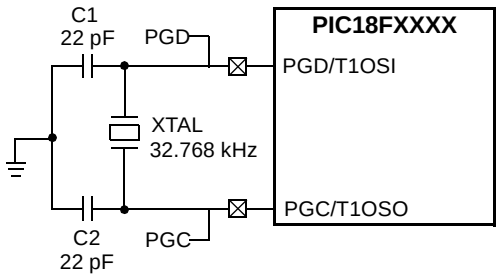
The user must provide a software time delay to ensure proper start-up of the Timer1 oscillator.

**Note:** The Timer1 oscillator shares the T1OSI and T1OSO pins with the PGD and PGC pins used for programming and debugging.

When using the Timer1 oscillator, In-Circuit Serial Programming (ICSP) may not function correctly (high voltage or low voltage), or the In-Circuit Debugger (ICD) may not communicate with the controller. As a result of using either ICSP or ICD, the Timer1 crystal may be damaged.

If ICSP or ICD operations are required, the crystal should be disconnected from the circuit (disconnect either lead), or installed after programming. The oscillator loading capacitors may remain in-circuit during ICSP or ICD operation.

**FIGURE 12-3: EXTERNAL COMPONENTS FOR THE TIMER1 LP OSCILLATOR**




**Note:** See the Notes with Table 12-1 for additional information about capacitor selection.

**TABLE 12-1: CAPACITOR SELECTION FOR THE TIMER OSCILLATOR**

| Osc Type | Freq | C1 | C2 |
|---|---|---|---|
| LP | 32 kHz | 22 pF(1) | 22 pF(1) |

**Note 1:** Microchip suggests this value as a starting point in validating the oscillator circuit. Oscillator operation should then be tested to ensure expected performance under all expected conditions ($V_{DD}$ and temperature).

**2:** Higher capacitance increases the stability of the oscillator, but also increases the start-up time.

**3:** Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

**4:** Capacitor values are for design guidance only.


© 2002-2015 Microchip Technology Inc.

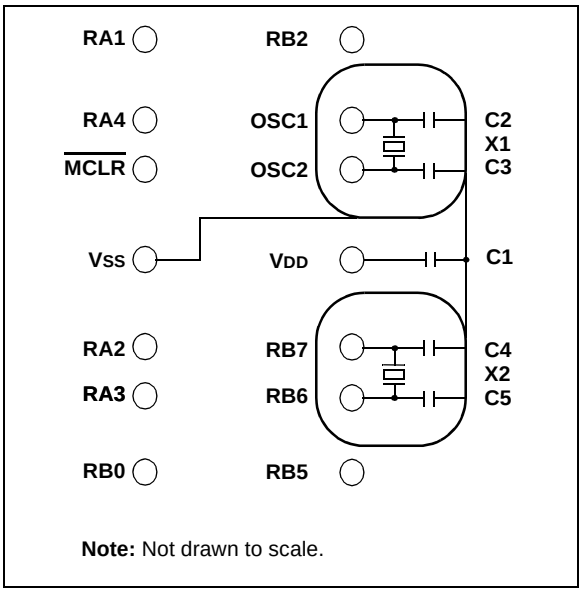
## 12.3 Timer1 Oscillator Layout Considerations

The Timer1 oscillator circuit draws very little power during operation. Due to the low-power nature of the oscillator, it may also be sensitive to rapidly changing signals in close proximity.

The oscillator circuit, shown in Figure 12-3, should be located as close as possible to the microcontroller. There should be no circuits passing within the oscillator circuit boundaries other than VSS or VDD.

If a high-speed circuit must be located near the oscillator (such as the CCP1 pin in output compare or PWM mode, or the primary oscillator using the OSC2 pin), a grounded guard ring around the oscillator circuit, as shown in Figure 12-4, may be helpful when used on a single sided PCB, or in addition to a ground plane.

**FIGURE 12-4: OSCILLATOR CIRCUIT WITH GROUNDED GUARD RING**




**Note:** Not drawn to scale.

## 12.4 Timer1 Interrupt

The TMR1 register pair (TMR1H:TMR1L) increments from 0000h to FFFFh and rolls over to 0000h. The Timer1 interrupt, if enabled, is generated on overflow, which is latched in interrupt flag bit, TMR1IF (PIR1<0>). This interrupt can be enabled/disabled by setting/clearing Timer1 Interrupt Enable bit, TMR1IE (PIE1<0>).

## 12.5 Resetting Timer1 Using a CCP Trigger Output

If the CCP module is configured in Compare mode to generate a "special event trigger" (CCP1M3:CCP1M0 = 1011), this signal will reset Timer1 and start an A/D conversion, if the A/D module is enabled (see **Section 15.4.4 "Special Event Trigger"** for more information).

| **Note:** | The special event triggers from the CCP1 module will not set interrupt flag bit, TMR1IF (PIR1<0>). |
|---|---|

Timer1 must be configured for either Timer or Synchronized Counter mode to take advantage of this feature. If Timer1 is running in Asynchronous Counter mode, this Reset operation may not work.

In the event that a write to Timer1 coincides with a special event trigger from CCP1, the write will take precedence.

In this mode of operation, the CCPR1H:CCPR1L register pair effectively becomes the period register for Timer1.

## 12.6 Timer1 16-Bit Read/Write Mode

Timer1 can be configured for 16-bit reads and writes (see Figure 12-2). When the RD16 control bit (T1CON<7>) is set, the address for TMR1H is mapped to a buffer register for the high byte of Timer1. A read from TMR1L will load the contents of the high byte of Timer1 into the Timer1 high byte buffer. This provides the user with the ability to accurately read all 16 bits of Timer1 without having to determine whether a read of the high byte, followed by a read of the low byte, is valid, due to a rollover between reads.

A write to the high byte of Timer1 must also take place through the TMR1H Buffer register. Timer1 high byte is updated with the contents of TMR1H when a write occurs to TMR1L. This allows a user to write all 16 bits to both the high and low bytes of Timer1 at once.

The high byte of Timer1 is not directly readable or writable in this mode. All reads and writes must take place through the Timer1 High Byte Buffer register. Writes to TMR1H do not clear the Timer1 prescaler. The prescaler is only cleared on writes to TMR1L.

## 12.7 Using Timer1 as a Real-Time Clock

Adding an external LP oscillator to Timer1 (such as the one described in **Section 12.2 "Timer1 Oscillator"**, above), gives users the option to include RTC functionality to their applications. This is accomplished with an inexpensive watch crystal to provide an accurate time base and several lines of application code to calculate the time. When operating in Sleep mode and using a battery or supercapacitor as a power source, it can completely eliminate the need for a separate RTC device and battery backup.

The application code routine, RTCisr, shown in Example 12-1, demonstrates a simple method to increment a counter at one-second intervals using an Interrupt Service Routine. Incrementing the TMR1 register pair to overflow, triggers the interrupt and calls the routine, which increments the seconds counter by one; additional counters for minutes and hours are incremented as the previous counter overflow.

Since the register pair is 16 bits wide, counting up to overflow the register directly from a 32.768 kHz clock would take two seconds. To force the overflow at the required one-second intervals, it is necessary to preload it; the simplest method is to set the MSb of TMR1H with a BSF instruction. Note that the TMR1L register is never preloaded or altered; doing so may introduce cumulative error over many cycles.

For this method to be accurate, Timer1 must operate in Asynchronous mode and the Timer1 overflow interrupt must be enabled (PIE1<0> = 1), as shown in the routine, RTCinit. The Timer1 oscillator must also be enabled and running at all times.

**EXAMPLE 12-1: IMPLEMENTING A REAL-TIME CLOCK USING A TIMER1 INTERRUPT SERVICE**

```
RTCinit
            MOVLW   0x80                ; Preload TMR1 register pair
            MOVWF   TMR1H               ; for 1 second overflow
            CLRF    TMR1L
            MOVLW   b'00001111'         ; Configure for external clock,
            MOVWF   T1OSC               ; Asynchronous operation, external oscillator
            CLRF    secs                ; Initialize timekeeping registers
            CLRF    mins                ;
            MOVLW   .12
            MOVWF   hours
            BSF     PIE1, TMR1IE        ; Enable Timer1 interrupt
            RETURN
RTCisr
            BSF     TMR1H, 7            ; Preload for 1 sec overflow
            BCF     PIR1, TMR1IF        ; Clear interrupt flag
            INCF    secs, F             ; Increment seconds
            MOVLW   .59                 ; 60 seconds elapsed?
            CPFSGT  secs
            RETURN                      ; No, done
            CLRF    secs                ; Clear seconds
            INCF    mins, F             ; Increment minutes
            MOVLW   .59                 ; 60 minutes elapsed?
            CPFSGT  mins
            RETURN                      ; No, done
            CLRF    mins                ; clear minutes
            INCF    hours, F            ; Increment hours
            MOVLW   .23                 ; 24 hours elapsed?
            CPFSGT  hours
            RETURN                      ; No, done
            MOVLW   .01                 ; Reset hours to 1
            MOVWF   hours
            RETURN                      ; Done
```


© 2002-2015 Microchip Technology Inc.

**TABLE 12-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 0000 000u |
| PIR1 | — | ADIF | RCIF | TXIF | — | CCP1IF | TMR2IF | TMR1IF | -000 -000 | -000 -000 |
| PIE1 | — | ADIE | RCIE | TXIE | — | CCP1IE | TMR2IE | TMR1IE | -000 -000 | -000 -000 |
| IPR1 | — | ADIP | RCIP | TXIP | — | CCP1IP | TMR2IP | TMR1IP | -111 -111 | -111 -111 |
| TMR1L | Holding Register for the Least Significant Byte of the 16-bit TMR1 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| TMR1H | Holding Register for the Most Significant Byte of the 16-bit TMR1 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| T1CON | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON | 0000 0000 | u0uu uuuu |

**Legend:** x = unknown, u = unchanged, – = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.

## 13.0 TIMER2 MODULE

The Timer2 module timer has the following features:

- 8-bit timer (TMR2 register)
- 8-bit period register (PR2)
- Readable and writable (both registers)
- Software programmable prescaler (1:1, 1:4, 1:16)
- Software programmable postscaler (1:1 to 1:16)
- Interrupt on TMR2 match with PR2

Timer2 has a control register shown in Register 13-1. TMR2 can be shut off by clearing control bit, TMR2ON (T2CON<2>), to minimize power consumption. Figure 13-1 is a simplified block diagram of the Timer2 module. Register 13-1 shows the Timer2 Control register. The prescaler and postscaler selection of Timer2 are controlled by this register.

### 13.1 Timer2 Operation

Timer2 can be used as the PWM time base for the PWM mode of the CCP module. The TMR2 register is readable and writable and is cleared on any device Reset. The input clock (FOSC/4) has a prescale option of 1:1, 1:4 or 1:16, selected by control bits, T2CKPS1:T2CKPS0 (T2CON<1:0>). The match output of TMR2 goes through a 4-bit postscaler (which gives a 1:1 to 1:16 scaling inclusive) to generate a TMR2 interrupt (latched in flag bit, TMR2IF (PIR1<1>)).

The prescaler and postscaler counters are cleared when any of the following occurs:

- A write to the TMR2 register
- A write to the T2CON register
- Any device Reset (Power-on Reset, $\overline{\text{MCLR}}$ Reset, Watchdog Timer Reset or Brown-out Reset)

TMR2 is not cleared when T2CON is written.

**REGISTER 13-1: T2CON: TIMER2 CONTROL REGISTER**

| U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | TOUTPS3 | TOUTPS2 | TOUTPS1 | TOUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **Unimplemented:** Read as ‘0’

bit 6-3 **TOUTPS<3:0>:** Timer2 Output Postscale Select bits
0000 = 1:1 Postscale
0001 = 1:2 Postscale
•
•
•
1111 = 1:16 Postscale

bit 2 **TMR2ON:** Timer2 On bit
1 = Timer2 is on
0 = Timer2 is off

bit 1-0 **T2CKPS<1:0>:** Timer2 Clock Prescale Select bits
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16


© 2002-2015 Microchip Technology Inc.

## 13.2 Timer2 Interrupt

The Timer2 module has an 8-bit period register, PR2. Timer2 increments from 00h until it matches PR2 and then resets to 00h on the next increment cycle. PR2 is a readable and writable register. The PR2 register is initialized to FFh upon Reset.

## 13.3 Output of TMR2

The output of TMR2 (before the postscaler) is fed to the Synchronous Serial Port module, which optionally uses it to generate the shift clock.

**FIGURE 13-1: TIMER2 BLOCK DIAGRAM**

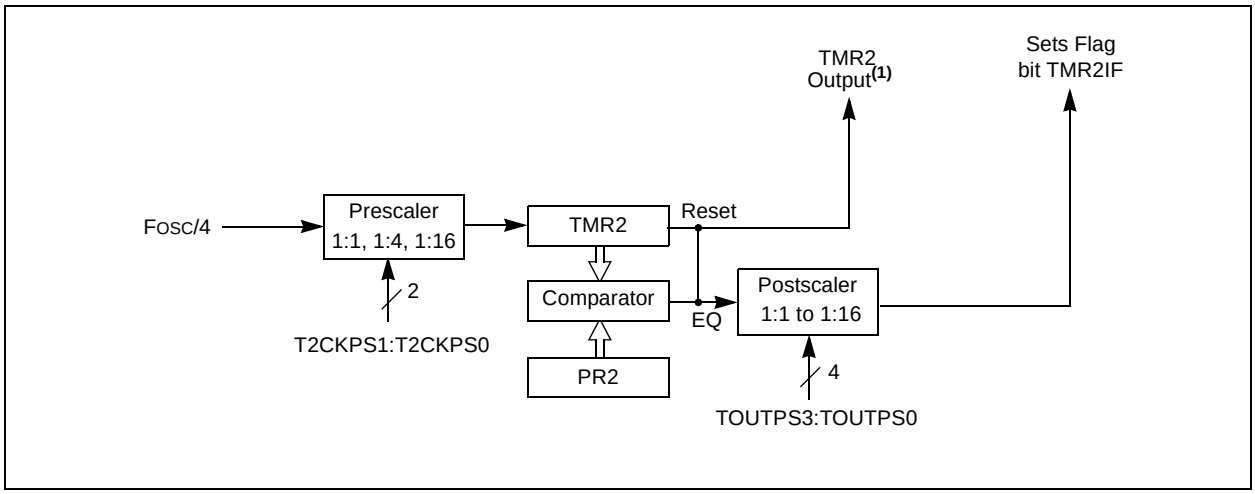



**TABLE 13-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 0000 000u |
| PIR1 | — | ADIF | RCIF | TXIF | — | CCP1IF | TMR2IF | TMR1IF | -000 -000 | -000 -000 |
| PIE1 | — | ADIE | RCIE | TXIE | — | CCP1IE | TMR2IE | TMR1IE | -000 -000 | -000 -000 |
| IPR1 | — | ADIP | RCIP | TXIP | — | CCP1IP | TMR2IP | TMR1IP | -111 -111 | -111 -111 |
| TMR2 | Timer2 Module Register | | | | | | | | 0000 0000 | 0000 0000 |
| T2CON | — | TOUTPS3 | TOUTPS2 | TOUTPS1 | TOUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | -000 0000 | -000 0000 |
| PR2 | Timer2 Period Register | | | | | | | | 1111 1111 | 1111 1111 |

**Legend:** x = unknown, u = unchanged, – = unimplemented, read as '0'. Shaded cells are not used by the Timer2 module.

## 14.0 TIMER3 MODULE

The Timer3 module timer/counter has the following features:

- 16-bit timer/counter
  (two 8-bit registers; TMR3H and TMR3L)
- Readable and writable (both registers)
- Internal or external clock select
- Interrupt-on-overflow from FFFFh to 0000h
- Reset from CCP module trigger

Figure 14-1 is a simplified block diagram of the Timer3 module.

Register 14-1 shows the Timer3 Control register. This register controls the operating mode of the Timer3 module and sets the CCP clock source.

Register 12-1 shows the Timer1 Control register. This register controls the operating mode of the Timer1 module, as well as contains the Timer1 Oscillator Enable bit (T1OSCEN), which can be a clock source for Timer3.

**REGISTER 14-1: T3CON: TIMER3 CONTROL REGISTER**

| R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| RD16 | — | T3CKPS1 | T3CKPS0 | T3CCP1 | $\overline{\text{T3SYNC}}$ | TMR3CS | TMR3ON |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **RD16:** 16-bit Read/Write Mode Enable bit
1 = Enables register read/write of Timer3 in one 16-bit operation
0 = Enables register read/write of Timer3 in two 8-bit operations

bit 6 **Unimplemented**: Read as ‘0’

bit 5-4 **T3CKPS<1:0>:** Timer3 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T3CCP1:** Timer3 and Timer1 to CCP1 Enable bits
1 = Timer3 is the clock source for compare/capture CCP module
0 = Timer1 is the clock source for compare/capture CCP module

bit 2 **$\overline{\text{T3SYNC}}$:** Timer3 External Clock Input Synchronization Control bit
(Not usable if the system clock comes from Timer1/Timer3.)
When TMR3CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR3CS = 0:
This bit is ignored. Timer3 uses the internal clock when TMR3CS = 0.

bit 1 **TMR3CS:** Timer3 Clock Source Select bit
1 = External clock input from Timer1 oscillator or T13CKI
(on the rising edge after the first falling edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR3ON:** Timer3 On bit
1 = Enables Timer3
0 = Stops Timer3


© 2002-2015 Microchip Technology Inc.

## 14.1 Timer3 Operation

Timer3 can operate in one of these modes:

- As a timer
- As a synchronous counter
- As an asynchronous counter

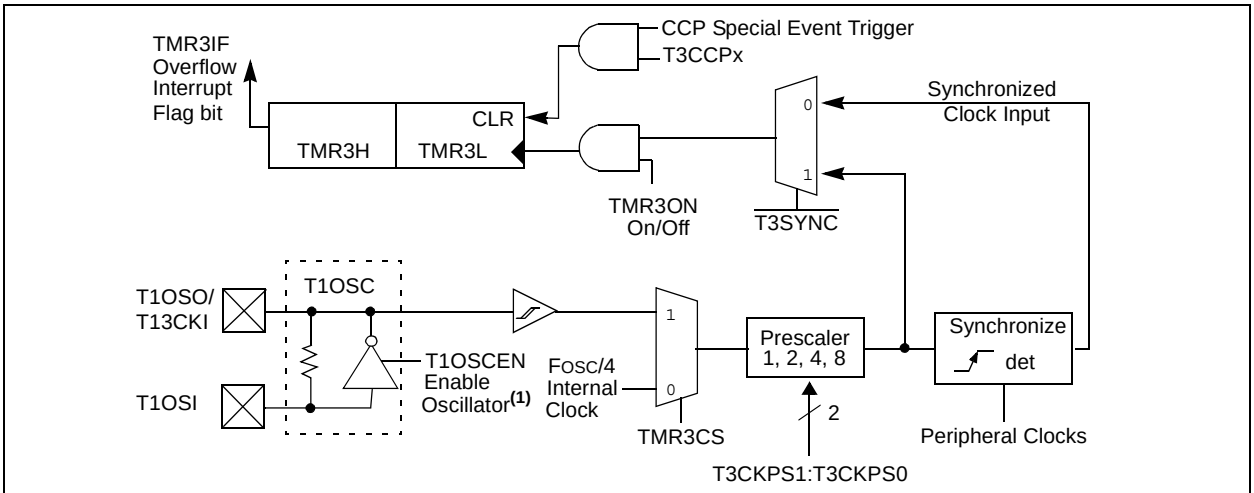
The operating mode is determined by the clock select bit, TMR3CS (T3CON<1>).

When TMR3CS = 0, Timer3 increments every instruction cycle. When TMR3CS = 1, Timer3 increments on every rising edge of the Timer1 external clock input or the Timer1 oscillator, if enabled.

When the Timer1 oscillator is enabled (T1OSCEN is set), the RB7/PGD/T1OSI/P1D/KBI3 and RB6/PGC/T1OSO/T13CKI/P1C/KBI2 pins become inputs. That is, the TRISB7:TRISB6 value is ignored and the pins are read as '0'.

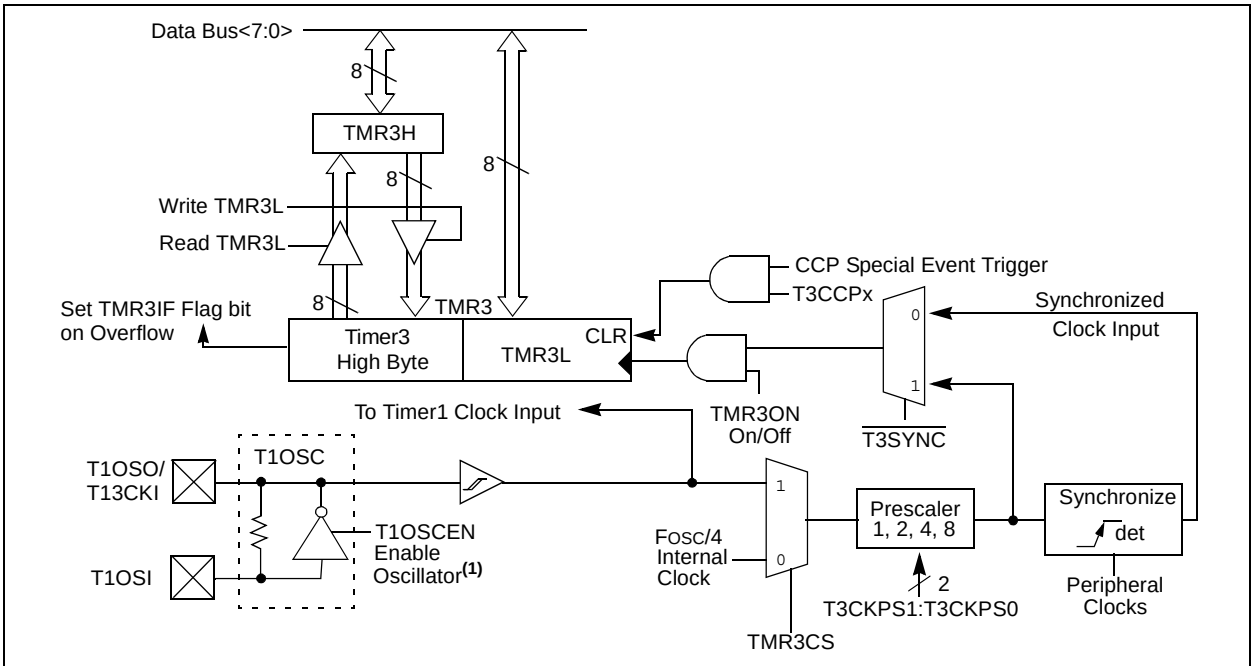
Timer3 also has an internal "Reset input". This Reset can be generated by the CCP module (see Section 15.4.4 "Special Event Trigger").

**FIGURE 14-1: TIMER3 BLOCK DIAGRAM**




**Note 1:** When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.

**FIGURE 14-2: TIMER3 BLOCK DIAGRAM CONFIGURED IN 16-BIT READ/WRITE MODE**




**Note 1:** When the T1OSCEN bit is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.

## 14.2 Timer1 Oscillator

The Timer1 oscillator may be used as the clock source for Timer3. The Timer1 oscillator is enabled by setting the T1OSCEN (T1CON<3>) bit. The oscillator is a low-power oscillator rated for 32 kHz crystals. See **Section 12.2 "Timer1 Oscillator"** for further details.

## 14.3 Timer3 Interrupt

The TMR3 register pair (TMR3H:TMR3L) increments from 0000h to FFFFh and rolls over to 0000h. The TMR3 interrupt, if enabled, is generated on overflow, which is latched in interrupt flag bit, TMR3IF (PIR2<1>). This interrupt can be enabled/disabled by setting/clearing TMR3 Interrupt Enable bit, TMR3IE (PIE2<1>).

## 14.4 Resetting Timer3 Using a CCP Trigger Output

If the CCP module is configured in Compare mode to generate a "special event trigger" (CCP1M3:CCP1M0 = 1011), this signal will reset Timer3. See **Section 15.4.4 "Special Event Trigger"** for more information.

| **Note:** | The special event triggers from the CCP module will not set interrupt flag bit, TMR3IF (PIR1<0>). |
|---|---|

Timer3 must be configured for either Timer or Synchronized Counter mode to take advantage of this feature. If Timer3 is running in Asynchronous Counter mode, this Reset operation may not work. In the event that a write to Timer3 coincides with a special event trigger from CCP1, the write will take precedence. In this mode of operation, the CCPR1H:CCPR1L register pair effectively becomes the period register for Timer3.

**TABLE 14-1: REGISTERS ASSOCIATED WITH TIMER3 AS A TIMER/COUNTER**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/ GIEH | PEIE/ GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 0000 000u |
| PIR2 | OSCFIF | — | — | EEIF | — | LVDIF | TMR3IF | — | 0--0 -00- | 0--0 -00- |
| PIE2 | OSCFIE | — | — | EEIE | — | LVDIE | TMR3IE | — | 0--0 -00- | 0--0 -00- |
| IPR2 | OSCFIP | — | — | EEIP | — | LVDIP | TMR3IP | — | 1--1 -11- | 1--1 -11- |
| TMR3L | Holding Register for the Least Significant Byte of the 16-bit TMR3 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| TMR3H | Holding Register for the Most Significant Byte of the 16-bit TMR3 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| T1CON | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON | 0000 0000 | u0uu uuuu |
| T3CON | RD16 | — | T3CKPS1 | T3CKPS0 | T3CCP1 | $\overline{\text{T3SYNC}}$ | TMR3CS | TMR3ON | 0-00 0000 | u-uu uuuu |

**Legend:** x = unknown, u = unchanged, – = unimplemented, read as '0'. Shaded cells are not used by the Timer3 module.


© 2002-2015 Microchip Technology Inc.

## 15.0 ENHANCED CAPTURE/ COMPARE/PWM (ECCP) MODULE

The Enhanced CCP module is implemented as a standard CCP module with Enhanced PWM capabilities. These capabilities allow for two or four output channels, user-selectable polarity, dead-band control and automatic shutdown and restart and are discussed in detail in **Section 15.5 "Enhanced PWM Mode"**.

The control register for CCP1 is shown in Register 15-1.

In addition to the expanded functions of the CCP1CON register, the ECCP module has two additional registers associated with Enhanced PWM operation and auto-shutdown features:

- PWM1CON
- ECCPAS

**REGISTER 15-1: CCP1CON REGISTER FOR ENHANCED CCP OPERATION**

| R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| P1M1 | P1M0 | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-6 **P1M<1:0>:** PWM Output Configuration bits

If CCP1M<3:2> = 00, 01, 10:

xx = P1A assigned as Capture/Compare input; P1B, P1C, P1D assigned as port pins

If CCP1M<3:2> = 11:

00 = Single output; P1A modulated; P1B, P1C, P1D assigned as port pins

01 = Full-bridge output forward; P1D modulated; P1A active; P1B, P1C inactive

10 = Half-bridge output; P1A, P1B modulated with dead-band control; P1C, P1D assigned as port pins

11 = Full-bridge output reverse; P1B modulated; P1C active; P1A, P1D inactive

bit 5-4 **DC1B<1:0>:** PWM Duty Cycle Least Significant bits

Capture mode:

Unused.

Compare mode:

Unused.

PWM mode:

These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPR1L.

bit 3-0 **CCP1M<3:0>:** ECCP1 Mode Select bits

0000 = Capture/Compare/PWM off (resets ECCP module)

0001 = Unused (reserved)

0010 = Compare mode, toggle output on match (ECCP1IF bit is set)

0011 = Unused (reserved)

0100 = Capture mode, every falling edge

0101 = Capture mode, every rising edge

0110 = Capture mode, every 4th rising edge

0111 = Capture mode, every 16th rising edge

1000 = Compare mode, set output on match (ECCP1IF bit is set)

1001 = Compare mode, clear output on match (ECCP1IF bit is set)

1010 = Compare mode, generate software interrupt on match (ECCP1IF bit is set, ECCP1 pin returns to port pin operation)

1011 = Compare mode, trigger special event (ECCP1IF bit is set; ECCP resets TMR1 or TMR3 and starts an A/D conversion if the A/D module is enabled)

1100 = PWM mode; P1A, P1C active-high; P1B, P1D active-high

1101 = PWM mode; P1A, P1C active-high; P1B, P1D active-low

1110 = PWM mode; P1A, P1C active-low; P1B, P1D active-high

1111 = PWM mode; P1A, P1C active-low; P1B, P1D active-low

© 2002-2015 Microchip Technology Inc.

## 15.1 ECCP Outputs

The Enhanced CCP module may have up to four outputs, depending on the selected operating mode. These outputs, designated P1A through P1D, are multiplexed with I/O pins on PORTB. The pin assignments are summarized in Table 15-1.

To configure I/O pins as PWM outputs, the proper PWM mode must be selected by setting the P1Mn and CCP1Mn bits (CCP1CON<7:6> and <3:0>, respectively). The appropriate TRISB direction bits for the port pins must also be set as outputs.

**TABLE 15-1: PIN ASSIGNMENTS FOR VARIOUS ECCP MODES**

| ECCP Mode | CCP1CON Configuration | RB3 | RB2 | RB6 | RB7 |
|---|---|---|---|---|---|
| Compatible CCP | 00xx 11xx | CCP1 | RB2/INT2 | RB6/PGC/T1OSO/T13CKI/KBI2 | RB7/PGD/T1OSI/KBI3 |
| Dual PWM | 10xx 11xx | P1A | P1B | RB6/PGC/T1OSO/T13CKI/KBI2 | RB7/PGD/T1OSI/KBI3 |
| Quad PWM | x1xx 11xx | P1A | P1B | P1C | P1D |

**Legend:** x = Don't care. Shaded cells indicate pin assignments not used by ECCP in a given mode.

**Note 1:** TRIS register values must be configured appropriately.

## 15.2 CCP Module

Capture/Compare/PWM Register 1 (CCPR1) is comprised of two 8-bit registers: CCPR1L (low byte) and CCPR1H (high byte). The CCP1CON register controls the operation of CCP1. All are readable and writable.

**TABLE 15-2: CCP MODE – TIMER RESOURCE**

| CCP Mode | Timer Resource |
|---|---|
| Capture | Timer1 or Timer3 |
| Compare | Timer1 or Timer3 |
| PWM | Timer2 |

## 15.3 Capture Mode

In Capture mode, CCPR1H:CCPR1L captures the 16-bit value of the TMR1 or TMR3 registers when an event occurs on pin RB3/CCP1/P1A. An event is defined as one of the following:

- every falling edge
- every rising edge
- every 4th rising edge
- every 16th rising edge

The event is selected by control bits, CCP1M3:CCP1M0 (CCP1CON<3:0>). When a capture is made, the interrupt request flag bit, CCP1IF (PIR1<2>), is set; it must be cleared in software. If another capture occurs before the value in register CCPR1 is read, the old captured value is overwritten by the new captured value.

### 15.3.1 CCP PIN CONFIGURATION

In Capture mode, the RB3/CCP1/P1A pin should be configured as an input by setting the TRISB<3> bit.

| **Note:** | If the RB3/CCP1/P1A is configured as an output, a write to the port can cause a capture condition. |
|---|---|

### 15.3.2 TIMER1/TIMER3 MODE SELECTION

The timers that are to be used with the capture feature (either Timer1 and/or Timer3) must be running in Timer mode or Synchronized Counter mode. In Asynchronous Counter mode, the capture operation may not work. The timer to be used with the CCP module is selected in the T3CON register.

### 15.3.3 SOFTWARE INTERRUPT

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep bit, CCP1IE (PIE1<2>), clear while changing capture modes to avoid false interrupts and should clear the flag bit, CCP1IF, following any such change in operating mode.


© 2002-2015 Microchip Technology Inc.

### 15.3.4 CCP PRESCALER

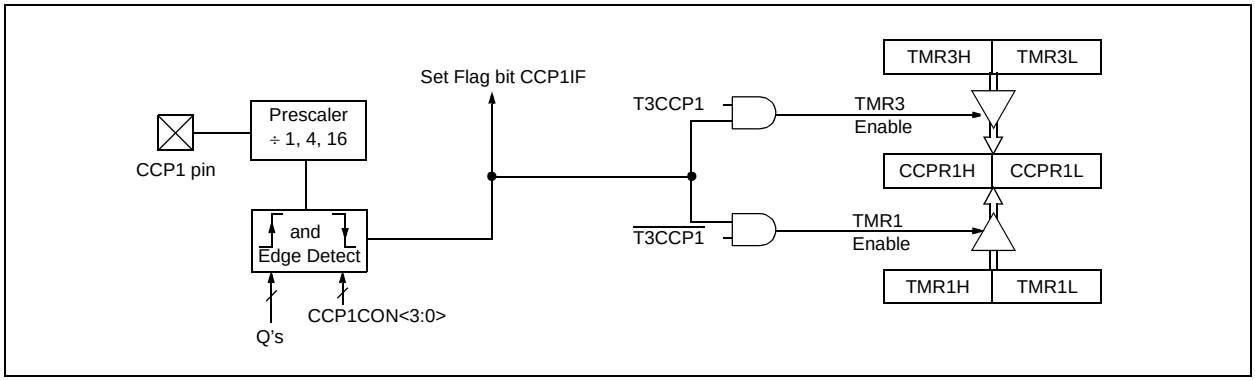
There are four prescaler settings, specified by bits CCP1M3:CCP1M0. Whenever the CCP module is turned off or the CCP module is not in Capture mode, the prescaler counter is cleared. This means that any Reset will clear the prescaler counter.

Switching from one capture prescaler to another may generate an interrupt. Also, the prescaler counter will not be cleared; therefore, the first capture may be from a non-zero prescaler. Example 15-1 shows the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the “false” interrupt.

**EXAMPLE 15-1: CHANGING BETWEEN CAPTURE PRESCALERS**

```
CLRF    CCP1CON         ; Turn CCP module off
MOVLW   NEW_CAPT_PS     ; Load WREG with the
                        ; new prescaler mode
                        ; value and CCP ON
MOVWF   CCP1CON         ; Load CCP1CON with
                        ; this value
```

**FIGURE 15-1: CAPTURE MODE OPERATION BLOCK DIAGRAM**




## 15.4 Compare Mode

In Compare mode, the 16-bit CCPR1 register value is constantly compared against either the TMR1 register pair value, or the TMR3 register pair value. When a match occurs, the RB3/CCP1/P1A pin:

- Is driven high
- Is driven low
- Toggles output (high-to-low or low-to-high)
- Remains unchanged (interrupt only)

The action on the pin is based on the value of control bits, CCP1M3:CCP1M0. At the same time, interrupt flag bit, CCP1IF, is set.

### 15.4.1 CCP PIN CONFIGURATION

The user must configure the RB3/CCP1/P1A pin as an output by clearing the TRISB<3> bit.

| **Note:** | Clearing the CCP1CON register will force the RB3/CCP1/P1A compare output latch to the default low level. This is not the PORTB I/O data latch. |
|---|---|

### 15.4.2 TIMER1/TIMER3 MODE SELECTION

Timer1 and/or Timer3 must be running in Timer mode or Synchronized Counter mode if the CCP module is using the compare feature. In Asynchronous Counter mode, the compare operation may not work.

### 15.4.3 SOFTWARE INTERRUPT MODE

When generate software interrupt is chosen, the RB3/CCP1/P1A pin is not affected. CCP1IF is set and an interrupt is generated (if enabled).

### 15.4.4 SPECIAL EVENT TRIGGER

In this mode, an internal hardware trigger is generated, which may be used to initiate an action.

The special event trigger output of CCP1 resets the TMR1 register pair. This allows the CCPR1 register to effectively be a 16-bit programmable period register for Timer1.

The special event trigger also sets the GO/$\overline{\text{DONE}}$ bit (ADCON0<1>). This starts a conversion of the currently selected A/D channel if the A/D is on.

**FIGURE 15-2: COMPARE MODE OPERATION BLOCK DIAGRAM**

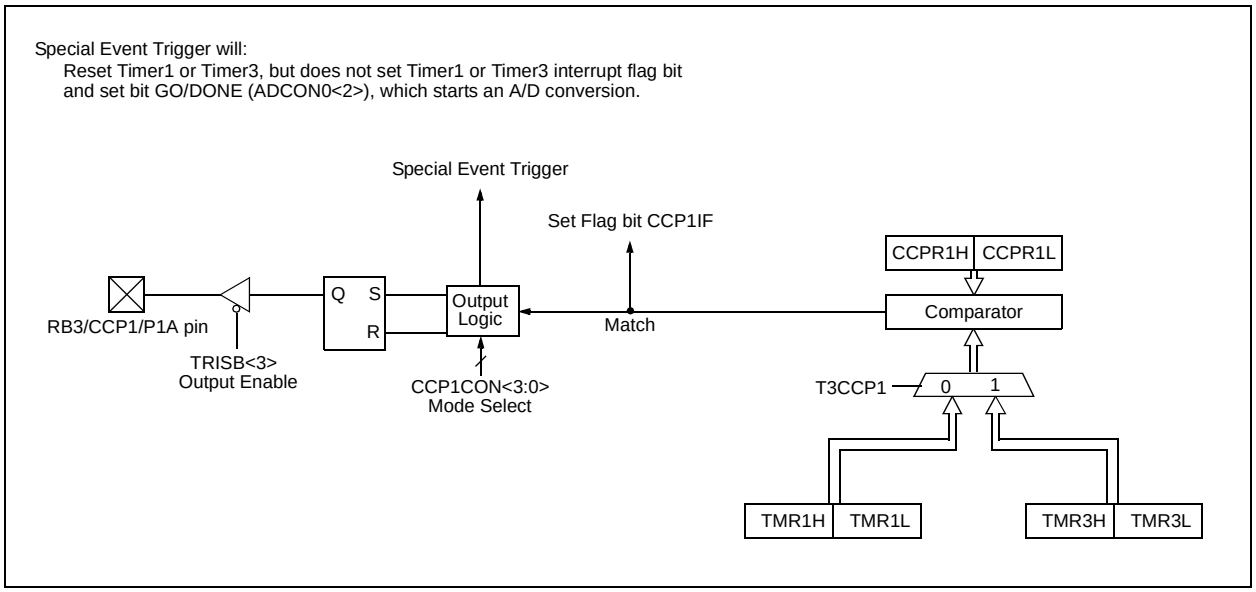



**TABLE 15-3: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE, TIMER1 AND TIMER3**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 0000 000u |
| PIR1 | — | ADIF | RCIF | TXIF | — | CCP1IF | TMR2IF | TMR1IF | -000 -000 | -000 -000 |
| PIE1 | — | ADIE | RCIE | TXIE | — | CCP1IE | TMR2IE | TMR1IE | -000 -000 | -000 -000 |
| IPR1 | — | ADIP | RCIP | TXIP | — | CCP1IP | TMR2IP | TMR1IP | -111 -111 | -111 -111 |
| TRISB | PORTB Data Direction Register | | | | | | | | 1111 1111 | 1111 1111 |
| TMR1L | Holding Register for the Least Significant Byte of the 16-bit TMR1 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| TMR1H | Holding Register for the Most Significant Byte of the 16-bit TMR1 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| T1CON | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON | 0000 0000 | uuuu uuuu |
| CCPR1L | Capture/Compare/PWM Register 1 (LSB) | | | | | | | | xxxx xxxx | uuuu uuuu |
| CCPR1H | Capture/Compare/PWM Register 1 (MSB) | | | | | | | | xxxx xxxx | uuuu uuuu |
| CCP1CON | P1M1 | P1M0 | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 | 0000 0000 | 0000 0000 |
| TMR3L | Holding Register for the Least Significant Byte of the 16-bit TMR3 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| TMR3H | Holding Register for the Most Significant Byte of the 16-bit TMR3 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| T3CON | RD16 | — | T3CKPS1 | T3CKPS0 | T3CCP1 | $\overline{\text{T3SYNC}}$ | TMR3CS | TMR3ON | 0-00 0000 | u-uu uuuu |
| ADCON0 | VCFG1 | VCFG0 | — | CHS2 | CHS1 | CHS0 | GO/$\overline{\text{DONE}}$ | ADON | 00-0 0000 | 00-0 0000 |

**Legend:** x = unknown, u = unchanged, – = unimplemented, read as '0'. Shaded cells are not used by Capture and Timer1.


© 2002-2015 Microchip Technology Inc.

## 15.5 Enhanced PWM Mode

The Enhanced PWM Mode provides additional PWM output options for a broader range of control applications. The module is an upwardly compatible version of the standard CCP module and offers up to four outputs, designated P1A through P1D. Users are also able to select the polarity of the signal (either active-high or active-low). The module's output mode and polarity are configured by setting the P1M1:P1M0 and CCP1M3CCP1M0 bits of the CCP1CON register (CCP1CON<7:6> and CCP1CON<3:0>, respectively).

Figure 15-3 shows a simplified block diagram of PWM operation. All control registers are double-buffered and are loaded at the beginning of a new PWM cycle (the period boundary when Timer2 resets) in order to prevent glitches on any of the outputs. The exception is the PWM Delay register, ECCP1DEL, which is loaded at either the duty cycle boundary or the boundary period (whichever comes first). Because of the buffering, the module waits until the assigned timer resets instead of starting immediately. This means that Enhanced PWM waveforms do not exactly match the standard PWM waveforms, but are instead offset by one full instruction cycle (4 TOSC).

As before, the user must manually configure the appropriate TRIS bits for output.

### 15.5.1 PWM PERIOD

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the equation:

**EQUATION 15-1: PWM PERIOD**

$$\text{PWM Period} = [(PR2) + 1] \cdot 4 \cdot T_{OSC} \cdot (\text{TMR2 Prescale Value})$$

PWM frequency is defined as 1/[PWM period]. When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCP1 pin is set (if PWM duty cycle = 0%, the CCP1 pin will not be set)
- The PWM duty cycle is copied from CCPR1L into CCPR1H

**Note:** The Timer2 postscaler (see **Section 13.0 "Timer2 Module"**) is not used in the determination of the PWM frequency. The postscaler could be used to have a servo update rate at a different frequency than the PWM output.

### 15.5.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the CCPR1L register and to the CCP1CON<5:4> bits. Up to 10-bit resolution is available. The CCPR1L contains the eight MSbs and the CCP1CON<5:4> contains the two LSbs. This 10-bit value is represented by CCPR1L:CCP1CON<5:4>. The PWM duty cycle is calculated by the equation:

**EQUATION 15-2: PWM DUTY CYCLE**

$$\text{PWM Duty Cycle} = (\text{CCPR1L:CCP1CON<5:4>}) \cdot T_{OSC} \cdot (\text{TMR2 Prescale Value})$$

CCPR1L and CCP1CON<5:4> can be written to at any time, but the duty cycle value is not copied into CCPR1H until a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCPR1H is a read-only register.

The CCPR1H register and a 2-bit internal latch are used to double-buffer the PWM duty cycle. This double-buffering is essential for glitchless PWM operation. When the CCPR1H and 2-bit latch match TMR2, concatenated with an internal 2-bit Q clock or two bits of the TMR2 prescaler, the CCP1 pin is cleared. The maximum PWM resolution (bits) for a given PWM frequency is given by the equation:

**EQUATION 15-3: PWM RESOLUTION**

$$\text{PWM Resolution (max)} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{ bits}$$

**Note:** If the PWM duty cycle value is longer than the PWM period, the CCP1 pin will not be cleared.

### 15.5.3 PWM OUTPUT CONFIGURATIONS

The P1M1:P1M0 bits in the CCP1CON register allow one of four configurations:

- Single Output
- Half-Bridge Output
- Full-Bridge Output, Forward mode
- Full-Bridge Output, Reverse mode

The Single Output mode is the Standard PWM mode discussed in **Section 15.5 "Enhanced PWM Mode"**. The Half-Bridge and Full-Bridge Output modes are covered in detail in the sections that follow.

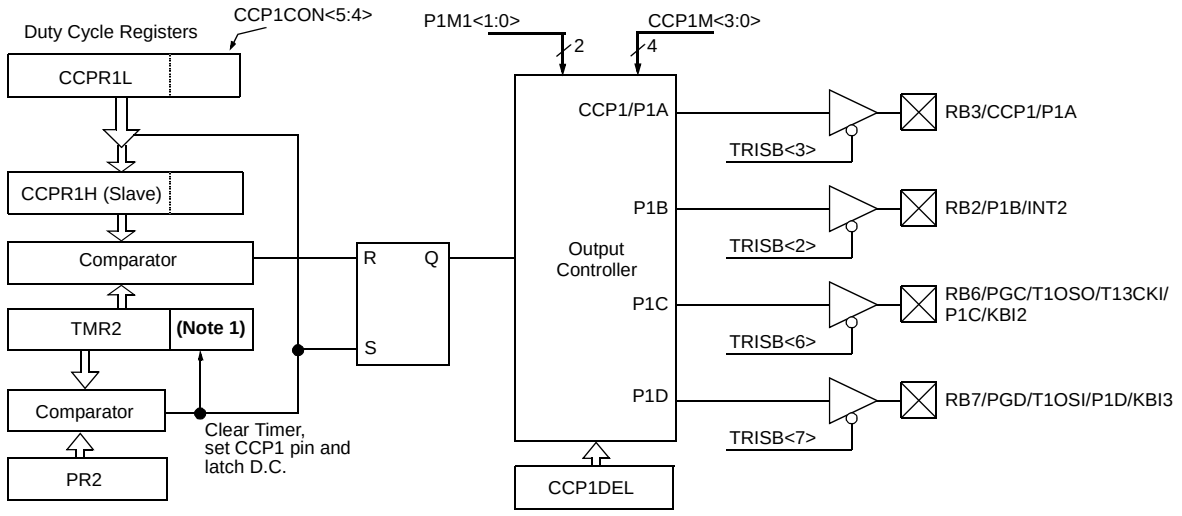
The general relationship of the outputs in all configurations is summarized in Figure 15-4.

**TABLE 15-4: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 40 MHz**

| PWM Frequency | 2.44 kHz | 9.77 kHz | 39.06 kHz | 156.25 kHz | 312.50 kHz | 416.67 kHz |
|---|---|---|---|---|---|---|
| Timer Prescaler (1, 4, 16) | 16 | 4 | 1 | 1 | 1 | 1 |
| PR2 Value | FFh | FFh | FFh | 3Fh | 1Fh | 17h |
| Maximum Resolution (bits) | 10 | 10 | 10 | 8 | 7 | 6.58 |

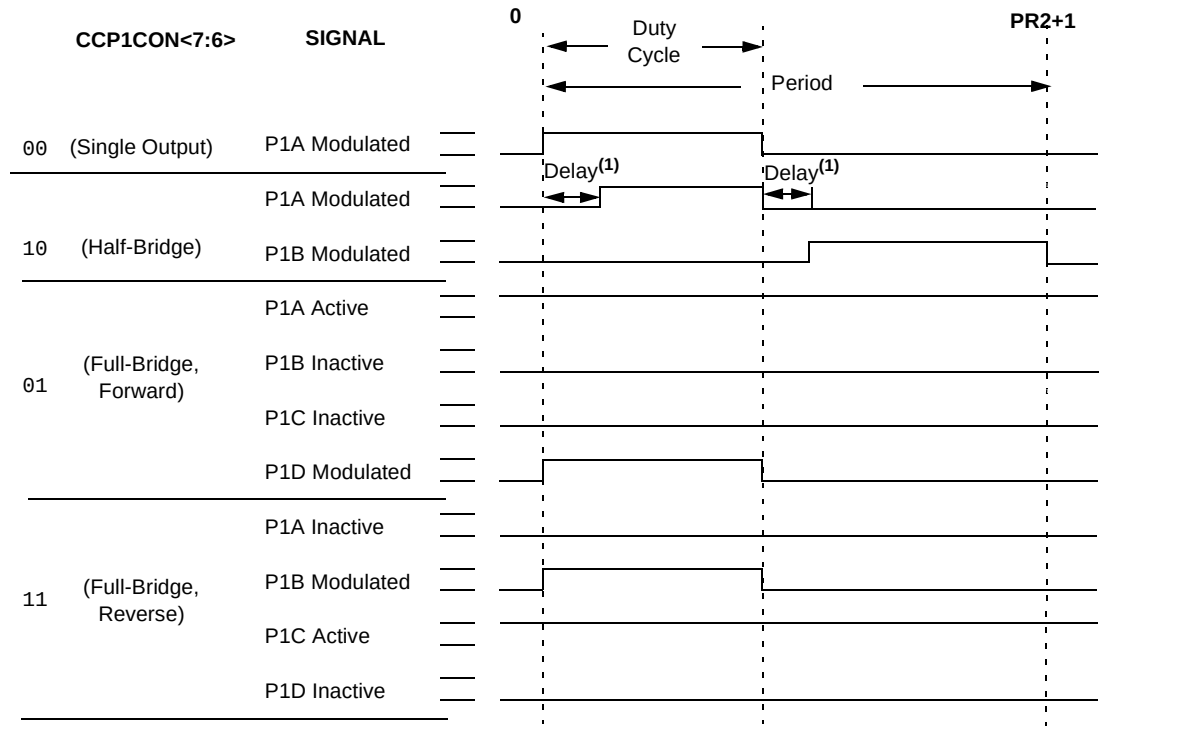
© 2002-2015 Microchip Technology Inc.

**FIGURE 15-3: SIMPLIFIED BLOCK DIAGRAM OF THE ENHANCED PWM MODULE**




**Note:** The 8-bit TMR2 register is concatenated with the 2-bit internal Q clock, or two bits of the prescaler to create the 10-bit time base.

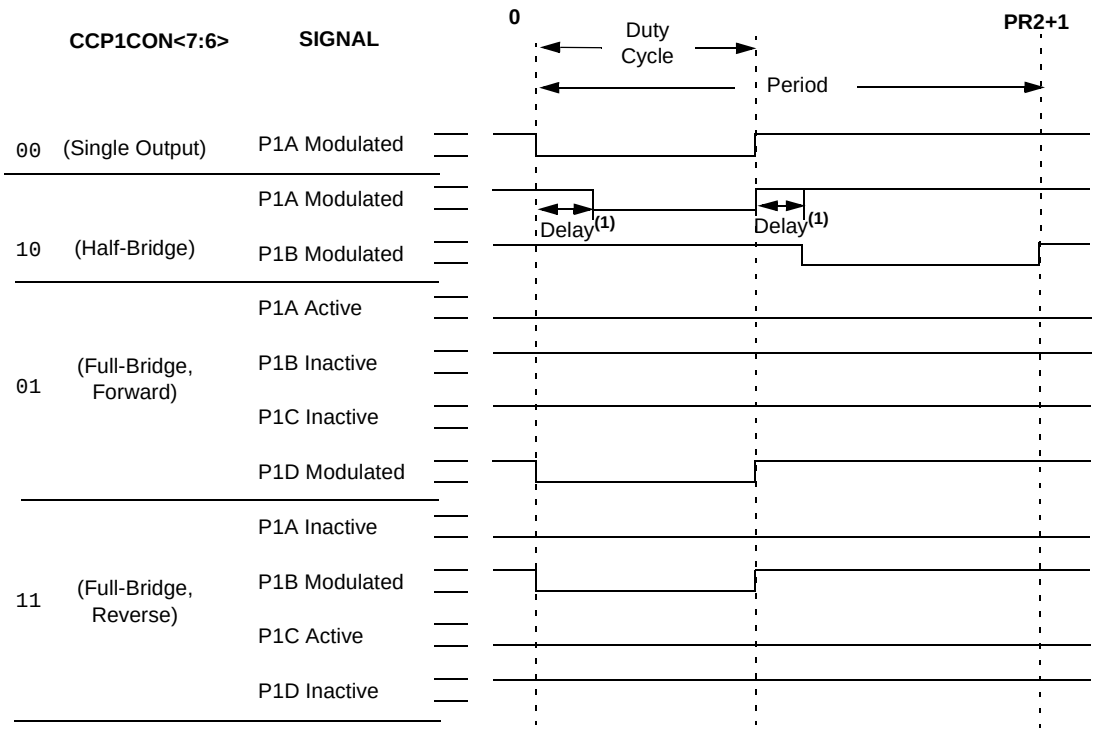
**FIGURE 15-4: PWM OUTPUT RELATIONSHIPS (ACTIVE-HIGH STATE)**





© 2002-2015 Microchip Technology Inc.

**FIGURE 15-5: PWM OUTPUT RELATIONSHIPS (ACTIVE-LOW STATE)**




**Relationships:**

- Period = 4 * TOSC * (PR2 + 1) * (TMR2 Prescale Value)
- Duty Cycle = TOSC * (CCPR1L<7:0>:CCP1CON<5:4>) * (TMR2 Prescale Value)
- Delay = 4 * TOSC * (PWM1CON<6:0>)

**Note 1:** Dead-band delay is programmed using the PWM1CON register (**Section 15.5.6 "Programmable Dead-Band Delay"**).

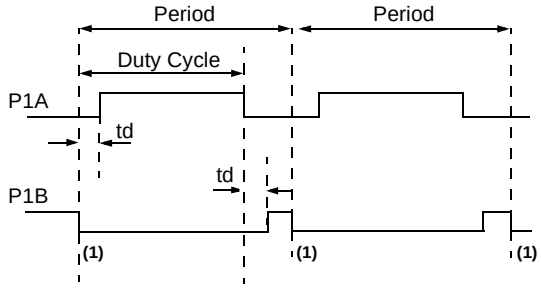
### 15.5.4 HALF-BRIDGE MODE

In the Half-Bridge Output mode, two pins are used as outputs to drive push-pull loads. The PWM output signal is output on the RB3/CCP1/P1A pin, while the complementary PWM output signal is output on the RB2/P1B/INT2 pin (Figure 15-6). This mode can be used for half-bridge applications, as shown in Figure 15-7, or for full-bridge applications, where four power switches are being modulated with two PWM signals.

In Half-Bridge Output mode, the programmable dead-band delay can be used to prevent shoot-through current in half-bridge power devices. The value of bits, PDC6:PDC0 (PWM1CON<6:0>), sets the number of instruction cycles before the output is driven active. If the value is greater than the duty cycle, the corresponding output remains inactive during the entire cycle. See **Section 15.5.6 "Programmable Dead-Band Delay"** for more details of the dead-band delay operations.

The TRISB<3> and TRISB<2> bits must be cleared to configure P1A and P1B as outputs.

**FIGURE 15-6: HALF-BRIDGE PWM OUTPUT (ACTIVE-HIGH)**

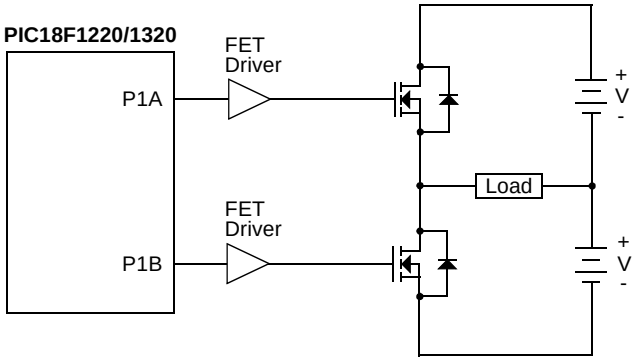



td = Dead-Band Delay

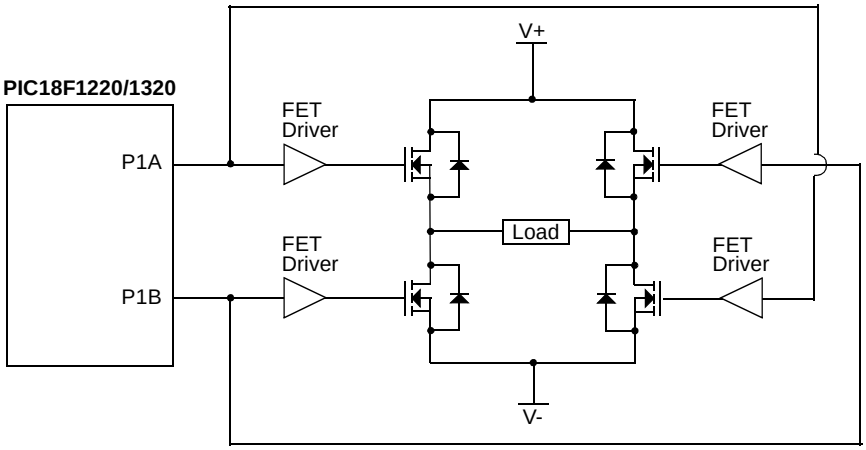
**Note 1:** At this time, the TMR2 register is equal to the PR2 register.

**FIGURE 15-7: EXAMPLES OF HALF-BRIDGE OUTPUT MODE APPLICATIONS**

**Standard Half-Bridge Circuit ("Push-Pull")**




**Half-Bridge Output Driving a Full-Bridge Circuit**





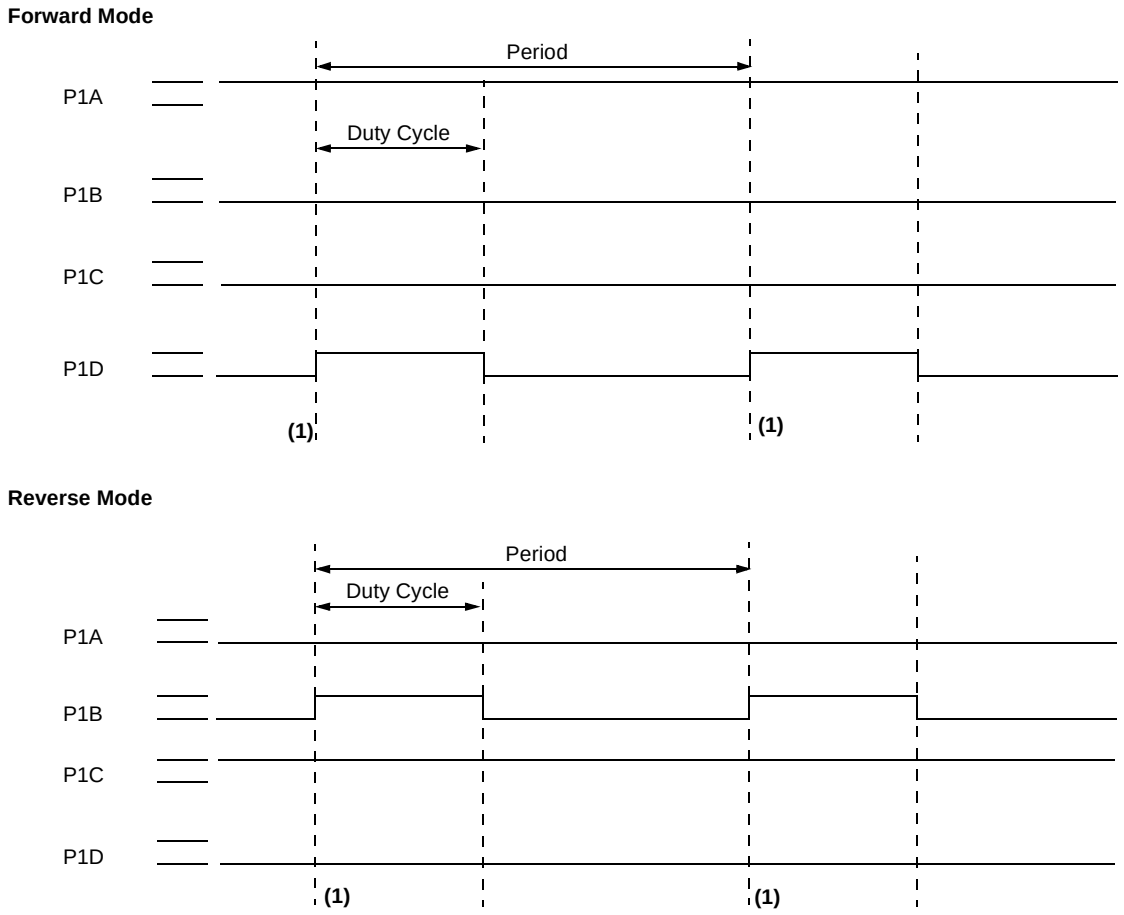
© 2002-2015 Microchip Technology Inc.

### 15.5.5 FULL-BRIDGE MODE

In Full-Bridge Output mode, four pins are used as outputs; however, only two outputs are active at a time. In the Forward mode, pin RB3/CCP1/P1A is continuously active and pin RB7/PGD/T1OSI/P1D/KBI3 is modulated. In the Reverse mode, pin RB6/PGC/T1OSO/T13CKI/P1C/KBI2 is continuously active and pin RB2/P1B/INT2 is modulated. These are illustrated in Figure 15-8.

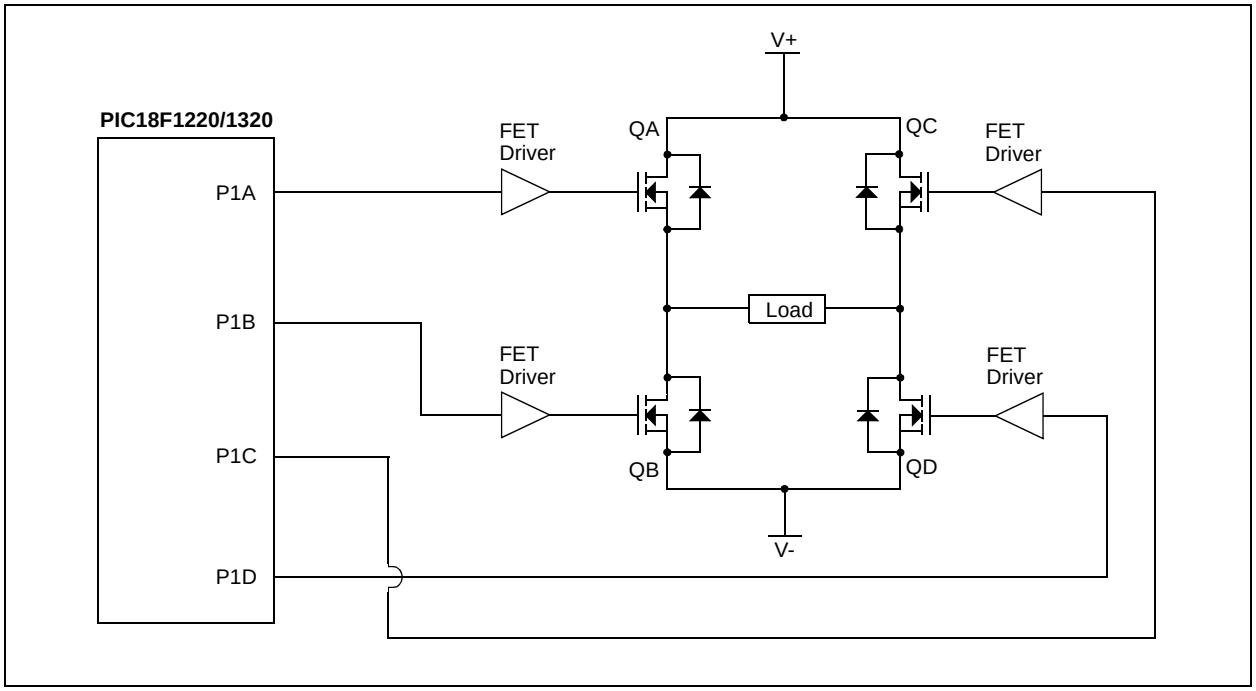
The TRISB<3:2> and TRISB<7:6> bits must be cleared to make the P1A, P1B, P1C and P1D pins output.

**FIGURE 15-8: FULL-BRIDGE PWM OUTPUT (ACTIVE-HIGH)**




**Note 1:** At this time, the TMR2 register is equal to the PR2 register.

**FIGURE 15-9: EXAMPLE OF FULL-BRIDGE APPLICATION**




### 15.5.5.1 Direction Change in Full-Bridge Mode

In the Full-Bridge Output mode, the P1M1 bit in the CCP1CON register allows the user to control the Forward/Reverse direction. When the application firmware changes this direction control bit, the module will assume the new direction on the next PWM cycle.

Just before the end of the current PWM period, the modulated outputs (P1B and P1D) are placed in their inactive state, while the unmodulated outputs (P1A and P1C) are switched to drive in the opposite direction. This occurs in a time interval of (4 TOSC * (Timer2 Prescale Value) before the next PWM period begins. The Timer2 prescaler will be either 1,4 or 16, depending on the value of the T2CKPS bit (T2CON<1:0>). During the interval from the switch of the unmodulated outputs to the beginning of the next period, the modulated outputs (P1B and P1D) remain inactive. This relationship is shown in Figure 15-10.

Note that in the Full-Bridge Output mode, the ECCP module does not provide any dead-band delay. In general, since only one output is modulated at all times, dead-band delay is not required. However, there is a situation where a dead-band delay might be required. This situation occurs when both of the following conditions are true:

1. The direction of the PWM output changes when the duty cycle of the output is at or near 100%.
2. The turn-off time of the power switch, including the power device and driver circuit, is greater than the turn-on time.

Figure 15-11 shows an example where the PWM direction changes from forward to reverse, at a near 100% duty cycle. At time t1, the output P1A and P1D become inactive, while output P1C becomes active. In this example, since the turn-off time of the power devices is longer than the turn-on time, a shoot-through current may flow through power devices QC and QD (see Figure 15-9) for the duration of ‘t’. The same phenomenon will occur to power devices QA and QB for PWM direction change from reverse to forward.

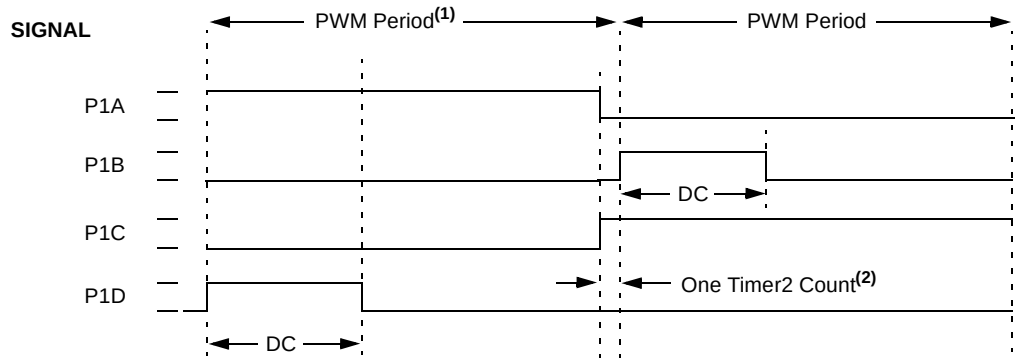
If changing PWM direction at high duty cycle is required for an application, one of the following requirements must be met:

1. Reduce PWM for a PWM period before changing directions.
2. Use switch drivers that can drive the switches off faster than they can drive them on.

Other options to prevent shoot-through current may exist.

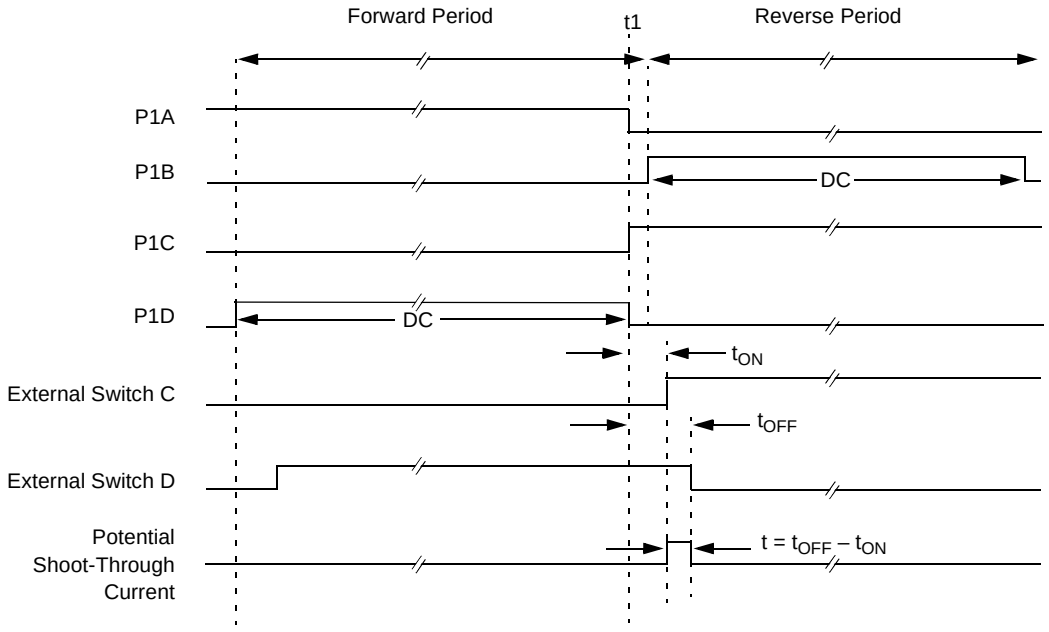

© 2002-2015 Microchip Technology Inc.

**FIGURE 15-10: PWM DIRECTION CHANGE (ACTIVE-HIGH)**




**Note 1:** The direction bit in the CCP1 Control register (CCP1CON<7>) is written any time during the PWM cycle.
**2:** When changing directions, the P1A and P1C toggle one Timer2 count before the end of the current PWM cycle. The modulated P1B and P1D signals are inactive at this time.

**FIGURE 15-11: PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE (ACTIVE-HIGH)**




**Note 1:** $t_{ON}$ is the turn-on delay of power switch QC and its driver.
**2:** $t_{OFF}$ is the turn-off delay of power switch QD and its driver.

### 15.5.6 PROGRAMMABLE DEAD-BAND DELAY

In half-bridge applications where all power switches are modulated at the PWM frequency at all times, the power switches normally require more time to turn off than to turn on. If both the upper and lower power switches are switched at the same time (one turned on and the other turned off), both switches may be on for a short period of time until one switch completely turns off. During this brief interval, a very high current (shoot-through current) may flow through both power switches, shorting the bridge supply. To avoid this potentially destructive shoot-through current from flowing during switching, turning on either of the power switches is normally delayed to allow the other switch to completely turn off.

In the Half-Bridge Output mode, a digitally programmable dead-band delay is available to avoid shoot-through current from destroying the bridge power switches. The delay occurs at the signal transition from the non-active state to the active state. See Figure 15-6 for an illustration. The lower seven bits of the PWM1CON register (Register 15-2) sets the delay period in terms of microcontroller instruction cycles (TCY or 4 TOSC).

### 15.5.7 ENHANCED PWM AUTO-SHUTDOWN

When the ECCP is programmed for any of the Enhanced PWM modes, the active output pins may be configured for auto-shutdown. Auto-shutdown immediately places the Enhanced PWM output pins into a defined shutdown state when a shutdown event occurs.

A shutdown event can be caused by the INT0, INT1 or INT2 pins (or any combination of these three sources). The auto-shutdown feature can be disabled by not selecting any auto-shutdown sources. The auto-shutdown sources to be used are selected using the ECCPAS2:ECCPAS0 bits (bits <6:4> of the ECCPAS register).

When a shutdown occurs, the output pins are asynchronously placed in their shutdown states, specified by the PSSAC1:PSSAC0 and PSSBD1:PSSBD0 bits (ECCPAS<3:0>). Each pin pair (P1A/P1C and P1B/P1D) may be set to drive high, drive low or be tri-stated (not driving). The ECCPASE bit (ECCPAS<7>) is also set to hold the Enhanced PWM outputs in their shutdown states.

The ECCPASE bit is set by hardware when a shutdown event occurs. If automatic restarts are not enabled, the ECCPASE bit is cleared by firmware when the cause of the shutdown clears. If automatic restarts are enabled, the ECCPASE bit is automatically cleared when the cause of the auto-shutdown has cleared.

If the ECCPASE bit is set when a PWM period begins, the PWM outputs remain in their shutdown state for that entire PWM period. When the ECCPASE bit is cleared, the PWM outputs will return to normal operation at the beginning of the next PWM period.

| **Note:** | Writing to the ECCPASE bit is disabled while a shutdown condition is active. |
|---|---|


© 2002-2015 Microchip Technology Inc.

**REGISTER 15-2: PWM1CON: PWM CONFIGURATION REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| PRSEN | PDC6 | PDC5 | PDC4 | PDC3 | PDC2 | PDC1 | PDC0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **PRSEN:** PWM Restart Enable bit

1 = Upon auto-shutdown, the ECCPASE bit clears automatically once the shutdown event goes away; the PWM restarts automatically

0 = Upon auto-shutdown, ECCPASE must be cleared in software to restart the PWM

bit 6-0 **PDC<6:0>:** PWM Delay Count bits

Number of FOSC/4 (4 * TOSC) cycles between the scheduled time when a PWM signal **should** transition active and the **actual** time it transitions active.

**REGISTER 15-3: ECCPAS: ENHANCED CAPTURE/COMPARE/PWM/AUTO-SHUTDOWN CONTROL REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| ECCPASE | ECCPAS2 | ECCPAS1 | ECCPAS0 | PSSAC1 | PSSAC0 | PSSBD1 | PSSBD0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **ECCPASE:** ECCP Auto-Shutdown Event Status bit
0 = ECCP outputs are operating
1 = A shutdown event has occurred; ECCP outputs are in shutdown state

bit 6 **ECCPAS2:** ECCP Auto-Shutdown bit 2
0 = INT0 pin has no effect
1 = INT0 pin low causes shutdown

bit 5 **ECCPAS1:** ECCP Auto-Shutdown bit 1
0 = INT2 pin has no effect
1 = INT2 pin low causes shutdown

bit 4 **ECCPAS0:** ECCP Auto-Shutdown bit 0
0 = INT1 pin has no effect
1 = INT1 pin low causes shutdown

bit 3-2 **PSSACn:** Pins A and C Shutdown State Control bits
00 = Drive Pins A and C to ‘0’
01 = Drive Pins A and C to ‘1’
1x = Pins A and C tri-state

bit 1-0 **PSSBDn:** Pins B and D Shutdown State Control bits
00 = Drive Pins B and D to ‘0’
01 = Drive Pins B and D to ‘1’
1x = Pins B and D tri-state


© 2002-2015 Microchip Technology Inc.

#### 15.5.7.1 Auto-Shutdown and Automatic Restart

The auto-shutdown feature can be configured to allow automatic restarts of the module, following a shutdown event. This is enabled by setting the PRSEN bit of the PWM1CON register (PWM1CON<7>).

In Shutdown mode with PRSEN = 1 (Figure 15-12), the ECCPASE bit will remain set for as long as the cause of the shutdown continues. When the shutdown condition clears, the ECCPASE bit is automatically cleared. If PRSEN = 0 (Figure 15-13), once a shutdown condition occurs, the ECCPASE bit will remain set until it is cleared by firmware. Once ECCPASE is cleared, the Enhanced PWM will resume at the beginning of the next PWM period.

| **Note:** | Writing to the ECCPASE bit is disabled while a shutdown condition is active. |
|---|---|

Independent of the PRSEN bit setting, the ECCPASE bit cannot be cleared as long as the cause of the shutdown persists.

The Auto-Shutdown mode can be forced by writing a '1' to the ECCPASE bit.

### 15.5.8 START-UP CONSIDERATIONS

When the ECCP module is used in the PWM mode, the application hardware must use the proper external pull-up and/or pull-down resistors on the PWM output pins. When the microcontroller is released from Reset, all of the I/O pins are in the high-impedance state. The external circuits must keep the power switch devices in the off state, until the microcontroller drives the I/O pins with the proper signal levels, or activates the PWM output(s).

The CCP1M1:CCP1M0 bits (CCP1CON<1:0>) allow the user to choose whether the PWM output signals are active-high or active-low for each pair of PWM output pins (P1A/P1C and P1B/P1D). The PWM output polarities must be selected before the PWM pins are configured as outputs. Changing the polarity configuration while the PWM pins are configured as outputs is not recommended, since it may result in damage to the application circuits.

The P1A, P1B, P1C and P1D output latches may not be in the proper states when the PWM module is initialized. Enabling the PWM pins for output at the same time as the ECCP module may cause damage to the application circuit. The ECCP module must be enabled in the proper output mode and complete a full PWM cycle, before configuring the PWM pins as outputs. The completion of a full PWM cycle is indicated by the TMR2IF bit being set as the second PWM period begins.

**FIGURE 15-12: PWM AUTO-SHUTDOWN (PRSEN = 1, AUTO-RESTART ENABLED)**

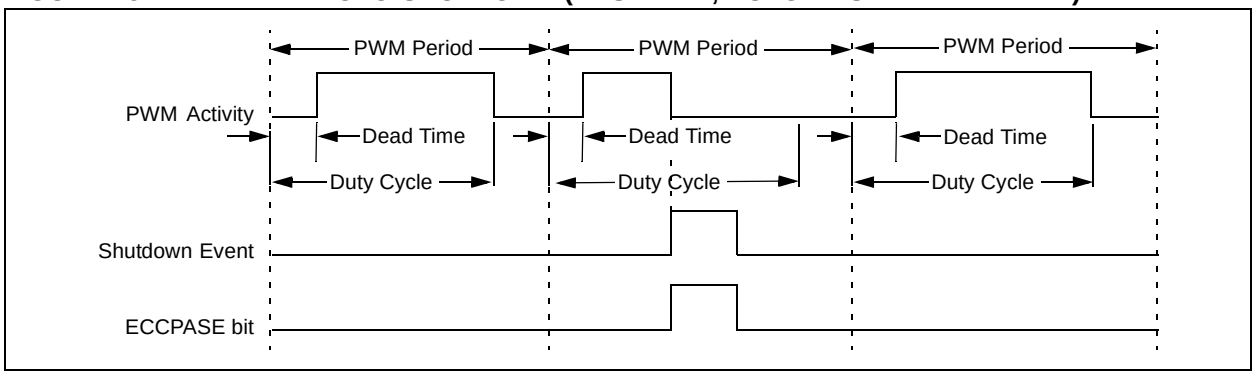
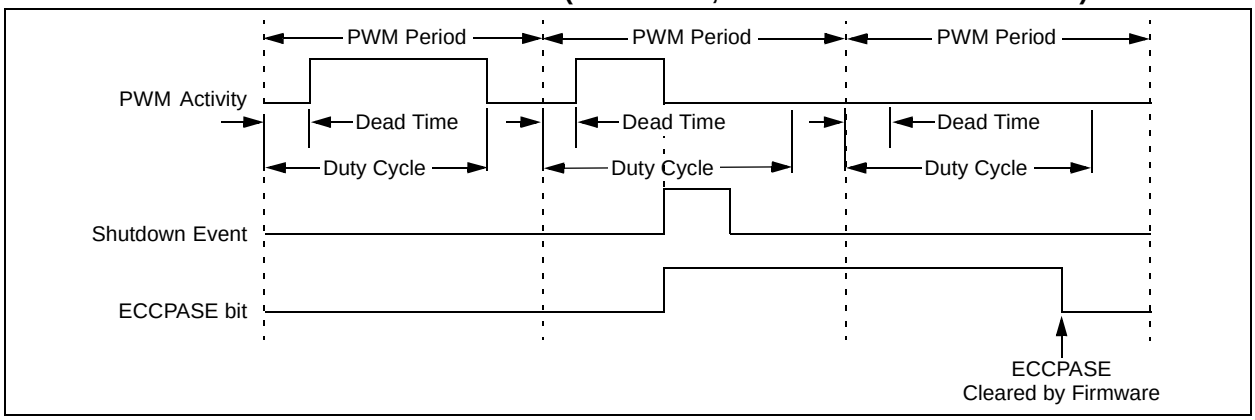



**FIGURE 15-13: PWM AUTO-SHUTDOWN (PRSEN = 0, AUTO-RESTART DISABLED)**



### 15.5.9 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the ECCP1 module for PWM operation:

1. Configure the PWM pins P1A and P1B (and P1C and P1D, if used) as inputs by setting the corresponding TRISB bits.
2. Set the PWM period by loading the PR2 register.
3. Configure the ECCP module for the desired PWM mode and configuration by loading the CCP1CON register with the appropriate values:
   - Select one of the available output configurations and direction with the P1M1:P1M0 bits.
   - Select the polarities of the PWM output signals with the CCP1M3:CCP1M0 bits.
4. Set the PWM duty cycle by loading the CCPR1L register and CCP1CON<5:4> bits.
5. For Half-Bridge Output mode, set the dead-band delay by loading PWM1CON<6:0> with the appropriate value.
6. If auto-shutdown operation is required, load the ECCPAS register:
   - Select the auto-shutdown sources using the ECCPAS<2:0> bits.
   - Select the shutdown states of the PWM output pins using PSSAC1:PSSAC0 and PSSBD1:PSSBD0 bits.
   - Set the ECCPASE bit (ECCPAS<7>).
7. If auto-restart operation is required, set the PRSEN bit (PWM1CON<7>).
8. Configure and start TMR2:
   - Clear the TMR2 interrupt flag bit by clearing the TMR2IF bit (PIR1<1>).
   - Set the TMR2 prescale value by loading the T2CKPS bits (T2CON<1:0>).
   - Enable Timer2 by setting the TMR2ON bit (T2CON<2>).
9. Enable PWM outputs after a new PWM cycle has started:
   - Wait until TMR2 overflows (TMR2IF bit is set).
   - Enable the CCP1/P1A, P1B, P1C and/or P1D pin outputs by clearing the respective TRISB bits.
   - Clear the ECCPASE bit (ECCPAS<7>).

### 15.5.10 OPERATION IN LOW-POWER MODES

In the Low-Power Sleep mode, all clock sources are disabled. Timer2 will not increment and the state of the module will not change. If the ECCP pin is driving a value, it will continue to drive that value. When the device wakes up, it will continue from this state. If Two-Speed Start-ups are enabled, the initial start-up frequency may not be stable if the INTOSC is being used.

In PRI_IDLE mode, the primary clock will continue to clock the ECCP module without change.

In all other low-power modes, the selected low-power mode clock will clock Timer2. Other low-power mode clocks will most likely be different than the primary clock frequency.

#### 15.5.10.1 Operation with Fail-Safe Clock Monitor

If the Fail-Safe Clock Monitor is enabled (CONFIG1H<6> is programmed), a clock failure will force the device into the Low-Power RC_RUN mode and the OSCFIF bit (PIR2<7>) will be set. The ECCP will then be clocked from the INTRC clock source, which may have a different clock frequency than the primary clock. By loading the IRCF2:IRCF0 bits on Resets, the user can enable the INTOSC at a high clock speed in the event of a clock failure.

See the previous section for additional details.

### 15.5.11 EFFECTS OF A RESET

Both power-on and subsequent Resets will force all ports to input mode and the CCP registers to their Reset states.

This forces the Enhanced CCP module to reset to a state compatible with the standard CCP module.


© 2002-2015 Microchip Technology Inc.

**TABLE 15-5: REGISTERS ASSOCIATED WITH ENHANCED PWM AND TIMER2**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 0000 000u |
| RCON | IPEN | — | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ | 0--1 11qq | 0--q qquu |
| PIR1 | — | ADIF | RCIF | TXIF | — | CCP1IF | TMR2IF | TMR1IF | -000 -000 | -000 -000 |
| PIE1 | — | ADIE | RCIE | TXIE | — | CCP1IE | TMR2IE | TMR1IE | -000 -000 | -000 -000 |
| IPR1 | — | ADIP | RCIP | TXIP | — | CCP1IP | TMR2IP | TMR1IP | -111 -111 | -111 -111 |
| TMR2 | Timer2 Module Register | | | | | | | | 0000 0000 | 0000 0000 |
| PR2 | Timer2 Module Period Register | | | | | | | | 1111 1111 | 1111 1111 |
| T2CON | — | TOUTPS3 | TOUTPS2 | TOUTPS1 | TOUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | -000 0000 | -000 0000 |
| TRISB | PORTB Data Direction Register | | | | | | | | 1111 1111 | 1111 1111 |
| CCPR1H | Enhanced Capture/Compare/PWM Register 1 High Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| CCPR1L | Enhanced Capture/Compare/PWM Register 1 Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| CCP1CON | P1M1 | P1M0 | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 | 0000 0000 | 0000 0000 |
| ECCPAS | ECCPASE | ECCPAS2 | ECCPAS1 | ECCPAS0 | PSSAC1 | PSSAC0 | PSSBD1 | PSSBD0 | 0000 0000 | 0000 0000 |
| PWM1CON | PRSEN | PDC6 | PDC5 | PDC4 | PDC3 | PDC2 | PDC1 | PDC0 | 0000 0000 | uuuu uuuu |
| OSCCON | IDLEN | IRCF2 | IRCF1 | IRCF0 | OSTS | IOFS | SCS1 | SCS0 | 0000 qq00 | 0000 qq00 |

**Legend:** x = unknown, u = unchanged, – = unimplemented, read as '0'.
Shaded cells are not used by the ECCP module in Enhanced PWM mode.

## 16.0 ENHANCED ADDRESSABLE UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (EUSART)

The Enhanced Addressable Universal Synchronous Asynchronous Receiver Transmitter (EUSART) module can be configured as a full-duplex asynchronous system that can communicate with peripheral devices, such as CRT terminals and personal computers. It can also be configured as a half-duplex synchronous system that can communicate with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs, etc.

The Enhanced Addressable USART module implements additional features, including automatic baud rate detection and calibration, automatic wake-up on Sync Break reception and 12-bit Break character transmit. These features make it ideally suited for use in Local Interconnect Network (LIN) bus systems.

The EUSART can be configured in the following modes:

- Asynchronous (full duplex) with:
  - Auto-wake-up on character reception
  - Auto-baud calibration
  - 12-bit Break character transmission
- Synchronous – Master (half duplex) with selectable clock polarity
- Synchronous – Slave (half duplex) with selectable clock polarity

The RB1/AN5/TX/CK/INT1 and RB4/AN6/RX/DT/KBI0 pins must be configured as follows for use with the Universal Synchronous Asynchronous Receiver Transmitter:

- SPEN (RCSTA<7>) bit must be set ( = 1),
- PCFG6:PCFG5 (ADCON1<5:6>) must be set ( = 1),
- TRISB<4> bit must be set ( = 1) and
- TRISB<1> bit must be set ( = 1).

| | |
|---|---|
| **Note:** | The EUSART control will automatically reconfigure the pin from input to output as needed. |

The operation of the Enhanced USART module is controlled through three registers:

- Transmit Status and Control (TXSTA)
- Receive Status and Control (RCSTA)
- Baud Rate Control (BAUDCTL)

These are detailed in on the following pages in Register 16-1, Register 16-2 and Register 16-3, respectively.

### 16.1 Asynchronous Operation in Power Managed Modes

The EUSART may operate in Asynchronous mode while the peripheral clocks are being provided by the internal oscillator block. This makes it possible to remove the crystal or resonator that is commonly connected as the primary clock on the OSC1 and OSC2 pins.

The factory calibrates the internal oscillator block output (INTOSC) for 8 MHz (see Table 22-6). However, this frequency may drift as $V_{DD}$ or temperature changes and this directly affects the asynchronous baud rate. Two methods may be used to adjust the baud rate clock, but both require a reference clock source of some kind.

The first (preferred) method uses the OSCTUNE register to adjust the INTOSC output back to 8 MHz. Adjusting the value in the OSCTUNE register allows for fine resolution changes to the system clock source (see **Section 3.6 "INTOSC Frequency Drift"** for more information).

The other method adjusts the value in the Baud Rate Generator (BRG). There may not be fine enough resolution when adjusting the Baud Rate Generator to compensate for a gradual change in the peripheral clock frequency.


© 2002-2015 Microchip Technology Inc.

**REGISTER 16-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R-1/1 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| CSRC | TX9 | TXEN(1) | SYNC | SENDB | BRGH | TRMT | TX9D |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7 **CSRC:** Clock Source Select bit

Asynchronous mode:
Don't care.

Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9:** 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN:** Transmit Enable bit(1)
1 = Transmit enabled
0 = Transmit disabled

bit 4 **SYNC:** EUSART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **SENDB:** Send Break Character bit

Asynchronous mode:
1 = Send Sync Break on next transmission (cleared by hardware upon completion)
0 = Sync Break transmission completed

Synchronous mode:
Don't care.

bit 2 **BRGH:** High Baud Rate Select bit

Asynchronous mode:
1 = High speed
0 = Low speed

Synchronous mode:
Unused in this mode.

bit 1 **TRMT:** Transmit Shift Register Status bit
1 = TSR Idle
0 = TSR busy

bit 0 **TX9D:** 9th bit of Transmit Data
Can be address/data bit or a parity bit.

**Note 1:** SREN/CREN overrides TXEN in Sync mode.

**REGISTER 16-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R-0/0 | R-0/0 | R-x |
|---|---|---|---|---|---|---|---|
| SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled (held in Reset)

bit 6 **RX9:** 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN:** Single Receive Enable bit
Asynchronous mode:
Don't care.
Synchronous mode – Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode – Slave:
Don't care.

bit 4 **CREN:** Continuous Receive Enable bit
Asynchronous mode:
1 = Enables receiver
0 = Disables receiver
Synchronous mode:
1 = Enables continuous receive until enable bit, CREN, is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, generates RCIF interrupt and loads RCREG when RX9D is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
Asynchronous mode 8-bit (RX9 = 0):
Don't care.

bit 2 **FERR:** Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receiving next valid byte)
0 = No framing error

bit 1 **OERR:** Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** 9th bit of Received Data
This can be address/data bit or a parity bit and must be calculated by user firmware.


© 2002-2015 Microchip Technology Inc.

**REGISTER 16-3: BAUDCTL: BAUD RATE CONTROL REGISTER**

| U-0 | R-1 | U-0 | R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

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

bit 6 **RCIDL**: Receive Operation Idle Status bit
1 = Receiver is Idle
0 = Receiver is busy

bit 5 **Unimplemented:** Read as '0'

bit 4 **SCKP**: Synchronous Clock Polarity Select bit
Asynchronous mode:
Unused in this mode.
Synchronous mode:
1 = Idle state for clock (CK) is a high level
0 = Idle state for clock (CK) is a low level

bit 3 **BRG16:** 16-bit Baud Rate Register Enable bit
1 = 16-bit Baud Rate Generator – SPBRGH and SPBRG
0 = 8-bit Baud Rate Generator – SPBRG only (Compatible mode), SPBRGH value ignored

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

bit 1 **WUE:** Wake-up Enable bit
Asynchronous mode:
1 = EUSART will continue to sample the RX pin – interrupt generated on falling edge; bit cleared in hardware on following rising edge
0 = RX pin not monitored or rising edge detected
Synchronous mode:
Unused in this mode.

bit 0 **ABDEN**: Auto-Baud Detect Enable bit
Asynchronous mode:
1 = Enable baud rate measurement on the next character – requires reception of a Sync byte (55h); cleared in hardware upon completion
0 = Baud rate measurement disabled or completed
Synchronous mode:
Unused in this mode.

## 16.2 EUSART Baud Rate Generator (BRG)

The BRG is a dedicated 8-bit or 16-bit generator, that supports both the Asynchronous and Synchronous modes of the EUSART. By default, the BRG operates in 8-bit mode; setting the BRG16 bit (BAUDCTL<3>) selects 16-bit mode.

The SPBRGH:SPBRG register pair controls the period of a free running timer. In Asynchronous mode, bits BRGH (TXSTA<2>) and BRG16 also control the baud rate. In Synchronous mode, bit BRGH is ignored. Table 16-1 shows the formula for computation of the baud rate for different EUSART modes which only apply in Master mode (internally generated clock).

Given the desired baud rate and FOSC, the nearest integer value for the SPBRGH:SPBRG registers can be calculated using the formulas in Table 16-1. From this, the error in baud rate can be determined. An example calculation is shown in Example 16-1. Typical baud rates and error values for the various asynchronous modes are shown in Table 16-2. It may be advantageous to use the high baud rate (BRGH = 1), or the 16-bit BRG to reduce the baud rate error, or achieve a slow baud rate for a fast oscillator frequency.

Writing a new value to the SPBRGH:SPBRG registers causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

### 16.2.1 POWER MANAGED MODE OPERATION

The system clock is used to generate the desired baud rate; however, when a power managed mode is entered, the clock source may be operating at a different frequency than in PRI_RUN mode. In Sleep mode, no clocks are present and in PRI_IDLE mode, the primary clock source continues to provide clocks to the Baud Rate Generator; however, in other power managed modes, the clock frequency will probably change. This may require the value in SPBRG to be adjusted.

If the system clock is changed during an active receive operation, a receive error or data loss may result. To avoid this problem, check the status of the RCIDL bit and make sure that the receive operation is Idle before changing the system clock.

### 16.2.2 SAMPLING

The data on the RB4/AN6/RX/DT/KBI0 pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RX pin.

**TABLE 16-1: BAUD RATE FORMULAS**

| Configuration Bits | | | BRG/EUSART Mode | Baud Rate Formula |
|---|---|---|---|---|
| SYNC | BRG16 | BRGH | | |
| 0 | 0 | 0 | 8-bit/Asynchronous | FOSC/[64 (n + 1)] |
| 0 | 0 | 1 | 8-bit/Asynchronous | FOSC/[16 (n + 1)] |
| 0 | 1 | 0 | 16-bit/Asynchronous | |
| 0 | 1 | 1 | 16-bit/Asynchronous | FOSC/[4 (n + 1)] |
| 1 | 0 | x | 8-bit/Synchronous | |
| 1 | 1 | x | 16-bit/Synchronous | |

**Legend:** x = Don't care, n = value of SPBRGH:SPBRG register pair

**EXAMPLE 16-1: CALCULATING BAUD RATE ERROR**

For a device with FOSC of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:
Desired Baud Rate= FOSC/(64 ([SPBRGH:SPBRG] + 1))
Solving for SPBRGH:SPBRG:
X = ((FOSC/Desired Baud Rate)/64) – 1
= ((16000000/9600)/64) – 1
= [25.042] = 25
Calculated Baud Rate=16000000/(64 (25 + 1))
= 9615
Error = (Calculated Baud Rate – Desired Baud Rate)/Desired Baud Rate
= (9615 – 9600)/9600 = 0.16%


© 2002-2015 Microchip Technology Inc.

**TABLE 16-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 0000 -010 | 0000 -010 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 -00x | 0000 -00x |
| BAUDCTL | — | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | -1-1 0-00 | -1-1 0-00 |
| SPBRGH | Baud Rate Generator Register High Byte | | | | | | | | 0000 0000 | 0000 0000 |
| SPBRG | Baud Rate Generator Register Low Byte | | | | | | | | 0000 0000 | 0000 0000 |

**Legend:** x = unknown, – = unimplemented, read as '0'. Shaded cells are not used by the BRG.

**TABLE 16-3: BAUD RATES FOR ASYNCHRONOUS MODES**

| BAUD RATE (K) | SYNC = 0, BRGH = 0, BRG16 = 0 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FOSC = 40.000 MHz | | | FOSC = 20.000 MHz | | | FOSC = 10.000 MHz | | | FOSC = 8.000 MHz | | |
| | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) |
| 0.3 | — | — | — | — | — | — | — | — | — | — | — | — |
| 1.2 | — | — | — | 1.221 | 1.73 | 255 | 1.202 | 0.16 | 129 | 1201 | -0.16 | 103 |
| 2.4 | 2.441 | 1.73 | 255 | 2.404 | 0.16 | 129 | 2.404 | 0.16 | 64 | 2403 | -0.16 | 51 |
| 9.6 | 9.615 | 0.16 | 64 | 9.766 | 1.73 | 31 | 9.766 | 1.73 | 15 | 9615 | -0.16 | 12 |
| 19.2 | 19.531 | 1.73 | 31 | 19.531 | 1.73 | 15 | 19.531 | 1.73 | 7 | — | — | — |
| 57.6 | 56.818 | -1.36 | 10 | 62.500 | 8.51 | 4 | 52.083 | -9.58 | 2 | — | — | — |
| 115.2 | 125.000 | 8.51 | 4 | 104.167 | -9.58 | 2 | 78.125 | -32.18 | 1 | — | — | — |

| BAUD RATE (K) | SYNC = 0, BRGH = 0, BRG16 = 0 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | FOSC = 4.000 MHz | | | FOSC = 2.000 MHz | | | FOSC = 1.000 MHz | | |
| | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) |
| 0.3 | 0.300 | 0.16 | 207 | 300 | -0.16 | 103 | 300 | -0.16 | 51 |
| 1.2 | 1.202 | 0.16 | 51 | 1201 | -0.16 | 25 | 1201 | -0.16 | 12 |
| 2.4 | 2.404 | 0.16 | 25 | 2403 | -0.16 | 12 | — | — | — |
| 9.6 | 8.929 | -6.99 | 6 | — | — | — | — | — | — |
| 19.2 | 20.833 | 8.51 | 2 | — | — | — | — | — | — |
| 57.6 | 62.500 | 8.51 | 0 | — | — | — | — | — | — |
| 115.2 | 62.500 | -45.75 | 0 | — | — | — | — | — | — |

**TABLE 16-3: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)**

| BAUD RATE (K) | SYNC = 0, BRGH = 1, BRG16 = 0 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FOSC = 40.000 MHz | | | FOSC = 20.000 MHz | | | FOSC = 10.000 MHz | | | FOSC = 8.000 MHz | | |
| | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) |
| 2.4 | — | — | — | — | — | — | 2.441 | 1.73 | 255 | 2403 | -0.16 | 207 |
| 9.6 | 9.766 | 1.73 | 255 | 9.615 | 0.16 | 129 | 9.615 | 0.16 | 64 | 9615 | -0.16 | 51 |
| 19.2 | 19.231 | 0.16 | 129 | 19.231 | 0.16 | 64 | 19.531 | 1.73 | 31 | 19230 | -0.16 | 25 |
| 57.6 | 58.140 | 0.94 | 42 | 56.818 | -1.36 | 21 | 56.818 | -1.36 | 10 | 55555 | 3.55 | 8 |
| 115.2 | 113.636 | -1.36 | 21 | 113.636 | -1.36 | 10 | 125.000 | 8.51 | 4 | — | — | — |

| BAUD RATE (K) | SYNC = 0, BRGH = 1, BRG16 = 0 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | FOSC = 4.000 MHz | | | FOSC = 2.000 MHz | | | FOSC = 1.000 MHz | | |
| | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) |
| 0.3 | — | — | — | — | — | — | 300 | -0.16 | 207 |
| 1.2 | 1.202 | 0.16 | 207 | 1201 | -0.16 | 103 | 1201 | -0.16 | 51 |
| 2.4 | 2.404 | 0.16 | 103 | 2403 | -0.16 | 51 | 2403 | -0.16 | 25 |
| 9.6 | 9.615 | 0.16 | 25 | 9615 | -0.16 | 12 | — | — | — |
| 19.2 | 19.231 | 0.16 | 12 | — | — | — | — | — | — |
| 57.6 | 62.500 | 8.51 | 3 | — | — | — | — | — | — |
| 115.2 | 125.000 | 8.51 | 1 | — | — | — | — | — | — |

| BAUD RATE (K) | SYNC = 0, BRGH = 0, BRG16 = 1 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FOSC = 40.000 MHz | | | FOSC = 20.000 MHz | | | FOSC = 10.000 MHz | | | FOSC = 8.000 MHz | | |
| | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) |
| 0.3 | 0.300 | 0.00 | 8332 | 0.300 | 0.02 | 4165 | 0.300 | 0.02 | 2082 | 300 | -0.04 | 1665 |
| 1.2 | 1.200 | 0.02 | 2082 | 1.200 | -0.03 | 1041 | 1.200 | -0.03 | 520 | 1201 | -0.16 | 415 |
| 2.4 | 2.402 | 0.06 | 1040 | 2.399 | -0.03 | 520 | 2.404 | 0.16 | 259 | 2403 | -0.16 | 207 |
| 9.6 | 9.615 | 0.16 | 259 | 9.615 | 0.16 | 129 | 9.615 | 0.16 | 64 | 9615 | -0.16 | 51 |
| 19.2 | 19.231 | 0.16 | 129 | 19.231 | 0.16 | 64 | 19.531 | 1.73 | 31 | 19230 | -0.16 | 25 |
| 57.6 | 58.140 | 0.94 | 42 | 56.818 | -1.36 | 21 | 56.818 | -1.36 | 10 | 55555 | 3.55 | 8 |
| 115.2 | 113.636 | -1.36 | 21 | 113.636 | -1.36 | 10 | 125.000 | 8.51 | 4 | — | — | — |

| BAUD RATE (K) | SYNC = 0, BRGH = 0, BRG16 = 1 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | FOSC = 4.000 MHz | | | FOSC = 2.000 MHz | | | FOSC = 1.000 MHz | | |
| | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) |
| 0.3 | 0.300 | 0.04 | 832 | 300 | -0.16 | 415 | 300 | -0.16 | 207 |
| 1.2 | 1.202 | 0.16 | 207 | 1201 | -0.16 | 103 | 1201 | -0.16 | 51 |
| 2.4 | 2.404 | 0.16 | 103 | 2403 | -0.16 | 51 | 2403 | -0.16 | 25 |
| 9.6 | 9.615 | 0.16 | 25 | 9615 | -0.16 | 12 | — | — | — |
| 19.2 | 19.231 | 0.16 | 12 | — | — | — | — | — | — |
| 57.6 | 62.500 | 8.51 | 3 | — | — | — | — | — | — |
| 115.2 | 125.000 | 8.51 | 1 | — | — | — | — | — | — |


© 2002-2015 Microchip Technology Inc.

**TABLE 16-3: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)**

| BAUD RATE (K) | SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FOSC = 40.000 MHz | | | FOSC = 20.000 MHz | | | FOSC = 10.000 MHz | | | FOSC = 8.000 MHz | | |
| | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) |
| 0.3 | 0.300 | 0.00 | 33332 | 0.300 | 0.00 | 16665 | 0.300 | 0.00 | 8332 | 300 | -0.01 | 6665 |
| 1.2 | 1.200 | 0.00 | 8332 | 1.200 | 0.02 | 4165 | 1.200 | 0.02 | 2082 | 1200 | -0.04 | 1665 |
| 2.4 | 2.400 | 0.02 | 4165 | 2.400 | 0.02 | 2082 | 2.402 | 0.06 | 1040 | 2400 | -0.04 | 832 |
| 9.6 | 9.606 | 0.06 | 1040 | 9.596 | -0.03 | 520 | 9.615 | 0.16 | 259 | 9615 | -0.16 | 207 |
| 19.2 | 19.193 | -0.03 | 520 | 19.231 | 0.16 | 259 | 19.231 | 0.16 | 129 | 19230 | -0.16 | 103 |
| 57.6 | 57.803 | 0.35 | 172 | 57.471 | -0.22 | 86 | 58.140 | 0.94 | 42 | 57142 | 0.79 | 34 |
| 115.2 | 114.943 | -0.22 | 86 | 116.279 | 0.94 | 42 | 113.636 | -1.36 | 21 | 117647 | -2.12 | 16 |

| BAUD RATE (K) | SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | FOSC = 4.000 MHz | | | FOSC = 2.000 MHz | | | FOSC = 1.000 MHz | | |
| | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) |
| 0.3 | 0.300 | 0.01 | 3332 | 300 | -0.04 | 1665 | 300 | -0.04 | 832 |
| 1.2 | 1.200 | 0.04 | 832 | 1201 | -0.16 | 415 | 1201 | -0.16 | 207 |
| 2.4 | 2.404 | 0.16 | 415 | 2403 | -0.16 | 207 | 2403 | -0.16 | 103 |
| 9.6 | 9.615 | 0.16 | 103 | 9615 | -0.16 | 51 | 9615 | -0.16 | 25 |
| 19.2 | 19.231 | 0.16 | 51 | 19230 | -0.16 | 25 | 19230 | -0.16 | 12 |
| 57.6 | 58.824 | 2.12 | 16 | 55555 | 3.55 | 8 | — | — | — |
| 115.2 | 111.111 | -3.55 | 8 | — | — | — | — | — | — |

### 16.2.3 AUTO-BAUD RATE DETECT

The Enhanced USART module supports the automatic detection and calibration of baud rate. This feature is active only in Asynchronous mode and while the WUE bit is clear.

The automatic baud rate measurement sequence (Figure 16-1) begins whenever a Start bit is received and the ABDEN bit is set. The calculation is self-averaging.

In the Auto-Baud Rate Detect (ABD) mode, the clock to the BRG is reversed. Rather than the BRG clocking the incoming RX signal, the RX signal is timing the BRG. In ABD mode, the internal Baud Rate Generator is used as a counter to time the bit period of the incoming serial byte stream.

Once the ABDEN bit is set, the state machine will clear the BRG and look for a Start bit. The Auto-Baud Detect must receive a byte with the value 55h (ASCII "U", which is also the LIN bus Sync character), in order to calculate the proper bit rate. The measurement is taken over both a low and a high bit time in order to minimize any effects caused by asymmetry of the incoming signal. After a Start bit, the SPBRG begins counting up using the preselected clock source on the first rising edge of RX. After eight bits on the RX pin, or the fifth rising edge, an accumulated value totaling the proper BRG period is left in the SPBRGH:SPBRG registers. Once the fifth edge is seen (should correspond to the Stop bit), the ABDEN bit is automatically cleared.

While calibrating the baud rate period, the BRG registers are clocked at 1/8th the preconfigured clock rate. Note that the BRG clock will be configured by the BRG16 and BRGH bits. Independent of the BRG16 bit setting, both the SPBRG and SPBRGH will be used as a 16-bit counter. This allows the user to verify that no carry occurred for 8-bit modes, by checking for 00h in the SPBRGH register. Refer to Table 16-4 for counter clock rates to the BRG.

While the ABD sequence takes place, the EUSART state machine is held in Idle. The RCIF interrupt is set once the fifth rising edge on RX is detected. The value in the RCREG needs to be read to clear the RCIF interrupt. RCREG content should be discarded.

| | |
|---|---|
| **Note 1:** | It is up to the user to determine that the incoming character baud rate is within the range of the selected BRG clock source. Some combinations of oscillator frequency and EUSART baud rates are not possible due to bit error rates. Overall system timing and communication baud rates must be taken into consideration when using the Auto-Baud Rate Detection feature. |

### 16.2.4 RECEIVING A SYNC (AUTO-BAUD RATE DETECT)

To receive a Sync (Auto-Baud Rate Detect):

1. Configure the EUSART for asynchronous receive. TXEN should remain clear. SPBRGH:SPBRG may be left as is. The controller should operate in either PRI_RUN or PRI_IDLE.
2. Enable RXIF interrupts. Set RCIE, PEIE, GIE.
3. Enable Auto-Baud Rate Detect. Set ABDEN.
4. When the next RCIF interrupt occurs, the received baud rate has been measured. Read RCREG to clear RCIF and discard. Check SPBRGH:SPBRG for a valid value. The EUSART is ready for normal communications. Return from the interrupt. Allow the primary clock to run (PRI_RUN or PRI_IDLE).
5. Process subsequent RCIF interrupts normally as in asynchronous reception. Remain in PRI_RUN or PRI_IDLE until communications are complete.

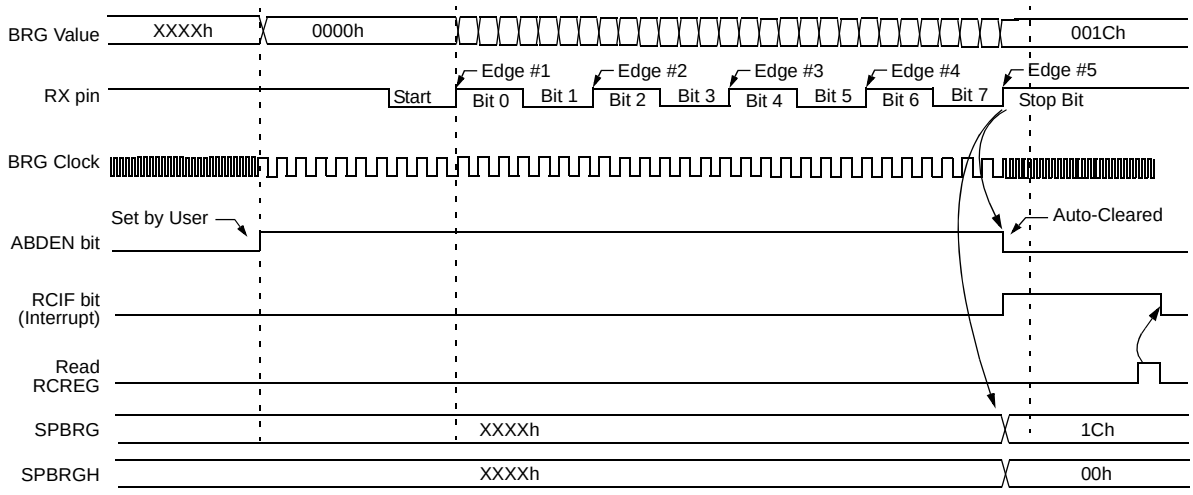
**TABLE 16-4: BRG COUNTER CLOCK RATES**

| BRG16 | BRGH | BRG Counter Clock |
|---|---|---|
| 0 | 0 | FOSC/512 |
| 0 | 1 | FOSC/128 |
| 1 | 0 | FOSC/128 |
| 1 | 1 | FOSC/32 |

**Note:** During the ABD sequence, SPBRG and SPBRGH are both used as a 16-bit counter, independent of BRG16 setting.


© 2002-2015 Microchip Technology Inc.

**FIGURE 16-1: AUTOMATIC BAUD RATE CALCULATION**




**Note 1:** The ABD sequence requires the EUSART module to be configured in Asynchronous mode and WUE = 0.

## 16.3 EUSART Asynchronous Mode

The Asynchronous mode of operation is selected by clearing the SYNC bit (TXSTA<4>). In this mode, the EUSART uses standard Non-Return-to-Zero (NRZ) format (one Start bit, eight or nine data bits and one Stop bit). The most common data format is eight bits. An on-chip dedicated 8-bit/16-bit Baud Rate Generator can be used to derive standard baud rate frequencies from the oscillator.

The EUSART transmits and receives the LSb first. The EUSART's transmitter and receiver are functionally independent, but use the same data format and baud rate. The Baud Rate Generator produces a clock, either x16 or x64 of the bit shift rate, depending on the BRGH and BRG16 bits (TXSTA<2> and BAUDCTL<3>). Parity is not supported by the hardware, but can be implemented in software and stored as the ninth data bit.

Asynchronous mode is available in all low-power modes; it is available in Sleep mode only when auto-wake-up on Sync Break is enabled. When in PRI_IDLE mode, no changes to the Baud Rate Generator values are required; however, other low-power mode clocks may operate at another frequency than the primary clock. Therefore, the Baud Rate Generator values may need to be adjusted.

When operating in Asynchronous mode, the EUSART module consists of the following important elements:

- Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- Asynchronous Receiver
- Auto-Wake-up on Sync Break Character
- 12-bit Break Character Transmit
- Auto-Baud Rate Detection

### 16.3.1 EUSART ASYNCHRONOUS TRANSMITTER

The EUSART transmitter block diagram is shown in Figure 16-2. The heart of the transmitter is the Transmit (Serial) Shift Register (TSR). The shift register obtains its data from the Read/Write Transmit Buffer register, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the Stop bit has been transmitted from the previous load. As soon as the Stop bit is transmitted, the TSR is loaded with new data from the TXREG register (if available).

Once the TXREG register transfers the data to the TSR register (occurs in one TCY), the TXREG register is empty and flag bit, TXIF (PIR1<4>), is set. This interrupt can be enabled/disabled by setting/clearing enable bit, TXIE (PIE1<4>). Flag bit, TXIF, will be set, regardless of the state of enable bit, TXIE, and cannot be cleared in software. Flag bit, TXIF, is not cleared immediately upon loading the Transmit Buffer register, TXREG. TXIF becomes valid in the second instruction cycle following the load instruction. Polling TXIF immediately following a load of TXREG will return invalid results.

While flag bit, TXIF, indicates the status of the TXREG register, another bit, TRMT (TXSTA<1>), shows the status of the TSR register. Status bit, TRMT, is a read-only bit, which is set when the TSR register is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR register is empty.

**Note 1:** The TSR register is not mapped in data memory, so it is not available to the user.

**2:** Flag bit, TXIF, is set when enable bit, TXEN, is set.

To set up an Asynchronous Transmission:

1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set transmit bit TX9. Can be used as address/data bit.
5. Enable the transmission by setting bit TXEN, which will also set bit TXIF.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Load data to the TXREG register (starts transmission).

If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**FIGURE 16-2: EUSART TRANSMIT BLOCK DIAGRAM**

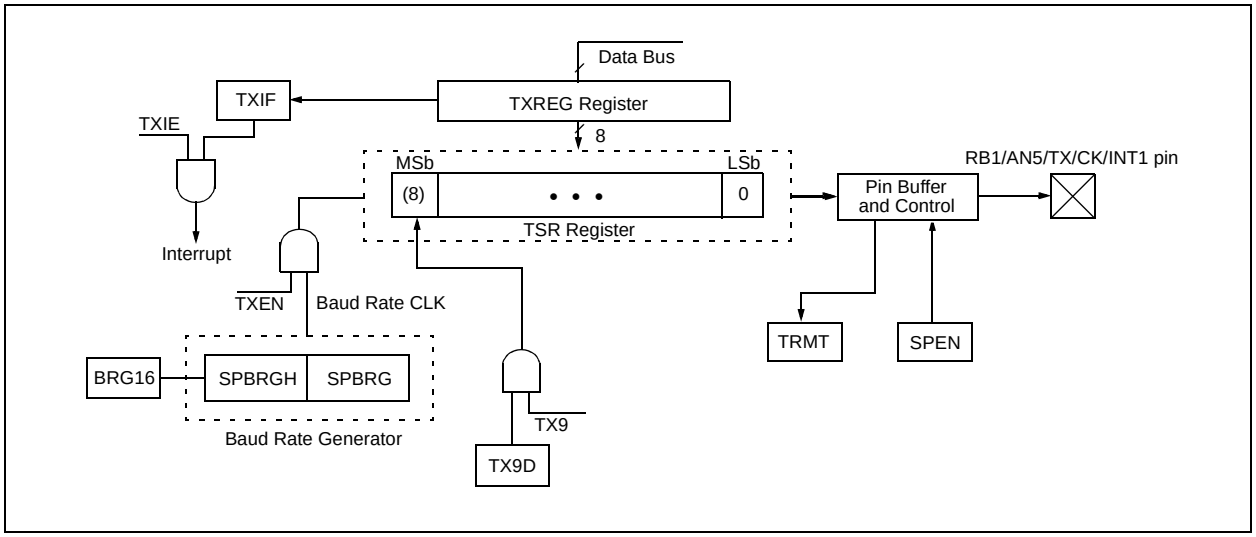
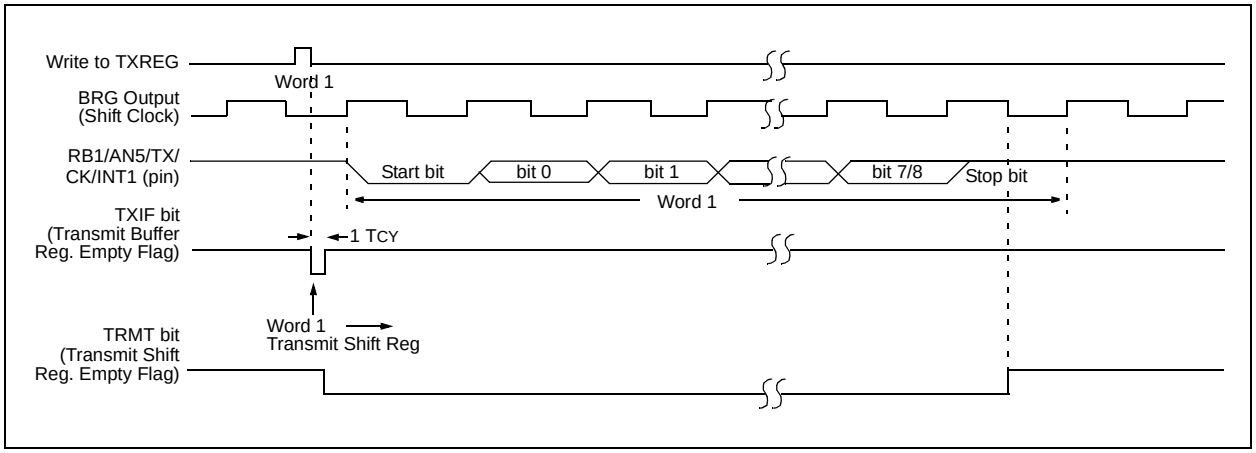


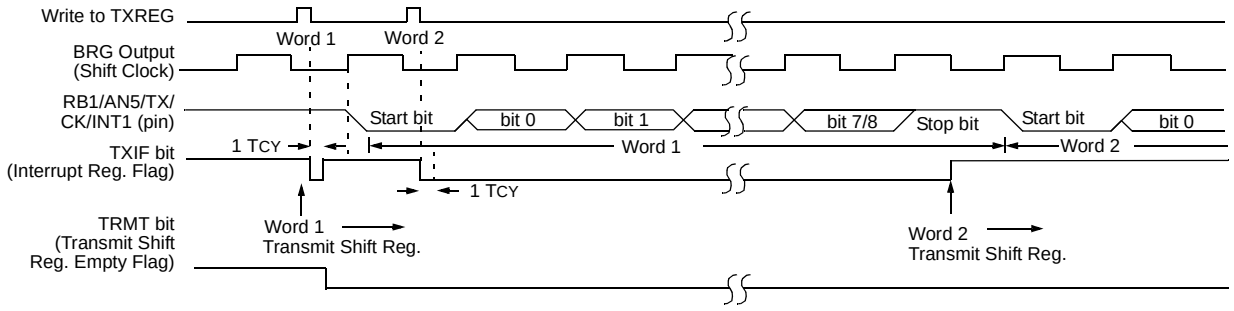

**FIGURE 16-3: ASYNCHRONOUS TRANSMISSION**





© 2002-2015 Microchip Technology Inc.

**FIGURE 16-4: ASYNCHRONOUS TRANSMISSION (BACK TO BACK)**




**Note:** This timing diagram shows two consecutive transmissions.

**TABLE 16-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 0000 000u |
| PIR1 | — | ADIF | RCIF | TXIF | — | CCP1IF | TMR2IF | TMR1IF | -000 -000 | -000 -000 |
| PIE1 | — | ADIE | RCIE | TXIE | — | CCP1IE | TMR2IE | TMR1IE | -000 -000 | -000 -000 |
| IPR1 | — | ADIP | RCIP | TXIP | — | CCP1IP | TMR2IP | TMR1IP | -111 -111 | -111 -111 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 -00x | 0000 -00x |
| TXREG | EUSART Transmit Register | | | | | | | | 0000 0000 | 0000 0000 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 0000 0010 | 0000 0010 |
| BAUDCTL | — | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | -1-1 0-00 | -1-1 0-00 |
| SPBRGH | Baud Rate Generator Register High Byte | | | | | | | | 0000 0000 | 0000 0000 |
| SPBRG | Baud Rate Generator Register Low Byte | | | | | | | | 0000 0000 | 0000 0000 |

**Legend:** x = unknown, – = unimplemented locations read as '0'. Shaded cells are not used for asynchronous transmission.

### 16.3.2 EUSART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 16-5. The data is received on the RB4/AN6/RX/DT/KBI0 pin and drives the data recovery block. The data recovery block is actually a high-speed shifter, operating at x16 times the baud rate, whereas the main receive serial shifter operates at the bit rate or at FOSC. This mode would typically be used in RS-232 systems.

To set up an Asynchronous Reception:

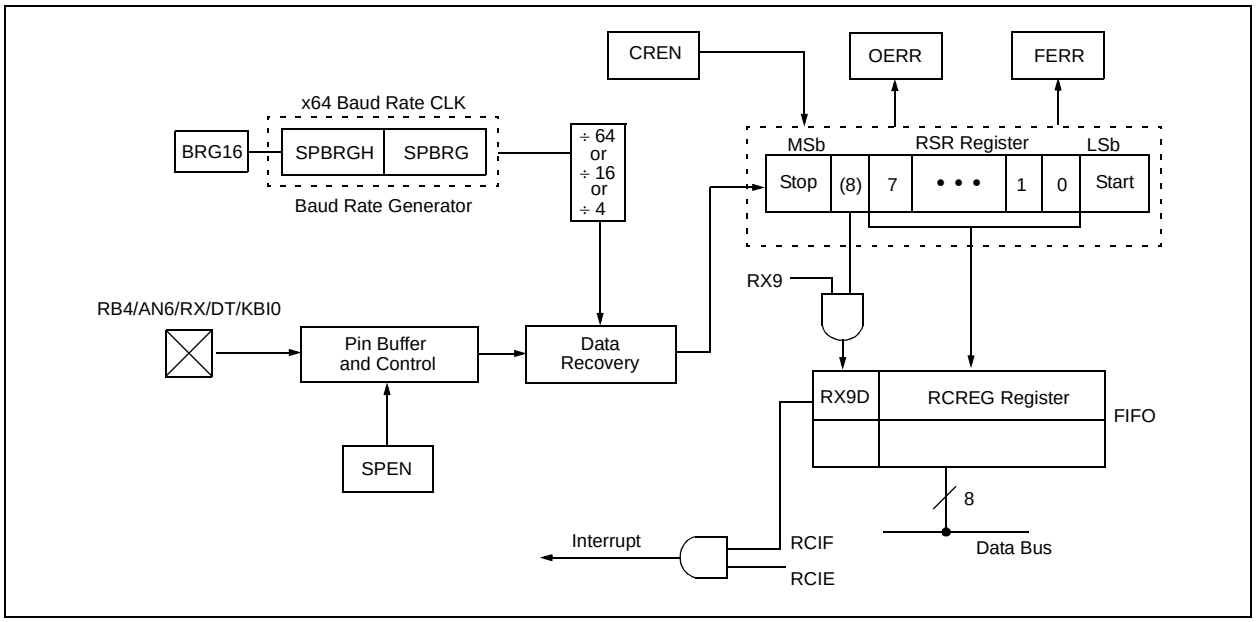
1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, set enable bit RCIE.
4. If 9-bit reception is desired, set bit RX9.
5. Enable the reception by setting bit CREN.
6. Flag bit, RCIF, will be set when reception is complete and an interrupt will be generated if enable bit RCIE was set.
7. Read the RCSTA register to get the 9th bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading the RCREG register.
9. If any error occurred, clear the error by clearing enable bit CREN.
10. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

### 16.3.3 SETTING UP 9-BIT MODE WITH ADDRESS DETECT

This mode would typically be used in RS-485 systems. To set up an Asynchronous Reception with Address Detect Enable:

1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If interrupts are required, set the RCEN bit and select the desired priority level with the RCIP bit.
4. Set the RX9 bit to enable 9-bit reception.
5. Set the ADDEN bit to enable address detect.
6. Enable reception by setting the CREN bit.
7. The RCIF bit will be set when reception is complete. The interrupt will be Acknowledged if the RCIE and GIE bits are set.
8. Read the RCSTA register to determine if any error occurred during reception, as well as read bit 9 of data (if applicable).
9. Read RCREG to determine if the device is being addressed.
10. If any error occurred, clear the CREN bit.
11. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and interrupt the CPU.

**FIGURE 16-5: EUSART RECEIVE BLOCK DIAGRAM**





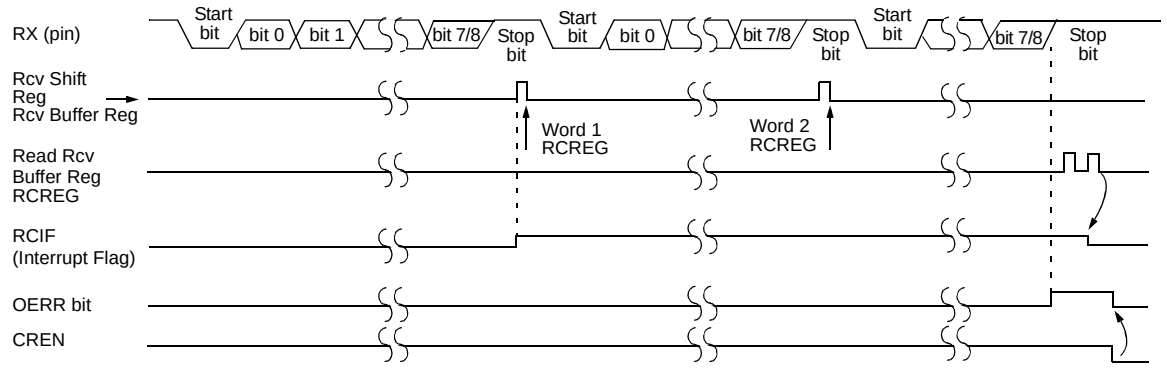
© 2002-2015 Microchip Technology Inc.

To set up an Asynchronous Transmission:

1. Initialize the SPBRG register for the appropriate baud rate. If a high-speed baud rate is desired, set bit BRGH (see **Section 16.2 "EUSART Baud Rate Generator (BRG)"**).
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set transmit bit TX9. Can be used as address/data bit.
5. Enable the transmission by setting bit TXEN, which will also set bit TXIF.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Load data to the TXREG register (starts transmission).

If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**FIGURE 16-6: ASYNCHRONOUS RECEPTION**




**Note:** This timing diagram shows three words appearing on the RX input. The RCREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.

**TABLE 16-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 0000 000u |
| PIR1 | — | ADIF | RCIF | TXIF | — | CCP1IF | TMR2IF | TMR1IF | -000 -000 | -000 -000 |
| PIE1 | — | ADIE | RCIE | TXIE | — | CCP1IE | TMR2IE | TMR1IE | -000 -000 | -000 -000 |
| IPR1 | — | ADIP | RCIP | TXIP | — | CCP1IP | TMR2IP | TMR1IP | -111 -111 | -111 -111 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x | 0000 000x |
| RCREG | EUSART Receive Register | | | | | | | | 0000 0000 | 0000 0000 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 0000 0010 | 0000 0010 |
| BAUDCTL | — | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | -1-1 0-00 | -1-1 0-00 |
| SPBRGH | Baud Rate Generator Register High Byte | | | | | | | | 0000 0000 | 0000 0000 |
| SPBRG | Baud Rate Generator Register Low Byte | | | | | | | | 0000 0000 | 0000 0000 |

**Legend:** x = unknown, – = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.

### 16.3.4 AUTO-WAKE-UP ON SYNC BREAK CHARACTER

During Sleep mode, all clocks to the EUSART are suspended. Because of this, the Baud Rate Generator is inactive and a proper byte reception cannot be performed. The auto-wake-up feature allows the controller to wake-up due to activity on the RX/DT line while the EUSART is operating in Asynchronous mode.

The auto-wake-up feature is enabled by setting the WUE bit (BAUDCTL<1>). Once set, the typical receive sequence on RX/DT is disabled and the EUSART remains in an Idle state, monitoring for a wake-up event independent of the CPU mode. A wake-up event consists of a high-to-low transition on the RX/DT line. (This coincides with the start of a Sync Break or a Wake-up Signal character for the LIN protocol.)

Following a wake-up event, the module generates an RCIF interrupt. The interrupt is generated synchronously to the Q clocks in normal operating modes (Figure 16-7) and asynchronously if the device is in Sleep mode (Figure 16-8). The interrupt condition is cleared by reading the RCREG register.

The WUE bit is automatically cleared once a low-to-high transition is observed on the RX line, following the wake-up event. At this point, the EUSART module is in Idle mode and returns to normal operation. This signals to the user that the Sync Break event is over.

#### 16.3.4.1 Special Considerations Using Auto-Wake-up

Since auto-wake-up functions by sensing rising edge transitions on RX/DT, information with any state changes before the Stop bit may signal a false end-of-character and cause data or framing errors. To work properly, therefore, the initial character in the transmission must be all '0's. This can be 00h (8 bytes) for standard RS-232 devices, or 000h (12 bits) for LIN bus.

Oscillator start-up time must also be considered, especially in applications using oscillators with longer start-up intervals (i.e., LP, XT or HS/PLL mode). The Sync Break (or Wake-up Signal) character must be of sufficient length and be followed by a sufficient period, to allow enough time for the selected oscillator to start and provide proper initialization of the EUSART.

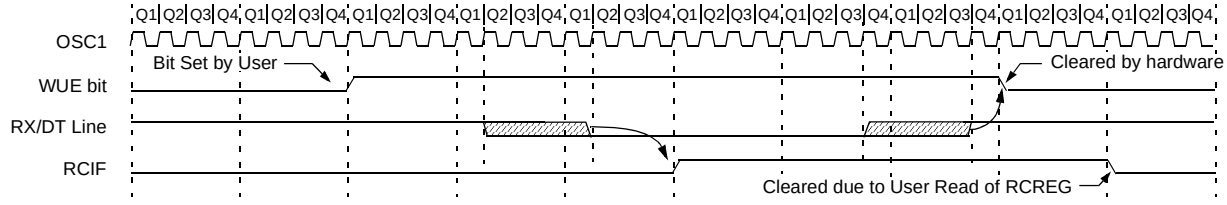
#### 16.3.4.2 Special Considerations Using the WUE Bit

The timing of WUE and RCIF events may cause some confusion when it comes to determining the validity of received data. As noted, setting the WUE bit places the EUSART in an Idle mode. The wake-up event causes a receive interrupt by setting the RCIF bit. The WUE bit is cleared after this when a rising edge is seen on RX/DT. The interrupt condition is then cleared by reading the RCREG register. Ordinarily, the data in RCREG will be dummy data and should be discarded.

The fact that the WUE bit has been cleared (or is still set) and the RCIF flag is set should not be used as an indicator of the integrity of the data in RCREG. Users should consider implementing a parallel method in firmware to verify received data integrity.

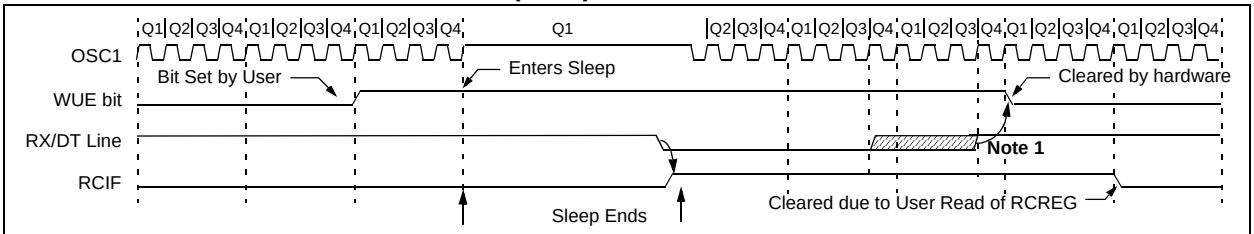
To assure that no actual data is lost, check the RCIDL bit to verify that a receive operation is not in process. If a receive operation is not occurring, the WUE bit may then be set just prior to entering the Sleep mode.

**FIGURE 16-7: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING NORMAL OPERATION**




**Note 1:** The EUSART remains in Idle while the WUE bit is set.

**FIGURE 16-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP**




**Note 1:** If the wake-up event requires a long oscillator warm-up time, the WUE bit may be cleared while the primary clock is still starting.
**2:** The EUSART remains in Idle while the WUE bit is set.


© 2002-2015 Microchip Technology Inc.

### 16.3.5 BREAK CHARACTER SEQUENCE

The Enhanced USART module has the capability of sending the special Break character sequences that are required by the LIN bus standard. The Break character transmit consists of a Start bit, followed by twelve '0' bits and a Stop bit. The Frame Break character is sent whenever the SENDB and TXEN bits (TXSTA<3> and TXSTA<5>) are set while the Transmit Shift register is loaded with data. Note that the value of data written to TXREG will be ignored and all '0's will be transmitted.

The SENDB bit is automatically reset by hardware after the corresponding Stop bit is sent. This allows the user to preload the transmit FIFO with the next transmit byte following the Break character (typically, the Sync character in the LIN specification).

Note that the data value written to the TXREG for the Break character is ignored. The write simply serves the purpose of initiating the proper sequence.

The TRMT bit indicates when the transmit operation is active or Idle, just as it does during normal transmission. See Figure 16-9 for the timing of the Break character sequence.

#### 16.3.5.1 Transmitting A Break Signal

The Enhanced USART module has the capability of sending the Break signal that is required by the LIN bus standard. The Break signal consists of a Start bit, followed by twelve '0' bits and a Stop bit. The Break signal is sent whenever the SENDB (TXSTA<3>) and TXEN (TXSTA<5>) bits are set and TXREG is loaded with data. The data written to TXREG will be ignored and all '0's will be transmitted.

SENDB is automatically cleared by hardware when the Break signal has been sent. This allows the user to preload the transmit FIFO with the next transmit byte following the Break character (typically, the Sync character in the LIN specification).

The TRMT bit indicates when the transmit operation is active or Idle, just as it does during normal transmission.

To send a Break Signal:

1. Configure the EUSART for asynchronous transmissions (steps 1-5). Initialize the SPBRG register for the appropriate baud rate. If a high-speed baud rate is desired, set bit BRGH (see **Section 16.2 "EUSART Baud Rate Generator (BRG)"**).
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set transmit bit TX9. Can be used as address/data bit.
5. Enable the transmission by setting bit TXEN, which will also set bit TXIF.
6. Set the SENDB bit.
7. Load a byte into TXREG. This triggers sending a Break signal. The Break signal is complete when TRMT is set. SENDB will also be cleared.

See Figure 16-9 for the timing of the Break signal sequence.

### 16.3.6 RECEIVING A BREAK CHARACTER

The Enhanced USART module can receive a Break character in two ways.

The first method forces configuration of the baud rate at a frequency of 9/13 the typical speed. This allows for the Stop bit transition to be at the correct sampling location (12 bits for Break versus Start bit and eight data bits for typical data).

The second method uses the auto-wake-up feature described in **Section 16.3.4 "Auto-Wake-up on Sync Break Character"**. By enabling this feature, the EUSART will sample the next two transitions on RX/DT, cause an RCIF interrupt and receive the next data byte followed by another interrupt.

Note that following a Break character, the user will typically want to enable the Auto-Baud Rate Detect feature. For both methods, the user can set the ABD bit before placing the EUSART in its Sleep mode.

#### 16.3.6.1 Transmitting a Break Sync

The following sequence will send a message frame header made up of a Break, followed by an auto-baud Sync byte. This sequence is typical of a LIN bus master.

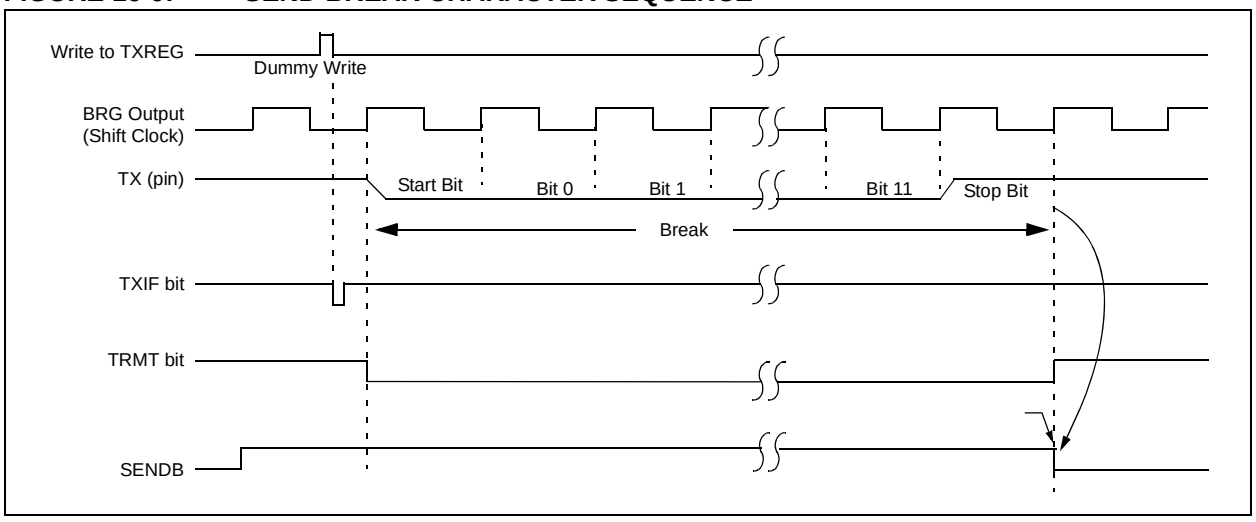
1. Configure the EUSART for the desired mode.
2. Set the TXEN and SENDB bits to set up the Break character.
3. Load the TXREG with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXREG to load the Sync character into the transmit FIFO buffer.
5. After the Break has been sent, the SENDB bit is reset by hardware. The Sync character now transmits in the preconfigured mode. When the TXREG becomes empty, as indicated by the TXIF, the next data byte can be written to TXREG.

#### 16.3.6.2 Receiving a Break Sync

To receive a Break Sync:

1. Configure the EUSART for asynchronous transmit and receive. TXEN should remain clear. SPBRGH:SPBRG may be left as is.
2. Enable auto-wake-up. Set WUE.
3. Enable RXIF interrupts. Set RCIE, PEIE, GIE.
4. The controller may be placed in any power managed mode.
5. An RCIF will be generated at the beginning of the Break signal. When the interrupt is received, read RCREG to clear RCIF and discard. Allow the controller to return to PRI_RUN mode.
6. Wait for the RX line to go high at the end of the Break signal. Wait for any of the following: WUE to clear automatically (poll), RB4/RX to go high (poll) or for RBIF to be set (poll or interrupt). If RBIF is used, check to be sure that RB4/RX is high before continuing.
7. Enable Auto-Baud Rate Detect. Set ABDEN.
8. Return from the interrupt. Allow the primary clock to start and stabilize (PRI_RUN or PRI_IDLE).
9. When the next RCIF interrupt occurs, the received baud rate has been measured. Read RCREG to clear RCIF and discard. Check SPBRGH:SPBRG for a valid value. The EUSART is ready for normal communications. Return from the interrupt. Allow the primary clock to run (PRI_RUN or PRI_IDLE).
10. Process subsequent RCIF interrupts normally as in asynchronous reception. TXEN should now be set if transmissions are needed. TXIF and TXIE may be set if transmit interrupts are desired. Remain in PRI_RUN or PRI_IDLE until communications are complete. Clear TXEN and return to step 2.

**FIGURE 16-9: SEND BREAK CHARACTER SEQUENCE**





© 2002-2015 Microchip Technology Inc.

## 16.4 EUSART Synchronous Master Mode

The Synchronous Master mode is entered by setting the CSRC bit (TXSTA<7>). In this mode, the data is transmitted in a half-duplex manner (i.e., transmission and reception do not occur at the same time). When transmitting data, the reception is inhibited and vice versa. Synchronous mode is entered by setting bit, SYNC (TXSTA<4>). In addition, enable bit, SPEN (RCSTA<7>), is set in order to configure the RB1/AN5/TX/CK/INT1 and RB4/AN6/RX/DT/KBI0 I/O pins to CK (clock) and DT (data) lines, respectively.

The Master mode indicates that the processor transmits the master clock on the CK line. Clock polarity is selected with the SCKP bit (BAUDCTL<5>); setting SCKP sets the Idle state on CK as high, while clearing the bit sets the Idle state as low. This option is provided to support Microwire devices with this module.

### 16.4.1 EUSART SYNCHRONOUS MASTER TRANSMISSION

The EUSART transmitter block diagram is shown in Figure 16-2. The heart of the transmitter is the Transmit (Serial) Shift Register (TSR). The shift register obtains its data from the Read/Write Transmit Buffer register, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSR is loaded with new data from the TXREG (if available).

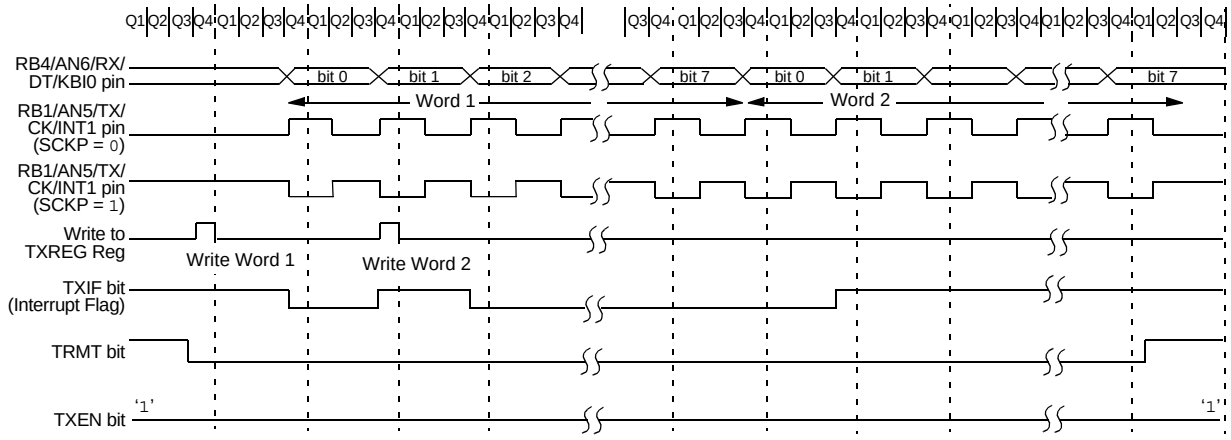
Once the TXREG register transfers the data to the TSR register (occurs in one TCYCLE), the TXREG is empty and interrupt bit, TXIF (PIR1<4>), is set. The interrupt can be enabled/disabled by setting/clearing enable bit, TXIE (PIE1<4>). Flag bit, TXIF, will be set, regardless of the state of enable bit, TXIE and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register.

While flag bit, TXIF, indicates the status of the TXREG register, another bit, TRMT (TXSTA<1>), shows the status of the TSR register. TRMT is a read-only bit, which is set when the TSR is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR register is empty. The TSR is not mapped in data memory, so it is not available to the user.

To set up a Synchronous Master Transmission:

1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set bit TX9.
5. Enable the transmission by setting bit TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Start transmission by loading data to the TXREG register.
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**FIGURE 16-10: SYNCHRONOUS TRANSMISSION**




**Note:** Sync Master mode, SPBRG = 0, continuous transmission of two 8-bit words.

**FIGURE 16-11: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)**

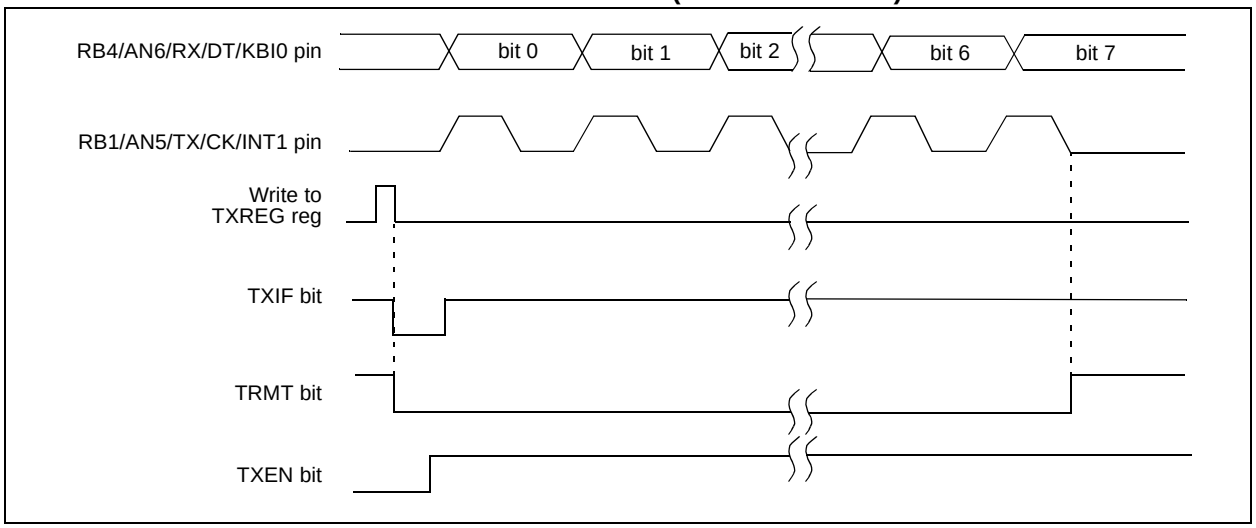



**TABLE 16-7: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 0000 000u |
| PIR1 | — | ADIF | RCIF | TXIF | — | CCP1IF | TMR2IF | TMR1IF | -000 -000 | -000 -000 |
| PIE1 | — | ADIE | RCIE | TXIE | — | CCP1IE | TMR2IE | TMR1IE | -000 -000 | -000 -000 |
| IPR1 | — | ADIP | RCIP | TXIP | — | CCP1IP | TMR2IP | TMR1IP | -111 -111 | -111 -111 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 -00x | 0000 -00x |
| TXREG | EUSART Transmit Register | | | | | | | | 0000 0000 | 0000 0000 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 0000 0010 | 0000 0010 |
| BAUDCTL | — | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | -1-1 0-00 | -1-1 0-00 |
| SPBRGH | Baud Rate Generator Register High Byte | | | | | | | | 0000 0000 | 0000 0000 |
| SPBRG | Baud Rate Generator Register Low Byte | | | | | | | | 0000 0000 | 0000 0000 |

**Legend:** x = unknown, – = unimplemented, read as '0'. Shaded cells are not used for synchronous master transmission.


© 2002-2015 Microchip Technology Inc.

### 16.4.2 EUSART SYNCHRONOUS MASTER RECEPTION

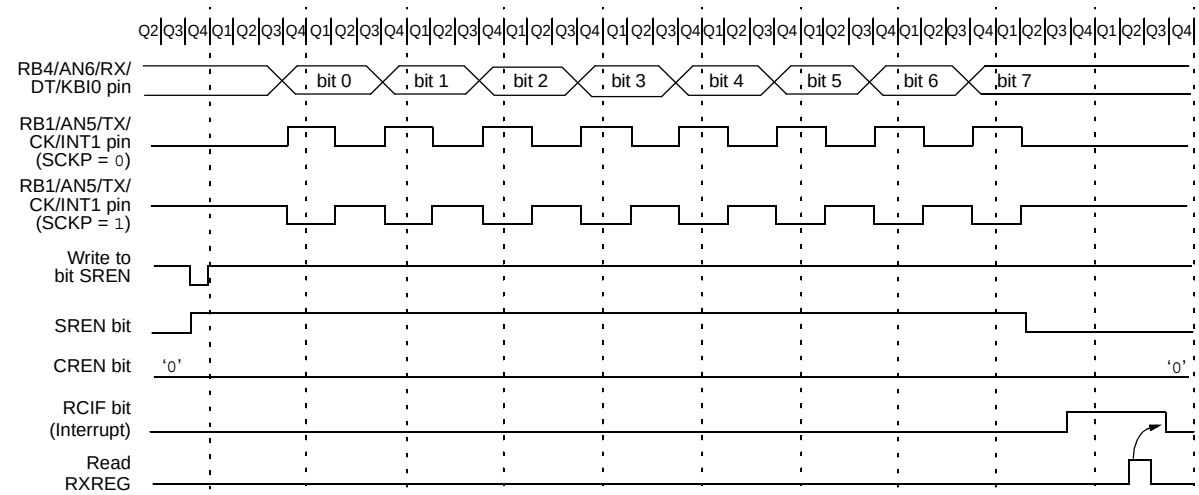
Once Synchronous mode is selected, reception is enabled by setting either the Single Receive Enable bit, SREN (RCSTA<5>), or the Continuous Receive Enable bit, CREN (RCSTA<4>). Data is sampled on the RB4/AN6/RX/DT/KBI0 pin on the falling edge of the clock.

If enable bit, SREN, is set, only a single word is received. If enable bit, CREN, is set, the reception is continuous until CREN is cleared. If both bits are set, then CREN takes precedence.

To set up a Synchronous Master Reception:

1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. Ensure bits CREN and SREN are clear.
4. If interrupts are desired, set enable bit RCIE.
5. If 9-bit reception is desired, set bit RX9.
6. If a single reception is required, set bit SREN. For continuous reception, set bit CREN.
7. Interrupt flag bit, RCIF, will be set when reception is complete and an interrupt will be generated if the enable bit, RCIE, was set.
8. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
9. Read the 8-bit received data by reading the RCREG register.
10. If any error occurred, clear the error by clearing bit CREN.
11. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**FIGURE 16-12: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)**




**Note:** Timing diagram demonstrates Sync Master mode with bit SREN = 1 and bit BRGH = 0.

**TABLE 16-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 0000 000u |
| PIR1 | — | ADIF | RCIF | TXIF | — | CCP1IF | TMR2IF | TMR1IF | -000 -000 | -000 -000 |
| PIE1 | — | ADIE | RCIE | TXIE | — | CCP1IE | TMR2IE | TMR1IE | -000 -000 | -000 -000 |
| IPR1 | — | ADIP | RCIP | TXIP | — | CCP1IP | TMR2IP | TMR1IP | -111 -111 | -111 -111 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x | 0000 000x |
| RCREG | EUSART Receive Register | | | | | | | | 0000 0000 | 0000 0000 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 0000 0010 | 0000 0010 |
| BAUDCTL | — | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | -1-1 0-00 | -1-1 0-00 |
| SPBRGH | Baud Rate Generator Register High Byte | | | | | | | | 0000 0000 | 0000 0000 |
| SPBRG | Baud Rate Generator Register Low Byte | | | | | | | | 0000 0000 | 0000 0000 |

**Legend:** x = unknown, – = unimplemented, read as '0'. Shaded cells are not used for synchronous master reception.


© 2002-2015 Microchip Technology Inc.

## 16.5 EUSART Synchronous Slave Mode

Synchronous Slave mode is entered by clearing bit, CSRC (TXSTA<7>). This mode differs from the Synchronous Master mode in that the shift clock is supplied externally at the RB1/AN5/TX/CK/INT1 pin (instead of being supplied internally in Master mode). This allows the device to transfer or receive data while in any low-power mode.

### 16.5.1 EUSART SYNCHRONOUS SLAVE TRANSMIT

The operation of the Synchronous Master and Slave modes are identical, except in the case of the Sleep mode.

If two words are written to the TXREG and then the SLEEP instruction is executed, the following will occur:

a) The first word will immediately transfer to the TSR register and transmit.
b) The second word will remain in the TXREG register.
c) Flag bit, TXIF, will not be set.
d) When the first word has been shifted out of TSR, the TXREG register will transfer the second word to the TSR and flag bit, TXIF, will now be set.
e) If enable bit, TXIE, is set, the interrupt will wake the chip from Sleep. If the global interrupt is enabled, the program will branch to the interrupt vector.

To set up a Synchronous Slave Transmission:

1. Enable the synchronous slave serial port by setting bits SYNC and SPEN and clearing bit CSRC.
2. Clear bits CREN and SREN.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set bit TX9.
5. Enable the transmission by setting enable bit TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Start transmission by loading data to the TXREG register.
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**TABLE 16-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 0000 000u |
| PIR1 | — | ADIF | RCIF | TXIF | — | CCP1IF | TMR2IF | TMR1IF | -000 -000 | -000 -000 |
| PIE1 | — | ADIE | RCIE | TXIE | — | CCP1IE | TMR2IE | TMR1IE | -000 -000 | -000 -000 |
| IPR1 | — | ADIP | RCIP | TXIP | — | CCP1IP | TMR2IP | TMR1IP | -111 -111 | -111 -111 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x | 0000 000x |
| TXREG | EUSART Transmit Register | | | | | | | | 0000 0000 | 0000 0000 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 0000 0010 | 0000 0010 |
| BAUDCTL | — | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | -1-1 0-00 | -1-1 0-00 |
| SPBRGH | Baud Rate Generator Register High Byte | | | | | | | | 0000 0000 | 0000 0000 |
| SPBRG | Baud Rate Generator Register Low Byte | | | | | | | | 0000 0000 | 0000 0000 |

**Legend:** x = unknown, – = unimplemented, read as '0'. Shaded cells are not used for synchronous slave transmission.

© 2002-2015 Microchip Technology Inc.

### 16.5.2 EUSART SYNCHRONOUS SLAVE RECEPTION

The operation of the Synchronous Master and Slave modes is identical, except in the case of Sleep, or any Idle mode and bit SREN, which is a “don’t care” in Slave mode.

If receive is enabled by setting the CREN bit prior to entering Sleep or any Idle mode, then a word may be received while in this low-power mode. Once the word is received, the RSR register will transfer the data to the RCREG register; if the RCIE enable bit is set, the interrupt generated will wake the chip from low-power mode. If the global interrupt is enabled, the program will branch to the interrupt vector.

To set up a Synchronous Slave Reception:

1. Enable the synchronous master serial port by setting bits SYNC and SPEN and clearing bit CSRC.
2. If interrupts are desired, set enable bit RCIE.
3. If 9-bit reception is desired, set bit RX9.
4. To enable reception, set enable bit CREN.
5. Flag bit, RCIF, will be set when reception is complete. An interrupt will be generated if enable bit, RCIE, was set.
6. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading the RCREG register.
8. If any error occurred, clear the error by clearing bit CREN.
9. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**TABLE 16-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 0000 000u |
| PIR1 | — | ADIF | RCIF | TXIF | — | CCP1IF | TMR2IF | TMR1IF | -000 -000 | -000 -000 |
| PIE1 | — | ADIE | RCIE | TXIE | — | CCP1IE | TMR2IE | TMR1IE | -000 -000 | -000 -000 |
| IPR1 | — | ADIP | RCIP | TXIP | — | CCP1IP | TMR2IP | TMR1IP | -111 -111 | -111 -111 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x | 0000 000x |
| RCREG | EUSART Receive Register | | | | | | | | 0000 0000 | 0000 0000 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 0000 0010 | 0000 0010 |
| BAUDCTL | — | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | -1-1 0-00 | -1-1 0-00 |
| SPBRGH | Baud Rate Generator Register High Byte | | | | | | | | 0000 0000 | 0000 0000 |
| SPBRG | Baud Rate Generator Register Low Byte | | | | | | | | 0000 0000 | 0000 0000 |

**Legend:** x = unknown, – = unimplemented, read as ‘0’. Shaded cells are not used for synchronous slave reception.


© 2002-2015 Microchip Technology Inc.

## 17.0 10-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) converter module has seven inputs for the PIC18F1220/1320 devices. This module allows conversion of an analog input signal to a corresponding 10-bit digital number.

A new feature for the A/D converter is the addition of programmable acquisition time. This feature allows the user to select a new channel for conversion and to set the GO/$\overline{\text{DONE}}$ bit immediately. When the GO/$\overline{\text{DONE}}$ bit is set, the selected channel is sampled for the programmed acquisition time before a conversion is actually started. This removes the firmware overhead that may have been required to allow for an acquisition (sampling) period (see Register 17-3 and **Section 17.3 "Selecting and Configuring Automatic Acquisition Time"**).

The module has five registers:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)
- A/D Control Register 2 (ADCON2)

The ADCON0 register, shown in Register 17-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 17-2, configures the functions of the port pins. The ADCON2 register, shown in Register 17-3, configures the A/D clock source, programmed acquisition time and justification.

© 2002-2015 Microchip Technology Inc.

**REGISTER 17-1: ADCON0: A/D CONTROL REGISTER 0**

| R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| VCFG1 | VCFG0 | — | CHS2 | CHS1 | CHS0 | GO/$\overline{\text{DONE}}$ | ADON |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-6 **VCFG<1:0>:** Voltage Reference Configuration bits

| | **A/D VREF+** | **A/D VREF-** |
|---|---|---|
| 00 | AVDD | AVSS |
| 01 | External VREF+ | AVSS |
| 10 | AVDD | External VREF- |
| 11 | External VREF+ | External VREF- |

bit 5 **Unimplemented:** Read as '0'

bit 4-2 **CHS<2:0>:** Analog Channel Select bits

000 = Channel 0 (AN0)
001 = Channel 1 (AN1)
010 = Channel 2 (AN2)
011 = Channel 3 (AN3)
100 = Channel 4 (AN4)
101 = Channel 5 (AN5)
110 = Channel 6 (AN6)
111 = Unimplemented(1)

bit 1 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit

When ADON = 1:
1 = A/D conversion in progress
0 = A/D Idle

bit 0 **ADON:** A/D On bit

1 = A/D converter module is enabled
0 = A/D converter module is disabled

Performing a conversion on unimplemented channels returns full-scale results.

**Note 1:** Performing a conversion on unimplemented channels returns full-scale results.


© 2002-2015 Microchip Technology Inc.

**REGISTER 17-2: ADCON1: A/D CONTROL REGISTER 1**

| U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | PCFG6 | PCFG5 | PCFG4 | PCFG3 | PCFG2 | PCFG1 | PCFG0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **Unimplemented:** Read as ‘0’

bit 6 **PCFG6:** A/D Port Configuration bit – AN6
1 = Pin configured as a digital port
0 = Pin configured as an analog channel – digital input disabled and reads ‘0’

bit 5 **PCFG5:** A/D Port Configuration bit – AN5
1 = Pin configured as a digital port
0 = Pin configured as an analog channel – digital input disabled and reads ‘0’

bit 4 **PCFG4:** A/D Port Configuration bit – AN4
1 = Pin configured as a digital port
0 = Pin configured as an analog channel – digital input disabled and reads ‘0’

bit 3 **PCFG3:** A/D Port Configuration bit – AN3
1 = Pin configured as a digital port
0 = Pin configured as an analog channel – digital input disabled and reads ‘0’

bit 2 **PCFG2:** A/D Port Configuration bit – AN2
1 = Pin configured as a digital port
0 = Pin configured as an analog channel – digital input disabled and reads ‘0’

bit 1 **PCFG1:** A/D Port Configuration bit – AN1
1 = Pin configured as a digital port
0 = Pin configured as an analog channel – digital input disabled and reads ‘0’

bit 0 **PCFG0:** A/D Port Configuration bit – AN0
1 = Pin configured as a digital port
0 = Pin configured as an analog channel – digital input disabled and reads ‘0’

© 2002-2015 Microchip Technology Inc.

**REGISTER 17-3: ADCON2: A/D CONTROL REGISTER 2**

| R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| ADFM | — | ACQT2 | ACQT1 | ACQT0 | ADCS2 | ADCS1 | ADCS0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **ADFM:** A/D Result Format Select bit
1 = Right justified
0 = Left justified

bit 6 **Unimplemented:** Read as ‘0’

bit 5-3 **ACQT<2:0>:** A/D Acquisition Time Select bits
000 = 0 TAD(1)
001 = 2 TAD
010 = 4 TAD
011 = 6 TAD
100 = 8 TAD
101 = 12 TAD
110 = 16 TAD
111 = 20 TAD)

bit 2-0 **ADCS<2:0>:** A/D Conversion Clock Select bits
000 = FOSC/2
001 = FOSC/8
010 = FOSC/32
011 = FRC (clock derived from A/D RC oscillator)(1)
100 = FOSC/4
101 = FOSC/16
110 = FOSC/64
111 = FRC (clock derived from A/D RC oscillator)(1)

**Note 1:** If the A/D FRC clock source is selected, a delay of one TCY (instruction cycle) is added before the A/D clock starts. This allows the SLEEP instruction to be executed before starting a conversion.


© 2002-2015 Microchip Technology Inc.

The analog reference voltage is software selectable to either the device's positive and negative supply voltage ($AV_{DD}$ and $AV_{SS}$), or the voltage level on the RA3/AN3/$V_{REF+}$ and RA2/AN2/$V_{REF-}$ pins.

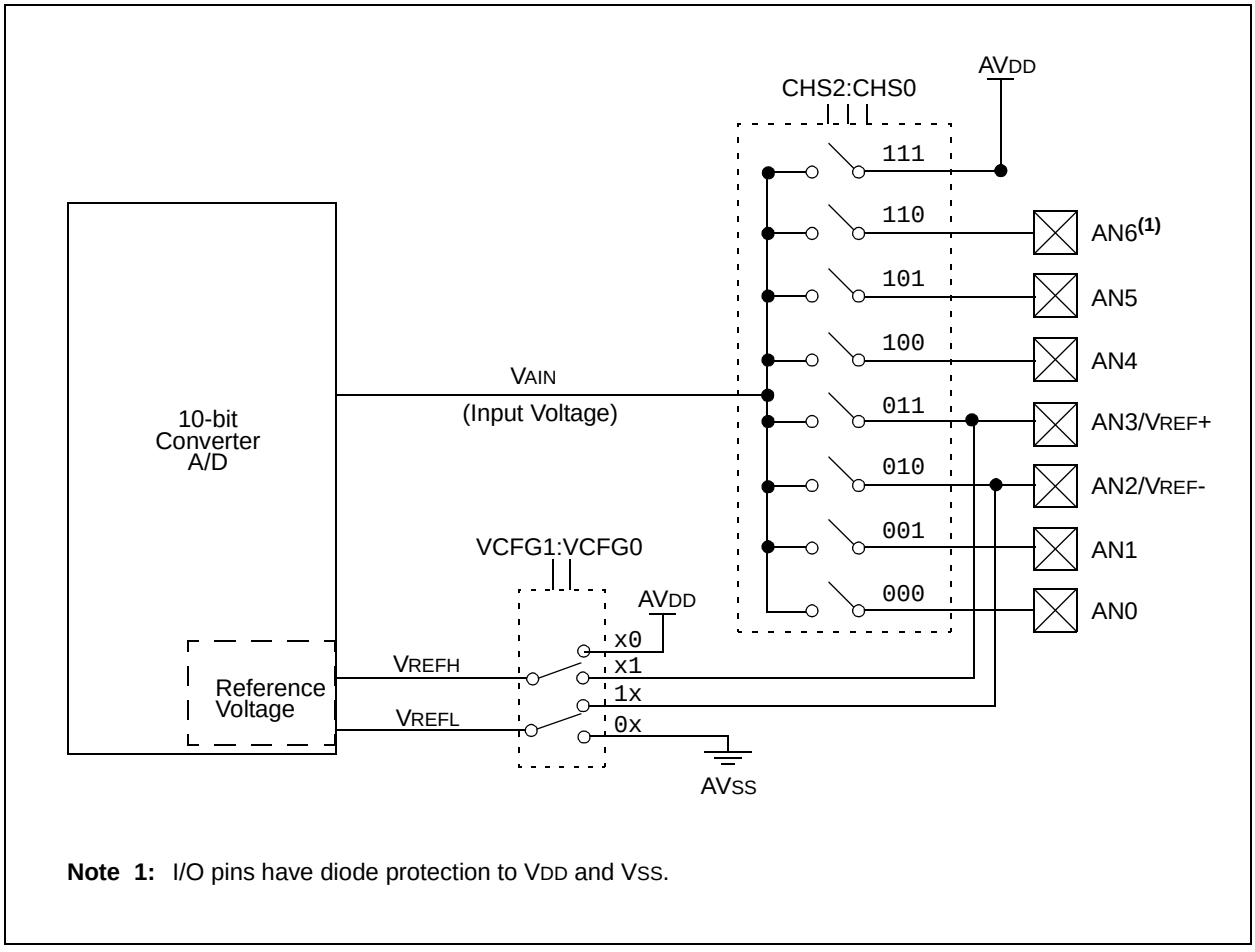
The A/D converter has a unique feature of being able to operate while the device is in Sleep mode. To operate in Sleep, the A/D conversion clock must be derived from the A/D's internal RC oscillator.

The output of the sample and hold is the input into the converter, which generates the result via successive approximation.

A device Reset forces all registers to their Reset state. This forces the A/D module to be turned off and any conversion in progress is aborted.

Each port pin associated with the A/D converter can be configured as an analog input, or as a digital I/O. The ADRESH and ADRESL registers contain the result of the A/D conversion. When the A/D conversion is complete, the result is loaded into the ADRESH/ADRESL registers, the GO/DONE bit (ADCON0 register) is cleared and A/D Interrupt Flag bit, ADIF, is set. The block diagram of the A/D module is shown in Figure 17-1.

**FIGURE 17-1: A/D BLOCK DIAGRAM**




**Note 1:** I/O pins have diode protection to $V_{DD}$ and $V_{SS}$.

The value in the ADRESH/ADRESL registers is not modified for a Power-on Reset. The ADRESH/ADRESL registers will contain unknown data after a Power-on Reset.

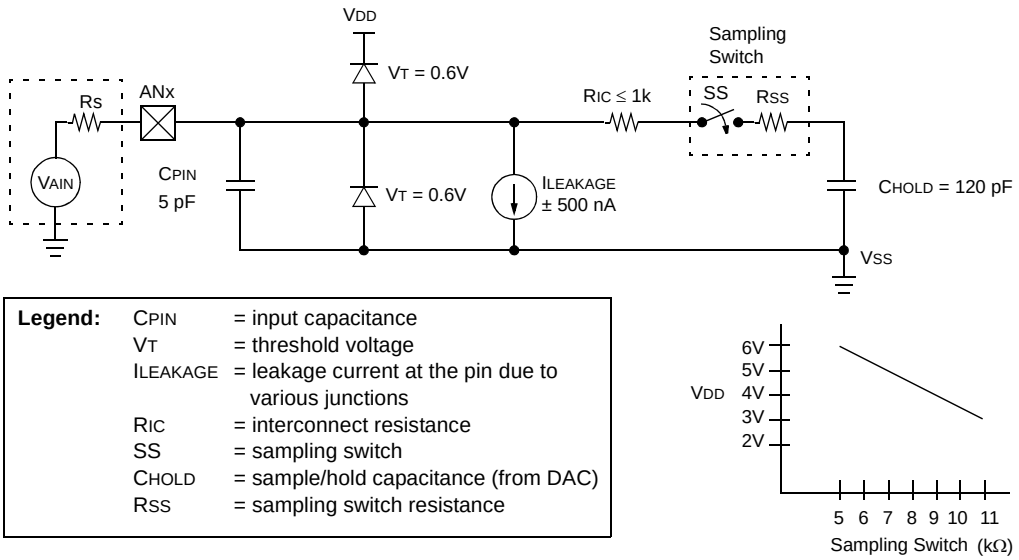
After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as an input. To determine acquisition time, see **Section 17.1 "A/D Acquisition Requirements"**. After this acquisition time has elapsed, the A/D conversion can be started. An acquisition time can be programmed to occur between setting the GO/DONE bit and the actual start of the conversion.

To do an A/D Conversion:

1. Configure the A/D module:
   - Configure analog pins, voltage reference and digital I/O (ADCON1)
   - Select A/D input channel (ADCON0)
   - Select A/D acquisition time (ADCON2)
   - Select A/D conversion clock (ADCON2)
   - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
   - Clear ADIF bit
   - Set ADIE bit
   - Set GIE bit
3. Wait the required acquisition time (if required).
4. Start conversion:
   - Set GO/DONE bit (ADCON0 register)
5. Wait for A/D conversion to complete, by either:
   - Polling for the GO/DONE bit to be cleared

   OR
   - Waiting for the A/D interrupt
6. Read A/D Result registers (ADRESH:ADRESL); clear bit, ADIF, if required.
7. For the next conversion, go to step 1 or step 2, as required. The A/D conversion time per bit is defined as TAD. A minimum wait of 2 TAD is required before the next acquisition starts.

**FIGURE 17-2: ANALOG INPUT MODEL**





© 2002-2015 Microchip Technology Inc.

## 17.1 A/D Acquisition Requirements

For the A/D converter to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The analog input model is shown in Figure 17-2. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD). The source impedance affects the offset voltage at the analog input (due to pin leakage current). **The maximum recommended impedance for analog sources is 2.5 kΩ.** After the analog input channel is selected (changed), the channel must be sampled for at least the minimum acquisition time before starting a conversion.

| **Note:** | When the conversion is started, the holding capacitor is disconnected from the input pin. |
|---|---|

To calculate the minimum acquisition time, Equation 17-1 may be used. This equation assumes that 1/2 LSb error is used (1024 steps for the A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

Example 17-1 shows the calculation of the minimum required acquisition time, TACQ. This calculation is based on the following application system assumptions:

| | | |
|---|---|---|
| CHOLD | = | 120 pF |
| Rs | = | 2.5 kΩ |
| Conversion Error | ≤ | 1/2 LSb |
| VDD | = | 5V → RSS = 7 kΩ |
| Temperature | = | 50°C (system max.) |
| VHOLD | = | 0V @ time = 0 |

## 17.2 A/D VREF+ and VREF- References

If external voltage references are used instead of the internal AVDD and AVSS sources, the source impedance of the VREF+ and VREF- voltage sources must be considered. During acquisition, currents supplied by these sources are insignificant. However, during conversion, the A/D module sinks and sources current through the reference sources.

In order to maintain the A/D accuracy, the voltage reference source impedances should be kept low to reduce voltage changes. These voltage changes occur as reference currents flow through the reference source impedance. **The maximum recommended impedance of the VREF+ and VREF- external reference voltage sources is 250Ω.**

### EQUATION 17-1: ACQUISITION TIME

$$T_{ACQ} = \text{Amplifier Settling Time} + \text{Holding Capacitor Charging Time} + \text{Temperature Coefficient}$$
$$= T_{AMP} + T_C + T_{COFF}$$

### EQUATION 17-2: A/D MINIMUM CHARGING TIME

$$V_{HOLD} = (\Delta V_{REF} - (\Delta V_{REF}/2048)) \cdot (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))})$$

or

$$T_C = -(C_{HOLD})(R_{IC} + R_{SS} + R_S) \ln(1/2048)$$

### EXAMPLE 17-1: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME

$T_{ACQ} = T_{AMP} + T_C + T_{COFF}$

$T_{AMP} = 5\ \mu s$

$T_{COFF} = (\text{Temp} - 25^oC)(0.05\ \mu s/^oC)$

$(50^oC - 25^oC)(0.05\ \mu s/^oC)$

$1.25\ \mu s$

Temperature coefficient is only required for temperatures > 25ºC. Below 25ºC, $T_{COFF} = 0\ \mu s$.

$T_C = -(C_{HOLD})(R_{IC} + R_{SS} + R_S) \ln(1/2047)\ \mu s$

$-(120\ pF)(1\ k\Omega + 7\ k\Omega + 2.5\ k\Omega) \ln(0.0004883)\ \mu s$

$9.61\ \mu s$

$T_{ACQ} = 5\ \mu s + 1.25\ \mu s + 9.61\ \mu s$

$12.86\ \mu s$

## 17.3 Selecting and Configuring Automatic Acquisition Time

The ADCON2 register allows the user to select an acquisition time that occurs each time the GO/DONE bit is set.

When the GO/DONE bit is set, sampling is stopped and a conversion begins. The user is responsible for ensuring the required acquisition time has passed between selecting the desired input channel and setting the GO/DONE bit. This occurs when the ACQT2:ACQT0 bits (ADCON2<5:3>) remain in their Reset state ('000') and is compatible with devices that do not offer programmable acquisition times.

If desired, the ACQT bits can be set to select a programmable acquisition time for the A/D module. When the GO/DONE bit is set, the A/D module continues to sample the input for the selected acquisition time, then automatically begins a conversion. Since the acquisition time is programmed, there may be no need to wait for an acquisition time between selecting a channel and setting the GO/DONE bit.

In either case, when the conversion is completed, the GO/DONE bit is cleared, the ADIF flag is set and the A/D begins sampling the currently selected channel again. If an acquisition time is programmed, there is nothing to indicate if the acquisition time has ended or if the conversion has begun.

## 17.4 Selecting the A/D Conversion Clock

The A/D conversion time per bit is defined as TAD. The A/D conversion requires 11 TAD per 10-bit conversion. The source of the A/D conversion clock is software selectable. There are seven possible options for TAD:

- 2 TOSC
- 4 TOSC
- 8 TOSC
- 16 TOSC
- 32 TOSC
- 64 TOSC
- Internal RC oscillator

For correct A/D conversions, the A/D conversion clock (TAD) must be as short as possible, but greater than the minimum TAD (approximately 2 μs, see parameter 130 for more information).

Table 17-1 shows the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

**TABLE 17-1: TAD VS. DEVICE OPERATING FREQUENCIES**

| AD Clock Source (TAD) | | Maximum Device Frequency | |
|---|---|---|---|
| **Operation** | **ADCS2:ADCS0** | **PIC18F1220/1320** | **PIC18LF1220/1320(4)** |
| 2 TOSC | 000 | 1.25 MHz | 666 kHz |
| 4 TOSC | 100 | 2.50 MHz | 1.33 MHz |
| 8 TOSC | 001 | 5.00 MHz | 2.66 MHz |
| 16 TOSC | 101 | 10.0 MHz | 5.33 MHz |
| 32 TOSC | 010 | 20.0 MHz | 10.65 MHz |
| 64 TOSC | 110 | 40.0 MHz | 21.33 MHz |
| RC(3) | x11 | 1.00 MHz(1) | 1.00 MHz(2) |

**Note 1:** The RC source has a typical TAD time of 4 μs.

**2:** The RC source has a typical TAD time of 6 μs.

**3:** For device frequencies above 1 MHz, the device must be in Sleep for the entire conversion or the A/D accuracy may be out of specification.

**4:** Low-power devices only.


© 2002-2015 Microchip Technology Inc.

## 17.5 Operation in Low-Power Modes

The selection of the automatic acquisition time and the A/D conversion clock is determined, in part, by the low-power mode clock source and frequency while in a low-power mode.

If the A/D is expected to operate while the device is in a low-power mode, the ACQT2:ACQT0 and ADCS2:ADCS0 bits in ADCON2 should be updated in accordance with the low-power mode clock that will be used. After the low-power mode is entered (either of the Run modes), an A/D acquisition or conversion may be started. Once an acquisition or conversion is started, the device should continue to be clocked by the same low-power mode clock source until the conversion has been completed. If desired, the device may be placed into the corresponding low-power (ANY)_IDLE mode during the conversion.

If the low-power mode clock frequency is less than 1 MHz, the A/D RC clock source should be selected.

Operation in the Low-Power Sleep mode requires the A/D RC clock to be selected. If bits, ACQT2:ACQT0, are set to '000' and a conversion is started, the conversion will be delayed one instruction cycle to allow execution of the SLEEP instruction and entry to Low-Power Sleep mode. The IDLEN and SCS bits in the OSCCON register must have already been cleared prior to starting the conversion.

## 17.6 Configuring Analog Port Pins

The ADCON1, TRISA and TRISB registers all configure the A/D port pins. The port pins needed as analog inputs must have their corresponding TRIS bits set (input). If the TRIS bit is cleared (output), the digital output level ($V_{OH}$ or $V_{OL}$) will be converted.

The A/D operation is independent of the state of the CHS2:CHS0 bits and the TRIS bits.

**Note 1:** 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 convert an analog input. Analog levels on a digitally configured input will be accurately converted.

**2:** Analog levels on any pin defined as a digital input may cause the digital input buffer to consume current out of the device's specification limits.

## 17.7 A/D Conversions

Figure 17-3 shows the operation of the A/D converter after the GO bit has been set and the ACQT2:ACQT0 bits are cleared. A conversion is started after the following instruction to allow entry into Low-Power Sleep mode before the conversion begins.

Figure 17-4 shows the operation of the A/D converter after the GO bit has been set and the ACQT2:ACQT0 bits are set to '010' and selecting a 4 TAD acquisition time before the conversion starts.

Clearing the GO/$\overline{\text{DONE}}$ bit during a conversion will abort the current conversion. The A/D Result register pair will NOT be updated with the partially completed A/D conversion sample. This means the ADRESH:ADRESL registers will continue to contain the value of the last completed conversion (or the last value written to the ADRESH:ADRESL registers).

After the A/D conversion is completed or aborted, a 2 TAD wait is required before the next acquisition can be started. After this wait, acquisition on the selected channel is automatically started.

| **Note:** | The GO/$\overline{\text{DONE}}$ bit should **NOT** be set in the same instruction that turns on the A/D. |
|---|---|

**FIGURE 17-3: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 000, TACQ = 0)**

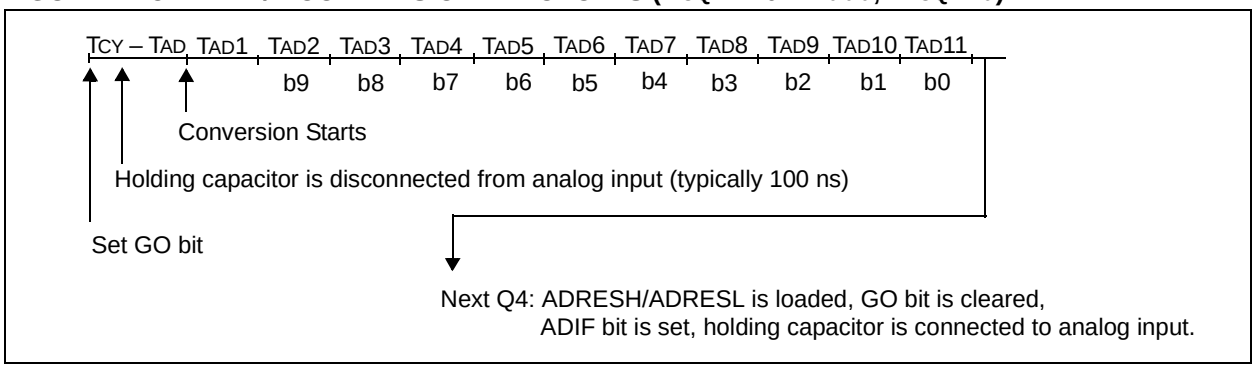
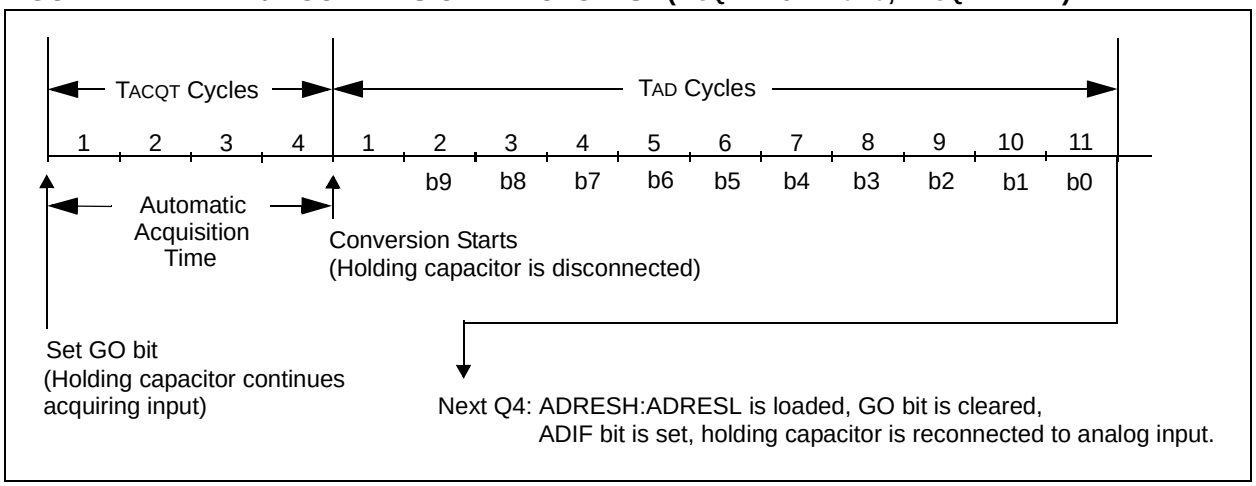



**FIGURE 17-4: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 010, TACQ = 4 TAD)**





© 2002-2015 Microchip Technology Inc.

## 17.8 Use of the CCP1 Trigger

An A/D conversion can be started by the "special event trigger" of the CCP1 module. This requires that the CCP1M3:CCP1M0 bits (CCP1CON<3:0>) be programmed as '1011' and that the A/D module is enabled (ADON bit is set). When the trigger occurs, the GO/DONE bit will be set, starting the A/D acquisition and conversion and the Timer1 (or Timer3) counter will be reset to zero. Timer1 (or Timer3) is reset to automatically repeat the A/D acquisition period with minimal software overhead (moving ADRESH/ADRESL to the desired location). The appropriate analog input channel must be selected and the minimum acquisition period is either timed by the user, or an appropriate TACQ time selected before the "special event trigger" sets the GO/DONE bit (starts a conversion).

If the A/D module is not enabled (ADON is cleared), the "special event trigger" will be ignored by the A/D module, but will still reset the Timer1 (or Timer3) counter.

**TABLE 17-2: SUMMARY OF A/D REGISTERS**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 0000 | 0000 0000 |
| PIR1 | — | ADIF | RCIF | TXIF | — | CCP1IF | TMR2IF | TMR1IF | -000 -000 | -000 -000 |
| PIE1 | — | ADIE | RCIE | TXIE | — | CCP1IE | TMR2IE | TMR1IE | -000 -000 | -000 -000 |
| IPR1 | — | ADIP | RCIP | TXIP | — | CCP1IP | TMR2IP | TMR1IP | -111 -111 | -111 -111 |
| PIR2 | OSCFIF | — | — | EEIF | — | LVDIF | TMR3IF | — | 0--0 -00- | 0--0 -00- |
| PIE2 | OSCFIE | — | — | EEIE | — | LVDIE | TMR3IE | — | 0--0 -00- | 0--0 -00- |
| IPR2 | OSCFIP | — | — | EEIP | — | LVDIP | TMR3IP | — | 1--1 -11- | 1--1 -11- |
| ADRESH | A/D Result Register High Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| ADRESL | A/D Result Register Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| ADCON0 | VCFG1 | VCFG0 | — | CHS2 | CHS1 | CHS0 | GO/DONE | ADON | 00-0 0000 | 00-0 0000 |
| ADCON1 | — | PCFG6 | PCFG5 | PCFG4 | PCFG3 | PCFG2 | PCFG1 | PCFG0 | -000 0000 | -000 0000 |
| ADCON2 | ADFM | — | ACQT2 | ACQT1 | ACQT0 | ADCS2 | ADCS1 | ADCS0 | 0-00 0000 | 0-00 0000 |
| PORTA | RA7(3) | RA6(2) | RA5(1) | RA4 | RA3 | RA2 | RA1 | RA0 | qq0x 0000 | uu0u 0000 |
| TRISA | TRISA7(3) | TRISA6(2) | — | PORTA Data Direction Register | | | | | qq-1 1111 | 11-1 1111 |
| PORTB | Read PORTB pins, Write LATB Latch | | | | | | | | xxxx xxxx | uuuu uuuu |
| TRISB | PORTB Data Direction Register | | | | | | | | 1111 1111 | 1111 1111 |
| LATB | PORTB Output Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |

**Legend:** x = unknown, u = unchanged, q = depends on CONFIG1H<3:0>, – = unimplemented, read as '0'. Shaded cells are not used for A/D conversion.

**Note 1:** RA5 port bit is available only as an input pin when the MCLRE bit in the Configuration register is '0'.

**2:** RA6 and TRISA6 are available only when the primary oscillator mode selection offers RA6 as a port pin; otherwise, RA6 always reads '0', TRISA6 always reads '1' and writes to both are ignored (see CONFIG1H<3:0>).

**3:** RA7 and TRISA7 are available only when the internal RC oscillator is configured as the primary oscillator in CONFIG1H<3:0>; otherwise, RA7 always reads '0', TRISA7 always reads '1' and writes to both are ignored.

## 18.0 LOW-VOLTAGE DETECT

In many applications, the ability to determine if the device voltage ($V_{DD}$) is below a specified voltage level is a desirable feature. A window of operation for the application can be created, where the application software can do “housekeeping tasks”, before the device voltage exits the valid operating range. This can be done using the Low-Voltage Detect module.

This module is a software programmable circuitry, where a device voltage trip point can be specified. When the voltage of the device becomes lower then the specified point, an interrupt flag is set. If the interrupt is enabled, the program execution will branch to the interrupt vector address and the software can then respond to that interrupt source.

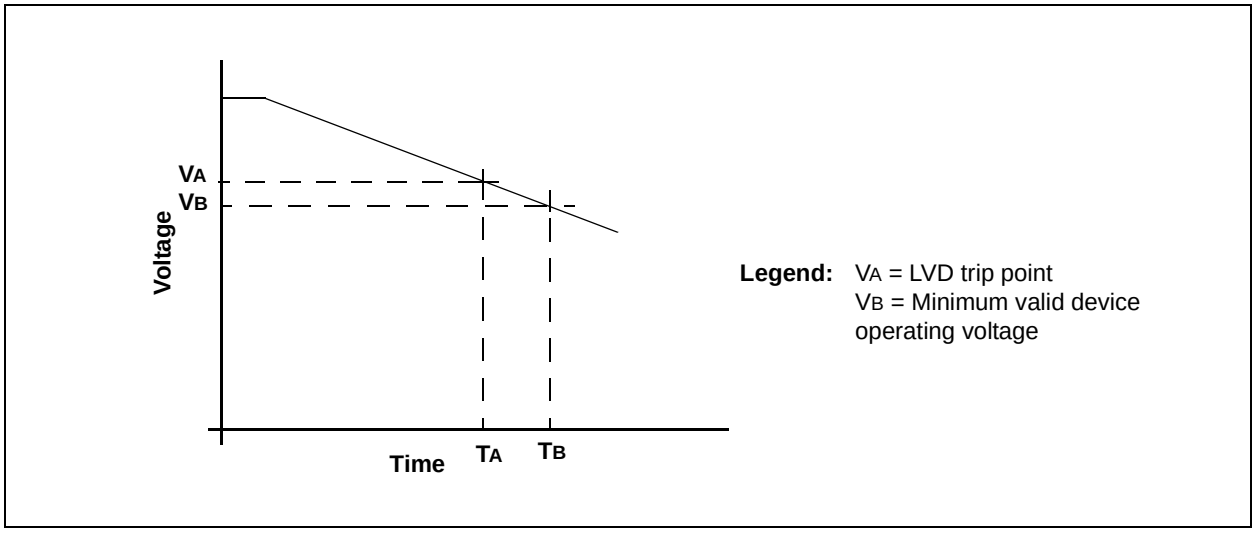
The Low-Voltage Detect circuitry is completely under software control. This allows the circuitry to be turned off by the software, which minimizes the current consumption for the device.

Figure 18-1 shows a possible application voltage curve (typically for batteries). Over time, the device voltage decreases. When the device voltage equals voltage $V_A$, the LVD logic generates an interrupt. This occurs at time $T_A$. The application software then has the time, until the device voltage is no longer in valid operating range, to shut down the system. Voltage point $V_B$ is the minimum valid operating voltage specification. This occurs at time $T_B$. The difference, $T_B - T_A$, is the total time for shutdown.

The block diagram for the LVD module is shown in Figure 18-2 (following page). A comparator uses an internally generated reference voltage as the set point. When the selected tap output of the device voltage crosses the set point (is lower than), the LVDIF bit is set.

Each node in the resistor divider represents a “trip point” voltage. The “trip point” voltage is the minimum supply voltage level at which the device can operate before the LVD module asserts an interrupt. When the supply voltage is equal to the trip point, the voltage tapped off of the resistor array is equal to the 1.2V internal reference voltage generated by the voltage reference module. The comparator then generates an interrupt signal setting the LVDIF bit. This voltage is software programmable to any one of 16 values (see Figure 18-2). The trip point is selected by programming the LVDL3:LVDL0 bits (LVDCON<3:0>).

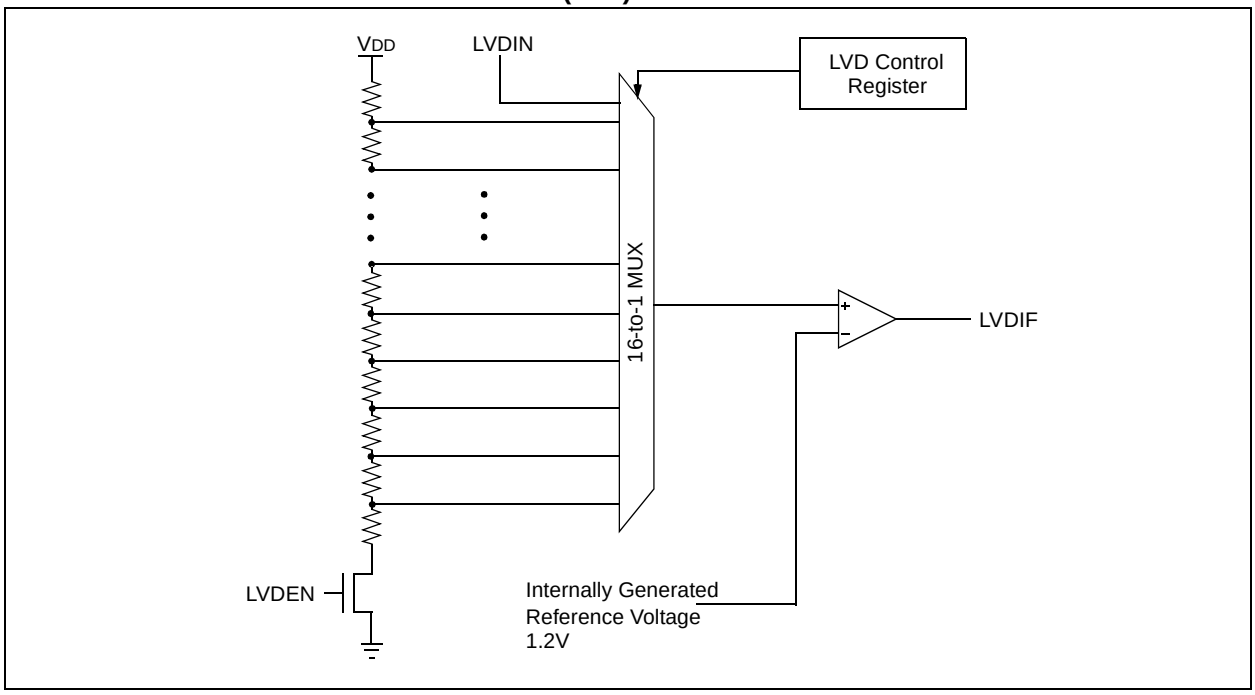
**FIGURE 18-1: TYPICAL LOW-VOLTAGE DETECT APPLICATION**





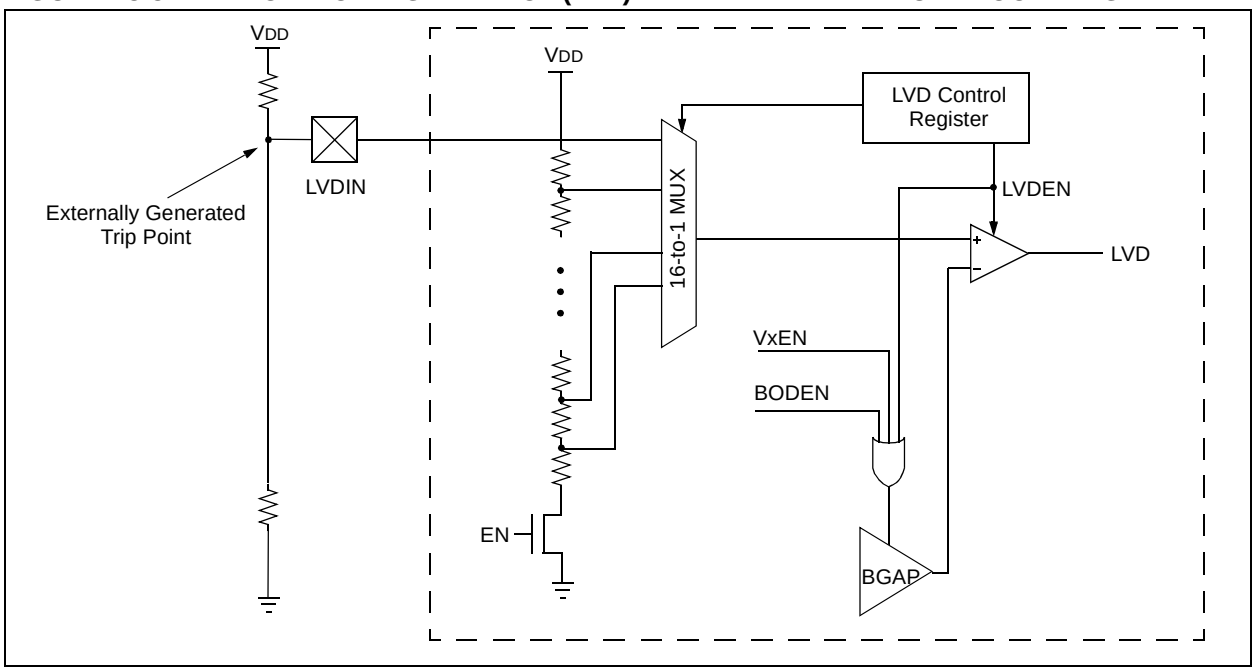
© 2002-2015 Microchip Technology Inc.

**FIGURE 18-2: LOW-VOLTAGE DETECT (LVD) BLOCK DIAGRAM**




The LVD module has an additional feature that allows the user to supply the trip voltage to the module from an external source. This mode is enabled when bits, LVDL3:LVDL0, are set to '1111'. In this state, the comparator input is multiplexed from the external input pin, LVDIN (Figure 18-3). This gives users flexibility, because it allows them to configure the Low-Voltage Detect interrupt to occur at any voltage in the valid operating range.

**FIGURE 18-3: LOW-VOLTAGE DETECT (LVD) WITH EXTERNAL INPUT BLOCK DIAGRAM**



## 18.1 Control Register

The Low-Voltage Detect Control register controls the operation of the Low-Voltage Detect circuitry.

**REGISTER 18-1: LVDCON: LOW-VOLTAGE DETECT CONTROL REGISTER**

| U-0 | U-0 | R-0/0 | R/W-0/0 | R/W-0/0 | R/W-1/1 | R/W-0/0 | R/W-1/1 |
|---|---|---|---|---|---|---|---|
| — | — | IRVST | LVDEN | LVDL3 | LVDL2 | LVDL1 | LVDL0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7-6 **Unimplemented:** Read as ‘0’

bit 5 **IRVST:** Internal Reference Voltage Stable Flag bit
1 = Indicates that the Low-Voltage Detect logic will generate the interrupt flag at the specified voltage range
0 = Indicates that the Low-Voltage Detect logic will not generate the interrupt flag at the specified voltage range and the LVD interrupt should not be enabled

bit 4 **LVDEN:** Low-Voltage Detect Power Enable bit
1 = Enables LVD, powers up LVD circuit
0 = Disables LVD, powers down LVD circuit

bit 3-0 **LVDL<3:0>:** Low-Voltage Detection Limit bits(1)
1111 = External analog input is used (input comes from the LVDIN pin)
1110 = 4.04V-5.15V
1101 = 3.76V-4.79V
1100 = 3.58V-4.56V
1011 = 3.41V-4.34V
1010 = 3.23V-4.11V
1001 = 3.14V-4.00V
1000 = 2.96V-3.77V
0111 = 2.70V-3.43V
0110 = 2.53V-3.21V
0101 = 2.43V-3.10V
0100 = 2.25V-2.86V
0011 = 2.16V-2.75V
0010 = 1.99V-2.53V
0001 = Reserved
0000 = Reserved

**Note 1:** LVDL<3:0> modes, which result in a trip point below the valid operating voltage of the device, are not tested.


© 2002-2015 Microchip Technology Inc.

## 18.2 Operation

Depending on the power source for the device voltage, the voltage normally decreases relatively slowly. This means that the LVD module does not need to be constantly operating. To decrease the current requirements, the LVD circuitry only needs to be enabled for short periods, where the voltage is checked. After doing the check, the LVD module may be disabled.

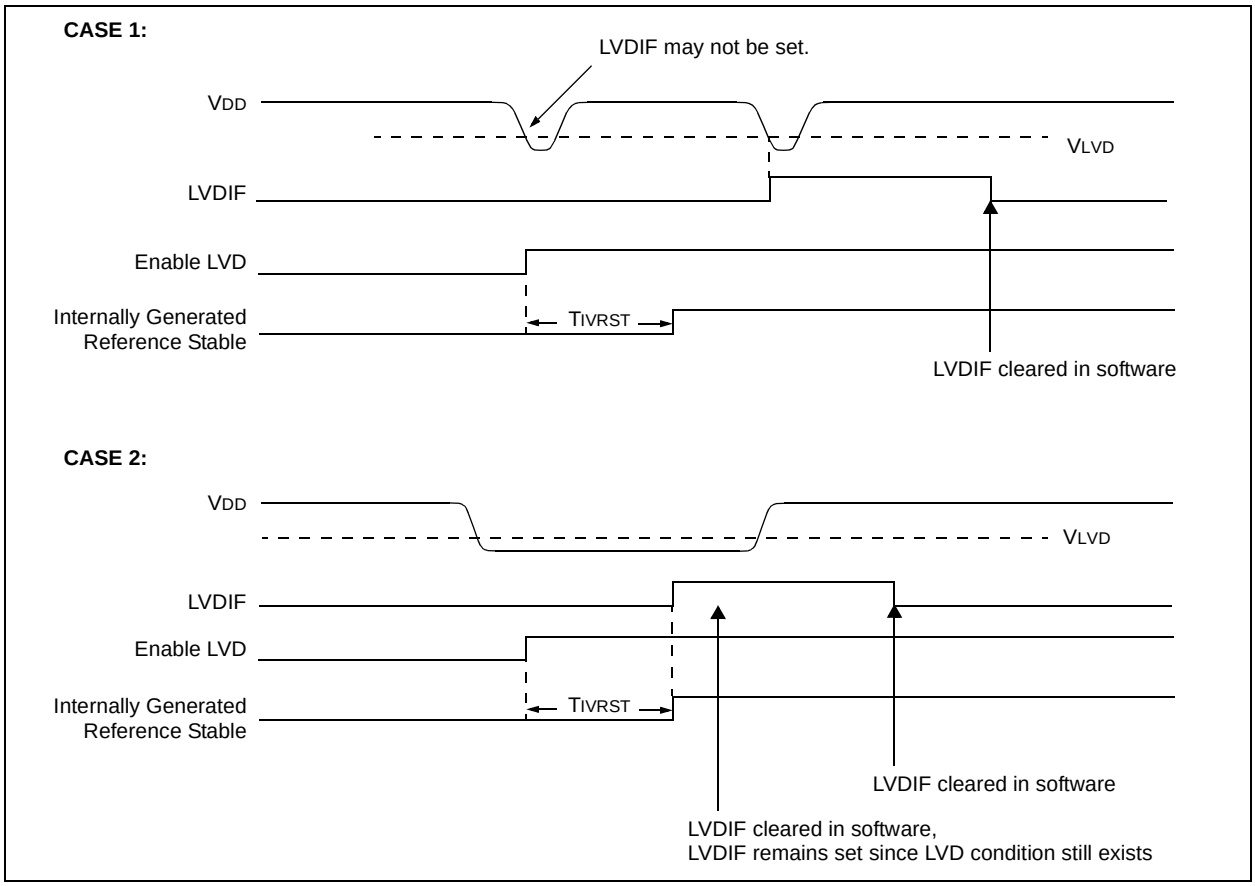
Each time that the LVD module is enabled, the circuitry requires some time to stabilize. After the circuitry has stabilized, all status flags may be cleared. The module will then indicate the proper state of the system.

The following steps are needed to set up the LVD module:

1. Write the value to the LVDL3:LVDL0 bits (LVDCON register), which selects the desired LVD trip point.
2. Ensure that LVD interrupts are disabled (the LVDIE bit is cleared or the GIE bit is cleared).
3. Enable the LVD module (set the LVDEN bit in the LVDCON register).
4. Wait for the LVD module to stabilize (the IRVST bit to become set).
5. Clear the LVD interrupt flag, which may have falsely become set, until the LVD module has stabilized (clear the LVDIF bit).
6. Enable the LVD interrupt (set the LVDIE and the GIE bits).

Figure 18-4 shows typical waveforms that the LVD module may be used to detect.

**FIGURE 18-4: LOW-VOLTAGE DETECT WAVEFORMS**



### 18.2.1 REFERENCE VOLTAGE SET POINT

The internal reference voltage of the LVD module may be used by other internal circuitry (the programmable Brown-out Reset). If these circuits are disabled (lower current consumption), the reference voltage circuit requires a time to become stable before a low-voltage condition can be reliably detected. This time is invariant of system clock speed. This start-up time is specified in electrical specification parameter 36. The low-voltage interrupt flag will not be enabled until a stable reference voltage is reached. Refer to the waveform in Figure 18-4.

### 18.2.2 CURRENT CONSUMPTION

When the module is enabled, the LVD comparator and voltage divider are enabled and will consume static current. The voltage divider can be tapped from multiple places in the resistor array. Total current consumption, when enabled, is specified in electrical specification parameter D022B.

## 18.3 Operation During Sleep

When enabled, the LVD circuitry continues to operate during Sleep. If the device voltage crosses the trip point, the LVDIF bit will be set and the device will wake-up from Sleep. Device execution will continue from the interrupt vector address if interrupts have been globally enabled.

## 18.4 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the LVD module to be turned off.


© 2002-2015 Microchip Technology Inc.

## 19.0 SPECIAL FEATURES OF THE CPU

PIC18F1220/1320 devices include several features intended to maximize system reliability, minimize cost through elimination of external components and offer code protection. These are:

- Oscillator Selection
- Resets:
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
  - Brown-out Reset (BOR)
- Interrupts
- Watchdog Timer (WDT)
- Fail-Safe Clock Monitor
- Two-Speed Start-up
- Code Protection
- ID Locations
- In-Circuit Serial Programming

Several oscillator options are available to allow the part to fit the application. The RC oscillator option saves system cost, while the LP crystal option saves power. These are discussed in detail in **Section 2.0 "Oscillator Configurations"**.

A complete discussion of device Resets and interrupts is available in previous sections of this data sheet.

In addition to their Power-up and Oscillator Start-up Timers provided for Resets, PIC18F1220/1320 devices have a Watchdog Timer, which is either permanently enabled via the Configuration bits, or software controlled (if configured as disabled).

The inclusion of an internal RC oscillator also provides the additional benefits of a Fail-Safe Clock Monitor (FSCM) and Two-Speed Start-up. FSCM provides for background monitoring of the peripheral clock and automatic switchover in the event of its failure. Two-Speed Start-up enables code to be executed almost immediately on start-up, while the primary clock source completes its start-up delays.

All of these features are enabled and configured by setting the appropriate Configuration register bits.

### 19.1 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 starting at program memory location 300000h.

The user will note that address 300000h is beyond the user program memory space. In fact, it belongs to the configuration memory space (300000h-3FFFFFh), which can only be accessed using table reads and table writes.

Programming the Configuration registers is done in a manner similar to programming the Flash memory. The EECON1 register WR bit starts a self-timed write to the Configuration register. In normal operation mode, a TBLWT instruction, with the TBLPTR pointing to the Configuration register, sets up the address and the data for the Configuration register write. Setting the WR bit starts a long write to the Configuration register. The Configuration registers are written a byte at a time. To write or erase a configuration cell, a TBLWT instruction can write a '1' or a '0' into the cell. For additional details on Flash programming, refer to **Section 6.5 "Writing to Flash Program Memory"**.

**TABLE 19-1: CONFIGURATION BITS AND DEVICE IDS**

| File Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Default/ Unprogrammed Value |
|---|---|---|---|---|---|---|---|---|---|---|
| 300001h | CONFIG1H | IESO | FSCM | — | — | FOSC3 | FOSC2 | FOSC1 | FOSC0 | 11-- 1111 |
| 300002h | CONFIG2L | — | — | — | — | BORV1 | BORV0 | BOR | $\overline{\text{PWRTEN}}$ | ---- 1111 |
| 300003h | CONFIG2H | — | — | — | WDTPS3 | WDTPS2 | WDTPS1 | WDTPS0 | WDT | ---1 1111 |
| 300005h | CONFIG3H | MCLRE | — | — | — | — | — | — | — | 1--- ---- |
| 300006h | CONFIG4L | $\overline{\text{DEBUG}}$ | — | — | — | — | LVP | — | STVR | 1--- -1-1 |
| 300008h | CONFIG5L | — | — | — | — | — | — | CP1 | CP0 | ---- --11 |
| 300009h | CONFIG5H | CPD | CPB | — | — | — | — | — | — | 11-- ---- |
| 30000Ah | CONFIG6L | — | — | — | — | — | — | WRT1 | WRT0 | ---- --11 |
| 30000Bh | CONFIG6H | WRTD | WRTB | WRTC | — | — | — | — | — | 111- ---- |
| 30000Ch | CONFIG7L | — | — | — | — | — | — | EBTR1 | EBTR0 | ---- --11 |
| 30000Dh | CONFIG7H | — | EBTRB | — | — | — | — | — | — | -1-- ---- |
| 3FFFFEh | DEVID1(1) | DEV2 | DEV1 | DEV0 | REV4 | REV3 | REV2 | REV1 | REV0 | xxxx xxxx(1) |
| 3FFFFFh | DEVID2(1) | DEV10 | DEV9 | DEV8 | DEV7 | DEV6 | DEV5 | DEV4 | DEV3 | 0000 0111 |

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

**Note 1:** See Register 19-12 for DEVID1 values. DEVID registers are read-only and cannot be programmed by the user.

**REGISTER 19-1: CONFIG1H: CONFIGURATION REGISTER 1 HIGH (BYTE ADDRESS 300001h)**

| R/P-1 | R/P-1 | U-0 | U-0 | R/P-1 | R/P-1 | R/P-1 | R/P-1 |
|---|---|---|---|---|---|---|---|
| IESO | FSCM | — | — | FOSC3 | FOSC2 | FOSC1 | FOSC0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | P = Programmable bit |

bit 7 **IESO:** Internal External Switchover bit
1 = Internal External Switchover mode enabled
0 = Internal External Switchover mode disabled

bit 6 **FSCM:** Fail-Safe Clock Monitor Enable bit
1 = Fail-Safe Clock Monitor enabled
0 = Fail-Safe Clock Monitor disabled

bit 5-4 **Unimplemented:** Read as ‘0’

bit 3-0 **FOSC<3:0>:** Oscillator Selection bits
11xx = External RC oscillator, CLKO function on RA6
1001 = Internal RC oscillator, CLKO function on RA6 and port function on RA7
1000 = Internal RC oscillator, port function on RA6 and port function on RA7
0111 = External RC oscillator, port function on RA6
0110 = HS oscillator, PLL enabled (clock frequency = 4 x FOSC1)
0101 = EC oscillator, port function on RA6
0100 = EC oscillator, CLKO function on RA6
0010 = HS oscillator
0001 = XT oscillator
0000 = LP oscillator


© 2002-2015 Microchip Technology Inc.

**REGISTER 19-2: CONFIG2L: CONFIGURATION REGISTER 2 LOW (BYTE ADDRESS 300002h)**

| U-0 | U-0 | U-0 | U-0 | R/P-1 | R/P-1 | R/P-1 | R/P-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | BORV1 | BORV0 | BOR(1) | $\overline{\text{PWRTEN}}$(1) |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | P = Programmable bit |

bit 7-4 **Unimplemented:** Read as ‘0’

bit 3-2 **BORV<1:0>:** Brown-out Reset Voltage bits
- 11 = Reserved
- 10 = $V_{BOR}$ set to 2.7V
- 01 = $V_{BOR}$ set to 4.2V
- 00 = $V_{BOR}$ set to 4.5V

bit 1 **BOR:** Brown-out Reset Enable bit(1)
- 1 = Brown-out Reset enabled
- 0 = Brown-out Reset disabled

bit 0 **$\overline{\text{PWRTEN}}$:** Power-up Timer Enable bit(1)
- 1 = PWRT disabled
- 0 = PWRT enabled

**Note 1:** The Power-up Timer is decoupled from Brown-out Reset, allowing these features to be independently controlled.

**REGISTER 19-3: CONFIG2H: CONFIGURATION REGISTER 2 HIGH (BYTE ADDRESS 300003h)**

| U-0 | U-0 | U-0 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | WDTPS3 | WDTPS2 | WDTPS1 | WDTPS0 | WDTEN |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | P = Programmable bit |

bit 7-5 **Unimplemented:** Read as ‘0’

bit 4-1 **WDTPS<3:0>:** Watchdog Timer Postscale Select bits

1111 = 1:32,768
1110 = 1:16,384
1101 = 1:8,192
1100 = 1:4,096
1011 = 1:2,048
1010 = 1:1,024
1001 = 1:512
1000 = 1:256
0111 = 1:128
0110 = 1:64
0101 = 1:32
0100 = 1:16
0011 = 1:8
0010 = 1:4
0001 = 1:2
0000 = 1:1

bit 0 **WDT:** Watchdog Timer Enable bit

1 = WDT enabled
0 = WDT disabled (control is placed on the SWDTEN bit)


© 2002-2015 Microchip Technology Inc.

**REGISTER 19-4: CONFIG3H: CONFIGURATION REGISTER 3 HIGH (BYTE ADDRESS 300005h)**

| R/P-1 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| MCLRE | — | — | — | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | P = Programmable bit |

bit 7 **MCLRE:** $\overline{\text{MCLR}}$ Pin Enable bit
1 = $\overline{\text{MCLR}}$ pin enabled, RA5 input pin disabled
0 = RA5 input pin enabled, $\overline{\text{MCLR}}$ disabled

bit 6-0 **Unimplemented:** Read as '0'

**REGISTER 19-5: CONFIG4L: CONFIGURATION REGISTER 4 LOW (BYTE ADDRESS 300006h)**

| R/P-1 | U-0 | U-0 | U-0 | U-0 | R/P-1 | U-0 | R/P-1 |
|---|---|---|---|---|---|---|---|
| $\overline{\text{DEBUG}}$ | — | — | — | — | LVP | — | STVR |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | P = Programmable bit |

bit 7 **$\overline{\text{DEBUG}}$:** Background Debugger Enable bit (see note)
1 = Background debugger disabled, RB6 and RB7 configured as general purpose I/O pins
0 = Background debugger enabled, RB6 and RB7 are dedicated to In-Circuit Debug

bit 6-3 **Unimplemented:** Read as '0'

bit 2 **LVP:** Low-Voltage ICSP Enable bit
1 = Low-Voltage ICSP enabled
0 = Low-Voltage ICSP disabled

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

bit 0 **STVR:** Stack Full/Underflow Reset Enable bit
1 = Stack full/underflow will cause Reset
0 = Stack full/underflow will not cause Reset

**Note:** The Timer1 oscillator shares the T1OSI and T1OSO pins with the PGD and PGC pins used for programming and debugging.

When using the Timer1 oscillator, In-Circuit Serial Programming (ICSP) may not function correctly (high voltage or low voltage), or the In-Circuit Debugger (ICD) may not communicate with the controller. As a result of using either ICSP or ICD, the Timer1 crystal may be damaged.

If ICSP or ICD operations are required, the crystal should be disconnected from the circuit (disconnect either lead) or installed after programming. The oscillator loading capacitors may remain in-circuit during ICSP or ICD operation.

**REGISTER 19-6: CONFIG5L: CONFIGURATION REGISTER 5 LOW (BYTE ADDRESS 300008h)**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/C-1 | R/C-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | CP1 | CP0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | C = Clearable bit |

bit 7-2 **Unimplemented:** Read as ‘0’

bit 1 **CP1:** Code Protection bit (PIC18F1320)
1 = Block 1 (001000-001FFFh) not code-protected
0 = Block 1 (001000-001FFFh) code-protected

bit 0 **CP0:** Code Protection bit (PIC18F1320)
1 = Block 0 (00200-000FFFh) not code-protected
0 = Block 0 (00200-000FFFh) code-protected

bit 1 **CP1:** Code Protection bit (PIC18F1220)
1 = Block 1 (000800-000FFFh) not code-protected
0 = Block 1 (000800-000FFFh) code-protected

bit 0 **CP0:** Code Protection bit (PIC18F1220)
1 = Block 0 (000200-0007FFh) not code-protected
0 = Block 0 (000200-0007FFh) code-protected

**REGISTER 19-7: CONFIG5H: CONFIGURATION REGISTER 5 HIGH (BYTE ADDRESS 300009h)**

| R/C-1 | R/C-1 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| CPD | CPB | — | — | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | C = Clearable bit |

bit 7 **CPD:** Data EEPROM Code Protection bit
1 = Data EEPROM not code-protected
0 = Data EEPROM code-protected

bit 6 **CPB:** Boot Block Code Protection bit
1 = Boot Block (000000-0001FFh) not code-protected
0 = Boot Block (000000-0001FFh) code-protected

bit 5-0 **Unimplemented:** Read as ‘0’


© 2002-2015 Microchip Technology Inc.

**REGISTER 19-8: CONFIG6L: CONFIGURATION REGISTER 6 LOW (BYTE ADDRESS 30000Ah)**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/P-1 | R/P-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | WRT1 | WRT0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | P = Programmable bit |

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

bit 1 **WRT1:** Write Protection bit (PIC18F1320)
1 = Block 1 (001000-001FFFh) not write-protected
0 = Block 1 (001000-001FFFh) write-protected

bit 0 **WRT0:** Write Protection bit (PIC18F1320)
1 = Block 0 (00200-000FFFh) not write-protected
0 = Block 0 (00200-000FFFh) write-protected

bit 1 **WRT1:** Write Protection bit (PIC18F1220)
1 = Block 1 (000800-000FFFh) not write-protected
0 = Block 1 (000800-000FFFh) write-protected

bit 0 **WRT0:** Write Protection bit (PIC18F1220)
1 = Block 0 (000200-0007FFh) not write-protected
0 = Block 0 (000200-0007FFh) write-protected

**REGISTER 19-9: CONFIG6H: CONFIGURATION REGISTER 6 HIGH (BYTE ADDRESS 30000Bh)**

| R/P-1 | R/P-1 | R-1 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| WRTD | WRTB | WRTC(1) | — | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | P = Programmable bit |

bit 7 **WRTD:** Data EEPROM Write Protection bit
1 = Data EEPROM not write-protected
0 = Data EEPROM write-protected

bit 6 **WRTB:** Boot Block Write Protection bit
1 = Boot Block (000000-0001FFh) not write-protected
0 = Boot Block (000000-0001FFh) write-protected

bit 5 **WRTC:** Configuration Register Write Protection bit(1)
1 = Configuration registers (300000-3000FFh) not write-protected
0 = Configuration registers (300000-3000FFh) write-protected

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

**Note 1:** This bit is read-only in normal execution mode; it can be written only in Program mode.

**REGISTER 19-10: CONFIG7L: CONFIGURATION REGISTER 7 LOW (BYTE ADDRESS 30000Ch)**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/P-1 | R/P-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | EBTR1 | EBTR0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | P = Programmable bit |

bit 7-2 **Unimplemented:** Read as ‘0’

bit 1 **EBTR1:** Table Read Protection bit (PIC18F1320)
1 = Block 1 (001000-001FFFh) not protected from table reads executed in other blocks
0 = Block 1 (001000-001FFFh) protected from table reads executed in other blocks

bit 0 **EBTR0:** Table Read Protection bit (PIC18F1320)
1 = Block 0 (00200-000FFFh) not protected from table reads executed in other blocks
0 = Block 0 (00200-000FFFh) protected from table reads executed in other blocks

bit 1 **EBTR1:** Table Read Protection bit (PIC18F1220)
1 = Block 1 (000800-000FFFh) not protected from table reads executed in other blocks
0 = Block 1 (000800-000FFFh) protected from table reads executed in other blocks

bit 0 **EBTR0:** Table Read Protection bit (PIC18F1220)
1 = Block 0 (000200-0007FFh) not protected from table reads executed in other blocks
0 = Block 0 (000200-0007FFh) protected from table reads executed in other blocks

**REGISTER 19-11: CONFIG7H: CONFIGURATION REGISTER 7 HIGH (BYTE ADDRESS 30000Dh)**

| U-0 | R/P-1 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | EBTRB | — | — | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | P = Programmable bit |

bit 7 **Unimplemented:** Read as ‘0’

bit 6 **EBTRB:** Boot Block Table Read Protection bit
1 = Boot Block (000000-0001FFh) not protected from table reads executed in other blocks
0 = Boot Block (000000-0001FFh) protected from table reads executed in other blocks

bit 5-0 **Unimplemented:** Read as ‘0’


© 2002-2015 Microchip Technology Inc.

**REGISTER 19-12: DEVID1: DEVICE ID REGISTER 1 FOR PIC18F1220/1320 DEVICES**

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| DEV2 | DEV1 | DEV0 | REV4 | REV3 | REV2 | REV1 | REV0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7-5 **DEV<2:0>:** Device ID bits
111 = PIC18F1220
110 = PIC18F1320

bit 4-0 **REV<4:0>:** Revision ID bits
These bits are used to indicate the device revision.

**REGISTER 19-13: DEVID2: DEVICE ID REGISTER 2 FOR PIC18F1220/1320 DEVICES**

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| DEV10 | DEV9 | DEV8 | DEV7 | DEV6 | DEV5 | DEV4 | DEV3 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7-0 **DEV<10:3>:** Device ID bits
These bits are used with the DEV<2:0> bits in the Device ID Register 1 to identify the part number.
0000 0111 = PIC18F1220/1320 devices

**Note 1:** These values for DEV<10:3> may be shared with other devices. The specific device is always identified by using the entire DEV<10:0> bit sequence.

## 19.2 Watchdog Timer (WDT)

For PIC18F1220/1320 devices, the WDT is driven by the INTRC source. When the WDT is enabled, the clock source is also enabled. The nominal WDT period is 4 ms and has the same stability as the INTRC oscillator.

The 4 ms period of the WDT is multiplied by a 16-bit postscaler. Any output of the WDT postscaler is selected by a multiplexer, controlled by bits in Configuration Register 2H. Available periods range from 4 ms to 131.072 seconds (2.18 minutes). The WDT and postscaler are cleared when any of the following events occur: execute a SLEEP or CLRWDT instruction, the IRCF bits (OSCCON<6:4>) are changed or a clock failure has occurred.

Adjustments to the internal oscillator clock period using the OSCTUNE register also affect the period of the WDT by the same factor. For example, if the INTRC period is increased by 3%, then the WDT period is increased by 3%.

**Note 1:** The CLRWDT and SLEEP instructions clear the WDT and postscaler counts when executed.

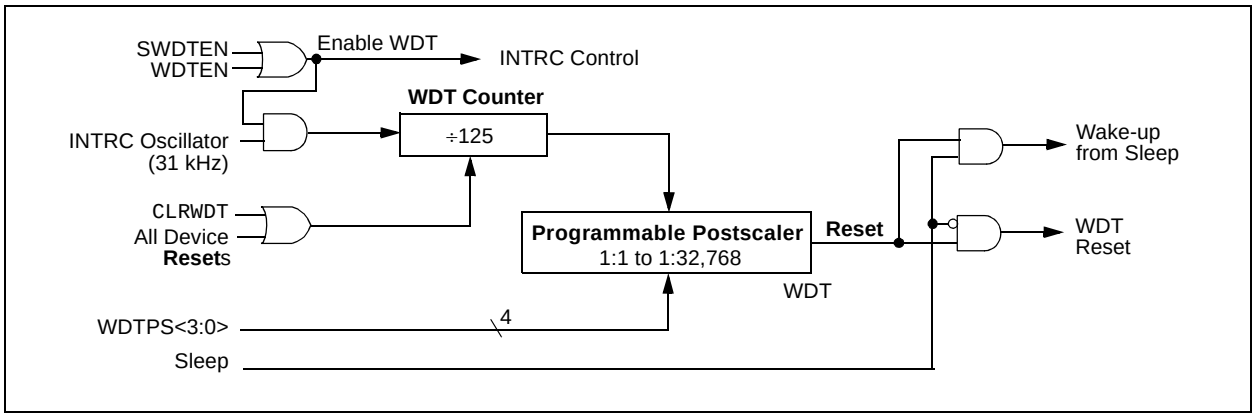
**2:** Changing the setting of the IRCF bits (OSCCON<6:4>) clears the WDT and postscaler counts.

**3:** When a CLRWDT instruction is executed the postscaler count will be cleared.

### 19.2.1 CONTROL REGISTER

Register 19-14 shows the WDTCON register. This is a readable and writable register, which contains a control bit that allows software to override the WDT enable Configuration bit, only if the Configuration bit has disabled the WDT.

**FIGURE 19-1: WDT BLOCK DIAGRAM**




**REGISTER 19-14: WDTCON: WATCHDOG TIMER CONTROL REGISTER**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | SWDTEN(1) |
| bit 7 | | | | | | | bit 0 |

| Legend: | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -m/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-1 **Unimplemented:** Read as '0'

bit 0 **SWDTEN:** Software Controlled Watchdog Timer Enable bit(1)

1 = Watchdog Timer is on
0 = Watchdog Timer is off

**Note 1:** This bit has no effect if the Configuration bit, WDTEN (CONFIG2H<0>), is enabled.


© 2002-2015 Microchip Technology Inc.

**TABLE 19-2: SUMMARY OF WATCHDOG TIMER REGISTERS**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| CONFIG2H | — | — | — | WDTPS3 | WDTPS2 | WDTPS2 | WDTPS0 | WDTEN |
| RCON | IPEN | — | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ |
| WDTCON | — | — | — | — | — | — | — | SWDTEN |

**Legend:** Shaded cells are not used by the Watchdog Timer.

## 19.3 Two-Speed Start-up

The Two-Speed Start-up feature helps to minimize the latency period from oscillator start-up to code execution by allowing the microcontroller to use the INTRC oscillator as a clock source until the primary clock source is available. It is enabled by setting the IESO bit in Configuration Register 1H (CONFIG1H<7>).

Two-Speed Start-up is available only if the primary oscillator mode is LP, XT, HS or HSPLL (crystal-based modes). Other sources do not require an OST start-up delay; for these, Two-Speed Start-up is disabled.

When enabled, Resets and wake-ups from Sleep mode cause the device to configure itself to run from the internal oscillator block as the clock source, following the time-out of the Power-up Timer after a Power-on Reset is enabled. This allows almost immediate code execution while the primary oscillator starts and the OST is running. Once the OST times out, the device automatically switches to PRI_RUN mode.

Because the OSCCON register is cleared on Reset events, the INTOSC (or postscaler) clock source is not initially available after a Reset event; the INTRC clock is used directly at its base frequency. To use a higher clock speed on wake-up, the INTOSC or postscaler clock sources can be selected to provide a higher clock speed by setting bits, IFRC2:IFRC0, immediately after Reset. For wake-ups from Sleep, the INTOSC or postscaler clock sources can be selected by setting IFRC2:IFRC0 prior to entering Sleep mode.

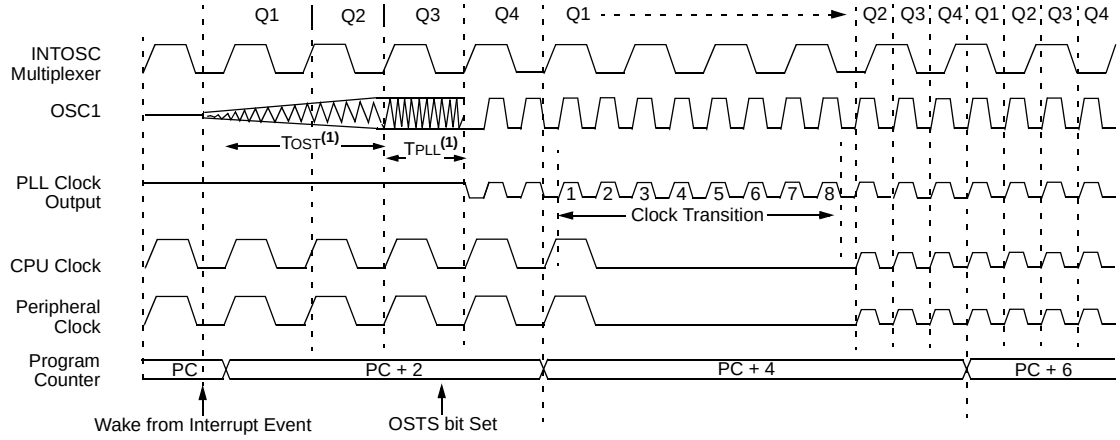
In all other power managed modes, Two-Speed Start-up is not used. The device will be clocked by the currently selected clock source until the primary clock source becomes available. The setting of the IESO bit is ignored.

### 19.3.1 SPECIAL CONSIDERATIONS FOR USING TWO-SPEED START-UP

While using the INTRC oscillator in Two-Speed Start-up, the device still obeys the normal command sequences for entering power managed modes, including serial SLEEP instructions (refer to **Section 3.1.3 "Multiple Sleep Commands"**). In practice, this means that user code can change the SCS1:SCS0 bit settings and issue SLEEP commands before the OST times out. This would allow an application to briefly wake-up, perform routine "housekeeping" tasks and return to Sleep before the device starts to operate from the primary oscillator.

User code can also check if the primary clock source is currently providing the system clocking by checking the status of the OSTS bit (OSCCON<3>). If the bit is set, the primary oscillator is providing the system clock. Otherwise, the internal oscillator block is providing the clock during wake-up from Reset or Sleep mode.

**FIGURE 19-2: TIMING TRANSITION FOR TWO-SPEED START-UP (INTOSC TO HSPLL)**



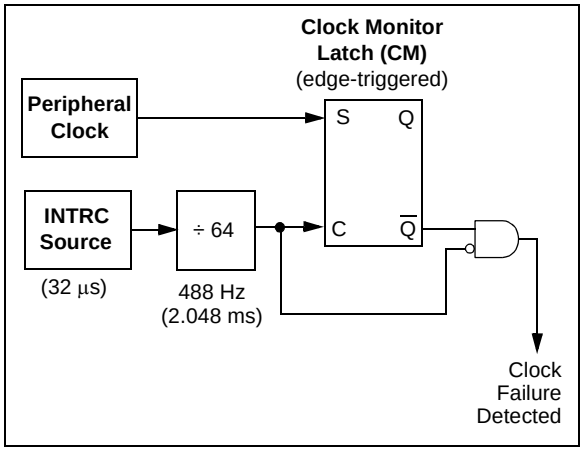

**Note 1:** TOST = 1024 TOSC; TPLL = 2 ms (approx). These intervals are not shown to scale.

## 19.4 Fail-Safe Clock Monitor

The Fail-Safe Clock Monitor (FSCM) allows the microcontroller to continue operation, in the event of an external oscillator failure, by automatically switching the system clock to the internal oscillator block. The FSCM function is enabled by setting the Fail-Safe Clock Monitor Enable bit, FSCM (CONFIG1H<6>).

When FSCM is enabled, the INTRC oscillator runs at all times to monitor clocks to peripherals and provide an instant backup clock in the event of a clock failure. Clock monitoring (shown in Figure 19-3) is accomplished by creating a sample clock signal, which is the INTRC output divided by 64. This allows ample time between FSCM sample clocks for a peripheral clock edge to occur. The peripheral system clock and the sample clock are presented as inputs to the Clock Monitor latch (CM). The CM is set on the falling edge of the system clock source, but cleared on the rising edge of the sample clock.

**FIGURE 19-3: FSCM BLOCK DIAGRAM**




Clock failure is tested for on the falling edge of the sample clock. If a sample clock falling edge occurs while CM is still set, a clock failure has been detected (Figure 19-4). This causes the following:

- the FSCM generates an oscillator fail interrupt by setting bit, OSCFIF (PIR2<7>);
- the system clock source is switched to the internal oscillator block (OSCCON is not updated to show the current clock source – this is the Fail-Safe condition); and
- the WDT is reset.

Since the postscaler frequency from the internal oscillator block may not be sufficiently stable, it may be desirable to select another clock configuration and enter an alternate power managed mode (see **Section 19.3.1 "Special Considerations for Using Two-Speed Start-up"** and **Section 3.1.3 "Multiple Sleep Commands"** for more details). This can be done to attempt a partial recovery, or execute a controlled shutdown.

To use a higher clock speed on wake-up, the INTOSC or postscaler clock sources can be selected to provide a higher clock speed by setting bits, IFRC2:IFRC0, immediately after Reset. For wake-ups from Sleep, the INTOSC or postscaler clock sources can be selected by setting IFRC2:IFRC0 prior to entering Sleep mode.

Adjustments to the internal oscillator block, using the OSCTUNE register, also affect the period of the FSCM by the same factor. This can usually be neglected, as the clock frequency being monitored is generally much higher than the sample clock frequency.

The FSCM will detect failures of the primary or secondary clock sources only. If the internal oscillator block fails, no failure would be detected, nor would any action be possible.

### 19.4.1 FSCM AND THE WATCHDOG TIMER

Both the FSCM and the WDT are clocked by the INTRC oscillator. Since the WDT operates with a separate divider and counter, disabling the WDT has no effect on the operation of the INTRC oscillator when the FSCM is enabled.

As already noted, the clock source is switched to the INTOSC clock when a clock failure is detected. Depending on the frequency selected by the IRCF2:IRCF0 bits, this may mean a substantial change in the speed of code execution. If the WDT is enabled with a small prescale value, a decrease in clock speed allows a WDT time-out to occur and a subsequent device Reset. For this reason, Fail-Safe Clock events also reset the WDT and postscaler, allowing it to start timing from when execution speed was changed and decreasing the likelihood of an erroneous time-out.


© 2002-2015 Microchip Technology Inc.

### 19.4.2 EXITING FAIL-SAFE OPERATION

The Fail-Safe condition is terminated by either a device Reset, or by entering a power managed mode. On Reset, the controller starts the primary clock source specified in Configuration Register 1H (with any required start-up delays that are required for the oscillator mode, such as OST or PLL timer). The INTOSC multiplexer provides the system clock until the primary clock source becomes ready (similar to a Two-Speed Start-up). The clock system source is then switched to the primary clock (indicated by the OSTS bit in the OSCCON register becoming set). The Fail-Safe Clock Monitor then resumes monitoring the peripheral clock.

The primary clock source may never become ready during start-up. In this case, operation is clocked by the INTOSC multiplexer. The OSCCON register will remain in its Reset state until a power managed mode is entered.

Entering a power managed mode by loading the OSCCON register and executing a SLEEP instruction will clear the Fail-Safe condition. When the Fail-Safe condition is cleared, the clock monitor will resume monitoring the peripheral clock.

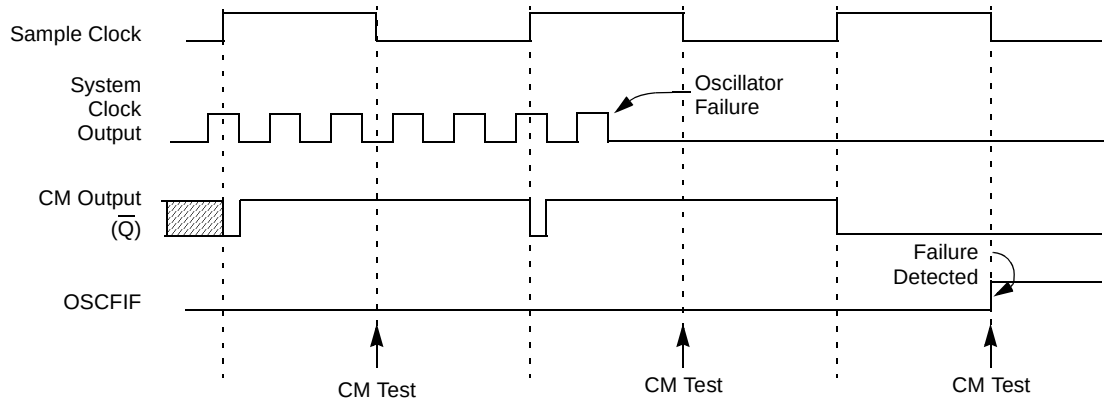
### 19.4.3 FSCM INTERRUPTS IN POWER MANAGED MODES

As previously mentioned, entering a power managed mode clears the Fail-Safe condition. By entering a power managed mode, the clock multiplexer selects the clock source selected by the OSCCON register. Fail-Safe monitoring of the power managed clock source resumes in the power managed mode.

If an oscillator failure occurs during power managed operation, the subsequent events depend on whether or not the oscillator failure interrupt is enabled. If enabled (OSCFIF = 1), code execution will be clocked by the INTOSC multiplexer. An automatic transition back to the failed clock source will not occur.

If the interrupt is disabled, the device will not exit the power managed mode on oscillator failure. Instead, the device will continue to operate as before, but clocked by the INTOSC multiplexer. While in Idle mode, subsequent interrupts will cause the CPU to begin executing instructions while being clocked by the INTOSC multiplexer. The device will not transition to a different clock source until the Fail-Safe condition is cleared.

**FIGURE 19-4: FSCM TIMING DIAGRAM**




**Note:** The system clock is normally at a much higher frequency than the sample clock. The relative frequencies in this example have been chosen for clarity.

### 19.4.4 POR OR WAKE FROM SLEEP

The FSCM is designed to detect oscillator failure at any point after the device has exited Power-on Reset (POR) or Low-Power Sleep mode. When the primary system clock is EC, RC or INTRC modes, monitoring can begin immediately following these events.

For oscillator modes involving a crystal or resonator (HS, HSPLL, LP or XT), the situation is somewhat different. Since the oscillator may require a start-up time considerably longer than the FCSM sample clock time, a false clock failure may be detected. To prevent this, the internal oscillator block is automatically configured as the system clock and functions until the primary clock is stable (the OST and PLL timers have timed out). This is identical to Two-Speed Start-up mode. Once the primary clock is stable, the INTRC returns to its role as the FSCM source

| **Note:** | The same logic that prevents false oscillator failure interrupts on POR or wake from Sleep will also prevent the detection of the oscillator's failure to start at all following these events. This can be avoided by monitoring the OSTS bit and using a timing routine to determine if the oscillator is taking too long to start. Even so, no oscillator failure interrupt will be flagged. |
|---|---|

As noted in **Section 19.3.1 "Special Considerations for Using Two-Speed Start-up"**, it is also possible to select another clock configuration and enter an alternate power managed mode while waiting for the primary system clock to become stable. When the new powered managed mode is selected, the primary clock is disabled.


© 2002-2015 Microchip Technology Inc.

## 19.5 Program Verification and Code Protection

The overall structure of the code protection on the PIC18 Flash devices differs significantly from other PIC devices.

The user program memory is divided into three blocks. One of these is a boot block of 512 bytes. The remainder of the memory is divided into two blocks on binary boundaries.

Each of the three blocks has three protection bits associated with them. They are:

- Code-Protect bit (CPn)
- Write-Protect bit (WRTn)
- External Block Table Read bit (EBTRn)

Figure 19-5 shows the program memory organization for 4 and 8-Kbyte devices and the specific code protection bit associated with each block. The actual locations of the bits are summarized in Table 19-3.

**FIGURE 19-5: CODE-PROTECTED PROGRAM MEMORY FOR PIC18F1220/1320**

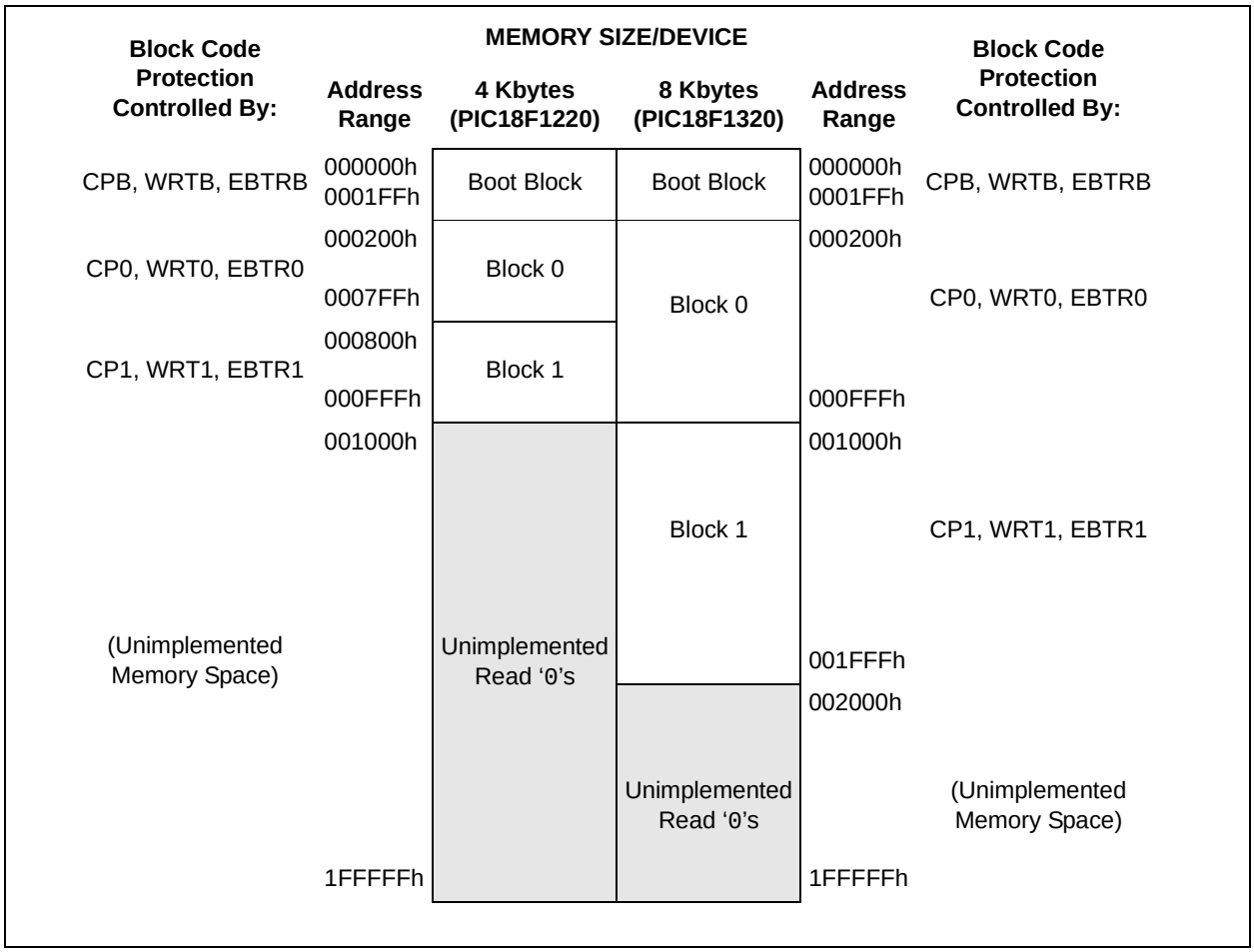



**TABLE 19-3: SUMMARY OF CODE PROTECTION REGISTERS**

| File Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 300008h | CONFIG5L | — | — | — | — | — | — | CP1 | CP0 |
| 300009h | CONFIG5H | CPD | CPB | — | — | — | — | — | — |
| 30000Ah | CONFIG6L | — | — | — | — | — | — | WRT1 | WRT0 |
| 30000Bh | CONFIG6H | WRTD | WRTB | WRTC | — | — | — | — | — |
| 30000Ch | CONFIG7L | — | — | — | — | — | — | EBTR1 | EBTR0 |
| 30000Dh | CONFIG7H | — | EBTRB | — | — | — | — | — | — |

**Legend:** Shaded cells are unimplemented.

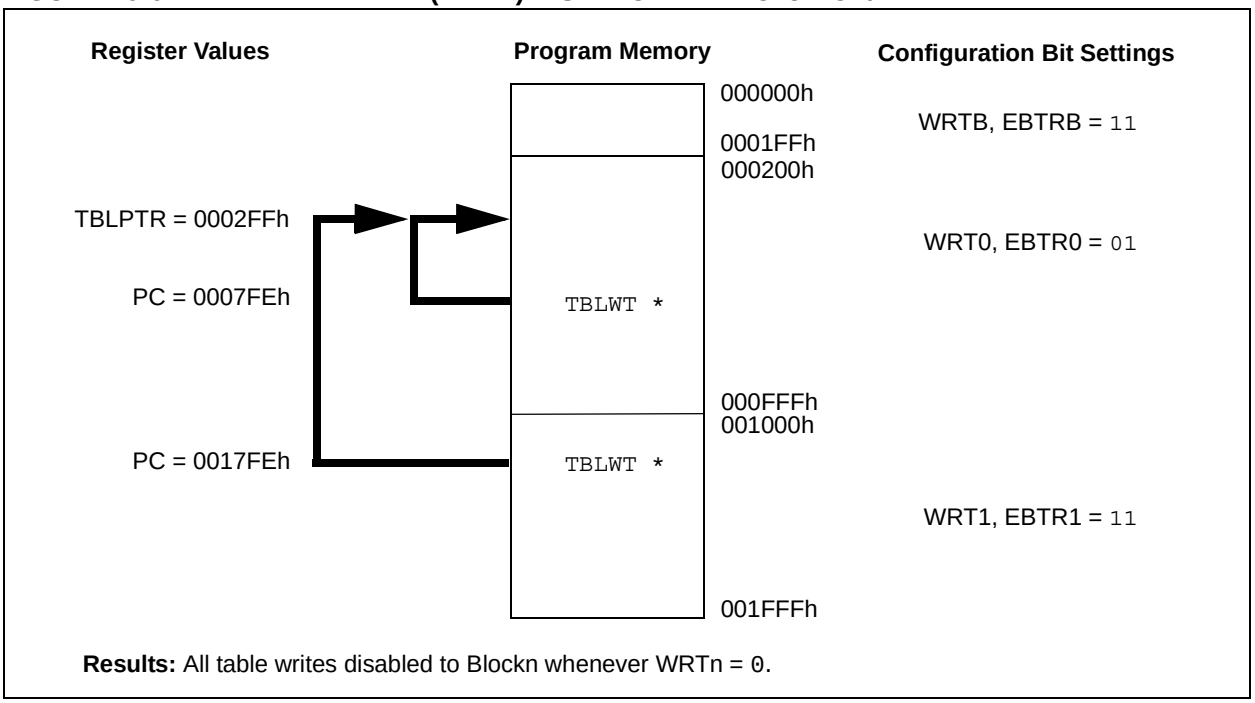
### 19.5.1 PROGRAM MEMORY CODE PROTECTION

The program memory may be read to, or written from, any location using the table read and table write instructions. The device ID may be read with table reads. The Configuration registers may be read and written with the table read and table write instructions.

In normal execution mode, the CPn bits have no direct effect. CPn bits inhibit external reads and writes. A block of user memory may be protected from table writes if the WRTn Configuration bit is '0'. The EBTRn bits control table reads. For a block of user memory with the EBTRn bit set to '0', a table read instruction that executes from within that block is allowed to read. A table read instruction that executes from a location outside of that block is not allowed to read and will result in reading '0's. Figures 19-6 through 19-8 illustrate table write and table read protection.

| **Note:** | Code protection bits may only be written to a '0' from a '1' state. It is not possible to write a '1' to a bit in the '0' state. Code protection bits are only set to '1' by a full Chip Erase or Block Erase function. The full Chip Erase and Block Erase functions can only be initiated via ICSP or an external programmer. |
|---|---|

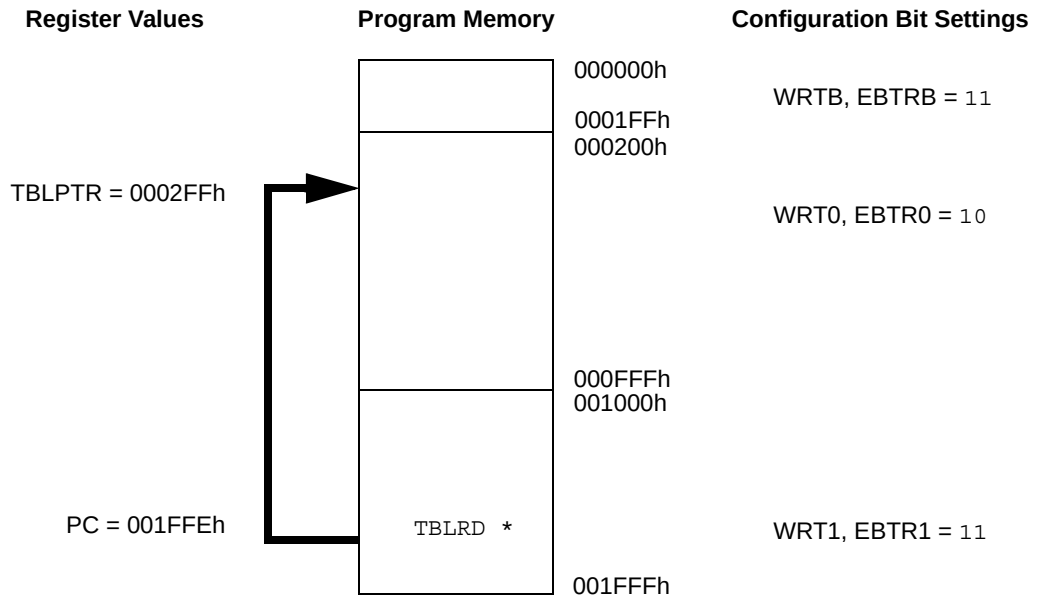
**FIGURE 19-6: TABLE WRITE (WRTn) DISALLOWED: PIC18F1320**




**Results:** All table writes disabled to Blockn whenever WRTn = 0.

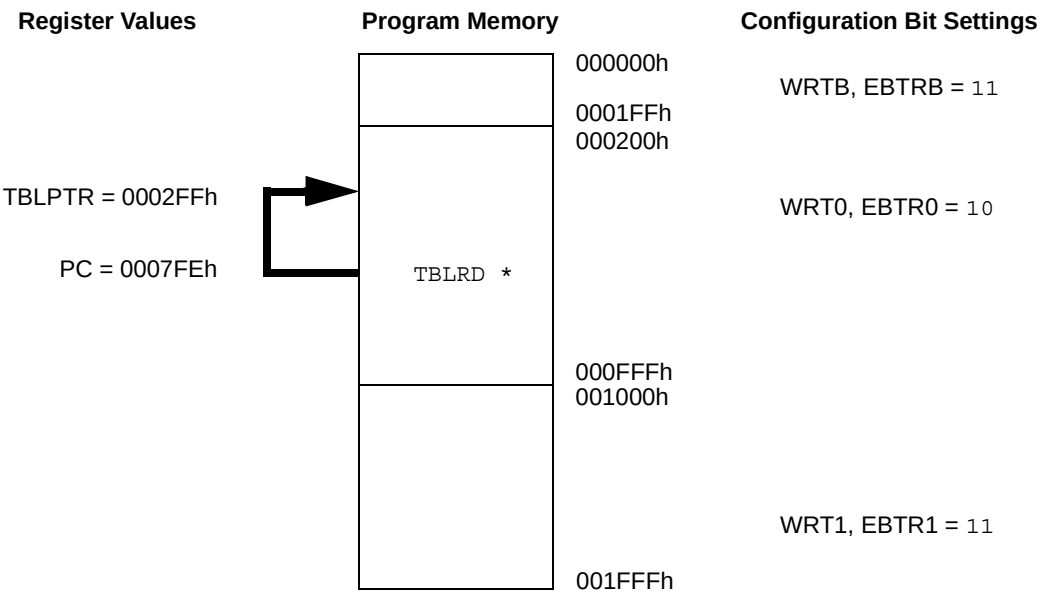

© 2002-2015 Microchip Technology Inc.

**FIGURE 19-7: EXTERNAL BLOCK TABLE READ (EBTRn) DISALLOWED: PIC18F1320**




**Results:** All table reads from external blocks to Blockn are disabled whenever EBTRn = 0.
TABLAT register returns a value of '0'.

**FIGURE 19-8: EXTERNAL BLOCK TABLE READ (EBTRn) ALLOWED: PIC18F1320**




**Results:** Table reads permitted within Blockn, even when EBTRBn = 0.
TABLAT register returns the value of the data at the location TBLPTR.

### 19.5.2 DATA EEPROM CODE PROTECTION

The entire data EEPROM is protected from external reads and writes by two bits: CPD and WRTD. CPD inhibits external reads and writes of data EEPROM. WRTD inhibits external writes to data EEPROM. The CPU can continue to read and write data EEPROM, regardless of the protection bit settings.

### 19.5.3 CONFIGURATION REGISTER PROTECTION

The Configuration registers can be write-protected. The WRTC bit controls protection of the Configuration registers. In normal execution mode, the WRTC bit is readable only. WRTC can only be written via ICSP or an external programmer.

## 19.6 ID Locations

Eight memory locations (200000h-200007h) are designated as ID locations, where the user can store checksum or other code identification numbers. These locations are both readable and writable during normal execution through the TBLRD and TBLWT instructions, or during program/verify. The ID locations can be read when the device is code-protected.

## 19.7 In-Circuit Serial Programming

PIC18F1220/1320 microcontrollers can be serially programmed while in the end application circuit. This is simply done with two lines for clock and data and three other lines for power, ground and the programming voltage. This allows customers to manufacture boards with unprogrammed devices and then program the microcontroller just before shipping the product. This also allows the most recent firmware or a custom firmware to be programmed (see Table 19-4).

| | |
|---|---|
| **Note:** | The Timer1 oscillator shares the T1OSI and T1OSO pins with the PGD and PGC pins used for programming and debugging.<br><br>When using the Timer1 oscillator, In-Circuit Serial Programming (ICSP) may not function correctly (high voltage or low voltage), or the In-Circuit Debugger (ICD) may not communicate with the controller. As a result of using either ICSP or ICD, the Timer1 crystal may be damaged.<br><br>If ICSP or ICD operations are required, the crystal should be disconnected from the circuit (disconnect either lead), or installed after programming. The oscillator loading capacitors may remain in-circuit during ICSP or ICD operation. |

**TABLE 19-4: ICSP/ICD CONNECTIONS**

| Signal | Pin | Notes |
|---|---|---|
| PGD | RB7/PGD/T1OSI/ P1D/KBI3 | Shared with T1OSC – protect crystal |
| PGC | RB6/PGC/T1OSO/ T13CKI/P1C/KBI2 | Shared with T1OSC – protect crystal |
| $\overline{\text{MCLR}}$ | $\overline{\text{MCLR}}$/VPP/RA5 | |
| VDD | VDD | |
| VSS | VSS | |
| PGM | RB5/PGM/KBI1 | Optional – pull RB5 low is LVP enabled |

## 19.8 In-Circuit Debugger

When the DEBUG bit in Configuration register, CONFIG4L, is programmed to a '0', the In-Circuit Debugger functionality is enabled. This function allows simple debugging functions when used with MPLAB® IDE. When the microcontroller has this feature enabled, some resources are not available for general use. Table 19-5 shows which resources are required by the background debugger.

**TABLE 19-5: DEBUGGER RESOURCES**

| | |
|---|---|
| I/O pins: | RB6, RB7 |
| Stack: | 2 levels |
| Program Memory: | 512 bytes |
| Data Memory: | 10 bytes |

To use the In-Circuit Debugger function of the microcontroller, the design must implement In-Circuit Serial Programming connections to $\overline{\text{MCLR}}$/VPP, VDD, VSS, RB7 and RB6. This will interface to the In-Circuit Debugger module available from Microchip, or one of the third party development tool companies (see the note following **Section 19.7 "In-Circuit Serial Programming"** for more information).


© 2002-2015 Microchip Technology Inc.

## 19.9 Low-Voltage ICSP Programming

The LVP bit in Configuration register, CONFIG4L, enables Low-Voltage Programming (LVP). When LVP is enabled, the microcontroller can be programmed without requiring high voltage being applied to the MCLR/VPP/RA5 pin, but the RB5/PGM/KBI1 pin is then dedicated to controlling Program mode entry and is not available as a general purpose I/O pin.

LVP is enabled in erased devices.

While programming using LVP, VDD is applied to the MCLR/VPP/RA5 pin as in normal execution mode. To enter Programming mode, VDD is applied to the PGM pin.

**Note 1:** High-voltage programming is always available, regardless of the state of the LVP bit or the PGM pin, by applying VIHH to the MCLR pin.

**2:** When Low-Voltage Programming is enabled, the RB5 pin can no longer be used as a general purpose I/O pin.

**3:** When LVP is enabled, externally pull the PGM pin to VSS to allow normal program execution.

If Low-Voltage Programming mode will not be used, the LVP bit can be cleared and RB5/PGM/KBI1 becomes available as the digital I/O pin RB5. The LVP bit may be set or cleared only when using standard high-voltage programming (VIHH applied to the MCLR/VPP/RA5 pin). Once LVP has been disabled, only the standard high-voltage programming is available and must be used to program the device.

Memory that is not code-protected can be erased, using either a Block Erase, or erased row by row, then written at any specified VDD. If code-protected memory is to be erased, a Block Erase is required. If a Block Erase is to be performed when using Low-Voltage Programming, the device must be supplied with VDD of 4.5V to 5.5V.

## 20.0 INSTRUCTION SET SUMMARY

The PIC18 instruction set adds many enhancements to the previous PIC instruction sets, while maintaining an easy migration from these PIC instruction sets.

Most instructions are a single program memory word (16 bits), but there are three instructions that require two program memory locations.

Each single-word instruction is a 16-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 instruction set is highly orthogonal and is grouped into four basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal** operations
- **Control** operations

The PIC18 instruction set summary in Table 20-1 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 20-1 shows the opcode field descriptions.

Most **byte-oriented** instructions have three operands:

1. The file register (specified by ‘f’)
2. The destination of the result (specified by ‘d’)
3. The accessed memory (specified by ‘a’)

The file register designator ‘f’ specifies which file register is to be used by the instruction.

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

All **bit-oriented** instructions have three operands:

1. The file register (specified by ‘f’)
2. The bit in the file register (specified by ‘b’)
3. The accessed memory (specified by ‘a’)

The bit field designator ‘b’ selects the number of the bit affected by the operation, while the file register designator ‘f’ represents the number of the file in which the bit is located.

The **literal** instructions may use some of the following operands:

- A literal value to be loaded into a file register (specified by ‘k’)
- The desired FSR register to load the literal value into (specified by ‘f’)
- No operand required (specified by ‘—’)

The **control** instructions may use some of the following operands:

- A program memory address (specified by ‘n’)
- The mode of the `CALL` or `RETURN` instructions (specified by ‘s’)
- The mode of the table read and table write instructions (specified by ‘m’)
- No operand required (specified by ‘—’)

All instructions are a single word, except for three double-word instructions. These three instructions were made double-word instructions so that all the required information is available in these 32 bits. In the second word, the 4 MSbs are ‘1’s. If this second word is executed as an instruction (by itself), it will execute as a `NOP`.

All 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`.

The double-word instructions execute in two instruction cycles.

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. Two-word branch instructions (if true) would take 3 μs.

Figure 20-1 shows the general formats that the instructions can have.

All examples use the format ‘`nnh`’ to represent a hexadecimal number, where ‘`h`’ signifies a hexadecimal digit.

The Instruction Set Summary, shown in Table 20-1, lists the instructions recognized by the Microchip Assembler (MPASM™). **Section 20.2 “Instruction Set”** provides a description of each instruction.

### 20.1 Read-Modify-Write Operations

Any instruction that specifies a file register as part of the instruction performs a Read-Modify-Write (R-M-W) operation. The register is read, the data is modified and the result is stored according to either the instruction or the destination designator ‘d’. A read operation is performed on a register even if the instruction writes to that register.

For example, a “`BCF PORTB,1`” instruction will read PORTB, clear bit 1 of the data, then write the result back to PORTB. The read operation would have the unintended result that any condition that sets the RBIF flag would be cleared. The R-M-W operation may also copy the level of an input pin to its corresponding output latch.

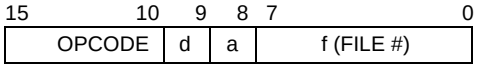

© 2002-2015 Microchip Technology Inc.

**TABLE 20-1: OPCODE FIELD DESCRIPTIONS**

| Field | Description |
|---|---|
| a | RAM access bit<br>a = 0: RAM location in Access RAM (BSR register is ignored)<br>a = 1: RAM bank is specified by BSR register |
| bbb | Bit address within an 8-bit file register (0 to 7). |
| BSR | Bank Select Register. Used to select the current RAM bank. |
| d | Destination select bit<br>d = 0: store result in WREG<br>d = 1: store result in file register f |
| dest | Destination either the WREG register or the specified register file location. |
| f | 8-bit register file address (0x00 to 0xFF). |
| fs | 12-bit register file address (0x000 to 0xFFF). This is the source address. |
| fd | 12-bit register file address (0x000 to 0xFFF). This is the destination address. |
| k | Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value). |
| label | Label name. |
| mm<br><br>*<br>*+<br>*-<br>+* | The mode of the TBLPTR register for the table read and table write instructions.<br>Only used with table read and table write instructions:<br>No change to register (such as TBLPTR with table reads and writes)<br>Post-Increment register (such as TBLPTR with table reads and writes)<br>Post-Decrement register (such as TBLPTR with table reads and writes)<br>Pre-Increment register (such as TBLPTR with table reads and writes) |
| n | The relative address (2's complement number) for relative branch instructions, or the direct address for call/branch and return instructions. |
| PRODH | Product of Multiply High Byte. |
| PRODL | Product of Multiply Low Byte. |
| s | Fast Call/Return mode select bit<br>s = 0: do not update into/from shadow registers<br>s = 1: certain registers loaded into/from shadow registers (Fast mode) |
| u | Unused or unchanged. |
| WREG | Working register (accumulator). |
| x | Don't care ('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. |
| TBLPTR | 21-bit Table Pointer (points to a program memory location). |
| TABLAT | 8-bit Table Latch. |
| TOS | Top-of-Stack. |
| PC | Program Counter. |
| PCL | Program Counter Low Byte. |
| PCH | Program Counter High Byte. |
| PCLATH | Program Counter High Byte Latch. |
| PCLATU | Program Counter Upper Byte Latch. |
| GIE | Global Interrupt Enable bit. |
| WDT | Watchdog Timer. |
| $\overline{TO}$ | Time-out bit. |
| $\overline{PD}$ | Power-down bit. |
| C, DC, Z, OV, N | ALU Status bits: Carry, Digit Carry, Zero, Overflow, Negative. |
| [ ] | Optional. |
| ( ) | Contents. |
| → | Assigned to. |
| < > | Register bit field. |
| ∈ | In the set of. |
| italics | User defined term (font is Courier). |

**FIGURE 20-1: GENERAL FORMAT FOR INSTRUCTIONS**

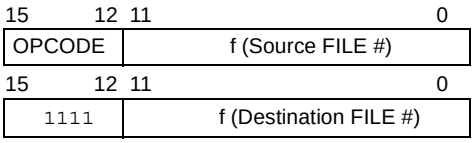
**Byte-oriented** file register operations — **Example Instruction**




ADDWF MYREG, W, B

d = 0 for result destination to be WREG register
d = 1 for result destination to be file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

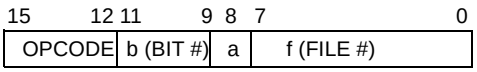
**Byte to Byte** move operations (2-word)




MOVFF MYREG1, MYREG2

f = 12-bit file register address

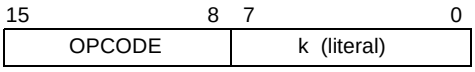
**Bit-oriented** file register operations




BSF MYREG, bit, B

b = 3-bit position of bit in file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

**Literal** operations

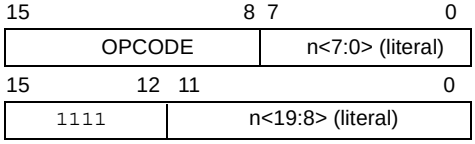



MOVLW 0x7F

k = 8-bit immediate value

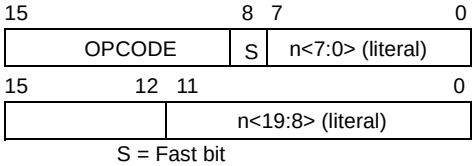
**Control** operations

**CALL, GOTO and Branch** operations



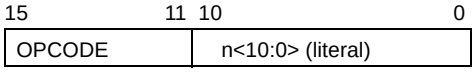

GOTO Label

n = 20-bit immediate value

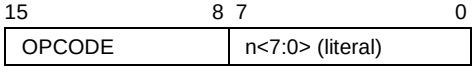



CALL MYFUNC

S = Fast bit




BRA MYFUNC




BC MYFUNC


© 2002-2015 Microchip Technology Inc.

**TABLE 20-1: PIC18FXXXX INSTRUCTION SET**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| | | | | MSb | | | LSb | | |
| **BYTE-ORIENTED FILE REGISTER OPERATIONS** | | | | | | | | | |
| ADDWF | f, d, a | Add WREG and f | 1 | 0010 | 01da | ffff | ffff | C, DC, Z, OV, N | 1, 2 |
| ADDWFC | f, d, a | Add WREG and Carry bit to f | 1 | 0010 | 00da | ffff | ffff | C, DC, Z, OV, N | 1, 2 |
| ANDWF | f, d, a | AND WREG with f | 1 | 0001 | 01da | ffff | ffff | Z, N | 1,2 |
| CLRF | f, a | Clear f | 1 | 0110 | 101a | ffff | ffff | Z | 2 |
| COMF | f, d, a | Complement f | 1 | 0001 | 11da | ffff | ffff | Z, N | 1, 2 |
| CPFSEQ | f, a | Compare f with WREG, skip = | 1 (2 or 3) | 0110 | 001a | ffff | ffff | None | 4 |
| CPFSGT | f, a | Compare f with WREG, skip > | 1 (2 or 3) | 0110 | 010a | ffff | ffff | None | 4 |
| CPFSLT | f, a | Compare f with WREG, skip < | 1 (2 or 3) | 0110 | 000a | ffff | ffff | None | 1, 2 |
| DECF | f, d, a | Decrement f | 1 | 0000 | 01da | ffff | ffff | C, DC, Z, OV, N | 1, 2, 3, 4 |
| DECFSZ | f, d, a | Decrement f, Skip if 0 | 1 (2 or 3) | 0010 | 11da | ffff | ffff | None | 1, 2, 3, 4 |
| DCFSNZ | f, d, a | Decrement f, Skip if Not 0 | 1 (2 or 3) | 0100 | 11da | ffff | ffff | None | 1, 2 |
| INCF | f, d, a | Increment f | 1 | 0010 | 10da | ffff | ffff | C, DC, Z, OV, N | 1, 2, 3, 4 |
| INCFSZ | f, d, a | Increment f, Skip if 0 | 1 (2 or 3) | 0011 | 11da | ffff | ffff | None | 4 |
| INFSNZ | f, d, a | Increment f, Skip if Not 0 | 1 (2 or 3) | 0100 | 10da | ffff | ffff | None | 1, 2 |
| IORWF | f, d, a | Inclusive OR WREG with f | 1 | 0001 | 00da | ffff | ffff | Z, N | 1, 2 |
| MOVF | f, d, a | Move f | 1 | 0101 | 00da | ffff | ffff | Z, N | 1 |
| MOVFF | $f_s$, $f_d$ | Move $f_s$ (source) to 1st word<br>$f_d$ (destination) 2nd word | 2 | 1100<br>1111 | ffff<br>ffff | ffff<br>ffff | ffff<br>ffff | None | |
| MOVWF | f, a | Move WREG to f | 1 | 0110 | 111a | ffff | ffff | None | |
| MULWF | f, a | Multiply WREG with f | 1 | 0000 | 001a | ffff | ffff | None | |
| NEGF | f, a | Negate f | 1 | 0110 | 110a | ffff | ffff | C, DC, Z, OV, N | 1, 2 |
| RLCF | f, d, a | Rotate Left f through Carry | 1 | 0011 | 01da | ffff | ffff | C, Z, N | |
| RLNCF | f, d, a | Rotate Left f (No Carry) | 1 | 0100 | 01da | ffff | ffff | Z, N | 1, 2 |
| RRCF | f, d, a | Rotate Right f through Carry | 1 | 0011 | 00da | ffff | ffff | C, Z, N | |
| RRNCF | f, d, a | Rotate Right f (No Carry) | 1 | 0100 | 00da | ffff | ffff | Z, N | |
| SETF | f, a | Set f | 1 | 0110 | 100a | ffff | ffff | None | |
| SUBFWB | f, d, a | Subtract f from WREG with borrow | 1 | 0101 | 01da | ffff | ffff | C, DC, Z, OV, N | 1, 2 |
| SUBWF | f, d, a | Subtract WREG from f | 1 | 0101 | 11da | ffff | ffff | C, DC, Z, OV, N | |
| SUBWFB | f, d, a | Subtract WREG from f with borrow | 1 | 0101 | 10da | ffff | ffff | C, DC, Z, OV, N | 1, 2 |
| SWAPF | f, d, a | Swap nibbles in f | 1 | 0011 | 10da | ffff | ffff | None | 4 |
| TSTFSZ | f, a | Test f, skip if 0 | 1 (2 or 3) | 0110 | 011a | ffff | ffff | None | 1, 2 |
| XORWF | f, d, a | Exclusive OR WREG with f | 1 | 0001 | 10da | ffff | ffff | Z, N | |
| **BIT-ORIENTED FILE REGISTER OPERATIONS** | | | | | | | | | |
| BCF | f, b, a | Bit Clear f | 1 | 1001 | bbba | ffff | ffff | None | 1, 2 |
| BSF | f, b, a | Bit Set f | 1 | 1000 | bbba | ffff | ffff | None | 1, 2 |
| BTFSC | f, b, a | Bit Test f, Skip if Clear | 1 (2 or 3) | 1011 | bbba | ffff | ffff | None | 3, 4 |
| BTFSS | f, b, a | Bit Test f, Skip if Set | 1 (2 or 3) | 1010 | bbba | ffff | ffff | None | 3, 4 |
| BTG | f, d, a | Bit Toggle f | 1 | 0111 | bbba | ffff | ffff | None | 1, 2 |

**Note 1:** When a Port register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

**2:** If this instruction is executed on the TMR0 register (and where applicable, d = 1), the prescaler will be cleared if assigned.

**3:** If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

**4:** Some instructions are 2-word instructions. The second word of these instructions will be executed as a `NOP`, unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

**5:** If the table write starts the write cycle to internal memory, the write will continue until terminated.

**TABLE 20-1: PIC18FXXXX INSTRUCTION SET (CONTINUED)**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word MSb LSb | Status Affected | Notes |
|---|---|---|---|---|---|---|
| **CONTROL OPERATIONS** | | | | | | |
| BC | n | Branch if Carry | 1 (2) | 1110 0010 nnnn nnnn | None | |
| BN | n | Branch if Negative | 1 (2) | 1110 0110 nnnn nnnn | None | |
| BNC | n | Branch if Not Carry | 1 (2) | 1110 0011 nnnn nnnn | None | |
| BNN | n | Branch if Not Negative | 1 (2) | 1110 0111 nnnn nnnn | None | |
| BNOV | n | Branch if Not Overflow | 1 (2) | 1110 0101 nnnn nnnn | None | |
| BNZ | n | Branch if Not Zero | 1 (2) | 1110 0001 nnnn nnnn | None | |
| BOV | n | Branch if Overflow | 1 (2) | 1110 0100 nnnn nnnn | None | |
| BRA | n | Branch Unconditionally | 2 | 1101 0nnn nnnn nnnn | None | |
| BZ | n | Branch if Zero | 1 (2) | 1110 0000 nnnn nnnn | None | |
| CALL | n, s | Call subroutine 1st word | 2 | 1110 110s kkkk kkkk | None | |
| | | 2nd word | | 1111 kkkk kkkk kkkk | | |
| CLRWDT | — | Clear Watchdog Timer | 1 | 0000 0000 0000 0100 | $\overline{TO}$, $\overline{PD}$ | |
| DAW | — | Decimal Adjust WREG | 1 | 0000 0000 0000 0111 | C | |
| GOTO | n | Go to address 1st word | 2 | 1110 1111 kkkk kkkk | None | |
| | | 2nd word | | 1111 kkkk kkkk kkkk | | |
| NOP | — | No Operation | 1 | 0000 0000 0000 0000 | None | |
| NOP | — | No Operation | 1 | 1111 xxxx xxxx xxxx | None | 4 |
| POP | — | Pop top of return stack (TOS) | 1 | 0000 0000 0000 0110 | None | |
| PUSH | — | Push top of return stack (TOS) | 1 | 0000 0000 0000 0101 | None | |
| RCALL | n | Relative Call | 2 | 1101 1nnn nnnn nnnn | None | |
| RESET | | Software device Reset | 1 | 0000 0000 1111 1111 | All | |
| RETFIE | s | Return from interrupt enable | 2 | 0000 0000 0001 000s | GIE/GIEH, PEIE/GIEL | |
| RETLW | k | Return with literal in WREG | 2 | 0000 1100 kkkk kkkk | None | |
| RETURN | s | Return from Subroutine | 2 | 0000 0000 0001 001s | None | |
| SLEEP | — | Go into Standby mode | 1 | 0000 0000 0000 0011 | $\overline{TO}$, $\overline{PD}$ | |

**Note 1:** When a Port register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

**2:** If this instruction is executed on the TMR0 register (and where applicable, d = 1), the prescaler will be cleared if assigned.

**3:** If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

**4:** Some instructions are 2-word instructions. The second word of these instructions will be executed as a NOP, unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

**5:** If the table write starts the write cycle to internal memory, the write will continue until terminated.


© 2002-2015 Microchip Technology Inc.

**TABLE 20-1: PIC18FXXXX INSTRUCTION SET (CONTINUED)**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| | | | | MSb | | | LSb | | |
| **LITERAL OPERATIONS** | | | | | | | | | |
| ADDLW | k | Add literal and WREG | 1 | 0000 | 1111 | kkkk | kkkk | C, DC, Z, OV, N | |
| ANDLW | k | AND literal with WREG | 1 | 0000 | 1011 | kkkk | kkkk | Z, N | |
| IORLW | k | Inclusive OR literal with WREG | 1 | 0000 | 1001 | kkkk | kkkk | Z, N | |
| LFSR | f, k | Move literal (12-bit) 2nd word to FSRx 1st word | 2 | 1110<br>1111 | 1110<br>0000 | 00ff<br>kkkk | kkkk<br>kkkk | None | |
| MOVLB | k | Move literal to BSR<3:0> | 1 | 0000 | 0001 | 0000 | kkkk | None | |
| MOVLW | k | Move literal to WREG | 1 | 0000 | 1110 | kkkk | kkkk | None | |
| MULLW | k | Multiply literal with WREG | 1 | 0000 | 1101 | kkkk | kkkk | None | |
| RETLW | k | Return with literal in WREG | 2 | 0000 | 1100 | kkkk | kkkk | None | |
| SUBLW | k | Subtract WREG from literal | 1 | 0000 | 1000 | kkkk | kkkk | C, DC, Z, OV, N | |
| XORLW | k | Exclusive OR literal with WREG | 1 | 0000 | 1010 | kkkk | kkkk | Z, N | |
| **DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS** | | | | | | | | | |
| TBLRD* | | Table read | 2 | 0000 | 0000 | 0000 | 1000 | None | |
| TBLRD*+ | | Table read with post-increment | | 0000 | 0000 | 0000 | 1001 | None | |
| TBLRD*- | | Table read with post-decrement | | 0000 | 0000 | 0000 | 1010 | None | |
| TBLRD+* | | Table read with pre-increment | | 0000 | 0000 | 0000 | 1011 | None | |
| TBLWT* | | Table write | 2 (5) | 0000 | 0000 | 0000 | 1100 | None | |
| TBLWT*+ | | Table write with post-increment | | 0000 | 0000 | 0000 | 1101 | None | |
| TBLWT*- | | Table write with post-decrement | | 0000 | 0000 | 0000 | 1110 | None | |
| TBLWT+* | | Table write with pre-increment | | 0000 | 0000 | 0000 | 1111 | None | |

**Note 1:** When a Port register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

**2:** If this instruction is executed on the TMR0 register (and where applicable, d = 1), the prescaler will be cleared if assigned.

**3:** If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

**4:** Some instructions are 2-word instructions. The second word of these instructions will be executed as a `NOP`, unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

**5:** If the table write starts the write cycle to internal memory, the write will continue until terminated.

## 20.2 Instruction Set

**ADDLW** **ADD literal to W**

Syntax: [ *label* ] ADDLW k

Operands: $0 \le k \le 255$

Operation: $(W) + k \rightarrow W$

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1111 | kkkk | kkkk |

Description: The contents of W are added to the 8-bit literal ‘k’ and the result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal ‘k’ | Process Data | Write to W |

Example: `ADDLW 0x15`

Before Instruction
W = 0x10

After Instruction
W = 0x25

**ADDWF** **ADD W to f**

Syntax: [ *label* ] ADDWF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $(W) + (f) \rightarrow dest$

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0010 | 01da | ffff | ffff |

Description: Add W to register ‘f’. If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’ (default). If ‘a’ is ‘0’, the Access Bank will be selected. If ‘a’ is ‘1’, the BSR is used.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write to destination |

Example: `ADDWF REG, W`

Before Instruction
W = 0x17
REG = 0xC2

After Instruction
W = 0xD9
REG = 0xC2


© 2002-2015 Microchip Technology Inc.

**ADDWFC** **ADD W and Carry bit to f**

Syntax: [ *label* ] ADDWFC f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (W) + (f) + (C) → dest

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0010 | 00da | ffff | ffff |

Description: Add W, the Carry flag and data memory location 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in data memory location 'f'. If 'a' is '0', the Access Bank will be selected. If 'a' is '1', the BSR will not be overridden.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `ADDWFC REG, W`

Before Instruction
Carry bit = 1
REG = 0x02
W = 0x4D

After Instruction
Carry bit = 0
REG = 0x02
W = 0x50

**ANDLW** **AND literal with W**

Syntax: [ *label* ] ANDLW k

Operands: $0 \le k \le 255$

Operation: (W) .AND. k → W

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1011 | kkkk | kkkk |

Description: The contents of W are AND'ed with the 8-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `ANDLW 0x5F`

Before Instruction
W = 0xA3

After Instruction
W = 0x03

**ANDWF** **AND W with f**

Syntax: [ *label* ] ANDWF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (W) .AND. (f) → dest

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0001 | 01da | ffff | ffff |

Description: The contents of W are AND'ed with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank will be selected. If 'a' is '1', the BSR will not be overridden (default).

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `ANDWF REG, W`

Before Instruction
W = 0x17
REG = 0xC2

After Instruction
W = 0x02
REG = 0xC2

**BC** **Branch if Carry**

Syntax: [ *label* ] BC n

Operands: $-128 \le n \le 127$

Operation: if Carry bit is '1'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 0010 | nnnn | nnnn |

Description: If the Carry bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example:
```
HERE    BC   JUMP
```
Before Instruction
PC = address (HERE)
After Instruction
If Carry = 1;
PC = address (JUMP)
If Carry = 0;
PC = address (HERE + 2)


© 2002-2015 Microchip Technology Inc.

**BCF** **Bit Clear f**

Syntax: [ *label* ] BCF f,b[,a]

Operands: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Operation: $0 \rightarrow f<b>$

Status Affected: None

Encoding:

| 1001 | bbba | ffff | ffff |
|---|---|---|---|

Description: Bit 'b' in register 'f' is cleared. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `BCF FLAG_REG, 7`

Before Instruction
FLAG_REG = 0xC7
After Instruction
FLAG_REG = 0x47

**BN** **Branch if Negative**

Syntax: [ *label* ] BN n

Operands: $-128 \le n \le 127$

Operation: if Negative bit is '1'
$(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

| 1110 | 0110 | nnnn | nnnn |
|---|---|---|---|

Description: If the Negative bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example: `HERE BN Jump`

Before Instruction
PC = address (HERE)
After Instruction
If Negative = 1;
PC = address (Jump)
If Negative = 0;
PC = address (HERE + 2)

**BNC** **Branch if Not Carry**

Syntax: [ *label* ] BNC n

Operands: -128 ≤ n ≤ 127

Operation: if Carry bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 0011 | nnnn | nnnn |

Description: If the Carry bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example: HERE BNC Jump

Before Instruction
PC = address (HERE)
After Instruction
If Carry = 0;
PC = address (Jump)
If Carry = 1;
PC = address (HERE + 2)

**BNN** **Branch if Not Negative**

Syntax: [ *label* ] BNN n

Operands: -128 ≤ n ≤ 127

Operation: if Negative bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 0111 | nnnn | nnnn |

Description: If the Negative bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example: HERE BNN Jump

Before Instruction
PC = address (HERE)
After Instruction
If Negative = 0;
PC = address (Jump)
If Negative = 1;
PC = address (HERE + 2)


© 2002-2015 Microchip Technology Inc.

**BNOV** **Branch if Not Overflow**

Syntax: [ *label* ] BNOV n

Operands: -128 ≤ n ≤ 127

Operation: if Overflow bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 0101 | nnnn | nnnn |

Description: If the Overflow bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example: HERE BNOV Jump

Before Instruction
PC = address (HERE)
After Instruction
If Overflow = 0;
PC = address (Jump)
If Overflow = 1;
PC = address (HERE + 2)

**BNZ** **Branch if Not Zero**

Syntax: [ *label* ] BNZ n

Operands: -128 ≤ n ≤ 127

Operation: if Zero bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 0001 | nnnn | nnnn |

Description: If the Zero bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example: HERE BNZ Jump

Before Instruction
PC = address (HERE)
After Instruction
If Zero = 0;
PC = address (Jump)
If Zero = 1;
PC = address (HERE + 2)

**BRA** **Unconditional Branch**

Syntax: [ *label* ] BRA n

Operands: $-1024 \le n \le 1023$

Operation: $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1101 | 0nnn | nnnn | nnnn |

Description: Add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a 2-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    BRA  Jump
```
Before Instruction
PC = address (HERE)
After Instruction
PC = address (Jump)

**BSF** **Bit Set f**

Syntax: [ *label* ] BSF f,b[,a]

Operands: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Operation: $1 \rightarrow f<b>$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1000 | bbba | ffff | ffff |

Description: Bit 'b' in register 'f' is set. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example:
```
BSF    FLAG_REG, 7
```
Before Instruction
FLAG_REG = 0x0A
After Instruction
FLAG_REG = 0x8A


© 2002-2015 Microchip Technology Inc.

**BTFSC** **Bit Test File, Skip if Clear**

Syntax: [ *label* ] BTFSC f,b[,a]

Operands: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Operation: skip if (f<b>) = 0

Status Affected: None

Encoding:

| 1011 | bbba | ffff | ffff |
|---|---|---|---|

Description: If bit 'b' in register 'f' is '0', then the next instruction is skipped.
If bit 'b' is '0', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a 2-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    BTFSC   FLAG, 1
FALSE   :
TRUE    :
```

Before Instruction
PC = address (HERE)
After Instruction
If FLAG<1> = 0;
PC = address (TRUE)
If FLAG<1> = 1;
PC = address (FALSE)

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

Syntax: [ *label* ] BTFSS f,b[,a]

Operands: $0 \le f \le 255$
$0 \le b < 7$
$a \in [0,1]$

Operation: skip if (f<b>) = 1

Status Affected: None

Encoding:

| 1010 | bbba | ffff | ffff |
|---|---|---|---|

Description: If bit 'b' in register 'f' is '1', then the next instruction is skipped.
If bit 'b' is '1', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a 2-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    BTFSS   FLAG, 1
FALSE   :
TRUE    :
```

Before Instruction
PC = address (HERE)
After Instruction
If FLAG<1> = 0;
PC = address (FALSE)
If FLAG<1> = 1;
PC = address (TRUE)

**BTG** **Bit Toggle f**

Syntax: [ *label* ] BTG f,b[,a]

Operands: $0 \le f \le 255$
$0 \le b < 7$
$a \in [0,1]$

Operation: $(\overline{f<b>}) \rightarrow f<b>$

Status Affected: None

Encoding:

| 0111 | bbba | ffff | ffff |
|---|---|---|---|

Description: Bit 'b' in data memory location 'f' is inverted. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example:
```
BTG     PORTB,  4
```
Before Instruction:
```
PORTB  =  0111 0101 [0x75]
```
After Instruction:
```
PORTB  =  0110 0101 [0x65]
```

**BOV** **Branch if Overflow**

Syntax: [ *label* ] BOV n

Operands: $-128 \le n \le 127$

Operation: if Overflow bit is '1'
$(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

| 1110 | 0100 | nnnn | nnnn |
|---|---|---|---|

Description: If the Overflow bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example:
```
HERE    BOV  JUMP
```
Before Instruction
```
PC          =   address (HERE)
```
After Instruction
```
If Overflow =   1;
   PC       =   address (JUMP)
If Overflow =   0;
   PC       =   address (HERE + 2)
```


© 2002-2015 Microchip Technology Inc.

**BZ** **Branch if Zero**

Syntax: [ *label* ] BZ n

Operands: $-128 \le n \le 127$

Operation: if Zero bit is '1'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 0000 | nnnn | nnnn |

Description: If the Zero bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example: `HERE BZ Jump`

Before Instruction
PC = address (HERE)
After Instruction
If Zero = 1;
PC = address (Jump)
If Zero = 0;
PC = address (HERE + 2)

**CALL** **Subroutine Call**

Syntax: [ *label* ] CALL k [,s]

Operands: $0 \le k \le 1048575$
$s \in [0,1]$

Operation: (PC) + 4 → TOS,
k → PC<20:1>,
if s = 1
(W) → WS,
(Status) → STATUSS,
(BSR) → BSRS

Status Affected: None

Encoding:

| | | | | |
|---|---|---|---|---|
| 1st word (k<7:0>) | 1110 | 110s | $k_7$kkk | kkkk$_0$ |
| 2nd word(k<19:8>) | 1111 | $k_{19}$kkk | kkkk | kkkk$_8$ |

Description: Subroutine call of entire 2-Mbyte memory range. First, return address (PC + 4) is pushed onto the return stack. If 's' = 1, the W, Status and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If 's' = 0, no update occurs (default). Then, the 20-bit value 'k' is loaded into PC<20:1>. CALL is a 2-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k'<7:0>, | Push PC to stack | Read literal 'k'<19:8>, Write to PC |
| No operation | No operation | No operation | No operation |

Example: `HERE CALL THERE, FAST`

Before Instruction
PC = address (HERE)
After Instruction
PC = address (THERE)
TOS = address (HERE + 4)
WS = W
BSRS = BSR
STATUSS = Status

© 2002-2015 Microchip Technology Inc.

**CLRF** **Clear f**

Syntax: [ *label* ] CLRF f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: 000h → f
1 → Z

Status Affected: Z

Encoding:

| | | | |
|---|---|---|---|
| 0110 | 101a | ffff | ffff |

Description: Clears the contents of the specified register. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: CLRF FLAG_REG

Before Instruction
FLAG_REG = 0x5A
After Instruction
FLAG_REG = 0x00

**CLRWDT** **Clear Watchdog Timer**

Syntax: [ *label* ] CLRWDT

Operands: None

Operation: 000h → WDT,
000h → WDT postscaler,
1 → $\overline{TO}$,
1 → $\overline{PD}$

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

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0100 |

Description: CLRWDT instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits, $\overline{TO}$ and $\overline{PD}$, are set.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Process Data | No operation |

Example: CLRWDT

Before Instruction
WDT Counter = ?
After Instruction
WDT Counter = 0x00
WDT Postscaler = 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1


© 2002-2015 Microchip Technology Inc.

**COMF** **Complement f**

Syntax: [ *label* ] COMF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $(\bar{f}) \rightarrow$ dest

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0001 | 11da | ffff | ffff |

Description: The contents of register 'f' are complemented. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: COMF REG, W

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

**CPFSEQ** **Compare f with W, skip if f = W**

Syntax: [ *label* ] CPFSEQ f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (f) – (W),
skip if (f) = (W)
(unsigned comparison)

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0110 | 001a | ffff | ffff |

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.
If 'f' = W, then the fetched instruction is discarded and a NOP is executed instead, making this a 2-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    CPFSEQ REG
NEQUAL  :
EQUAL   :
```

Before Instruction
PC Address = HERE
W = ?
REG = ?
After Instruction
If REG = W;
PC = Address (EQUAL)
If REG ≠ W;
PC = Address (NEQUAL)

**CPFSGT** **Compare f with W, skip if f > W**

Syntax: [ *label* ] CPFSGT f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (f) – (W),
skip if (f) > (W)
(unsigned comparison)

Status Affected: None

Encoding:

| 0110 | 010a | ffff | ffff |
|---|---|---|---|

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.
If the contents of 'f' are greater than the contents of WREG, then the fetched instruction is discarded and a NOP is executed instead, making this a 2-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE      CPFSGT REG
NGREATER  :
GREATER   :
```

Before Instruction
```
PC  =  Address (HERE)
W   =  ?
```
After Instruction
```
If REG  >  W;
    PC  =  Address (GREATER)
If REG  ≤  W;
    PC  =  Address (NGREATER)
```

**CPFSLT** **Compare f with W, skip if f < W**

Syntax: [ *label* ] CPFSLT f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (f) – (W),
skip if (f) < (W)
(unsigned comparison)

Status Affected: None

Encoding:

| 0110 | 000a | ffff | ffff |
|---|---|---|---|

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.
If the contents of 'f' are less than the contents of W, then the fetched instruction is discarded and a NOP is executed instead, making this a 2-cycle instruction. If 'a' is '0', the Access Bank will be selected. If 'a' is '1', the BSR will not be overridden (default).

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    CPFSLT REG
NLESS   :
LESS    :
```

Before Instruction
```
PC  =  Address (HERE)
W   =  ?
```
After Instruction
```
If REG  <  W;
    PC  =  Address (LESS)
If REG  ≥  W;
    PC  =  Address (NLESS)
```


© 2002-2015 Microchip Technology Inc.

**DAW** **Decimal Adjust W Register**

Syntax: [ *label* ] DAW

Operands: None

Operation: If [W<3:0> > 9] or [DC = 1] then
(W<3:0>) + 6 → W<3:0>;
else
(W<3:0>) → W<3:0>;

If [W<7:4> > 9] or [C = 1] then
(W<7:4>) + 6 → W<7:4>;
else
(W<7:4>) → W<7:4>;

Status Affected: C, DC

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0111 |

Description: DAW adjusts the 8-bit value in W, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result. The Carry bit may be set by DAW regardless of its setting prior to the DAW instruction.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register W | Process Data | Write W |

Example 1: DAW

Before Instruction
W = 0xA5
C = 0
DC = 0

After Instruction
W = 0x05
C = 1
DC = 0

Example 2:

Before Instruction
W = 0xCE
C = 0
DC = 0

After Instruction
W = 0x34
C = 1
DC = 0

**DECF** **Decrement f**

Syntax: [ *label* ] DECF f [,d [,a]]

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (f) – 1 → dest

Status Affected: C, DC, N, OV, Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 01da | ffff | ffff |

Description: Decrement register ‘f’. If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’ (default). If ‘a’ is ‘0’, the Access Bank will be selected, overriding the BSR value. If ‘a’ = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write to destination |

Example: DECF CNT

Before Instruction
CNT = 0x01
Z = 0

After Instruction
CNT = 0x00
Z = 1

**DECFSZ** **Decrement f, skip if 0**

Syntax: [ *label* ] DECFSZ f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – 1 → dest,
skip if result = 0

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0010 | 11da | ffff | ffff |

Description: The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).
If the result is '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a 2-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    DECFSZ  CNT
        GOTO    LOOP
CONTINUE
```
Before Instruction
PC = Address (HERE)
After Instruction
CNT = CNT – 1
If CNT = 0;
PC = Address (CONTINUE)
If CNT ≠ 0;
PC = Address (HERE + 2)

**DCFSNZ** **Decrement f, skip if not 0**

Syntax: [ *label* ] DCFSNZ f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – 1 → dest,
skip if result ≠ 0

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0100 | 11da | ffff | ffff |

Description: The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).
If the result is not '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a 2-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    DCFSNZ TEMP
ZERO    :
NZERO   :
```
Before Instruction
TEMP = ?
After Instruction
TEMP = TEMP – 1,
If TEMP = 0;
PC = Address (ZERO)
If TEMP ≠ 0;
PC = Address (NZERO)


© 2002-2015 Microchip Technology Inc.

**GOTO** **Unconditional Branch**

Syntax: [ *label* ] GOTO k

Operands: $0 \le k \le 1048575$

Operation: $k \rightarrow PC<20:1>$

Status Affected: None

Encoding:

| | | | | |
|---|---|---|---|---|
| 1st word (k<7:0>) | 1110 | 1111 | $k_7kkk$ | $kkkk_0$ |
| 2nd word(k<19:8>) | 1111 | $k_{19}kkk$ | kkkk | $kkkk_8$ |

Description: GOTO allows an unconditional branch anywhere within the entire 2-Mbyte memory range. The 20-bit value 'k' is loaded into PC<20:1>. GOTO is always a 2-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k'<7:0>, | No operation | Read literal 'k'<19:8>, Write to PC |
| No operation | No operation | No operation | No operation |

Example: GOTO THERE

After Instruction
PC = Address (THERE)

**INCF** **Increment f**

Syntax: [ *label* ] INCF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) + 1 → dest

Status Affected: C, DC, N, OV, Z

Encoding:

| 0010 | 10da | ffff | ffff |
|---|---|---|---|

Description: The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: INCF CNT

Before Instruction
CNT = 0xFF
Z = 0
C = ?
DC = ?

After Instruction
CNT = 0x00
Z = 1
C = 1
DC = 1

**INCFSZ** **Increment f, skip if 0**

Syntax: [ *label* ] INCFSZ f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

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

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0011 | 11da | ffff | ffff |

Description: The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).
If the result is '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a 2-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    INCFSZ   CNT
NZERO   :
ZERO    :
```

Before Instruction
PC = Address (HERE)

After Instruction
CNT = CNT + 1
If CNT = 0;
PC = Address (ZERO)
If CNT ≠ 0;
PC = Address (NZERO)

**INFSNZ** **Increment f, skip if not 0**

Syntax: [ *label* ] INFSNZ f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) + 1 → dest,
skip if result ≠ 0

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0100 | 10da | ffff | ffff |

Description: The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).
If the result is not '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a 2-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    INFSNZ REG
ZERO
NZERO
```

Before Instruction
PC = Address (HERE)

After Instruction
REG = REG + 1
If REG ≠ 0;
PC = Address (NZERO)
If REG = 0;
PC = Address (ZERO)


© 2002-2015 Microchip Technology Inc.

**IORLW** **Inclusive OR literal with W**

Syntax: [ *label* ] IORLW k

Operands: $0 \le k \le 255$

Operation: (W) .OR. k → W

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1001 | kkkk | kkkk |

Description: The contents of W are OR'ed with the 8-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: IORLW 0x35

Before Instruction
W = 0x9A

After Instruction
W = 0xBF

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

Syntax: [ *label* ] IORWF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (W) .OR. (f) → dest

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0001 | 00da | ffff | ffff |

Description: Inclusive OR W with register 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: IORWF RESULT, W

Before Instruction
RESULT = 0x13
W = 0x91

After Instruction
RESULT = 0x13
W = 0x93

**LFSR** **Load FSR**

Syntax: [ *label* ] LFSR f,k

Operands: $0 \le f \le 2$
$0 \le k \le 4095$

Operation: $k \rightarrow FSRf$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 1111 | 1110 0000 | 00ff $k_7$kkk | $k_{11}$kkk kkkk |

Description: The 12-bit literal ‘k’ is loaded into the file select register pointed to by ‘f’.

Words: 2

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal ‘k’ MSB | Process Data | Write literal ‘k’ MSB to FSRfH |
| Decode | Read literal ‘k’ LSB | Process Data | Write literal ‘k’ to FSRfL |

Example: `LFSR 2, 0x3AB`

After Instruction
FSR2H = 0x03
FSR2L = 0xAB

**MOVF** **Move f**

Syntax: [ *label* ] MOVF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $f \rightarrow dest$

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0101 | 00da | ffff | ffff |

Description: The contents of register ‘f’ are moved to a destination dependent upon the status of ‘d’. If ‘d’ is ‘f’, the result is placed in W. If ‘d’ is ‘f’, the result is placed back in register ‘f’ (default). Location ‘f’ can be anywhere in the 256-byte bank. If ‘a’ is ‘0’, the Access Bank will be selected, overriding the BSR value. If ‘a’ = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write W |

Example: `MOVF REG, W`

Before Instruction
REG = 0x22
W = 0xFF

After Instruction
REG = 0x22
W = 0x22


© 2002-2015 Microchip Technology Inc.

**MOVFF** **Move f to f**

Syntax: [ *label* ] MOVFF $f_s$,$f_d$

Operands: $0 \le f_s \le 4095$
$0 \le f_d \le 4095$

Operation: $(f_s) \rightarrow f_d$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1st word (source) | 1100 | ffff | ffff | $ffff_s$ |
| 2nd word (destin.) | 1111 | ffff | ffff | $ffff_d$ |

Description: The contents of source register '$f_s$' are moved to destination register '$f_d$'. Location of source '$f_s$' can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination '$f_d$' can also be anywhere from 000h to FFFh. Either source or destination can be W (a useful special situation). MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).

The MOVFF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

The MOVFF instruction should not be used to modify interrupt settings while any interrupt is enabled (see page 70).

Words: 2

Cycles: 2 (3)

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' (src) | Process Data | No operation |
| Decode | No operation<br>No dummy read | No operation | Write register 'f' (dest) |

Example: MOVFF REG1, REG2

Before Instruction
REG1 = 0x33
REG2 = 0x11

After Instruction
REG1 = 0x33,
REG2 = 0x33

**MOVLB** **Move literal to low nibble in BSR**

Syntax: [ *label* ] MOVLB k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow BSR$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0001 | kkkk | kkkk |

Description: The 8-bit literal 'k' is loaded into the Bank Select Register (BSR).

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write literal 'k' to BSR |

Example: MOVLB 5

Before Instruction
BSR register = 0x02

After Instruction
BSR register = 0x05

**MOVLW** **Move literal to W**

Syntax: [ *label* ] MOVLW k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow W$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1110 | kkkk | kkkk |

Description: The 8-bit literal ‘k’ is loaded into W.

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 [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: $(W) \rightarrow f$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0110 | 111a | ffff | ffff |

Description: Move data from W to register ‘f’. Location ‘f’ can be anywhere in the 256-byte bank. If ‘a’ is ‘0’, the Access Bank will be selected, overriding the BSR value. If ‘a’ = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write register ‘f’ |

Example: MOVWF REG

Before Instruction

W = 0x4F
REG = 0xFF

After Instruction

W = 0x4F
REG = 0x4F


© 2002-2015 Microchip Technology Inc.

**MULLW** **Multiply Literal with W**

Syntax: [ *label* ] MULLW k

Operands: $0 \le k \le 255$

Operation: (W) x k → PRODH:PRODL

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1101 | kkkk | kkkk |

Description: An unsigned multiplication is carried out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in the PRODH:PRODL register pair. PRODH contains the high byte. W is unchanged.
None of the Status flags are affected.
Note that neither Overflow nor Carry is possible in this operation. A Zero result is possible but not detected.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write registers PRODH: PRODL |

Example: `MULLW 0xC4`

Before Instruction
W = 0xE2
PRODH = ?
PRODL = ?

After Instruction
W = 0xE2
PRODH = 0xAD
PRODL = 0x08

**MULWF** **Multiply W with f**

Syntax: [ *label* ] MULWF f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (W) x (f) → PRODH:PRODL

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 001a | ffff | ffff |

Description: An unsigned multiplication is carried out between the contents of W and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte.
Both W and 'f' are unchanged.
None of the Status flags are affected.
Note that neither Overflow nor Carry is possible in this operation. A Zero result is possible, but not detected. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write registers PRODH: PRODL |

Example: `MULWF REG`

Before Instruction
W = 0xC4
REG = 0xB5
PRODH = ?
PRODL = ?

After Instruction
W = 0xC4
REG = 0xB5
PRODH = 0x8A
PRODL = 0x94

**NEGF** **Negate f**

Syntax: [ *label* ] NEGF f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: $(\bar{f}) + 1 \rightarrow f$

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0110 | 110a | ffff | ffff |

Description: Location ‘f’ is negated using two’s complement. The result is placed in the data memory location ‘f’. If ‘a’ is ‘0’, the Access Bank will be selected, overriding the BSR value. If ‘a’ = 1, then the bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write register ‘f’ |

Example:
```
NEGF    REG, 1
```
Before Instruction
```
REG     =   0011 1010 [0x3A]
```
After Instruction
```
REG     =   1100 0110 [0xC6]
```

**NOP** **No Operation**

Syntax: [ *label* ] NOP

Operands: None

Operation: No operation

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 1111 | 0000 xxxx | 0000 xxxx | 0000 xxxx |

Description: No operation.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | No operation |

Example:

None.


© 2002-2015 Microchip Technology Inc.

**POP** **Pop Top of Return Stack**

Syntax: [ *label* ] POP

Operands: None

Operation: (TOS) → bit bucket

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0110 |

Description: The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack.
This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Pop TOS value | No operation |

Example:
```
POP
GOTO    NEW
```
Before Instruction
TOS = 0x0031A2
Stack (1 level down) = 0x014332

After Instruction
TOS = 0x014332
PC = NEW

**PUSH** **Push Top of Return Stack**

Syntax: [ *label* ] PUSH

Operands: None

Operation: (PC + 2) → TOS

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0101 |

Description: The PC + 2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack.
This instruction allows implementing a software stack by modifying TOS and then pushing it onto the return stack.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Push PC + 2 onto return stack | No operation | No operation |

Example:
```
PUSH
```
Before Instruction
TOS = 0x00345A
PC = 0x000124

After Instruction
PC = 0x000126
TOS = 0x000126
Stack (1 level down) = 0x00345A

**RCALL** **Relative Call**

Syntax: [ *label* ] RCALL n

Operands: $-1024 \leq n \leq 1023$

Operation: $(PC) + 2 \rightarrow TOS$,
$(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1101 | 1nnn | nnnn | nnnn |

Description: Subroutine call with a jump up to 1K from the current location. First, return address (PC + 2) is pushed onto the stack. Then, add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a 2-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n'<br>Push PC to stack | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

Example: HERE RCALL Jump

Before Instruction
PC = Address (HERE)

After Instruction
PC = Address (Jump)
TOS = Address (HERE + 2)

**RESET** **Reset**

Syntax: [ *label* ] RESET

Operands: None

Operation: Reset all registers and flags that are affected by a $\overline{MCLR}$ Reset.

Status Affected: All

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 1111 | 1111 |

Description: This instruction provides a way to execute a $\overline{MCLR}$ Reset in software.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Start Reset | No operation | No operation |

Example: RESET

After Instruction
Registers = Reset Value
Flags* = Reset Value


© 2002-2015 Microchip Technology Inc.

**RETFIE** **Return from Interrupt**

Syntax: [ *label* ] RETFIE [s]

Operands: s ∈ [0,1]

Operation: (TOS) → PC,
1 → GIE/GIEH or PEIE/GIEL,
if s = 1
(WS) → W,
(STATUSS) → Status,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged.

Status Affected: GIE/GIEH, PEIE/GIEL.

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0001 | 000s |

Description: Return from interrupt. Stack is popped and Top-of-Stack (TOS) is loaded into the PC. Interrupts are enabled by setting either the high or low priority Global Interrupt Enable bit. If 's' = 1, the contents of the shadow registers, WS, STATUSS and BSRS, are loaded into their corresponding registers, W, Status and BSR. If 's' = 0, no update of these registers occurs (default).

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | Pop PC from stack<br>Set GIEH or GIEL |
| No operation | No operation | No operation | No operation |

Example: RETFIE 1

After Interrupt
PC = TOS
W = WS
BSR = BSRS
Status = STATUSS
GIE/GIEH, PEIE/GIEL = 1

**RETLW** **Return Literal to W**

Syntax: [ *label* ] RETLW k

Operands: 0 ≤ k ≤ 255

Operation: k → W,
(TOS) → PC,
PCLATU, PCLATH are unchanged

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1100 | kkkk | kkkk |

Description: W is loaded with the 8-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). The high address latch (PCLATH) remains unchanged.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Pop PC from stack, Write to W |
| No operation | No operation | No operation | No operation |

Example:

```
    CALL TABLE ; W contains table
               ; offset value
               ; W now has
               ; table value
    :
TABLE
    ADDWF PCL  ; W = offset
    RETLW k0   ; Begin table
    RETLW k1   ;
    :
    :
    RETLW kn   ; End of table
```

Before Instruction
W = 0x07
After Instruction
W = value of kn

**RETURN** **Return from Subroutine**

Syntax: [ *label* ] RETURN [s]

Operands: s ∈ [0,1]

Operation: (TOS) → PC,
if s = 1
(WS) → W,
(STATUSS) → Status,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0001 | 001s |

Description: Return from subroutine. The stack is popped and the top of the stack is loaded into the program counter. If 's'= 1, the contents of the shadow registers, WS, STATUSS and BSRS, are loaded into their corresponding registers, W, Status and BSR. If 's' = 0, no update of these registers occurs (default).

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Process Data | Pop PC from stack |
| No operation | No operation | No operation | No operation |

Example: RETURN

After Interrupt
PC = TOS

**RLCF** **Rotate Left f through Carry**

Syntax: [ *label* ] RLCF f [,d [,a]]

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

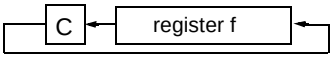
Operation: (f<n>) → dest<n + 1>,
(f<7>) → C,
(C) → dest<0>

Status Affected: C, N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0011 | 01da | ffff | ffff |

Description: The contents of register 'f' are rotated one bit to the left through the Carry flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).




Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: RLCF REG, W

Before Instruction
REG = 1110 0110
C = 0

After Instruction
REG = 1110 0110
W = 1100 1100
C = 1


© 2002-2015 Microchip Technology Inc.

**RLNCF** **Rotate Left f (no carry)**

Syntax: [ *label* ] RLNCF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

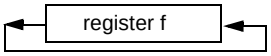
Operation: (f<n>) → dest<n + 1>,
(f<7>) → dest<0>

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0100 | 01da | ffff | ffff |

Description: The contents of register ‘f’ are rotated one bit to the left. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’ (default). If ‘a’ is ‘0’, the Access Bank will be selected, overriding the BSR value. If ‘a’ is ‘1’, then the bank will be selected as per the BSR value (default).




Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write to destination |

Example: RLNCF REG

Before Instruction
REG = 1010 1011
After Instruction
REG = 0101 0111

**RRCF** **Rotate Right f through Carry**

Syntax: [ *label* ] RRCF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

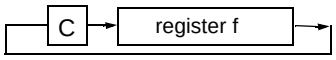
Operation: (f<n>) → dest<n – 1>,
(f<0>) → C,
(C) → dest<7>

Status Affected: C, N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0011 | 00da | ffff | ffff |

Description: The contents of register ‘f’ are rotated one bit to the right through the Carry flag. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is placed back in register ‘f’ (default). If ‘a’ is ‘0’, the Access Bank will be selected, overriding the BSR value. If ‘a’ is ‘1’, then the bank will be selected as per the BSR value (default).




Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write to destination |

Example: RRCF REG, W

Before Instruction
REG = 1110 0110
C = 0
After Instruction
REG = 1110 0110
W = 0111 0011
C = 0

**RRNCF** **Rotate Right f (no carry)**

Syntax: [ *label* ] RRNCF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

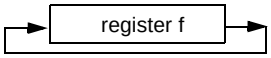
Operation: (f<n>) → dest<n – 1>,
(f<0>) → dest<7>

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0100 | 00da | ffff | ffff |

Description: The contents of register ‘f’ are rotated one bit to the right. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is placed back in register ‘f’ (default). If ‘a’ is ‘0’, the Access Bank will be selected, overriding the BSR value. If ‘a’ is ‘1’, then the bank will be selected as per the BSR value (default).




Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write to destination |

Example 1: RRNCF REG, 1, 0

Before Instruction
REG = 1101 0111
After Instruction
REG = 1110 1011

Example 2: RRNCF REG, W

Before Instruction
W = ?
REG = 1101 0111
After Instruction
W = 1110 1011
REG = 1101 0111

**SETF** **Set f**

Syntax: [ *label* ] SETF f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: FFh → f

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0110 | 100a | ffff | ffff |

Description: The contents of the specified register are set to FFh. If ‘a’ is ‘0’, the Access Bank will be selected, overriding the BSR value. If ‘a’ is ‘1’, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write register ‘f’ |

Example: SETF REG

Before Instruction
REG = 0x5A
After Instruction
REG = 0xFF


© 2002-2015 Microchip Technology Inc.

**SLEEP** **Enter Sleep mode**

Syntax: [ *label* ] SLEEP

Operands: None

Operation: 00h → WDT,
0 → WDT postscaler,
1 → $\overline{TO}$,
0 → $\overline{PD}$

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

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0011 |

Description: The Power-down Status bit ($\overline{PD}$) is cleared. The Time-out status bit ($\overline{TO}$) is set. The Watchdog Timer and its postscaler are cleared. The processor is put into Sleep mode with the oscillator stopped.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Process Data | Go to Sleep |

Example: SLEEP

Before Instruction
$\overline{TO}$ = ?
$\overline{PD}$ = ?

After Instruction
$\overline{TO}$ = 1 †
$\overline{PD}$ = 0

† If WDT causes wake-up, this bit is cleared.

**SUBFWB** **Subtract f from W with borrow**

Syntax: [ *label* ] SUBFWB f [,d [,a]]

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (W) – (f) – ($\overline{C}$) → dest

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0101 | 01da | ffff | ffff |

Description: Subtract register 'f' and Carry flag (borrow) from W (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored in register 'f' (default). If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example 1: SUBFWB REG

Before Instruction
REG = 0x03
W = 0x02
C = 0x01

After Instruction
REG = 0xFF
W = 0x02
C = 0x00
Z = 0x00
N = 0x01 ; result is negative

Example 2: SUBFWB REG, 0, 0

Before Instruction
REG = 2
W = 5
C = 1

After Instruction
REG = 2
W = 3
C = 1
Z = 0
N = 0 ; result is positive

Example 3: SUBFWB REG, 1, 0

Before Instruction
REG = 1
W = 2
C = 0

After Instruction
REG = 0
W = 2
C = 1
Z = 1 ; result is zero
N = 0

**SUBLW** **Subtract W from literal**

Syntax: [ *label* ] SUBLW k

Operands: $0 \le k \le 255$

Operation: $k - (W) \rightarrow W$

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1000 | kkkk | kkkk |

Description: W is subtracted from the 8-bit literal ‘k’. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal ‘k’ | Process Data | Write to W |

Example 1: `SUBLW 0x02`

Before Instruction
W = 1
C = ?

After Instruction
W = 1
C = 1 ; result is positive
Z = 0
N = 0

Example 2: `SUBLW 0x02`

Before Instruction
W = 2
C = ?

After Instruction
W = 0
C = 1 ; result is zero
Z = 1
N = 0

Example 3: `SUBLW 0x02`

Before Instruction
W = 3
C = ?

After Instruction
W = FF ; (2’s complement)
C = 0 ; result is negative
Z = 0
N = 1

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

Syntax: [ *label* ] SUBWF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $(f) - (W) \rightarrow dest$

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0101 | 11da | ffff | ffff |

Description: Subtract W from register ‘f’ (2’s complement method). If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’ (default). If ‘a’ is ‘0’, the Access Bank will be selected, overriding the BSR value. If ‘a’ is ‘1’, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write to destination |

Example 1: `SUBWF REG`

Before Instruction
REG = 3
W = 2
C = ?

After Instruction
REG = 1
W = 2
C = 1 ; result is positive
Z = 0
N = 0

Example 2: `SUBWF REG, W`

Before Instruction
REG = 2
W = 2
C = ?

After Instruction
REG = 2
W = 0
C = 1 ; result is zero
Z = 1
N = 0

Example 3: `SUBWF REG`

Before Instruction
REG = 0x01
W = 0x02
C = ?

After Instruction
REG = 0xFFh ;(2’s complement)
W = 0x02
C = 0x00 ;result is negative
Z = 0x00
N = 0x01


© 2002-2015 Microchip Technology Inc.

**SUBWFB** **Subtract W from f with Borrow**

Syntax: [ *label* ] SUBWFB f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – (W) – ($\overline{C}$) → dest

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0101 | 10da | ffff | ffff |

Description: Subtract W and the Carry flag (borrow) from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example 1:
```
SUBWFB  REG, 1, 0
```
Before Instruction
```
REG  =  0x19   (0001 1001)
W    =  0x0D   (0000 1101)
C    =  0x01
```
After Instruction
```
REG  =  0x0C   (0000 1011)
W    =  0x0D   (0000 1101)
C    =  0x01
Z    =  0x00
N    =  0x00   ; result is positive
```

Example 2:
```
SUBWFB  REG, 0, 0
```
Before Instruction
```
REG  =  0x1B   (0001 1011)
W    =  0x1A   (0001 1010)
C    =  0x00
```
After Instruction
```
REG  =  0x1B   (0001 1011)
W    =  0x00
C    =  0x01
Z    =  0x01   ; result is zero
N    =  0x00
```

Example 3:
```
SUBWFB  REG, 1, 0
```
Before Instruction
```
REG  =  0x03   (0000 0011)
W    =  0x0E   (0000 1101)
C    =  0x01
```
After Instruction
```
REG  =  0xF5   (1111 0100)
                ; [2's comp]
W    =  0x0E   (0000 1101)
C    =  0x00
Z    =  0x00
N    =  0x01   ; result is negative
```

**SWAPF** **Swap f**

Syntax: [ *label* ] SWAPF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f<3:0>) → dest<7:4>,
(f<7:4>) → dest<3:0>

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0011 | 10da | ffff | ffff |

Description: The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in register 'f' (default). If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example:
```
SWAPF   REG
```
Before Instruction
```
REG  =  0x53
```
After Instruction
```
REG  =  0x35
```

**TBLRD** **Table Read**

Syntax: [ *label* ] TBLRD ( *; *+; *-; +*)

Operands: None

Operation: if TBLRD *,
(Prog Mem (TBLPTR)) → TABLAT;
TBLPTR – No Change;
if TBLRD *+,
(Prog Mem (TBLPTR)) → TABLAT;
(TBLPTR) + 1 → TBLPTR;
if TBLRD *-,
(Prog Mem (TBLPTR)) → TABLAT;
(TBLPTR) – 1 → TBLPTR;
if TBLRD +*,
(TBLPTR) + 1 → TBLPTR;
(Prog Mem (TBLPTR)) → TABLAT;

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 10nn<br>nn = 0*<br>= 1*+<br>= 2*-<br>= 3+* |

Description: This instruction is used to read the contents of Program Memory (P.M.). To address the program memory, a pointer called Table Pointer (TBLPTR) is used. The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word

TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The TBLRD instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | No operation |
| No operation | No operation (Read Program Memory) | No operation | No operation (Write TABLAT) |

**TBLRD** **Table Read (Continued)**

Example 1: `TBLRD *+ ;`

Before Instruction

| | | |
|---|---|---|
| TABLAT | = | 0x55 |
| TBLPTR | = | 0x00A356 |
| MEMORY(0x00A356) | = | 0x34 |

After Instruction

| | | |
|---|---|---|
| TABLAT | = | 0x34 |
| TBLPTR | = | 0x00A357 |

Example 2: `TBLRD +* ;`

Before Instruction

| | | |
|---|---|---|
| TABLAT | = | 0xAA |
| TBLPTR | = | 0x01A357 |
| MEMORY(0x01A357) | = | 0x12 |
| MEMORY(0x01A358) | = | 0x34 |

After Instruction

| | | |
|---|---|---|
| TABLAT | = | 0x34 |
| TBLPTR | = | 0x01A358 |


© 2002-2015 Microchip Technology Inc.

**TBLWT** **Table Write**

Syntax: [ *label* ] TBLWT ( *; *+; *-; +*)

Operands: None

Operation: if TBLWT*,
(TABLAT) → Holding Register;
TBLPTR – No Change;
if TBLWT*+,
(TABLAT) → Holding Register;
(TBLPTR) + 1 → TBLPTR;
if TBLWT*-,
(TABLAT) → Holding Register;
(TBLPTR) – 1 → TBLPTR;
if TBLWT+*,
(TBLPTR) + 1 → TBLPTR;
(TABLAT) → Holding Register;

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 11nn<br>nn = 0*<br>= 1*+<br>= 2*-<br>= 3+* |

Description: This instruction uses the 3 LSBs of TBLPTR to determine which of the eight holding registers the TABLAT is written to. The holding registers are used to program the contents of Program Memory (P.M.). (Refer to **Section 6.0 "Flash Program Memory"** for additional details on programming Flash memory.)
The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range. The LSb of the TBLPTR selects which byte of the program memory location to access.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word

TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The `TBLWT` instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

**TBLWT** **Table Write (Continued)**

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | No operation |
| No operation | No operation (Read TABLAT) | No operation | No operation (Write to Holding Register) |

Example 1: `TBLWT *+;`

Before Instruction
TABLAT = 0x55
TBLPTR = 0x00A356
HOLDING REGISTER (0x00A356) = 0xFF

After Instructions (table write completion)
TABLAT = 0x55
TBLPTR = 0x00A357
HOLDING REGISTER (0x00A356) = 0x55

Example 2: `TBLWT +*;`

Before Instruction
TABLAT = 0x34
TBLPTR = 0x01389A
HOLDING REGISTER (0x01389A) = 0xFF
HOLDING REGISTER (0x01389B) = 0xFF

After Instruction (table write completion)
TABLAT = 0x34
TBLPTR = 0x01389B
HOLDING REGISTER (0x01389A) = 0xFF
HOLDING REGISTER (0x01389B) = 0x34

**TSTFSZ** **Test f, skip if 0**

Syntax: [ *label* ] TSTFSZ f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: skip if f = 0

Status Affected: None

Encoding:

| 0110 | 011a | ffff | ffff |
|---|---|---|---|

Description: If 'f' = 0, the next instruction, fetched during the current instruction execution is discarded and a NOP is executed, making this a 2-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    TSTFSZ CNT
NZERO   :
ZERO    :
```

Before Instruction
PC = Address (HERE)

After Instruction
If CNT = 0x00,
PC = Address (ZERO)
If CNT ≠ 0x00,
PC = Address (NZERO)

**XORLW** **Exclusive OR literal with W**

Syntax: [ *label* ] XORLW k

Operands: $0 \le k \le 255$

Operation: (W) .XOR. k → W

Status Affected: N, Z

Encoding:

| 0000 | 1010 | kkkk | kkkk |
|---|---|---|---|

Description: The contents of W are XOR'ed with the 8-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: XORLW 0xAF

Before Instruction
W = 0xB5

After Instruction
W = 0x1A


© 2002-2015 Microchip Technology Inc.

**XORWF** **Exclusive OR W with f**

Syntax: [ *label* ] XORWF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (W) .XOR. (f) → dest

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0001 | 10da | ffff | ffff |

Description: Exclusive OR the contents of W with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f' (default). If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: XORWF REG

Before Instruction
REG = 0xAF
W = 0xB5

After Instruction
REG = 0x1A
W = 0xB5

## 21.0 DEVELOPMENT SUPPORT

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

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

### 21.1 MPLAB X Integrated Development Environment Software

The MPLAB X IDE is a single, unified graphical user interface for Microchip and third-party software, and hardware development tool that runs on Windows®, Linux and Mac OS® X. Based on the NetBeans IDE, MPLAB X IDE is an entirely new IDE with a host of free software components and plug-ins for high-performance application development and debugging. Moving between tools and upgrading from software simulators to hardware debugging and programming tools is simple with the seamless user interface.

With complete project management, visual call graphs, a configurable watch window and a feature-rich editor that includes code completion and context menus, MPLAB X IDE is flexible and friendly enough for new users. With the ability to support multiple tools on multiple projects with simultaneous debugging, MPLAB X IDE is also suitable for the needs of experienced users.

Feature-Rich Editor:

- Color syntax highlighting
- Smart code completion makes suggestions and provides hints as you type
- Automatic code formatting based on user-defined rules
- Live parsing

User-Friendly, Customizable Interface:

- Fully customizable interface: toolbars, toolbar buttons, windows, window placement, etc.
- Call graph window

Project-Based Workspaces:

- Multiple projects
- Multiple tools
- Multiple configurations
- Simultaneous debugging sessions

File History and Bug Tracking:

- Local file history feature
- Built-in support for Bugzilla issue tracker


© 2002-2015 Microchip Technology Inc.

## 21.2 MPLAB XC Compilers

The MPLAB XC Compilers are complete ANSI C compilers for all of Microchip's 8, 16, and 32-bit MCU and DSC devices. These compilers provide powerful integration capabilities, superior code optimization and ease of use. MPLAB XC Compilers run on Windows, Linux or MAC OS X.

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

The free MPLAB XC Compiler editions support all devices and commands, with no time or memory restrictions, and offer sufficient code optimization for most applications.

MPLAB XC Compilers include an assembler, linker and utilities. 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. MPLAB XC Compiler uses the assembler to produce its object 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 X IDE compatibility

## 21.3 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 X IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multipurpose source files
- Directives that allow complete control over the assembly process

## 21.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler. 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

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

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC DSC devices. MPLAB XC 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 X IDE compatibility

## 21.6 MPLAB X SIM Software Simulator

The MPLAB X SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC 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 X SIM Software Simulator fully supports symbolic debugging using the MPLAB XC 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.

## 21.7 MPLAB REAL ICE In-Circuit Emulator System

The 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 all 8, 16 and 32-bit MCU, and DSC devices with the easy-to-use, powerful graphical user interface of the MPLAB X IDE.

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 (RJ-11) 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 X IDE. MPLAB REAL ICE offers significant advantages over competitive emulators including full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, logic probes, a ruggedized probe interface and long (up to three meters) interconnection cables.

## 21.8 MPLAB ICD 3 In-Circuit Debugger System

The MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost-effective, high-speed hardware debugger/programmer for Microchip Flash DSC and MCU devices. It debugs and programs PIC Flash microcontrollers and dsPIC DSCs with the powerful, yet easy-to-use graphical user interface of the MPLAB 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.

## 21.9 PICkit 3 In-Circuit Debugger/ Programmer

The MPLAB PICkit 3 allows debugging and programming of PIC and dsPIC Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB 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™ (ICSP™).

## 21.10 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at $V_{DDMIN}$ and $V_{DDMAX}$ for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages, and a modular, detachable socket assembly to support various package types. The ICSP cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices, and incorporates an MMC card for file storage and data applications.


© 2002-2015 Microchip Technology Inc.

## 21.11 Demonstration/Development Boards, Evaluation Kits, and Starter Kits

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, KEELOQ® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Also available are starter kits that contain everything needed to experience the specified device. This usually includes a single application and debug capability, all on one board.

Check the Microchip web page (www.microchip.com) for the complete list of demonstration, development and evaluation kits.

## 21.12 Third-Party Development Tools

Microchip also offers a great collection of tools from third-party vendors. These tools are carefully selected to offer good value and unique functionality.

- Device Programmers and Gang Programmers from companies, such as SoftLog and CCS
- Software Tools from companies, such as Gimpel and Trace Systems
- Protocol Analyzers from companies, such as Saleae and Total Phase
- Demonstration Boards from companies, such as MikroElektronika, Digilent® and Olimex
- Embedded Ethernet Solutions from companies, such as EZ Web Lynx, WIZnet and IPLogika®

## 22.0 ELECTRICAL CHARACTERISTICS

### Absolute Maximum Ratings(†)

Ambient temperature under bias ..... -40°C to +125°C
Storage temperature ..... -65°C to +150°C
Voltage on any pin with respect to $V_{SS}$ (except $V_{DD}$, $\overline{MCLR}$ and RA4) ..... -0.3V to ($V_{DD}$ + 0.3V)
Voltage on $V_{DD}$ with respect to $V_{SS}$ ..... -0.3V to +5.5V
Voltage on $\overline{MCLR}$ with respect to $V_{SS}$ **(Note 2)** ..... 0V to +13.25V
Voltage on RA4 with respect to Vss ..... 0V to +8.5V
Total power dissipation **(Note 1)** ..... 1.0W
Maximum current out of $V_{SS}$ pin ..... 300 mA
Maximum current into $V_{DD}$ pin ..... 250 mA
Input clamp current, $I_{IK}$ ($V_I$ < 0 or $V_I$ > $V_{DD}$) ..... ±20 mA
Output clamp current, $I_{OK}$ ($V_O$ < 0 or $V_O$ > $V_{DD}$) ..... ±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 all ports ..... 200 mA
Maximum current sourced by all ports ..... 200 mA

**Note 1:** Power dissipation is calculated as follows:
$Pdis = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum(V_{OL} \times I_{OL})$

**2:** Voltage spikes below $V_{SS}$ at the $\overline{MCLR}/V_{PP}$ 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 $\overline{MCLR}/V_{PP}$ pin, rather than pulling this pin directly to $V_{SS}$.

† 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.


© 2002-2015 Microchip Technology Inc.

**FIGURE 22-1: PIC18F1220/1320 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)**

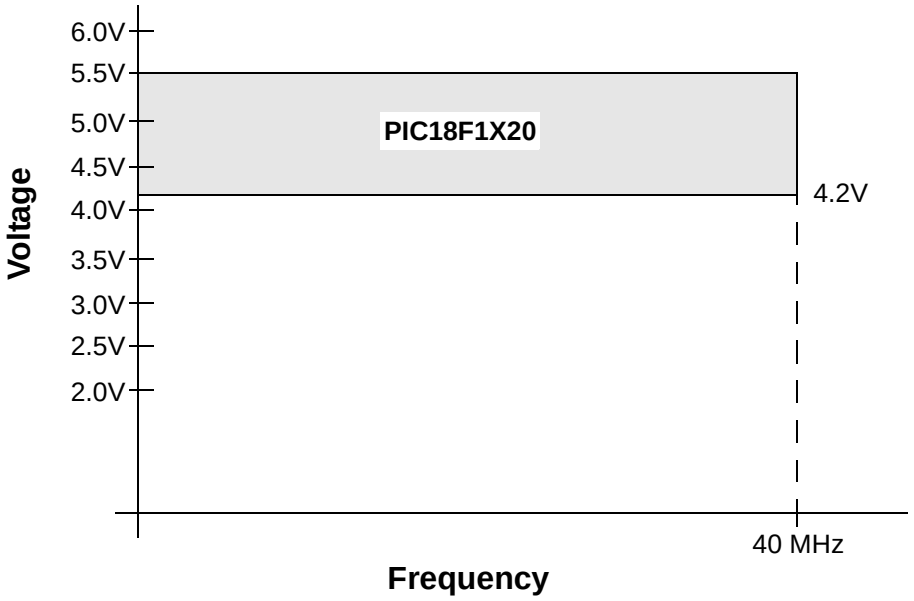
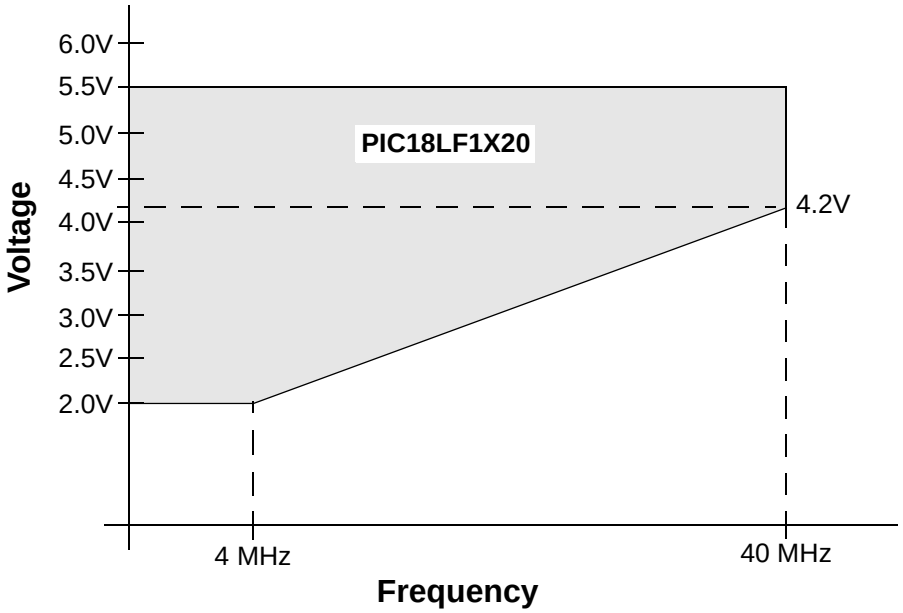


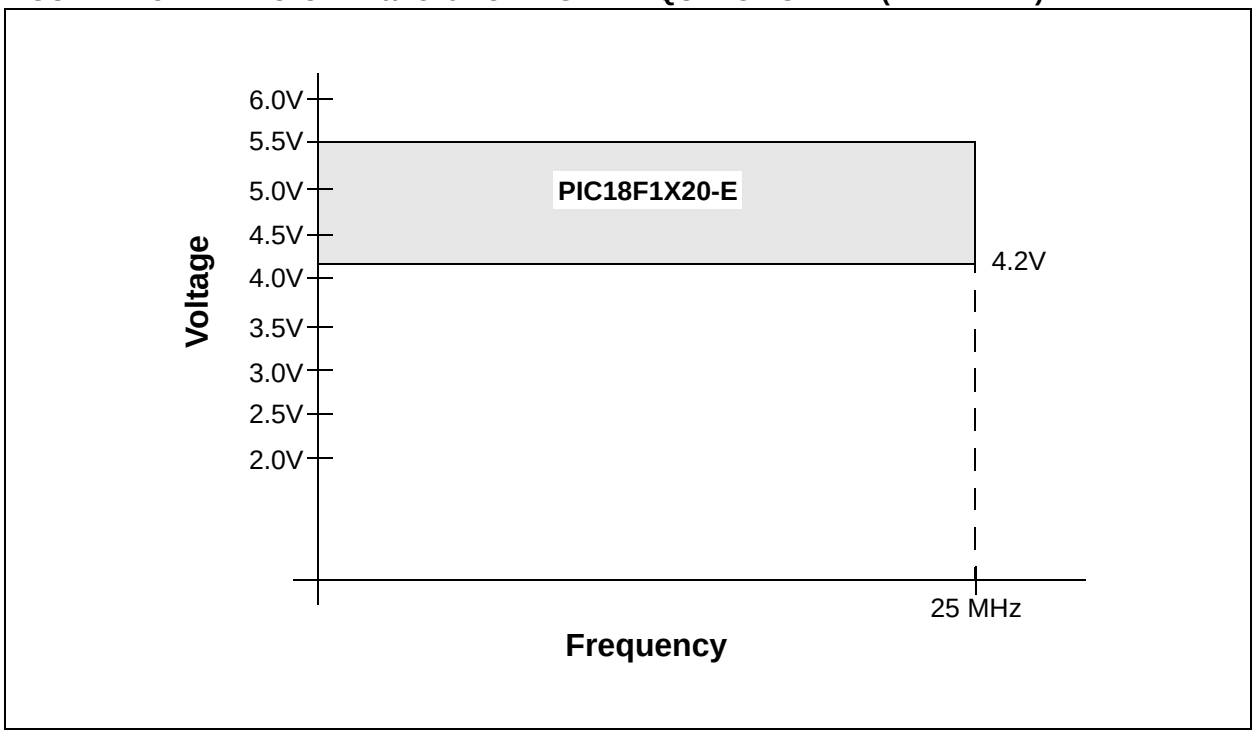

**FIGURE 22-2: PIC18LF1220/1320 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)**




$F_{MAX}$ = (16.36 MHz/V) ($V_{DDAPPMIN}$ – 2.0V) + 4 MHz

**Note:** $V_{DDAPPMIN}$ is the minimum voltage of the PIC® device in the application.

**FIGURE 22-3: PIC18F1220/1320 VOLTAGE-FREQUENCY GRAPH (EXTENDED)**





© 2002-2015 Microchip Technology Inc.

## 22.1 DC Characteristics: Supply Voltage PIC18F1220/1320 (Industrial) PIC18LF1220/1320 (Industrial)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **PIC18LF1220/1320** (Industrial) | | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | |
| **PIC18F1220/1320** (Industrial, Extended) | | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended | | | | |
| **Param No.** | **Symbol** | **Characteristic** | **Min.** | **Typ.** | **Max.** | **Units** | **Conditions** |
| | VDD | **Supply Voltage** | | | | | |
| D001 | | PIC18LF1220/1320 | 2.0 | — | 5.5 | V | HS, XT, RC and LP Oscillator mode |
| | | PIC18F1220/1320 | 4.2 | — | 5.5 | V | |
| D002 | VDR | **RAM Data Retention Voltage(1)** | 1.5 | — | — | V | |
| D003 | VPOR | **VDD Start Voltage** to ensure internal Power-on Reset signal | — | — | 0.7 | V | See **Section 4.1 "Power-on Reset (POR)"** for details. |
| D004 | SVDD | **VDD Rise Rate** to ensure internal Power-on Reset signal | 0.05 | — | — | V/ms | See **Section 4.1 "Power-on Reset (POR)"** for details. |
| | VBOR | **Brown-out Reset Voltage** | | | | | |
| D005D | | PIC18LF1220/1320 | Industrial Low Voltage **(-10°C to +85°C)** | | | | |
| | | BORV1:BORV0 = 11 | N/A | N/A | N/A | V | Reserved |
| | | BORV1:BORV0 = 10 | 2.50 | 2.72 | 2.94 | V | |
| | | BORV1:BORV0 = 01 | 3.88 | 4.22 | 4.56 | V | **(Note 2)** |
| | | BORV1:BORV0 = 00 | 4.18 | 4.54 | 4.90 | V | **(Note 2)** |
| D005F | | PIC18LF1220/1320 | Industrial Low Voltage **(-40°C to -10°C)** | | | | |
| | | BORV1:BORV0 = 11 | N/A | N/A | N/A | V | Reserved |
| | | BORV1:BORV0 = 10 | 2.34 | 2.72 | 3.10 | V | |
| | | BORV1:BORV0 = 01 | 3.63 | 4.22 | 4.81 | V | **(Note 2)** |
| | | BORV1:BORV0 = 00 | 3.90 | 4.54 | 5.18 | V | **(Note 2)** |
| D005G | | PIC18F1220/1320 | Industrial **(-10°C to +85°C)** | | | | |
| | | BORV1:BORV0 = 1x | N/A | N/A | N/A | V | Reserved |
| | | BORV1:BORV0 = 01 | 3.88 | 4.22 | 4.56 | V | **(Note 2)** |
| | | BORV1:BORV0 = 00 | 4.18 | 4.54 | 4.90 | V | **(Note 2)** |
| D005H | | PIC18F1220/1320 | Industrial **(-40°C to -10°C)** | | | | |
| | | BORV1:BORV0 = 1x | N/A | N/A | N/A | V | Reserved |
| | | BORV1:BORV0 = 01 | N/A | N/A | N/A | V | Reserved |
| | | BORV1:BORV0 = 00 | 3.90 | 4.54 | 5.18 | V | **(Note 2)** |
| D005J | | PIC18F1220/1320 | Extended **(-10°C to +85°C)** | | | | |
| | | BORV1:BORV0 = 1x | N/A | N/A | N/A | V | Reserved |
| | | BORV1:BORV0 = 01 | 3.88 | 4.22 | 4.56 | V | **(Note 3)** |
| | | BORV1:BORV0 = 00 | 4.18 | 4.54 | 4.90 | V | **(Note 3)** |
| D005K | | PIC18F1220/1320 | Extended **(-40°C to -10°C, +85°C to +125°C)** | | | | |
| | | BORV1:BORV0 = 1x | N/A | N/A | N/A | V | Reserved |
| | | BORV1:BORV0 = 01 | N/A | N/A | N/A | V | Reserved |
| | | BORV1:BORV0 = 00 | 3.90 | 4.54 | 5.18 | V | **(Note 3)** |

**Legend:** Shading of rows is to assist in readability of the table.

**Note 1:** This is the limit to which VDD can be lowered in Sleep mode, or during a device Reset, without losing RAM data.

**2:** When BOR is on and BORV<1:0> = 0x, the device will operate correctly at 40 MHz for any VDD at which the BOR allows execution (low-voltage and industrial devices only).

**3:** When BOR is on and BORV<1:0> = 0x, the device will operate correctly at 25 MHz for any VDD at which the BOR allows execution (extended devices only).

## 22.2 DC Characteristics: Power-Down and Supply Current PIC18F1220/1320 (Industrial) PIC18LF1220/1320 (Industrial)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **PIC18LF1220/1320** (Industrial) | | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | | |
| **PIC18F1220/1320** (Industrial, Extended) | | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | | |
| **Param No.** | **Device** | **Typ.** | **Max.** | **Units** | **Conditions** | | |
| | **Power-Down Current (IPD)(1)** | | | | | | |
| | PIC18LF1220/1320 | 0.1 | 0.5 | µA | -40°C | | VDD = 2.0V, (**Sleep** mode) |
| | | 0.1 | 0.5 | µA | +25°C | | |
| | | 0.2 | 1.9 | µA | +85°C | | |
| | PIC18LF1220/1320 | 0.1 | 0.5 | µA | -40°C | | VDD = 3.0V, (**Sleep** mode) |
| | | 0.1 | 0.5 | µA | + 25°C | | |
| | | 0.3 | 1.9 | µA | +85°C | | |
| | All devices | 0.1 | 2.0 | µA | -40°C | | VDD = 5.0V, (**Sleep** mode) |
| | | 0.1 | 2.0 | µA | +25°C | | |
| | | 0.4 | 6.5 | µA | +85°C | | |
| | Extended devices | 11.2 | 50 | µA | +125°C | | |
| | **Supply Current (IDD)(2,3)** | | | | | | |
| | PIC18LF1220/1320 | 8 | 40 | µA | -40°C | VDD = 2.0V | FOSC = 31 kHz (**RC_RUN** mode, Internal oscillator source) |
| | | 9 | 40 | µA | +25°C | | |
| | | 11 | 40 | µA | +85°C | | |
| | PIC18LF1220/1320 | 25 | 68 | µA | -40°C | VDD = 3.0V | |
| | | 25 | 68 | µA | +25°C | | |
| | | 20 | 68 | µA | +85°C | | |
| | All devices | 55 | 80 | µA | -40°C | VDD = 5.0V | |
| | | 55 | 80 | µA | +25°C | | |
| | | 50 | 80 | µA | +85°C | | |
| | Extended devices | 50 | 80 | µA | +125°C | | |

**Legend:** Shading of rows is to assist in readability of the table.

**Note 1:** The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD or VSS and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

The test conditions for all IDD measurements in active operation mode are:

OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
$\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.

**3:** For RC oscillator configurations, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kΩ.

**4:** Standard low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.


© 2002-2015 Microchip Technology Inc.

## 22.2 DC Characteristics: Power-Down and Supply Current PIC18F1220/1320 (Industrial) PIC18LF1220/1320 (Industrial) (Continued)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **PIC18LF1220/1320** (Industrial) | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | | |
| **PIC18F1220/1320** (Industrial, Extended) | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended | | | | | |
| **Param No.** | **Device** | **Typ.** | **Max.** | **Units** | **Conditions** | | |
| | **Supply Current (IDD)(2,3)** | | | | | | |
| | PIC18LF1220/1320 | 140 | 220 | µA | -40°C | VDD = 2.0V | FOSC = 1 MHz (**RC_RUN** mode, Internal oscillator source) |
| | | 145 | 220 | µA | +25°C | | |
| | | 155 | 220 | µA | +85°C | | |
| | PIC18LF1220/1320 | 215 | 330 | µA | -40°C | VDD = 3.0V | |
| | | 225 | 330 | µA | +25°C | | |
| | | 235 | 330 | µA | +85°C | | |
| | All devices | 385 | 550 | µA | -40°C | VDD = 5.0V | |
| | | 390 | 550 | µA | +25°C | | |
| | | 405 | 550 | µA | +85°C | | |
| | Extended devices | 410 | 650 | µA | +125°C | | |
| | PIC18LF1220/1320 | 410 | 600 | µA | -40°C | VDD = 2.0V | FOSC = 4 MHz (**RC_RUN** mode, Internal oscillator source) |
| | | 425 | 600 | µA | +25°C | | |
| | | 435 | 600 | µA | +85°C | | |
| | PIC18LF1220/1320 | 650 | 900 | µA | -40°C | VDD = 3.0V | |
| | | 670 | 900 | µA | +25°C | | |
| | | 680 | 900 | µA | +85°C | | |
| | All devices | 1.2 | 1.8 | mA | -40°C | VDD = 5.0V | |
| | | 1.2 | 1.8 | mA | +25°C | | |
| | | 1.2 | 1.8 | mA | +85°C | | |
| | Extended devices | 1.2 | 1.8 | mA | +125°C | | |

**Legend:** Shading of rows is to assist in readability of the table.

**Note 1:** The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD or VSS and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
MCLR = VDD; WDT enabled/disabled as specified.

**3:** For RC oscillator configurations, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kΩ.

**4:** Standard low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

## 22.2 DC Characteristics: Power-Down and Supply Current PIC18F1220/1320 (Industrial) PIC18LF1220/1320 (Industrial) (Continued)

| PIC18LF1220/1320 (Industrial) | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial |
|---|---|
| PIC18F1220/1320 (Industrial, Extended) | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended |

| Param No. | Device | Typ. | Max. | Units | Conditions | | |
|---|---|---|---|---|---|---|---|
| | **Supply Current (IDD)(2,3)** | | | | | | |
| | PIC18LF1220/1320 | 4.7 | 8 | µA | -40°C | VDD = 2.0V | FOSC = 31 kHz (**RC_IDLE** mode, Internal oscillator source) |
| | | 5.0 | 8 | µA | +25°C | | |
| | | 5.8 | 11 | µA | +85°C | | |
| | PIC18LF1220/1320 | 7.0 | 11 | µA | -40°C | VDD = 3.0V | |
| | | 7.8 | 11 | µA | +25°C | | |
| | | 8.7 | 15 | µA | +85°C | | |
| | All devices | 12 | 16 | µA | -40°C | VDD = 5.0V | |
| | | 14 | 16 | µA | +25°C | | |
| | | 14 | 22 | µA | +85°C | | |
| | Extended devices | 25 | 75 | µA | +125°C | | |
| | PIC18LF1220/1320 | 75 | 150 | µA | -40°C | VDD = 2.0V | FOSC = 1 MHz (**RC_IDLE** mode, Internal oscillator source) |
| | | 85 | 150 | µA | +25°C | | |
| | | 95 | 150 | µA | +85°C | | |
| | PIC18LF1220/1320 | 110 | 180 | µA | -40°C | VDD = 3.0V | |
| | | 125 | 180 | µA | +25°C | | |
| | | 135 | 180 | µA | +85°C | | |
| | All devices | 180 | 380 | µA | -40°C | VDD = 5.0V | |
| | | 195 | 380 | µA | +25°C | | |
| | | 200 | 380 | µA | +85°C | | |
| | Extended devices | 350 | 435 | µA | +125°C | | |

**Legend:** Shading of rows is to assist in readability of the table.

**Note 1:** The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD or VSS and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
MCLR = VDD; WDT enabled/disabled as specified.

**3:** For RC oscillator configurations, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kΩ.

**4:** Standard low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.


© 2002-2015 Microchip Technology Inc.

## 22.2 DC Characteristics: Power-Down and Supply Current PIC18F1220/1320 (Industrial) PIC18LF1220/1320 (Industrial) (Continued)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **PIC18LF1220/1320** (Industrial) | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | | |
| **PIC18F1220/1320** (Industrial, Extended) | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended | | | | | |
| **Param No.** | **Device** | **Typ.** | **Max.** | **Units** | **Conditions** | | |
| | **Supply Current (IDD)(2,3)** | | | | | | |
| | PIC18LF1220/1320 | 140 | 275 | µA | -40°C | VDD = 2.0V | FOSC = 4 MHz (**RC_IDLE** mode, Internal oscillator source) |
| | | 140 | 275 | µA | +25°C | | |
| | | 150 | 275 | µA | +85°C | | |
| | PIC18LF1220/1320 | 220 | 375 | µA | -40°C | VDD = 3.0V | |
| | | 220 | 375 | µA | +25°C | | |
| | | 220 | 375 | µA | +85°C | | |
| | All devices | 390 | 800 | µA | -40°C | VDD = 5.0V | |
| | | 400 | 800 | µA | +25°C | | |
| | | 380 | 800 | µA | +85°C | | |
| | Extended devices | 410 | 800 | µA | +125°C | | |
| | PIC18LF1220/1320 | 150 | 250 | µA | -40°C | VDD = 2.0V | FOSC = 1 MHz (**PRI_RUN** mode, EC oscillator) |
| | | 150 | 250 | µA | +25°C | | |
| | | 160 | 250 | µA | +85°C | | |
| | PIC18LF1220/1320 | 340 | 350 | µA | -40°C | VDD = 3.0V | |
| | | 300 | 350 | µA | +25°C | | |
| | | 280 | 350 | µA | +85°C | | |
| | All devices | 0.72 | 1.0 | mA | -40°C | VDD = 5.0V | |
| | | 0.63 | 1.0 | mA | +25°C | | |
| | | 0.58 | 1.0 | mA | +85°C | | |
| | Extended devices | 0.53 | 1.0 | mA | +125°C | | |

**Legend:** Shading of rows is to assist in readability of the table.

**Note 1:** The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD or VSS and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

The test conditions for all IDD measurements in active operation mode are:

OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;

MCLR = VDD; WDT enabled/disabled as specified.

**3:** For RC oscillator configurations, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kΩ.

**4:** Standard low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

## 22.2 DC Characteristics: Power-Down and Supply Current PIC18F1220/1320 (Industrial) PIC18LF1220/1320 (Industrial) (Continued)

| PIC18LF1220/1320 (Industrial) | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial |
|---|---|
| **PIC18F1220/1320 (Industrial, Extended)** | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended |

| Param No. | Device | Typ. | Max. | Units | Conditions | | |
|---|---|---|---|---|---|---|---|
| | **Supply Current (IDD)(2,3)** | | | | | | |
| | PIC18LF1220/1320 | 415 | 600 | µA | -40°C | VDD = 2.0V | FOSC = 4 MHz (**PRI_RUN** mode, EC oscillator) |
| | | 425 | 600 | µA | +25°C | | |
| | | 435 | 600 | µA | +85°C | | |
| | PIC18LF1220/1320 | 0.87 | 1.0 | mA | -40°C | VDD = 3.0V | |
| | | 0.75 | 1.0 | mA | +25°C | | |
| | | 0.75 | 1.0 | mA | +85°C | | |
| | All devices | 1.6 | 2.0 | mA | -40°C | VDD = 5.0V | |
| | | 1.6 | 2.0 | mA | +25°C | | |
| | | 1.5 | 2.0 | mA | +85°C | | |
| | Extended devices | 1.5 | 2.0 | mA | +125°C | | |
| | Extended devices | 6.3 | 9.0 | mA | +125°C | VDD = 4.2V | FOSC = 25 MHz (**PRI_RUN** mode, EC oscillator) |
| | | 9.7 | 10.0 | mA | +125°C | VDD = 5.0V | |
| | All devices | 9.4 | 12 | mA | -40°C | VDD = 4.2V | FOSC = 40 MHz (**PRI_RUN** mode, EC oscillator) |
| | | 9.5 | 12 | mA | +25°C | | |
| | | 9.6 | 12 | mA | +85°C | | |
| | All devices | 11.9 | 15 | mA | -40°C | VDD = 5.0V | |
| | | 12.1 | 15 | mA | +25°C | | |
| | | 12.2 | 15 | mA | +85°C | | |

**Legend:** Shading of rows is to assist in readability of the table.

**Note 1:** The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD or VSS and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
$\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.

**3:** For RC oscillator configurations, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kΩ.

**4:** Standard low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.


© 2002-2015 Microchip Technology Inc.

## 22.2 DC Characteristics: Power-Down and Supply Current PIC18F1220/1320 (Industrial) PIC18LF1220/1320 (Industrial) (Continued)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **PIC18LF1220/1320** (Industrial) | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | | |
| **PIC18F1220/1320** (Industrial, Extended) | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended | | | | | |
| **Param No.** | **Device** | **Typ.** | **Max.** | **Units** | **Conditions** | | |
| | **Supply Current (IDD)(2,3)** | | | | | | |
| | PIC18LF1220/1320 | 35 | 50 | µA | -40°C | VDD = 2.0V | FOSC = 1 MHz (**PRI_IDLE** mode, EC oscillator) |
| | | 35 | 50 | µA | +25°C | | |
| | | 35 | 60 | µA | +85°C | | |
| | PIC18LF1220/1320 | 55 | 80 | µA | -40°C | VDD = 3.0V | |
| | | 50 | 80 | µA | +25°C | | |
| | | 60 | 100 | µA | +85°C | | |
| | All devices | 105 | 150 | µA | -40°C | VDD = 5.0V | |
| | | 110 | 150 | µA | +25°C | | |
| | | 115 | 150 | µA | +85°C | | |
| | Extended devices | 125 | 300 | µA | +125°C | | |
| | PIC18LF1220/1320 | 135 | 180 | µA | -40°C | VDD = 2.0V | FOSC = 4 MHz (**PRI_IDLE** mode, EC oscillator) |
| | | 140 | 180 | µA | +25°C | | |
| | | 140 | 180 | µA | +85°C | | |
| | PIC18LF1220/1320 | 215 | 280 | µA | -40°C | VDD = 3.0V | |
| | | 225 | 280 | µA | +25°C | | |
| | | 230 | 280 | µA | +85°C | | |
| | All devices | 410 | 525 | µA | -40°C | VDD = 5.0V | |
| | | 420 | 525 | µA | +25°C | | |
| | | 430 | 525 | µA | +85°C | | |
| | Extended devices | 450 | 800 | µA | +125°C | | |
| | Extended devices | 2.2 | 3.0 | mA | +125°C | VDD = 4.2V | FOSC = 25 MHz (**PRI_IDLE** mode, EC oscillator) |
| | | 2.7 | 3.5 | mA | +125°C | VDD = 5.0V | |

**Legend:** Shading of rows is to assist in readability of the table.

**Note 1:** The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD or VSS and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
$\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.

**3:** For RC oscillator configurations, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kΩ.

**4:** Standard low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

## 22.2 DC Characteristics: Power-Down and Supply Current PIC18F1220/1320 (Industrial) PIC18LF1220/1320 (Industrial) (Continued)

| PIC18LF1220/1320 (Industrial) | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial |
|---|---|
| **PIC18F1220/1320 (Industrial, Extended)** | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended |

| Param No. | Device | Typ. | Max. | Units | Conditions | | |
|---|---|---|---|---|---|---|---|
| | **Supply Current (IDD)(2,3)** | | | | | | |
| | All devices | 3.2 | 4.1 | mA | -40°C | VDD = 4.2 V | FOSC = 40 MHz (**PRI_IDLE** mode, EC oscillator) |
| | | 3.2 | 4.1 | mA | +25°C | | |
| | | 3.3 | 4.1 | mA | +85°C | | |
| | All devices | 4.0 | 5.1 | mA | -40°C | VDD = 5.0V | |
| | | 4.1 | 5.1 | mA | +25°C | | |
| | | 4.1 | 5.1 | mA | +85°C | | |
| | PIC18LF1220/1320 | 5.1 | 9 | µA | -10°C | VDD = 2.0V | FOSC = 32 kHz(4) (**SEC_RUN** mode, Timer1 as clock) |
| | | 5.8 | 9 | µA | +25°C | | |
| | | 7.9 | 11 | µA | +70°C | | |
| | PIC18LF1220/1320 | 7.9 | 12 | µA | -10°C | VDD = 3.0V | |
| | | 8.9 | 12 | µA | +25°C | | |
| | | 10.5 | 14 | µA | +70°C | | |
| | All devices | 12.5 | 20 | µA | -10°C | VDD = 5.0V | |
| | | 16.3 | 20 | µA | +25°C | | |
| | | 18.4 | 25 | µA | +70°C | | |

**Legend:** Shading of rows is to assist in readability of the table.

**Note 1:** The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD or VSS and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
$\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.

**3:** For RC oscillator configurations, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kΩ.

**4:** Standard low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.


© 2002-2015 Microchip Technology Inc.

## 22.2 DC Characteristics: Power-Down and Supply Current PIC18F1220/1320 (Industrial) PIC18LF1220/1320 (Industrial) (Continued)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **PIC18LF1220/1320** (Industrial) | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | | |
| **PIC18F1220/1320** (Industrial, Extended) | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended | | | | | |
| **Param No.** | **Device** | **Typ.** | **Max.** | **Units** | **Conditions** | | |
| | **Supply Current (IDD)(2,3)** | | | | | | |
| | PIC18LF1220/1320 | 9.2 | 15 | µA | -10°C | VDD = 2.0V | FOSC = 32 kHz(4) (**SEC_IDLE** mode, Timer1 as clock) |
| | | 9.6 | 15 | µA | +25°C | | |
| | | 12.7 | 18 | µA | +70°C | | |
| | PIC18LF1220/1320 | 22 | 30 | µA | -10°C | VDD = 3.0V | |
| | | 21 | 30 | µA | +25°C | | |
| | | 20 | 35 | µA | +70°C | | |
| | All devices | 50 | 80 | µA | -10°C | VDD = 5.0V | |
| | | 45 | 80 | µA | +25°C | | |
| | | 45 | 80 | µA | +70°C | | |

**Legend:** Shading of rows is to assist in readability of the table.

**Note 1:** The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD or VSS and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
$\overline{\text{MCLR}}$ = VDD; WDT enabled/disabled as specified.

**3:** For RC oscillator configurations, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kΩ.

**4:** Standard low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

## 22.2 DC Characteristics: Power-Down and Supply Current PIC18F1220/1320 (Industrial) PIC18LF1220/1320 (Industrial) (Continued)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **PIC18LF1220/1320** (Industrial) | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | | |
| **PIC18F1220/1320** (Industrial, Extended) | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended | | | | | |
| **Param No.** | **Device** | **Typ.** | **Max.** | **Units** | **Conditions** | | |
| | **Module Differential Currents (ΔIWDT, ΔIBOR, ΔILVD, ΔIOSCB, ΔIAD)** | | | | | | |
| D022 (ΔIWDT) | **Watchdog Timer** | 1.5 | 4.0 | µA | -40°C | VDD = 2.0V | |
| | | 2.2 | 4.0 | µA | +25°C | | |
| | | 3.1 | 5.0 | µA | +85°C | | |
| | | 2.5 | 6.0 | µA | -40°C | VDD = 3.0V | |
| | | 3.3 | 6.0 | µA | +25°C | | |
| | | 4.7 | 7.0 | µA | +85°C | | |
| | | 3.7 | 10.0 | µA | -40°C | VDD = 5.0V | |
| | | 4.5 | 10.0 | µA | +25°C | | |
| | | 6.1 | 13.0 | µA | +85°C | | |
| D022A (ΔIBOR) | **Brown-out Reset** | 19 | 35.0 | µA | -40°C to +85°C | VDD = 3.0V | |
| | | 24 | 45.0 | µA | -40°C to +85°C | VDD = 5.0V | |
| D022B (ΔILVD) | **Low-Voltage Detect** | 8.5 | 25.0 | µA | -40°C to +85°C | VDD = 2.0V | |
| | | 16 | 35.0 | µA | -40°C to +85°C | VDD = 3.0V | |
| | | 20 | 45.0 | µA | -40°C to +85°C | VDD = 5.0V | |
| D025 (ΔIOSCB) | **Timer1 Oscillator** | 1.7 | 3.5 | µA | -40°C | VDD = 2.0V | 32 kHz on Timer1(4) |
| | | 1.8 | 3.5 | µA | +25°C | | |
| | | 2.1 | 4.5 | µA | +85°C | | |
| | | 2.2 | 4.5 | µA | -40°C | VDD = 3.0V | 32 kHz on Timer1(4) |
| | | 2.6 | 4.5 | µA | +25°C | | |
| | | 2.8 | 5.5 | µA | +85°C | | |
| | | 3.0 | 6.0 | µA | -40°C | VDD = 5.0V | 32 kHz on Timer1(4) |
| | | 3.3 | 6.0 | µA | +25°C | | |
| | | 3.6 | 7.0 | µA | +85°C | | |
| D026 (ΔIAD) | **A/D Converter** | 1.0 | 3.0 | µA | -40°C to +85°C | VDD = 2.0V | A/D on, not converting |
| | | 1.0 | 4.0 | µA | -40°C to +85°C | VDD = 3.0V | |
| | | 2.0 | 10.0 | µA | -40°C to +85°C | VDD = 5.0V | |
| | | 1.0 | 8.0 | µA | -40°C to +125°C | VDD = 5.0V | |

**Legend:** Shading of rows is to assist in readability of the table.

**Note 1:** The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD or VSS and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
MCLR = VDD; WDT enabled/disabled as specified.

**3:** For RC oscillator configurations, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kΩ.

**4:** Standard low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.


© 2002-2015 Microchip Technology Inc.

## 22.3 DC Characteristics: PIC18F1220/1320 (Industrial) PIC18LF1220/1320 (Industrial)

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | |
|---|---|---|---|---|---|---|
| **Param No.** | **Symbol** | **Characteristic** | **Min.** | **Max.** | **Units** | **Conditions** |
| | VIL | **Input Low Voltage** | | | | |
| | | I/O ports: | | | | |
| D030 | | with TTL buffer | VSS | 0.15 VDD | V | VDD < 4.5V |
| D030A | | | — | 0.8 | V | 4.5V ≤ VDD ≤ 5.5V |
| D031 | | with Schmitt Trigger buffer | VSS | 0.2 VDD | V | |
| D032 | | $\overline{MCLR}$ | VSS | 0.2 VDD | V | |
| D032A | | OSC1 (in XT, HS and LP modes) and T1OSI | VSS | 0.3 VDD | V | |
| D033 | | OSC1 (in RC and EC mode)(1) | VSS | 0.2 VDD | V | |
| | VIH | **Input High Voltage** | | | | |
| | | I/O ports: | | | | |
| D040 | | with TTL buffer | 0.25 VDD + 0.8V | VDD | V | VDD < 4.5V |
| D040A | | | 2.0 | VDD | V | 4.5V ≤ VDD ≤ 5.5V |
| D041 | | with Schmitt Trigger buffer | 0.8 VDD | VDD | V | |
| D042 | | $\overline{MCLR}$, OSC1 (EC mode) | 0.8 VDD | VDD | V | |
| D042A | | OSC1 (in XT, HS and LP modes) and T1OSI | 1.6 VDD | VDD | V | |
| D043 | | OSC1 (RC mode)(1) | 0.9 VDD | VDD | V | |
| | IIL | **Input Leakage Current(2,3)** | | | | |
| D060 | | I/O ports | — | ±1 | µA | VSS ≤ VPIN ≤ VDD, Pin at high-impedance |
| D061 | | $\overline{MCLR}$ | — | ±5 | µA | VSS ≤ VPIN ≤ VDD |
| D063 | | OSC1 | — | ±5 | µA | VSS ≤ VPIN ≤ VDD |
| | IPU | **Weak Pull-up Current** | | | | |
| D070 | IPURB | PORTB weak pull-up current | 50 | 400 | µA | VDD = 5V, VPIN = VSS |

**Note 1:** In RC oscillator configuration, the OSC1/CLKI pin is a Schmitt Trigger input. It is not recommended that the PIC® device be driven with an external clock while in RC mode.

**2:** The leakage current on the $\overline{MCLR}$ 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 current sourced by the pin.

**4:** Parameter is characterized but not tested.

## 22.3 DC Characteristics: PIC18F1220/1320 (Industrial) PIC18LF1220/1320 (Industrial) (Continued)

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | |
|---|---|---|---|---|---|---|
| **Param No.** | **Symbol** | **Characteristic** | **Min.** | **Max.** | **Units** | **Conditions** |
| | VOL | **Output Low Voltage** | | | | |
| D080 | | I/O ports | — | 0.6 | V | IOL = 8.5 mA, VDD = 4.5V, -40°C to +85°C |
| D083 | | OSC2/CLKO (RC mode) | — | 0.6 | V | IOL = 1.6 mA, VDD = 4.5V, -40°C to +85°C |
| | VOH | **Output High Voltage(3)** | | | | |
| D090 | | I/O ports | VDD – 0.7 | — | V | IOH = -3.0 mA, VDD = 4.5V, -40°C to +85°C |
| D092 | | OSC2/CLKO (RC mode) | VDD – 0.7 | — | V | IOH = -1.3 mA, VDD = 4.5V, -40°C to +85°C |
| D150 | VOD | **Open-Drain High Voltage** | — | 8.5 | V | RA4 pin |
| | | **Capacitive Loading Specs on Output Pins** | | | | |
| D100(4) | COSC2 | 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 (in RC mode) | — | 50 | pF | To meet the AC timing specifications |
| D102 | CB | SCL, SDA | — | 400 | pF | In $I^2C$ mode |

**Note 1:** In RC oscillator configuration, the OSC1/CLKI pin is a Schmitt Trigger input. It is not recommended that the PIC® device be driven with an external clock while in RC mode.

**2:** The leakage current on the $\overline{MCLR}$ 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 current sourced by the pin.

**4:** Parameter is characterized but not tested.


© 2002-2015 Microchip Technology Inc.

**TABLE 22-1: MEMORY PROGRAMMING REQUIREMENTS**

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ†** | **Max.** | **Units** | **Conditions** |
| | | **Internal Program Memory Programming Specifications(1)** | | | | | |
| D110 | VPP | Voltage on $\overline{MCLR}$/VPP pin | 9.00 | — | 13.25 | V | **(Note 2)** |
| D112 | IPP | Current into $\overline{MCLR}$/VPP pin | — | — | 5 | µA | |
| D113 | IDDP | Supply Current during Programming | — | — | 10 | mA | |
| | | **Data EEPROM Memory** | | | | | |
| D120 | ED | Byte Endurance | 100K | 1M | — | E/W | -40°C to +85°C |
| D121 | VDRW | VDD for Read/Write | VMIN | — | 5.5 | V | Using EECON to read/write<br>VMIN = Minimum operating voltage |
| D122 | TDEW | Erase/Write Cycle Time | — | 4 | — | ms | |
| D123 | TRETD | Characteristic Retention | 40 | — | — | Year | Provided no other specifications are violated |
| D124 | TREF | Number of Total Erase/Write Cycles before Refresh(3) | 1M | 10M | — | E/W | -40°C to +85°C |
| | | **Program Flash Memory** | | | | | |
| D130 | EP | Cell Endurance | 10K | 100K | — | E/W | -40°C to +85°C |
| D131 | VPR | VDD for Read | VMIN | — | 5.5 | V | VMIN = Minimum operating voltage |
| D132 | VIE | VDD for Block Erase | 4.5 | — | 5.5 | V | Using ICSP port |
| D132A | VIW | VDD for Externally Timed Erase or Write | 4.5 | — | 5.5 | V | Using ICSP port |
| D132B | VPEW | VDD for Self-Timed Write | VMIN | — | 5.5 | V | VMIN = Minimum operating voltage |
| D133 | TIE | ICSP™ Block Erase Cycle Time | — | 4 | — | ms | VDD > 4.5V |
| D133A | TIW | ICSP Erase or Write Cycle Time (externally timed) | 1 | — | — | ms | VDD > 4.5V |
| D133A | TIW | Self-Timed Write Cycle Time | — | 2 | — | ms | |
| D134 | TRETD | Characteristic Retention | 40 | — | — | Year | Provided no other specifications are violated |

† Data in “Typ” column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** These specifications are for programming the on-chip program memory through the use of table write instructions.

**2:** The pin may be kept in this range at times other than programming, but it is not recommended.

**3:** Refer to **Section 7.8 “Using the Data EEPROM”** for a more detailed discussion on data EEPROM endurance.

**FIGURE 22-4: LOW-VOLTAGE DETECT CHARACTERISTICS**

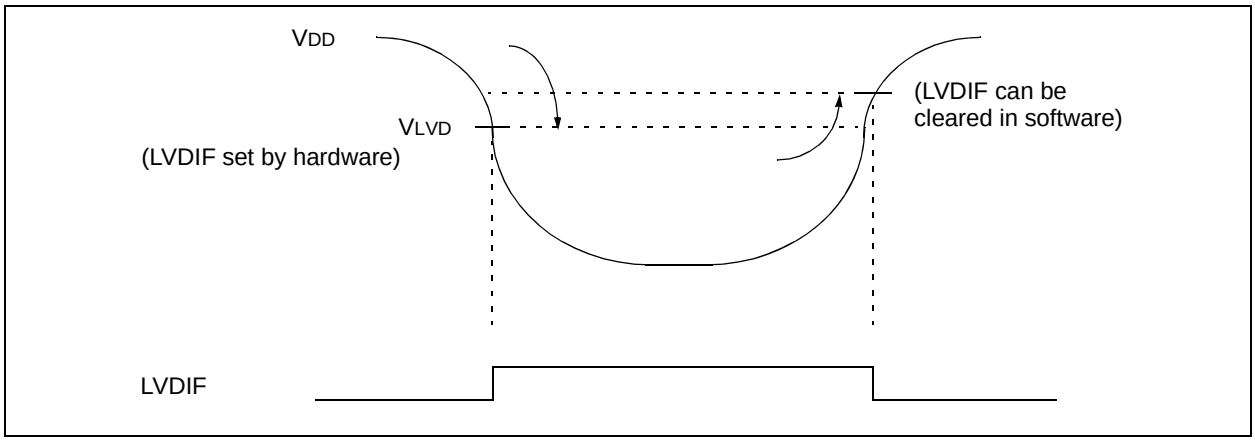



**TABLE 22-2: LOW-VOLTAGE DETECT CHARACTERISTICS**

| **PIC18LF1220/1320** (Industrial) | | | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | |
|---|---|---|---|---|---|---|---|---|
| **PIC18F1220/1320** (Industrial, Extended) | | | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | |
| **Param No.** | **Symbol** | **Characteristic** | | **Min.** | **Typ†** | **Max.** | **Units** | **Conditions** |
| D420D | | LVD Voltage on VDD Transition High-to-Low | | Industrial Low Voltage **(-10°C to +85°C)** | | | | |
| | | PIC18LF1220/1320 | LVDL<3:0> = 0000 | N/A | N/A | N/A | V | Reserved |
| | | | LVDL<3:0> = 0001 | N/A | N/A | N/A | V | Reserved |
| | | | LVDL<3:0> = 0010 | 2.08 | 2.26 | 2.44 | V | |
| | | | LVDL<3:0> = 0011 | 2.26 | 2.45 | 2.65 | V | |
| | | | LVDL<3:0> = 0100 | 2.35 | 2.55 | 2.76 | V | |
| | | | LVDL<3:0> = 0101 | 2.55 | 2.77 | 2.99 | V | |
| | | | LVDL<3:0> = 0110 | 2.64 | 2.87 | 3.10 | V | |
| | | | LVDL<3:0> = 0111 | 2.82 | 3.07 | 3.31 | V | |
| | | | LVDL<3:0> = 1000 | 3.09 | 3.36 | 3.63 | V | |
| | | | LVDL<3:0> = 1001 | 3.29 | 3.57 | 3.86 | V | |
| | | | LVDL<3:0> = 1010 | 3.38 | 3.67 | 3.96 | V | |
| | | | LVDL<3:0> = 1011 | 3.56 | 3.87 | 4.18 | V | |
| | | | LVDL<3:0> = 1100 | 3.75 | 4.07 | 4.40 | V | |
| | | | LVDL<3:0> = 1101 | 3.93 | 4.28 | 4.62 | V | |
| | | | LVDL<3:0> = 1110 | 4.23 | 4.60 | 4.96 | V | |

**Legend:** Shading of rows is to assist in readability of the table.

† Production tested at TAMB = 25°C. Specifications over temperature limits ensured by characterization.


© 2002-2015 Microchip Technology Inc.

**TABLE 22-2: LOW-VOLTAGE DETECT CHARACTERISTICS (CONTINUED)**

| **PIC18LF1220/1320** (Industrial) | | | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | |
|---|---|---|---|---|---|---|---|---|
| **PIC18F1220/1320** (Industrial, Extended) | | | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | |
| **Param No.** | **Symbol** | **Characteristic** | | **Min.** | **Typ†** | **Max.** | **Units** | **Conditions** |
| D420F | | LVD Voltage on VDD Transition High-to-Low | | Industrial Low Voltage **(-40°C to -10°C)** | | | | |
| | | PIC18LF1220/1320 | LVDL<3:0> = 0000 | N/A | N/A | N/A | V | Reserved |
| | | | LVDL<3:0> = 0001 | N/A | N/A | N/A | V | Reserved |
| | | | LVDL<3:0> = 0010 | 1.99 | 2.26 | 2.53 | V | |
| | | | LVDL<3:0> = 0011 | 2.16 | 2.45 | 2.75 | V | |
| | | | LVDL<3:0> = 0100 | 2.25 | 2.55 | 2.86 | V | |
| | | | LVDL<3:0> = 0101 | 2.43 | 2.77 | 3.10 | V | |
| | | | LVDL<3:0> = 0110 | 2.53 | 2.87 | 3.21 | V | |
| | | | LVDL<3:0> = 0111 | 2.70 | 3.07 | 3.43 | V | |
| | | | LVDL<3:0> = 1000 | 2.96 | 3.36 | 3.77 | V | |
| | | | LVDL<3:0> = 1001 | 3.14 | 3.57 | 4.00 | V | |
| | | | LVDL<3:0> = 1010 | 3.23 | 3.67 | 4.11 | V | |
| | | | LVDL<3:0> = 1011 | 3.41 | 3.87 | 4.34 | V | |
| | | | LVDL<3:0> = 1100 | 3.58 | 4.07 | 4.56 | V | |
| | | | LVDL<3:0> = 1101 | 3.76 | 4.28 | 4.79 | V | |
| | | | LVDL<3:0> = 1110 | 4.04 | 4.60 | 5.15 | V | |
| | | LVD Voltage on VDD Transition High-to-Low | | Industrial **(-10°C to +85°C)** | | | | |
| D420G | | PIC18F1220/1320 | LVDL<3:0> = 1101 | 3.93 | 4.28 | 4.62 | V | |
| | | | LVDL<3:0> = 1110 | 4.23 | 4.60 | 4.96 | V | |
| | | LVD Voltage on VDD Transition High-to-Low | | Industrial **(-40°C to -10°C)** | | | | |
| D420H | | PIC18F1220/1320 | LVDL<3:0> = 1101 | 3.76 | 4.28 | 4.79 | V | |
| | | | LVDL<3:0> = 1110 | 4.04 | 4.60 | 5.15 | V | |
| | | LVD Voltage on VDD Transition High-to-Low | | Extended **(-10°C to +85°C)** | | | | |
| D420J | | PIC18F1220/1320 | LVDL<3:0> = 1101 | 3.94 | 4.28 | 4.62 | V | |
| | | | LVDL<3:0> = 1110 | 4.23 | 4.60 | 4.96 | V | |
| | | LVD Voltage on VDD Transition High-to-Low | | Extended **(-40°C to -10°C, +85°C to +125°C)** | | | | |
| D420K | | PIC18F1220/1320 | LVDL<3:0> = 1101 | 3.77 | 4.28 | 4.79 | V | |
| | | | LVDL<3:0> = 1110 | 4.05 | 4.60 | 5.15 | V | |

**Legend:** Shading of rows is to assist in readability of the table.

† Production tested at TAMB = 25°C. Specifications over temperature limits ensured by characterization.

## 22.4 AC (Timing) Characteristics

### 22.4.1 TIMING PARAMETER SYMBOLOGY

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

1. TppS2ppS
2. TppS
3. TCC:ST ($I^2C$ specifications only)
4. Ts ($I^2C$ specifications only)

| | | | |
|---|---|---|---|
| T | | | |
| F | Frequency | T | Time |

Lowercase letters (pp) and their meanings:

| | | | |
|---|---|---|---|
| pp | | | |
| cc | CCP1 | osc | OSC1 |
| ck | CLKO | rd | $\overline{RD}$ |
| cs | $\overline{CS}$ | rw | $\overline{RD}$ or $\overline{WR}$ |
| di | SDI | sc | SCK |
| do | SDO | ss | $\overline{SS}$ |
| dt | Data in | t0 | T0CKI |
| io | I/O port | t1 | T13CKI |
| mc | $\overline{MCLR}$ | wr | $\overline{WR}$ |

Uppercase letters and their meanings:

| | | | |
|---|---|---|---|
| S | | | |
| F | Fall | P | Period |
| H | High | R | Rise |
| I | Invalid (High-Impedance) | V | Valid |
| L | Low | Z | High-Impedance |
| $I^2C$ only | | | |
| AA | output access | High | High |
| BUF | Bus free | Low | Low |

TCC:ST ($I^2C$ specifications only)

| | | | |
|---|---|---|---|
| CC | | | |
| HD | Hold | SU | Setup |
| ST | | | |
| DAT | DATA input hold | STO | Stop condition |
| STA | Start condition | | |


© 2002-2015 Microchip Technology Inc.

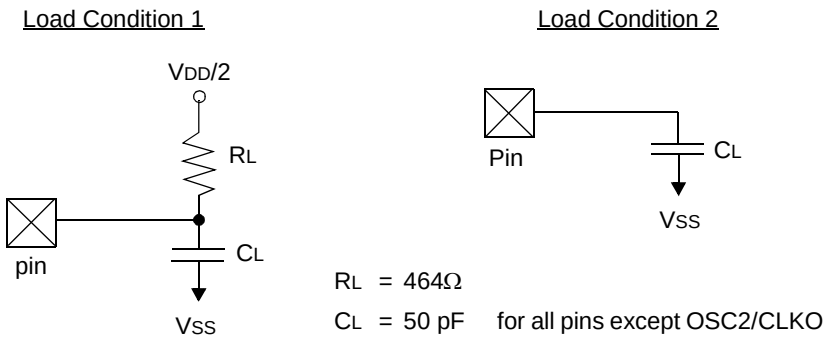
### 22.4.2 TIMING CONDITIONS

The temperature and voltages specified in Table 22-3 apply to all timing specifications unless otherwise noted. Figure 22-5 specifies the load conditions for the timing specifications.

**TABLE 22-3: TEMPERATURE AND VOLTAGE SPECIFICATIONS – AC**

| | |
|---|---|
| **AC CHARACTERISTICS** | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended<br>Operating voltage VDD range as described in DC spec **Section 22.1** and **Section 22.3**.<br>LF parts operate for industrial temperatures only. |

**FIGURE 22-5: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS**



22.4.3 TIMING DIAGRAMS AND SPECIFICATIONS

**FIGURE 22-6: EXTERNAL CLOCK TIMING (ALL MODES EXCEPT PLL)**

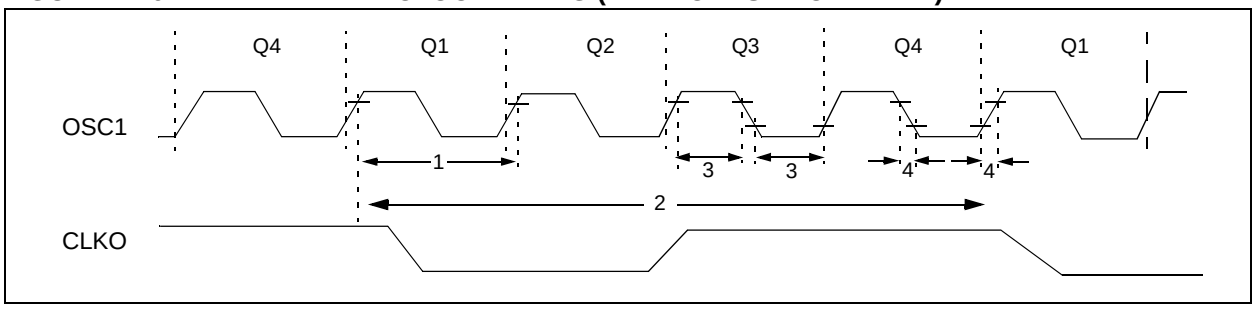



**TABLE 22-4: EXTERNAL CLOCK TIMING REQUIREMENTS**

| Param. No. | Symbol | Characteristic | Min. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| 1A | FOSC | External CLKI Frequency(1) | DC | 40 | MHz | EC, ECIO (LF and Industrial) |
| | | | DC | 25 | MHz | EC, ECIO (Extended) |
| | | Oscillator Frequency(1) | DC | 4 | MHz | RC oscillator |
| | | | DC | 1 | MHz | XT oscillator |
| | | | DC | 25 | MHz | HS oscillator |
| | | | 1 | 10 | MHz | HS + PLL oscillator |
| | | | DC | 33 | kHz | LP Oscillator mode |
| 1 | TOSC | External CLKI Period(1) | 25 | — | ns | EC, ECIO (LF and Industrial) |
| | | | 40 | — | ns | EC, ECIO (Extended) |
| | | Oscillator Period(1) | 250 | — | ns | RC oscillator |
| | | | 1000 | — | ns | XT oscillator |
| | | | 25 | — | ns | HS oscillator |
| | | | 100 | 1000 | ns | HS + PLL oscillator |
| | | | 30 | — | µs | LP oscillator |
| 2 | TCY | Instruction Cycle Time(1) | 100 | — | ns | TCY = 4/FOSC |
| 3 | TosL, TosH | External Clock in (OSC1) High or Low Time | 30 | — | ns | XT oscillator |
| | | | 2.5 | — | µs | LP oscillator |
| | | | 10 | — | ns | HS oscillator |
| 4 | TosR, TosF | External Clock in (OSC1) Rise or Fall Time | — | 20 | ns | XT oscillator |
| | | | — | 50 | ns | LP oscillator |
| | | | — | 7.5 | ns | HS oscillator |

**Note 1:** Instruction cycle period (TCY) equals four times the input oscillator time base period for all configurations except PLL. All specified values are based on characterization data for that particular oscillator type under standard operating conditions, with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at “min.” values with an external clock applied to the OSC1/CLKI pin. When an external clock input is used, the “max.” cycle time limit is “DC” (no clock) for all devices.


© 2002-2015 Microchip Technology Inc.

**TABLE 22-5: PLL CLOCK TIMING SPECIFICATIONS, HS/HSPLL MODE ($V_{DD}$ = 4.2V TO 5.5V)**

| Param No. | Sym. | Characteristic | Min. | Typ† | Max. | Units | Conditions |
|---|---|---|---|---|---|---|---|
| F10 | $F_{OSC}$ | Oscillator Frequency Range | 4 | — | 10 | MHz | HS and HSPLL mode only |
| F11 | $F_{SYS}$ | On-Chip VCO System Frequency | 16 | — | 40 | MHz | HSPLL mode only |
| F12 | $T_{PLL}$ | PLL Start-up Time (Lock Time) | — | — | 2 | ms | HSPLL mode only |
| F13 | $\Delta_{CLK}$ | CLKO Stability (Jitter) | -2 | — | +2 | % | HSPLL mode only |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**TABLE 22-6: INTERNAL RC ACCURACY: PIC18F1220/1320 (INDUSTRIAL)**
**PIC18LF1220/1320 (INDUSTRIAL)**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **PIC18LF1220/1320** (Industrial) | | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ $T_A$ ≤ +85°C for industrial | | | | | | |
| **PIC18F1220/1320** (Industrial) | | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ $T_A$ ≤ +85°C for industrial<br>-40°C ≤ $T_A$ ≤ +125°C for extended | | | | | | |
| **Param No.** | **Device** | **Min.** | **Typ.** | **Max.** | **Units** | **Conditions** | | |
| | **INTOSC Accuracy @ Freq = 8 MHz, 4 MHz, 2 MHz, 1 MHz, 500 kHz, 250 kHz, 125 kHz(1)** | | | | | | | |
| | PIC18LF1220/1320 | -2 | +/-1 | 2 | % | +25°C | $V_{DD}$ = 2.7-3.3V | |
| | | -5 | — | 5 | % | -10°C to +85°C | $V_{DD}$ = 2.7-3.3V | |
| | | -10 | — | 10 | % | -40°C to +85°C | $V_{DD}$ = 2.7-3.3V | |
| | PIC18F1220/ 1320PIC18F1220/1320 | -2 | +/-1 | 2 | % | +25°C | $V_{DD}$ = 4.5-5.5V | |
| | | -5 | — | 5 | % | -10°C to +85°C | $V_{DD}$ = 4.5-5.5V | |
| | | -10 | — | 10 | % | -40°C to +85°C | $V_{DD}$ = 4.5-5.5V | |
| | **INTRC Accuracy @ Freq = 31 kHz(2)** | | | | | | | |
| | PIC18LF1220/1320 | 26.562 | — | 35.938 | kHz | -40°C to +85°C | $V_{DD}$ = 2.7-3.3V | |
| | PIC18F1220/ 1320PIC18F1220/1320 | 26.562 | — | 35.938 | kHz | -40°C to +85°C | $V_{DD}$ = 4.5-5.5V | |

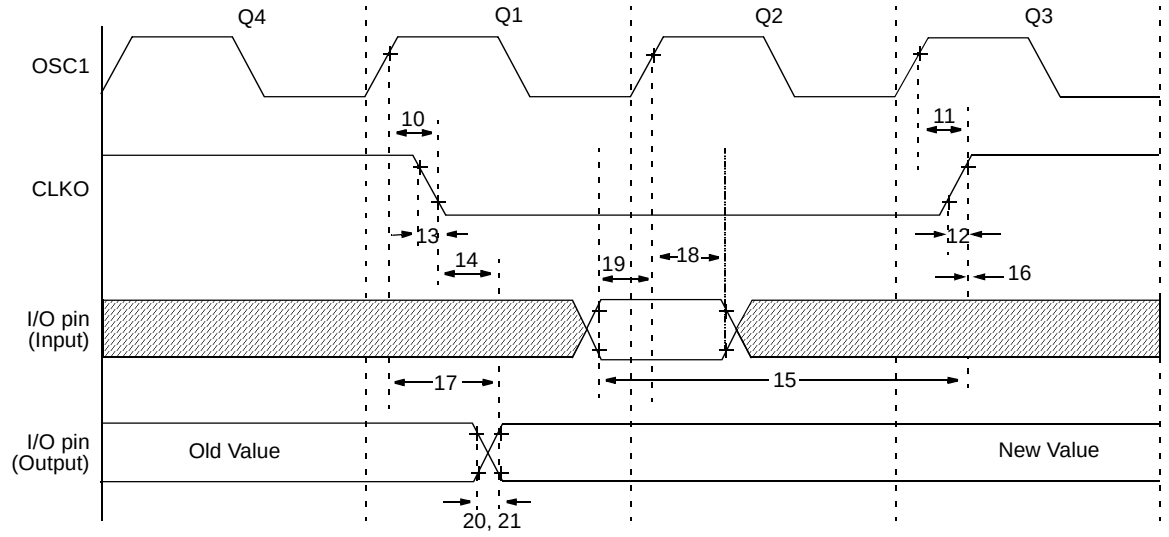
**Legend:** Shading of rows is to assist in readability of the table.

**Note 1:** Frequency calibrated at 25°C. OSCTUNE register can be used to compensate for temperature and $V_{DD}$ drift.

**2:** INTRC frequency after calibration.

**3:** Change of INTRC frequency as $V_{DD}$ changes.

**FIGURE 22-7: CLKO AND I/O TIMING**




**Note:** Refer to Figure 22-5 for load conditions.

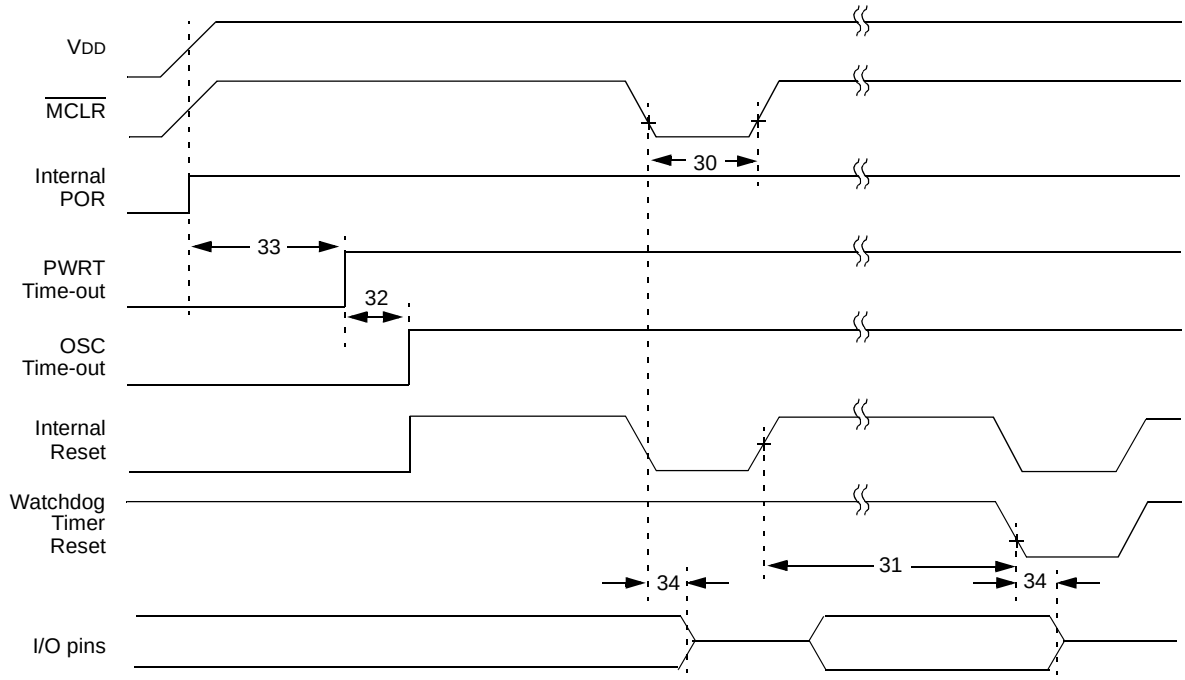
**TABLE 22-7: CLKO AND I/O TIMING REQUIREMENTS**

| Param. No. | Symbol | Characteristic | | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 10 | TosH2ckL | OSC1↑ to CLKO↓ | | — | 75 | 200 | ns | **(Note 1)** |
| 11 | TosH2ckH | OSC1↑ to CLKO↑ | | — | 75 | 200 | ns | **(Note 1)** |
| 12 | TckR | CLKO Rise Time | | — | 35 | 100 | ns | **(Note 1)** |
| 13 | TckF | CLKO Fall Time | | — | 35 | 100 | ns | **(Note 1)** |
| 14 | TckL2ioV | CLKO↓ to Port Out Valid | | — | — | 0.5 TCY + 20 | ns | **(Note 1)** |
| 15 | TioV2ckH | Port In Valid before CLKO↑ | | 0.25 TCY + 25 | — | — | ns | **(Note 1)** |
| 16 | TckH2ioI | Port In Hold after CLKO↑ | | 0 | — | — | ns | **(Note 1)** |
| 17 | TosH2ioV | OSC1↑ (Q1 cycle) to Port Out Valid | | — | 50 | 150 | ns | |
| 18 | TosH2ioI | OSC1↑ (Q2 cycle) to Port Input Invalid (I/O in hold time) | PIC18F1X20 | 100 | — | — | ns | |
| 18A | | | PIC18LF1X20 | 200 | — | — | ns | |
| 19 | TioV2osH | Port Input Valid to OSC1↑ (I/O in setup time) | | 0 | — | — | ns | |
| 20 | TioR | Port Output Rise Time | PIC18F1X20 | — | 10 | 25 | ns | |
| 20A | | | PIC18LF1X20 | — | — | 60 | ns | |
| 21 | TioF | Port Output Fall Time | PIC18F1X20 | — | 10 | 25 | ns | |
| 21A | | | PIC18LF1X20 | — | — | 60 | ns | |

**Note 1:** Measurements are taken in RC mode, where CLKO output is 4 x TOSC.


© 2002-2015 Microchip Technology Inc.

**FIGURE 22-8: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING**




**Note:** Refer to Figure 22-5 for load conditions.

**FIGURE 22-9: BROWN-OUT RESET TIMING**

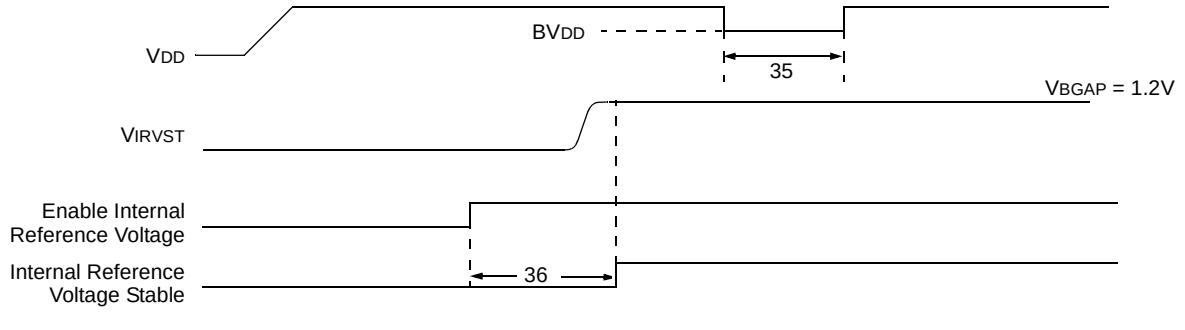
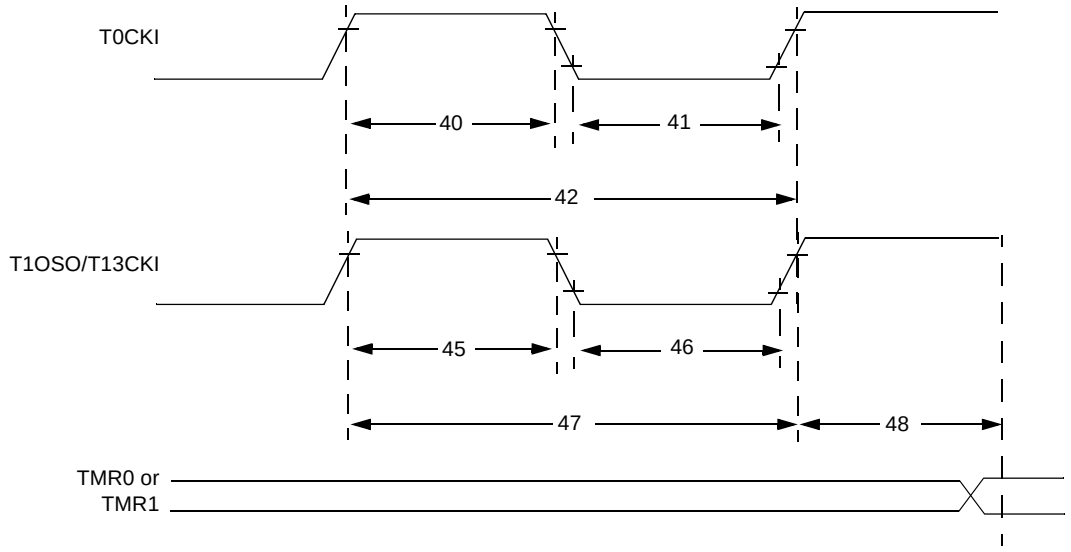


**TABLE 22-8: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER AND BROWN-OUT RESET REQUIREMENTS**

| Param. No. | Symbol | Characteristic | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 30 | TmcL | $\overline{MCLR}$ Pulse Width (low) | 2 | — | — | µs | |
| 31 | TWDT | Watchdog Timer Time-out Period (No postscaler) | 3.48 | 4.00 | 4.71 | ms | |
| 32 | TOST | Oscillation Start-up Timer Period | 1024 TOSC | — | 1024 TOSC | — | TOSC = OSC1 period |
| 33 | TPWRT | Power-up Timer Period | — | 65.5 | 132 | ms | |
| 34 | TIOZ | I/O High-Impedance from $\overline{MCLR}$ Low or Watchdog Timer Reset | — | 2 | — | µs | |
| 35 | TBOR | Brown-out Reset Pulse Width | 200 | — | — | µs | VDD ≤ BVDD (see D005) |
| 36 | TIVRST | Time for Internal Reference Voltage to become stable | — | 20 | 50 | µs | |
| 37 | TLVD | Low-Voltage Detect Pulse Width | 200 | — | — | µs | VDD ≤ VLVD |

**FIGURE 22-10: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS**



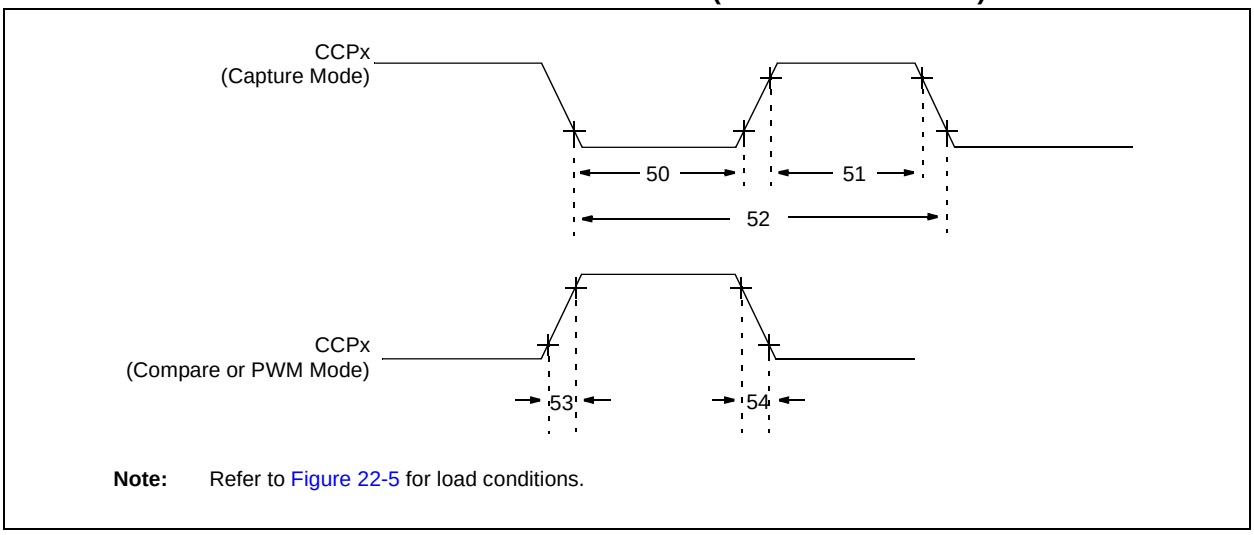

**Note:** Refer to Figure 22-5 for load conditions.


© 2002-2015 Microchip Technology Inc.

**TABLE 22-9: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS**

| Param No. | Symbol | Characteristic | | | Min. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 40 | Tt0H | T0CKI High Pulse Width | | No prescaler | 0.5 TCY + 20 | — | ns | |
| | | | | With prescaler | 10 | — | ns | |
| 41 | Tt0L | T0CKI Low Pulse Width | | No prescaler | 0.5 TCY + 20 | — | ns | |
| | | | | With prescaler | 10 | — | ns | |
| 42 | Tt0P | T0CKI Period | | No prescaler | TCY + 10 | — | ns | |
| | | | | With prescaler | Greater of: 20 ns or $\frac{T_{CY} + 40}{N}$ | — | ns | N = prescale value (1, 2, 4,..., 256) |
| 45 | Tt1H | T13CKI High Time | Synchronous, no prescaler | | 0.5 TCY + 20 | — | ns | |
| | | | Synchronous, with prescaler | PIC18F1X20 | 10 | — | ns | |
| | | | | PIC18LF1X20 | 25 | — | ns | |
| | | | Asynchronous | PIC18F1X20 | 30 | — | ns | |
| | | | | PIC18LF1X20 | 50 | — | ns | |
| 46 | Tt1L | T13CKI Low Time | Synchronous, no prescaler | | 0.5 TCY + 5 | — | ns | |
| | | | Synchronous, with prescaler | PIC18F1X20 | 10 | — | ns | |
| | | | | PIC18LF1X20 | 25 | — | ns | |
| | | | Asynchronous | PIC18F1X20 | 30 | — | ns | |
| | | | | PIC18LF1X20 | 50 | — | ns | |
| 47 | Tt1P | T13CKI Input Period | Synchronous | | Greater of: 20 ns or $\frac{T_{CY} + 40}{N}$ | — | ns | N = prescale value (1, 2, 4, 8) |
| | | | Asynchronous | | 60 | — | ns | |
| | Ft1 | T13CKI Oscillator Input Frequency Range | | | DC | 50 | kHz | |
| 48 | Tcke2tmrl | Delay from External T13CKI Clock Edge to Timer Increment | | | 2 TOSC | 7 TOSC | — | |

**FIGURE 22-11: CAPTURE/COMPARE/PWM TIMINGS (ALL CCP MODULES)**

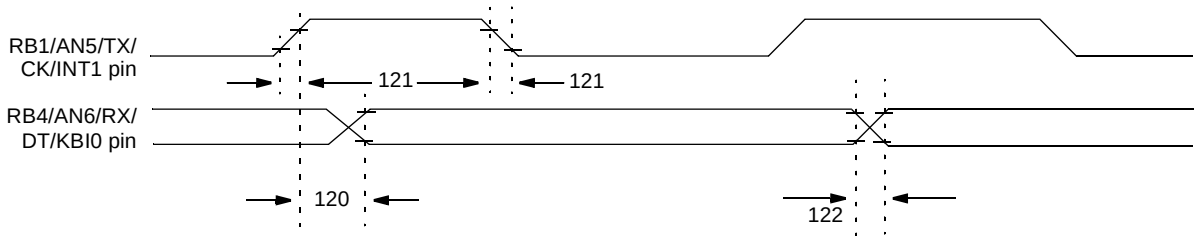



**Note:** Refer to Figure 22-5 for load conditions.

**TABLE 22-10: CAPTURE/COMPARE/PWM REQUIREMENTS (ALL CCP MODULES)**

| Param. No. | Symbol | Characteristic | | | Min. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 50 | TccL | CCPx Input Low Time | No prescaler | | 0.5 TCY + 20 | — | ns | |
| | | | With prescaler | PIC18F1X20 | 10 | — | ns | |
| | | | | PIC18LF1X20 | 20 | — | ns | |
| 51 | TccH | CCPx Input High Time | No prescaler | | 0.5 TCY + 20 | — | ns | |
| | | | With prescaler | PIC18F1X20 | 10 | — | ns | |
| | | | | PIC18LF1X20 | 20 | — | ns | |
| 52 | TccP | CCPx Input Period | | | $\frac{3 T_{CY} + 40}{N}$ | — | ns | N = prescale value (1, 4 or 16) |
| 53 | TccR | CCPx Output Fall Time | | PIC18F1X20 | — | 25 | ns | |
| | | | | PIC18LF1X20 | — | 45 | ns | |
| 54 | TccF | CCPx Output Fall Time | | PIC18F1X20 | — | 25 | ns | |
| | | | | PIC18LF1X20 | — | 45 | ns | |

**FIGURE 22-12: EUSART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING**

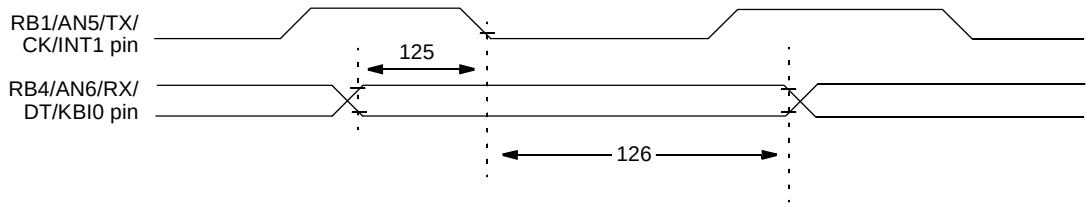



**Note:** Refer to Figure 22-5 for load conditions.

**TABLE 22-11: EUSART SYNCHRONOUS TRANSMISSION REQUIREMENTS**

| Param. No. | Symbol | Characteristic | | Min. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 120 | TckH2dtV | SYNC XMIT (MASTER & SLAVE) Clock High to Data Out Valid | PIC18F1X20 | — | 40 | ns | |
| | | | PIC18LF1X20 | — | 100 | ns | |
| 121 | Tckrf | Clock Out Rise Time and Fall Time (Master mode) | PIC18F1X20 | — | 20 | ns | |
| | | | PIC18LF1X20 | — | 50 | ns | |
| 122 | Tdtrf | Data Out Rise Time and Fall Time | PIC18F1X20 | — | 20 | ns | |
| | | | PIC18LF1X20 | — | 50 | ns | |

**FIGURE 22-13: EUSART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING**




**Note:** Refer to Figure 22-5 for load conditions.


© 2002-2015 Microchip Technology Inc.

**TABLE 22-12: EUSART SYNCHRONOUS RECEIVE REQUIREMENTS**

| Param. No. | Symbol | Characteristic | Min. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| 125 | TdtV2ckl | SYNC RCV (MASTER & SLAVE)<br>Data Hold before CK↓ (DT hold time) | 10 | — | ns | |
| 126 | TckL2dtl | Data Hold after CK↓ (DT hold time) | 15 | — | ns | |

**TABLE 22-13: A/D CONVERTER CHARACTERISTICS: PIC18F1220/1320 (INDUSTRIAL)**
**PIC18LF1220/1320 (INDUSTRIAL)**

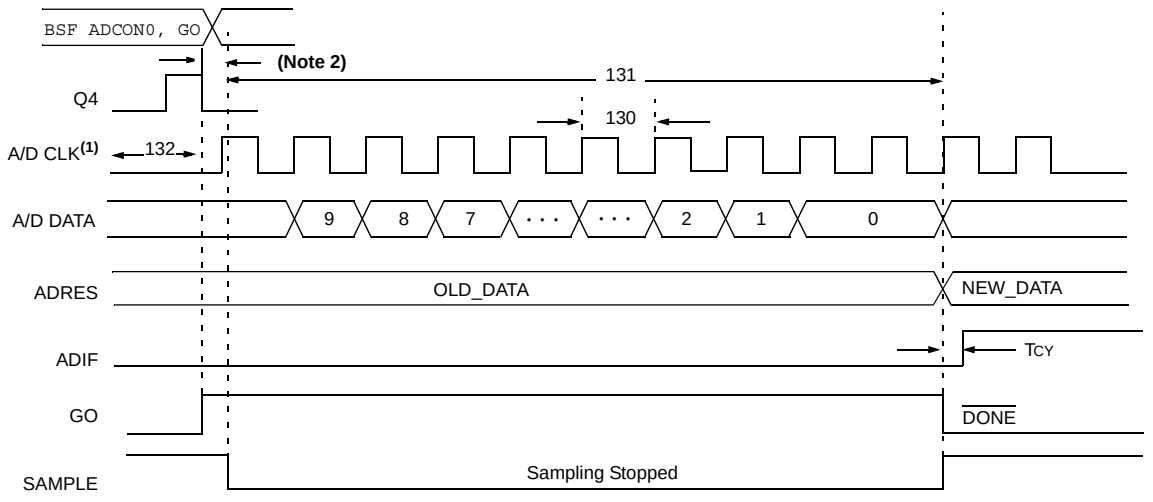
| Param No. | Symbol | Characteristic | | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| A01 | NR | Resolution | | — | — | 10 | bit | ΔVREF ≥ 3.0V |
| A03 | EIL | Integral Linearity Error | | — | — | <±1 | LSb | ΔVREF ≥ 3.0V |
| A04 | EDL | Differential Linearity Error | | — | — | <±1 | LSb | ΔVREF ≥ 3.0V |
| A06 | EOFF | Offset Error | | — | — | <±1 | LSb | ΔVREF ≥ 3.0V |
| A07 | EGN | Gain Error | | — | — | <±1 | LSb | ΔVREF ≥ 3.0V |
| A10 | — | Monotonicity | | guaranteed(2) | | | — | |
| A20 | ΔVREF | Reference Voltage Range (VREFH – VREFL) | | 3 | — | AVDD – AVSS | V | For 10-bit resolution |
| A21 | VREFH | Reference Voltage High | | AVSS + 3.0V | — | AVDD + 0.3V | V | For 10-bit resolution |
| A22 | VREFL | Reference Voltage Low | | AVSS – 0.3V | — | AVDD – 3.0V | V | For 10-bit resolution |
| A25 | VAIN | Analog Input Voltage | | VREFL | — | VREFH | V | |
| A28 | AVDD | Analog Supply Voltage | | VDD – 0.3 | — | VDD + 0.3 | V | |
| A29 | AVSS | Analog Supply Voltage | | VSS – 0.3 | — | VSS + 0.3 | V | |
| A30 | ZAIN | Recommended Impedance of Analog Voltage Source | | — | — | 2.5 | kΩ | |
| A40 | IAD | A/D Conversion Current (VDD) | PIC18F1X20 | — | 180 | — | μA | Average current consumption when A/D is on **(Note 1)** |
| | | | PIC18LF1X20 | — | 90 | — | μA | |
| A50 | IREF | VREF Input Current **(Note 3)** | | —<br>— | —<br>— | ±5<br>±150 | μA<br>μA | During VAIN acquisition.<br>During A/D conversion cycle. |

**Note 1:** When A/D is off, it will not consume any current other than minor leakage current. The power-down current specification includes any such leakage from the A/D module.

**2:** The A/D conversion result never decreases with an increase in the input voltage and has no missing codes.

**3:** VREFH current is from RA3/AN3/VREF+ pin or AVDD, whichever is selected as the VREFH source. VREFL current is from RA2/AN2/VREF- pin or AVSS, whichever is selected as the VREFL source.

**FIGURE 22-14: A/D CONVERSION TIMING**




**Note 1:** If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the SLEEP instruction to be executed.

**2:** This is a minimal RC delay (typically 100 ns), which also disconnects the holding capacitor from the analog input.

**TABLE 22-14: A/D CONVERSION REQUIREMENTS**

| Param No. | Symbol | Characteristic | | Min. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 130 | TAD | A/D Clock Period | PIC18F1X20 | 1.6 | 20(5) | µs | TOSC based, VREF ≥ 3.0V |
| | | | PIC18LF1X20 | 3.0 | 20(5) | µs | TOSC based, VREF full range |
| | | | PIC18F1X20 | 2.0 | 6.0 | µs | A/D RC mode |
| | | | PIC18LF1X20 | 3.0 | 9.0 | µs | A/D RC mode |
| 131 | TCNV | Conversion Time (not including acquisition time) **(Note 1)** | | 11 | 12 | TAD | |
| 132 | TACQ | Acquisition Time **(Note 3)** | | 15<br>10 | —<br>— | µs<br>µs | -40°C ≤ Temp ≤ +125°C<br>0°C ≤ Temp ≤ +125°C |
| 135 | TSWC | Switching Time from Convert → Sample | | — | **(Note 4)** | | |
| 136 | TAMP | Amplifier Settling Time **(Note 2)** | | 1 | — | µs | This may be used if the "new" input voltage has not changed by more than 1 LSb (i.e., 5 mV @ 5.12V) from the last sampled voltage (as stated on CHOLD). |

**Note 1:** ADRES register may be read on the following TCY cycle.

**2:** See **Section 17.0 "10-Bit Analog-to-Digital Converter (A/D) Module"** for minimum conditions when input voltage has changed more than 1 LSb.

**3:** The time for the holding capacitor to acquire the "New" input voltage, when the voltage changes full scale after the conversion (AVDD to AVSS, or AVSS to AVDD). The source impedance (*RS*) on the input channels is 50Ω.

**4:** On the next Q4 cycle of the device clock.

**5:** The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.


© 2002-2015 Microchip Technology Inc.

## 22.5 High Temperature Operation

This section outlines the specifications for the following devices operating in the high temperature range between -40°C and 150°C.(6)

- PIC18F1220
- PIC18F1320

When the value of any parameter is identical for both the 125°C Extended and the 150°C High Temp. temperature ranges, then that value will be found in the standard specification tables shown earlier in this chapter, under the fields listed for the 125°C Extended temperature range. If the value of any parameter is unique to the 150°C High Temp. temperature range, then it will be listed here, in this section of the data sheet.

If a Silicon Errata exists for the product and it lists a modification to the 125°C Extended temperature range value, one that is also shared at the 150°C High Temp. temperature range, then that modified value will apply to both temperature ranges.

**Note 1:** Data contained in this section is applicable to the following devices: PIC18F1220 and PIC18F1320.

**2:** Writes are **not allowed** for Flash program memory above 125°C.

**3:** All AC timing specifications are increased by 30%.

**4:** Figure 22-3; The frequency range is decreased to 20 MHz.

**5:** The temperature range indicator in the catalog part number and device marking is “H” for -40°C to 150°C.

Example: PIC18F1220T-H/SO indicates the device is shipped in a Tape and Reel configuration, in the SOIC package, and is rated for operation from -40°C to 150°C.

**6:** AEC-Q100 reliability testing for devices intended to operate at 150°C is 1,000 hours. Any design in which the total operating time from 125°C to 150°C will be greater than 1,000 hours is not warranted without prior written approval from Microchip Technology Inc.

**TABLE 22-15: ABSOLUTE MAXIMUM RATINGS**

| Parameter | Source/Sink | Value | Units |
|---|---|---|---|
| Max. Current: $V_{SS}$ | Sink | 300 | mA |
| Max. Current: $V_{DD}$ | Source | 250 | mA |
| Max. Current: Pin | Sink | 20 | mA |
| Max. Current: Pin | Source | 15 | mA |
| Max. Port Current: All ports combined | Sink | 20 | mA |
| Max. Port Current: All ports combined | Source | 15 | mA |
| Max. Junction Temperature | | 155 | °C |

**Note:** 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 above maximum rating conditions for extended periods may affect device reliability.

**TABLE 22-16: DC CHARACTERISTICS FOR IDD SPECIFICATIONS FOR PIC18F1220/1320-H (High Temp.)**

| Symbol | Device Characteristics | Min. | Typ. | Max. | Units | Condition | |
|---|---|---|---|---|---|---|---|
| | | | | | | VDD | Note |
| IPD | Power Down Current | — | — | 220 | μA | 5.0 | Sleep mode |
| IDD | Supply Current | — | — | 250 | μA | 5.0 | FOSC = 31 kHz (RC_Run mode, Internal Oscillator source) |
| IDD | Supply Current | — | — | 800 | μA | 5.0 | FOSC = 1 MHz (RC_Run mode, Internal Oscillator source) |
| IDD | Supply Current | — | — | 2.0 | mA | 5.0 | FOSC = 4 MHz (RC_Run mode, Internal Oscillator source) |
| IDD | Supply Current | — | — | 240 | μA | 5.0 | FOSC = 31 kHz (RC_Idle mode, Internal Oscillator source) |
| IDD | Supply Current | — | — | 600 | μA | 5.0 | FOSC = 1 MHz (RC_Idle mode, Internal Oscillator source) |
| IDD | Supply Current | — | — | 1.0 | mA | 5.0 | FOSC = 4 MHz (RC_Idle mode, Internal Oscillator source) |
| IDD | Supply Current | — | — | 1.2 | mA | 5.0 | FOSC = 1 MHz (PRI_Run mode, EC Oscillator) |
| IDD | Supply Current | — | — | 2.2 | mA | 5.0 | FOSC = 4 MHz (PRI_Run mode, EC Oscillator) |
| IDD | Supply Current | — | — | 10.0 | mA | 5.0 | FOSC = 20 MHz (PRI_Run mode, EC Oscillator) |
| IDD | Supply Current | — | — | 500 | μA | 5.0 | FOSC = 1 MHz (PRI_Idle mode, EC Oscillator) |
| IDD | Supply Current | — | — | 1.0 | mA | 5.0 | FOSC = 4 MHz (PRI_Idle mode, EC Oscillator) |
| IDD | Supply Current | — | — | 3.5 | mA | 5.0 | FOSC = 20 MHz (PRI_Idle mode, EC Oscillator) |

**TABLE 22-17: ADC CHARACTERISTICS FOR PIC18F1220/1320-H (High Temp.)**

| Param No. | Sym. | Characteristic | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|---|
| D026 | A/D Conv | | — | — | 30 | μA | VDD = 5.0V, A/D on, Not converting |


© 2002-2015 Microchip Technology Inc.

**TABLE 22-18: DC CHARACTERISTICS FOR $I_{PD}$ SPECIFICATIONS FOR PIC18F1220/1320-H (High Temp.)**

| Param No. | Sym. | Device Characteristics | Min. | Typ. | Max. | Units | |
|---|---|---|---|---|---|---|---|
| D030 | $V_{IL}$ | I/O Ports with TTL Buffer Supply Current | $V_{SS}$ | — | 0.20 $V_{DD}$ | μA | $V_{DD}$ < 4.5V |
| D030A | $V_{IL}$ | | $V_{SS}$ | — | 0.7 | V | 4.5V ≤ $V_{DD}$ ≤ 5.5V |
| D031 | $V_{IL}$ | I/O Ports with Schmitt Trigger Buffer | $V_{SS}$ | — | 0.25 $V_{DD}$ | V | |
| D032 | $V_{IL}$ | $\overline{MCLR}$ | $V_{SS}$ | — | 0.25 $V_{DD}$ | V | |
| D032A | $V_{IL}$ | OSC1 (XT, HS and LP modes) and T1OSI | $V_{SS}$ | — | 0.35 $V_{DD}$ | V | |
| D033 | $V_{IL}$ | OSC1 (RC and EC modes) | $V_{SS}$ | — | 0.25 $V_{DD}$ | V | |
| D040 | $V_{IH}$ | I/O with TTL Buffer | 0.3 $V_{DD}$ + 0.8V | — | $V_{DD}$ | V | $V_{DD}$ < 4.5V |
| D040A | $V_{IH}$ | | 2.1 | — | $V_{DD}$ | V | 4.5V ≤ $V_{DD}$ ≤ 5.5V |
| D041 | $V_{IH}$ | I/O Ports with Schmitt Trigger Buffer | 0.85 $V_{DD}$ | — | $V_{DD}$ | V | |
| D042 | $V_{IH}$ | $\overline{MCLR}$, OSC1 (EC mode) | 0.85 $V_{DD}$ | — | $V_{DD}$ | V | |
| D042A | $V_{IH}$ | OSC1 (XT, HS and LP modes) and T1OSI | 1.65 | — | $V_{DD}$ | V | |
| D070 | $I_{PURB}$ | Port B Weak Pull-up Current | 25 | — | 450 | μA | $V_{DD}$ = 5V, $V_{PIN}$ = $V_{SS}$ |
| | $I_{AB}$ | Combined Source Current for Ports A and B | — | — | 15 | mA | $V_{DD}$ = 4.5V, $I_{OH}$ = 3.0 mA, $V_{PIN}$ = $V_{OH}$ |
| | $I_{AB}$ | Combined Sink Current for Ports A and B | — | — | 25 | mA | $V_{DD}$ = 4.5V, $I_{OL}$ = 8.5 mA, $V_{PIN}$ = $V_{OL}$ |

**TABLE 22-19: EXTERNAL CLOCK TIMING REQUIREMENTS FOR PIC18F1220/1320-H (High Temp.)**

| Param No. | Sym. | Characteristic | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 1A | $F_{OSC}$ | External CLKI Frequency | DC | — | 20 | MHz | EC, ECIO (Extended) |
| 1A | $F_{OSC}$ | Oscillator Frequency | DC | — | 20 | MHz | HS Oscillator |
| 1 | $T_{OSC}$ | External CLKI Period | 50 | — | — | ns | EC, ECIO (Extended) |
| 1 | $T_{OSC}$ | Oscillator Period | 50 | — | — | ns | HS Oscillator |

**TABLE 22-20: INTERNAL RC ACCURACY FOR PIC18F1220/1320-H (High Temp.)**

| Param No. | Device | Min. | Typ. | Max. | Units | Temp. | Conditions |
|---|---|---|---|---|---|---|---|
| | PIC18F1220/1320 | 2.0 | — | 2.0 | % | 25°C | 4.5V ≤ $V_{DD}$ ≤ 5.5V |
| | PIC18F1220/1320 | 20 | — | 20 | % | -40°C to 150°C | 4.5V ≤ $V_{DD}$ ≤ 5.5V |

## 23.0 DC AND AC CHARACTERISTICS GRAPHS AND TABLES

**Note:** The graphs and tables provided following this note are a statistical summary based on a limited number of samples and are provided for informational purposes only. The performance characteristics listed herein are not tested or guaranteed. In some graphs or tables, the data presented may be outside the specified operating range (e.g., outside specified power supply range) and therefore, outside the warranted range.

"Typical" represents the mean of the distribution at 25°C. "Maximum" or "minimum" represents (mean + 3σ) or (mean – 3σ) respectively, where σ is a standard deviation, over the whole temperature range.

**FIGURE 23-1: TYPICAL IDD vs. FOSC OVER VDD PRI_RUN, EC MODE, +25°C**

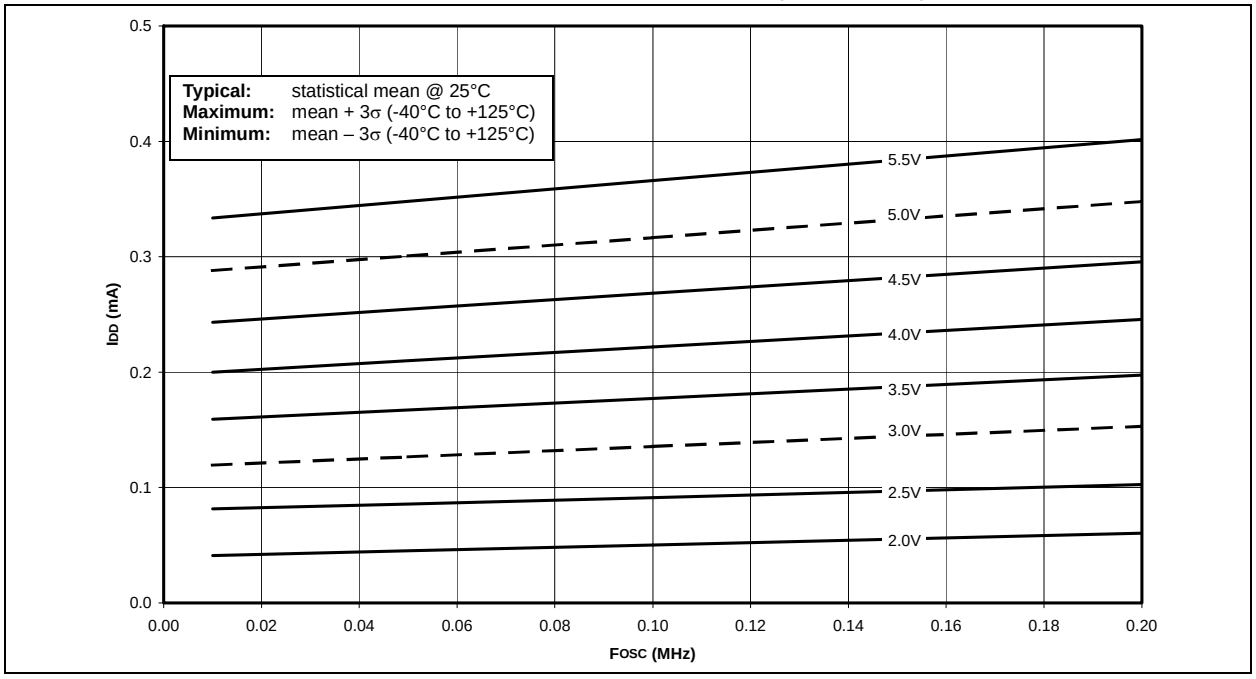
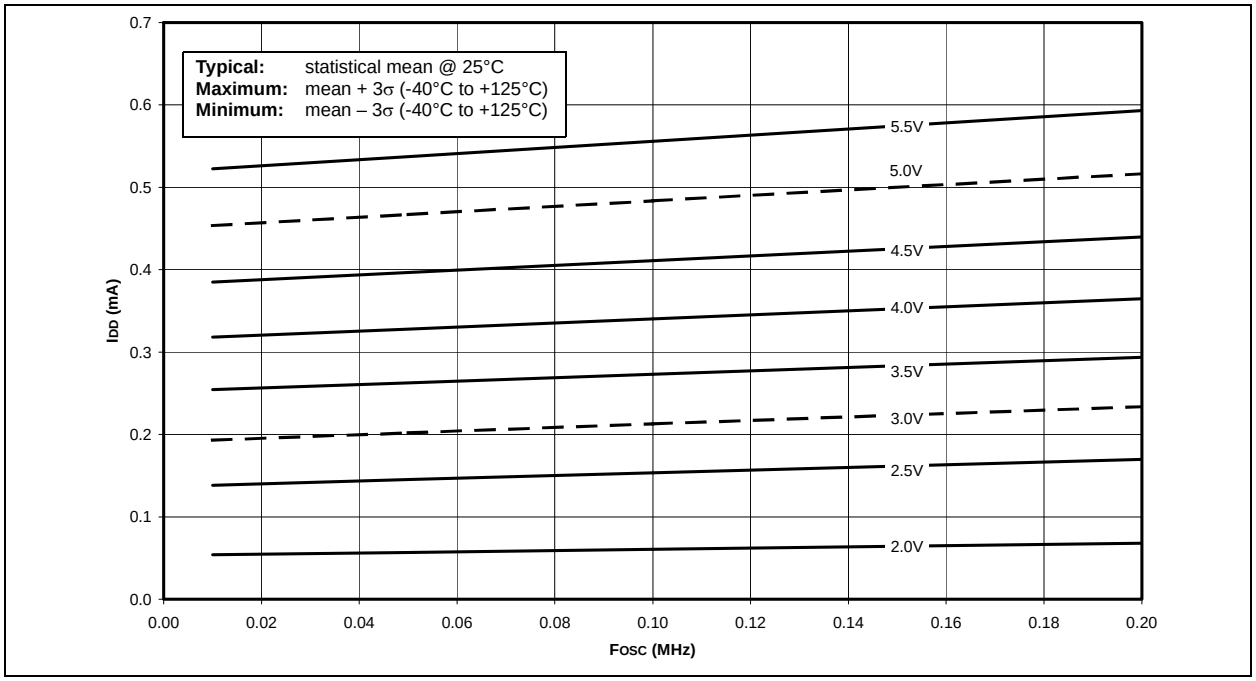



**FIGURE 23-2: MAXIMUM IDD vs. FOSC OVER VDD PRI_RUN, EC MODE, -40°C TO +85°C**





© 2002-2015 Microchip Technology Inc.

**FIGURE 23-3: MAXIMUM IDD vs. FOSC OVER VDD PRI_RUN, EC MODE, -40°C TO +125°C**

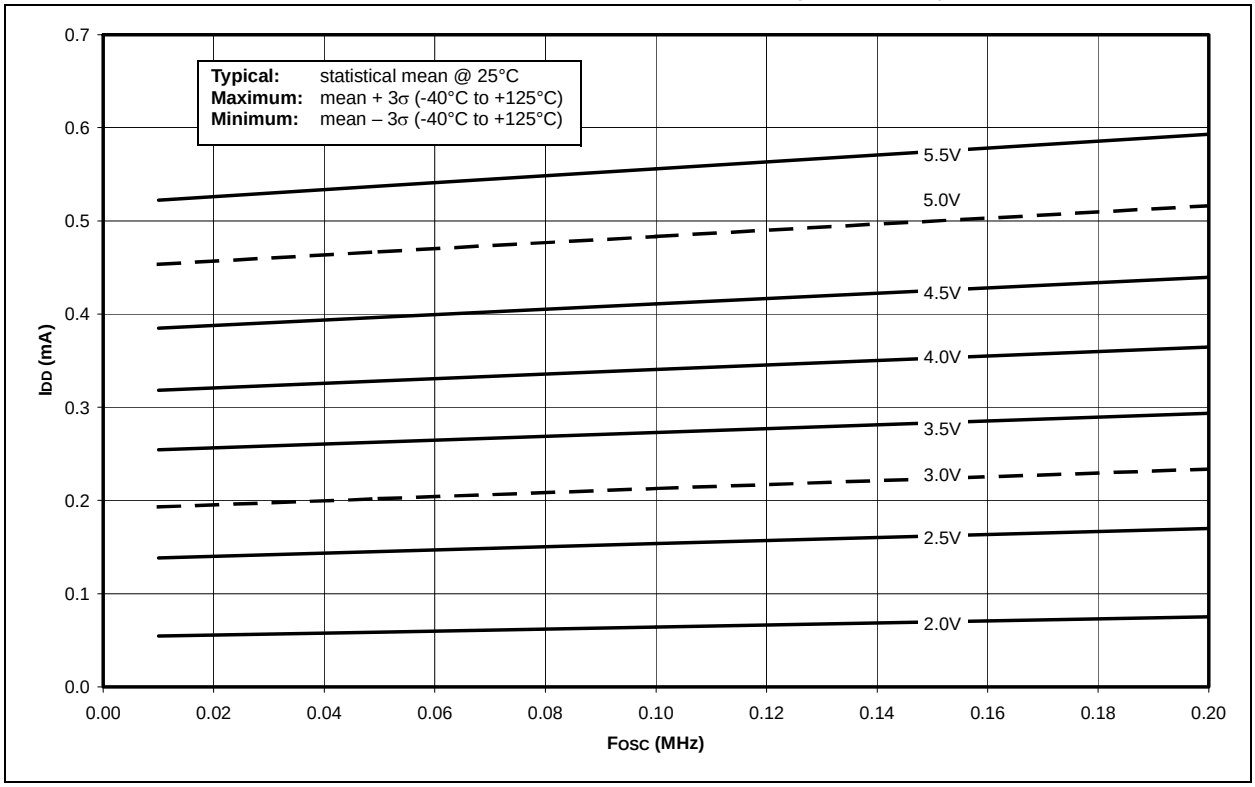



**FIGURE 23-4: TYPICAL IDD vs. FOSC OVER VDD PRI_RUN, EC MODE, +25°C**

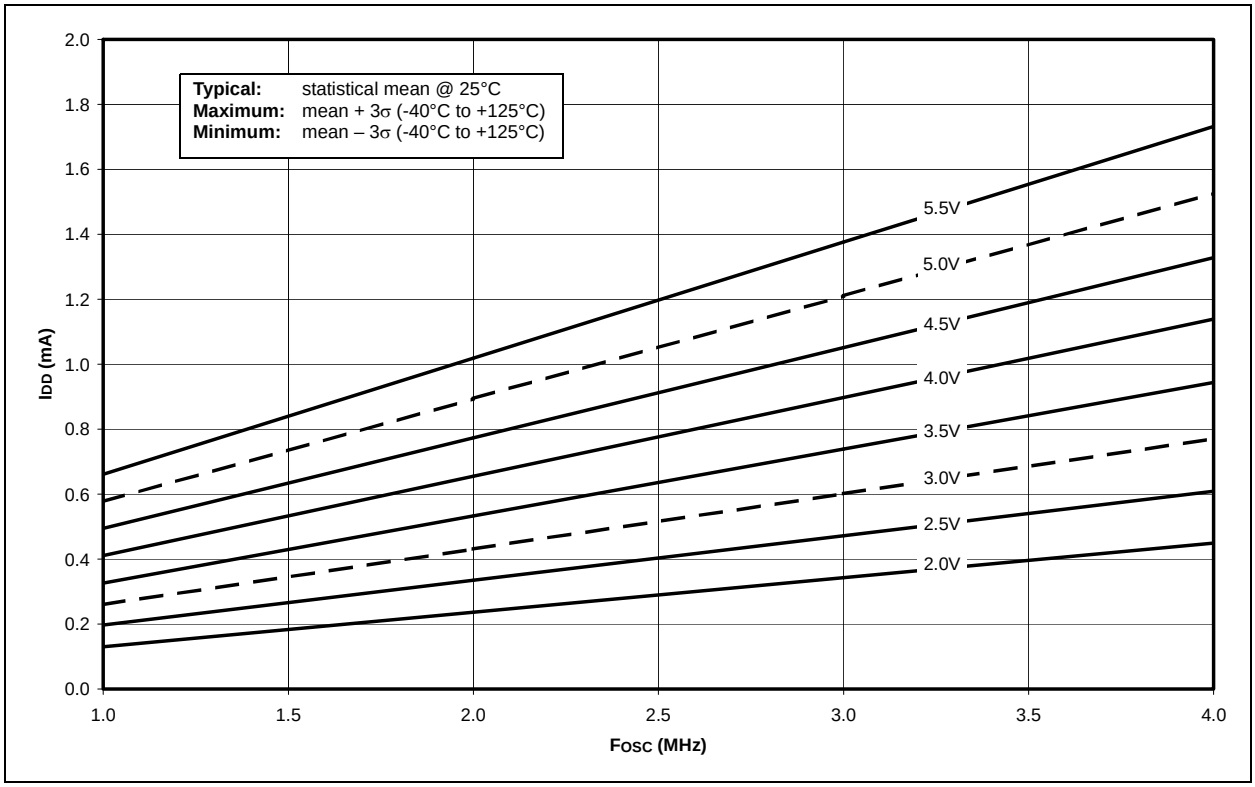


**FIGURE 23-5: MAXIMUM IDD vs. FOSC OVER VDD PRI_RUN, EC MODE, -40°C TO +125°C**

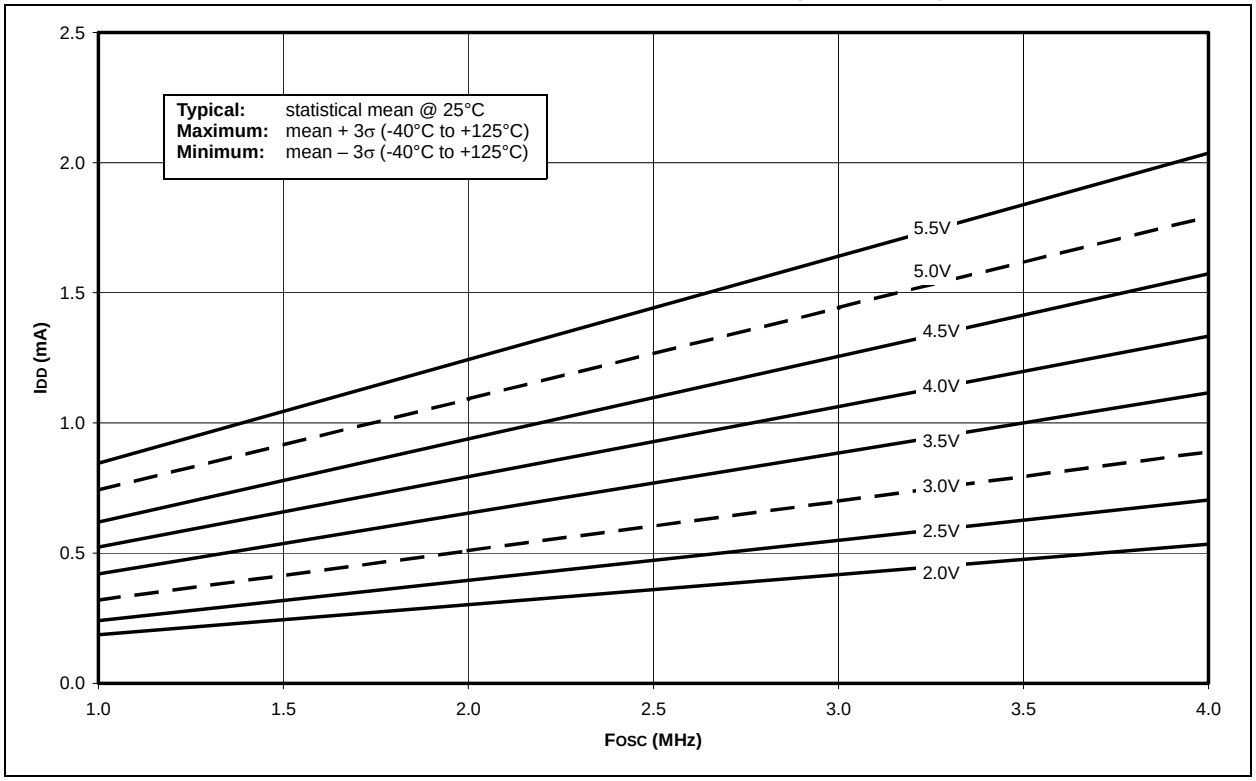
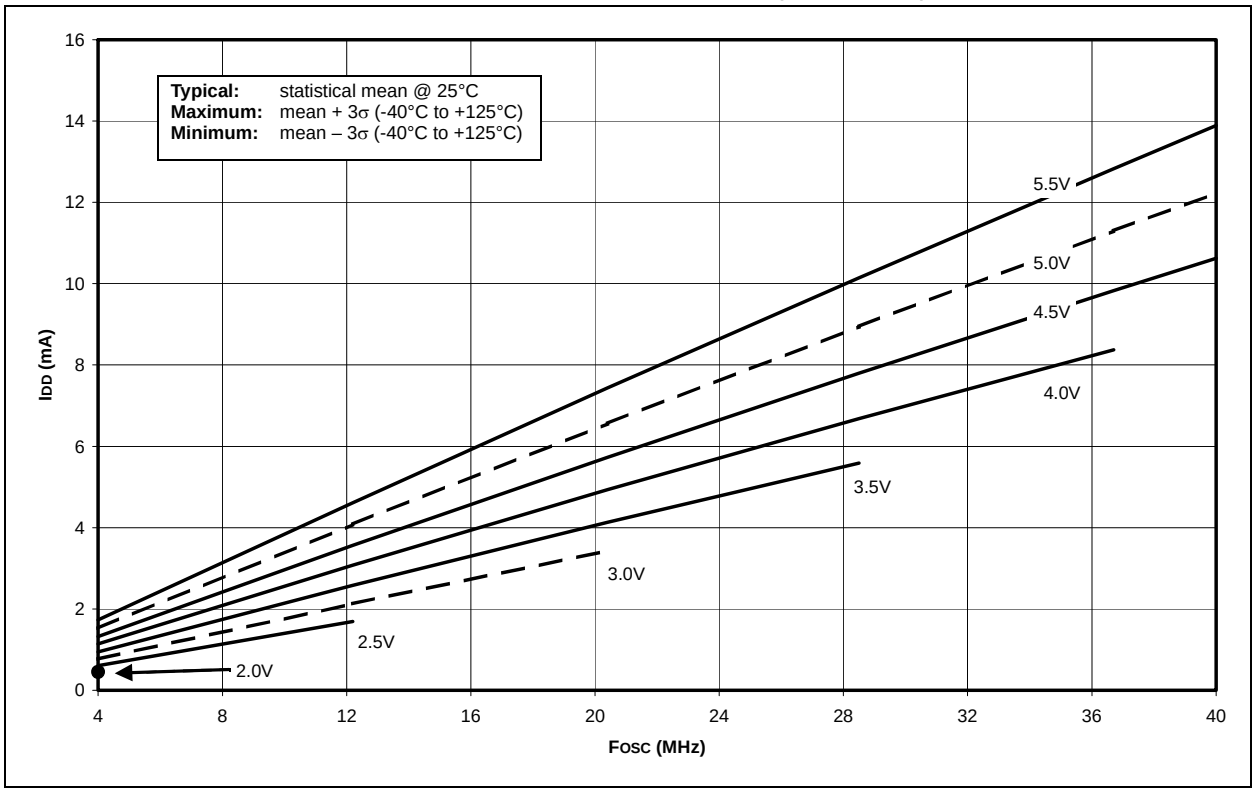



**FIGURE 23-6: TYPICAL IDD vs. FOSC OVER VDD PRI_RUN, EC MODE, +25°C**





© 2002-2015 Microchip Technology Inc.

**FIGURE 23-7: MAXIMUM IDD vs. FOSC OVER VDD PRI_RUN, EC MODE, -40°C TO +125°C**

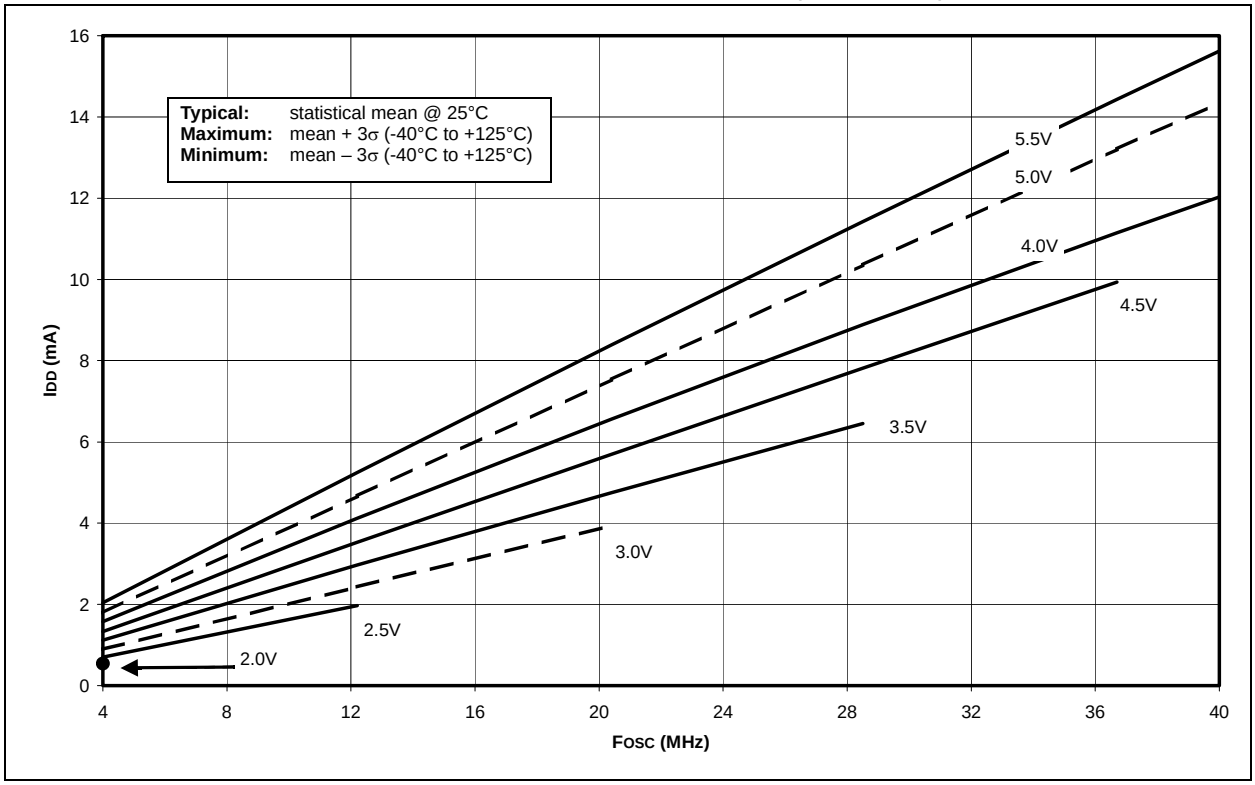



**FIGURE 23-8: TYPICAL IDD vs. FOSC OVER VDD PRI_IDLE, EC MODE, +25°C**

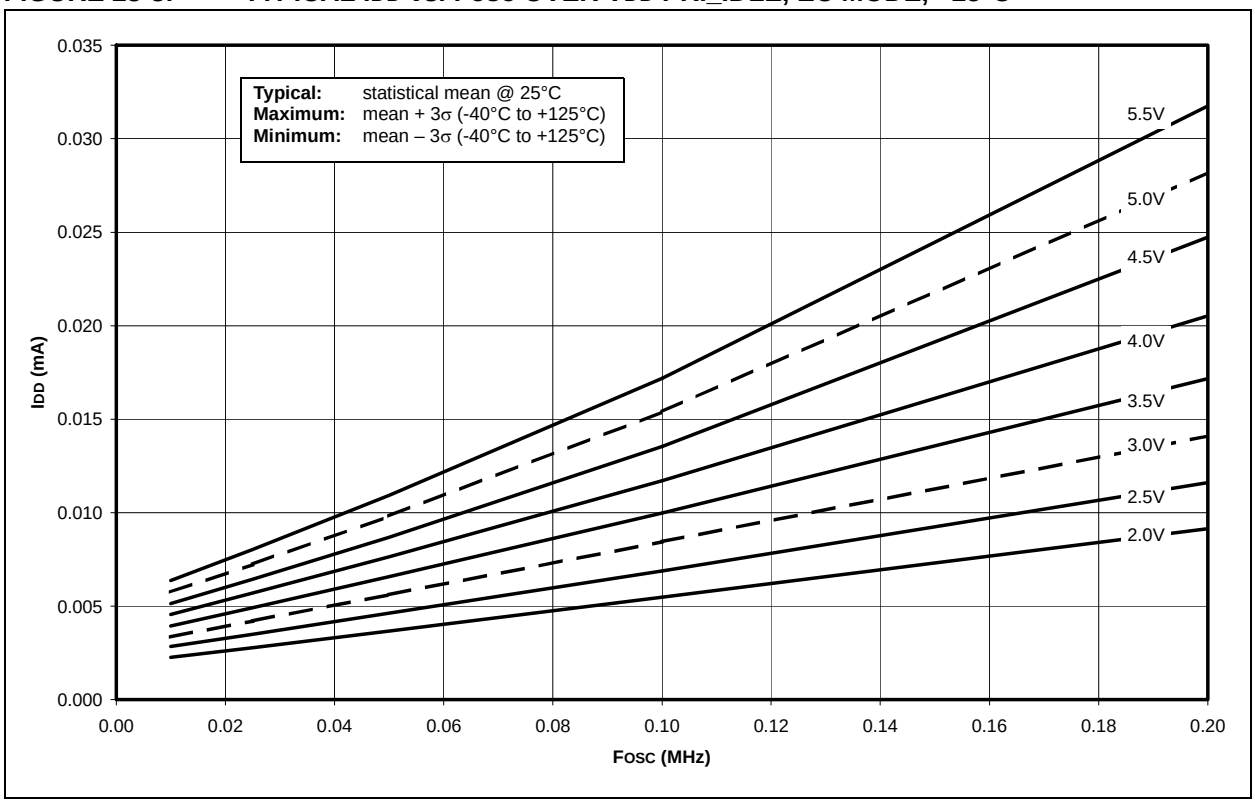


**FIGURE 23-9: MAXIMUM IDD vs. FOSC OVER VDD PRI_IDLE, EC MODE, -40°C TO +85°C**

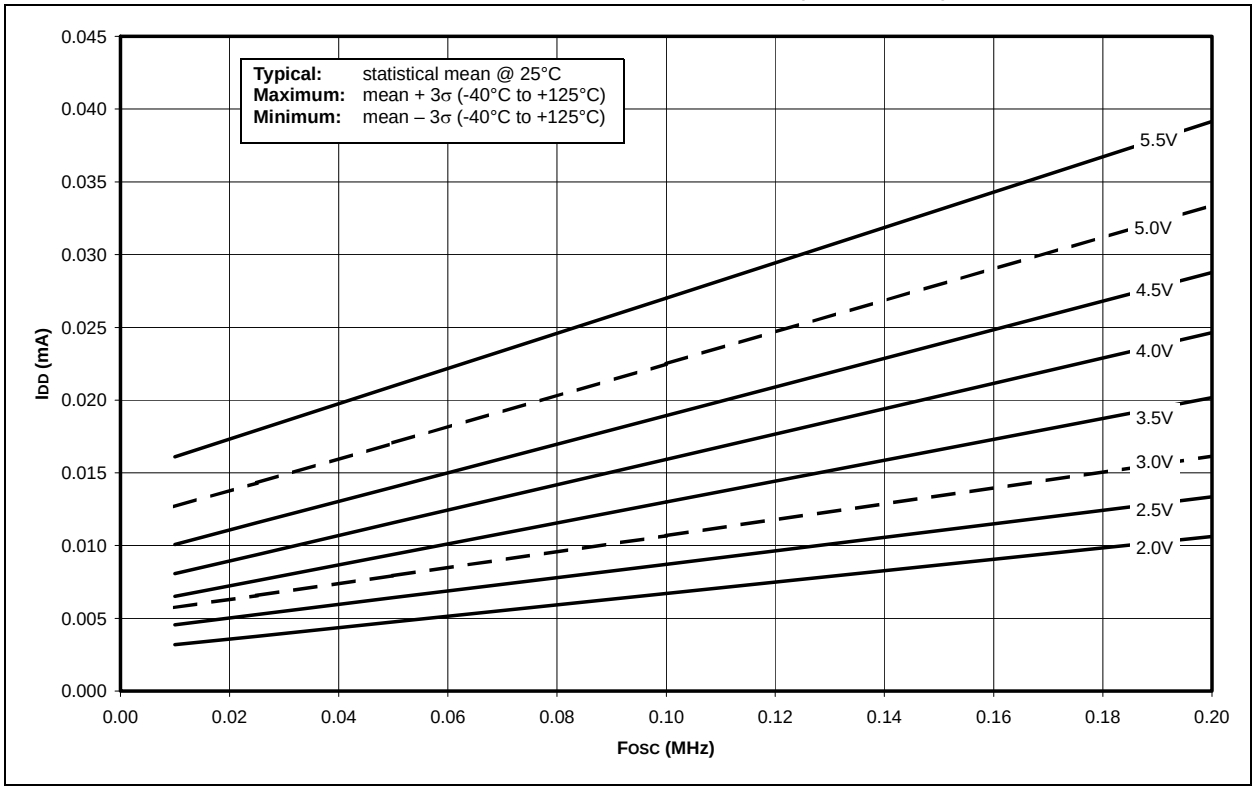
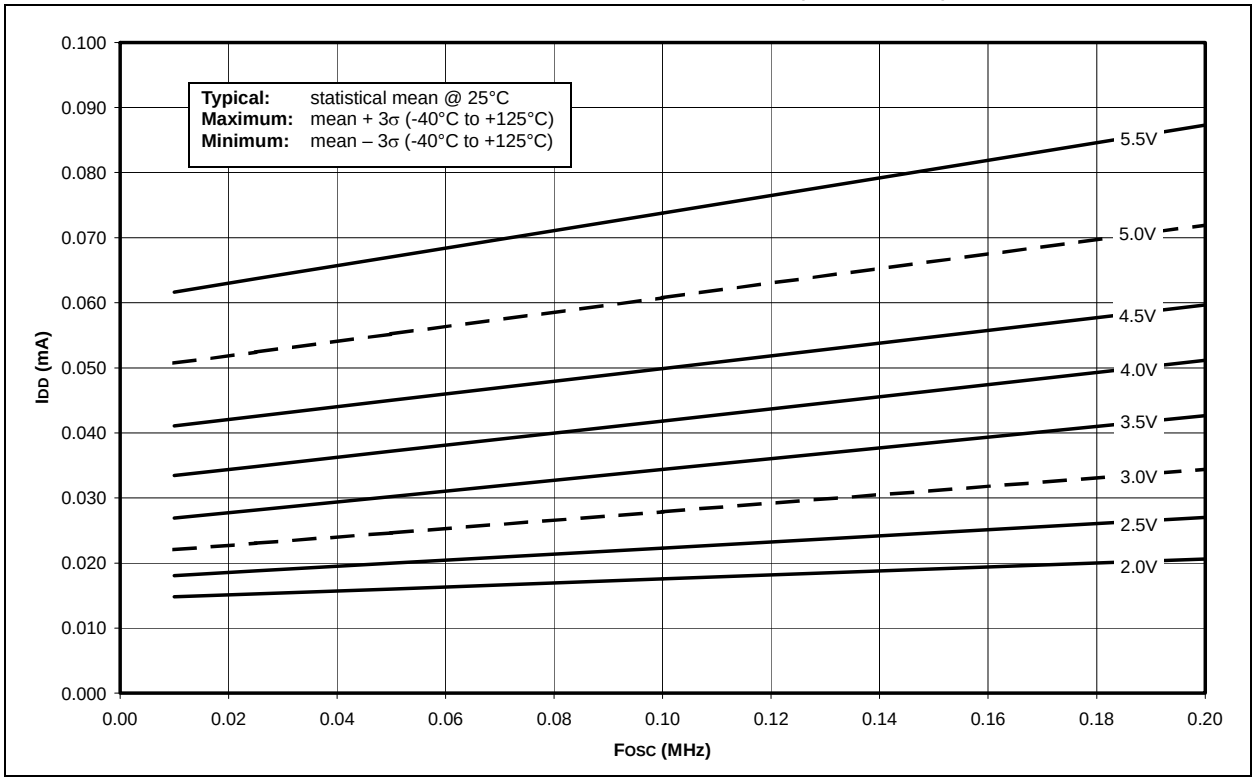



**FIGURE 23-10: MAXIMUM IDD vs. FOSC OVER VDD PRI_IDLE, EC MODE, -40°C TO +125°C**





© 2002-2015 Microchip Technology Inc.

**FIGURE 23-11: TYPICAL IDD vs. FOSC OVER VDD PRI_IDLE, EC MODE, +25°C**

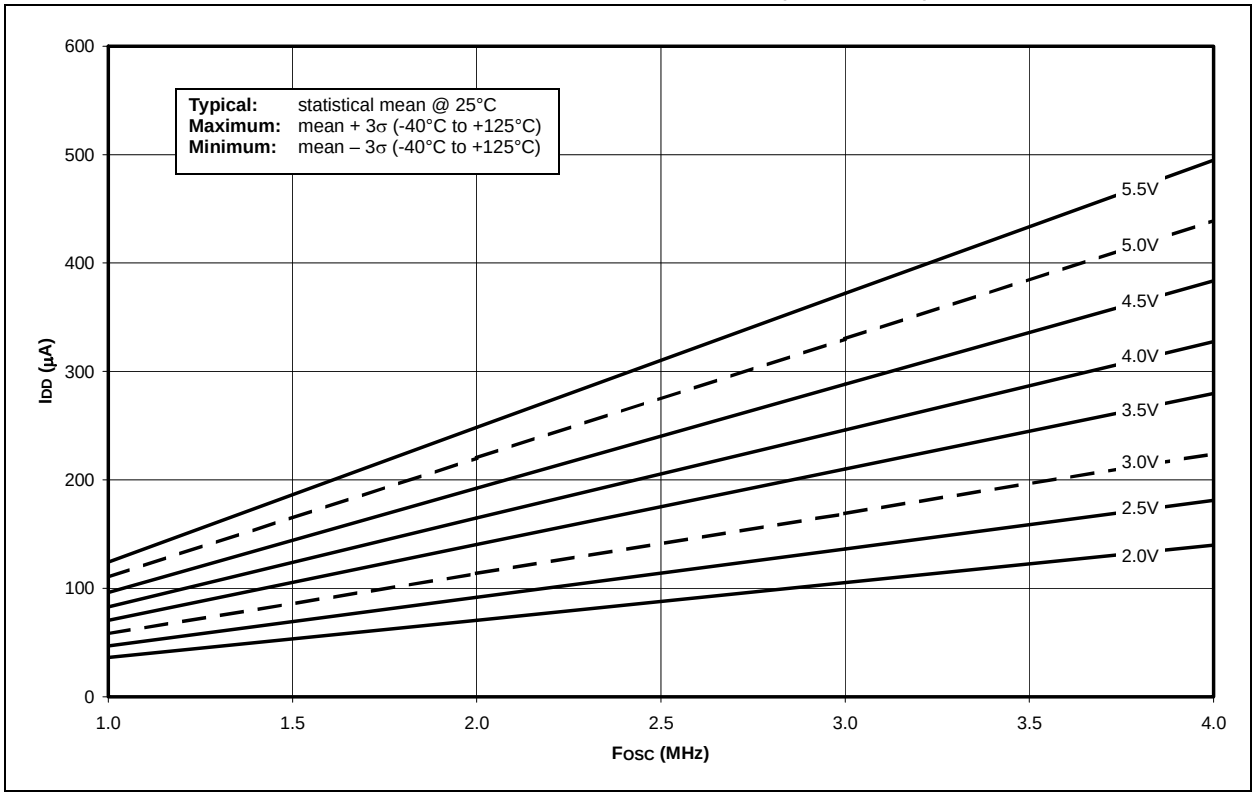



**FIGURE 23-12: MAXIMUM IDD vs. FOSC OVER VDD PRI_IDLE, EC MODE, -40°C TO +125°C**

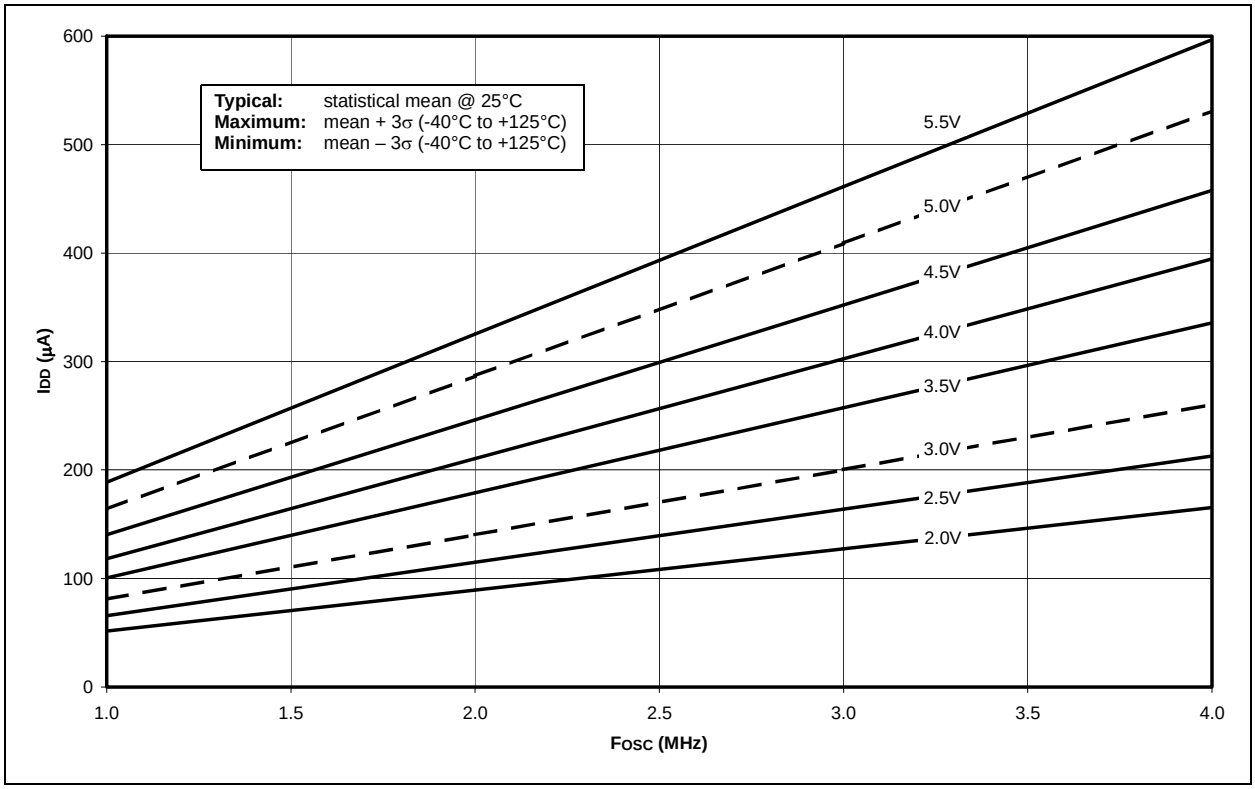


**FIGURE 23-13: TYPICAL IDD vs. FOSC OVER VDD PRI_IDLE, EC MODE, +25°C**

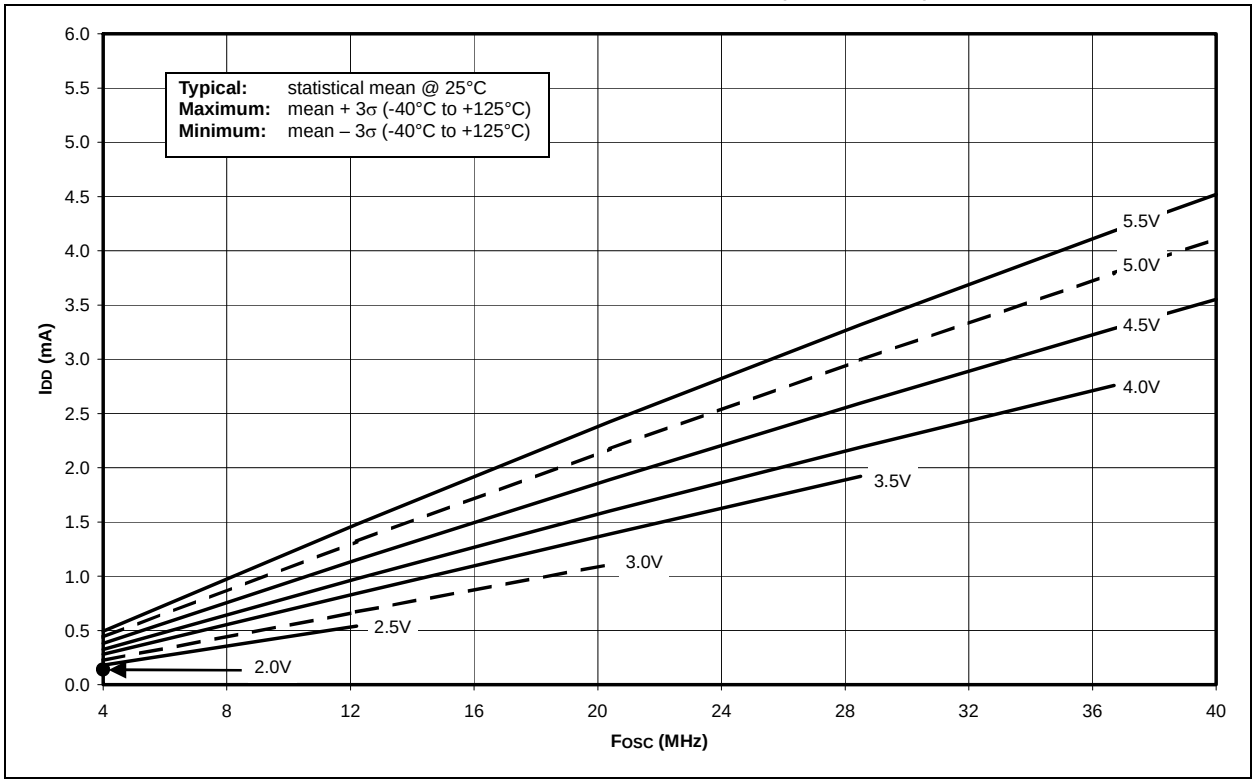
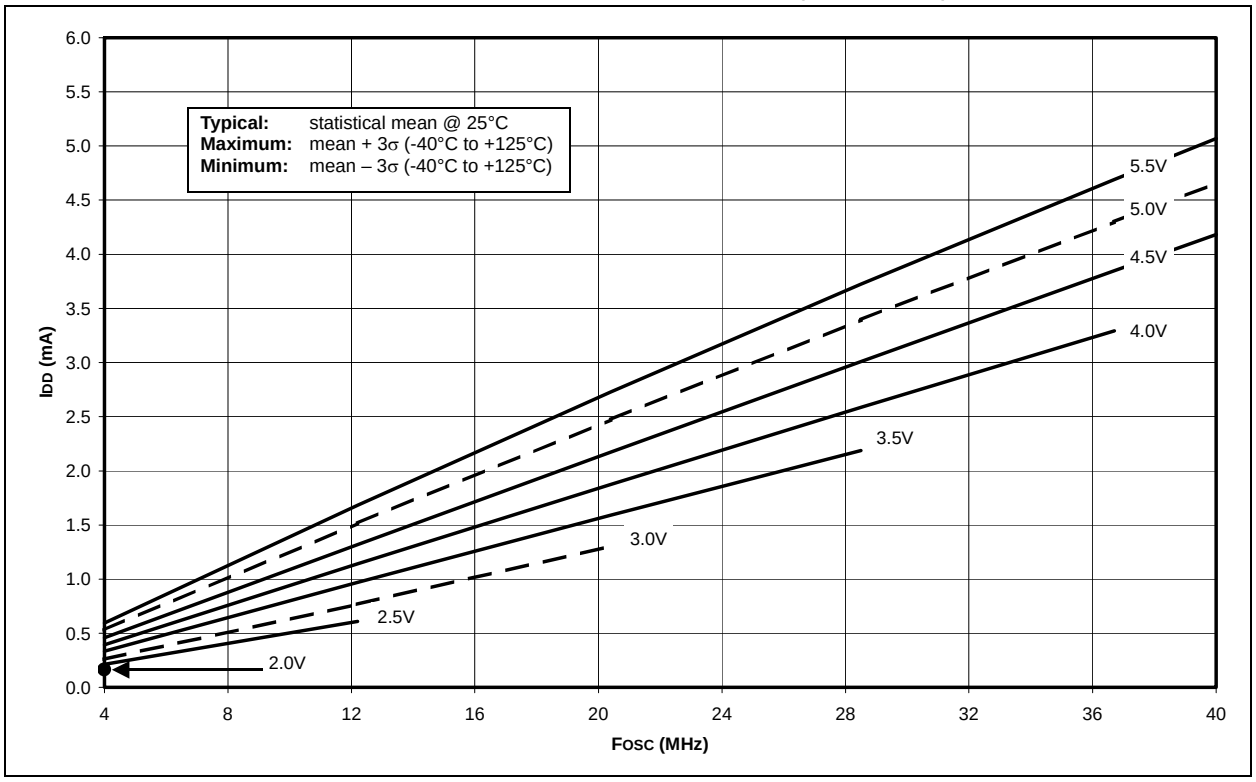



**FIGURE 23-14: MAXIMUM IDD vs. FOSC OVER VDD PRI_IDLE, EC MODE, -40°C TO +125°C**





© 2002-2015 Microchip Technology Inc.

**FIGURE 23-15: TYPICAL IPD vs. VDD (+25°C), 125 kHz TO 8 MHz RC_RUN MODE, ALL PERIPHERALS DISABLED**

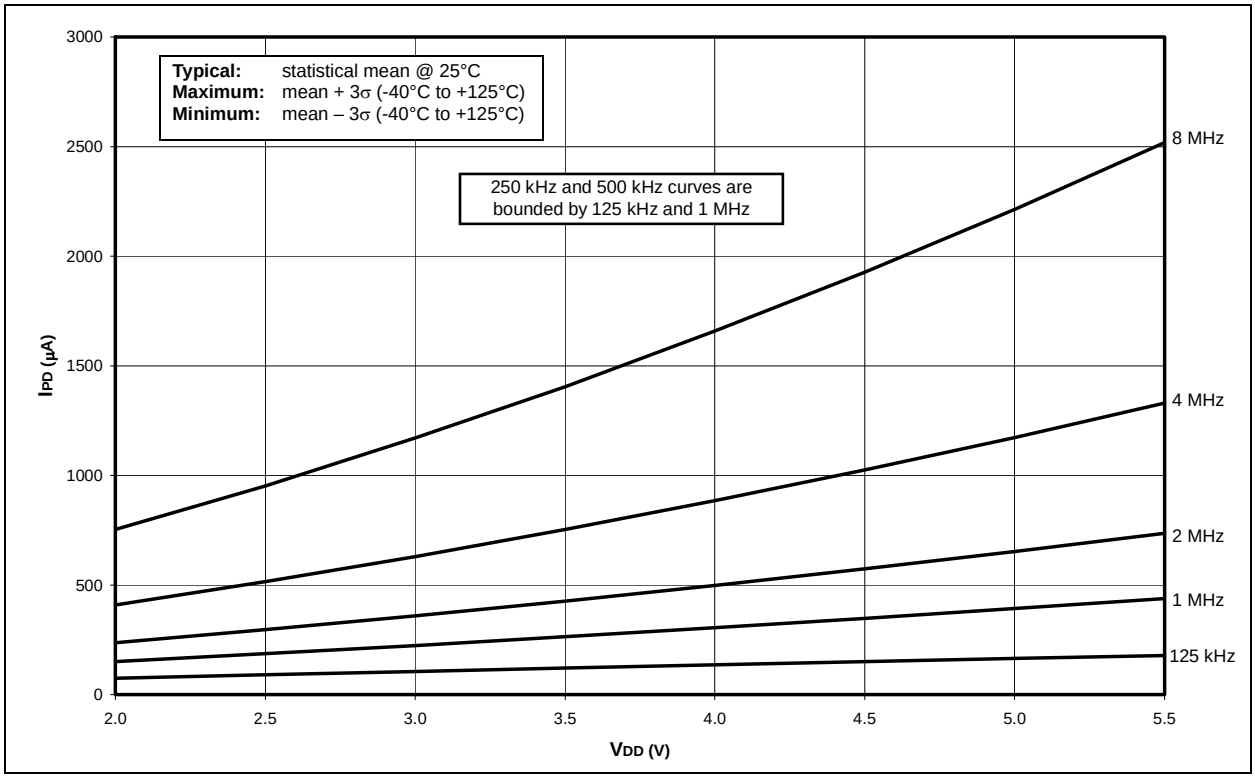



**FIGURE 23-16: MAXIMUM IPD vs. VDD (-40°C TO +125°C), 125 kHz TO 8 MHz RC_RUN MODE, ALL PERIPHERALS DISABLED**

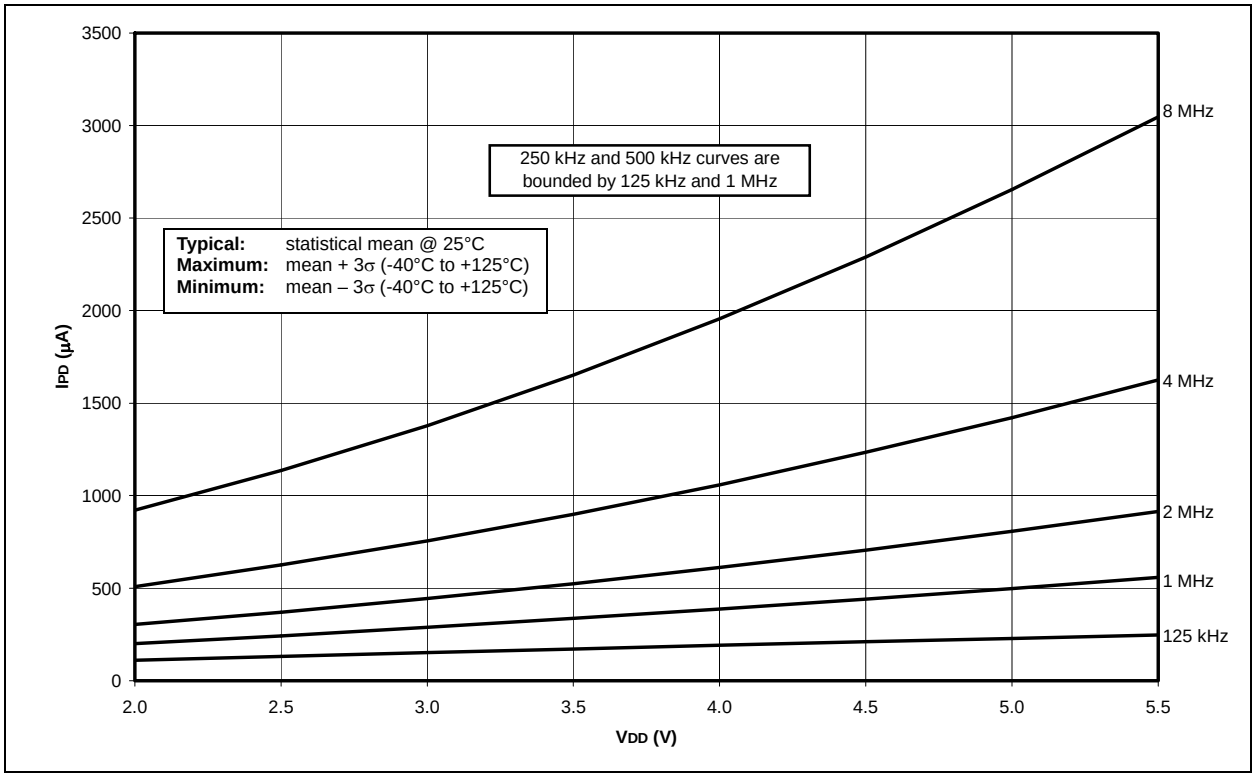


**FIGURE 23-17: TYPICAL AND MAXIMUM $I_{PD}$ vs. $V_{DD}$ (-40°C TO +125°C), 31.25 kHz RC_RUN MODE, ALL PERIPHERALS DISABLED**

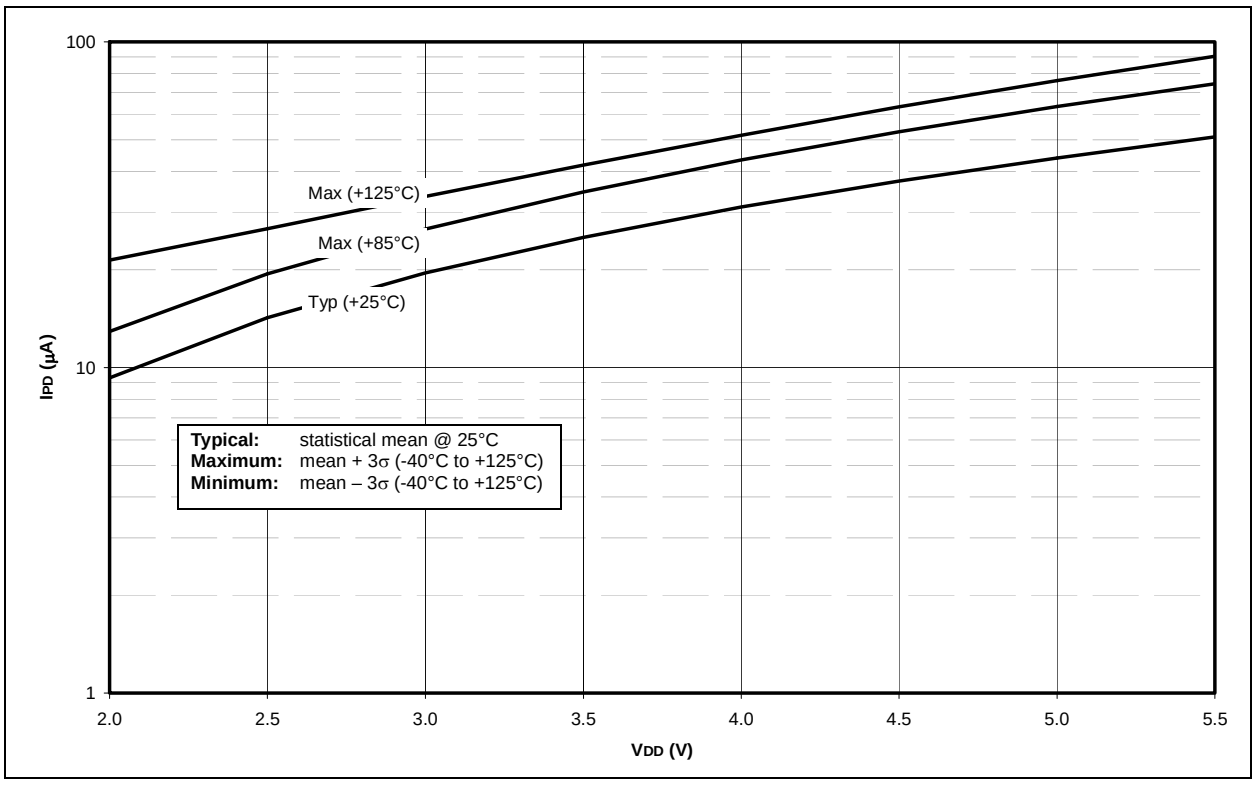
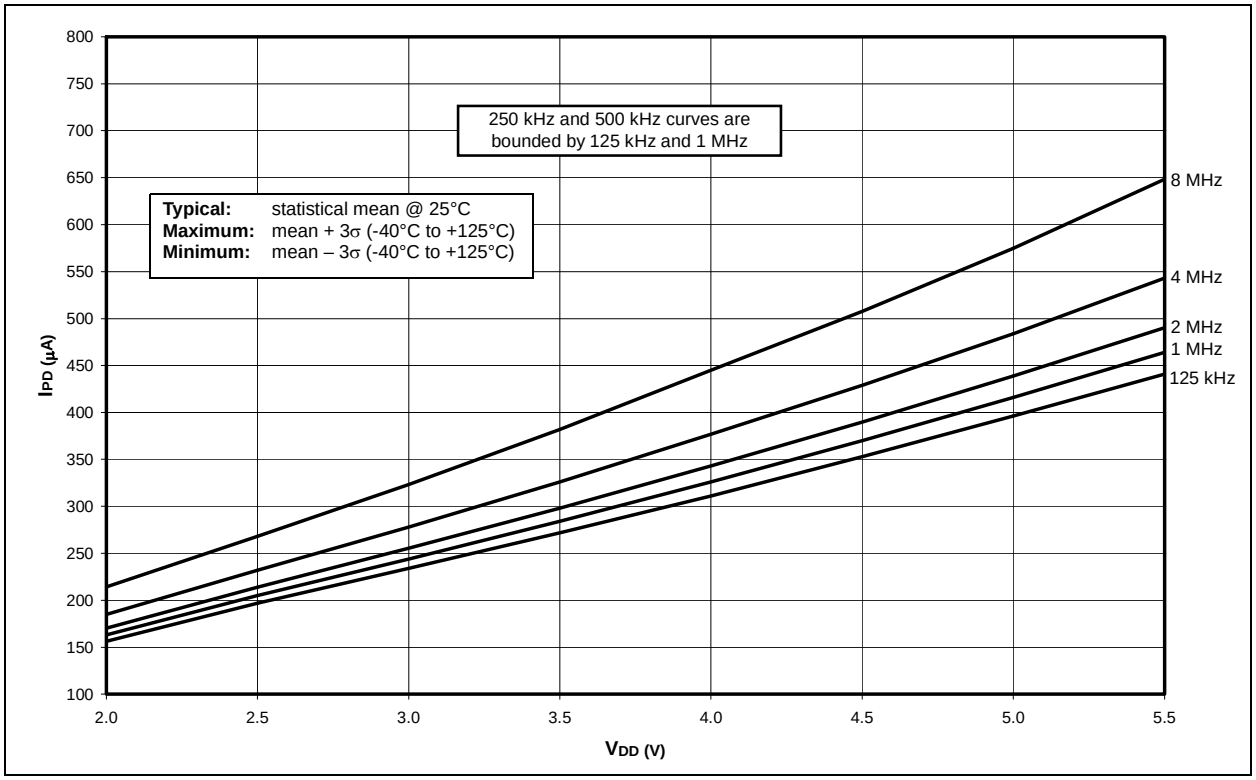



**FIGURE 23-18: TYPICAL $I_{PD}$ vs. $V_{DD}$ (+25°C), 125 kHz TO 8 MHz RC_IDLE MODE, ALL PERIPHERALS DISABLED**





© 2002-2015 Microchip Technology Inc.

**FIGURE 23-19: MAXIMUM IPD vs. VDD (-40°C TO +125°C), 125 kHz TO 8 MHz RC_IDLE MODE, ALL PERIPHERALS DISABLED**

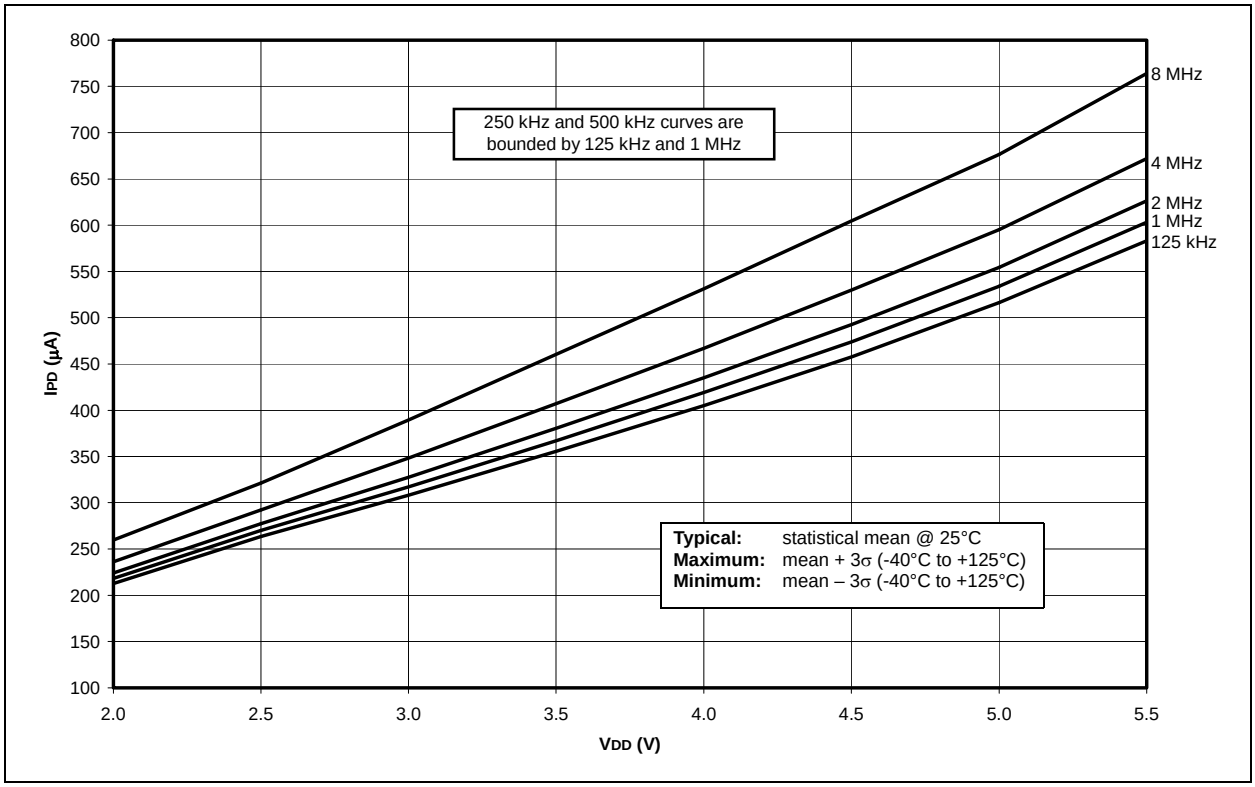



**FIGURE 23-20: TYPICAL AND MAXIMUM IPD vs. VDD (-40°C TO +125°C), 31.25 kHz RC_IDLE MODE, ALL PERIPHERALS DISABLED**

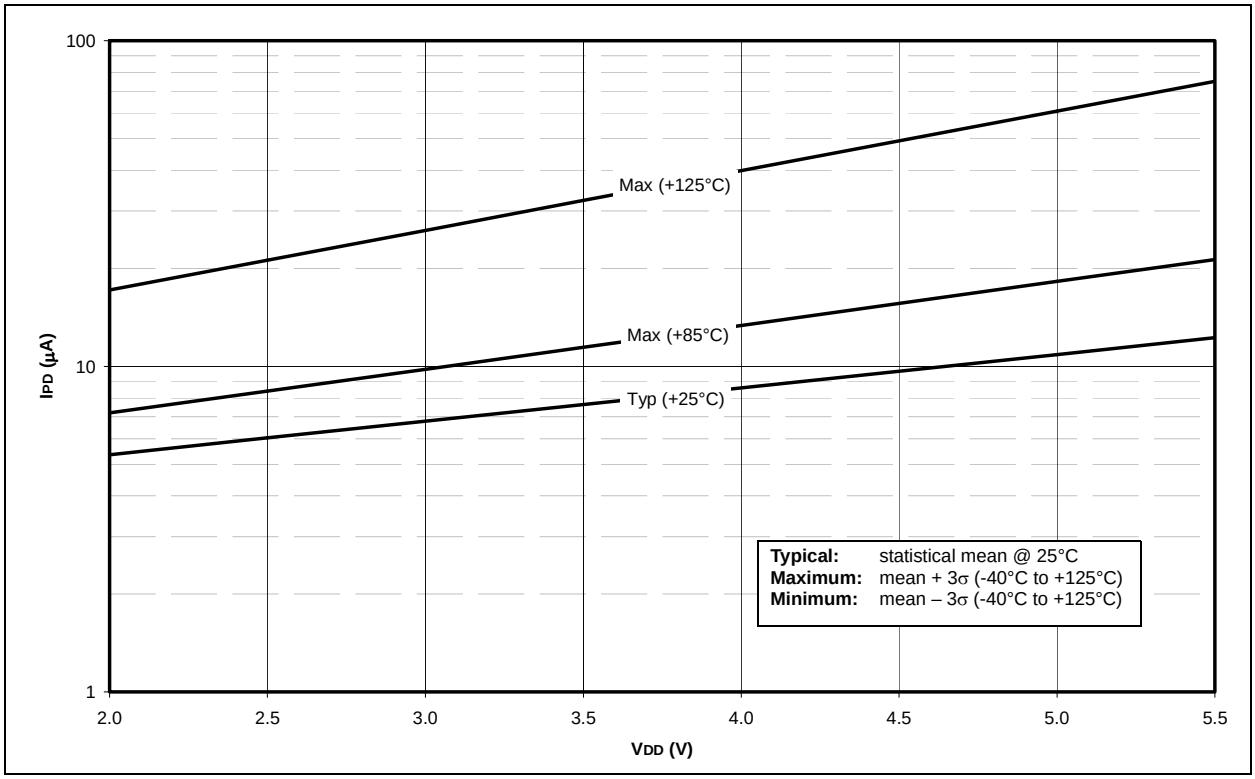


**FIGURE 23-21: IPD SEC_RUN MODE, -10°C TO +70°C, 32.768 kHz XTAL, 2 x 22 pF, ALL PERIPHERALS DISABLED**

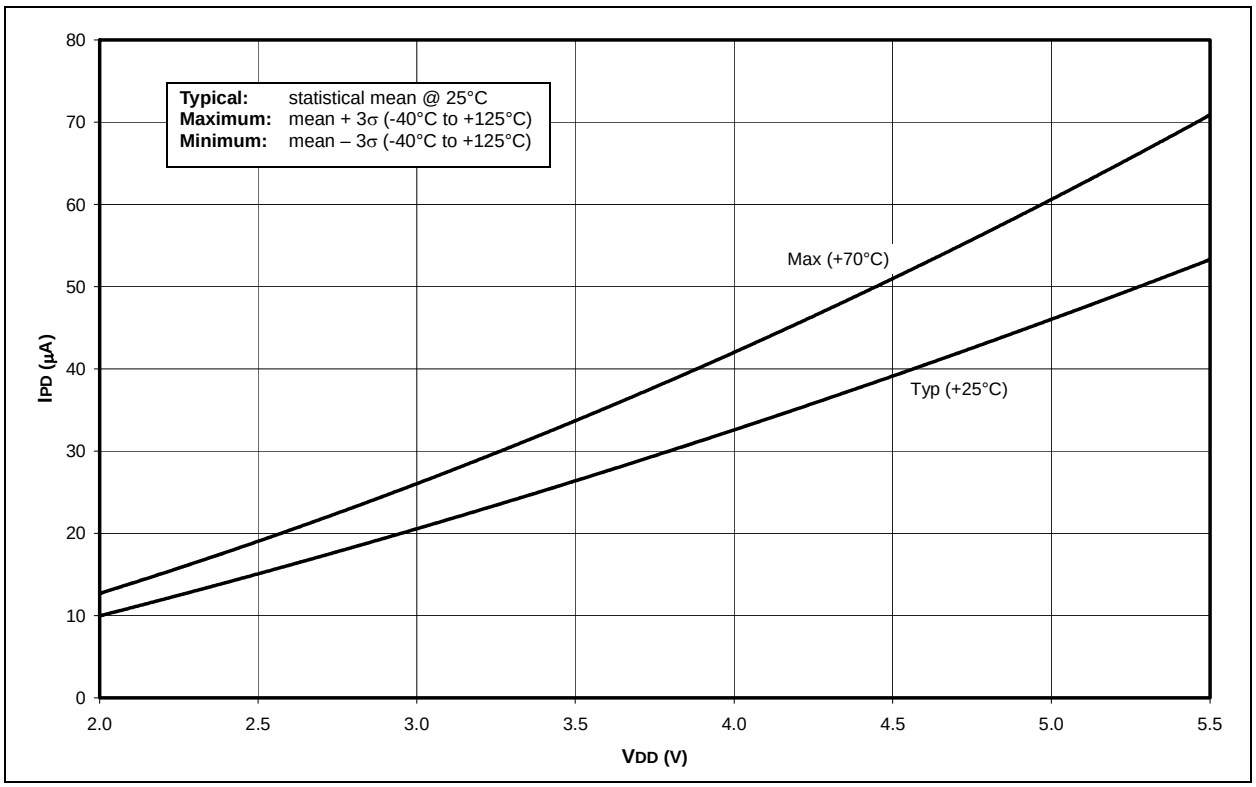
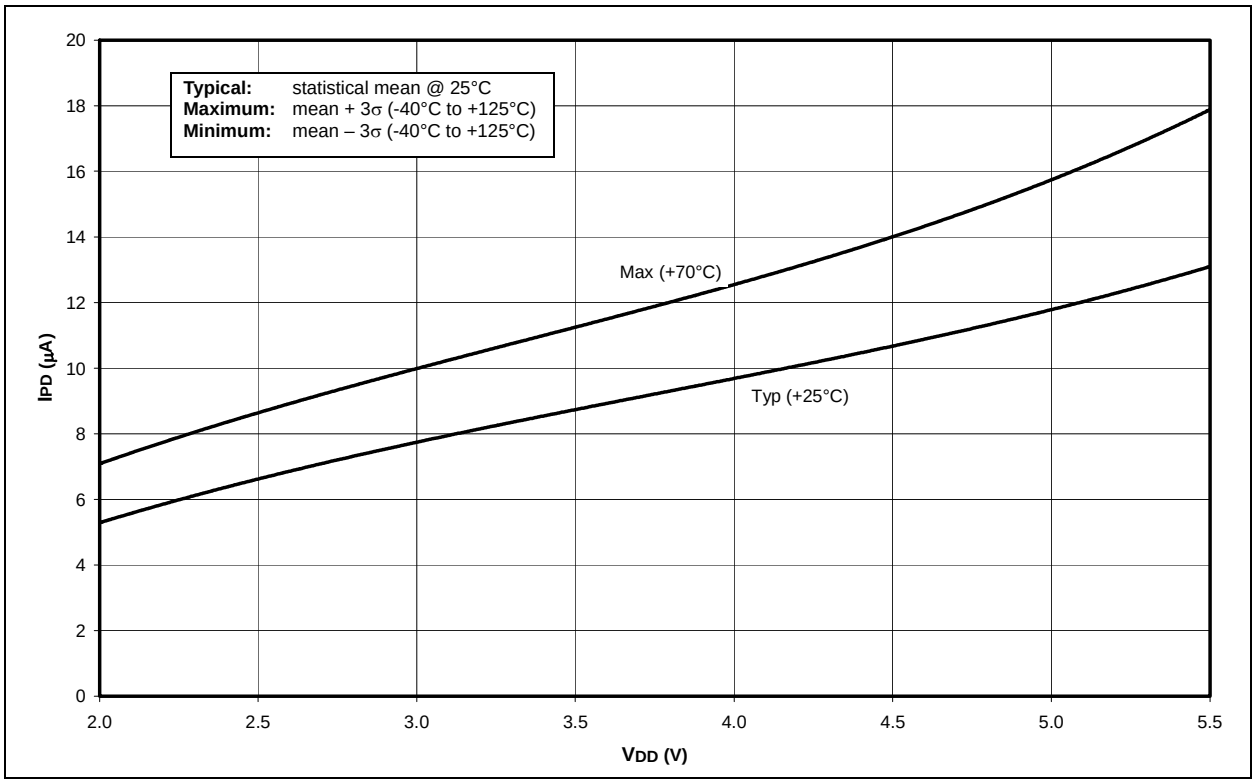



**FIGURE 23-22: IPD SEC_IDLE MODE, -10°C TO +70°C, 32.768 kHz, 2 x 22 pF, ALL PERIPHERALS DISABLED**





© 2002-2015 Microchip Technology Inc.

**FIGURE 23-23: TOTAL IPD, -40°C TO +125°C SLEEP MODE, ALL PERIPHERALS DISABLED**

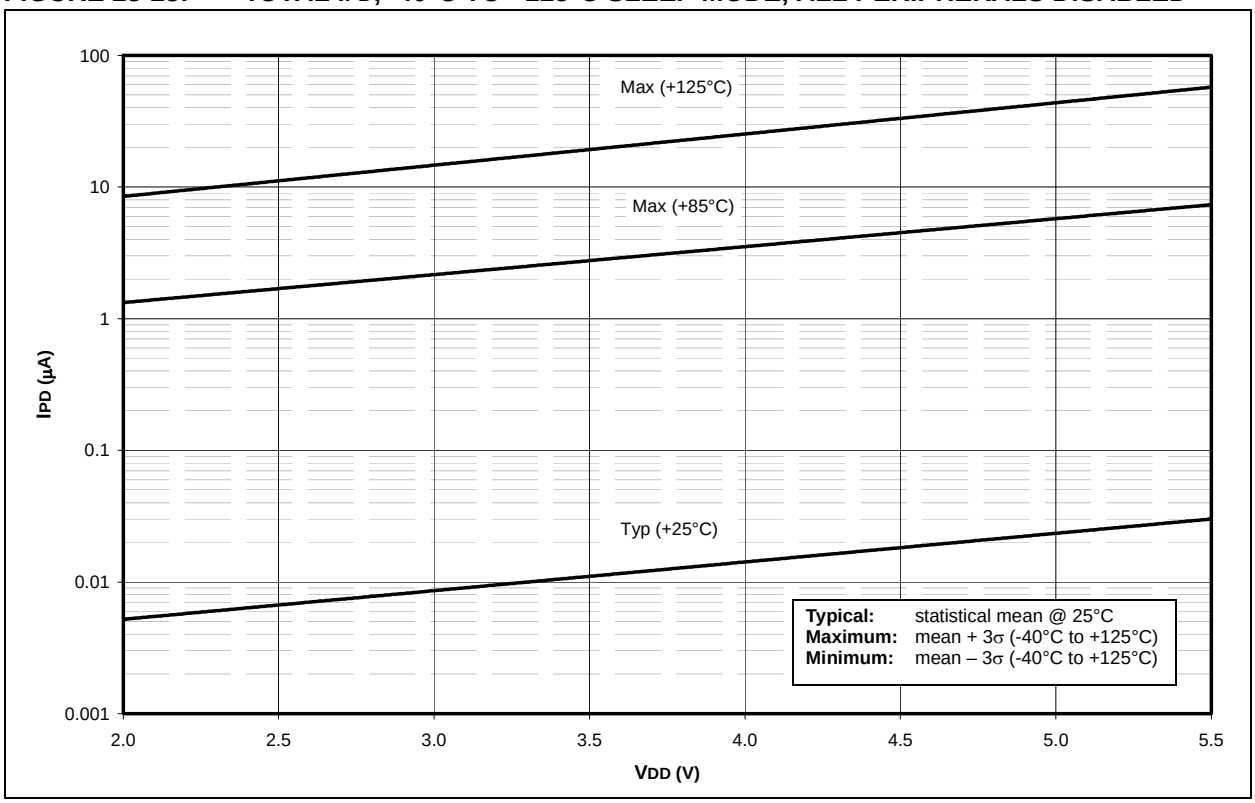



**FIGURE 23-24: VOH vs. IOH OVER TEMPERATURE (-40°C TO +125°C), VDD = 3.0V**

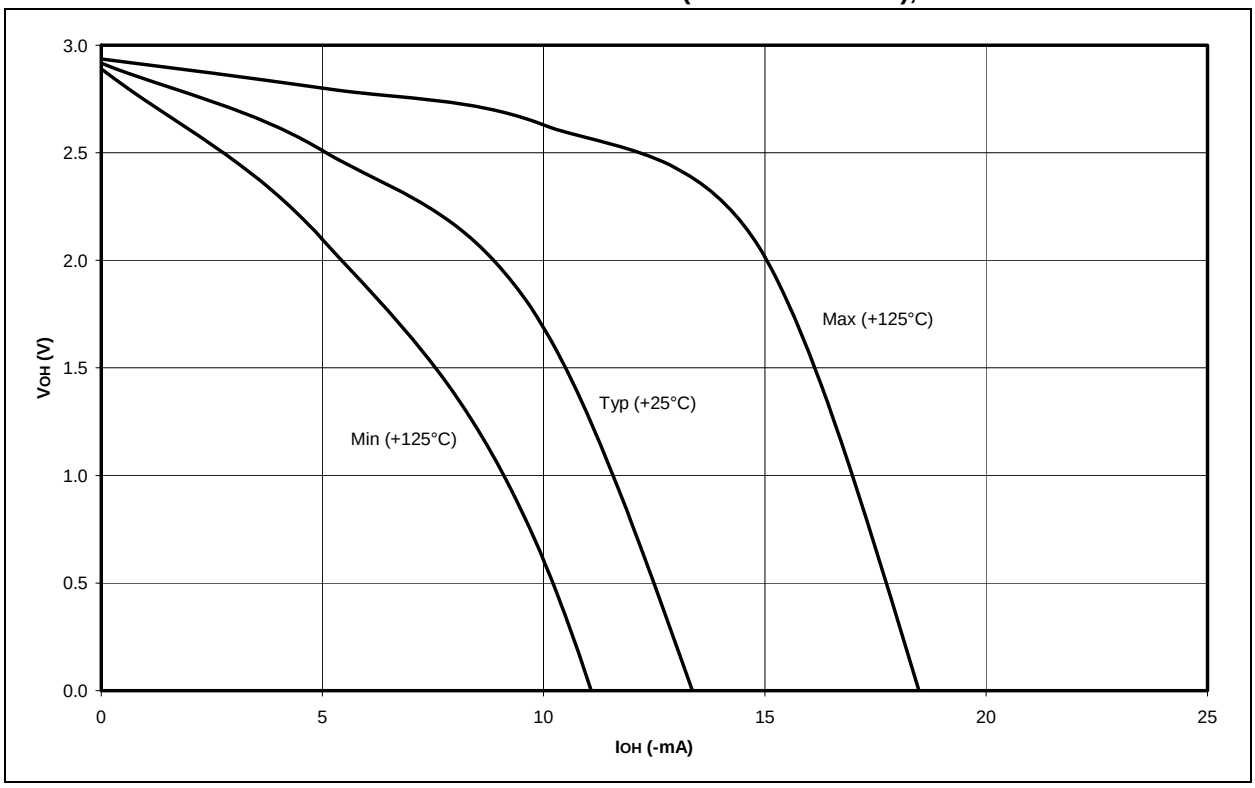


**FIGURE 23-25: $V_{OH}$ vs. $I_{OH}$ OVER TEMPERATURE (-40°C TO +125°C), $V_{DD}$ = 5.0V**

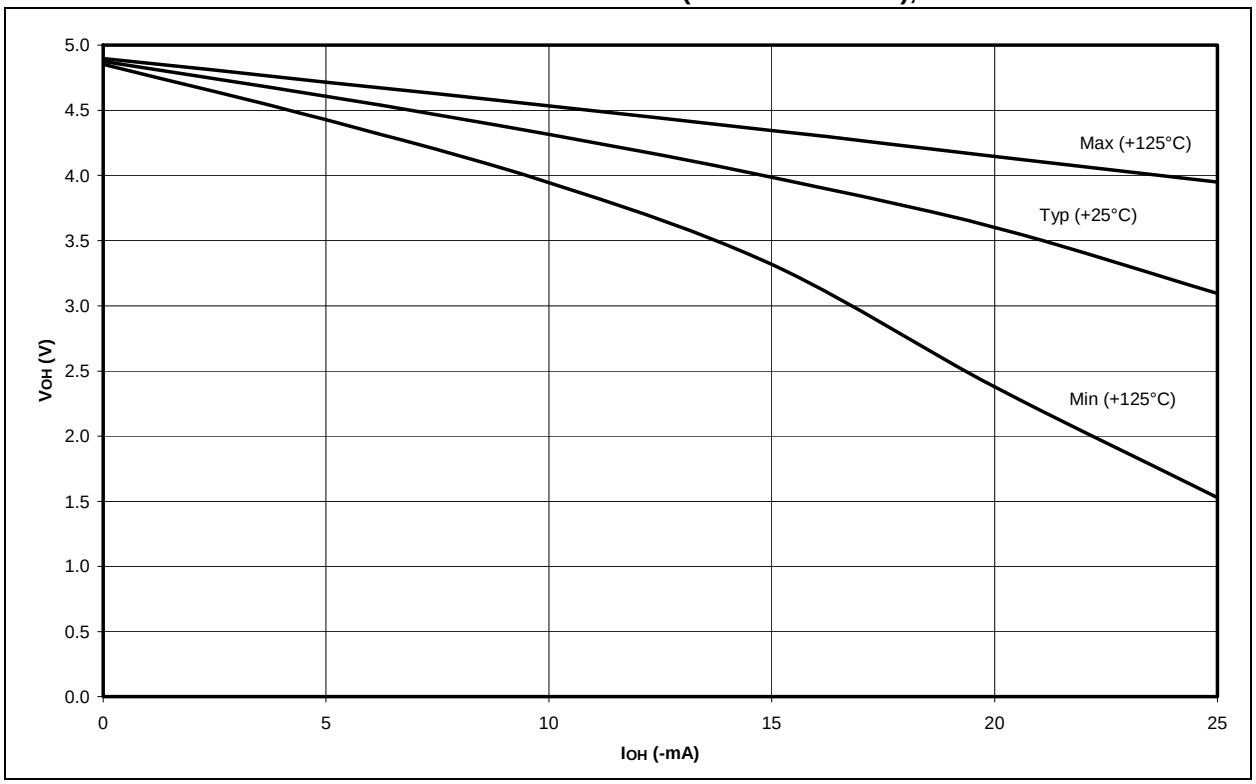
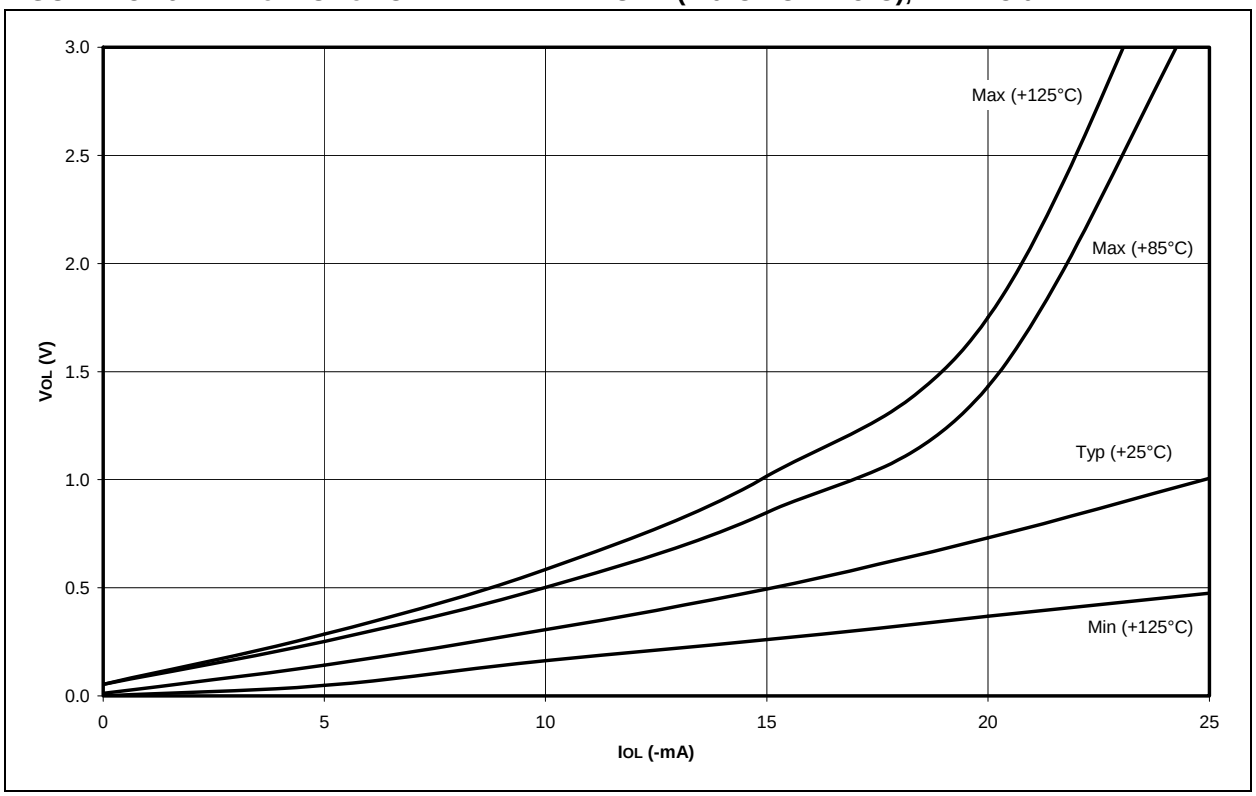



**FIGURE 23-26: $V_{OL}$ vs. $I_{OL}$ OVER TEMPERATURE (-40°C TO +125°C), $V_{DD}$ = 3.0V**





© 2002-2015 Microchip Technology Inc.

**FIGURE 23-27: $V_{OL}$ vs. $I_{OL}$ OVER TEMPERATURE (-40°C TO +125°C), $V_{DD}$ = 5.0V**

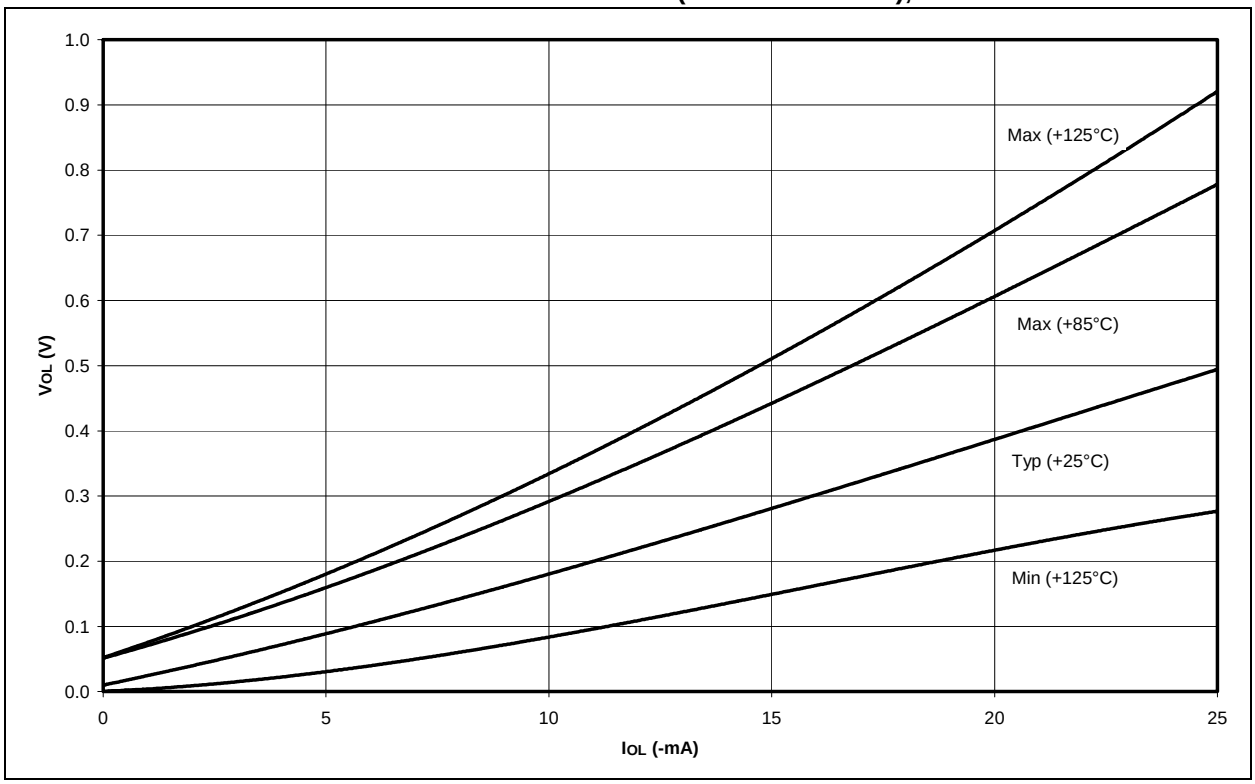



**FIGURE 23-28: $\Delta I_{PD}$ TIMER1 OSCILLATOR, -10°C TO +70°C SLEEP MODE, TMR1 COUNTER DISABLED**

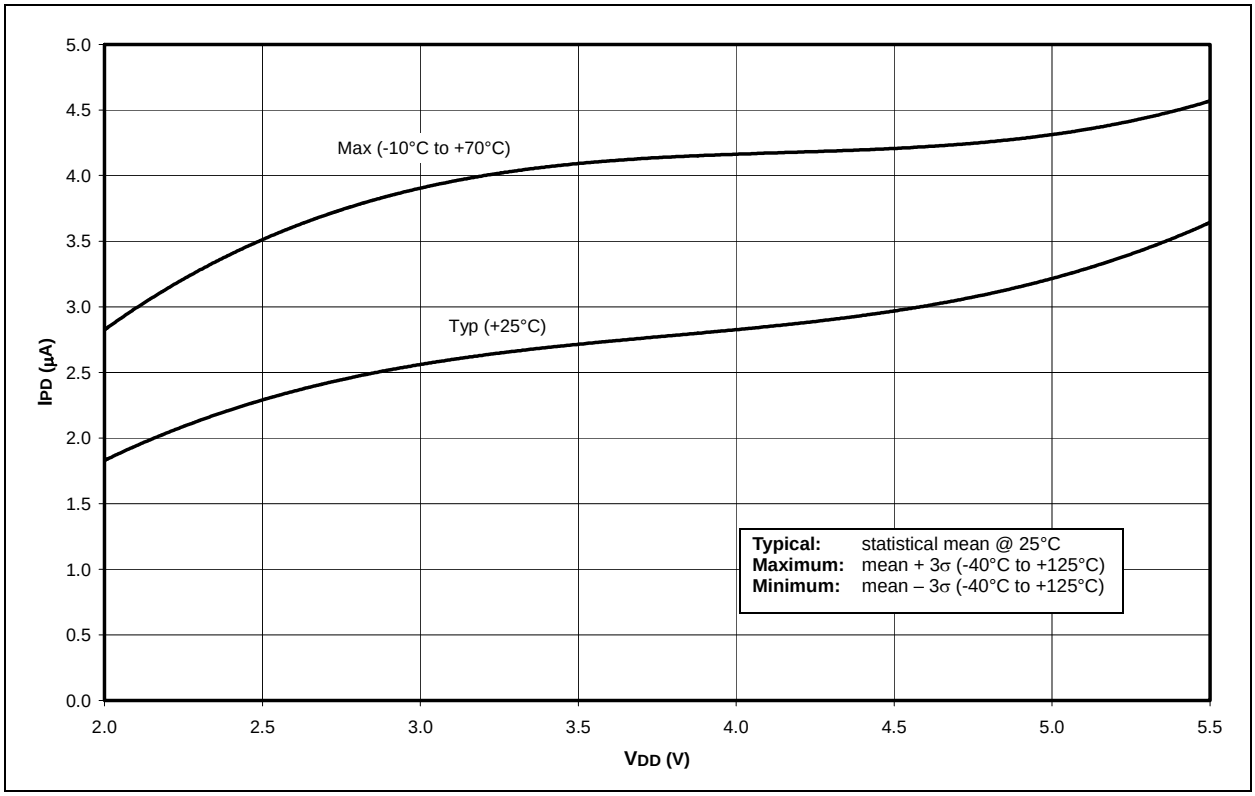


**FIGURE 23-29: ΔIPD FSCM vs. VDD OVER TEMPERATURE PRI_IDLE MODE, EC OSCILLATOR AT 32 kHz, -40°C TO +125°C**

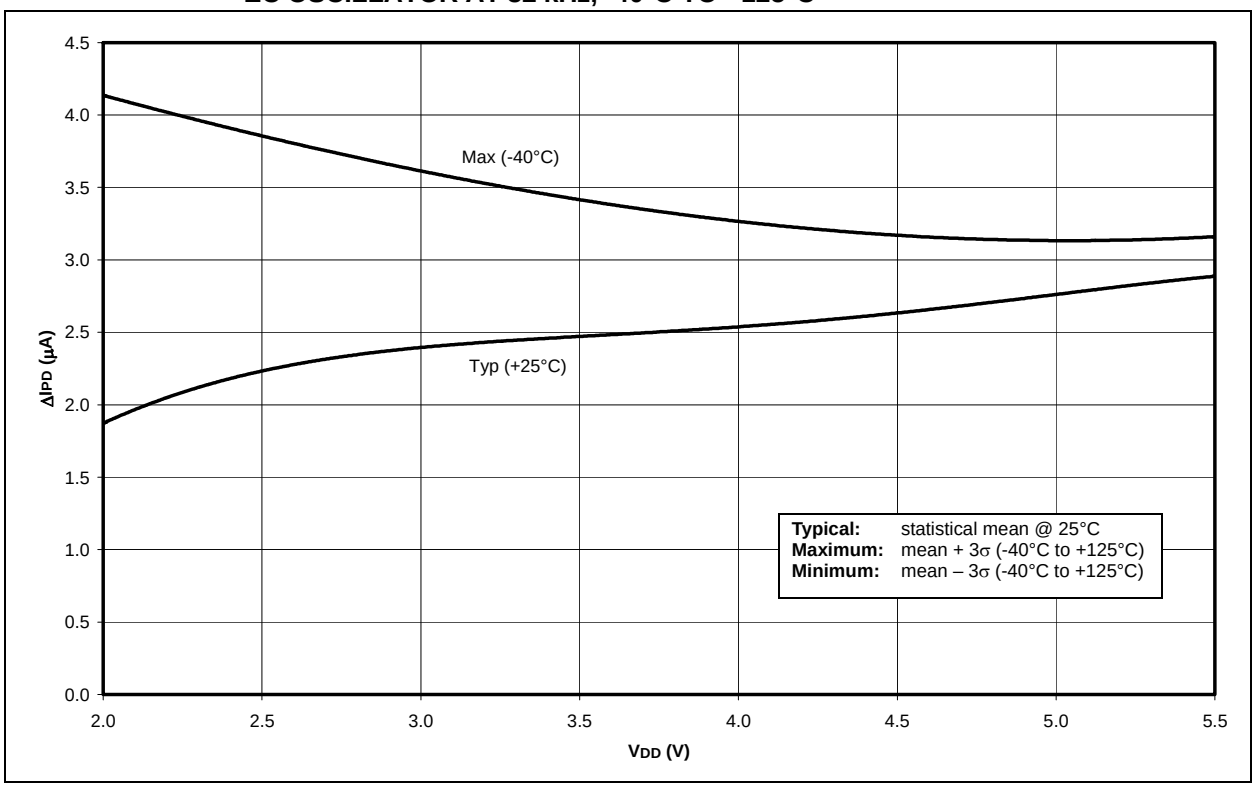
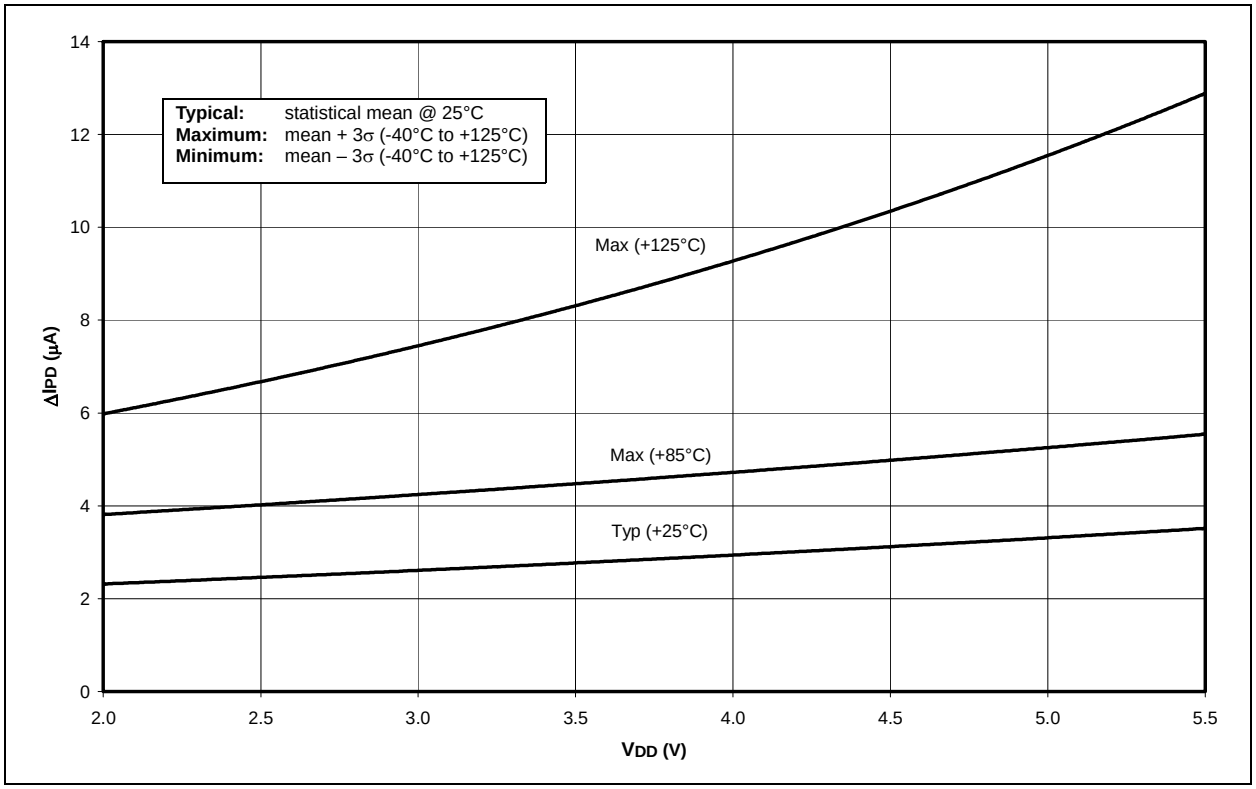



**FIGURE 23-30: ΔIPD WDT, -40°C TO +125°C SLEEP MODE, ALL PERIPHERALS DISABLED**





© 2002-2015 Microchip Technology Inc.

**FIGURE 23-31: ΔIPD LVD vs. VDD SLEEP MODE, LVDL3:LVDL0 = 0001 (2V)**

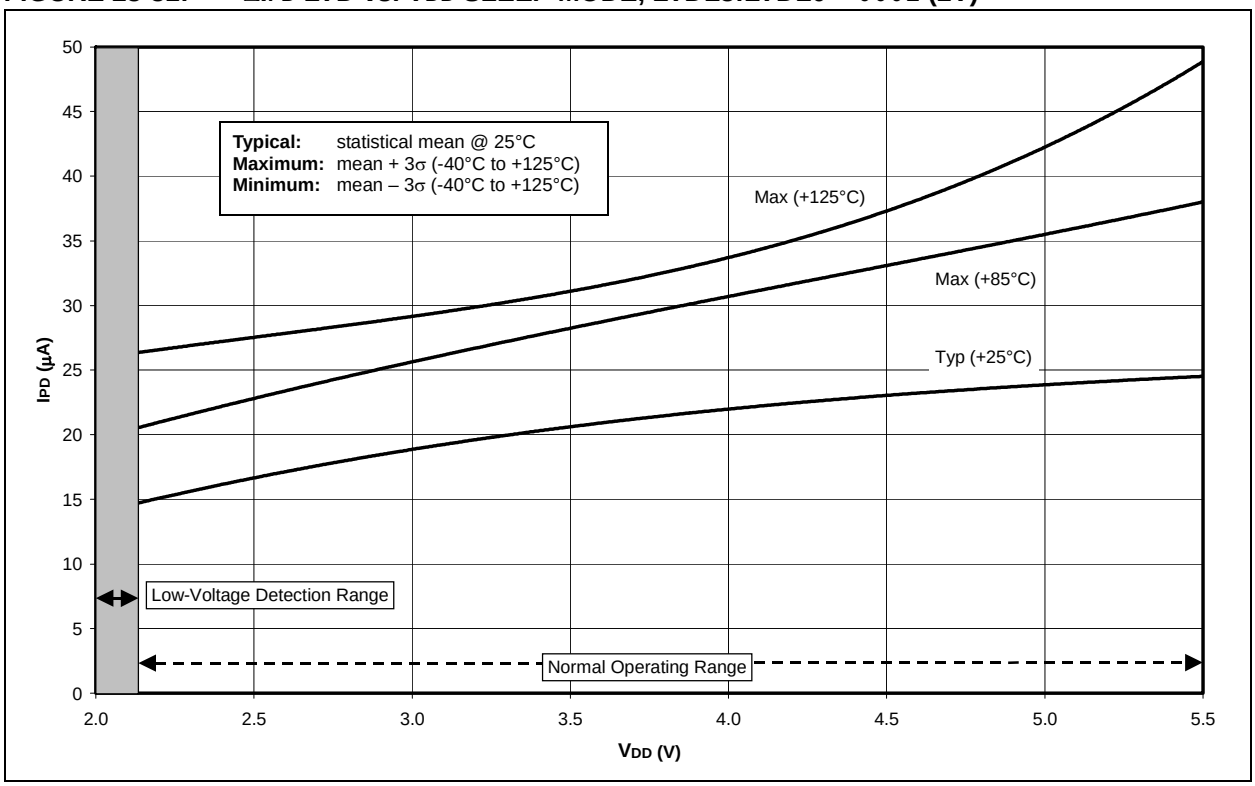



**FIGURE 23-32: ΔIPD BOR vs. VDD, -40°C TO +125°C SLEEP MODE, BORV1:BORV0 = 11 (2V)**

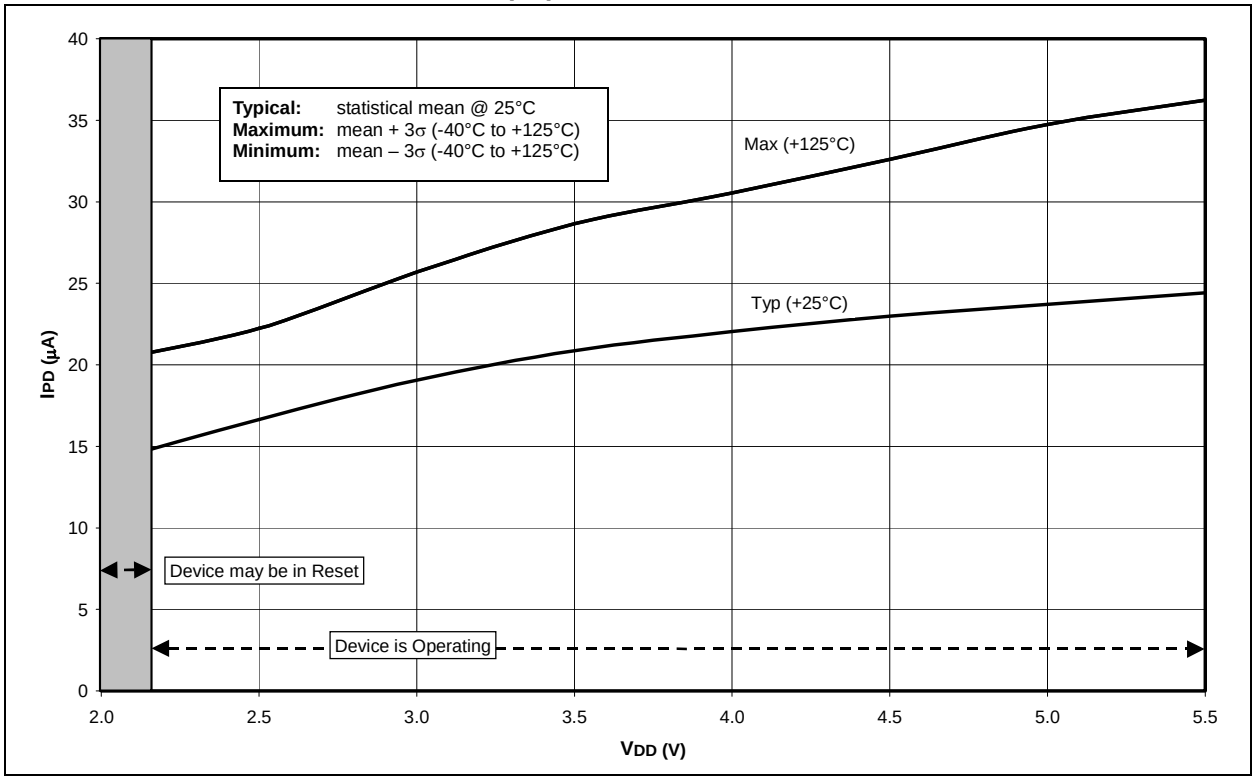


**FIGURE 23-33: ΔIPD A/D, -40°C TO +125°C SLEEP MODE, A/D ENABLED (NOT CONVERTING)**

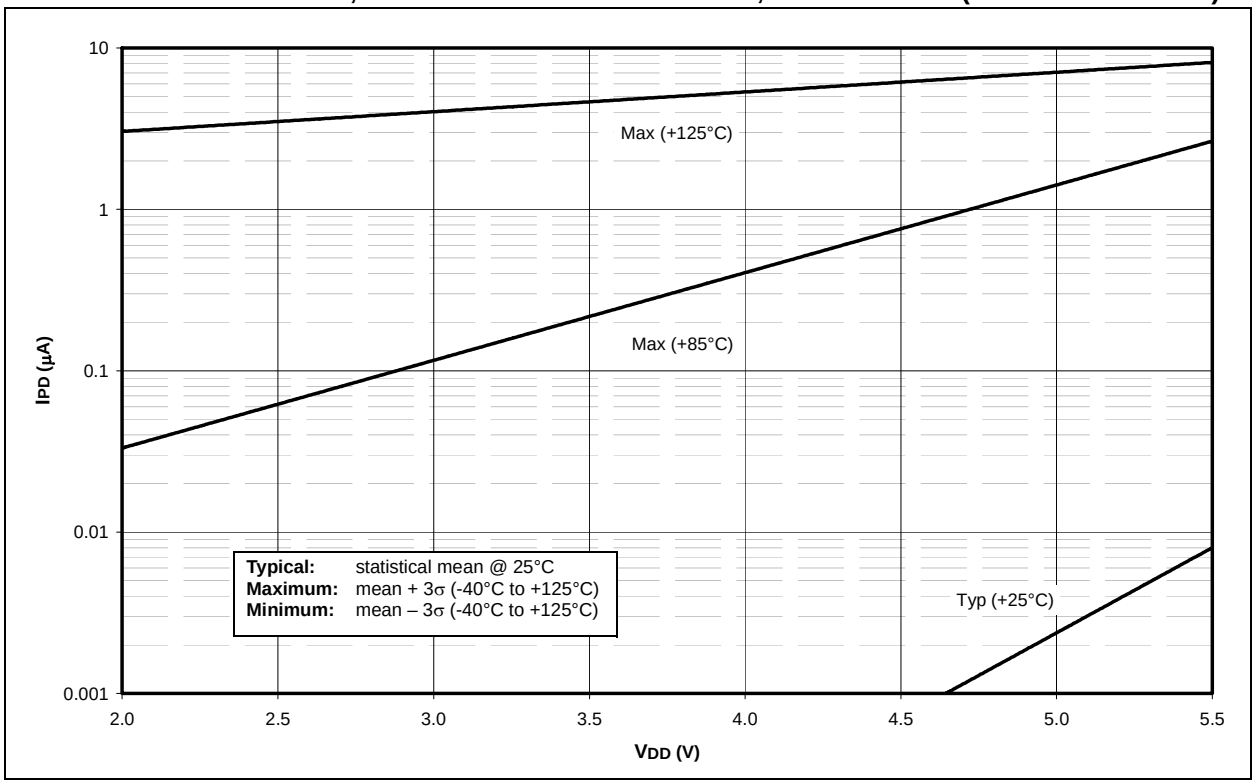
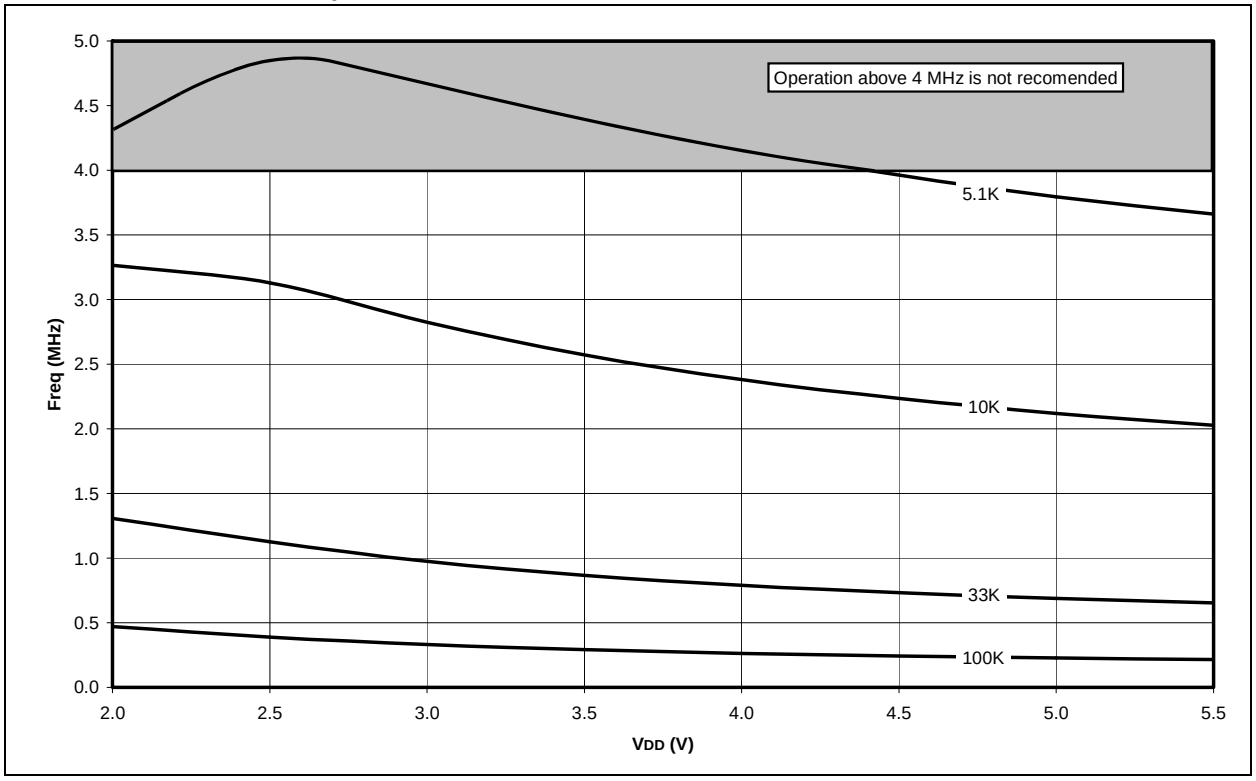



**FIGURE 23-34: AVERAGE FOSC vs. VDD FOR VARIOUS R's EXTERNAL RC MODE, C = 20 pF, TEMPERATURE = +25°C**





© 2002-2015 Microchip Technology Inc.

**FIGURE 23-35: AVERAGE FOSC vs. VDD FOR VARIOUS R's EXTERNAL RC MODE, C = 100 pF, TEMPERATURE = +25°C**

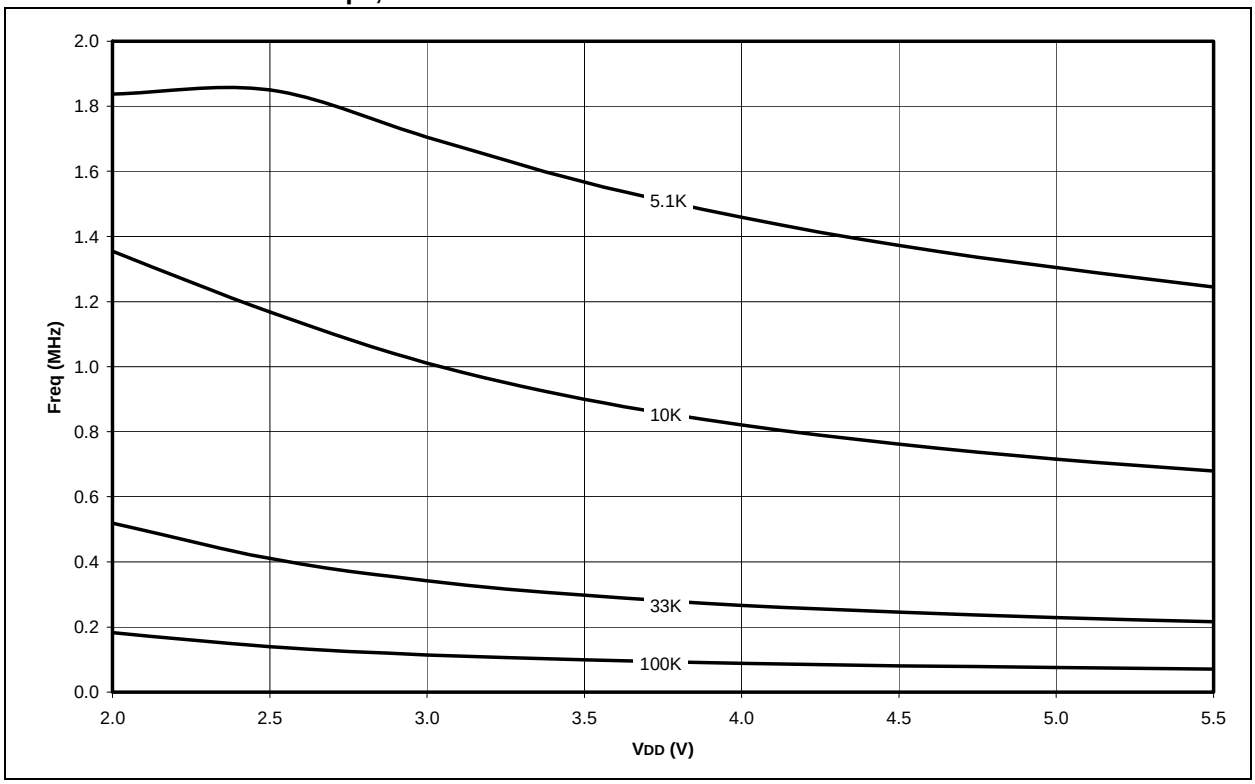
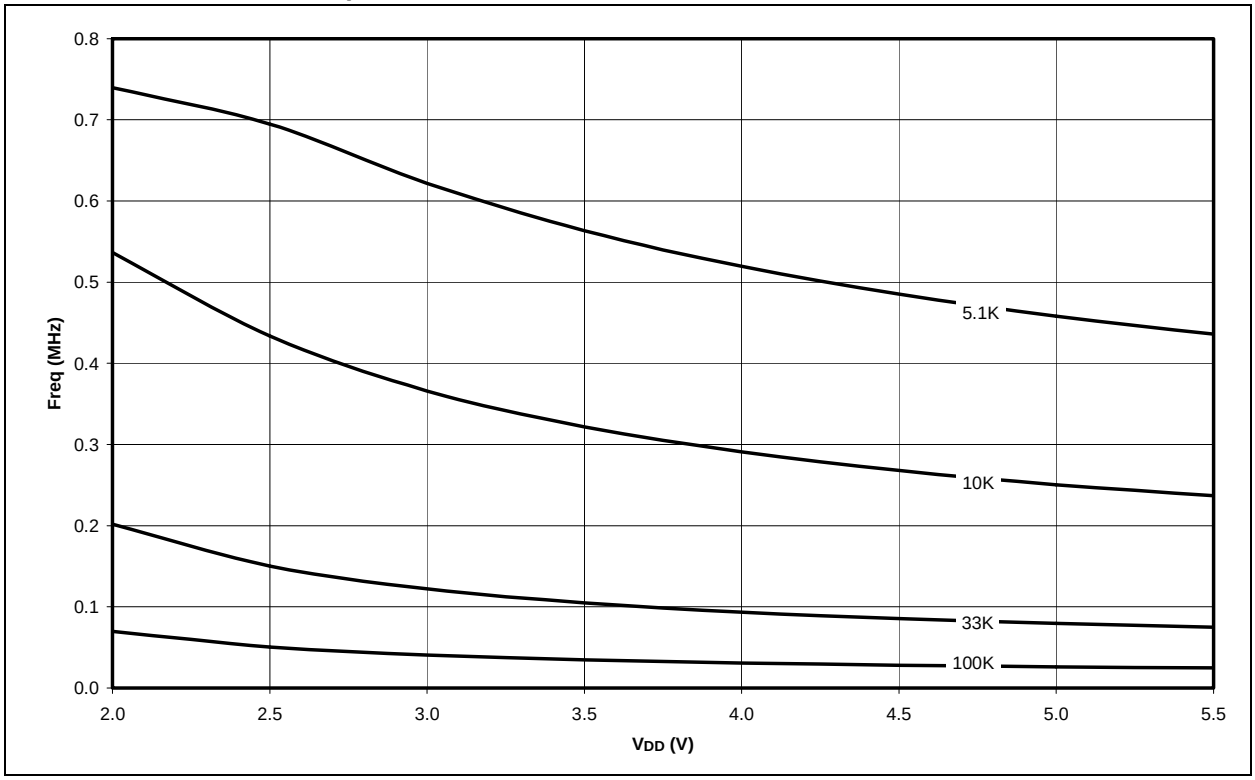


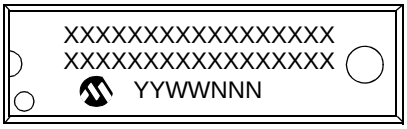

**FIGURE 23-36: AVERAGE FOSC vs. VDD FOR VARIOUS R's EXTERNAL RC MODE, C = 300 pF, TEMPERATURE = +25°C**



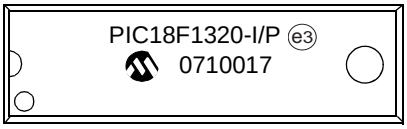
## 24.0 PACKAGING INFORMATION

### 24.1 Package Marking Information

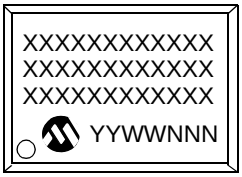
18-Lead PDIP



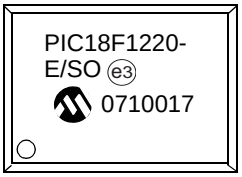

Example



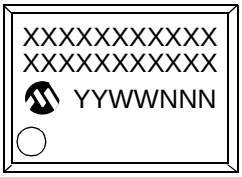

18-Lead SOIC



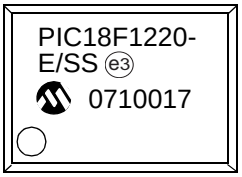

Example



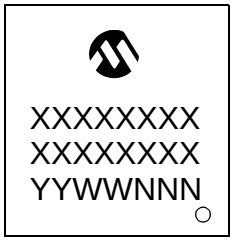

20-Lead SSOP



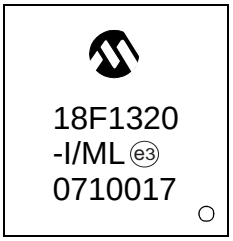

Example




28-Lead QFN




Example




| | | |
|---|---|---|
| **Legend:** | 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.

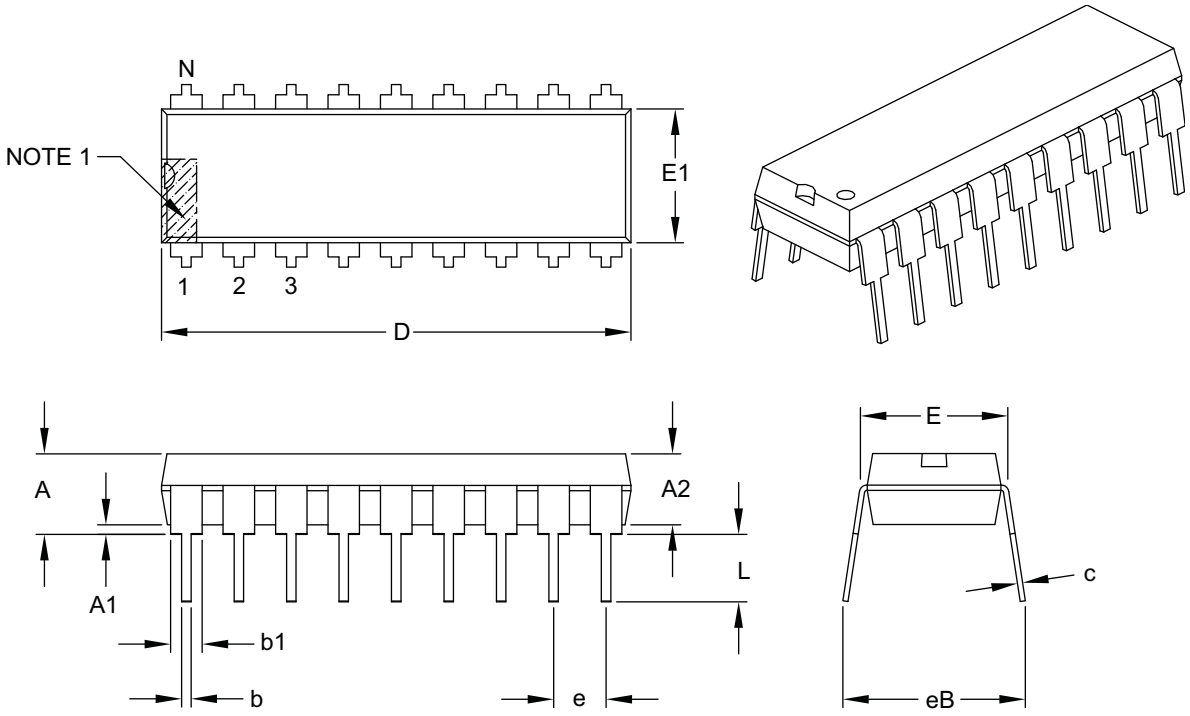

© 2002-2015 Microchip Technology Inc.

## 24.2 Package Details

The following sections give the technical details of the packages.

### 18-Lead Plastic Dual In-Line (P) – 300 mil Body [PDIP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




| Units | | INCHES | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 18 | |
| Pitch | e | | .100 BSC | |
| Top to Seating Plane | A | – | – | .210 |
| Molded Package Thickness | A2 | .115 | .130 | .195 |
| Base to Seating Plane | A1 | .015 | – | – |
| Shoulder to Shoulder Width | E | .300 | .310 | .325 |
| Molded Package Width | E1 | .240 | .250 | .280 |
| Overall Length | D | .880 | .900 | .920 |
| Tip to Seating Plane | L | .115 | .130 | .150 |
| Lead Thickness | c | .008 | .010 | .014 |
| Upper Lead Width | b1 | .045 | .060 | .070 |
| Lower Lead Width | b | .014 | .018 | .022 |
| Overall Row Spacing § | eB | – | – | .430 |

**Notes:**

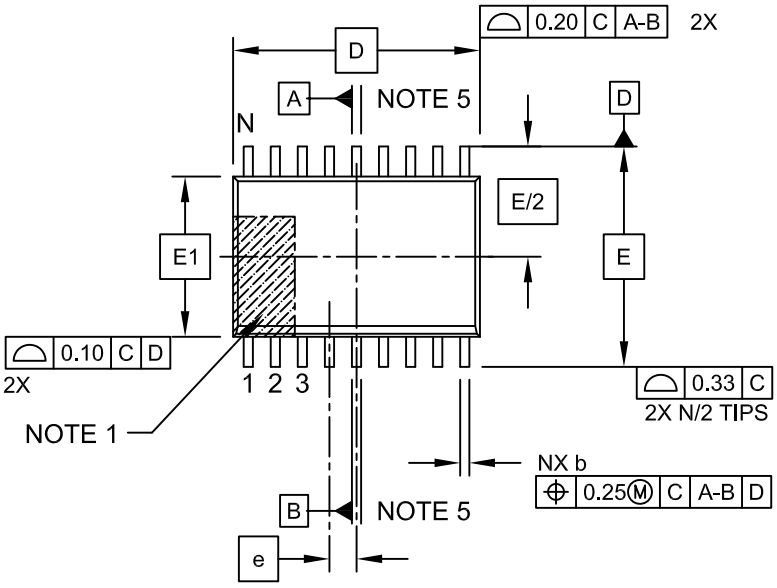
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

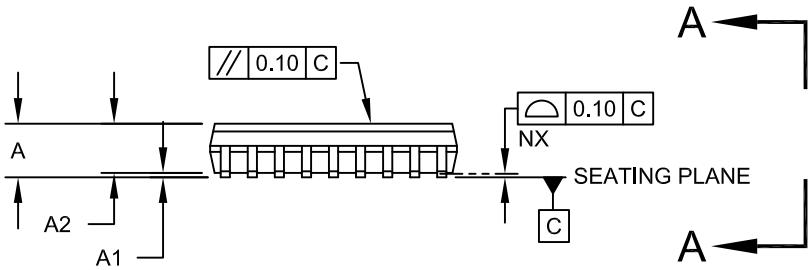
Microchip Technology Drawing C04-007B

### 18-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

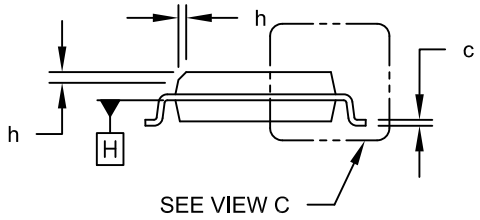
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




**TOP VIEW**




**SIDE VIEW**



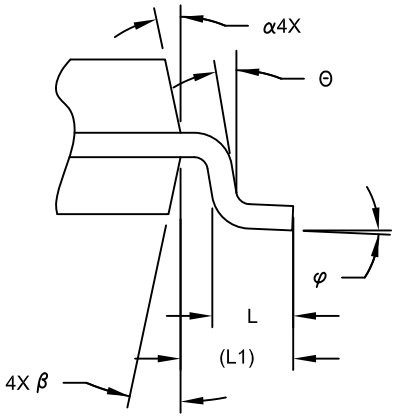

**VIEW A-A**

Microchip Technology Drawing C04-051C Sheet 1 of 2

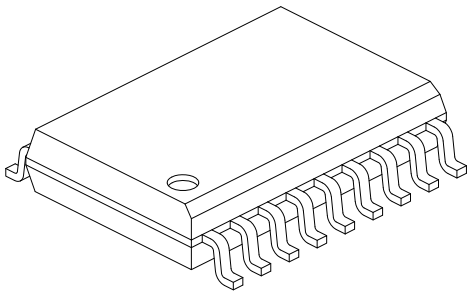

© 2002-2015 Microchip Technology Inc.

## 18-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




VIEW C



| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 18 | |
| Pitch | e | | 1.27 BSC | |
| Overall Height | A | - | - | 2.65 |
| Molded Package Thickness | A2 | 2.05 | - | - |
| Standoff § | A1 | 0.10 | - | 0.30 |
| Overall Width | E | | 10.30 BSC | |
| Molded Package Width | E1 | | 7.50 BSC | |
| Overall Length | D | | 11.55 BSC | |
| Chamfer (Optional) | h | 0.25 | - | 0.75 |
| Foot Length | L | 0.40 | - | 1.27 |
| Footprint | L1 | | 1.40 REF | |
| Lead Angle | Θ | 0° | - | - |
| Foot Angle | φ | 0° | - | 8° |
| Lead Thickness | c | 0.20 | - | 0.33 |
| Lead Width | b | 0.31 | - | 0.51 |
| Mold Draft Angle Top | α | 5° | - | 15° |
| Mold Draft Angle Bottom | β | 5° | - | 15° |

**Notes:**

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic
3. Dimension D does not include mold flash, protrusions or gate burrs, which shall not exceed 0.15 mm per end. Dimension E1 does not include interlead flash or protrusion, which shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M

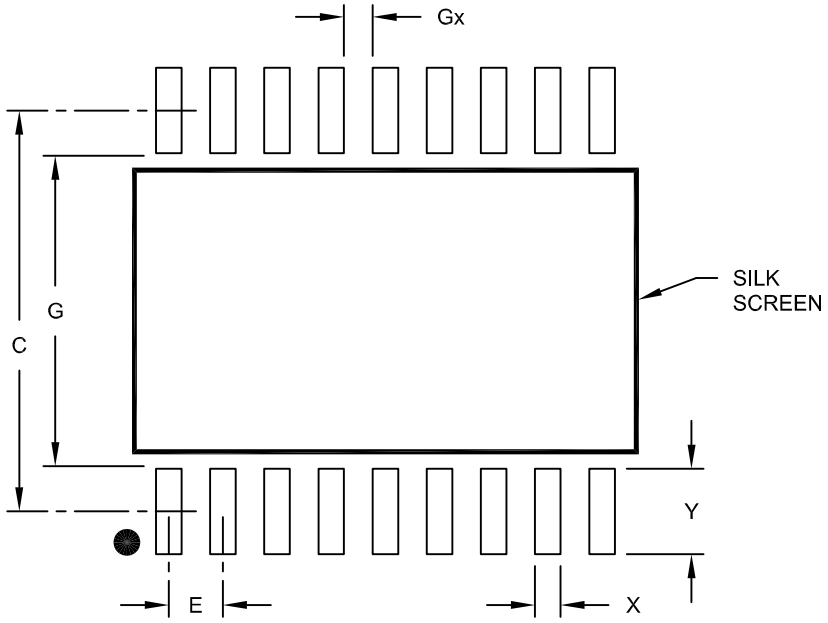
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.
5. Datums A & B to be determined at Datum H.

Microchip Technology Drawing No. C04-051C Sheet 2 of 2

18-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 1.27 BSC | |
| Contact Pad Spacing | C | | 9.40 | |
| Contact Pad Width | X | | | 0.60 |
| Contact Pad Length | Y | | | 2.00 |
| Distance Between Pads | Gx | 0.67 | | |
| Distance Between Pads | G | 7.40 | | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

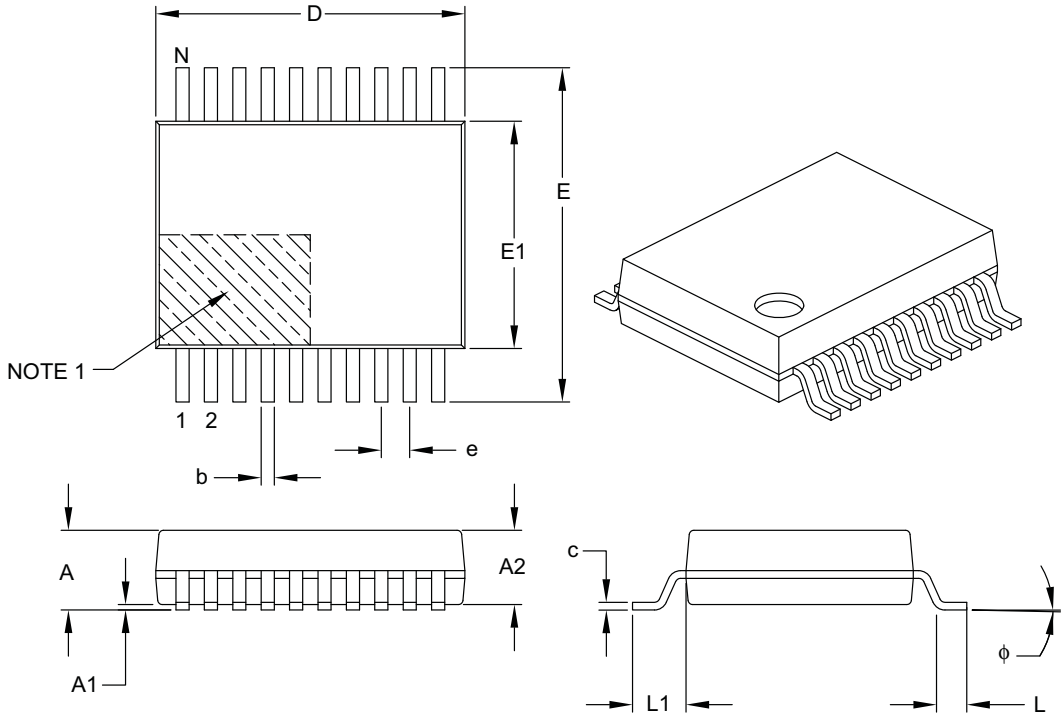
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2051A


© 2002-2015 Microchip Technology Inc.

## 20-Lead Plastic Shrink Small Outline (SS) – 5.30 mm Body [SSOP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 20 | |
| Pitch | e | | 0.65 BSC | |
| Overall Height | A | – | – | 2.00 |
| Molded Package Thickness | A2 | 1.65 | 1.75 | 1.85 |
| Standoff | A1 | 0.05 | – | – |
| Overall Width | E | 7.40 | 7.80 | 8.20 |
| Molded Package Width | E1 | 5.00 | 5.30 | 5.60 |
| Overall Length | D | 6.90 | 7.20 | 7.50 |
| Foot Length | L | 0.55 | 0.75 | 0.95 |
| Footprint | L1 | | 1.25 REF | |
| Lead Thickness | c | 0.09 | – | 0.25 |
| Foot Angle | ϕ | 0° | 4° | 8° |
| Lead Width | b | 0.22 | – | 0.38 |

**Notes:**

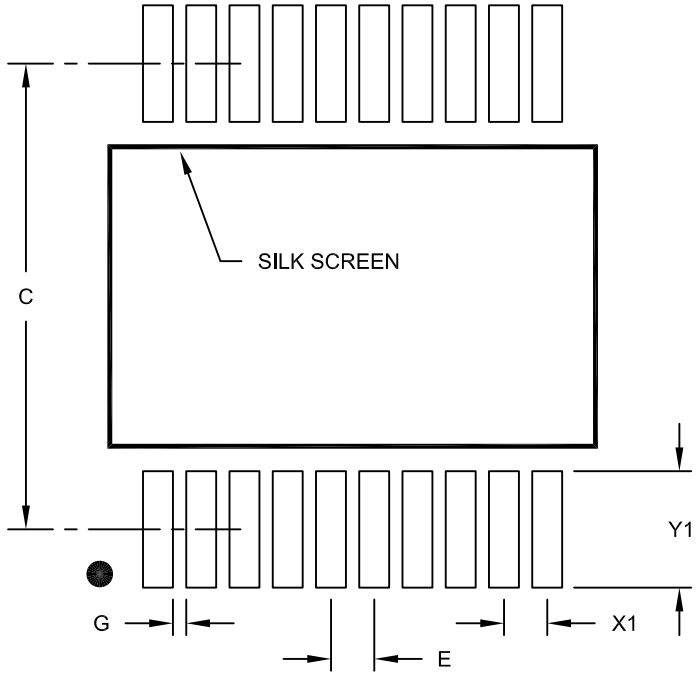
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.20 mm per side.
3. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-072B

20-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | 0.65 BSC | | |
| Contact Pad Spacing | C | | 7.20 | |
| Contact Pad Width (X20) | X1 | | | 0.45 |
| Contact Pad Length (X20) | Y1 | | | 1.75 |
| Distance Between Pads | G | 0.20 | | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

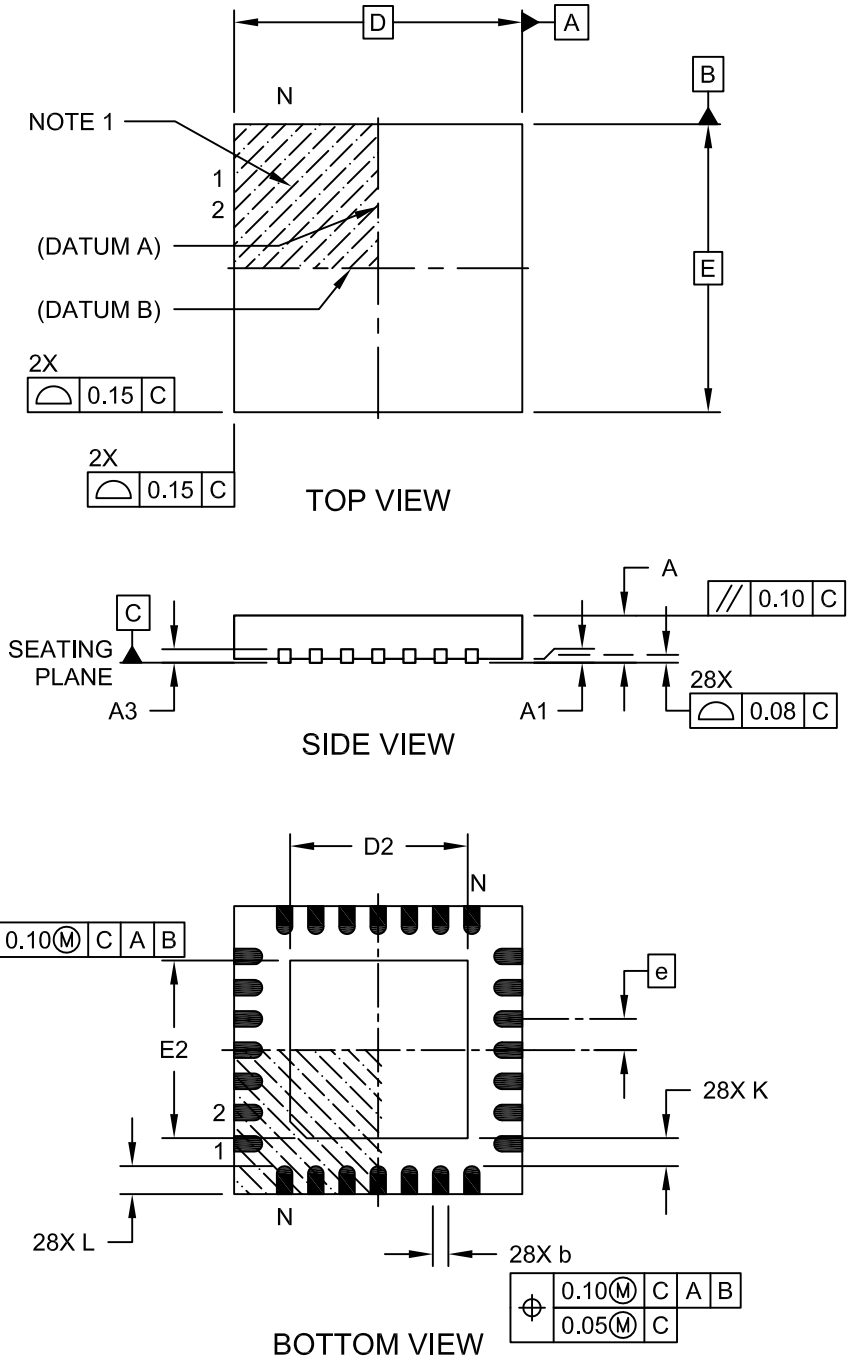
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2072A


© 2002-2015 Microchip Technology Inc.

**28-Lead Plastic Quad Flat, No Lead Package (ML) - 6x6 mm Body [QFN]**
**With 0.55 mm Terminal Length**

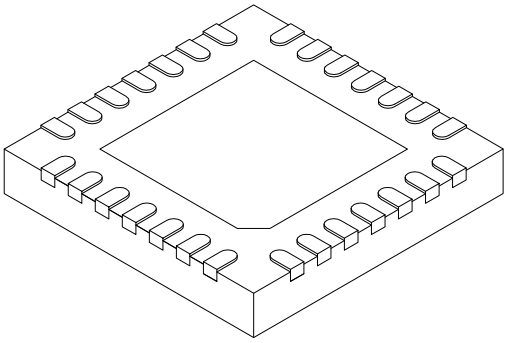
| **Note:** | For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging |
|---|---|




Microchip Technology Drawing C04-105C Sheet 1 of 2

## 28-Lead Plastic Quad Flat, No Lead Package (ML) - 6x6 mm Body [QFN] With 0.55 mm Terminal Length

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



| | Units | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension | Limits | MIN | NOM | MAX |
| Number of Pins | N | | 28 | |
| Pitch | e | | 0.65 BSC | |
| Overall Height | A | 0.80 | 0.90 | 1.00 |
| Standoff | A1 | 0.00 | 0.02 | 0.05 |
| Terminal Thickness | A3 | | 0.20 REF | |
| Overall Width | E | | 6.00 BSC | |
| Exposed Pad Width | E2 | 3.65 | 3.70 | 4.20 |
| Overall Length | D | | 6.00 BSC | |
| Exposed Pad Length | D2 | 3.65 | 3.70 | 4.20 |
| Terminal Width | b | 0.23 | 0.30 | 0.35 |
| Terminal Length | L | 0.50 | 0.55 | 0.70 |
| Terminal-to-Exposed Pad | K | 0.20 | - | - |

**Notes:**

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M.

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

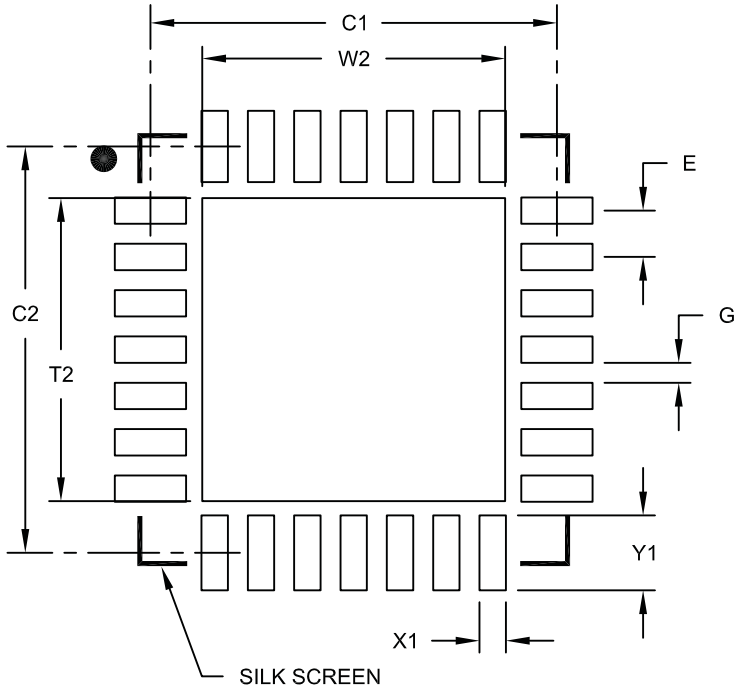
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-105C Sheet 2 of 2


© 2002-2015 Microchip Technology Inc.

### 28-Lead Plastic Quad Flat, No Lead Package (ML) – 6x6 mm Body [QFN] with 0.55 mm Contact Length

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 0.65 BSC | |
| Optional Center Pad Width | W2 | | | 4.25 |
| Optional Center Pad Length | T2 | | | 4.25 |
| Contact Pad Spacing | C1 | | 5.70 | |
| Contact Pad Spacing | C2 | | 5.70 | |
| Contact Pad Width (X28) | X1 | | | 0.37 |
| Contact Pad Length (X28) | Y1 | | | 1.00 |
| Distance Between Pads | G | 0.20 | | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2105A

## APPENDIX A: REVISION HISTORY

### Revision A (August 2002)

Original data sheet for PIC18F1220/1320 devices.

### Revision B (November 2002)

This revision includes significant changes to **Section 2.0**, **Section 3.0** and **Section 19.0**, as well as updates to the Electrical Specifications in **Section 22.0** and includes minor corrections to the data sheet text.

### Revision C (May 2004)

This revision includes updates to the Electrical Specifications in **Section 22.0**, the DC and AC Characteristics Graphs and Tables in **Section 23.0** and includes minor corrections to the data sheet text.

### Revision D (October 2006)

This revision includes updates to the packaging diagrams.

### Revision E (January 2007)

This revision includes updates to the packaging diagrams.

### Revision F (February 2007)

This revision includes updates to the packaging diagrams.

### Revision G (April 2015)

Added Section 22.5: High Temperature Operation in the Electrical Specifications section.

## APPENDIX B: DEVICE DIFFERENCES

The differences between the devices listed in this data sheet are shown in Table B-1.

**TABLE B-1: DEVICE DIFFERENCES**

| Features | PIC18F1220 | PIC18F1320 |
|---|---|---|
| Program Memory (Bytes) | 4096 | 8192 |
| Program Memory (Instructions) | 2048 | 4096 |
| Interrupt Sources | 15 | 15 |
| I/O Ports | Ports A, B | Ports A, B |
| Enhanced Capture/Compare/PWM Modules | 1 | 1 |
| 10-bit Analog-to-Digital Module | 7 input channels | 7 input channels |
| Packages | 18-pin SDIP<br>18-pin SOIC<br>20-pin SSOP<br>28-pin QFN | 18-pin SDIP<br>18-pin SOIC<br>20-pin SSOP<br>28-pin QFN |


© 2002-2015 Microchip Technology Inc.

## APPENDIX C: CONVERSION CONSIDERATIONS

This appendix discusses the considerations for converting from previous versions of a device to the ones listed in this data sheet. Typically, these changes are due to the differences in the process technology used. An example of this type of conversion is from a PIC16C74A to a PIC16C74B.

**Not Applicable**

## APPENDIX D: MIGRATION FROM BASELINE TO ENHANCED DEVICES

This section discusses how to migrate from a baseline device (i.e., PIC16C5X) to an enhanced MCU device (i.e., PIC18FXXX).

The following are the list of modifications over the PIC16C5X microcontroller family:

**Not Currently Available**

## APPENDIX E: MIGRATION FROM MID-RANGE TO ENHANCED DEVICES

A detailed discussion of the differences between the mid-range MCU devices (i.e., PIC16CXXX) and the enhanced devices (i.e., PIC18FXXX) is provided in *AN716, "Migrating Designs from PIC16C74A/74B to PIC18C442"*. The changes discussed, while device specific, are generally applicable to all mid-range to enhanced device migrations.

This Application Note is available as Literature Number DS00716.

## APPENDIX F: MIGRATION FROM HIGH-END TO ENHANCED DEVICES

A detailed discussion of the migration pathway and differences between the high-end MCU devices (i.e., PIC17CXXX) and the enhanced devices (i.e., PIC18FXXX) is provided in *AN726, "PIC17CXXX to PIC18CXXX Migration"*.

This Application Note is available as Literature Number DS00726.


© 2002-2015 Microchip Technology Inc.

## THE MICROCHIP WEB SITE

Microchip provides online support via our web site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

## CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

## CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

**Technical support is available through the web site at: http://www.microchip.com/support**

## PIC18F1220/1320 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. | – | X | /XX | XXX |
|---|---|---|---|---|
| Device | | Temperature Range | Package | Pattern |

| | |
|---|---|
| **Device:** | PIC18F1220/1320(1), PIC18F1220/1320T(2); VDD range 4.2V to 5.5V<br><br>PIC18LF1220/1320(1), PIC18LF1220/1320T(2); VDD range 2.5V to 5.5V |
| **Temperature Range:** | I = -40°C to +85°C (Industrial)<br>E = -40°C to +125°C (Extended) |
| **Package:** | SO = SOIC SS = SSOP<br>P = PDIP ML = QFN |
| **Pattern:** | QTP, SQTP, Code or Special Requirements (blank otherwise) |

**Examples:**

a) PIC18LF1320-I/P 301 = Industrial temp., PDIP package, Extended VDD limits, QTP pattern #301.

b) PIC18LF1220-I/SO = Industrial temp., SOIC package, Extended VDD limits.

**Note 1:** F = Standard Voltage range
LF = Wide Voltage Range

**2:** T = in tape and reel – SOIC package only


© 2002-2015 Microchip Technology Inc.

**Note the following details of the code protection feature on Microchip devices:**

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE**.** Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

**Trademarks**

The Microchip name and logo, the Microchip logo, dsPIC, FlashFlex, flexPWR, JukeBlox, KEELOQ, KEELOQ logo, Kleer, LANCheck, MediaLB, MOST, MOST logo, MPLAB, OptoLyzer, PIC, PICSTART, PIC$^{32}$ logo, RightTouch, SpyNIC, SST, SST Logo, SuperFlash and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

The Embedded Control Solutions Company and mTouch are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, BodyCom, chipKIT, chipKIT logo, CodeGuard, dsPICDEM, dsPICDEM.net, ECAN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, KleerNet, KleerNet logo, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, RightTouch logo, REAL ICE, SQI, Serial Quad I/O, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademarks of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2002-2015, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-63277-236-7

**QUALITY MANAGEMENT SYSTEM**
**CERTIFIED BY DNV**
**═ ISO/TS 16949 ═**

*Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.*

# Worldwide Sales and Service

## AMERICAS

**Corporate Office**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

**Atlanta**
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

**Austin, TX**
Tel: 512-257-3370

**Boston**
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

**Chicago**
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

**Cleveland**
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

**Dallas**
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

**Detroit**
Novi, MI
Tel: 248-848-4000

**Houston, TX**
Tel: 281-894-5983
Indianapolis

**Noblesville, IN**
Tel: 317-773-8323
Fax: 317-773-5453

**Los Angeles**
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

**New York, NY**
Tel: 631-435-6000

**San Jose, CA**
Tel: 408-735-9110

**Canada - Toronto**
Tel: 905-673-0699
Fax: 905-673-6509

## ASIA/PACIFIC

**Asia Pacific Office**
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon

**Hong Kong**
Tel: 852-2943-5100
Fax: 852-2401-3431

**Australia - Sydney**
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

**China - Beijing**
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

**China - Chengdu**
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

**China - Chongqing**
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

**China - Dongguan**
Tel: 86-769-8702-9880

**China - Hangzhou**
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

**China - Hong Kong SAR**
Tel: 852-2943-5100
Fax: 852-2401-3431

**China - Nanjing**
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

**China - Qingdao**
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

**China - Shanghai**
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

**China - Shenyang**
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

**China - Shenzhen**
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

**China - Wuhan**
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

**China - Xian**
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

## ASIA/PACIFIC

**China - Xiamen**
Tel: 86-592-2388138
Fax: 86-592-2388130

**China - Zhuhai**
Tel: 86-756-3210040
Fax: 86-756-3210049

**India - Bangalore**
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

**India - New Delhi**
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

**India - Pune**
Tel: 91-20-3019-1500

**Japan - Osaka**
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

**Japan - Tokyo**
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

**Korea - Daegu**
Tel: 82-53-744-4301
Fax: 82-53-744-4302

**Korea - Seoul**
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

**Malaysia - Kuala Lumpur**
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

**Malaysia - Penang**
Tel: 60-4-227-8870
Fax: 60-4-227-4068

**Philippines - Manila**
Tel: 63-2-634-9065
Fax: 63-2-634-9069

**Singapore**
Tel: 65-6334-8870
Fax: 65-6334-8850

**Taiwan - Hsin Chu**
Tel: 886-3-5778-366
Fax: 886-3-5770-955

**Taiwan - Kaohsiung**
Tel: 886-7-213-7828

**Taiwan - Taipei**
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

**Thailand - Bangkok**
Tel: 66-2-694-1351
Fax: 66-2-694-1350

## EUROPE

**Austria - Wels**
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

**Denmark - Copenhagen**
Tel: 45-4450-2828
Fax: 45-4485-2829

**France - Paris**
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

**Germany - Dusseldorf**
Tel: 49-2129-3766400

**Germany - Munich**
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

**Germany - Pforzheim**
Tel: 49-7231-424750

**Italy - Milan**
Tel: 39-0331-742611
Fax: 39-0331-466781

**Italy - Venice**
Tel: 39-049-7625286

**Netherlands - Drunen**
Tel: 31-416-690399
Fax: 31-416-690340

**Poland - Warsaw**
Tel: 48-22-3325737

**Spain - Madrid**
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

**Sweden - Stockholm**
Tel: 46-8-5090-4654

**UK - Wokingham**
Tel: 44-118-921-5800
Fax: 44-118-921-5820

01/27/15


© 2002-2015 Microchip Technology Inc.

## Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

- View PIC18F1320T-I/SO on WIN SOURCE
- Microchip Technology Information

## Optimize Your Supply Chain with WIN SOURCE Solutions

- Global Sourcing Solution
- Obsolete Management
- Cost Control Management
- Shortage Management
- Alternative Solution
- Excess Inventory Management